Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Μηχανικών Πληροφορικής

Πτυχιακή εργασία

Τίτλος: Δημιουργία Διακομιστή διαδικτύου που θα υποστηρίζει εφαρμογές σε περιβάλλον Android

Γεώργιος Ζεάκης (ΑΜ: 3964)

Επιβλέπων καθηγητής : Παπαδάκης Νικόλαος

Επιτροπή Αξιολόγησης :

Ημερομηνία παρουσίασης:

Ευχαριστίες

Ευχαριστώ τη σύζυγο και τα παιδιά μου για την υπομονή τους. Επίσης, ευχαριστώ τον κ. Παπαδάκη Νίκο για την εμπιστοσύνη του.

Abstract

The purpose of this thesis was to study, design and develop an application on the Android operating system and an internet server to support this application. The application provides users with information on major events on the site and, hence, on their location. This information is posted by certified users of the application and may involve road accidents, pet loss, thefts, natural disasters, job offer or search, etc.

The location of the user's smart device is received from the positioning subsystem of the device itself. The web server provides the application with the appropriate processing and storage of user data and user information and events by location and region.

The system developed is designed to quickly disseminate information and instantly update communities through a handy communication channel, thus increasing the chances of solving problems, making the right decisions, meeting needs, etc. A key point of the system is that the user does not remain a passive receiver to changes and events that take part in a community, but he can also participate actively in the community in which he or she lives.

Περίληψη

Σκοπός της πτυχιακής εργασίας αυτής ήταν η μελέτη, σχεδίαση και ανάπτυξη μιας εφαρμογής στο λειτουργικό σύστημα Android και ενός διακομιστή διαδικτύου που θα υποστηρίζει την εφαρμογή αυτή. Η εφαρμογή παρέχει στους χρήστες πληροφορίες για σημαντικά συμβάντα στην τοποθεσία και κατ' επέκταση στην περιοχή στην οποία βρίσκονται. Οι πληροφορίες αυτές αναρτούνται από πιστοποιημένους χρήστες της εφαρμογής και μπορεί να αφορούν τροχαία ατυχήματα, απώλεια κατοικίδιου ζώου, κλοπές, φυσικές καταστροφές, προσφορά/αναζήτηση εργασίας/υπηρεσίας κα.

Η θέση της έξυπνης συσκευής του χρήστη λαμβάνετε από το υποσύστημα εντοπισμού θέσης της ίδιας της συσκευής. Ο διακομιστής διαδικτύου παρέχει στην εφαρμογή την κατάλληλη επεξεργασία και αποθήκευση των δεδομένων των χρηστών και τις πληροφορίες-συμβάντα που αναρτούν οι χρήστες, ανά τοποθεσία και περιοχή.

Το σύστημα που αναπτύχθηκε αποσκοπεί στη γρήγορη διάδοση πληροφοριών και την άμεση ενημέρωση κοινοτήτων μέσω ενός εύχρηστου διαύλου επικοινωνίας, αυξάνοντας έτσι τις πιθανότητες επίλυσης προβλημάτων, λήψης σωστών αποφάσεων, κάλυψης αναγκών κα. Ένα βασικό σημείο του συστήματος είναι ότι ο χρήστης δεν μένει παθητικός δέκτης στις αλλαγές και τα γεγονότα που λαμβάνουν μέρος σε μια κοινότητα, αντιθέτως μπορεί και ο ίδιος να συμμετάσχει ενεργά στην κοινότητα στην οποία βρίσκεται.

Πίνακας Περιεχομένων

Πίνακας Εικόνων

1 Εισαγωγή

Επείγοντα περιστατικά και φυσικές καταστροφές συμβαίνουν καθημερινά στην ζωή μας, συμβάντα τα οποία θέτουν σε κίνδυνο ολόκληρες πόλεις, περιοχές και ανθρώπους. Οι πολίτες πρέπει να είναι προετοιμασμένοι για αυτά τα περιστατικά και να μπορούν να ανταπεξέλθουν άμεσα και σωστά, θα πρέπει να μπορούν να ενημερώνουν και να ενημερώνονται χρησιμοποιώντας ένα κοινό και εύκολο στη χρήση κανάλι επικοινωνίας. Σε μια κρίσιμη κατάσταση η πληροφορίες που την αφορούν πρέπει να μπορούν να διαδοθούν όσο πιο γρήγορα γίνετε στους κατάλληλους ανθρώπους.

1.1 Κίνητρο για την διεξαγωγή της Εργασίας

Σήμερα, περισσότερο από ποτέ, η επικοινωνία καθορίζει τις ζωές μας. Η άνοδος των κοινωνικών δικτύων ανέδειξε την ανάγκη του ανθρώπου για κοινωνικές επαφές αλλά και την επιτακτική πλέον ανάγκη για ενημέρωση συμβάντων και γεγονότων τα οποία με τον ένα ή τον άλλο τρόπο επηρεάζουν την καθημερινότητά μας, έμμεσα ή άμεσα. Παρ' όλα αυτά όμως η χρήση εργαλείων και υπηρεσιών για την άμεση και στοχευόμενη ενημέρωση του χρήστη με βάση την τοποθεσία του είναι περιορισμένη.

Η παροχή υπηρεσιών βασιζόμενες στην τοποθεσία (Location-based service, LBS)(1) είναι υπηρεσίες που βασίζονται στην γεωγραφική θέση της συσκευής που χρησιμοποιεί ο χρήστης με σκοπό, μεταξύ άλλων, την παροχή πληροφοριών σχετικών με την θέση αυτή. Οι υπηρεσίες αυτές μπορούν να χρησιμοποιούνται στα κοινωνικά δίκτυα, στις υπηρεσίες ασφάλειας, για διασκέδαση, για επικοινωνία, στην εργασία, κα.

Υπάρχει λοιπόν η ανάγκη για στοχευόμενη ενημέρωση για σημαντικά και κρίσιμα γεγονότα, ανάγκη για ένα κοινό κανάλι επικοινωνίας μεταξύ των μελών μια κοινότητας, μιας γειτονιάς ή ακόμα περισσότερο μεταξύ των πολιτών μιας πόλης ή μιας χώρας. Το κοινό κανάλι επικοινωνίας μας επιτρέπει να συντονιζόμαστε και να συγχρονιζόμαστε, επιτρέπει τον συντονισμό και τον συγχρονισμό των μελών μια κοινότητας με τέτοιο τρόπο ώστε να λειτουργεί και να αντιμετωπίζει ένα πρόβλημα ή μια κατάσταση ως μια συλλογική οντότητα. Έτσι αυξάνονται οι δυνατότητες επιλογής και λήψης σωστών αποφάσεων, εξοικονομείται ανθρώπινος κόπος και ενέργεια.

1.2 Σκοπός και Στόχοι Εργασίας

Στόχος της Εργασίας αυτής είναι η δημιουργία μιας εφαρμογής για κινητά έξυπνα τηλέφωνα και της αντίστοιχης υποδομής για την υποστήριξή της, με σκοπό την παροχή υπηρεσιών πληροφόρησης χρησιμοποιώντας αναρτήσεις των ίδιων των χρηστών της εφαρμογής. Η εφαρμογή αυτή δημιουργεί ένα πλαίσιο-δίαυλο επικοινωνίας, όπου τα άτομα έχουν τη δυνατότητα να αλληλεπιδρούν, να συνεργάζονται και να ανταλλάσσουν σημαντικές πληροφορίες που εξαρτώνται από την τοποθεσία του χρήστη. Οποιοσδήποτε μπορεί να φέρει σε επαφή άτομα (μέσω μιας ανάρτησης) τα οποία έχουν κάτι κοινό ή να θέλουν να συνεισφέρουν εθελοντικά και όλοι μαζί μπορούν να αλλήλο-επωφεληθούν. Ο καθένας μπορεί να συμμετέχει ή να προσφέρει στον συνάνθρωπό του, στην γειτονιά του ή/και στην πόλη του.

Επίσης, σε καταστάσεις διαχείρισης κρίσεων, η πρόσβαση σε πληροφορίες **γεωγραφικά εξαρτώμενες και χρονικά έγκυρες** είναι καθοριστικός παράγοντας. Οι χρήστες που βρίσκονται στις καταστάσεις αυτές μπορούν να αναρτούν σχετικές πληροφορίες και η εφαρμογή αναλαμβάνει να ενημερώσει τους υπόλοιπους χρήστες που βρίσκονται στην περιοχή, σε συγκεκριμένη ακτίνα που εξαρτάται από την κατηγορία και το είδος της κατάστασης.

1.3 Παραδείγματα χρήσης της εφαρμογής XAlerts

Υπάρχουν διάφορες περιπτώσεις χρήσεων για την εφαρμογή μας που έχει όνομα XAlerts, όπως ιατρική επείγουσα ανάγκη, μαζική επικοινωνία, αίτηση παροχής υπηρεσιών, προσωπική ασφάλεια, αποτυχία υλικού / διεργασίας, ενημέρωση για φυσικές καταστροφές, κατάσταση έκτακτης ανάγκης, κα.

Πιο αναλυτικά:

- μπορούμε να χρησιμοποιήσουμε το XAlerts για να ειδοποιήσουμε για μια κατάσταση έκτακτης ανάγκης κατά τη διάρκεια δημόσιας έκτακτης φροντίδας ή για ένα δίκτυο ιδιωτικής φροντίδας όπως ένας Οργανισμός ή μια οικογένεια.
- Φυσικές καταστροφές: Χρησιμοποιώντας τη δυνατότητα Geo-fencing (2), οι κοινότητες των πολιτών μπορούν να ειδοποιηθούν κατά τη διάρκεια φυσικών καταστροφών, όπως σεισμοί, τυφώνες, πλημμύρες, εστίες ασθενειών ή ανθρωπογενείς καταστροφές
- Αιτήσεις παροχής υπηρεσιών: Οι πλησιέστεροι πάροχοι υπηρεσιών / δημόσιων υπηρεσιών όπως ηλεκτρικό, πυροσβεστικό, αστυνομικό, ασθενοφόρο, νερό και αποχέτευση καθώς και οικιακοί παροχείς υπηρεσιών, όπως η επισκευή, μπορούν να ειδοποιηθούν για την συνδρομή τους
- Προσωπική ασφάλεια: Τα άτομα μπορούν να ειδοποιούν ΑΜΕΣΩΣ τους υπόλοιπους ανθρώπους στην περιοχή για την ακριβή τοποθεσία σας και για οποιαδήποτε κατάσταση κρίσης που μπορεί να βρίσκεστε.

1.4 Δομή Εργασίας

Στο κεφάλαιο 1 παρουσιάζεται μια εισαγωγική αναφορά για τα αίτια που μας παρακίνησαν για την επιλογή του θέματος της Εργασίας, αναφέρουμε τους σκοπούς, τους στόχους και τα οφέλη της λύσης που πραγματευόμαστε. Στο κεφάλαιο 2 αναλύουμε το προτεινόμενο σύστημα και περιγράφουμε την αρχιτεκτονική του. Στο κεφάλαιο 3 παρουσιάζουμε συνοπτικά τα εργαλεία, τις βιβλιοθήκες και τις τεχνικές που χρησιμοποιήσαμε. Στο κεφάλαιο 3 δείχνουμε την υλοποίηση όλων των επιμέρους υποσυστημάτων της πλατφόρμας, σε μορφή εγχειριδίου με εικόνες και επεξηγηματικά σχόλια. Τέλος, στο κεφάλαιο 4 διατυπώνουμε τα συμπεράσματά μας και προτείνουμε μελλοντικές δυνατότητες και επεκτάσεις για το σύστημά μας.

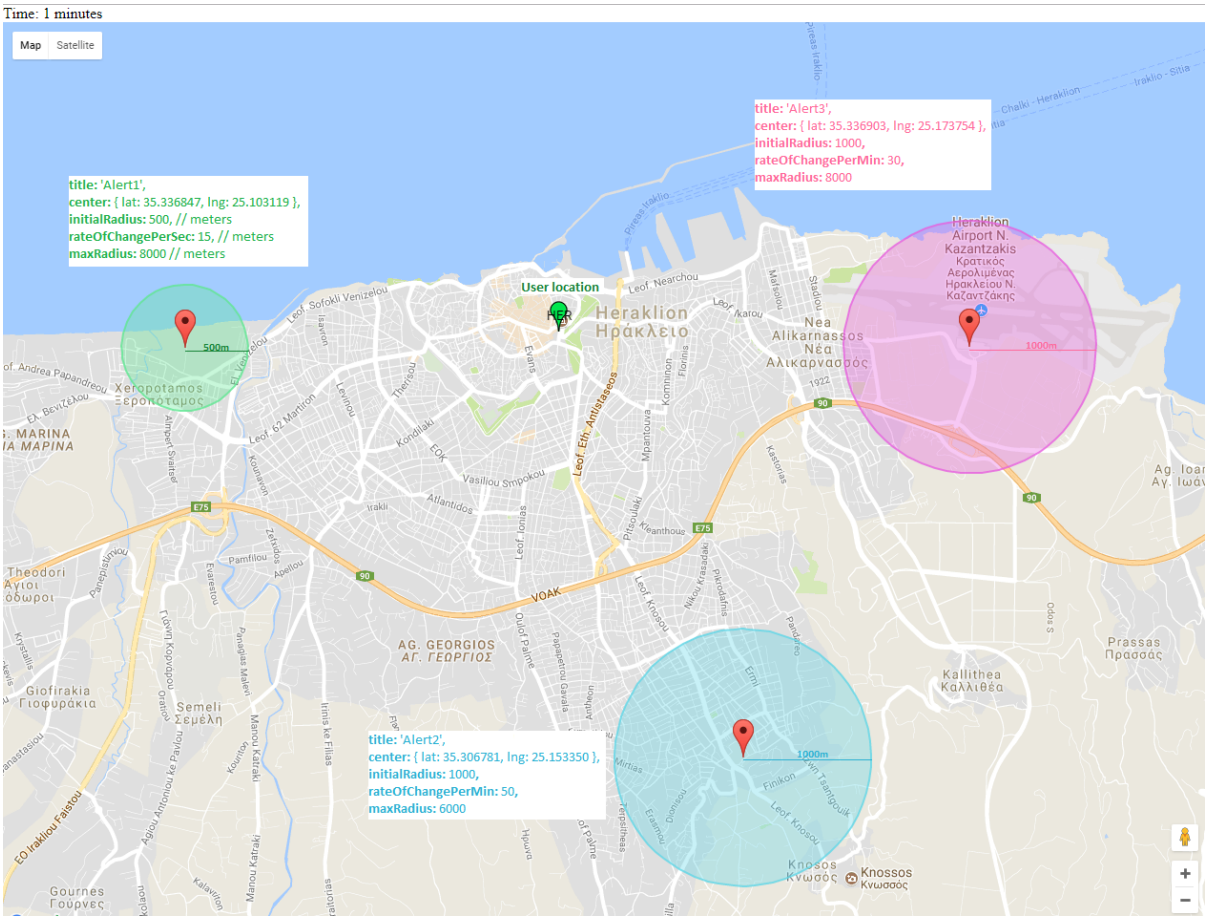
2 Ανάλυση του συστήματος XAlerts

Το κεφάλαιο αυτό ασχολείται με το κύριος μέρος της πτυχιακής δηλαδή με την ανάλυση των απαιτήσεων, την σχεδίαση της λύσης και την αρχιτεκτονική του συστήματος. Πριν προχωρήσουμε όμως στην ανάλυση πρέπει να ορίσουμε τι είναι το σύστημα γενικότερα.

2.1 Ορισμός του συστήματος

Το XAlerts είναι μια πλατφόρμα αποστολής μηνυμάτων που βασίζεται σε κινητές εφαρμογές και είναι κατασκευασμένη για την δημιουργία και ανεύρεση ειδοποιήσεων ή αιτημάτων παροχής πληροφοριών και υπηρεσιών εντός ορισμένης γεωγραφικής περιοχής με χρονικά εξαρτώμενη εμβέλεια.

Όπως μπορούμε να δούμε και στην Εικόνα 1 κάθε Ενημέρωση (Alert) που δημιουργείται δεν γνωστοποιείται σε όλους τους χρήστες της εφαρμογής αλλά διανέμεται μόνο σε χρήστες που βρίσκονται εντός συγκεκριμένης γεωγραφικής θέσης. Έτσι στο παράδειγμά μας, ένας χρήστης που βρίσκεται στην θέση που ορίζεται από το σημάδι (Marker) με πράσινο χρώμα στον χάρτη (Εικόνα 1) δεν θα λάβει τις τρεις Ενημερώσεις (Alerts) που φαίνονται με τα σημάδια κόκκινου χρώματος.



Εικόνα 1: Χρονικά εξαρτώμενη εμβέλεια Ενημερώσεων

Η κάθε ενημέρωση όμως έχει δυναμική εμβέλεια η οποία εξαρτάται από μια αρχική και μια τελική τιμή αλλά και από ένα ρυθμό μεταβολής που ορίζει την εμβέλεια ανάλογα με τα πόσα λεπτά της ώρας έχουν περάσει από την αρχική ημερομηνία και ώρα δημιουργίας της Ενημέρωσης. Έτσι για παράδειγμα, για την Ενημέρωση με τίτλο "Alert3" (με ροζ χρώμα) βλέπουμε ότι αρχικά δημιουργήθηκε στο γεωγραφικό σημείο με συντεταγμένες που ορίζονται στο πεδίο "center", έχει αρχική εμβέλεια 1000 μέτρα (πεδίο "initialRadius"), άρα όσοι άλλοι χρήστες της εφαρμογής μας βρίσκονται μέσα σε αυτήν θα ενημερωθούν άμεσα κατά την δημιουργία της Ενημέρωσης αυτής, έχει ως ρυθμό μεταβολής της εμβέλειας τα 30 μέτρα (πεδίο "rateOfChangePerMin") και τέλος έχουμε ένα όριο της εμβέλειας που ορίζεται στο πεδίο "maxRadius". Άρα αν ο χρήστης παραμείνει στο σημείο που βρίσκεται για χρονικό διάστημα πχ. 3 ωρών τότε θα λάβει την ενημέρωση στην συσκευή του αφού η εμβέλεια του "Alert3" θα έχει μεγαλώσει κατά 180λεπτά επί 30μέτρα ανά λεπτό = 5400μέτρα + 1000 μέτρα η αρχική ακτίνα = 6400μέτρα. Βλέπουμε ότι τα 6400 < 8000μέτρα που είναι μέγιστο άρα βρισκόμαστε μέσα στα επιτρεπτά όρια.

2.2 Υπολογισμός αποστάσεων εμβέλειας ενημερώσεων

Δοθείσας τοποθεσίας (γεωγραφικές συντεταγμένες), ο τρόπος που υπολογίζει η εφαρμογή ποιοι χρήστες θα λάβουν ποιες ενημερώσεις και πότε, είναι κάνοντας χρήση του τύπου Haversine (Haversine formula) (3). Ο τύπος Haversine (Εικόνα 2) υπολογίζει το τόξο μεγίστου κύκλου μεταξύ δύο σημείων επιφάνειας μιας σφαίρας, δηλαδή την συντομότερη απόσταση μεταξύ των δύο σημείων αυτών (4).

$$\operatorname{hav}\left(\frac{d}{r}\right) = \operatorname{hav}(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)\operatorname{hav}(\lambda_2 - \lambda_1)$$

Εικόνα 2: Haversine formula (3)

Στον παραπάνω τύπο έχουμε όπου d η απόσταση μεταξύ των δύο σημείων, r είναι η ακτίνα του κύκλου, φ1 και φ2 το γεωγραφικό πλάτος του πρώτου και του δεύτερου σημείου αντίστοιχα (σε ακτίνια) και όπου λ1 & λ2 γεωγραφικό μήκος του πρώτου και του δεύτερου σημείου αντίστοιχα (σε ακτίνια). Όσον αφορά την συνάρτηση hav(θ) αυτή είναι ίση με το $\sin^2(\theta/2) = (1 - \cos(\theta))/2$. Έτσι λύνοντας ως προς την απόσταση d, έχουμε $d=r*hav^{-1}(h) = 2*r*\arcsin(\sqrt{h})$ όπου h=hav(d/r). Επομένως καταλήγουμε στην παρακάτω εξίσωση:

$$d = 2r\arcsin\left(\sqrt{\operatorname{hav}(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)\operatorname{hav}(\lambda_2 - \lambda_1)}\right)$$
$$= 2r\arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

Εικόνα 3: Τύπος εύρεσης συντομότερης απόστασης σε επιφάνεια σφαίρας (3)

Άρα αφού μπορούμε να υπολογίσουμε την απόσταση μεταξύ του χρήστη και όλων των Ενημερώσεων που έχουμε καταχωρήσει στο σύστημά μας, αυτό που μένει είναι να βρούμε και να εμφανίσουμε στον χρήστη όλες αυτές τις Ενημερώσεις που περιλαμβάνουν την γεωγραφική θέση του χρήστη μέσα στην τρέχουσα εμβέλειά τους. Επειδή η διαδικασία αυτή είναι χρονοβόρα στον υπολογισμό της (αφού οι Ενημερώσεις μπορεί να είναι χιλιάδες) έχουμε αναθέσει στην Βάση δεδομένων τους υπολογισμούς αυτούς υλοποιώντας τον παραπάνω τύπο σε SQL όπως θα δούμε αναλυτικά στο κεφάλαιο 3.

2.3 Ανάλυση Απαιτήσεων της εφαρμογής

Οι απαιτήσεις των επιμέρους συστημάτων είναι οι ακόλουθες:

- Εφαρμογή χρήστη
  - Λειτουργία σε περιβάλλον Android
  - Πιστοποίηση ταυτότητας χρήστη
  - Δημιουργία, επεξεργασία, διαγραφή Ενημέρωσης
  - Δημιουργία, επεξεργασία, διαγραφή σχολίου Ενημέρωσης
  - Προσάρτηση εικόνας σε Ενημέρωση
  - Προσάρτηση εικόνας σε σχόλιο Ενημέρωσης
  - Παρουσίαση τοπικών ενημερώσεων σε διαδραστική λίστας
  - Παρουσίαση τοπικών ενημερώσεων σε διαδραστικό χάρτη
  - Παρουσίαση σχολίων για κάθε Ενημέρωση μαζί με τις εικόνες τους
  - Φιλτράρισμα Ενημερώσεων ανάλογα με την κατηγορία που ανήκουν

- Διακομιστής
  - Αποθήκευση Ενημερώσεων, σχολίων, κατηγοριών και εικόνων κάθε χρήστη σε σχεσιακή βάση δεδομένων
  - Παροχή επικοινωνίας μέσω Restful API τεχνικής (5)
  - Παροχή Ενημερώσεων σε συγκεκριμένες περιοχές και κατηγορίες

- Εφαρμογή Διαχειριστή
  - Διεπαφή Διαδικτύου
  - Δυναμική διεπαφή χρήστη (Responsive Web design) (6)
  - Διαχείριση περιεχομένου (Ενημερώσεις, Σχόλια, Κατηγορίες) αλλά και των χρηστών

2.3.1 Μοντέλα της εφαρμογής

Για την δημιουργία ενός διαύλου επικοινωνίας αναπτύξαμε μια εφαρμογή για το λειτουργικό σύστημα Android η οποία δίνει την δυνατότητα σε πιστοποιημένους χρήστες να δημιουργούν και να διαχειρίζονται Ενημερώσεις. Επίσης όπως θα δούμε παρακάτω, ο διακομιστής υποστηρίζει την εφαρμογή παρέχοντας μια διεπαφή επικοινωνίας και αποθήκευσης των δεδομένων επικοινωνώντας με την σειρά του με σχεσιακή βάση δεδομένων.

Έτσι καταλήξαμε σε τέσσερα κύριες οντότητες με τις οποίες θα εργαστούμε:

1. Μοντέλο Ενημέρωσης (Alert model)
2. Μοντέλο Σχολίου (Alert Comment model)
3. Μοντέλο Κατηγορίας Ενημέρωσης (Alert Category model)
4. Μοντέλο Χρήστη (User model)

Το Μοντέλο Ενημέρωσης (Alert model) περιλαμβάνει πληροφορίες όπως:

- Ο δημιουργός της Ενημέρωσης που είναι στην ουσία ένας εξουσιοδοτημένος χρήστης
- Η τοποθεσία δημιουργίας της Ενημέρωσης που λαμβάνεται από το σύστημα προσδιορισμού θέσης της συσκευής του χρήστη
- Κατηγορία: Κάθε Ενημέρωση ανήκει σε μια κατηγορία ανάλογα με κάποια κριτήρια
- Ημερομηνία δημιουργίας
- Τίτλος
- Περιγραφή
- Εικόνα, κάθε Ενημέρωση μπορεί να περιλαμβάνει μια εικόνα
- Σχόλια, κάθε Ενημέρωση μπορεί να περιλαμβάνει ένα ή περισσότερα σχόλια

Το Μοντέλο Σχολίου (Alert Comment model) περιλαμβάνει πληροφορίες όπως:

- Ο δημιουργός του Σχολίου
- Η Ενημέρωση την οποία αφορά
- Ημερομηνία δημιουργίας
- Τίτλος
- Περιγραφή
- Εικόνα

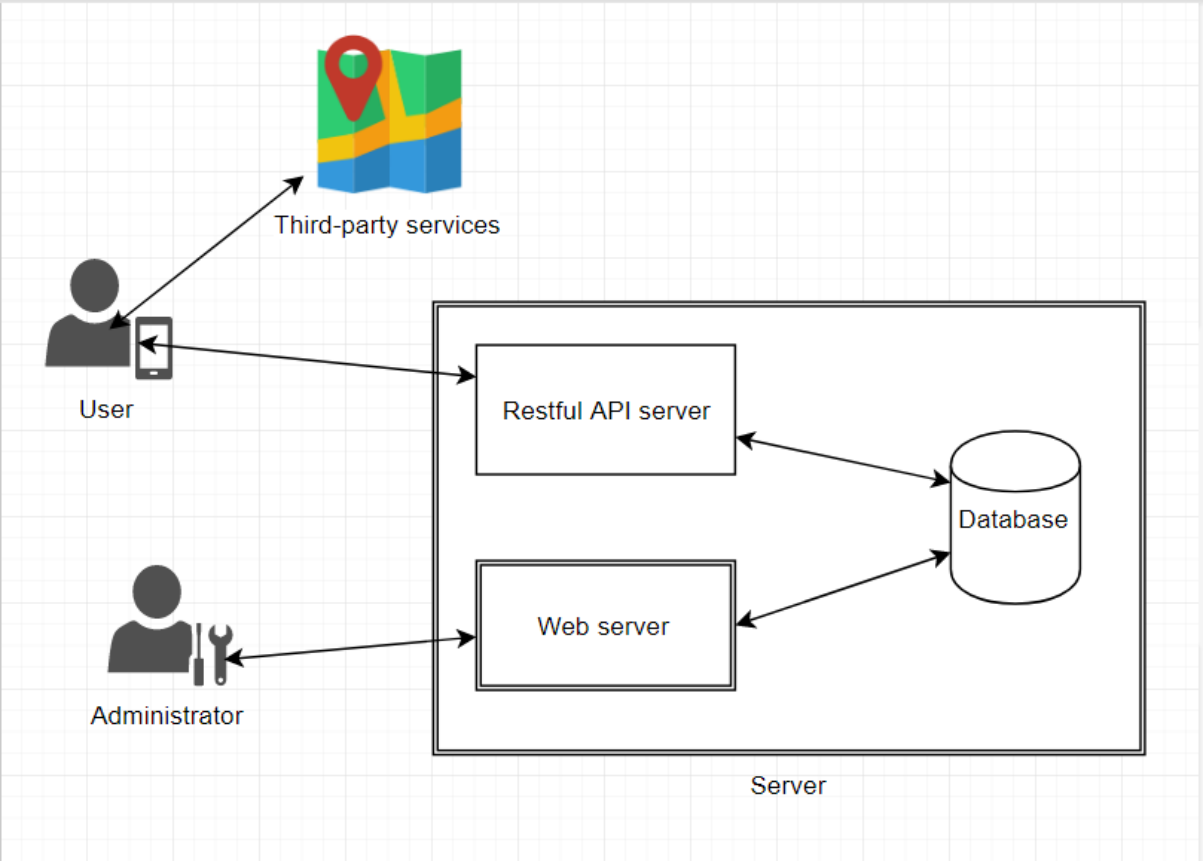
Το Μοντέλο Κατηγορίας Ενημέρωσης (Alert Category model) περιλαμβάνει πληροφορίες όπως:

- Όνομα Κατηγορίας
- Αρχική ακτίνα Ενημέρωσης (Initial Radius)
- Τελική (μέγιστη) ακτίνα Ενημέρωσης (Max Radius)
- Ρυθμός Μεταβολής ακτίνας Ενημέρωσης (Rate Of Change Per Minute)

Το Μοντέλο Χρήστη (User model) περιλαμβάνει πληροφορίες όπως:

- Ηλεκτρονική διεύθυνση (Email address)
- Κωδικός πρόσβασης, ουσιαστικά αποθηκεύουμε μόνο την έξοδο (σύνοψη) από μια κρυπτογραφική συνάρτηση κατακερματισμού.
- Κατάσταση λογαριασμού (Status)
- Όνομα και Επώνυμο
- Κλειδί κρυπτογράφησης επικοινωνίας, είναι μοναδικό για κάθε χρήστη πετυχαίνοντας έτσι μεγαλύτερη ασφάλεια για το σύστημά μας

2.4 Αρχιτεκτονική του συστήματος



Εικόνα 4: Αρχιτεκτονική του συστήματος

Στην Εικόνα 4 βλέπουμε την αρχιτεκτονική του συστήματος όπου μπορούμε να διακρίνουμε τέσσερις κύριες οντότητες:

- Ο Χρήστης (User): ο τελικός χρήστης της εφαρμογής που χρησιμοποιεί έξυπνη συσκευή (Smartphone) με λειτουργικό σύστημα Android
- Ο Διαχειριστής (Administrator) ο οποίος διαχειρίζεται το σύστημα μέσω διεπαφής διαδικτύου
- ο Διακομιστής (Server) που περιλαμβάνει:
  - τον Restful API server
  - τον Web server για τις ανάγκες της διαχείρισης
  - και ο Database server για την αποθήκευση όλων των δεδομένων
- Υπηρεσίες τρίτων

Αρχικά η εφαρμογή του Χρήστη επικοινωνεί με τον Restful API server από όπου αντλεί όλα τα δεδομένα που χρειάζεται καθώς επίσης και τα απαραίτητα API Keys για την εξουσιοδότηση των υπηρεσιών τρίτων παρόχων, όπως το Google Maps.

Ο Restful API server είναι ένας ανεξάρτητος αυτόνομος διακομιστής διαδικτύου ο οποίος παρέχει όλη την απαραίτητη διεπαφή στα πρότυπα του Restful API micro service (5), αντλεί με την σειρά του τα ζητούμενα δεδομένα από την Βάση δεδομένων (Database). Περισσότερες λεπτομέρειες για την υλοποίηση του θα δούμε στο κεφάλαιο 3.

Ο Web server (7) λειτουργεί και αυτός ανεξάρτητα και αυτόνομα ώστε να είναι όσο το δυνατόν πιο ασφαλής. Στο παράδειγμά μας και στην υλοποίηση φιλοξενείται στον ίδιο διακομιστή με τους υπόλοιπους server για τις ανάγκες της Εργασίας. Παρ' όλα αυτά θα μπορούσε να βρίσκεται

οπουδήποτε αλλού, πχ. πίσω από ένα ασφαλές δίκτυο. Έχει υλοποιηθεί με τα ίδια ακριβώς εργαλεία που υλοποιήσαμε τον Restful API server αλλά επιτελεί διαφορετικό σκοπό που είναι η παροχή ενός ασφαλούς, απλού και εύκολου περιβάλλοντος διαχείρισης περιεχομένου και χρηστών για τις ανάγκες του συστήματος. Ο λογαριασμός του χρήστη Διαχειριστής (Administrator) είναι ανεξάρτητος από την υπόλοιπη εφαρμογή για να είναι πιο ασφαλής.

Τέλος η Βάση δεδομένων με την οποία επικοινωνούν τόσο ο Restful API server όσο και ο Web server. Η βάση διατηρεί τα δεδομένα, τις οντότητες και τις συσχετίσεις μεταξύ τους. Έχει υλοποιηθεί με την MySQL σχεσιακή βάση δεδομένων και όπως θα δούμε στο κεφάλαιο 3 το σχήμα της είναι αρκετά απλό και κατανοητό.

2.5 Εργαλεία, βιβλιοθήκες και τεχνικές

Για την πτυχιακή αυτή μελετήθηκαν υπάρχουσες προτάσεις και αρχιτεκτονικές και μέσα από τη μελέτη αυτή καθορίστηκαν οι ανάγκες των υποσυστημάτων. Πιο συγκεκριμένα,

- για όλα τα συστήματα χρησιμοποιήθηκε η γλώσσα προγραμματισμού Java (8) και μόνο ανοικτού κώδικα εργαλεία και βιβλιοθήκες.
- Επίσης χρησιμοποιήσαμε την σχεσιακή Βάση Δεδομένων MySQL (9)
- την τεχνολογία Ajax (10) μέσω της βιβλιοθήκης JQuery (11)
- την Freemarker μηχανή προτύπων (template engine) (12)
- την βιβλιοθήκη GSON (13) της Google για μετατροπή των Java αντικειμένων σε σειριακή μορφή (κειμένου)
- την JCommander (14) για την διαχείριση των αρχικών ρυθμίσεων και παραμέτρων της εφαρμογής του Restful API server
- την υπηρεσία Google Maps (15) για εύκολη διαχείριση γεωγραφικών συντεταγμένων και τοποθεσιών
- και τον κειμενογράφο CKEditor (16), επεξεργαστή κειμένου ο οποίος ενσωματώνεται στον κώδικά των ιστοσελίδων μας και προσφέρει στον χρήστη άνετη επεξεργασία κειμένου με πολλά εργαλεία επεξεργασίας και επιλογές.

Όλα τα παραπάνω είναι λίγο-πολύ γνωστά και για αυτό έχουμε μόνο βιβλιογραφική αναφορά, στην συνέχεια όμως θα περιγράψουμε τα σημαντικά στοιχεία που χρήζουν αναφοράς.

2.5.1 Πάροχος φιλοξενίας Okeanos (cloud service)

Ο Ωκεανός είναι ένας Ελληνικός πάροχος υπηρεσιών φιλοξενίας και cloud service (IaaS Service). Δημιουργήθηκε από το GRNET (17) για την Ακαδημαϊκή και Ερευνητική κοινότητα. Η πλατφόρμα πάνω στην οποία στηρίχθηκε ο Ωκεανός ονομάζεται Synnefo (18). Το Synnefo είναι μια πλήρης και ανοικτού κώδικα (open source cloud stack software) πλατφόρμα που γράφτηκε στην γλώσσα προγραμματισμού Python και παρέχει υπηρεσίες σύννεφου (cloud service) όπως: υπηρεσίες αποθήκευσης, δικτύου, υπολογιστικές υπηρεσίες, κα.

Για τις ανάγκες της Εργασίας μας, αιτηθήκαμε μια εικονική μηχανή την οποία μας παρείχαν χωρίς κανένα πρόβλημα. Έτσι εγκαταστήσαμε ένα Ubuntu server (19) λειτουργικό σύστημα το οποίο χρησιμοποιήθηκε για την φιλοξενία του Restful API διακομιστή, του Web Server και της Βάσης Δεδομένων.

2.5.2 Το Git και το Bitbucket για έλεγχο εκδόσεων πηγαίου κώδικα

Το Git (20) είναι ένα σύστημα ελέγχου εκδόσεων (λέγεται και σύστημα ελέγχου αναθεωρήσεων ή σύστημα ελέγχου πηγαίου κώδικα) με έμφαση στην ταχύτητα, στην ακεραιότητα των δεδομένων και στην υποστήριξη για κατανεμημένες μη γραμμικές ροές εργασίας. Το Git σχεδιάστηκε και αναπτύχθηκε αρχικά από τον Linus Torvalds (21) για τη ανάπτυξη του πυρήνα Linux το 2005 και έχει γίνει από τότε το πιο διαδεδομένο σύστημα ελέγχου εκδόσεων για ανάπτυξη λογισμικού. Όπως τα περισσότερα άλλα κατανεμημένα συστήματα ελέγχου εκδόσεων/αναθεωρήσεων και αντίθετα με τα περισσότερα συστήματα πελάτη-διακομιστή, κάθε κατάλογος εργασίας του Git είναι ένα ολοκληρωμένο αποθετήριο λογισμικού με πλήρες ιστορικό και δυνατότητες πλήρους παρακολούθησης της έκδοσης, ανεξάρτητα από την πρόσβαση δικτύου ή ενός κεντρικού διακομιστή. Όπως ο πυρήνας Linux, το Git είναι Ελεύθερο λογισμικό που διανέμεται κάτω από τους όρους της έκδοσης 2 της Γενικής Άδειας Δημόσιας Χρήσης GNU.

Το Bitbucket (22) είναι μια υπηρεσία διαδικτύου η οποία, μεταξύ άλλων υλοποιεί το Git. Χρησιμοποιήσαμε την υπηρεσία αυτή για τον έλεγχο εκδόσεων όλης της εφαρμογής και των υποσυστημάτων της έτσι ώστε να μπορούμε να έχουμε σε ένα σημείο, προσβάσιμο από παντού, όλο τον πηγαίο κώδικα με το ιστορικό του και την πιο πρόσφατη έκδοσή του.

2.5.3 Restful Web Services - REST API

Οι υπηρεσίες διαδικτύου Restful (5) είναι στην ουσία μια αρχιτεκτονική διαδικτύου με βάση την οποία τα πάντα στο διαδίκτυο είναι απλά πόροι (resources). Οι υπηρεσίες διαδικτύου Restful σχεδιάστηκαν να είναι επεξεργαστικά ελαφριές, να μπορούν να επεκταθούν αλλά και να συντηρούνται εύκολα. Η κύρια χρήση τους είναι η δημιουργία διαδικτυακών διεπαφών (API), είναι ένας τρόπος παροχής διαλειτουργικότητας μεταξύ συστημάτων υπολογιστών στο Διαδίκτυο, οι υπηρεσίες Web που είναι συμβατές με το REST επιτρέπουν στα αιτούμενα συστήματα να έχουν πρόσβαση και να χειρίζονται τις αναπαραστάσεις κειμένου των πόρων του Διαδικτύου χρησιμοποιώντας ένα ομοιόμορφο και προκαθορισμένο σύνολο λειτουργιών.

Όλη η επικοινωνία της εφαρμογής του χρήστη με τον διακομιστή γίνετε αποκλειστικά μέσω Restful web services. Αυτό μας δίνει την δυνατότητα να αλλάξουμε, όποτε το επιθυμούμε ή απαιτηθεί, την γραφική διεπαφή χωρίς να κάνουμε τίποτα απολύτως στον διακομιστή. Επίσης δίνουμε την δυνατότητα σε οποιαδήποτε άλλη εφαρμογή ή σύστημα να επικοινωνήσει με το δικό μας σύστημα απλά δίνοντάς του την λίστα με τους πόρους που εξυπηρετούμε (resources/end points).

2.5.4 JSON Web Tokens (JWT) και Nimbus JOSE+JWT

Το JSON Web Token (JWT) (23) είναι ένα ανοικτό πρότυπο (RFC 7519) (24) που ορίζει έναν συμπαγή και αυτόνομο τρόπο για την ασφαλή μετάδοση πληροφοριών ως αντικείμενα JSON, μεταξύ των μερών σε ένα κανάλι επικοινωνίας. Αυτές οι πληροφορίες μπορούν να επαληθευτούν και είναι αξιόπιστες επειδή υπογράφονται ψηφιακά. Τα JWTs μπορούν να υπογραφούν χρησιμοποιώντας ένα μυστικό κλειδί (με τον αλγόριθμο HMAC) ή ένα ζεύγος δημόσιου / ιδιωτικού κλειδιού χρησιμοποιώντας RSA. Λόγο του μικρού τους μεγέθους μπορούν να σταλθούν μέσω του URL η ως παράμετρος σε HTTP POST αίτημα. Επίσης μπορεί να περιλαμβάνει όλη την απαραίτητη πληροφορία για τον χρήστη και την τρέχουσα συνεδρία του (session). Αυτό έχει το πλεονέκτημα να μην χρειάζεται η τήρηση στοιχείων συνεδριών των χρηστών από την μεριά του διακομιστή γλυτώνοντας έτσι επεξεργαστική ισχύ, αιτήματα προς την βάση δεδομένων και πολύτιμο εύρος ζώνης.

Στο εφαρμογή μας, από την στιγμή που ένας χρήστης εισέλθει επιτυχώς στο σύστημα, όλη η επικοινωνία του με τον διακομιστή πραγματοποιείται χρησιμοποιώντας JWT με μοναδικό μυστικό κλειδί για κάθε χρήστη. Η βιβλιοθήκη που χρησιμοποιούμε για το JWT είναι η Nimbus-jose-jwt (25), ότι καλύτερο μπορέσαμε να βρούμε αφού υλοποιεί όλες τις προδιαγραφές του προτύπου, είναι

εύκολη στην χρήση, γραμμένη σε Java, με πολύ καλή τεκμηρίωση και αρκετά παραδείγματα για όλες τις δυνατότητες που παρέχει.

2.5.5 Spark micro framework

Το Spark (26) είναι ένα ελεύθερο και ανοιχτού κώδικα λογισμικό για δημιουργία εφαρμογών διαδικτύου και ειδικότερα για δημιουργία Restful micro services. Είναι γραμμένο σε Java και ο δημιουργός του, Per Wendel εμπνεύστηκε από το Sinatra (27) για την υλοποίησή του. Το Spark δεν ακολουθεί την κλασική τεχνική Model-View-Controller όπως άλλα frameworks (JAX-RS, Play framework και Spring MVC) αλλά στοχεύει στην γρήγορη δημιουργία εφαρμογών διαδικτύου με όσο το δυνατόν λιγότερο κώδικα και κόπο κάνοντας χρήση των Java 8 lambda expressions (28), ειδικότερα στην πιο πρόσφατη έκδοση (version 2, 2014). Το Spark εξ ορισμού τρέχει μέσω του ενσωματωμένου Jetty web server (7) αλλά μπορεί να τρέξει και σε άλλους διακομιστές διαδικτύου όπως ο Apache Tomcat (29).

3 Υλοποίηση

Στο κεφάλαιο αυτό θα δούμε αναλυτικά τα κύρια μέρη του συστήματος, δηλ. το Σύστημα Διαχείρισης, την εφαρμογή του χρήστη και το Σύστημα του διακομιστή. Και τα τρία αυτά υποσυστήματα είναι ανεξάρτητα μεταξύ τους αλλά χρησιμοποιούν μια κοινή βάση δεδομένων για την αποθήκευση των δεδομένων της εφαρμογής και τις συσχετίσεις μεταξύ των οντοτήτων. Θα δούμε επίσης και μερικά εργαλεία και υπηρεσίες που χρησιμοποιήσαμε για τις ανάγκες της εφαρμογής.

3.1 Google Maps: Ενεργοποίηση, ενσωμάτωση και χρήση

Επειδή οι χάρτες της Google (15) χρησιμοποιούνται στο Σύστημα Διαχείρισης και στην εφαρμογή του χρήστη, θα αναφέρουμε περιληπτικά πως ενεργοποιούνται για ενσωμάτωση και χρήση στις εφαρμογές αυτές.

Για να μπορεί κάποιος να χρησιμοποιήσει οποιαδήποτε υπηρεσία της Google θα πρέπει να έχει ένα λογαριασμό στην εταιρεία αυτή. Από την στιγμή που έχουμε δημιουργήσει τον λογαριασμό και συνδεθούμε στο σύστημα της Google μεταβαίνουμε στη σελίδα https://console.developers.google.com/apis/credentials από όπου δημιουργούμε ένα project και στην συνέχεια τα API keys που χρειαζόμαστε για πρόσβαση στις αντίστοιχες υπηρεσίες. Ποιο συγκεκριμένα για την εφαρμογή Android, το API key που δημιουργήσαμε πρέπει να το εισάγουμε στο “AndroidManifest.xml” αρχείο, ως στοιχείο «παιδί» (child element) του <application> στοιχείου και συγκεκριμένα ως

```
<meta-data
   android:name="com.google.android.geo.API_KEY"
   android:value="YOUR_API_KEY"/>
```

όπου το “YOUR_API_KEY” το αντικαθιστούμε με το δικό μας API key που δημιουργήσαμε προηγουμένως. Με αυτό τον τρόπο το framework του Android γνωρίζει πώς να το χρησιμοποιήσει αυτόματα σε κάθε αίτημα που κάνει προς τους διακομιστές της Google Maps που υποστηρίζουν την υπηρεσία αυτή. Στην Εικόνα 5 βλέπουμε ένα απόσπασμα του αρχείου AndroidManifest.xml όπου φαίνετε η παράγραφος που αφορά το API key στις σειρές 21-23.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3            xmlns:tools="http://schemas.android.com/tools"
4            package="com.zeageorge.xalerts_app">
5
6      <uses-permission android:name="android.permission.INTERNET"/>
7      <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
8      <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
9      <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
10     <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
11
12     <application
13         android:name=".App"
14         android:allowBackup="true"
15         android:icon="@mipmap/ic_launcher"
16         android:label="@string/app_title"
17         android:roundIcon="@mipmap/ic_launcher_round"
18         android:supportsRtl="true"
19         android:theme="@style/AppTheme"
20         tools:replace="android:label">
21         <meta-data
22             android:name="com.google.android.geo.API_KEY"
23             android:value="AIzaSyCXVX4KckCb6ooTdAURVyNXsXD19zHWgYQ"/>
24
25         <activity
26             android:name=".activity.MainActivity"
27             android:label="@string/app_title"
28             android:theme="@style/AppTheme.NoActionBar">
29         </activity>
```

Εικόνα 5: AndroidManifest.xml

Το άλλο σημείο που χρησιμοποιούμε την υπηρεσία Google Maps είναι το Σύστημα Διαχείρισης που θα δούμε αναλυτικά παρακάτω. Στο σύστημα αυτό, για να μπορέσουμε να εμφανίσουμε ενσωματωμένο τον χάρτη στις δικές μας ιστοσελίδες θα πρέπει να εισάγουμε την παρακάτω δήλωση script στον HTML κώδικα των σελίδων που θέλουμε να εμφανίζεται ο χάρτης :

```
<script
src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY
&callback=initMap2" async defer></script>
```

όπου το "YOUR_API_KEY" το αντικαθιστούμε με το δικό μας API key που έχουμε δημιουργήσει. Μετά την κλήση του script και όταν τελειώσει η αρχικοποίηση, θα καλεστεί η ρουτίνα που έχουμε δηλώσει στην μεταβλητή callback. Στο παράδειγμά μας θα καλεστεί η ρουτίνα "initMap2". Περισσότερες πληροφορίες και παραδείγματα χρήσης μπορούμε να βρούμε στο Google Maps JavaScript API (30).

```
114
115   // Removes the markers from the map, but keeps them in the array.
116   function clearMarkers2() {
117     setMapOnAll2(null);
118   }
119
120   // Shows any markers currently in the array.
121   function showMarkers2() {
122     setMapOnAll2(map2);
123   }
124
125   // Deletes all markers in the array by removing references to them.
126   function deleteMarkers2() {
127     clearMarkers2();
128     markers2 = [];
129   }
130 </script>
131 <script src="https://maps.googleapis.com/maps/api/js?key=AIzaSyBAVaQBQn1a2NtxrfmfJQGSUmLS-gh_I6Q&callback=initMap2" async defer></script>
132 <#include "footer.html" encoding="utf-8">
```

Εικόνα 6: Script ενεργοποίησης Google Maps

Στην Εικόνα 6 βλέπουμε ένα απόσπασμα από τον κώδικα της σελίδας που αφορά την επεξεργασία μιας Ενημέρωσης, εδώ βλέπουμε πως ενσωματώνουμε το απαραίτητο script με το δικό μας API key και την δική μας callback function (σειρά 131).

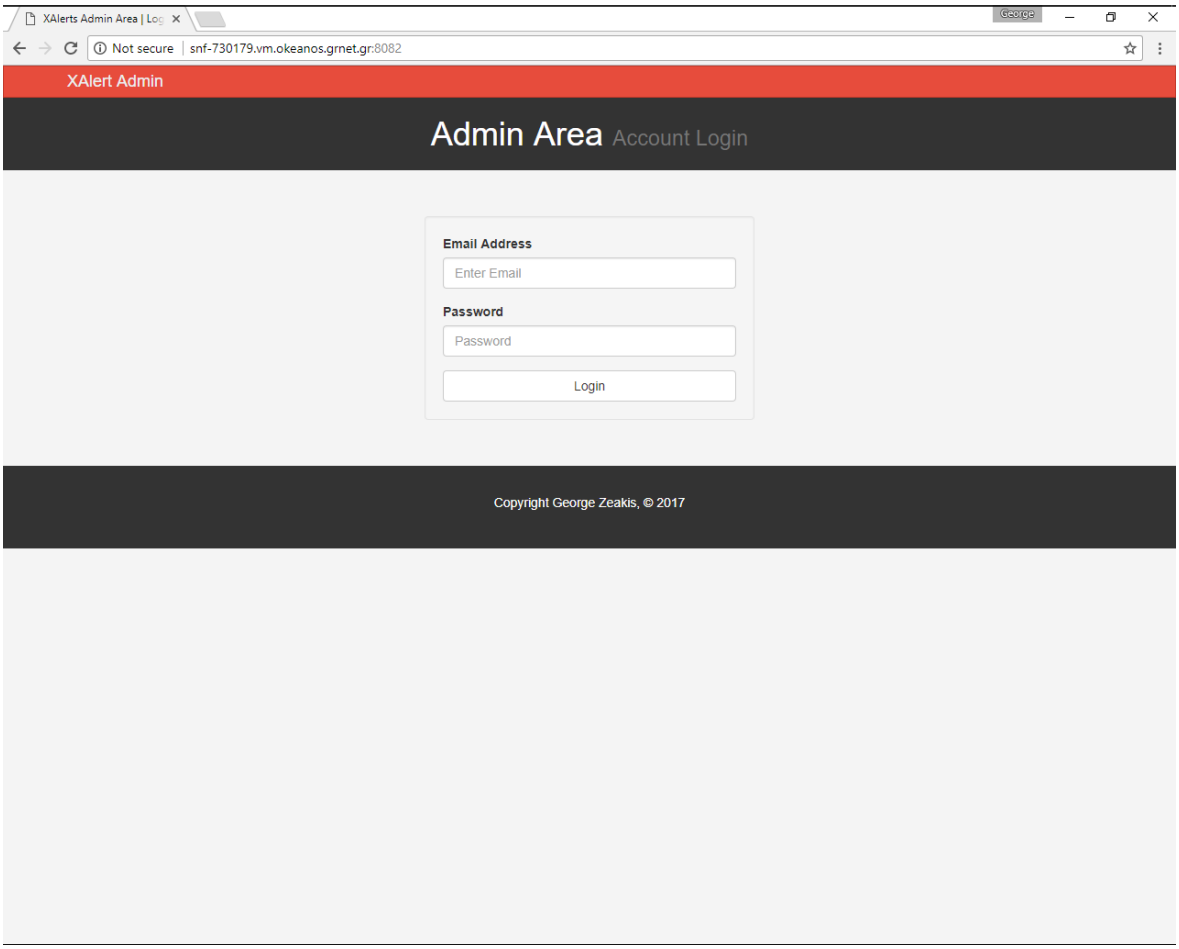
3.2 Σύστημα διαχείρισης

Το Σύστημα διαχείρισης δημιουργήθηκε με χρήση της γλώσσας προγραμματισμού Java (8), ποιο συγκεκριμένα, χρησιμοποιήθηκε το Spark java web micro framework (26) με την Free marker template engine (12) και τον Jetty διακομιστή διαδικτύου (7). Το πρότυπο της γραφικής διεπαφής είναι το Admin Bootstrap Theme της Traversy Media (31) , μια απλή, όμορφη και εύχρηστη διεπαφή για τις ανάγκες διαχείρισης του συστήματος. Τέλος, το όλο σύστημα φιλοξενήθηκε από τον πάροχο Okeanos IAAS (32).

Το συγκεκριμένο σύστημα υλοποιεί τις βασικές λειτουργίες διαχείρισης όπως η δημιουργία, η επεξεργασία και η διαγραφή περιεχομένου (Alerts, Alert Categories και Alert Comments) αλλά και διαχείριση των χρηστών του συστήματος, δηλαδή προσθήκη νέων χρηστών, επεξεργασία και διαγραφή χρηστών. Επίσης το σύστημα διαθέτει ένα ευέλικτο τρόπο για αναζήτηση-φιλτράρισμα περιεχομένου και χρηστών έτσι ώστε να μπορούμε να βρίσκουμε εύκολα αυτό που ψάχνουμε, είτε είναι περιεχόμενο είτε χρήστες.

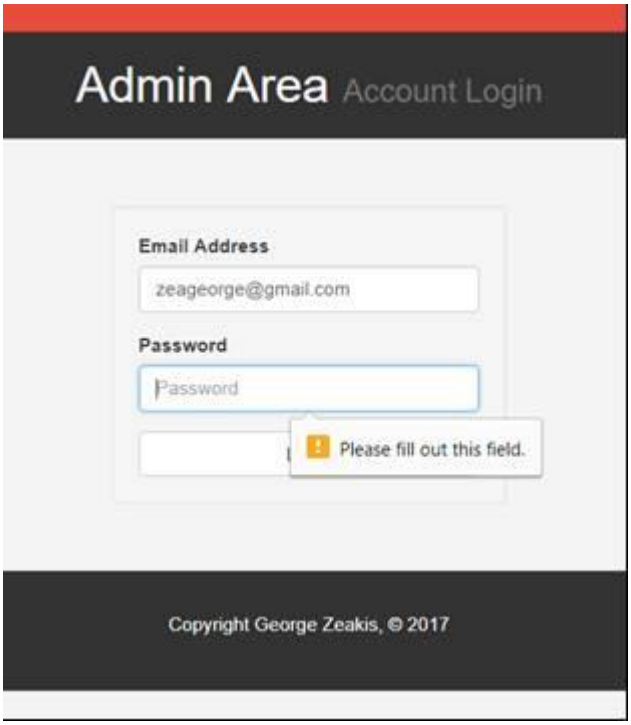
Στην συνέχεια θα δούμε αναλυτικά με εικόνες και θα περιγράψουμε όλες τις λειτουργίες του υποσυστήματος αυτού.

3.2.1 Η γραφική διεπαφή



Εικόνα 7: Login page

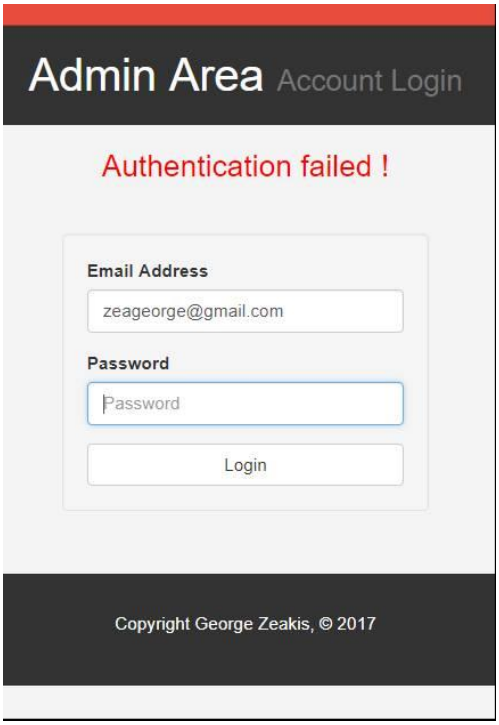
Στην Εικόνα 7 βλέπουμε την αρχική σελίδα όπου γίνετε η πιστοποίηση του χρήστη, δηλ του διαχειριστή του συστήματος. Εδώ πρέπει να εισάγουμε σωστά το όνομα χρήστη και τον κωδικό πρόσβασης ώστε να εισέλθουμε στο σύστημα.



Εικόνα 8: Validation of fields in Login page

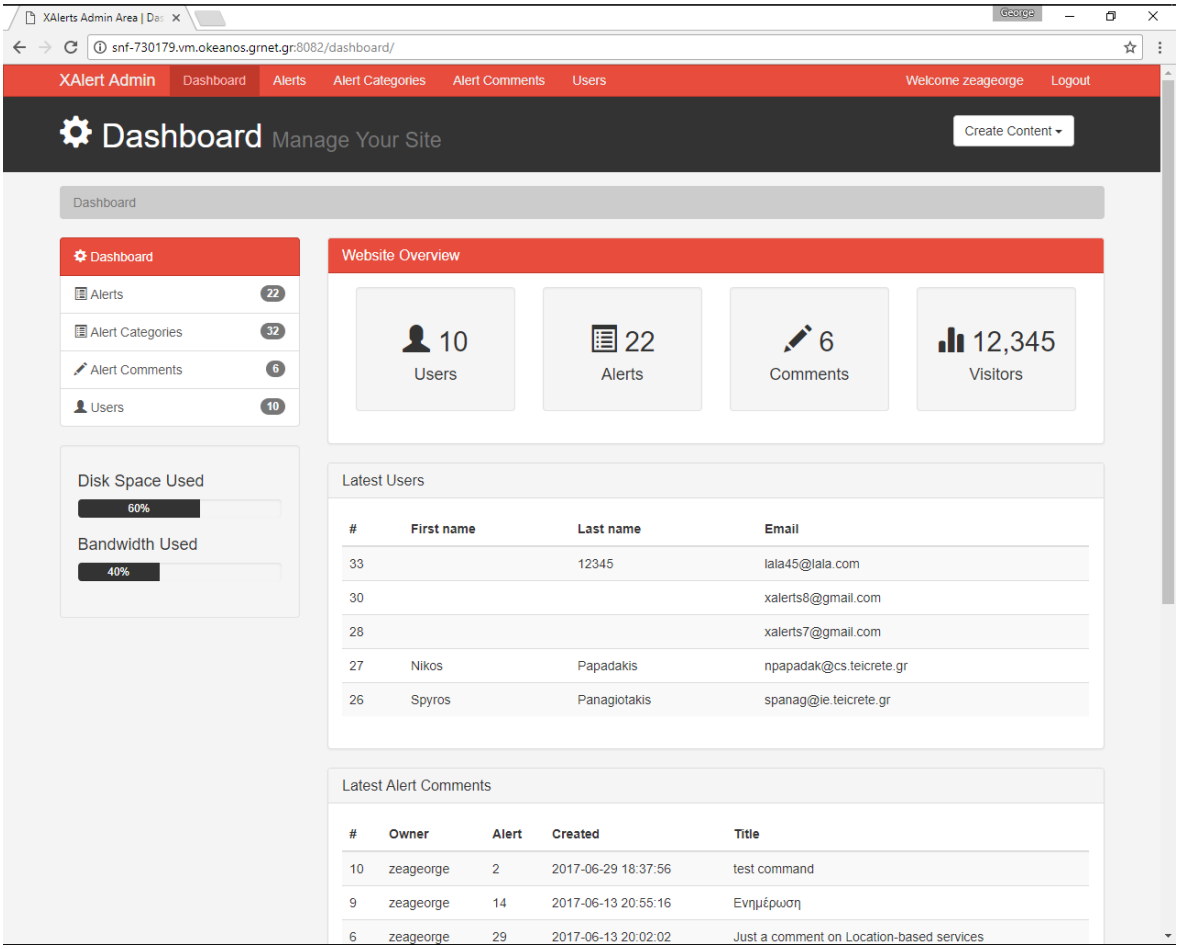
Επίσης χρησιμοποιήσαμε αρκετά στοιχεία της HTML5 (33), ένα από αυτά είναι ο έλεγχος των στοιχείων που εισάγει ο χρήστης, στην μεριά του πελάτη.

Στην Εικόνα 8 βλέπουμε ένα παράδειγμα όπου το πεδίο Password είναι υποχρεωτικό και δεν μπορεί να προχωρήσει ο χρήστης εάν δεν ορίσει τιμή για αυτό. Η HTML5 αναλαμβάνει να ενημερώσει αυτόματα τον χρήστη για τα αποτελέσματα της επικύρωση των στοιχείων με αυτόματο μήνυμα.



Εικόνα 9: Authentication

Σε περίπτωση που ο χρήστης δώσει λάθος στοιχεία τότε το σύστημα τον ενημερώνει με κατάλληλο μήνυμα [Εικόνα 9].

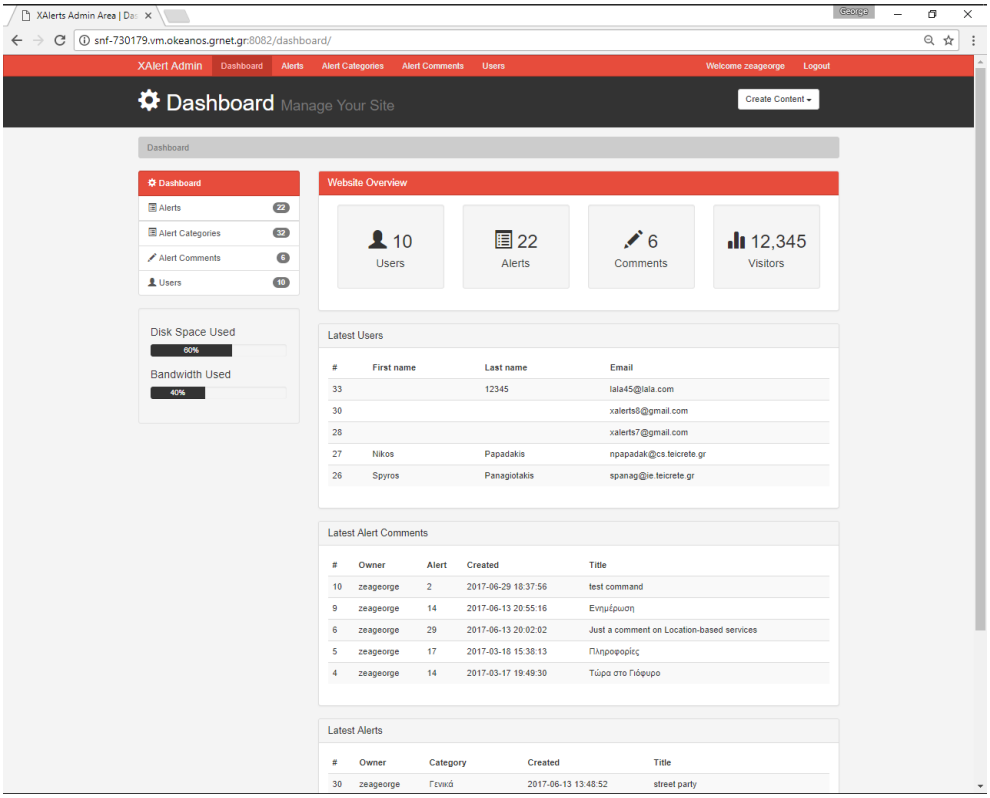


Εικόνα 10: Dashboard

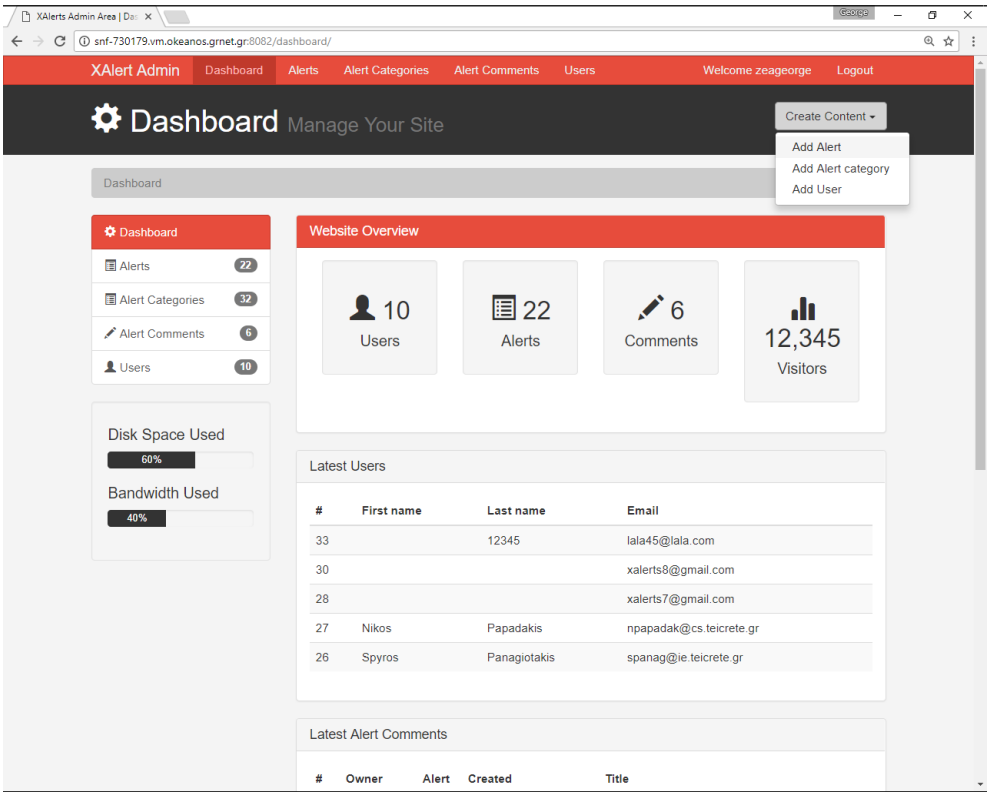
Μετά την επιτυχή είσοδο του διαχειριστή στο σύστημα, του παρουσιάζουμε την Εικόνα 10 (βλ. επίσης την Εικόνα 11) όπου με μια ματιά μπορεί να δει την συνολική εικόνα λειτουργίας του συστήματος. Το Dashboard είναι η κεντρική σελίδα από όπου μπορούμε να εκτελέσουμε όλες τις λειτουργίες που θέλουμε, δηλ. να διαχειριστούμε το περιεχόμενο και τους χρήστες. Επίσης βλέπουμε τα πιο πρόσφατα σημαντικά γεγονότα που συμβαίνουν στο σύστημα, όπως τους πιο πρόσφατους χρήστες, σχόλια και Alerts.

Από εδώ, ο διαχειριστής μπορεί επίσης να δει άλλα χρήσιμα στοιχεία όπως δείκτες που αφορούν τον διακομιστή, πχ. ελεύθερος αποθηκευτικός χώρος, εύρος ζώνης που έχει χρησιμοποιηθεί κα. Επίσης βλέπουμε πόσοι χρήστες υπάρχουν στο σύστημα, πόσα Alerts, πόσα Alert Comments & Alert Categories. Τέλος από εδώ μπορούμε να προσθέσουμε περιεχόμενο και χρήστες από το κουμπί Create Content πάνω δεξιά [Εικόνα 12].

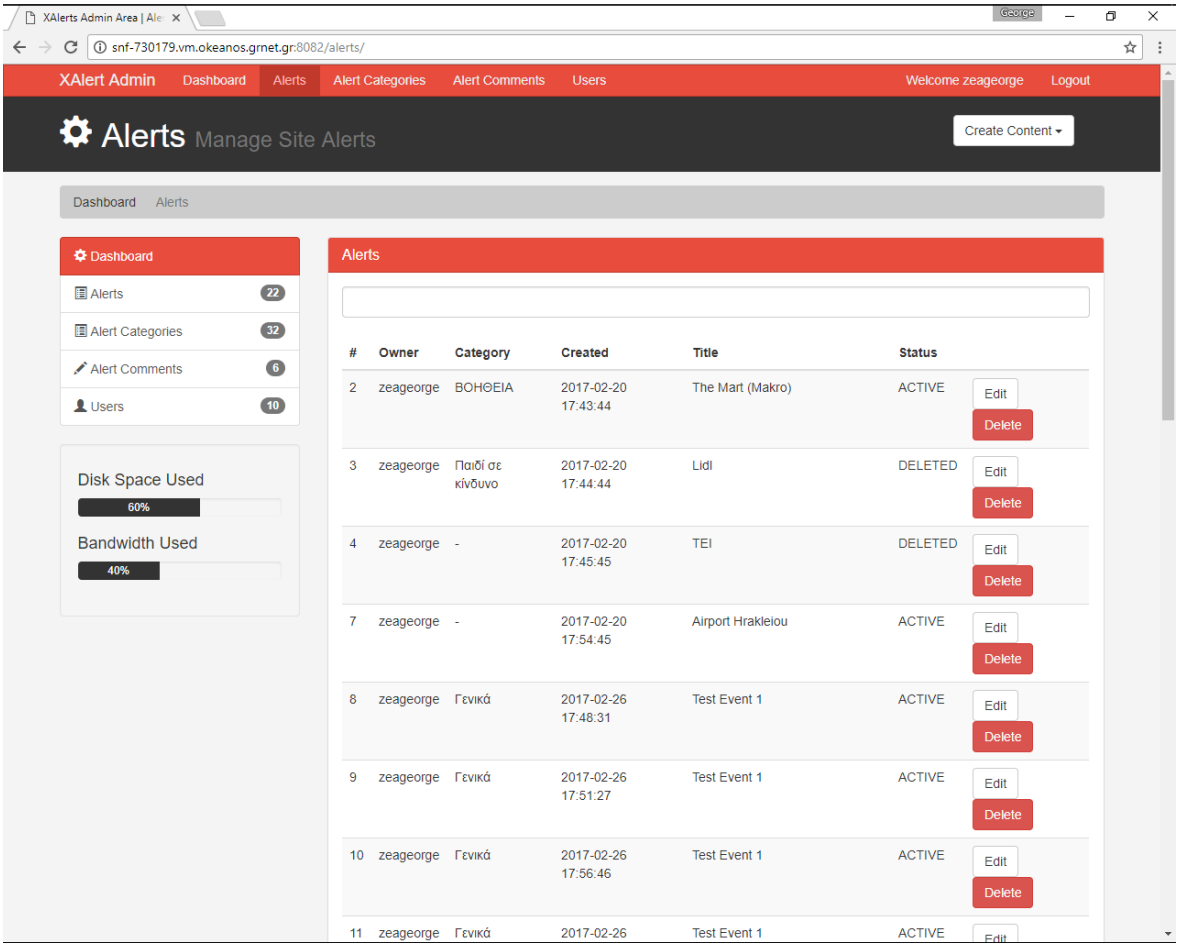
Όπως μπορούμε να δούμε η επιλογή του προτύπου είναι κατάλληλη αφού μας επιτραπεί με μεγάλη ευκολία, από μια σελίδα, να έχουμε μια συνολική εικόνα του συστήματος, ένα καθαρό, λιτό και πλήρες περιβάλλον για τον διαχειριστή.



Εικόνα 11: Dashboard details



Εικόνα 12: Create Content & Users

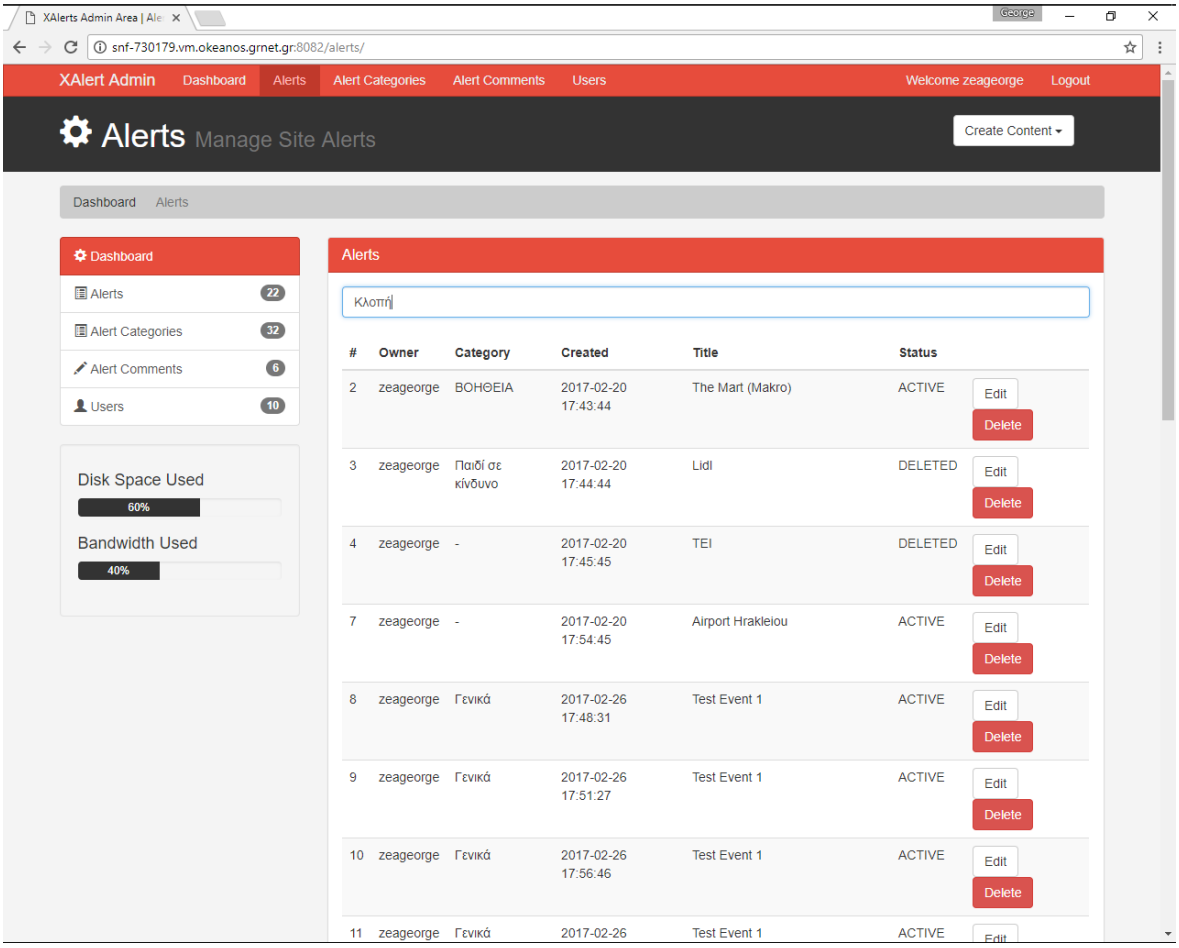


Εικόνα 13: Alerts list

Επιλέγοντας από το μενού την επιλογή “Alerts” μεταβαίνουμε στην σελίδα διαχείρισης των Alerts. Εδώ μας εμφανίζετε μια λίστα με όλα τα Alerts που έχουν αναρτήσει οι χρήστες. Η λίστα περιλαμβάνει τον μοναδικό αριθμό του κάθε Alert, το όνομα χρήστη του δημιουργού (ιδιοκτήτη), την κατηγορία που ανήκει, την ημερομηνία δημιουργίας, τον τίτλο και την τρέχουσα κατάσταση του κάθε Alert. Για κάθε ένα από αυτά υπάρχουν τα κουμπιά “Edit” & “Delete” για την επεξεργασία και την διαγραφή τους, αντίστοιχα. Αξίζει να σημειωθεί ότι το όνομα χρήστη και το όνομα της κατηγορίας είναι σύνδεσμοι όπου όταν επιλεγούν με το ποντίκι τότε ο χρήστης μεταβαίνει στις αντίστοιχες σελίδες επεξεργασίας αυτών [βλ. Εικόνα 14 και Εικόνα 15].

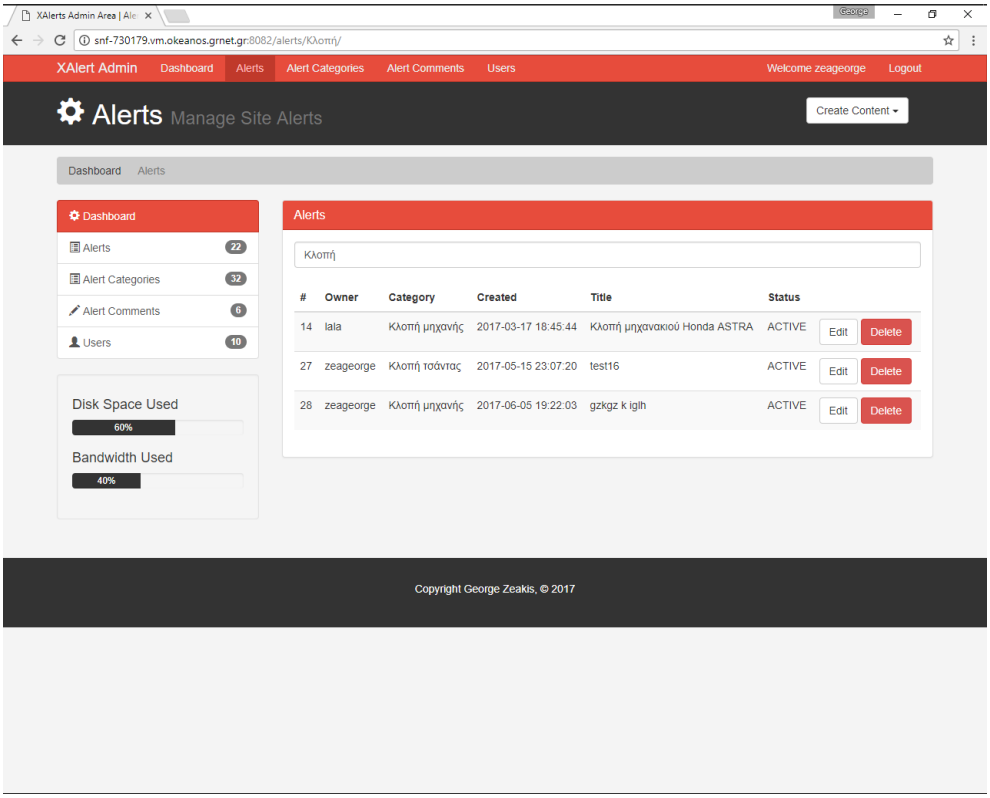
Εικόνα 14: User shortcuts in Alerts list

Εικόνα 15: Category shortcuts in Alerts list

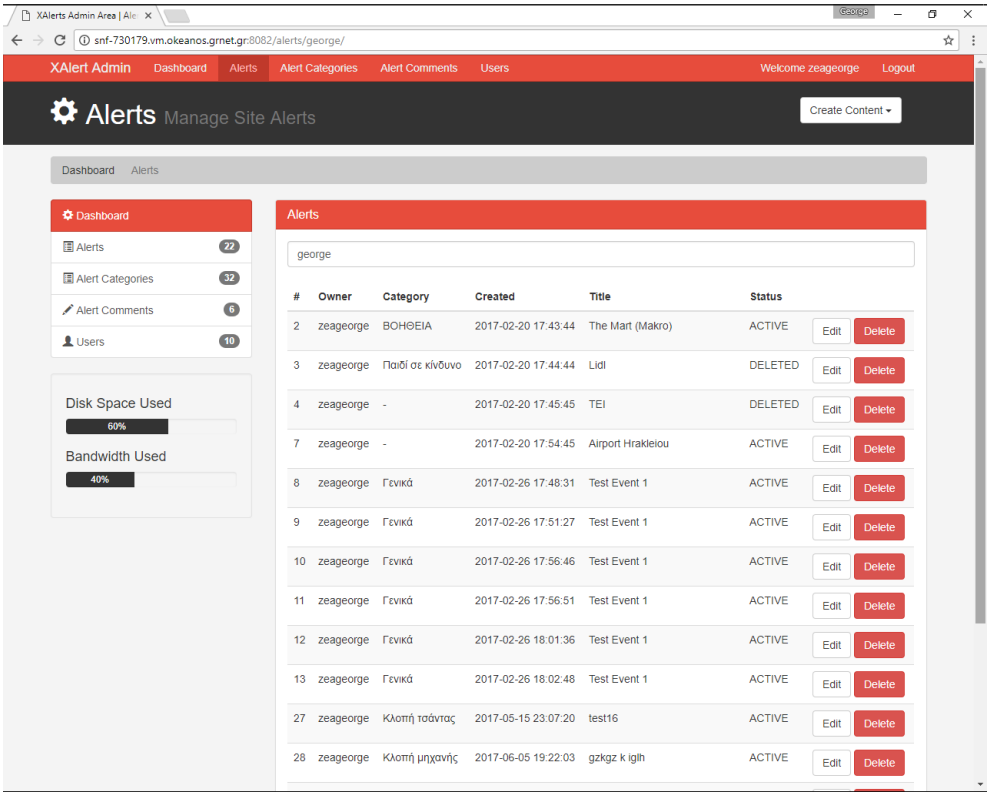


Εικόνα 16: Filtering/Searching in Alerts list

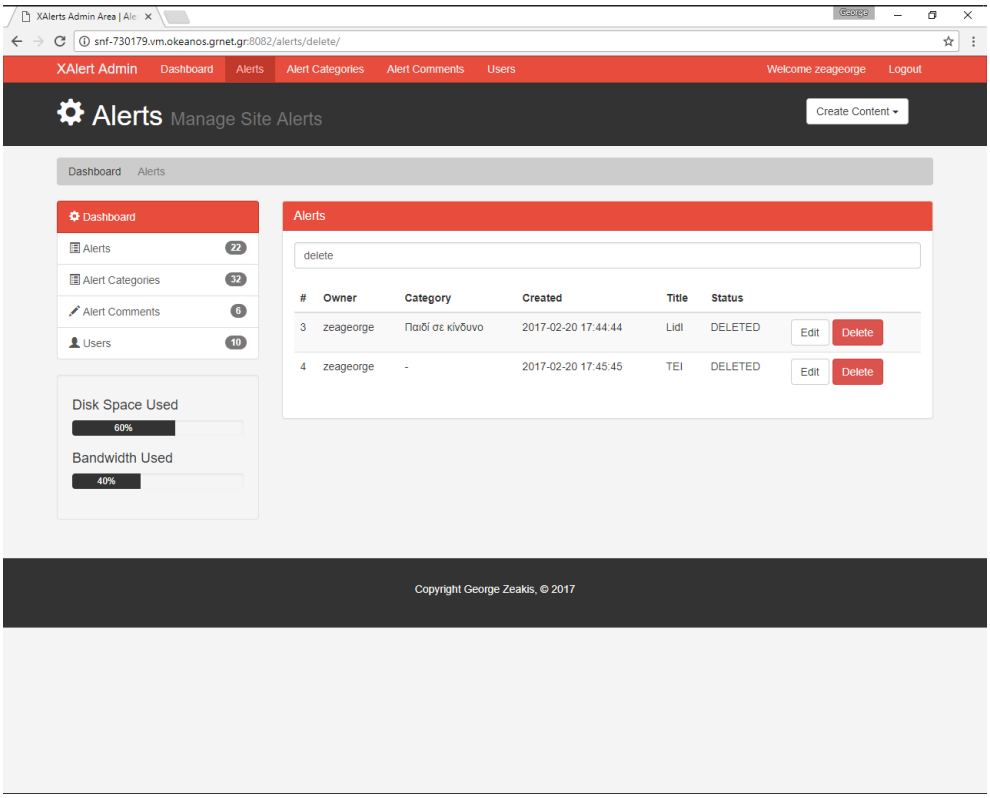
Το σύστημα προσφέρει επίσης τη λειτουργία της αναζήτησης (φιλτράρισμα) στην λίστα των Alerts. Εάν εισάγουμε πχ. την λέξη «κλοπή» στην μπάρα εισόδου κειμένου που βρίσκεται πάνω από την λίστα [Εικόνα 16], και πατήσουμε το πλήκρο “Enter”, τότε το σύστημα θα αναζητήσει την λέξη αυτή στα πεδία Ιδιοκτήτης, Κατηγορία, Τίτλος και Κατάσταση και θα μας εμφανίσει μόνο τα Alerts που περιέχουν την λέξη αυτή [Εικόνα 17, Εικόνα 18, Εικόνα 19].



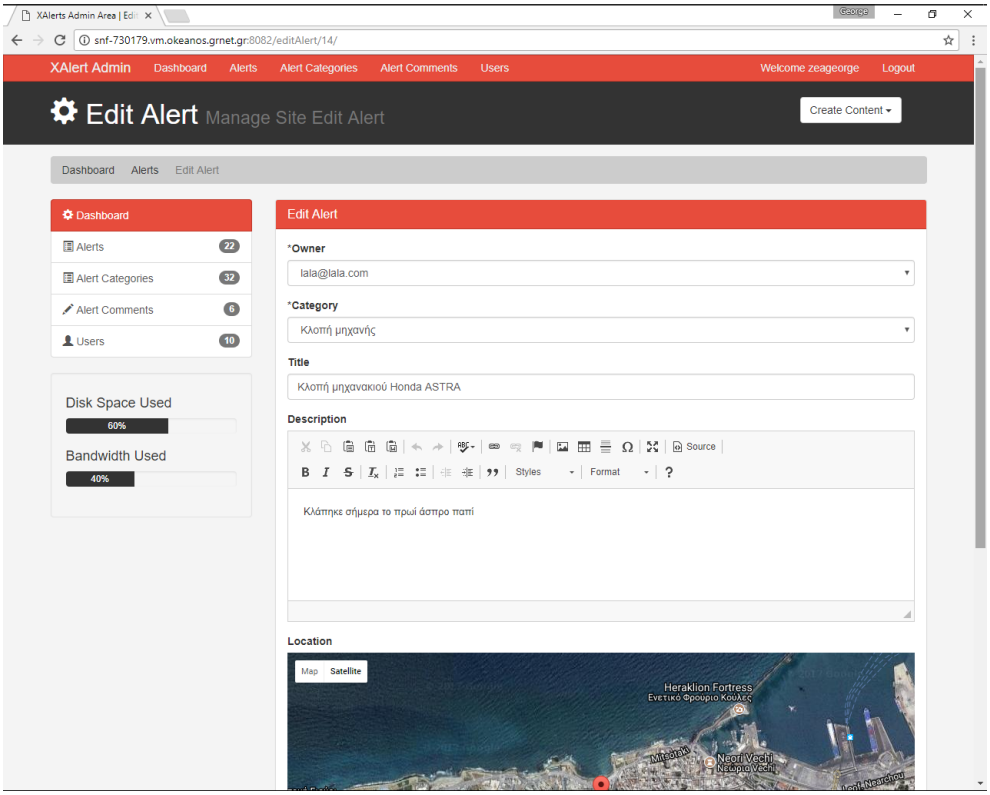
Εικόνα 17: Results of Filtering/Searching in Alerts list



Εικόνα 18: Results of Filtering/Searching in Alerts list, another example



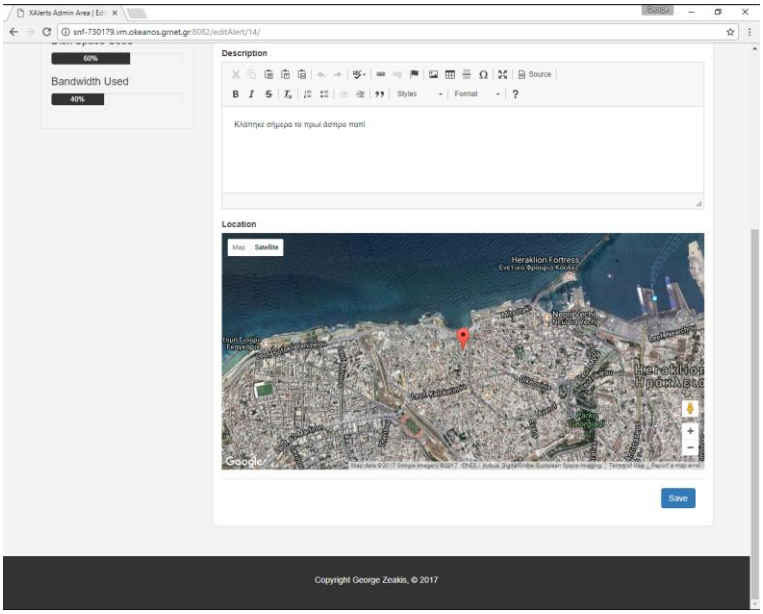
Εικόνα 19: Results of Filtering/Searching in Alerts list, another example



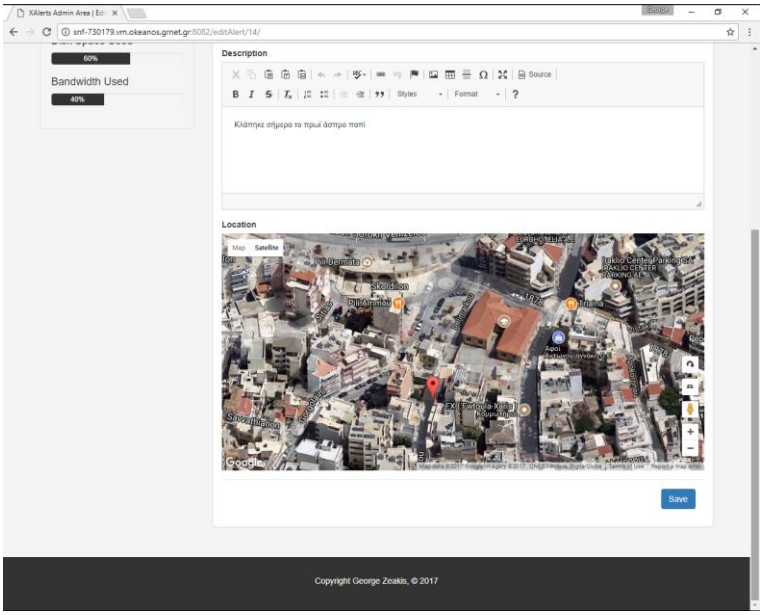
Εικόνα 20: Edit Alert

Στην Εικόνα 20 βλέπουμε τη σελίδα επεξεργασίας ενός Alert όπου μπορούμε να επιλέξουμε εάν θέλουμε, να αλλάξουμε τον ιδιοκτήτη του Alert, μπορούμε να αλλάξουμε την κατηγορία, τον τίτλο, το κείμενο της ανάρτησης καθώς επίσης και την τοποθεσία [Εικόνα 21]

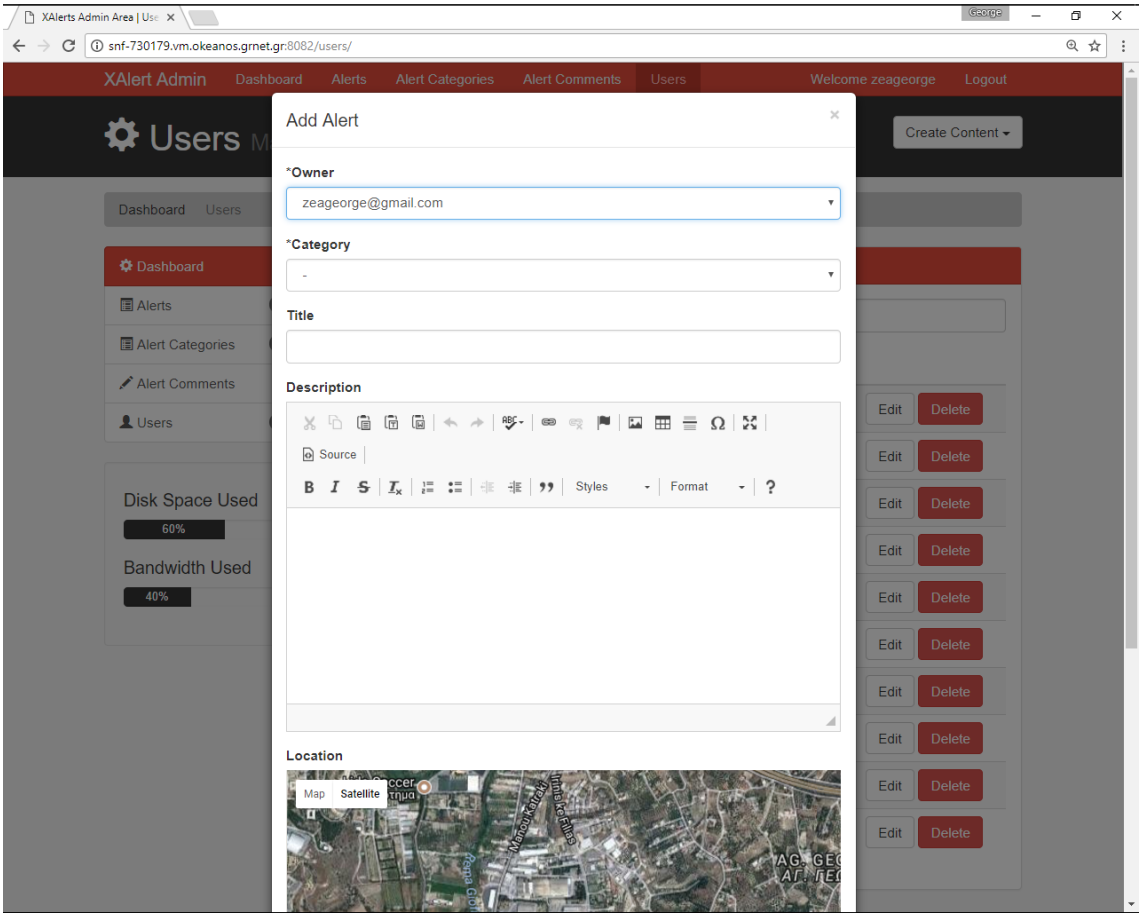
Είναι σημεντικό εδώ να αναφέρουμε ότι χρησιμοποιούμε τον CKEditor (16) επεξεργαστή κειμένου ο οποίος ενσωματώνεται στον κώδικά μας και προσφέρει άνετη επεξεργασία κειμένου με πολλά εργαλεία επεξεργασίας. Τέλος, χρησιμοποιούμε το Google Maps (15) για την αποικόνηση των τοποθεσιών των αναρτήσεων [Εικόνα 22]. Εδώ με ένα κλικ του ποντικιού επάνω στην χάρτη επιλέγουμε την τοποθεσία του Alert.



Εικόνα 21: Edit Alert location



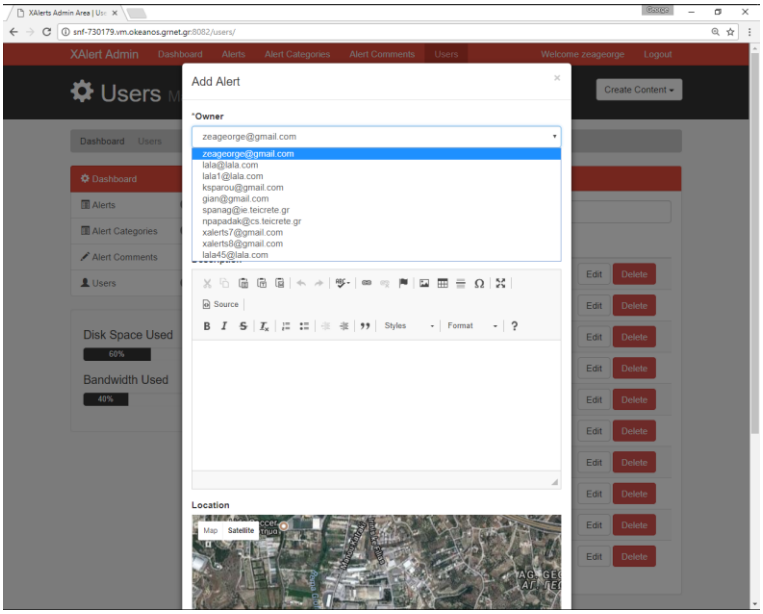
Εικόνα 22: Edit Alert location (zoomed)



Εικόνα 23: Create new Alert

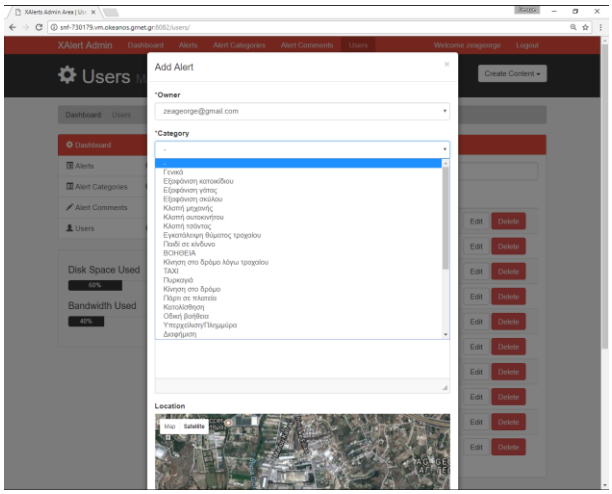
Επιλέγοντας το κουμπί “Create Content” και στην συνέχεια “Add Alert” μας παρουσιάζεται μια φόρμα για την δημιουργία ενός νέου Alert [Εικόνα 23]. Στην φόρμα αυτή πρέπει να επιλέξουμε

τον ιδιοκτήτη [Εικόνα 24], την κατηγορία [Εικόνα 25], να συμπληρώσουμε τον τίτλο και την περιγραφή της ενημέρωσης και τέλος να επιλέξουμε την τοποθεσία του Alert [Εικόνα 26]

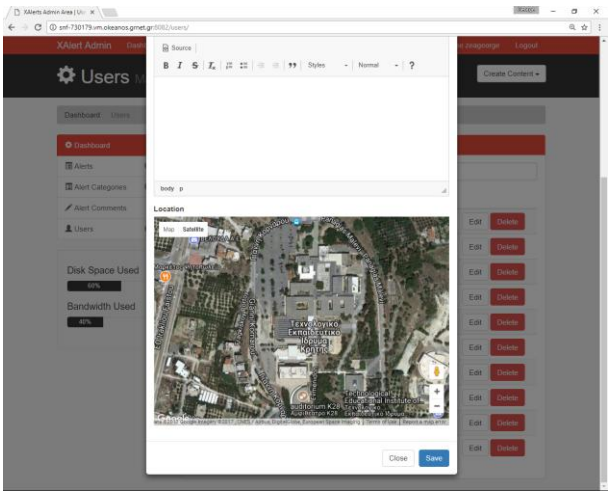


Εικόνα 24: Choose Owner for the new Alert

Κάνοντας κλικ επάνω στις "Κατηγορίες" εμφανίζετε μια λίστα με τις ονομασίες όλων των κατηγοριών από όπου ο διαχειριστής καλείτε να επιλέξει μια κατηγορία για το νέο Alert [Εικόνα 25].



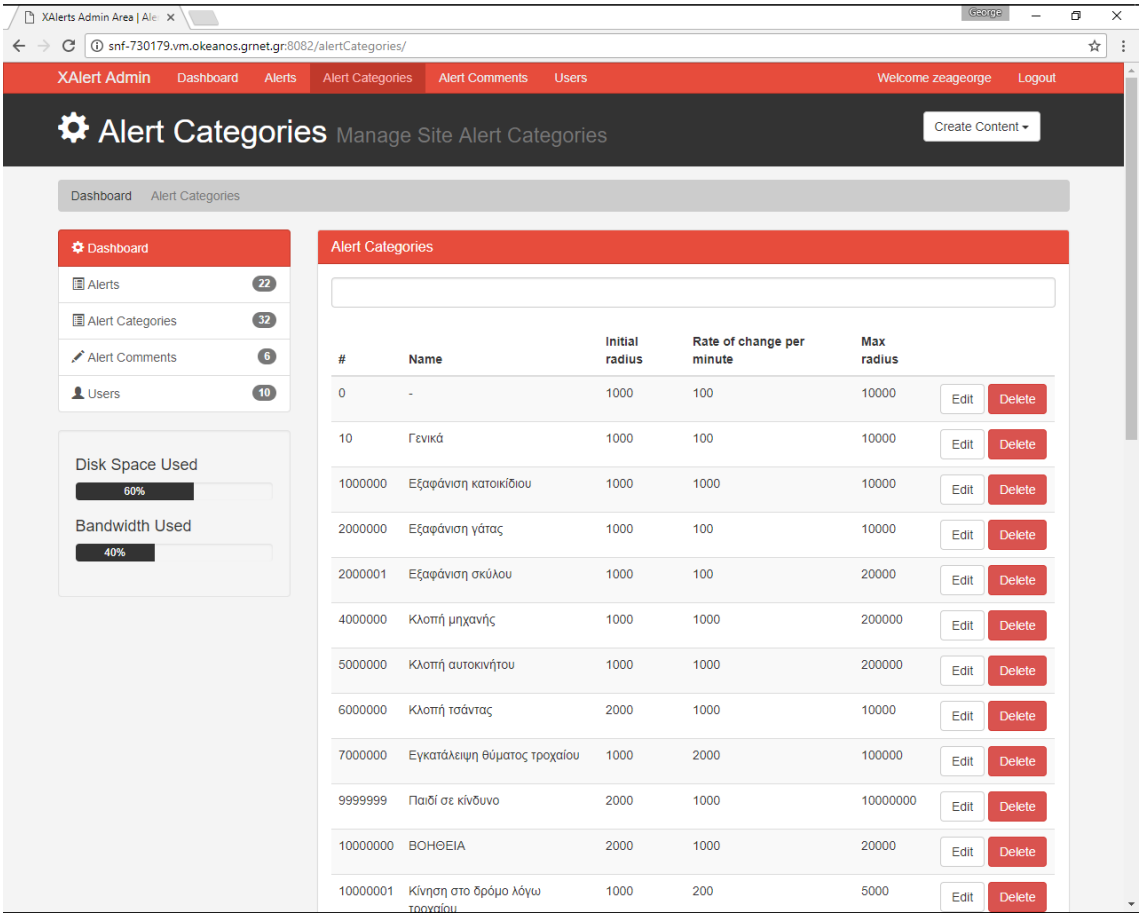
Εικόνα 25: Choose Category for the new Alert



Εικόνα 26: Choose Location for the new Alert

Με το Google Maps (15) έχουμε την δυνατότητα να περιηγηθούμε σε οποιοδήποτε μέρος, να δούμε τον χάρτη με εικόνες από δορυφόρο αλλά και να εστιάσουμε πολύ κοντά στην επιφάνεια της Γής με την λειτουργία Zoom In/Out. Όταν ο χρήστης κάνει κλικ με το ποντίκι του σε ένα σημείο στο χάρτη τότε αυτόματα το σύστημα καταγράφει τις συντεταγμένες του επιλεγμένου σημείου και τις καταχωρεί ως σημείο αναφοράς του νέου Alert [Εικόνα 26].

Στις παρακάτω εικόνες [Εικόνα 27 & Εικόνα 28] βλέπουμε την λίστα των Κατηγοριών των Αναρτήσεων όπου ο διαχειριστής μπορεί να δει πατώντας στο μενού “Alert Categories”. Στην λίστα αυτή εμφανίζονται όλες οι κατηγορίες που υπάρχουν στο σύστημα με τον μοναδικό αριθμό ταυτότητας, το όνομα της κατηγορίας, την αρχική ακτίνα ενημέρωσης χρηστών, το ρυθμό μεταβολής ανά λεπτό, τη μέγιστη ακτίνα «δράσης/ενημέρωσης» και τέλος τα κουμπιά “Edit” και “Delete” για την επεξεργασία και τη διαγραφή αντίστοιχα.



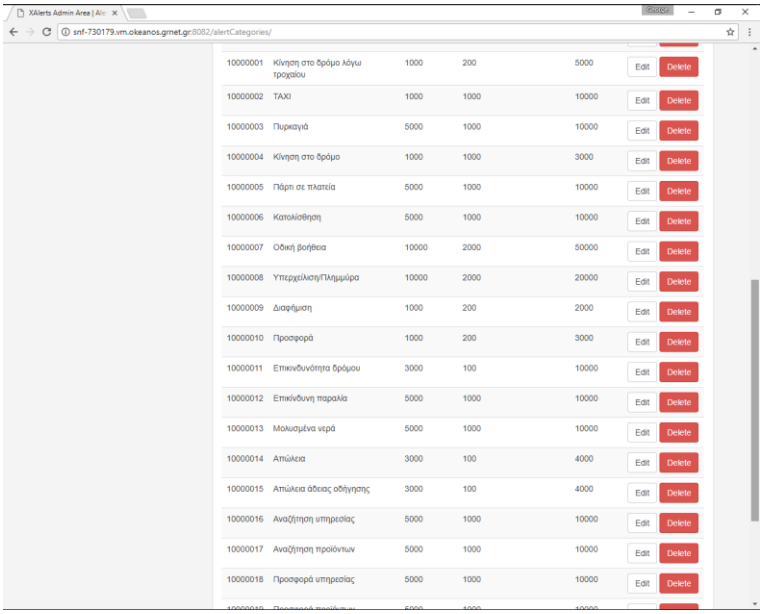
Εικόνα 27: Alert Categories list

Κάθε Alert πρέπει να ανήκει σε μια κατηγορία. Ο ρόλος των κατηγοριών, πέρα από την προφανή λειτουργία διαχωρισμού και κατηγοριοποίησης των Ενημερώσεων, είναι να παρέχει στο σύστημα τις απαραίτητες τιμές ώστε να μπορέσει να υπολογίσει ποιους χρήστες θα ενημερώσει ανάλογα με την κατηγορία του κάθε Alert.

Όταν κάποιος χρήστης δημιουργήσει ένα νέο Alert και το αντιστοιχήσει πχ. στην κατηγορία «Εξαφάνιση σκύλου» (#2000001) τότε το σύστημα αρχικά θα ενημερώσει όλους τους χρήστες που βρίσκονται σε ακτίνα 1000 μέτρων αφού αυτή είναι η αρχική ακτίνα ενημέρωσης χρηστών (Initial radius). Στην συνέχεια για κάθε λεπτό που περνάει θα προσθέτει 100 μέτρα στην αρχική ακτίνα. Η νέα ακτίνα που προκύπτει θα είναι πλέον η ακτίνα ενημέρωσης χρηστών. Εάν η νέα ακτίνα ενημέρωσης είναι μεγαλύτερη από την μέγιστη ακτίνα τότε η μέγιστη ακτίνα θα λαμβάνετε υπόψη και η νέα ακτίνα θα απορρίπτεται.

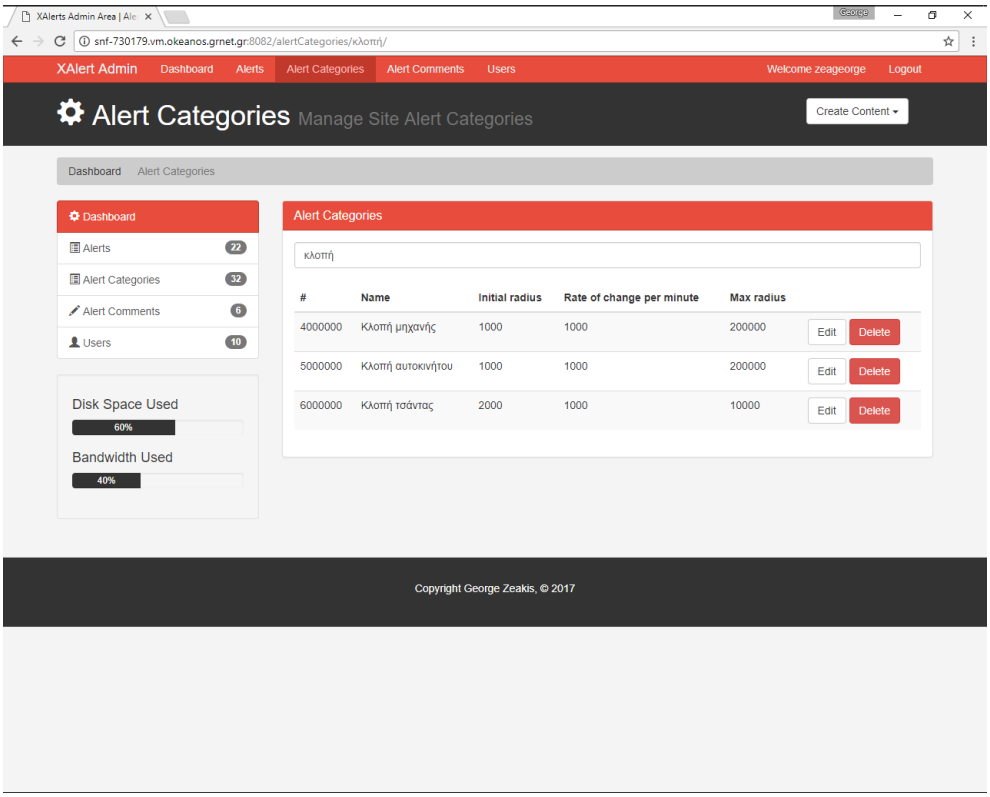
Οι τιμές των Initial radius, Rate of change per minute και Max radius πρέπει να επιλέγονται βάση λογικής, πχ. μια γάτα συνήθως δεν απομακρύνεται με την ίδια ταχύτητα σε σχέση με ένα σκύλο, έτσι ο ρυθμός μεταβολής ανά λεπτό θα είναι διαφορετικός στις δύο αυτές κατηγορίες. Επίσης ένας σκύλος καλύπτει μεγαλύτερες αποστάσεις από μια γάτα, άρα η μέγιστη ακτίνα ενημέρωσης χρηστών θα είναι μεγαλύτερη στην κατηγορία που αφορά τον σκύλο απ' ότι αυτή που αφορά την γάτα.

Αρχικά στο σύστημα έχουμε δημιουργήσει μια ευρεία γκάμα κατηγοριών [Εικόνα 28] ώστε να καλύψουμε ένα μεγάλο εύρος χρήσεων.



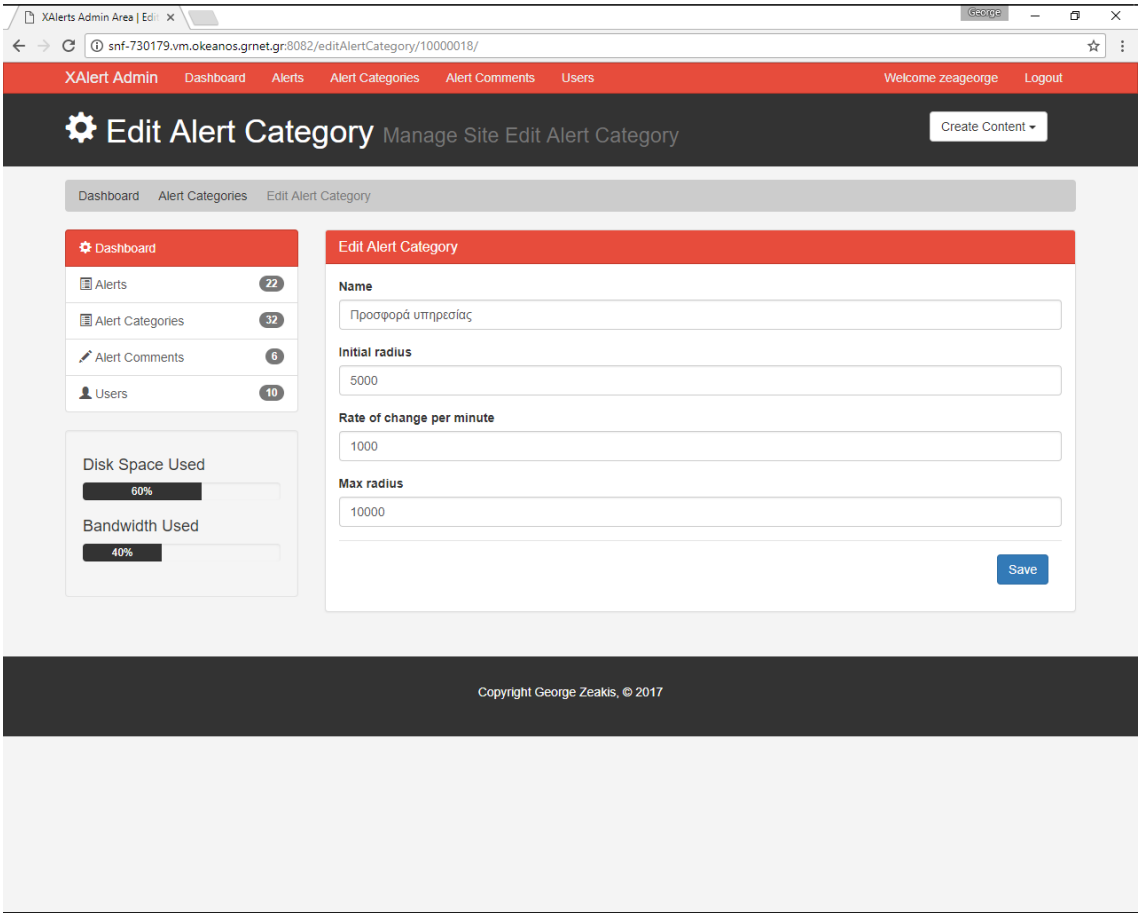
Εικόνα 28: Alert Categories list (continued)

Στην Εικόνα 29 μπορούμε να δούμε την λειτουργία της αναζήτησης όπου πληκτρολογώντας μια λέξη προς αναζήτηση και πατώντας το πλήκτρο “Enter”, το σύστημα μας επιστρέφει όλες τις κατηγορίες στις οποίες περιέχετε η λέξη που εισάγαμε. Έτσι μπορούμε εύκολα να βρούμε τις κατηγορίες που ψάχνουμε.



Εικόνα 29: Filtering/Searching in Categories list

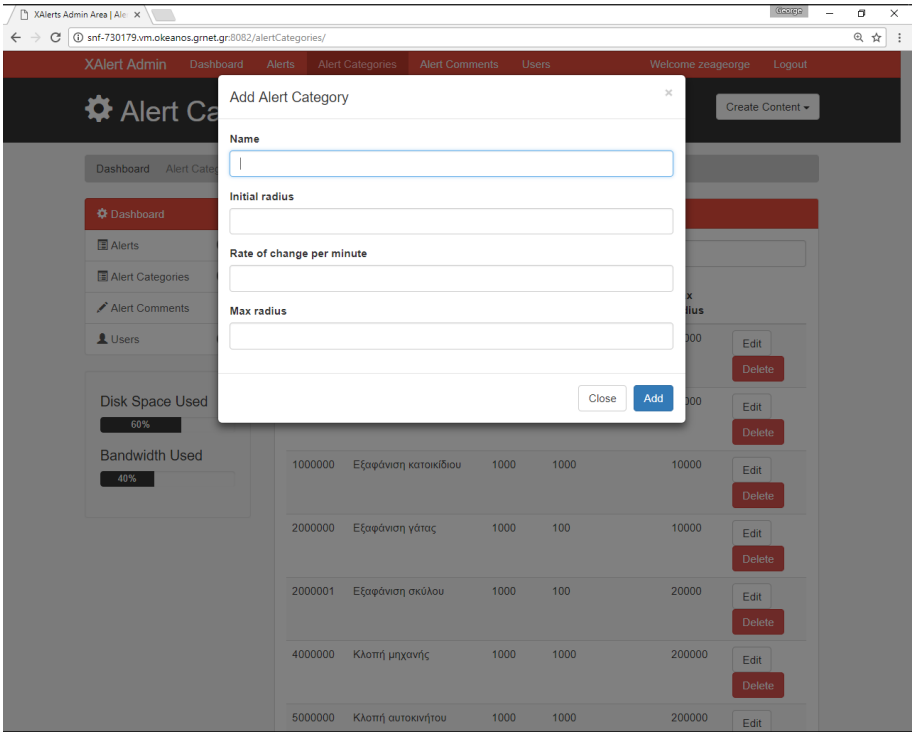
Στην παρακάτω εικόνα [Εικόνα 30] βλέπουμε την φόρμα επεξεργασίας μια κατηγορίας όπου μπορούμε να αλλάξουμε την ονομασία της κατηγορίας, την αρχική ακτίνα ενημέρωσης, τον ρυθμό μεταβολής ανά λεπτό και τέλος την μέγιστη ακτίνα ενημέρωσης. Πατώντας το κουμπί “Save” οι αλλαγές είναι άμεσες και οι νέες τιμές λαμβάνουν χώρα στην αμέσως επόμενη ενημέρωση χρηστών.



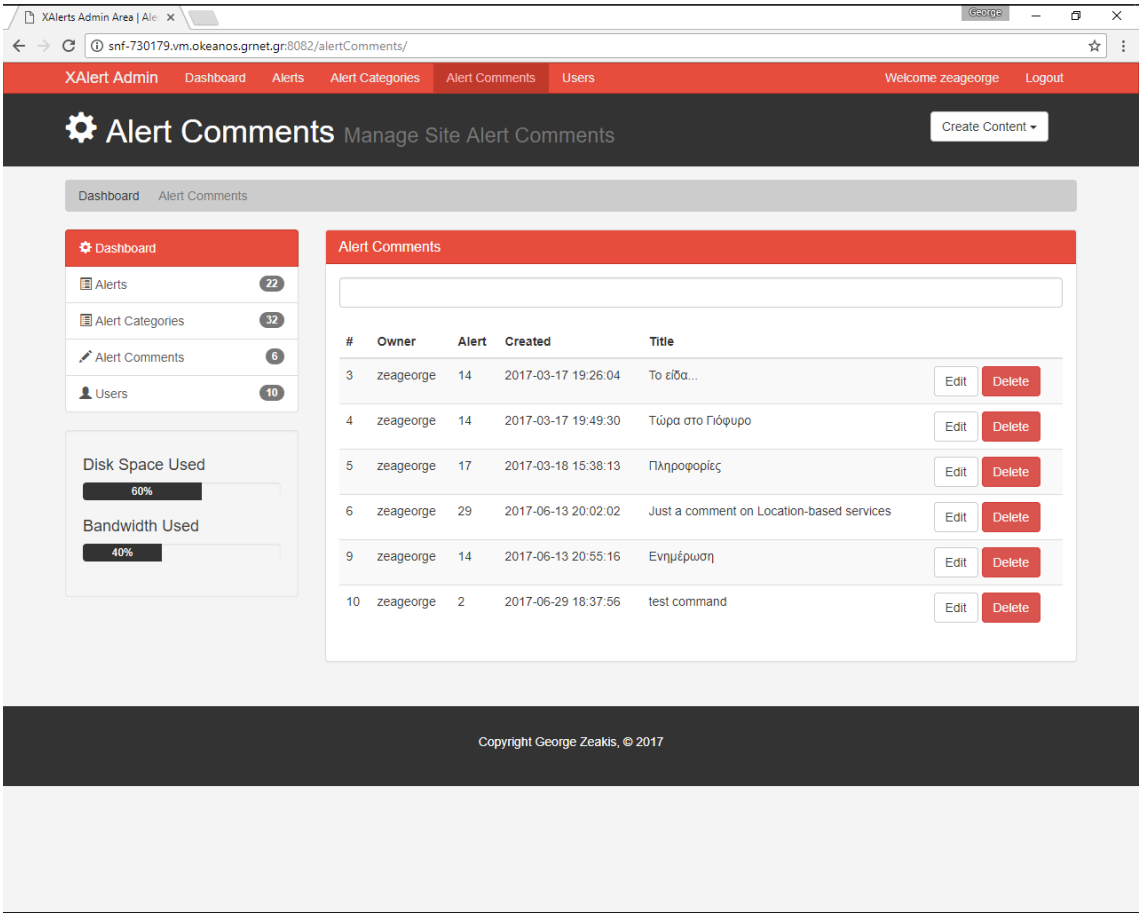
Εικόνα 30: Edit a Category

Επίσης μπορούμε να προσθέσουμε μια κατηγορία πατώντας στο κουμπί “Create Content”, πάνω δεξιά, και στην συνέχεια επιλέγοντας “Add Alert Category”. Έτσι μας εμφανίζεται η Εικόνα 31 όπου μας ζητείτε να εισάγουμε το όνομα της νέας κατηγορίας και από ένα θετικό αριθμό για τα πεδία: την τιμή της αρχικής ακτίνας ενημέρωσης χρηστών, τον ρυθμό μεταβολής ανά λεπτό και τέλος την μέγιστη ακτίνα ενημέρωσης χρηστών.

Η μονάδα μέτρησης για τις τιμές των πεδίων αυτών είναι το Μέτρο, πχ. για το πεδίο Initial Radius νοείτε η ακτίνα σε μέτρα από το σημείο δημιουργίας του Alert, όσοι χρήστες βρίσκονται μέσα σε αυτή την ακτίνα κατά τη δημιουργία του Alert θα ενημερωθούν άμεσα.

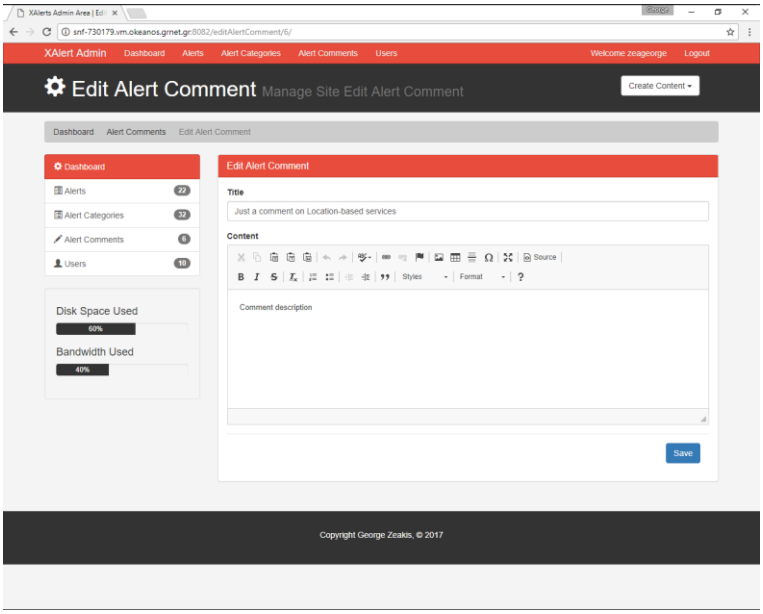


Εικόνα 31: Add new Category



Εικόνα 32: Comments list

Κάθε Alert μπορεί να έχει ένα ή περισσότερα σχόλια τα οποία αναρτούν οι χρήστες. Τα σχόλια αυτά αφορούν την Ενημέρωση στην οποία είναι αντιστοιχισμένα και μπορεί να είναι μια απάντηση σε κάποιο Alert ή σε κάποιο άλλο σχόλιο. Κάθε σχόλιο περιέχει ένα μοναδικό αριθμό ταυτότητας, έναν ιδιοκτήτη, τον αριθμό ταυτότητας της Ενημέρωσης που το αφορά, την ημερομηνία δημιουργίας, ένα τίτλο, μια περιγραφή και μια φωτογραφία, όπως βλέπουμε στην λίστα της Εικόνα 32

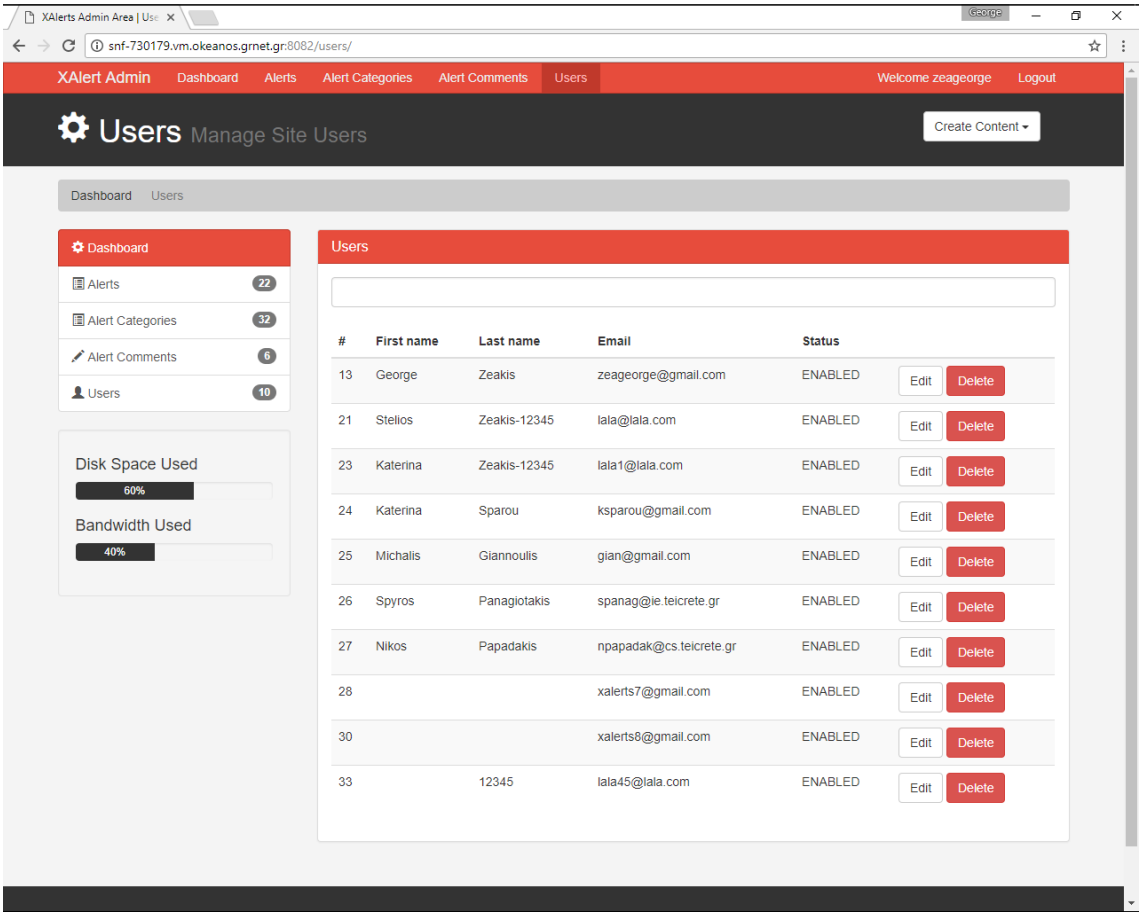


Εικόνα 33: Edit Comments

Κι εδώ έχουμε μια παρόμοια λίστα με τις προηγούμενες όπου για κάθε εγγραφή υπάρχουν δύο κουμπιά για την επεξεργασία και την διαγραφή αντίστοιχα. Επίσης έχουμε το γνωστό πεδίο εισαγωγή κειμένου με το οποίο κάνουμε αναζήτηση εγγραφών. Και σε αυτή την λίστα το όνομα χρήστη του

ιδιοκτήτη (Owner) αλλά και ο αριθμός ταυτότητας του Alert είναι και τα δύο σύνδεσμοι στις αντίστοιχες σελίδες επεξεργασίας του χρήστη και του Alert. Δηλαδή όταν κάνουμε κλικ με το ποντίκι επάνω στο όνομα του ιδιοκτήτη τότε μεταβαίνουμε στην σελίδα επεξεργασίας του αντίστοιχου χρήστη. Τέλος στην λίστα μας παρουσιάζονται επίσης τα πεδία Ημερομηνία δημιουργίας του σχολίου αλλά και ο τίτλος του.

Όταν πατάμε το κουμπί "Edit" σε κάποιο σχόλιο τότε μεταβαίνουμε στην σελίδα επεξεργασίας [Εικόνα 33] από όπου μπορούμε να αλλάξουμε τον τίτλο και το κείμενο του σχολίου. Έτσι ένας διαχειριστής μπορεί να ελέγχει άμεσα τι δημοσιοποιεί κάθε χρήστης ως σχόλιο και μπορεί να ελέγχει για κείμενα που παραβιάζουν τυχών Όρους χρήσης του συστήματος.



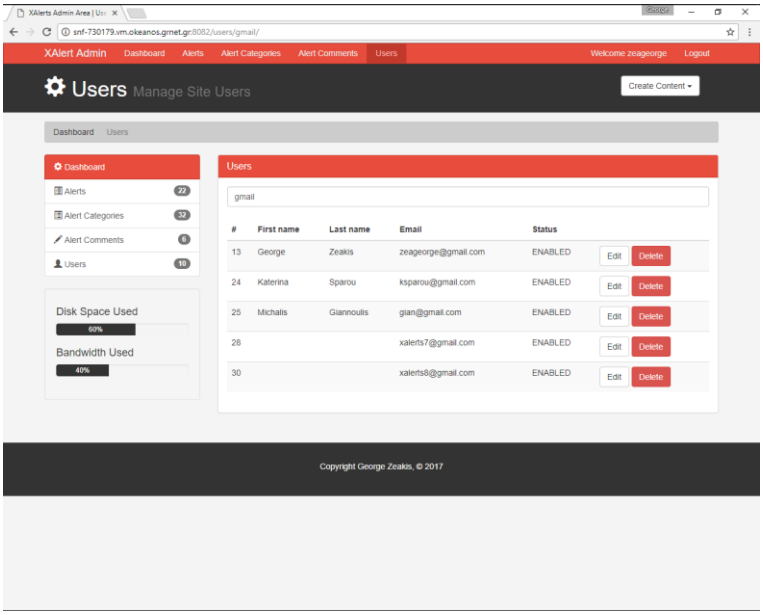
Εικόνα 34: Manage Users

Στην Εικόνα 34 βλέπουμε την κεντρική σελίδα διαχείρισης χρηστών όπου, ξανά, χρησιμοποιώντας την ίδια τεχνική και τον ίδιο, γνώριμο τρόπο, μας παρουσιάζεται ένα πεδίο εισαγωγής κειμένου για την αναζήτηση-φιλτράρισμα, μια λίστα με εγγραφές και τα αντίστοιχα κουμπιά ελέγχου για κάθε εγγραφή. Παρουσιάζουμε δηλαδή ένα ομοιογενές περιβάλλον για όλες τις λειτουργίες διαχείρισης κάνοντας έτσι την δουλειά του διαχειριστή, όσο πιο εύκολη γίνετε.

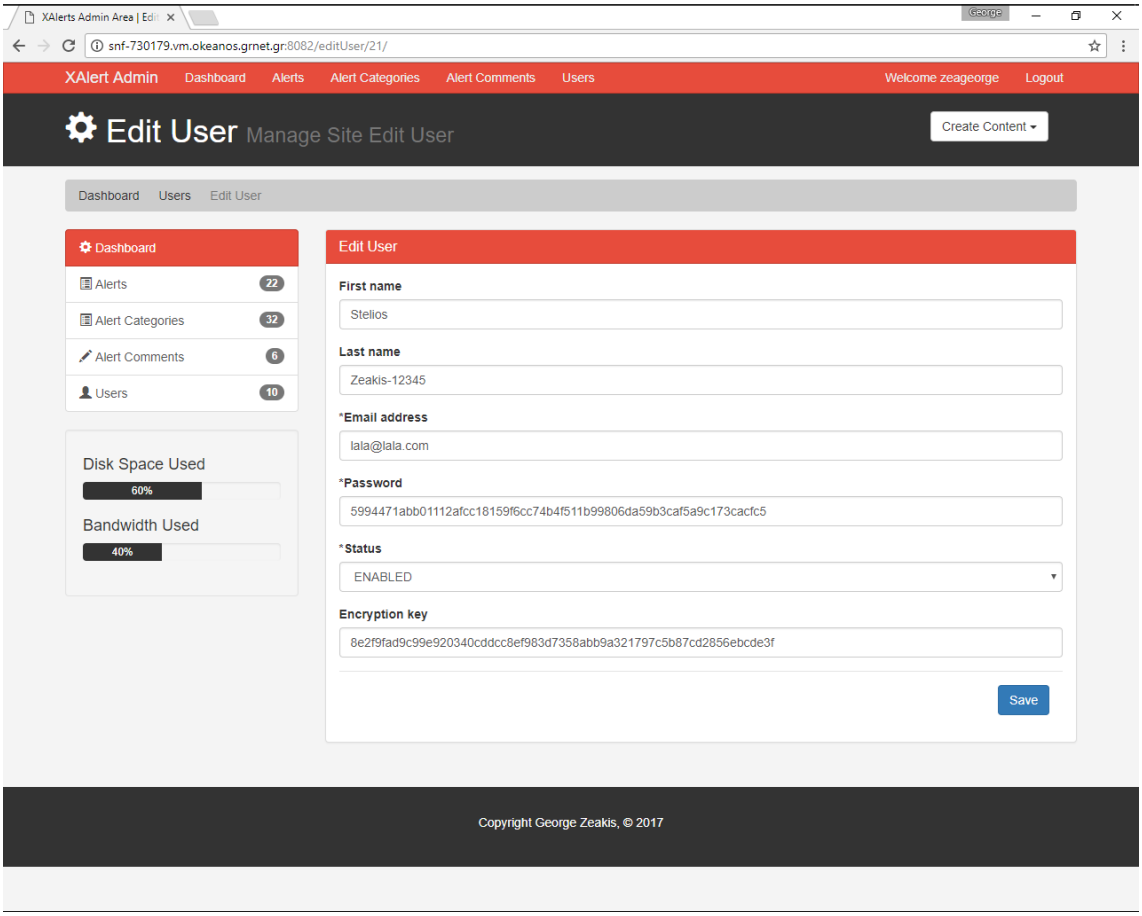
Στην λίστα βλέπουμε πάλι ένα μοναδικό αριθμό ταυτότητας για κάθε χρήστη, το όνομά του, το επώνυμο, την ηλεκτρονική του διεύθυνση αλλά και την κατάστασή του. Ο λογαριασμός ενός χρήστη μπορεί να βρίσκετε σε διάφορες καταστάσεις όπως μπορούμε να δούμε και στην Εικόνα 37 καταστάσεις όπου μπορεί να μεταβεί λόγο πχ. ανάρμοστης συμπεριφοράς του χρήστης (Blocked/Suspended) κα.

Ένα παράδειγμα αναζήτησης χρηστών μπορούμε να δούμε στην Εικόνα 35 όπου εισάγοντας την λέξη “gmail” στο πεδίο αναζήτησης και πατώντας το πλήκτρο “Enter” το σύστημα πραγματοποιεί αναζήτηση στα πεδία “First name”, “Last name”, “Email” και “Status” και μας επιστρέφει όλες τις εγγραφές που περιέχουν την συγκεκριμένη λέξη, ολόκληρη ή ως τμήμα τους.

Με τον ίδιο τρόπο, εάν θέλουμε να βρούμε όλους του χρήστες που έχουν Status Blocked τότε δεν έχουμε παρά να βάλουμε στην γραμμή αναζήτησης την λέξη blocked και θα πάρουμε ως αποτέλεσμα όλους του χρήστες που είναι BLOCKED.



Εικόνα 35: Filter/Search Users

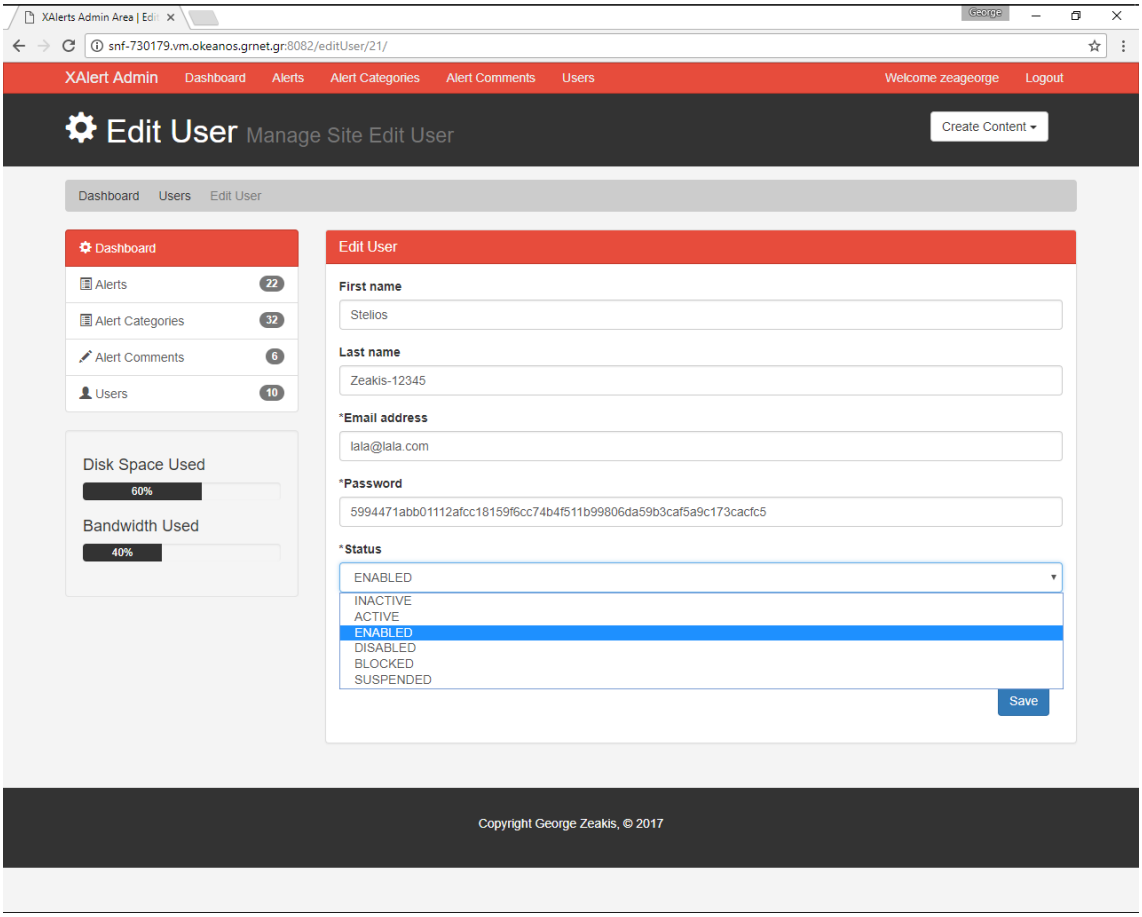


Εικόνα 36: Edit Users

Όσον αφορά την επεξεργασία ενός λογαριασμού χρήστη, βλέπουμε στην Εικόνα 36 την φόρμα από την οποία μπορούμε να αλλάξουμε το όνομα του χρήστη, το επώνυμο, την ηλεκτρονική του διεύθυνση, την κατάσταση του λογαριασμού του, τον κωδικό πρόσβασης και τέλος το κλειδί κρυπτογράφησης της επικοινωνίας του χρήστη με το σύστημα.

Ως κωδικό πρόσβασης, όπως βλέπουμε και στο παράδειγμα, δεν αποθηκεύουμε το κείμενο που μας έδωσε ο χρήστης αλλά την έξοδο από μια κρυπτογραφική συνάρτηση κατακερματισμού (34) (35), και συγκεκριμένα της SHA-256 (36) καθιστώντας έτσι την εύρεση του αρχικού κωδικού σχεδόν αδύνατη. Όσον αφορά το Encryption Key αυτό χρησιμοποιείται για την ψηφιακή υπογραφή του JSON Web token (JWT)(23) του κλειδιού (token) που χρησιμοποιείται για τις ανάγκες της Restful API τεχνικής (5).

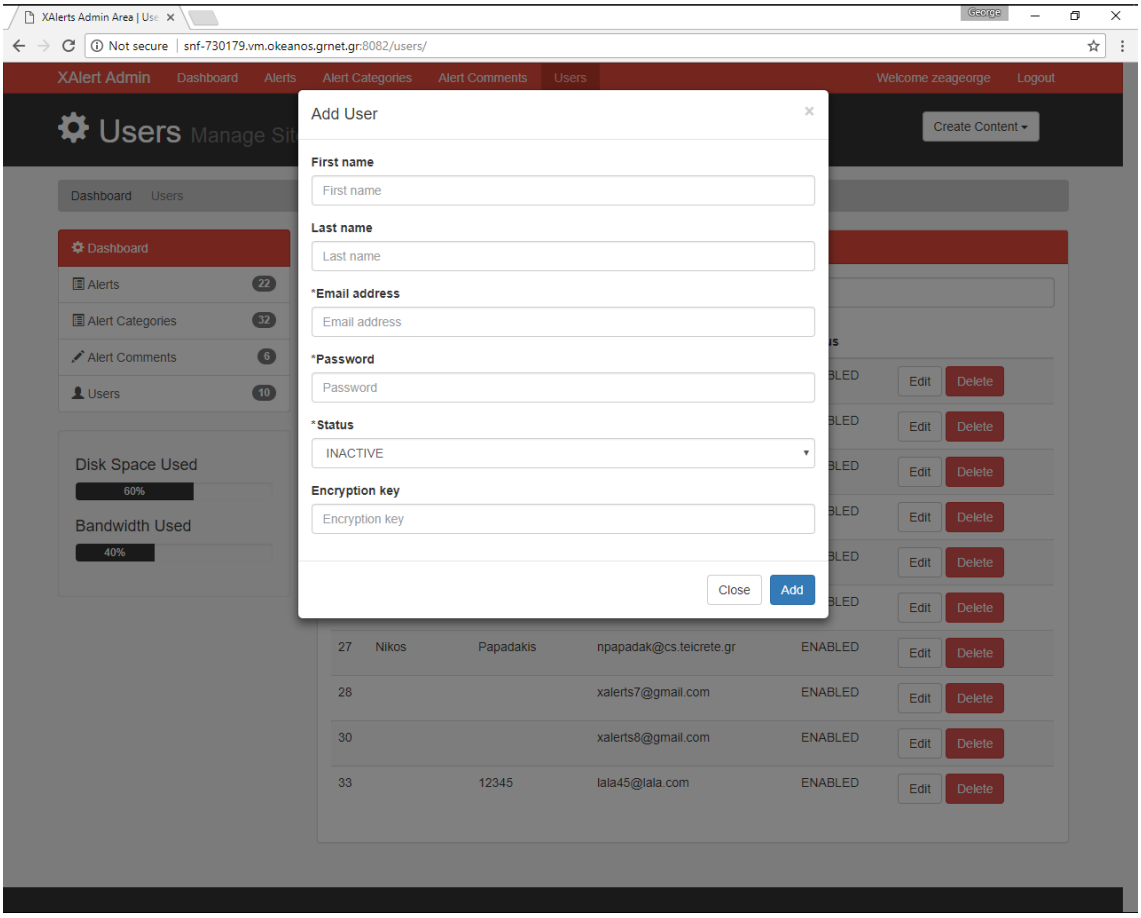
Με τις δύο προαναφερθείσες τεχνικές προσπαθήσαμε να κάνουμε το σύστημα όσο πιο ασφαλές γίνεται. Δηλαδή ακόμα και εάν κάποιος κακόβολους χρήστης ή σύστημα αποκτήσει πρόσβαση στην βάση δεδομένων να μην μπορεί να ανακτήσει τους κωδικούς πρόσβασης των χρηστών και τέλος να μην μπορεί κάποιος να αλλοιώσει την επικοινωνία ενός χρήστη με το σύστημα.



Εικόνα 37: Change User status

Όπως αναφέραμε και προηγουμένως, ο λογαριασμός ενός χρήστη μπορεί να βρίσκετε σε διάφορες καταστάσεις όπως:

- INACTIVE: ο χρήστης έχει κάνει εγγραφή στο σύστημα αλλά δεν έχει επιβεβαιώσει ακόμα την ηλεκτρονική του διεύθυνση
- ACTIVE: ο χρήστης έχει επιβεβαιώσει την ηλεκτρονική του διεύθυνση αλλά δεν έχει συνδεθεί ακόμα στο σύστημα
- ENABLED: από την στιγμή που ένας χρήστης έχει επιβεβαιώσει την ηλεκτρονική του διεύθυνση και συνδεθεί στο σύστημα την πρώτη φορά τότε η κατάσταση του λογαριασμού του μεταβαίνει στην τιμή Enabled.
- DISABLED: ο λογαριασμός έχει απενεργοποιηθεί για κάποιο λόγο
- BLOCKED: ο λογαριασμός έχει απενεργοποιηθεί προσωρινά για λόγους όπως η παραβίαση των όρων χρήσης, ανάρτηση ψευδών Ενημερώσεων ή/και σχολίων, παρενόχληση άλλων χρηστών ή/και συστημάτων, παραβίαση τοπικής νομοθεσίας, κα.
- SUSPENDED: ο λογαριασμός έχει απενεργοποιηθεί μόνιμα για κάποιο λόγο.



Εικόνα 38: Add new User account

Στην τελευταία εικόνα του συστήματος διαχείρισης [Εικόνα 38] βλέπουμε την φόρμα εισαγωγής νέου χρήστη όπου τα απαραίτητα πεδία προς συμπλήρωση είναι η ηλεκτρονική διεύθυνση, ο κωδικός πρόσβασης, η αρχική κατάσταση του λογαριασμού και το κλειδί κρυπτογράφησης για τους σκοπούς της ψηφιακής υπογραφής του JSON web token API key, όπως εξηγήσαμε σε προηγούμενο κεφάλαιο. Πατώντας το κουμπί “Close” η φόρμα εξαφανίζεται και ο χρήστης επιστρέφει στην προηγούμενη σελίδα που ήταν. Πατώντας όμως το κουμπί “Add” ο λογαριασμός του νέου χρήστη δημιουργείτε άμεσα και ανάλογα με την επιλεγείσα κατάσταση μπορεί ο χρήστης να συνδεθεί άμεσα.

Σε αυτό το σημείο τελειώνει η περιγραφή του Συστήματος Διαχείρισης και από τις εικόνες που είδαμε και περιγράψαμε μπορούμε να διακρίνουμε την ομοιογένεια των λειτουργιών, την ευκολία διαχείρισης περιεχομένου και χρηστών αλλά και την άμεση ενημέρωση του διαχειριστή αφού με μια ματιά στην αρχική σελίδα (Dashboard) μπορεί να δει όλα τα σημαντικά γεγονότα που λαμβάνουν χώρα στο σύστημά του.

3.3 Η εφαρμογή στο λειτουργικό σύστημα Android

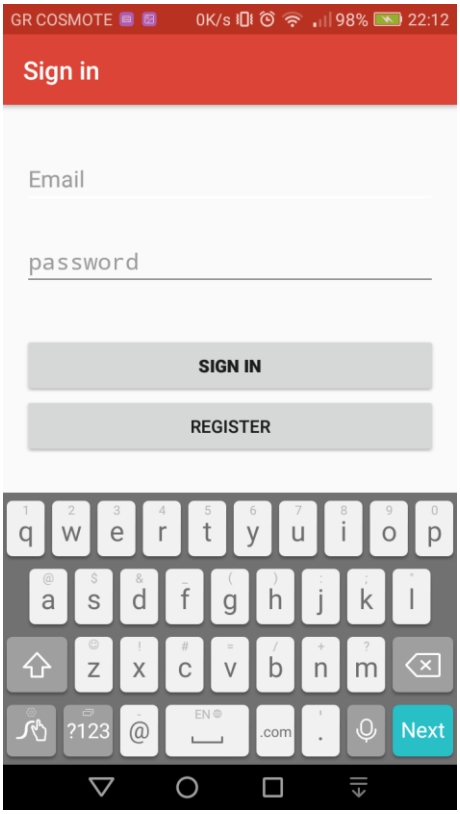
Στο κεφάλαιο αυτό θα δούμε αναλυτικά την εφαρμογή που αναπτύχθηκε για το λειτουργικό σύστημα Android (37) και θα παρουσιάσουμε την γραφική διεπαφή που χρησιμοποιεί ο τελικός χρήστης. Η εφαρμογή χρησιμοποιεί διάφορες δραστηριότητες (Activities) για κάθε λειτουργία, όπως η εμφάνιση των Ενημερώσεων σε λίστα αλλά και επάνω στον χάρτη, η επεξεργασία και η διαγραφή Ενημερώσεων, η εμφάνιση εικόνων, κα. Για την δημιουργία της εφαρμογής χρησιμοποιήθηκαν το ολοκληρωμένο περιβάλλον ανάπτυξης (integrated development environment, IDE) Android Studio (38) και το Material Design (39).

Εκτός των κλασικών βιβλιοθηκών για Android της Google (play-services, appcompat-v7, design, recyclerview, cardview-v7, gridlayout-v7, κα.), χρησιμοποιήθηκαν και οι παρακάτω βιβλιοθήκες τρίτων για τις διάφορες λειτουργίες της εφαρμογής:

- Android-EasyLocation (40) για τον εύκολο υπολογισμό και χρήση των πληροφοριών τοποθεσίας της συσκευής του χρήστη
- Google GSON (13) για μετατροπή των Java αντικειμένων σε σειριακή μορφή (κειμένου)
- Η βιβλιοθήκη ImagePicker (41) για την λειτουργία επιλογής φωτογραφιών και εικόνων
- Ενσωματωμένο ασύγχρονο πελάτη HTTP (42) για την επικοινωνία με τον διακομιστή
- το PhotoView (43) για την παρουσίαση των εικόνων των Ενημερώσεων και των σχολίων τους. Η συγκεκριμένη βιβλιοθήκη προσφέρει επίσης επιπλέον εργαλεία όπως η μεγέθυνση όλης ή μέρους της εικόνας.
- και τέλος το Java JWT: JSON Web Token for Java and Android (44) για την διαχείριση των κλειδιών των RESTful API κλήσεων προς τον διακομιστή.

3.3.1 Η γραφική διεπαφή

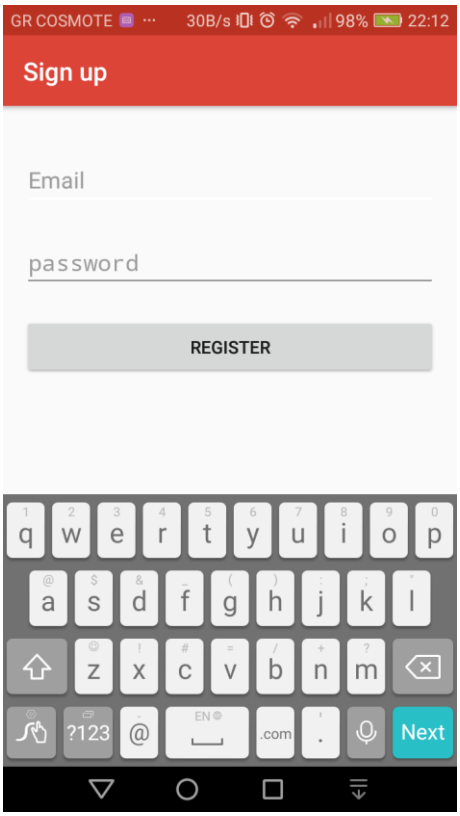
Η γραφική διεπαφή χρησιμοποιεί την βιβλιοθήκη Material Design της Google (45)(39) η οποία μα παρέχει έτοιμα συστατικά παραμετροποιημένα έτσι ώστε να ταιριάζουν όλα σε ένα τελικό προσεγμένο και αισθητικό αποτέλεσμα.



Εικόνα 39: Sign in Activity

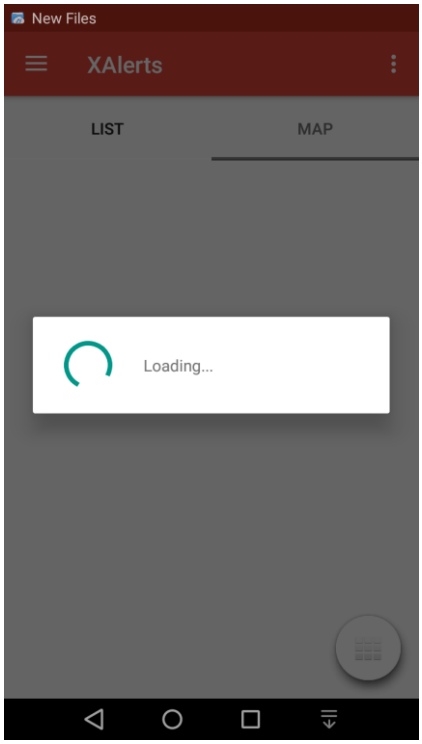
Ξεκινώντας την εφαρμογή μας δίνετε η δυνατότητα να εισέλθουμε στο σύστημα εισάγοντας την ηλεκτρονική μας διεύθυνση και τον κωδικό πρόσβασης [Εικόνα 39]. Από εδώ μπορούμε επίσης να επιλέξουμε να εγγραφούμε στο σύστημα, εάν δεν έχουμε κάποιο λογαριασμό επιλέγοντας το κουμπί “REGISTER”.

Είναι σημαντικό να αναφέρουμε ότι από την στιγμή που ένας χρήστης έχει συνδεθεί επιτυχώς στην εφαρμογή τότε κάθε φορά που την εκκινεί η εφαρμογή θα συνδέεται μόνη της χωρίς να χρειάζεται ο χρήστης να ξαναεισάγει τα στοιχεία του.



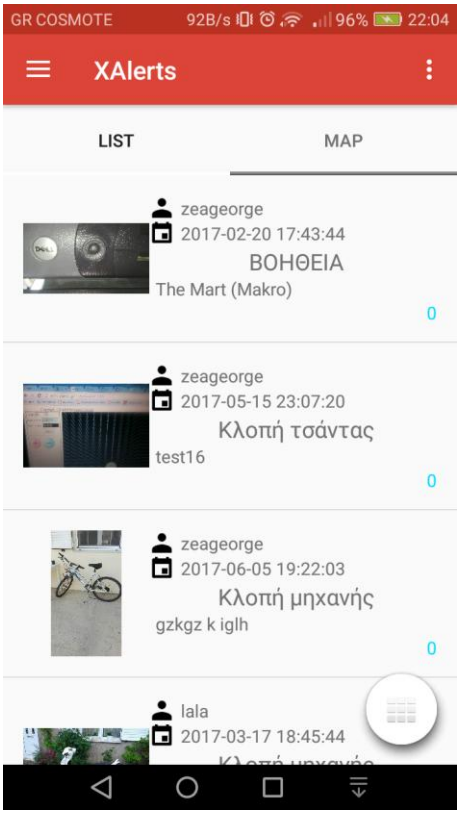
Εικόνα 40: Sign up Activity

Εάν επιλέξουμε “REGISTER” στην προηγούμενη εικόνα τότε μεταβαίνουμε στην δραστηριότητα “Sign in” [Εικόνα 40] από όπου μπορούμε να εγγραφούμε στο σύστημα εισάγοντας την ηλεκτρονική μας διεύθυνση και ένα κωδικό πρόσβασης



Εικόνα 41: Loading list of Alerts

Μετά την επιτυχή εισαγωγή στο σύστημα η εφαρμογή υπολογίζει τις γεωγραφικές συντεταγμένες από το υποσύστημα GPS (46) του κινητού μας τηλεφώνου και τις στέλνει στον διακομιστή. Ο διακομιστής με την σειρά του υπολογίζει και επιστρέφει στον χρήστη, όλα τα Alerts που υπάρχουν στην συγκεκριμένη περιοχή.



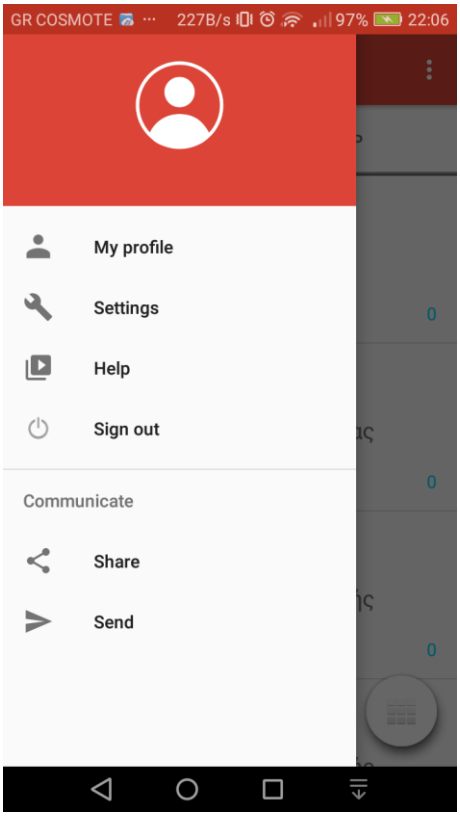
Εικόνα 42: List of Alerts

Μόλις η εφαρμογή παραλάβει τα δεδομένα τα παρουσιάζει στον παρουσιάζει στον χρήστη σε μορφή διαδραστικής λίστας [

Εικόνα 42]. Στην λίστα εμφανίζονται μια προς μια η κάθε ανακοίνωση με πληροφορίες όπως το όνομα χρήστη, η ημερομηνία και ώρα δημιουργίας, το όνομα της κατηγορίας στην οποία ανήκει η ανάρτηση κα τέλος ο τίτλος της.

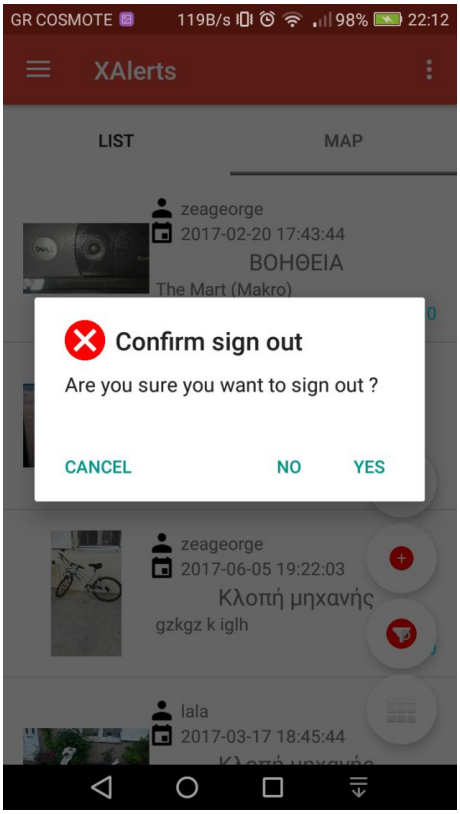
Όταν επιλέξουμε μια ανάρτηση τότε μεταβαίνουμε σε άλλη δραστηριότητα στην οποία μπορούμε να δούμε περισσότερες λεπτομέρειες για την επιλεγμένη ανάρτηση.

Τέλος στην παρούσα δραστηριότητα μπορούμε να επιλέξουμε τον χάρτη, τις ρυθμίσεις και τις επιλογές της εφαρμογής. Όλα αυτά θα τα δούμε αναλυτικά παρακάτω.



Εικόνα 43: Βασικές επιλογές εφαρμογής

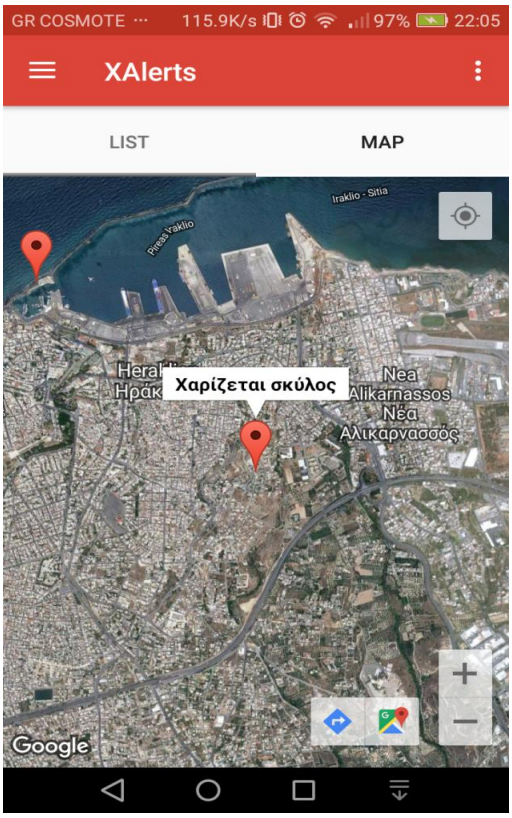
Στην Εικόνα 43 βλέπουμε τις βασικές επιλογές της εφαρμογής. Για τους σκοπούς της πτυχιακής εργασίας έχει υλοποιηθεί μόνο η επιλογή “Sign out” η οποία μας αποσυνδέει από το σύστημα.



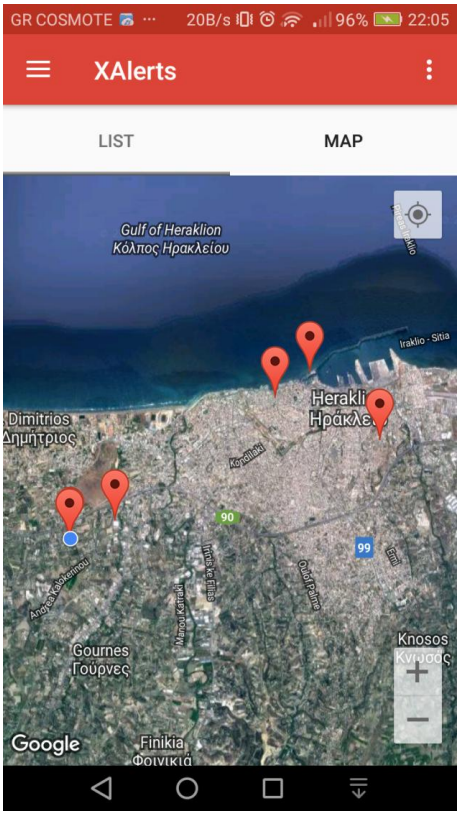
Εικόνα 44: Sign out

Για επιβεβαίωση εξόδου από την εφαρμογή το σύστημα μας ρωτάει με κατάλληλο μήνυμα καθώς εάν αποσυνδεθούμε τότε θα πρέπει να ξαναεισάγουμε τα στοιχεία εισόδου.

Όταν επιλέξουμε κάποιο σημάδι στην χάρτη τότε μας εμφανίζεται ο τίτλος της Ενημέρωσης που αφορά το σημάδι αυτό (Εικόνα 45). Έτσι μπορούμε να έχουμε μια γενική εικόνα των σημαντικών συμβάντων στην περιοχή που βρισκόμαστε.

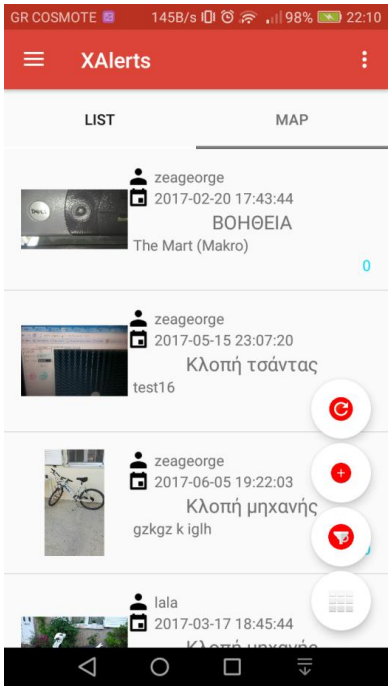


Εικόνα 45: Alerts πάνω στον χάρτη



Εικόνα 46: Google Maps

Στην Εικόνα 46 βλέπουμε στο δεύτερο TAB τον χάρτη όπου διακρίνονται οι τοποθεσίες των Ενημερώσεων. Κάθε κόκκινο σημάδι αντιστοιχεί σε ένα Alert και όταν το επιλέγουμε με το δάχτυλό μας τότε εμφανίζετε ο τίτλος του Alert (Εικόνα 45). Ο χάρτης που χρησιμοποιούμε είναι μια υπηρεσία του Google Maps (15) η οποία μας προσφέρει διάφορα έτοιμα εργαλεία και ευκολίες, μια από αυτές είναι το zoom in/out με το οποίο μπορούμε να πλησιάζουμε ή να απομακρυνόμαστε από την επιφάνεια της Γης. Με αυτό τον τρόπο μπορούμε να έχουμε μια γενική εικόνα των Ενημερώσεων στην περιοχή που βρισκόμαστε. Επίσης μας δίνετε η δυνατότητα να βρούμε την συντομότερη διαδρομή από το σημείο που βρισκόμαστε μέχρι κάποιο Alert που επιλέξουμε.



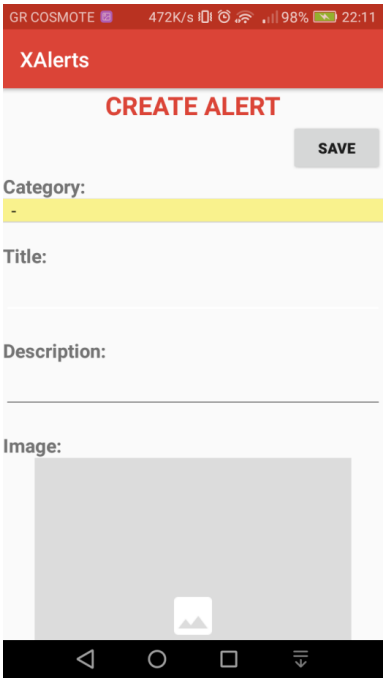
Εικόνα 47: Βασικές επιλογές της λίστας

Στην Εικόνα 47 βλέπουμε τις βασικές επιλογές της λίστας που έχει ο χρήστης, αυτές είναι η ανανέωση της λίστας, η δημιουργία ενός νέου Alert και το φιλτράρισμα των στοιχείων της λίστα με βάση την κατηγορίας στην οποία ανήκουν.

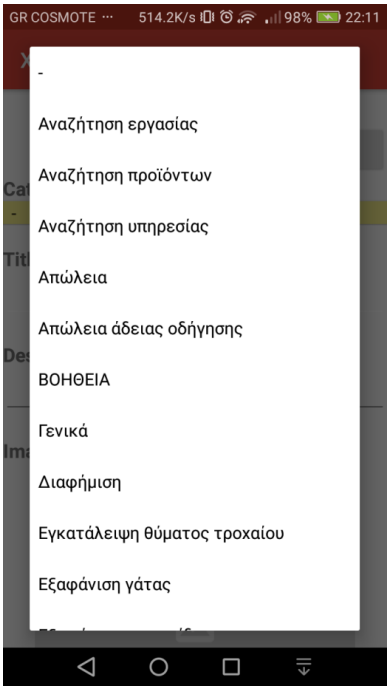
Η διεπαφή που επιλέχθηκε για το μενού αυτό είναι το Floating Action Button (FAB) (47), μια διαδραστική εικόνα την οποία αν πατήσουμε με το δάκτυλό μας τότε αναδύονται διάφορες άλλες διαδραστικές εικόνες όπου κάθε μια επιτελεί διαφορετική λειτουργία. Έτσι όταν ο χρήστης πατήσει στην Δημιουργία νέου Alert (η εικόνα με το σύμβολο «+») μεταβαίνει στην δραστηριότητα “Create Alert”, βλ. Εικόνα 48.

Στην Εικόνα 48 βλέπουμε την δραστηριότητα που αφορά την δημιουργία νέου Alert. Μας δίνετε η δυνατότητα να επιλέξουμε την κατηγορία της νέας Ενημέρωσης, να εισάγουμε τον τίτλο της, την περιγραφή της και τέλος να προσαρτήσουμε μια εικόνα σε αυτήν.

Για να επιλέξουμε εικόνα πρέπει να πατήσουμε με το δάκτυλό μας επάνω στην γκρι περιοχή που βρίσκετε κάτω από την ταμπέλα “Image:” Μόλις το κάνουμε αυτό τότε μας εμφανίζονται οι εικόνες που έχουμε αποθηκευμένες στην συσκευή μας (Εικόνα 57) από τις οποίες μπορούμε να επιλέξουμε μια για να την προσαρτήσουμε στην Ενημέρωσή μας.

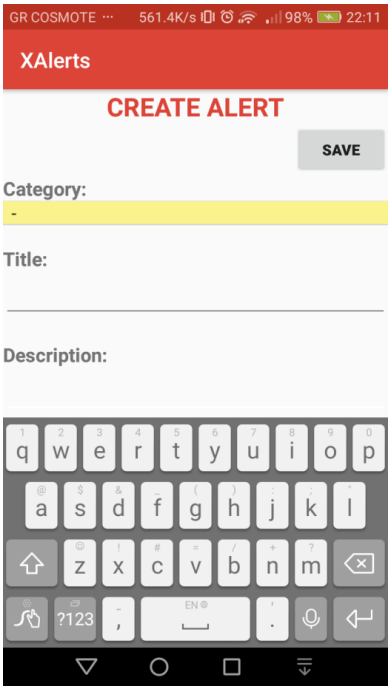


Εικόνα 48: Δημιουργία νέου Alert



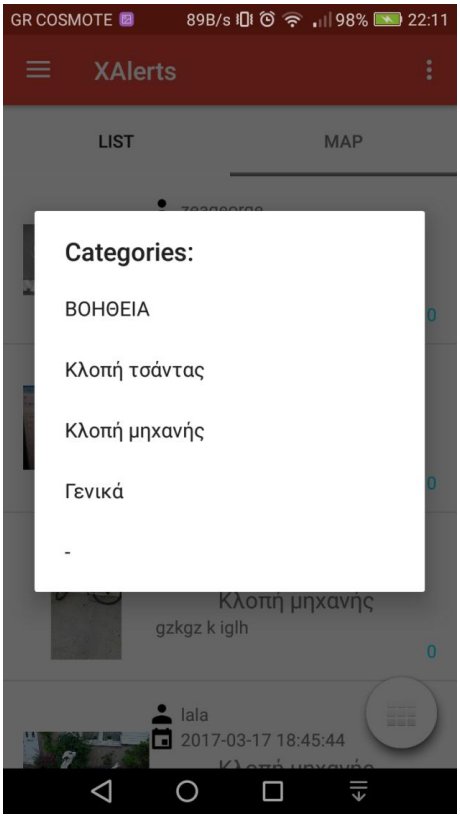
Εικόνα 49: Επιλογή κατηγορίας

Επίσης πατώντας επάνω στο κίτρινο πεδίο μας εμφανίζεται μια λίστα που περιέχει όλες τις κατηγορίες των Alerts έτσι ώστε να επιλέξουμε σε ποια κατηγορία θα αντιστοιχίσουμε το νέο μας Alert. Ανάλογα με την κατηγορία που θα επιλέξουμε θα ενημερωθούν αντίστοιχα οι άλλοι χρήστες στην περιοχή αφού κάθε κατηγορία έχει διαφορετικές ιδιότητες αρχικής/τελικής ακτίνας και ρυθμό μεταβολής (βλ. Εικόνα 27, CategoriesDescription).

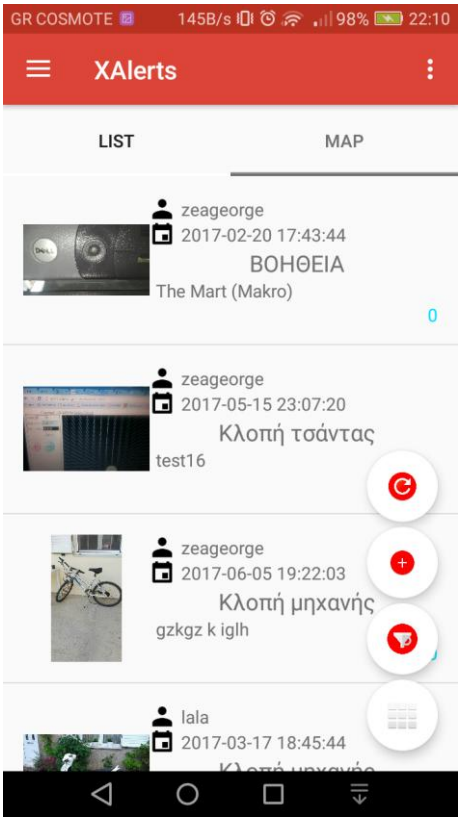


Εικόνα 50: Εισαγωγή Τίτλου και Περιγραφής

Στην συνέχεια και αφού έχουμε επιλέξει την κατηγορία, κάνουμε κλικ στο πεδίο “Title” όπου αυτόματα αναδύεται το εικονικό πληκτρολόγιο του λειτουργικού και συμπληρώνουμε τον τίτλο της Ενημέρωσης (Alert), κατόπιν συμπληρώνουμε με τον ίδιο τρόπο, την περιγραφή (πεδίο “Description”). Και τα δύο αυτά πεδία είναι πεδία εισαγωγής απλού κειμένου όπου ο Τίτλος μπορεί να έχει μέγεθος μέχρι 255 χαρακτήρες ενώ η περιγραφή δεν έχει περιορισμό στο πλήθος των χαρακτήρων.

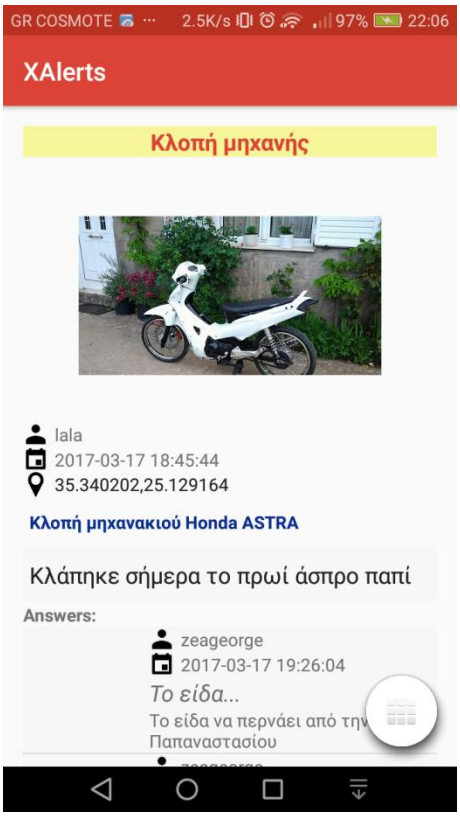


Εικόνα 51: Φιλτράρισμα ανά κατηγορία



Εικόνα 52: Επιλογές χρήστη

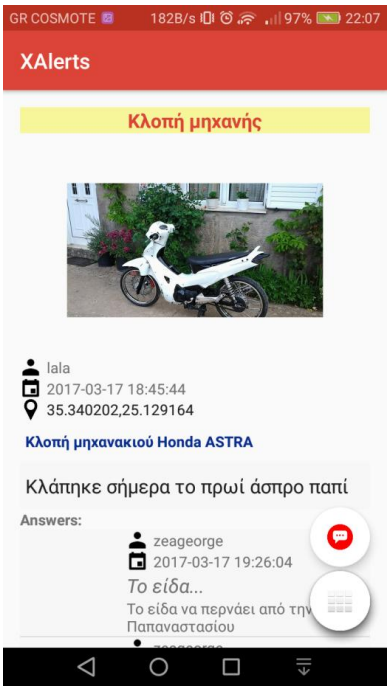
Στην Εικόνα 52 βλέπουμε ξανά τις επιλογές που έχει ο χρήστης όταν βρίσκετε στην λίστα των Ενημερώσεων (LIST Tab). Επιλέγοντας την τρίτη εικόνα, αυτή με το χωνί, μας εμφανίζεται μια λίστα με τα ονόματα των κατηγοριών από όπου μπορούμε να επιλέξουμε μια κατηγορία (Εικόνα 51). Επιλέγοντας μια κατηγορία το σύστημα φιλτράρει τα αποτελέσματα με βάση την επιλεγείσα κατηγορία και μας εμφανίζει τις Ενημερώσεις που ανήκουν σε αυτήν. Έτσι μπορούμε εύκολα να δούμε κατά προτεραιότητα τις Ενημερώσεις που μας ενδιαφέρουν.



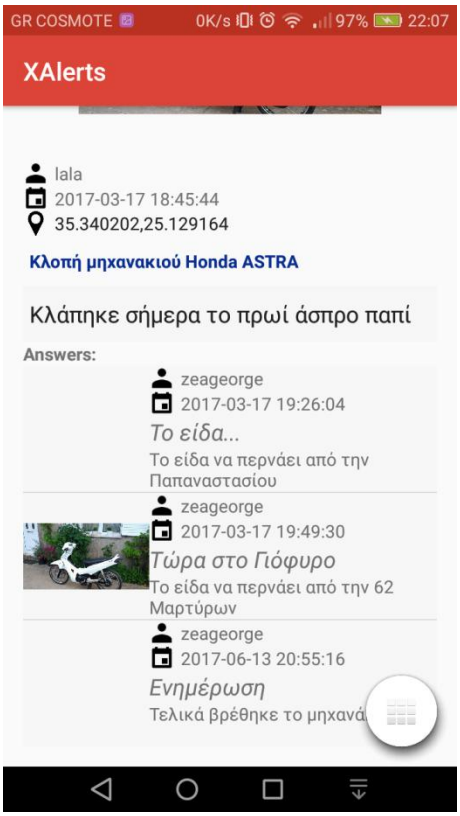
Εικόνα 53: Αναλυτικά στοιχεία ενός Alert

Πατώντας επάνω σε μια Ενημέρωση, στην λίστα, μεταφερόμαστε σε δραστηριότητα που αφορά τις λεπτομέρειες και ενέργειες της επιλεγείσας Ενημέρωσης (Εικόνα 53). Εδώ μπορούμε να δούμε την κατηγορία που ανήκει, την φωτογραφία που την αφορά, το όνομα χρήστη του δημιουργού της Ενημέρωσης, την ημερομηνία και ώρα της δημιουργίας της, τις συντεταγμένες (μήκος και πλάτος) όπου βρισκόταν η συσκευή την στιγμή της δημιουργίας της Ενημέρωσης, τον τίτλο, την περιγραφή και τέλος τα σχόλια που αφορούν την Ενημέρωση αυτή.

Επίσης, κάτω δεξιά βλέπουμε το γνώριμο κουμπί (FAB) το οποίο αλλάζει δυναμικά και παρουσιάζει μενού ανάλογα με την δραστηριότητα που βρίσκετε ο χρήστης αλλά και ανάλογα με τα δικαιώματα που έχει. Έτσι όπως μπορούμε να δούμε και στο παράδειγμα της Εικόνα 54, ο χρήστης της εφαρμογής δεν είναι ο δημιουργός της επιλεγείσας Ενημέρωσης και για αυτό του παρουσιάζεται μόνο μια επιλογή, αυτή τη δημιουργίας σχολίου στην Ενημέρωση αυτή. Όπως θα δούμε παρακάτω, σε περίπτωση που ο χρήστης είναι και δημιουργός μιας Ενημέρωσης τότε έχουμε περισσότερες επιλογές.

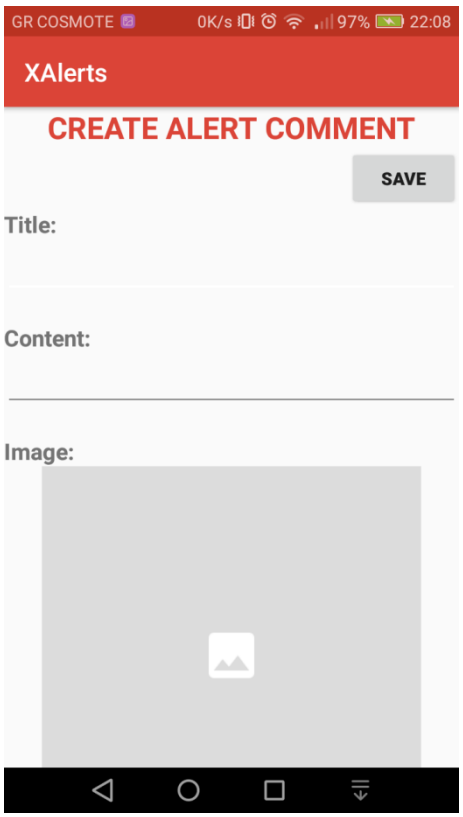


Εικόνα 54: Επιλογές απλού χρήστη

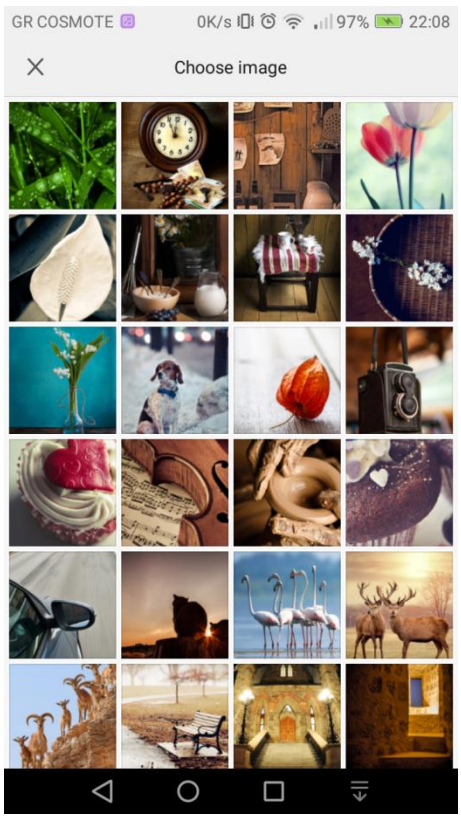


Εικόνα 55: Εμφάνιση σχολίων ενός Alert

Κάθε Ενημέρωση μπορεί να έχει σχόλια. Κάθε σχόλιο αφορά μια Ενημέρωση και μπορεί να είναι μια απάντηση ή οτιδήποτε άλλο, δημιουργόντας έτσι μια μορφή διαλόγου με θέμα την συγκεκριμένη Ενημέρωση. Τα σχόλια παρουσιάζονται σε μορφή λίστας κάτω από την περιγραφή της Ενημέρωσης. Κάθε στοιχείο της λίστας αυτής φέρει το όνομα χρήστη του δημιουργού του σχολίου, την ημερομηνία και ώρα της δημιουργίας του, ένα τίτλο, μια περιγραφή και τέλος μια εικόνα. Κάνοντας κλικ επάνω σε ένα σχόλιο μας εμφανίζεται η εικόνα του σε μεγέθυνση.

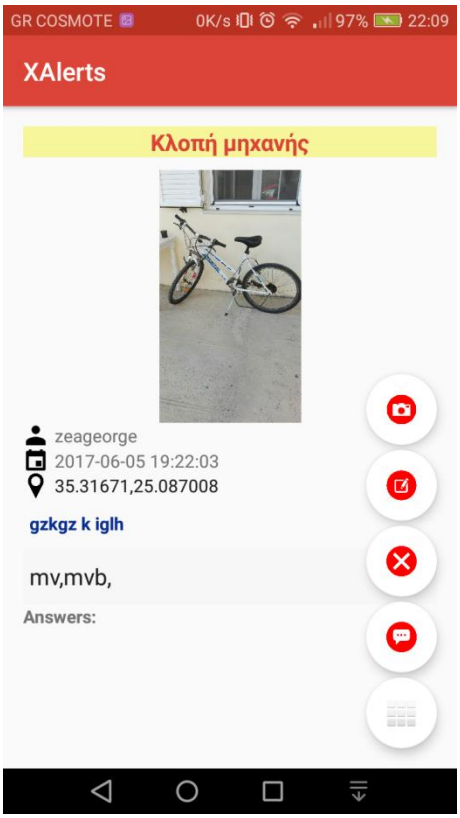


Εικόνα 56: Δημιουργία σχολίου



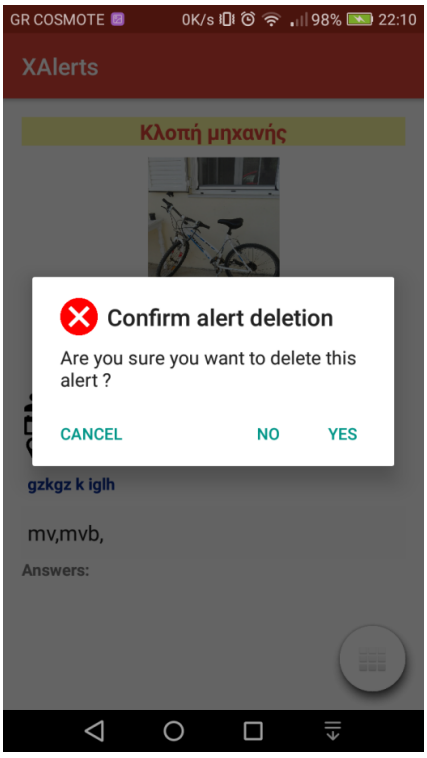
Εικόνα 57: Επιλογή εικόνας

Όσο βρισκόμαστε στην δραστηριότητα λεπτομερειών μιας Ενημέρωσης μπορούμε να δημιουργήσουμε ένα νέο σχόλιο. Έτσι πατώντας επάνω στο γκρι στρογγυλό κουμπί κάτω δεξιά, αναδύονται άλλα μενού, όπου ένα από αυτό, το συννεφάκι με τις τελίτσες, μας οδηγεί στην δραστηριότητα δημιουργίας νέου σχολίου (Εικόνα 56). Εδώ μας ζητείτε να εισάγουμε ένα τίτλο για το σχόλιο, την περιγραφή του και μια εικόνα (προαιρετικά) . Όσον αφορά την εικόνα, για την εισαγωγή της θα πρέπει να πατήσουμε επάνω στο γκρι τετράγωνο κάτω από την ταμπέλα “Image:”, τότε θα εμφανιστεί μια λίστα με τις αποθηκευμένες εικόνες που υπάρχουν στην συσκευή μας (Εικόνα 57). Από αυτές θα πρέπει να επιλέξουμε μια. Στην συνέχεια πατώντας το κουμπί “Save” η εφαρμογή ελέγχει τα στοιχεία που εισήγαγε ο χρήστης και κατόπιν τα αποστέλλει στον διακομιστή για αποθήκευση.

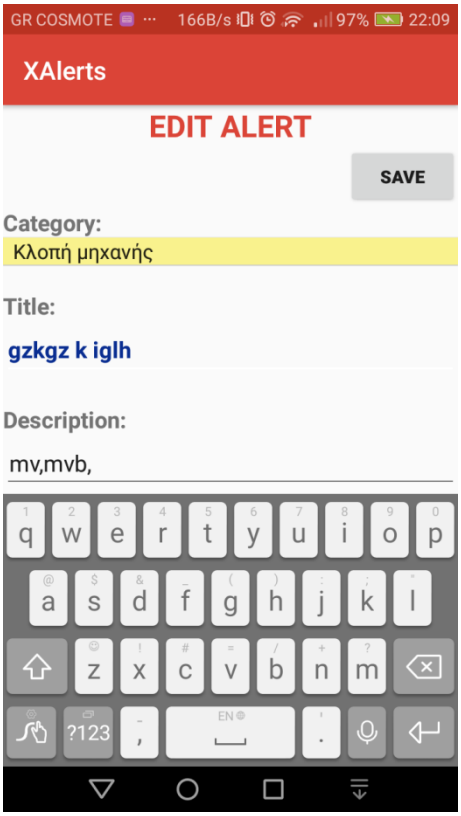


Εικόνα 58: Επιλογές ιδιοκτήτη ενός Alert

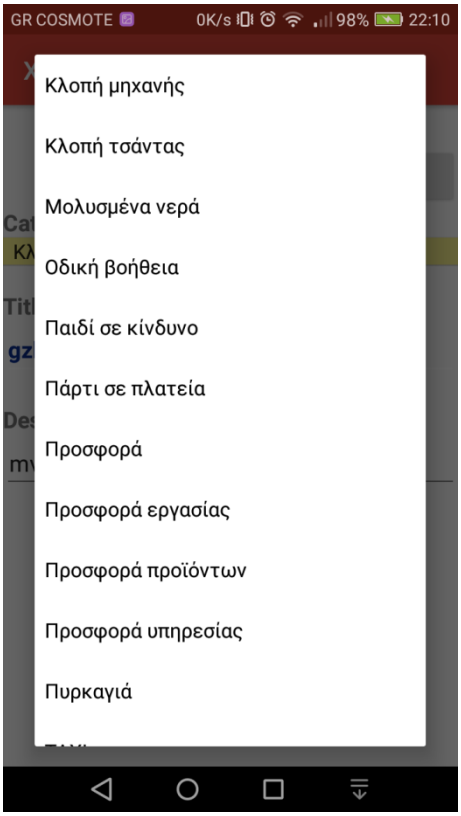
Όταν ο χρήστης της εφαρμογής είναι δημιουργός μιας Ενημέρωσης τότε έχει περισσότερες επιλογές όπως μπορούμε να δούμε στην Εικόνα 58, αυτές είναι με τη σειρά εμφάνισης: η προσθήκη εικόνας στην Ενημέρωση ή η αντικατάσταση μιας υπάρχουσας εικόνας, δηλαδή πατώντας το στρογγυλό κόκκινο εικονίδιο με την φωτογραφική μηχανή εμφανίζετε η λίστα με τις φωτογραφίες (Εικόνα 57) για να επιλέξουμε. Η δεύτερη επιλογή είναι η επεξεργασία της Ενημέρωσης που θα δούμε παρακάτω, η τρίτη επιλογή είναι η διαγραφή της Ενημέρωσης και όλων των σχολίων που έχει. Ποιο αναλυτικά πατώντας επάνω στο εικονίδιο με το "X" εμφανίζετε ένα παράθυρο διαλόγου όπως στην Εικόνα 59, όπου μας ζητείτε να επιβεβαιώσουμε την ενέργειά μας αυτή. Έτσι εάν επιλέξουμε "YES" στον διάλογο τότε θα διαγραφεί άμεσα η συγκεκριμένη Ενημέρωση καθώς επίσης και όλες οι φωτογραφίες της, τα σχόλιά της και οι φωτογραφίες των σχολίων της.



Εικόνα 59: Διαγραφή ενός Alert



Εικόνα 60: Επεξεργασία ενός Alert



Εικόνα 61: Επιλογή κατηγορίας

Όταν επιλέξουμε την επεξεργασία μιας Ενημέρωσης μεταβαίνουμε στην δραστηριότητα επαξεργασίας (Εικόνα 60) από όπου μπορούμε να αλλάξουμε την κατηγορία, τον τίτλο και την περιγραφή της Ενημέρωσης. Στην Εικόνα 61 βλέπουμε την λίστα με τις κατηγορίες που μπορεί να επιλέξει ο χρήστης σε περίπτωση που πατήσει επάνω στο κίτρινο πεδίο της δραστηριότητας “EDIT ALERT” με σκοπό την αλλαγή της κατηγορίας. Στην συνέχεια πατώντας το κουμπί “Save” η εφαρμογή στέλνει τις αλλαγές που έχουμε πραγματοποιήσει στον διακομιστή κι αυτός με την σειρά του ενημερώνει την Βάση δεδομένων.

3.4 Ο RESTful API server

Ο RESTful API web server υλοποιήθηκε με την γλώσσα προγραμματισμού Java. Πιο συγκεκριμένα χρησιμοποιήσαμε το Sparkjava web framework (26) για την υλοποίηση των υπηρεσιών διαδικτύου, το Freemarker template engine (12), την βιβλιοθήκη JCommander για την διαχείριση των αρχικών ρυθμίσεων του διακομιστή, τον mysql-connector-java (48) για την διασύνδεση της εφαρμογής με την βάση δεδομένων και τέλος την JWT (23) για την διαχείριση των API keys. Ο web server φιλοξενήθηκε στον Okeanos IAAS (32) και για την δημιουργία της εφαρμογής χρησιμοποιήθηκε το ολοκληρωμένο περιβάλλον ανάπτυξης (integrated development environment, IDE) NetBeans (49).

Στην Εικόνα 62 βλέπουμε τους πόρους (resources) που εξυπηρετεί ο διακομιστής. Πιο αναλυτικά, ο κάθε πόρος μπορεί να έχει ένα ή περισσότερους διαφορετικούς τρόπους κλήσης ανάλογα με το ρήμα του HTTP πρωτοκόλλου που χρησιμοποιούμε. Πχ. για τον πόρο /api/v1.0/alerts/{alertId}βλέπουμε ότι μπορούμε να το καλέσουμε χρησιμοποιώντας τα GET, PUT, DELETE και POST κάθε ένα από τα οποία επιτελεί μια ξεχωριστή λειτουργία σύμφωνα με το πρότυπο REST. Έτσι για να αιτηθούμε τα δεδομένα μιας συγκεκριμένης Ενημέρωσης θα χρησιμοποιήσουμε το εξής HTTP αίτημα:

```
GET /api/v1.0/alerts/2/ HTTP/1.1
Host: snf-730179.vm.okeanos.grnet.gr
Authorization: Bearer
eyJhbGciOiJIUzI1NiJ9.eeyJzdWIiOiIyMSIsIm5iZiI6MTQ4OTkyMzE3NiwiaXNzIjoiaHR0cDpcL1wvc25mLTczMDE3
OS52bS5va2Vhbm9zLmdybmV0LmdyXC9hcGlcL3YxLjBcLyIsImV4cCI6MTQ5MDAwOTU3NiwiaWF0IjoxNDg5OTIzMTc2LC
JqdGkiOiI3In0.xh7DXriug7zazD_WJyjinLVRg3N41BOM490bvETIO-ww
Cache-Control: no-cache
```

όπου στην πρώτη γραμμή μπορούμε να δούμε το ρήμα GET και στην θέση του {alertId} έχουμε βάλει τον μοναδικό αριθμό ταυτότητας της Ενημέρωσης που θέλουμε, δηλαδή το 2. Αντίστοιχα χρησιμοποιούμε το PUT για την επεξεργασία μιας Ενημέρωσης, το DELETE για την διαγραφή της και το POST για την δημιουργία μιας νέας Ενημέρωσης.

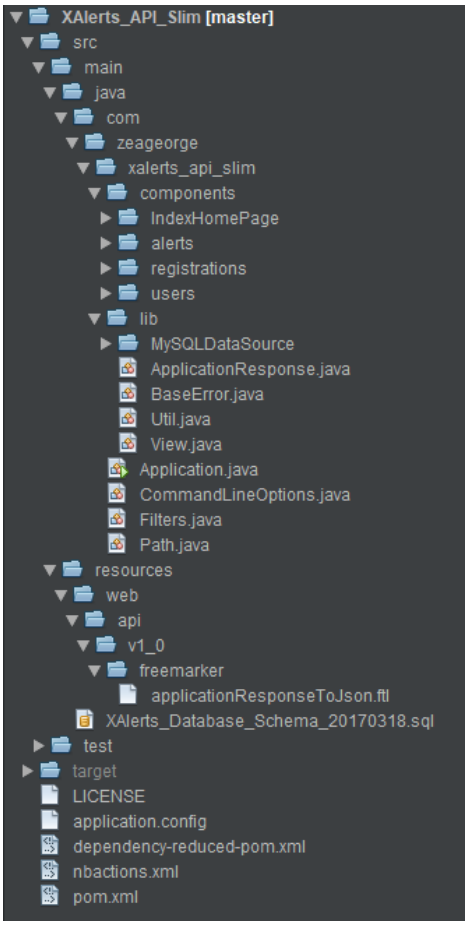
	HTTP METHODS									
Resource (Path/URL)	GET	PUT	PATCH	DELETE	POST	OPTIONS	CONNECT	TRACE	HEAD	Auth	Description
/api/v1.0/	X										Service main page
/api/v1.0/login/					X						Provide the user credentials to get the JWT
/api/v1.0/registrations/register/					X						Just a simple registration
/api/v1.0/alerts/?lat={lat}&lon={lon}&alt={altitude}/	X									X	Get the alerts in your area
/api/v1.0/alerts/{alertId}/	X									X	Get the alert with id={alertId}
/api/v1.0/alerts/{alertId}/		X								X	Change your alert with id={alertId}
/api/v1.0/alerts/{alertId}/				X						X	Delete your alert with id={alertId}
/api/v1.0/alerts/					X					X	Create a new alert.
/api/v1.0/alerts/{alertId}/images/	X									X	Get the images of the alert with id={alertId}
/api/v1.0/alerts/{alertId}/images/					X					X	Create or replace an existing image for the alert with id={alertId}
/api/v1.0/alerts/{alertId}/comments/	X									X	Get the list of comments for the alert with id={alertId}
/api/v1.0/alerts/{alertId}/comments/					X					X	Create a new comment for the alert with id={alertId}
/api/v1.0/alerts/{alertId}/comments/{cid}/images/	X									X	Get the images of the comment with id={cid} of the alert with id={alertId}
/api/v1.0/alerts/{alertId}/comments/{cid}/images/					X					X	Create an image for the comment with id={cid} of the alert with id={alertId}
/api/v1.0/alerts/categories/	X									X	Get all alert Categories

Εικόνα 62: API resources

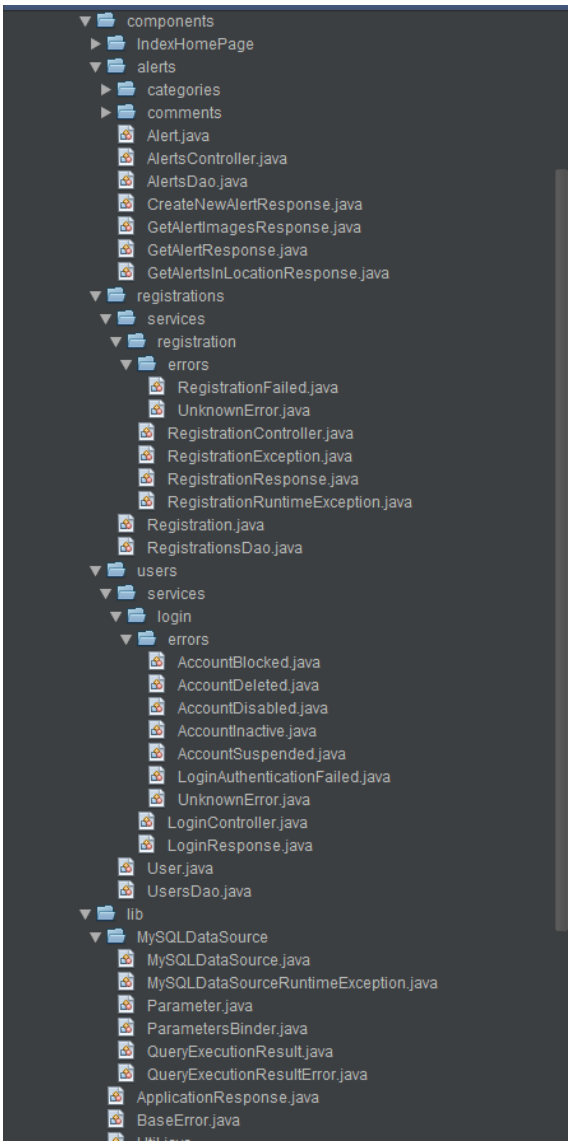
Έτσι μπορούμε να δούμε ότι δεν εκτελούν όλοι οι πόροι όλες τις λειτουργίες παρά μόνο αυτές που θέλουμε δίνοντας έτσι μια λογική στην εφαρμογή μας. Μπορούμε επίσης να διακρίνουμε τις κύριες οντότητες (μοντέλα, βλ. 2.3.1) που οδηγούν το σύστημα και σε ποιους πόρους πρέπει ο χρήστης να παρέχει τα διαπιστευτήριά του (τελευταία στήλη “Auth”), δηλαδή σε ποια αιτήματα πρέπει να στέλνουμε και το JWT μέσω του προκαθορισμένου (από το πρότυπο) πεδίου “Authorization: Bearer <JWT>” στο Header του HTTP request. Τέλος στην στήλη “Description” έχουμε μια μικρή περιγραφή για το τι κάνει ο κάθε πόρος. Ο παραπάνω πίνακας είναι ένα βοήθημα

του προγραμματιστή ώστε να έχει σε ένα σημείο την γενική εικόνα των υπηρεσιών που παρέχει ο διακομιστής του.

Όσον αφορά τον κώδικα της εφαρμογής, βλέπουμε στις εικόνες Εικόνα 63 και Εικόνα 64 την δομή των φακέλων η οποία έχει επίσης μεγάλη σημασία. Βλέπουμε λοιπόν ότι για κάθε υπηρεσία (πόρο) ξεχωριστά υπάρχει μια συγκεκριμένη δομή φακέλων ώστε να είναι εύκολα διαχειρίσιμη. Παράδειγμα για τους χρήστες, στον φάκελο users υπάρχουν τα αρχεία που αφορούν το μοντέλο του χρήστη, δηλαδή το αρχείο “User.java” καθώς και το“UsersDao.java”. Το “User.java” περιέχει τις πληροφορίες και τις λειτουργίες που αφορούν τους χρήστες και το “UsersDao.java” περιέχει όλες τις μεθόδους που αφορούν τις λειτουργίες επικοινωνίας με την Βάση Δεδομένων. Τα αρχεία αυτά βρίσκονται στην ρίζα του component “users” διότι χρησιμοποιούνται από άλλα components ή/και από υπηρεσίες του ιδίου συστατικού.



Εικόνα 63: Δομή φακέλων εφαρμογής (α)



Εικόνα 64: Δομή φακέλων εφαρμογής (β)

Για τις ανάγκες της Εργασίας αναπτύξαμε μια αρκετά καλή βιβλιοθήκη για επικοινωνία της εφαρμογής με την Βάση Δεδομένων MySQL, την “MySQLDataSource” η οποία μας επιτρέπει να χρησιμοποιούμε ονομαστικές μεταβλητές για κράτηση θέσης στα ερωτήματα SQL και όχι απλά ερωτηματικά “?”. Με αυτό τον τρόπο ο κώδικας SQL γίνεται πιο εύκολος στο διάβασμα και κατανοητός.

Παράδειγμα αντί να γράψουμε:

```
PreparedStatement p = con.prepareStatement("select * from people where (first_name = ? or
last_name = ?) and address = ?");
int i = 1;
p.setString(i++, name);
p.setString(i++, name);
p.setString(i++, address);
```

μπορούμε να γράψουμε:

```
String query = "select * from people where (first_name = :name or last_name= :name) and
address = :address");
NamedParameterStatement p = new NamedParameterStatement(con, query);
p.setString("name", name);
p.setString("address", address);
```

το οποίο είναι πολύ πιο κατανοητό.

Τέλος, στην Εικόνα 65 βλέπουμε ένα απόσπασμα από την κύρια μέθοδο εξυπηρέτησης πόρων του συστήματος από όπου μπορούμε να διακρίνουμε πόσο απλό είναι και με πόσο λίγο κώδικα, κάνοντας χρήση των μεθόδων του Sparkjava framework, μπορούμε να ορίσουμε τις παρεχόμενες υπηρεσίες και σε ποιες χρειαζόμαστε πιστοποίηση του χρήστη. Ο συγκεκριμένος τρόπος γραφής, πχ. στην γραμμή 191, είναι το lambda expressions (28). Για την πλήρη λίστα του κώδικα μπορούμε να δούμε το Παράρτημα Γ.2 – Απόσπασμα πηγαίου κώδικα της κύριας μεθόδου του Restful API server ή τα αρχεία πηγαίου κώδικα που συνοδεύουν την Εργασία. Επίσης μαζί με την Εργασία έχουμε επισυνάψει και το Διάγραμμα κλάσεων (Class Diagram) (είναι αρκετά μεγάλο για να χωρέσει εδώ !)

```
183    public void run() {
184      before("*", Filters.init);
185      before("*", Filters.addTrailingSlashes);
186      before("*", Filters.validateAccessToken);
187
188      post(Path.Web.API.V1_0.LOGIN, LoginController.login);
189      post(Path.Web.API.V1_0.REGISTER, RegistrationController.register);
190
191      get(Path.Web.API.V1_0.GET_ALERT_CATEGORIES, (Request request, Response response) -> {
192        authorize(response);
193        return CategoriesController.getAlertCategories.handle(request, response);
194      });
195
196      get(Path.Web.API.V1_0.GET_ALERTS_IN_LOCATION, (Request request, Response response) -> {
197        authorize(response);
198        return AlertsController.getAlertsInLocation.handle(request, response);
199      });
200
201      get(Path.Web.API.V1_0.GET_ALERT, (Request request, Response response) -> {
202        authorize(response);
203        return AlertsController.getAlert.handle(request, response);
204      });
205
206      post(Path.Web.API.V1_0.CREATE_NEW_ALERT, (Request request, Response response) -> {
207        authorize(response);
208        return AlertsController.createNewAlert.handle(request, response);
209      });
210
211      delete(Path.Web.API.V1_0.DELETE_ALERT, (Request request, Response response) -> {
212        authorize(response);
213        return AlertsController.deleteAlert.handle(request, response);
214      });
215
216  //    patch(Path.Web.API.V1_0.EDIT_ALERT, (Request request, Response response) -> {
217  //      authorize(response);
218  //      return AlertsController.editAlert.handle(request, response);
219  //    });
220
221      put(Path.Web.API.V1_0.EDIT_ALERT, (Request request, Response response) -> {
222        authorize(response);
223        return AlertsController.editAlert.handle(request, response);
224      });
225
226      get(Path.Web.API.V1_0.GET_ALERT_IMAGES, (Request request, Response response) -> {
227        authorize(response);
228        return AlertsController.getAlertImages.handle(request, response);
229      });
```

Εικόνα 65: Απόσπασμα πηγαίου κώδικα της κύριας μεθόδου

3.5 Η Βάση δεδομένων

Η Βάση δεδομένων που χρησιμοποιούμε για την εφαρμογή είναι η σχεσιακή βάση MySQL (9) και για την διασύνδεση της Java με την βάση χρησιμοποιήσαμε την βιβλιοθήκη MySQL Connector /J (48) η οποία μας επιτρέπει να αλληλεπιδρούμε με την βάση χρησιμοποιώντας ελάχιστο κώδικα παρέχοντας μας όμως ταυτόχρονα πολλές δυνατότητες.

3.5.1 Η βιβλιοθήκη MySQLDataSource

Όπως αναφέραμε στο προηγούμενο υποκεφάλαιο (3.4 Ο RESTful API server), για τις ανάγκες της Εργασίας αναπτύξαμε μια βιβλιοθήκη για επικοινωνία της εφαρμογής με την Βάση Δεδομένων MySQL, την "MySQLDataSource" που ενσωματώνει αρκετές από τις λειτουργίες της βιβλιοθήκης MySQL Connector /J παρέχοντας στον προγραμματιστή ένα σύνολο από μεθόδους που απλοποιούν την διαδικασία του προγραμματισμού. Στην Εικόνα 66 βλέπουμε την μέθοδο createNewUser της κλάσης UsersDao.java (DAO, Data Access Object (50)). Μπορούμε να δούμε πόσο εύκολο είναι να αλληλεπιδράσουμε με την βάση μας, στην γραμμή 254 απλά «ανοίγουμε» ένα κανάλι επικοινωνίας, στην γραμμή 256 καλούμε την μέθοδο "insertSingle" της βιβλιοθήκης μας "MySQLDataSource" για να εισάγουμε μια εγγραφή του τύπου "User" στην βάση και στην γραμμή 268 κλείνουμε το κανάλι επικοινωνίας.

Συγκριτικά με τον κλασικό τρόπο γραφής μιας παρόμοιας μεθόδου, η δική μας βιβλιοθήκη παρέχει πιο καθαρό κώδικα που μπορεί να συντηρηθεί και να επεκταθεί πολύ εύκολα. Τέλος, η βιβλιοθήκη μας είναι γραμμένη με τέτοιο τρόπο ώστε να αποτρέπει τον προγραμματιστή από τα συνηθισμένα λάθη και είναι πιο ασφαλής και γρήγορη αφού χρησιμοποιεί την τεχνική των έτοιμων δηλώσεων (Prepared statement) (51).

```
245
246     /**
247      * Creates a new User
248      *
249      * @param user A com.zeageorge.xalerts_api_slim.components.users.User class
250      *
251      * @return boolean true on success or false on failure
252      */
253     public static boolean createNewUser(final User user) {
254         DATASOURCE.open();
255
256         QueryExecutionResult insertResult = DATASOURCE.insertSingle(
257             TABLE_NAME,
258             new Parameter(EMAIL_ADDRESS.getValue(), user.getEmailAddress(), String.class),
259             new Parameter(PASSWORD.getValue(), user.getPassword(), String.class),
260             new Parameter(STATUS.getValue(), user.getStatus().getValue(), Integer.class),
261             new Parameter(WEIGHT.getValue(), user.getWeight(), Integer.class),
262             new Parameter(ENCRYPTION_KEY.getValue(), user.getEncryptionKey(), String.class),
263             new Parameter(FIRST_NAME.getValue(), user.getFirstName(), String.class),
264             new Parameter(LAST_NAME.getValue(), user.getLastName(), String.class),
265             new Parameter(RADIUS.getValue(), user.getRadius(), String.class)
266         );
267
268         DATASOURCE.close();
269
270         return insertResult.getNumberOfAffectedRows() > 0;
271     }
```

Εικόνα 66: Η μέθοδος createNewUser της κλάσης UsersDao.java

3.5.2 Η εμβέλεια των Ενημερώσεων στην SQL

Όπως αναλύσαμε στο υποκεφάλαιο 2.2 Υπολογισμός αποστάσεων εμβέλειας ενημερώσεων για να μπορέσουμε να υπολογίσουμε την εμβέλεια της κάθε Ενημέρωσης και να βρούμε εάν ένας χρήστης βρίσκετε μέσα σε αυτήν ώστε να την λάβει, χρησιμοποιήσαμε τον τύπο Haversine. Επειδή όμως ενδέχεται να έχουμε αρκετές χιλιάδες ενεργές Ενημερώσεις και ο υπολογισμός τους είναι χρονοβόρος, αναθέσαμε την διαδικασία αυτή απ' ευθείας στην βάση δεδομένων. Έτσι έπρεπε να μεταφράσουμε τον τύπο Haversine σε SQL γλώσσα. Στην Εικόνα 67 βλέπουμε την ολοκληρωμένη δήλωση όπου για τις μεταβλητές ισχύουν τα παρακάτω:

- :kmPerDegree = 111.045 Km (κατά προσέγγιση)
- :lat = γεωγραφικό πλάτος του χρήστη
- :lon = γεωγραφικό μήκος του χρήστη
- latitude = γεωγραφικό πλάτος της Ενημέρωσης
- longitude = γεωγραφικό μήκος της Ενημέρωσης

```
1. SELECT
2.  gzea_alerts.*,
3.  c.name AS categoryName,
4.  (c.initialRadius + (TIMESTAMPDIFF(MINUTE,eventDate,NOW())*c.rateOfChangePerMinute)) AS er,
6.  IF ((SELECT er) > c.maxRadius, c.maxRadius, (SELECT er)) AS eventRadius,
7.    (:kmPerDegree*DEGREES(ACOS(COS(RADIANS(:lat)) * COS(RADIANS(latitude))
9.    * COS(RADIANS(:lon) - RADIANS(longitude)) + SIN(RADIANS(:lat))
11.   * SIN(RADIANS(latitude)))))*1000 AS distance_in_m
12. FROM gzea_alerts
13. JOIN gzea_alert_categories c ON c.id=categoryId
14. WHERE status IN (1) -- Alert is in active state
15. HAVING distance_in_m <= eventRadius
16. ORDER BY c.id DESC, distance_in_m ASC
17. LIMIT 15;
```

Εικόνα 67: SQL δήλωση για τον υπολογισμό της εμβέλειας

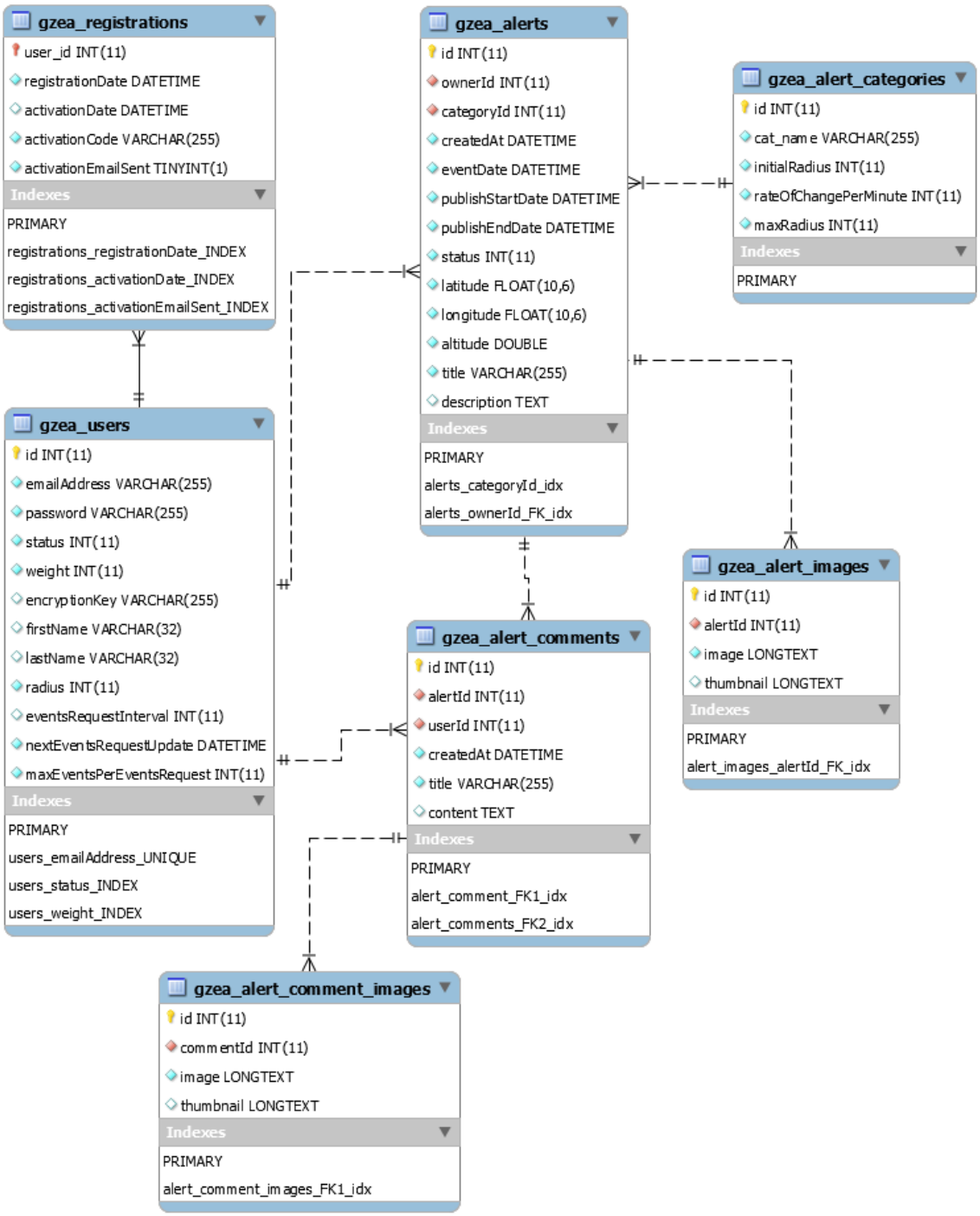
Αυτό που κάνει η παραπάνω δήλωση είναι: για κάθε Ενημέρωση που έχουμε καταχωρημένη στην βάση μας, βρες την τρέχουσα (χρονικά) εμβέλεια της. Εάν η απόσταση από το γεωγραφικό σημείο του χρήστη μέχρι το σημείο της Ενημέρωσης είναι μικρότερο από την τρέχουσα εμβέλεια (having distance_in_m <= eventRadius) τότε ο χρήστης βρίσκετε μέσα στη τρέχουσα εμβέλεια κι έτσι πρέπει να συμπεριληφθεί η Ενημέρωση στα αποτελέσματα. Επίσης πρέπει η Ενημέρωση να είναι ενεργή (where status in (1)). Τέλος ταξινομεί τα αποτελέσματα ανά κρισιμότητα (ταυτότητα της κατηγορίας που ανήκει η Ενημέρωση) και ανά απόσταση από την Ενημέρωση (τις πιο κοντινές Ενημερώσεις), βλ. ORDER BY, και μας επιστρέφει 15 εγγραφές (limit 15). Τις τιμές για το υπολογισμό της τρέχουσας εμβέλειας τις λαμβάνουμε από την κατηγορία στην οποία ανήκει η Ενημέρωση (βλ. initialRadius, maxRadius και rateOfChangePerMinute).

Τέλος να αναφέρουμε ότι τις καλύτερες πηγές που βρήκαμε στο διαδίκτυο για το θέμα αυτό είναι οι

- Geo (proximity) Search with MySQL (52) του Alexander Rubin
- Fast nearest-location finder for SQL (53) του Ollie Jones

3.5.3 Το σχήμα της βάσης

Στην Εικόνα 68 βλέπουμε το Ενισχυμένη μοντέλο οντοτήτων-συσχετίσεων (Enhanced entity–relationship model, EER) (54) όπου διακρίνουμε τους πίνακες με τα πεδία τους, τις σχέσεις μεταξύ τους καθώς και τα «κλειδιά» του κάθε πίνακα. Το "gzea_" που έχουμε βάλει πριν το όνομα του κάθε πίνακα είναι το prefix που προσθέτει περισσότερη ασφάλεια στην βάση μας ώστε να μην μπορεί κάποιος κακόβουλος να μαντέψει τα ονόματα των πινάκων σε περίπτωση επίθεσης "SQL injection".



Εικόνα 68: Database EER Diagram

Έχουμε επτά πίνακες:

1. registrations[1]: ο πίνακας αυτός περιέχει 5 πεδία:
    1. user_id[1]: το μοναδικό id του νέου χρήστη
    2. registrationDate[1]: την ημερομηνία δηλαδή που εγγράφηκε ο χρήστης στο σύστημα
    3. activationDate[1]: η ημερομηνία στην οποία ο χρήστης ενεργοποίησε τον λογαριασμό του (βλ. activationCode παρακάτω).
    4. activationCode[1]: όταν ένας χρήστης εγγράφεται στο σύστημα ο λογαριασμός του δεν ενεργοποιείται άμεσα αλλά του στέλνουμε ένα email στο οποίο αναφέρουμε ένα μοναδικό κωδικό ενεργοποίησης που πρέπει να εισάγει ο χρήστης ώστε να ενεργοποιηθεί ο λογαριασμός του.
    5. activationEmailSent[1]: εδώ καταχωρούμε εάν το σύστημα έστειλε επιτυχώς το email ενεργοποίησης του λογαριασμού
2. users: ο πίνακας αυτός περιέχει 12 πεδία:
    1. id: το μοναδικό id του χρήστη
    2. emailAddress: η ηλεκτρονική διεύθυνση του χρήστη
    3. password: ο κωδικός πρόσβασης. Εδώ αποθηκεύουμε την έξοδο από συνάρτηση κατακερματισμού και όχι το plain text.
    4. status: η κατάσταση του λογαριασμού του χρήστη. Αν είναι δηλαδή ενεργός ο λογαριασμός, ανενεργός, μπλοκαρισμένος, κα.
    5. weight[1]: πεδίο για μελλοντική δυνατότητα-επέκταση
    6. encryptionKey: ένα μοναδικό κλειδί για κάθε χρήστη που χρησιμοποιείται για την ψηφιακή υπογραφή του JWT.
    7. firstName: το όνομα του χρήστη
    8. lastName: το επώνυμο του χρήστη
    9. radius[1]: πεδίο για μελλοντική δυνατότητα-επέκταση
    10. eventsRequestInterval[1]: πεδίο για μελλοντική δυνατότητα-επέκταση
    11. nextEventsRequestUpdate[1]: πεδίο για μελλοντική δυνατότητα-επέκταση
    12. maxEventsPerEventsRequest[1]: πεδίο για μελλοντική δυνατότητα-επέκταση
3. alerts: ο πίνακας αυτός περιέχει 13 πεδία:
    1. id: το μοναδικό id της Ενημέρωσης
    2. ownerId: ο αριθμός ταυτότητας του χρήστη που δημιούργησε την Ενημέρωση
    3. categoryId: ο αριθμός ταυτότητας της κατηγορίας στην οποία ανήκει η Ενημέρωση
    4. createdAt: η ημερομηνία δημιουργίας της ενημέρωσης
    5. eventDate[1]: η ημερομηνία που συνέβη αυτό που περιγράφουμε στην Ενημέρωση. Υπάρχει περίπτωση να αναρτήσουμε μια Ενημέρωση για κάτι που συνέβη στο παρελθόν.
    6. publishStartDate[1]: η ημερομηνία έκδοσης της ενημέρωσης. Πχ. μπορούμε να δημιουργήσουμε μια Ενημέρωση αλλά θέλουμε να γίνει γνωστή στους χρήστες μετά από Χ ημέρες.
    7. publishEndDate[1]: η ημερομηνία που θα σταματήσει το σύστημα να γνωστοποιεί την Ενημέρωση.
    8. status: η κατάσταση της ενημέρωσης: Ενεργή, ανενεργή, κα.
    9. latitude: το γεωγραφικό πλάτος της συσκευής από την οποία δημιουργήθηκε η Ενημέρωση.
    10. longitude: το γεωγραφικό μήκος της συσκευής από την οποία δημιουργήθηκε η Ενημέρωση.
    11. altitude[1]: το γεωγραφικό ύψος της συσκευής από την οποία δημιουργήθηκε η Ενημέρωση.
    12. title: ο Τίτλος της Ενημέρωσης
    13. description: Η περιγραφή, το κείμενο της Ενημέρωσης
4. alert_categories: ο πίνακας αυτός περιέχει 5 πεδία:
    1. id: το μοναδικό id της Κατηγορίας της Ενημέρωσης
    2. cat_name: το όνομα της κατηγορίας
    3. initialRadius: η αρχική ακτίνα εμβέλειας ενημέρωσης των χρηστών

    4. rateOfChangePerMinute: ο ρυθμός μεταβολής συναρτήσει του χρόνου (σε λεπτά της ώρας). Δηλαδή πόσο θα αυξάνει η εμβέλεια της ενημέρωσης ανά λεπτό της ώρας που θα περνάει.
    5. maxRadius: η μέγιστη απόσταση της εμβέλειας της Ενημέρωσης.
5. alert_comments: ο πίνακας αυτός περιέχει 6 πεδία:
    1. id, το μοναδικό id του Σχόλιου της Ενημέρωσης
    2. alertId: ο μοναδικός αριθμός ταυτότητας της Ενημέρωσης στην οποία ανήκει το Σχόλιο
    3. userId: ο αριθμός ταυτότητας του δημιουργού του Σχόλιου
    4. createdAt: η ημερομηνία δημιουργίας του Σχόλιου
    5. title: ο τίτλος του
    6. content: το περιεχόμενο του Σχόλιου
6. alert_images: ο πίνακας αυτός περιέχει 4 πεδία:
    1. id: το μοναδικό id της Εικόνας της Ενημέρωσης
    2. alertId: ο μοναδικός αριθμός ταυτότητας της Ενημέρωσης στην οποία ανήκει η Εικόνα.
    3. image: τα δεδομένα της Εικόνας σε μορφή Base64 Encoding
    4. thumbnail[1]: τα δεδομένα της Εικόνας σε μορφή Base64 Encoding αλλά σε μικρότερη διάσταση.
7. alert_comment_images: ο πίνακας αυτός περιέχει 4 πεδία:
    1. id: το μοναδικό id της Εικόνας του Σχόλιου της Ενημέρωσης
    2. commentId: ο μοναδικός αριθμός ταυτότητας του Σχόλιου της Ενημέρωσης στο οποία ανήκει η Εικόνα.
    3. image: τα δεδομένα της Εικόνας σε μορφή Base64 Encoding
    4. thumbnail[1]: τα δεδομένα της Εικόνας σε μορφή Base64 Encoding αλλά σε μικρότερη διάσταση.

Τέλος στο Παράρτημα Β – Το σχήμα της Βάσης δεδομένων βλέπουμε το Σχήμα της βάσης μας όπου μπορούμε να διακρίνουμε τα αντίστοιχα μοντέλα που αναφέραμε στην παράγραφο 2.3.1 Μοντέλα της εφαρμογής

[1] Μελλοντική λειτουργία

4 Αποτελέσματα - Συμπεράσματα

Η γρήγορη και έγκυρη ενημέρωση και επικοινωνία είναι πλέον μείζονος σημασίας. Με την εφαρμογή αυτή παρέχουμε στους χρήστες ένα εργαλείο επικοινωνίας μέσα σε γεωγραφικά και χρονικά πλαίσια. Οι χρήστες μπορούν να ενημερώνουν και να ενημερώνονται για σημαντικά συμβάντα στην περιοχή που βρίσκονται. Μπορούν να επικοινωνούν αιτήματα για παροχή ή/και αναζήτηση προϊόντων και υπηρεσιών δημιουργώντας έτσι τοπικές αγορές αφού τα αιτήματα αυτά αφορούν περιοχές περιορισμένης εμβέλειας.

Το σύστημα αυτό μπορεί να χρησιμοποιηθεί από όλες τις ηλικιακές ομάδες αφού παρέχει την δυνατότητα για προσθήκη διαφόρων κατηγοριών άρα και χρήσεων. Επίσης μπορεί να χρησιμοποιηθεί από διάφορες επαγγελματικές ομάδες αλλά και από καταστήματα, πχ. για προσφορές τοπικού χαρακτήρα. Όμως η πιο σημαντική του χρήση αφορά στην παροχή βοήθειας σε καταστάσεις κρίσης όπου κάθε άνθρωπος, ζώο ή περιουσία μπορεί να κινδυνέψει. Η συλλογικότητα μιας ομάδας εθελοντών ή και επαγγελματιών μπορεί να βοηθήσει τα μέγιστα και να καθορίσει την έκβαση μιας κατάστασης. Πιστεύουμε ότι κάτι αντίστοιχο θα έπρεπε να υπάρχει σε εθνικό επίπεδο από το ίδιο το κράτος.

Το αποτέλεσμα από την χρήση αυτής της εφαρμογής είναι ότι μας γλυτώνει χρόνο και κόπο, μας δίνει την δυνατότητα να παρέχουμε την βοήθειά μας εθελοντικά ή να αναζητήσουμε εμείς οι ίδιοι βοήθεια. Μας δίνετε η δυνατότητα να στηρίξουμε την περιοχή μας, την γειτονιά μας και τον τόπο μας. Μας δίνει την δυνατότητα να λειτουργήσουμε συλλογικά όταν αυτό απαιτηθεί.

Όσον αφορά την Εργασία, η απόφαση για το θέμα που πραγματεύεται ήταν καθοριστική αφού το όλο σύστημα είναι αρκετά πολύπλοκο. Έπρεπε να ασχοληθούμε με νέες τεχνολογίες και τεχνικές καθώς επίσης και με διαφορετικά συστήματα. Αναλύσαμε, προγραμματίσαμε και αναπτύξαμε τα συστήματα αυτά αλλά και την διεπαφή τους. Πχ. η δημιουργία εφαρμογών στην πλατφόρμα Android όσο τετριμμένη κι αν είναι σήμερα, μας δυσκόλεψε αρκετά αφού η επιλογή της Material Design ως γραφική διεπαφή στο Android είναι ακόμα προβληματική, πχ. αν θέλουμε να παρουσιάσουμε δύο λίστες μέσα σε ένα container.

Ασχοληθήκαμε με τεχνολογίες και τεχνικές που αφορούν τόσο το back-end όσο και το front-end, την γραφική διεπαφή της εφαρμογής αλλά και αυτή του συστήματος διαχείρισης που είναι διαφορετική από αυτή του Material Design. Δημιουργήσαμε δύο διαφορετικούς διακομιστές για λόγους ασφάλειας, ένα για την διαχείριση και ένα για τους χρήστες της εφαρμογής. Ο διακομιστής που υποστηρίζει την εφαρμογή υλοποιεί την τεχνική Restful παρέχοντας μια διεπαφή API εξασφαλίζοντας έτσι την διαλειτουργικότητα με άλλα συστήματα. Ασχοληθήκαμε με σχεσιακές βάσεις δεδομένων, με γεωγραφικά δεδομένα, με υπηρεσίες βασισμένες στην τοποθεσία, με υπολογισμούς γεωγραφικής εμβέλειας και αποστάσεων, δημιουργήσαμε σύστημα διαχείρισης χρηστών αλλά και περιεχομένου, σχεδιάσαμε και υλοποιήσαμε την δική μας βιβλιοθήκη για την διασύνδεση με την βάση δεδομένων, αναλύσαμε τις απαιτήσεις των επιμέρους υποσυστημάτων και αναπτύξαμε τα απαραίτητα μοντέλα, ασχοληθήκαμε με εικονικές μηχανές και την φιλοξενία των διακομιστών μας και των υπηρεσιών που παρέχουν. Τέλος είδαμε πως μπορούμε να αλληλεπιδράσουμε με διαδικτυακές υπηρεσίες άλλων συστημάτων πως μπορούμε να τις ενσωματώσουμε και να τις χρησιμοποιήσουμε στο δικό μας σύστημα.

4.1 Μελλοντική δυνατότητες και επεκτάσεις

Το σύστημά μας, αν και λειτουργικό, απέχει αρκετά από ένα πλήρη προϊόν που μπορεί να χρησιμοποιηθεί απρόσκοπτα. Υπάρχουν αρκετά που πρέπει να γίνουν, κάποια από αυτά τα αναφέρουμε παρακάτω με σειρά προτεραιότητας:

1. Συνεργασία με δημόσιες αρχές (public authorities) όπως κράτος, δήμοι, περιφέρειες, πυροσβεστική, νοσοκομεία, μετεωρολογική υπηρεσία, υπηρεσία πολίτικης προστασίας, κα.
2. Ειδικά «βάρη» για τους χρήστες, για την βαθμολόγηση, αξιολόγηση, κατάταξη και την αξιοπιστία τους όσον αφορά τις αναρτήσεις των Ενημερώσεων. Οι χρήστες θα μπορούν να αξιολογούν κάθε ανάρτηση δίνοντας έτσι στον δημιουργό της κάποια βαθμολογία η οποία θα αντανακλά την αξιοπιστία του χρήστη.
3. Αναζήτηση και φιλτράρισμα των Ενημερώσεων και με άλλα πεδία όπως η ημερομηνία, ο δημιουργός, ο αριθμός σχολίων, αναζήτηση της πιο δραστήριας Ενημέρωσης, δηλαδή αυτής με τα πιο πολλά και πρόσφατα σχόλια, κα.
4. Διαχείριση των Σχολίων των Ενημερώσεων
5. Χρήση τεχνολογίας Server PUSH (55) για πιο γρήγορη και άμεση επικοινωνία
6. Υποστήριξη περισσότερων πλατφόρμων και συσκευών όπως Desktop, iPhone, κα.
7. Υποστήριξη προσωρινής μνήμης (Cache) για πιο γρήγορη φόρτωση αποτελεσμάτων
8. Συνεργασία με κοινωνικά δίκτυα για αυτόματη ανάρτηση σε αυτά
9. Υποστήριξη περισσότερων τύπων αρχείων που μπορούμε να επισυνάψουμε στις Ενημερώσεις και στα σχόλιά τους, όπως αρχεία ήχου, video, κα.
10. Υποστήριξη παραπάνω από μια φωτογραφεία στις Ενημερώσεις και στα σχόλιά τους

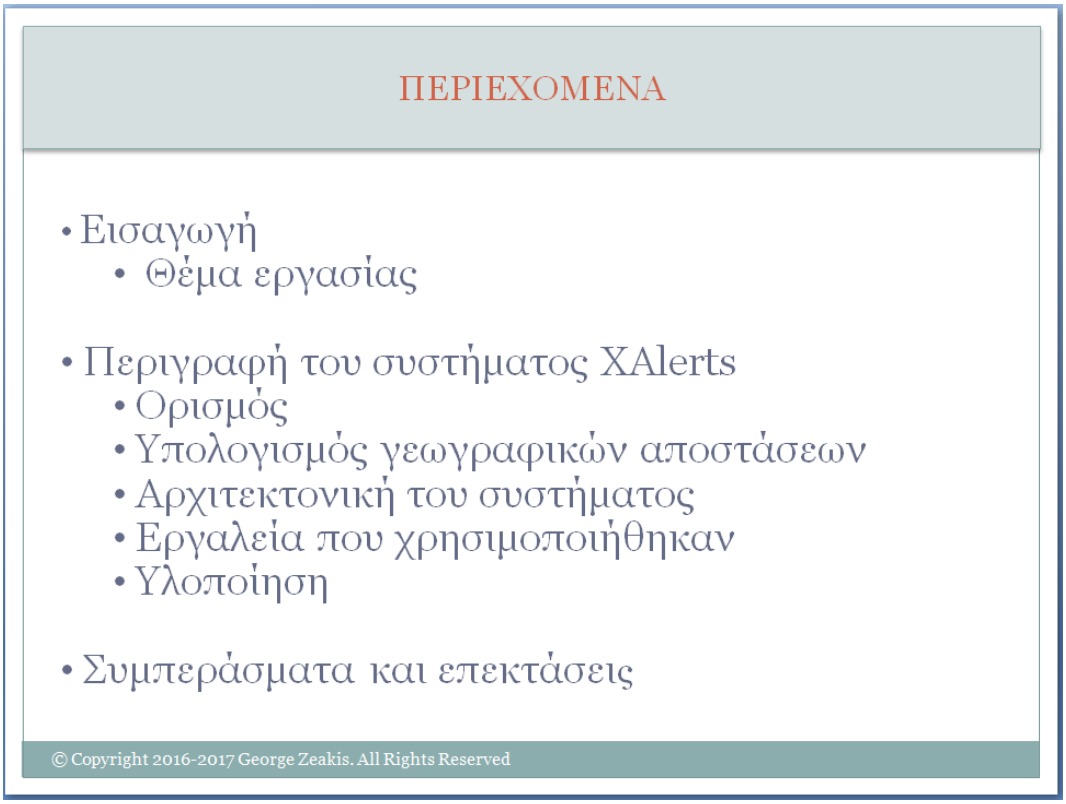
Bibliography[2]

1. *Location-based service - Wikipedia.* [Online] https://en.wikipedia.org/wiki/Location-based_service.
2. *Geo-fence - Wikipedia.* [Online] https://en.wikipedia.org/wiki/Geo-fence.
3. *Haversine formula.* [Online] https://en.wikipedia.org/wiki/Haversine_formula.
4. *Ορθοδρομία.* [Online] https://el.wikipedia.org/wiki/Ορθοδρομία.
5. Representational state transfer (REST). [Online] https://en.wikipedia.org/wiki/Representational_state_transfer.
6. *Responsive Web Design.* [Online] https://en.wikipedia.org/wiki/Responsive_web_design.
7. *Jetty - Servlet Engine and Http Server.* [Online] http://www.eclipse.org/jetty/.
8. *Java (programming language) - Wikipedia.* [Online] https://en.wikipedia.org/wiki/Java_(programming_language).
9. *MySQL Database.* [Online] https://www.mysql.com/.
10. *Ajax (programming) - Wikipedia.* [Online] https://en.wikipedia.org/wiki/Ajax_(programming).
11. *JQuery.* [Online] https://jquery.com/.
12. *FreeMarker Java Template Engine.* [Online] http://freemarker.org/.
13. *google/gson: A Java serialization/deserialization library to convert Java Objects into JSON and back.* [Online] https://github.com/google/gson.
14. *JCommander.* [Online] http://jcommander.org/.
15. *Google Maps.* [Online] https://www.google.com/maps/.
16. *CKEditor.com | The best web text editor for everyone.* [Online] http://ckeditor.com/.
17. *GRNet.* [Online] https://grnet.gr/.
18. *Synnefo.* [Online] https://www.synnefo.org/.
19. *Ubuntu Server.* [Online] https://www.ubuntu.com/server.
20. *Git - Wikipedia.* [Online] https://en.wikipedia.org/wiki/Git.
21. *Linus Torvalds - Wikipedia.* [Online] https://en.wikipedia.org/wiki/Linus_Torvalds.
22. *Bitbucket | The Git solution for professional teams.* [Online] https://bitbucket.org/.
23. JSON Web Tokens - jwt.io. [Online] https://jwt.io/.
24. *RFC 7519 - JSON Web Token (JWT).* [Online] https://tools.ietf.org/html/rfc7519.
25. *JOSE + JWT library for Java | Connect2id.* [Online] http://connect2id.com/products/nimbus-jose-jwt.
26. *Spark Framework: An expressive web framework for Kotlin and Java.* [Online] http://sparkjava.com/.
27. *Sinatra.* [Online] https://en.wikipedia.org/wiki/Sinatra_(software).
28. *JavaLambda.* [Online] https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html.
29. *Apache Tomcat.* [Online] https://tomcat.apache.org/.
30. *Google Maps JavaScript API | Google Developers.* [Online] https://developers.google.com/maps/documentation/javascript/.
31. *Traversy Media, Build A CMS Admin Bootstrap Theme From Scratch - YouTube.* [Online] https://www.youtube.com/watch?v=pXbEcGUtHgo.
32. *Okeanos IAAS.* [Online] https://okeanos-global.grnet.gr/home/.
33. *4.10 Forms — HTML5.* [Online] https://www.w3.org/TR/html5/forms.html.
34. *Cryptographic hash function - Wikipedia.* [Online] https://en.wikipedia.org/wiki/Cryptographic_hash_function.
35. *Hash function - Wikipedia.* [Online] https://en.wikipedia.org/wiki/Hash_function.
36. *SHA-2 - Wikipedia.* [Online] https://en.wikipedia.org/wiki/SHA-2.

[2] ISO 690 – Numerical Reference

37. *Android.* [Online] https://www.android.com/.
38. *Android Studio.* [Online] https://developer.android.com/studio/index.html.
39. *Material Design for Android | Android Developers.* [Online] https://developer.android.com/design/material/index.html.
40. *Android-EasyLocation.* [Online] https://github.com/akhgupta/Android-EasyLocation.
41. *ImagePicker.* [Online] https://github.com/Mariovc/ImagePicker.
42. *Asynchronous Http Client for Android.* [Online] https://github.com/loopj/android-async-http.
43. *PhotoView.* [Online] https://github.com/chrisbanes/PhotoView.
44. *JWT for Android.* [Online] https://github.com/jwtk/jjwt.
45. *Material Design.* [Online] https://material.io/.
46. *Making Your App Location-Aware | Android Developers.* [Online] https://developer.android.com/training/location/index.html.
47. *Buttons: Floating Action Button - Components - Material design guidelines.* [Online] https://material.io/guidelines/components/buttons-floating-action-button.html.
48. *MySQL Connector/J.* [Online] https://dev.mysql.com/downloads/connector/j/.
49. *NetBeans.* [Online] https://netbeans.org/.
50. *Data Access Object.* [Online] https://en.wikipedia.org/wiki/Data_access_object.
51. *Prepared Statement.* [Online] https://en.wikipedia.org/wiki/Prepared_statement.
52. *Geo (proximity) Search with MySQL .* [Online] http://www.arubin.org/files/geo_search.pdf.
53. *Fast nearest-location finder for SQL.* [Online] http://www.plumislandmedia.net/mysql/haversine-mysql-nearest-loc/.
54. *Enhanced Entity–Relationship Model.* [Online] https://en.wikipedia.org/wiki/Enhanced_entity-relationship_model.
55. *Push Technology.* [Online] https://en.wikipedia.org/wiki/Push_technology.

Παράρτημα Α - Διαφάνειες Παρουσίασης



ΕΙΣΑΓΩΓΗ

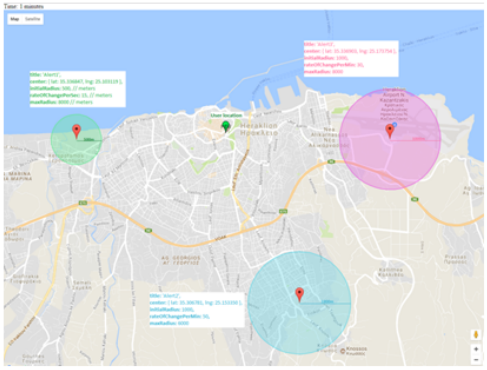
- Η έγκυρη και έγκαιρη ενημέρωση και επικοινωνία: καθοριστικός παράγοντας στην ζωή μας, σε κρίσιμες καταστάσεις, κα.
- Ανάγκη για ένα δίαυλο επικοινωνίας σε προκαθορισμένο τόπο και χρόνο
- Παροχή υπηρεσιών βασιζόμενες στην τοποθεσία (Location-based service, LBS) των έξυπνων συσκευών

ΘΕΜΑ ΕΡΓΑΣΙΑΣ

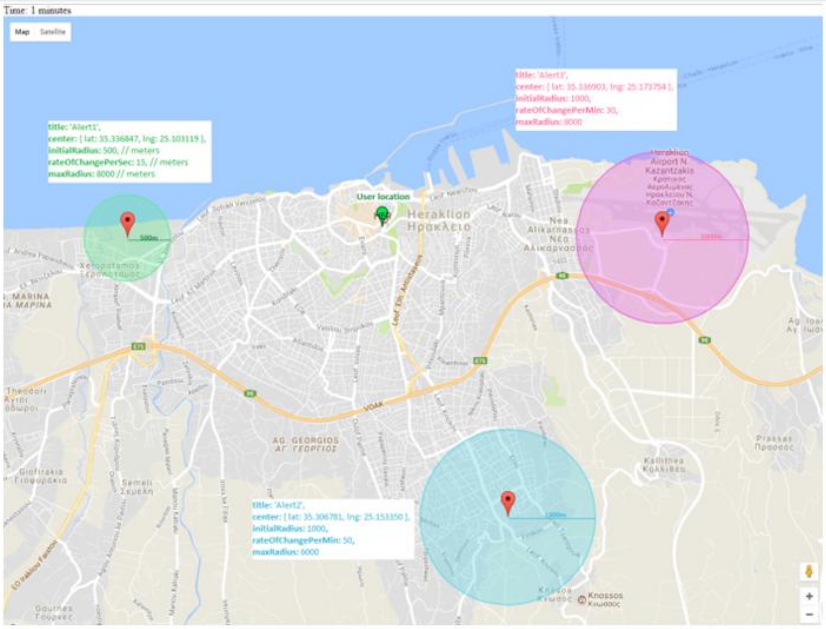
- Δημιουργία εφαρμογής για έξυπνα κινητά τηλέφωνα και της απαραίτητης υποδομής
- Παροχή υπηρεσιών στοχευόμενης πληροφόρησης
- Παροχή περιεχομένου από τους ίδιους του χρήστες
- Γεωγραφικά εξαρτώμενες και χρονικά έγκυρες πληροφορίες
- Αλληλεπίδραση, συλλογικότητα, συνεργασιμότητα μεταξύ των χρηστών

ΤΟ ΣΥΣΤΗΜΑ XALERTS: ΟΡΙΣΜΟΣ

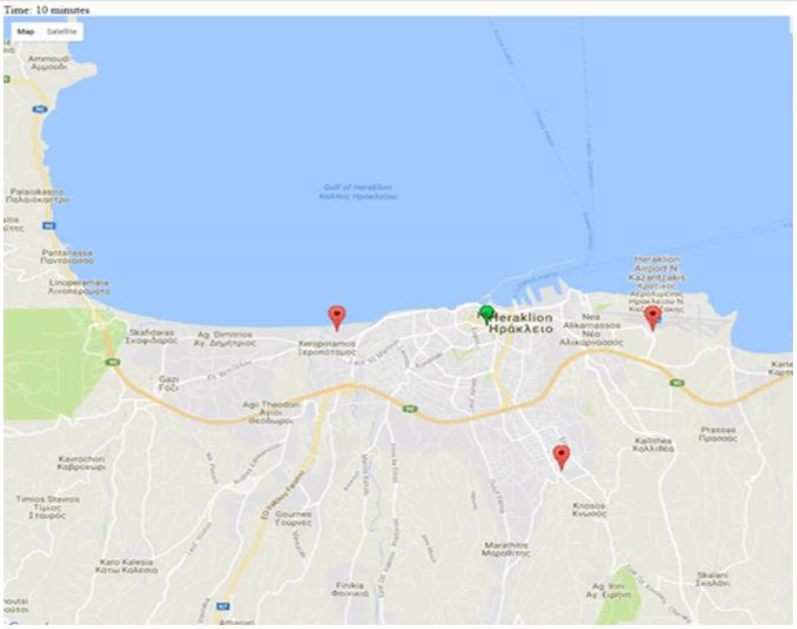
• " Το XAlerts είναι μια πλατφόρμα αποστολής μηνυμάτων που βασίζεται σε κινητές εφαρμογές και είναι κατασκευασμένη για την δημιουργία και ανεύρεση ειδοποιήσεων ή αιτημάτων παροχής πληροφοριών και υπηρεσιών εντός ορισμένης γεωγραφικής περιοχής με χρονικά εξαρτώμενη εμβέλεια."



ΠΑΡΑΔΕΙΓΜΑ ΣΤΟΝ ΧΑΡΤΗ



ΕΙΚΟΝΙΚΗ ΛΕΙΤΟΥΡΓΙΑ



ΥΠΟΛΟΓΙΣΜΟΣ ΓΕΩΓΡΑΦΙΚΩΝ ΑΠΟΣΤΑΣΕΩΝ

- Τύπος Haversine

- φ2,λ2: User
- φ1,λ1: Alert
- d: distance

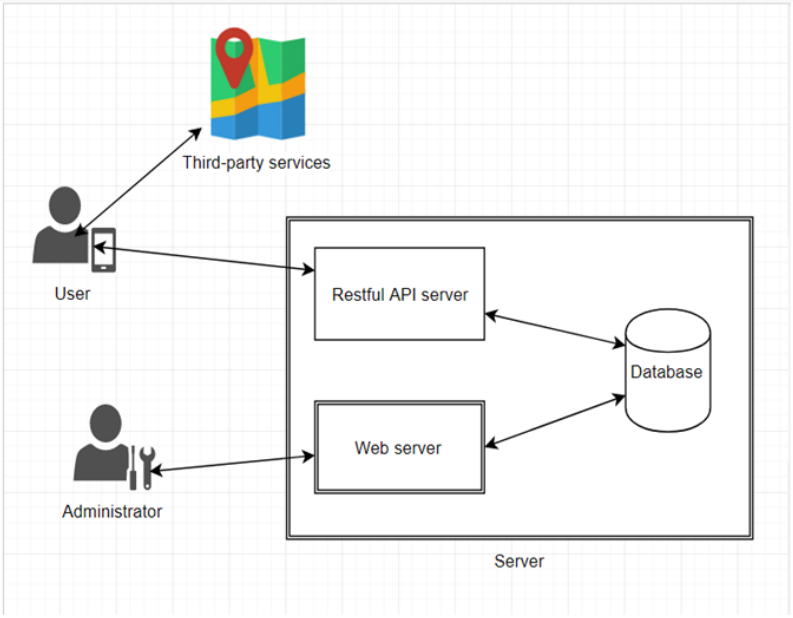
$$\operatorname{hav}\left(\frac{d}{r}\right) = \operatorname{hav}(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)\operatorname{hav}(\lambda_2 - \lambda_1)$$

$$d = 2r\arcsin\left(\sqrt{\operatorname{hav}(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)\operatorname{hav}(\lambda_2 - \lambda_1)}\right)$$

$$= 2r\arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

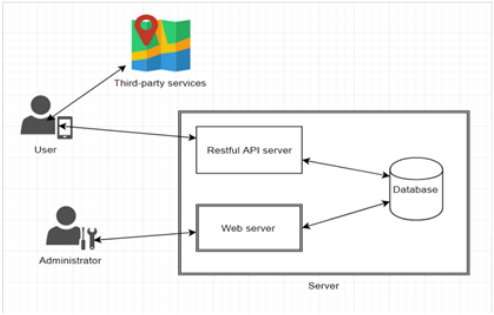
- όπου hav(θ) = sin^2(θ/2) = (1 − cos(θ))/2
- λύνοντας ως προς d -> απόσταση του χρήστη από την Ενημέρωση

ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ



ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ

- 4 οντότητες:
  - ο Χρήστης
  - ο Διαχειριστής
  - ο Διακομιστής
    - Restful API server
    - Web server
    - Database server
  - Υπηρεσίες τρίτων



ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ ΠΑΡΕΧΟΜΕΝΕΣ ΥΠΗΡΕΣΙΕΣ API

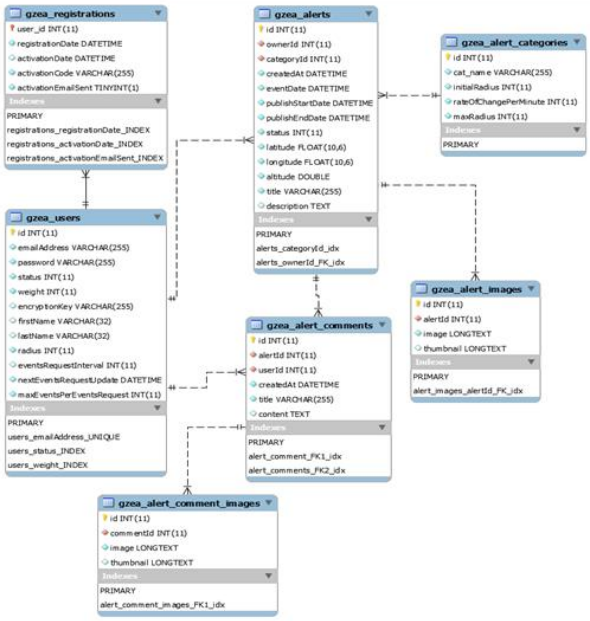
	HTTP METHODS									
Resource (Path/URL)	GET	PUT	PATCH	DELETE	POST	OPTIONS	CONNECT	TRACE	HEAD	Auth	Description
/api/v1.0/	X										Service main page
/api/v1.0/login/					X						Provide the user credentials to get the JWT
/api/v1.0/registrations/register/					X						Just a simple registration
/api/v1.0/alerts/?lat={lat}&lon={lon}&alt={altitude}/	X									X	Get the alerts in your area
/api/v1.0/alerts/{alertId}/	X									X	Get the alert with id={alertId}
/api/v1.0/alerts/{alertId}/		X								X	Change your alert with id={alertId}
/api/v1.0/alerts/{alertId}/				X						X	Delete your alert with id={alertId}
/api/v1.0/alerts/					X					X	Create a new alert.
/api/v1.0/alerts/{alertId}/images/	X									X	Get the images of the alert with id={alertId}
/api/v1.0/alerts/{alertId}/images/					X					X	Create or replace an existing image for the alert with id={alertId}
/api/v1.0/alerts/{alertId}/comments/	X									X	Get the list of comments for the alert with id={alertId}
/api/v1.0/alerts/{alertId}/comments/					X					X	Create a new comment for the alert with id={alertId}
/api/v1.0/alerts/{alertId}/comments/{cid}/images/	X									X	Get the images of the comment with id={cid} of the alert with id={alertId}
/api/v1.0/alerts/{alertId}/comments/{cid}/images/					X					X	Create an image for the comment with id={cid} of the alert with id={alertId}
/api/v1.0/alerts/categories/	X									X	Get all alert Categories

ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ ΥΠΟΛΟΓΙΣΜΟΣ ΑΠΟΣΤΑΣΕΩΝ ΣΕ SQL

```
1. SELECT
2.  gzea_alerts.*,
3.  c.name AS categoryName,
4.  (c.initialRadius + (TIMESTAMPDIFF(MINUTE,eventDate,NOW())*c.rateOfChangePerMinute)) AS er,
6.  IF ((SELECT er) > c.maxRadius, c.maxRadius, (SELECT er)) AS eventRadius,
7.    (:kmPerDegree*DEGREES(ACOS(COS(RADIANS(:lat)) * COS(RADIANS(latitude))
9.    * COS(RADIANS(:lon) - RADIANS(longitude)) + SIN(RADIANS(:lat))
11.   * SIN(RADIANS(latitude)))))*1000 AS distance_in_m
12. FROM gzea_alerts
13. JOIN gzea_alert_categories c ON c.id=categoryId
14. WHERE status IN (1) -- Alert is in active state
15. HAVING distance_in_m <= eventRadius
16. ORDER BY c.id DESC, distance_in_m ASC
17. LIMIT 15;
```

- :kmPerDegree = 111.045 Km (κατά προσέγγιση)
- :lat = γεωγραφικό πλάτος του χρήστη
- :lon = γεωγραφικό μήκος του χρήστη
- latitude = γεωγραφικό πλάτος της Ενημέρωσης
- longitude = γεωγραφικό μήκος της Ενημέρωσης

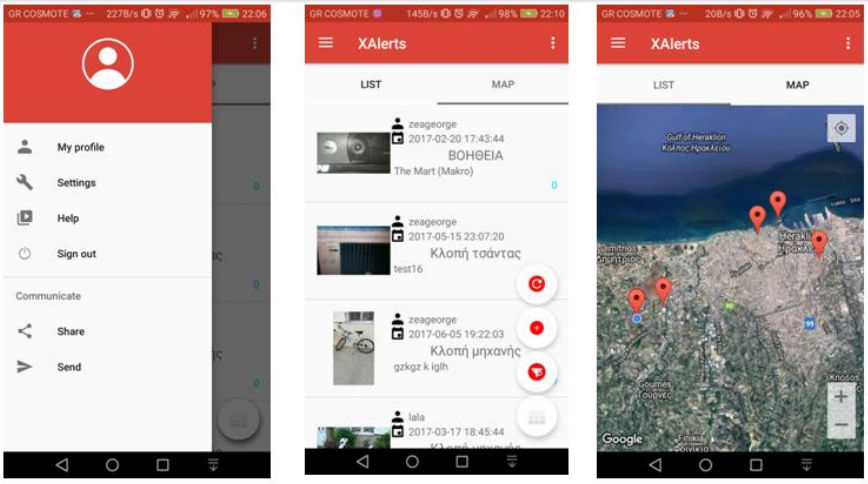
ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ
DATABASE SCHEMA (EER DIAGRAM)



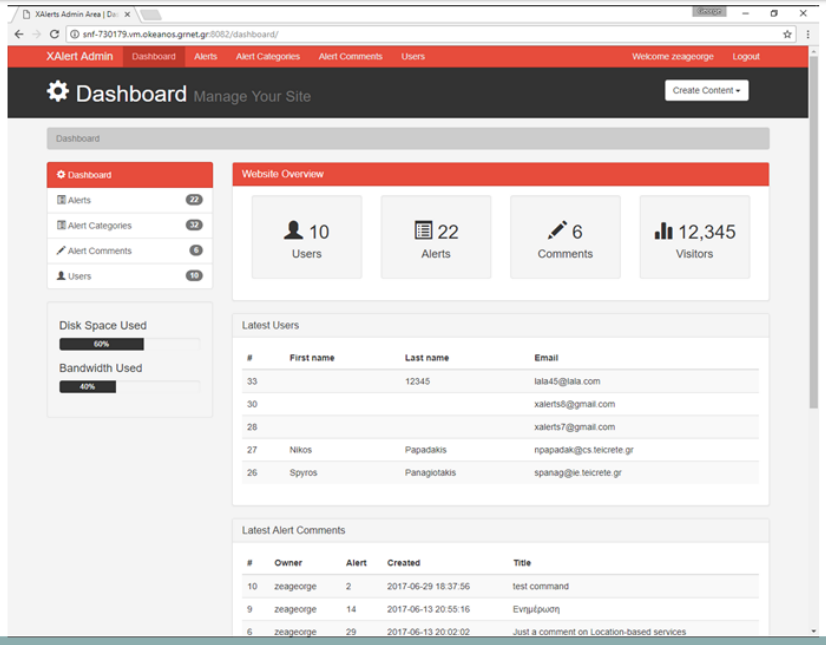
ΕΡΓΑΛΕΙΑ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΗΘΗΚΑΝ

- Ανοικτού κώδικα εργαλεία και βιβλιοθήκες
- Okeanos IaaS (cloud service)
- Java
- Spark micro framework (Jetty web server),
- MySQL,
- Google Maps
- Git (Bitbucket)
- Restful web services
- JSON web tokens, JWT
- Google Material Design

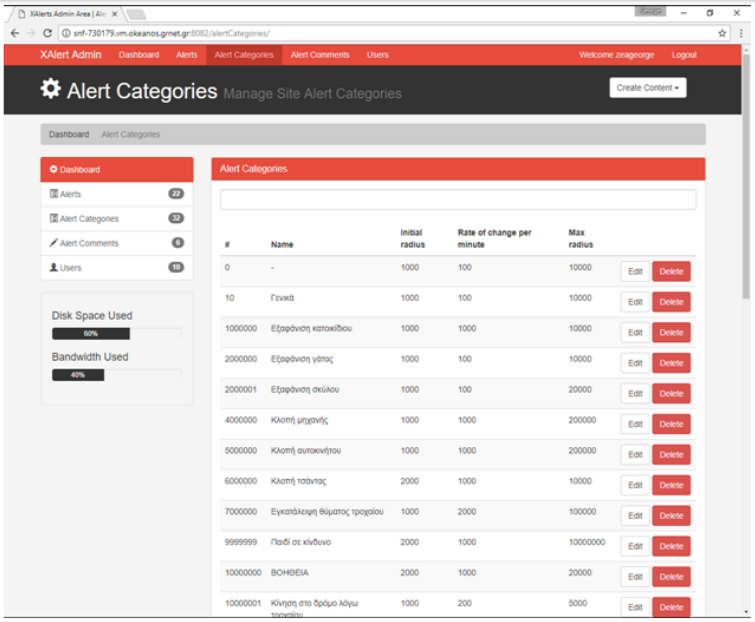
ΥΛΟΠΟΙΗΣΗ: Η ΕΦΑΡΜΟΓΗ ΤΟΥ ΧΡΗΣΤΗ



ΥΛΟΠΟΙΗΣΗ: Η ΕΦΑΡΜΟΓΗ ΤΟΥ ΔΙΑΧΕΙΡΙΣΤΗ



ΥΛΟΠΟΙΗΣΗ: Η ΕΦΑΡΜΟΓΗ ΤΟΥ ΔΙΑΧΕΙΡΙΣΤΗ



ΣΥΜΠΕΡΑΣΜΑΤΑ

- Η γρήγορη και έγκυρη ενημέρωση και επικοινωνία είναι πλέον μείζονος σημασίας
- Διασύνδεση μέσω ενός καθολικού δίαυλου επικοινωνίας
- Πληροφορία γεωγραφικά εξαρτώμενη και χρονικά έγκυρη
- Απλή και εύχρηστη εφαρμογή για κινητά τηλέφωνα
- Ανάγκη για συντονισμό & συγχρονισμό σε καταστάσεις κρίσης
- Εξοικονόμηση χρόνου, κόπου & πόρων

ΕΠΕΚΤΑΣΕΙΣ

- Συνεργασία με δημόσιες αρχές (public authorities) όπως κράτος, δήμοι, περιφέρειες, πυροσβεστική, νοσοκομεία, μετεωρολογική υπηρεσία, υπηρεσία πολίτικης προστασίας, κα.
- Ειδικά «βάρη» για τους χρήστες, για την αξιολόγηση και την αξιοπιστία τους όσον αφορά τις αναρτήσεις των Ενημερώσεων.
- Αναζήτηση και φιλτράρισμα των Ενημερώσεων και με άλλα πεδία
- Χρήση τεχνολογίας Server PUSH (55)
- Υποστήριξη προσωρινής μνήμης (Cache)
- Συνεργασία με κοινωνικά δίκτυα για αυτόματη ανάρτηση σε αυτά
- Υποστήριξη περισσότερων τύπων αρχείων που μπορούμε να επισυνάψουμε στις Ενημερώσεις και στα σχόλιά τους
- Υποστήριξη παραπάνω από μια φωτογραφεία στις Ενημερώσεις και στα σχόλιά τους

Ερωτήσεις - απορίες

Ευχαριστώ

Ζεάκης Γεώργιος


zeageorge@gmail.com


https://github.com/zeageorge

Παράρτημα Β – Το σχήμα της Βάσης δεδομένων

```
user@snf-730179:20170723140707:~$ mysqldump -d -u root -p xalerts
Enter password:
-- MySQL dump 10.13  Distrib 5.7.18, for Linux (x86_64)
--
-- Host: localhost    Database: xalerts
-- ------------------------------------------------------
-- Server version       5.7.18-0ubuntu0.16.04.1

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `gzea_alert_categories`
--

DROP TABLE IF EXISTS `gzea_alert_categories`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `gzea_alert_categories` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `cat_name` varchar(255) CHARACTER SET utf8 NOT NULL,
  `initialRadius` int(11) NOT NULL DEFAULT '1000' COMMENT 'in
meters',
  `rateOfChangePerMinute` int(11) NOT NULL DEFAULT '100',
  `maxRadius` int(11) NOT NULL DEFAULT '10000',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=10000022 DEFAULT CHARSET=utf8
COLLATE=utf8_bin;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `gzea_alert_comment_images`
--

DROP TABLE IF EXISTS `gzea_alert_comment_images`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `gzea_alert_comment_images` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `commentId` int(11) NOT NULL,
```

```
  `image` longtext COLLATE utf8_bin NOT NULL,
  `thumbnail` longtext CHARACTER SET utf8,
  PRIMARY KEY (`id`),
  KEY `alert_comment_images_FK1_idx` (`commentId`),
  CONSTRAINT `alert_comment_images_FK1` FOREIGN KEY (`commentId`)
REFERENCES `gzea_alert_comments` (`id`) ON DELETE CASCADE ON UPDATE
CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8
COLLATE=utf8_bin;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `gzea_alert_comments`
--

DROP TABLE IF EXISTS `gzea_alert_comments`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `gzea_alert_comments` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `alertId` int(11) NOT NULL,
  `userId` int(11) NOT NULL,
  `createdAt` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `title` varchar(255) COLLATE utf8_bin NOT NULL,
  `content` text CHARACTER SET utf8,
  PRIMARY KEY (`id`),
  KEY `alert_comment_FK1_idx` (`alertId`),
  KEY `alert_comments_FK2_idx` (`userId`),
  CONSTRAINT `alert_comment_FK1` FOREIGN KEY (`alertId`) REFERENCES
`gzea_alerts` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `alert_comments_FK2` FOREIGN KEY (`userId`) REFERENCES
`gzea_users` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB AUTO_INCREMENT=11 DEFAULT CHARSET=utf8
COLLATE=utf8_bin;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `gzea_alert_images`
--

DROP TABLE IF EXISTS `gzea_alert_images`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `gzea_alert_images` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `alertId` int(11) NOT NULL,
  `image` longtext COLLATE utf8_bin NOT NULL,
  `thumbnail` longtext CHARACTER SET utf8,
  PRIMARY KEY (`id`),
  KEY `alert_images_alertId_FK_idx` (`alertId`),
  CONSTRAINT `alert_images_alertId_FK` FOREIGN KEY (`alertId`)
REFERENCES `gzea_alerts` (`id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=14 DEFAULT CHARSET=utf8
COLLATE=utf8_bin;
/*!40101 SET character_set_client = @saved_cs_client */;
```

```
--
-- Table structure for table `gzea_alerts`
--

DROP TABLE IF EXISTS `gzea_alerts`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `gzea_alerts` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `ownerId` int(11) NOT NULL,
  `categoryId` int(11) NOT NULL DEFAULT '0',
  `createdAt` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT
'When this event was put(submitted by the user) on system.\n\nThe
DATETIME type is used for values that contain both date and time
parts. \nMySQL retrieves and displays DATETIME values in ''YYYY-MM-
DD HH:MM:SS'' format. \nThe supported range is ''1000-01-01
00:00:00'' to ''9999-12-31 23:59:59''.\n\nA DATETIME or TIMESTAMP
value can include a trailing fractional seconds part in up to
microseconds (6 digits) precision. \nIn particular, any fractional
part in a value inserted into a DATETIME or TIMESTAMP column is
stored rather than discarded. With the fractional part included, the
format for these values is ''YYYY-MM-DD HH:MM:SS[.fraction]'', the
range for DATETIME values is ''1000-01-01 00:00:00.000000'' to
''9999-12-31 23:59:59.999999'', and the range for TIMESTAMP values
is ''1970-01-01 00:00:01.000000'' to ''2038-01-19
03:14:07.999999''.\nThe fractional part should always be separated
from the rest of the time by a decimal point; no other fractional
seconds delimiter is recognized.',
  `eventDate` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT
'When the event started/took place',
  `publishStartDate` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP
COMMENT 'When to publish this event',
  `publishEndDate` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP
COMMENT 'When to unpublish/suppress this event',
  `status` int(11) NOT NULL DEFAULT '1' COMMENT '-1=deleted,
0=inactive, 1=active, 2=blocked, 3=reported, 4=completed',
  `latitude` float(10,6) NOT NULL DEFAULT '0.000000',
  `longitude` float(10,6) NOT NULL DEFAULT '0.000000',
  `altitude` double NOT NULL DEFAULT '0',
  `title` varchar(255) CHARACTER SET utf8 NOT NULL,
  `description` text COLLATE utf8_bin,
  PRIMARY KEY (`id`),
  KEY `alerts_categoryId_idx` (`categoryId`),
  KEY `alerts_ownerId_FK_idx` (`ownerId`),
  CONSTRAINT `alerts_categoryId` FOREIGN KEY (`categoryId`)
REFERENCES `gzea_alert_categories` (`id`) ON DELETE CASCADE ON
UPDATE CASCADE,
  CONSTRAINT `alerts_ownerId_FK` FOREIGN KEY (`ownerId`) REFERENCES
`gzea_users` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB AUTO_INCREMENT=31 DEFAULT CHARSET=utf8
COLLATE=utf8_bin;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `gzea_registrations`
--
```

```
DROP TABLE IF EXISTS `gzea_registrations`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `gzea_registrations` (
  `user_id` int(11) NOT NULL,
  `registrationDate` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP
COMMENT 'When the user submitted the registration data.',
  `activationDate` datetime DEFAULT NULL COMMENT 'When the user
activated his/her account.',
  `activationCode` varchar(255) CHARACTER SET utf8 NOT NULL COMMENT
'This activation code is send to the user email address.\nThe user
must click the link that contains this activation code to activate
his/her account.',
  `activationEmailSent` tinyint(1) NOT NULL DEFAULT '0',
  PRIMARY KEY (`user_id`),
  KEY `registrations_registrationDate_INDEX` (`registrationDate`),
  KEY `registrations_activationDate_INDEX` (`activationDate`),
  KEY `registrations_activationEmailSent_INDEX`
(`activationEmailSent`),
  CONSTRAINT `registrations_user_id_fk` FOREIGN KEY (`user_id`)
REFERENCES `gzea_users` (`id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `gzea_users`
--

DROP TABLE IF EXISTS `gzea_users`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `gzea_users` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `emailAddress` varchar(255) COLLATE utf8_bin NOT NULL,
  `password` varchar(255) COLLATE utf8_bin NOT NULL,
  `status` int(11) NOT NULL DEFAULT '2' COMMENT 'INACTIVE(0):
Registered User hasn''t activated his/her account yet.\nACTIVE(1):
User has activated his/her account.\nENABLED(2): User/account is
enabled and fully working.\nDISABLED(3):
(apenergopoihsh/apenergopoihmeno).\nBLOCKED(4): User is reported
(desmeumeno).\nSUSPENDED(5): (o logariasmos exei
anastalei).\nDELETED(63): User has been deleted.\n\n',
  `weight` int(11) NOT NULL DEFAULT '1' COMMENT 'The more weight a
User has, the more important alerts he/she can create.',
  `encryptionKey` varchar(255) CHARACTER SET utf8 DEFAULT NULL
COMMENT 'This encryption key/secret will be used to encrypt
communication for every session. It must change at least three times
in every session.',
  `firstName` varchar(32) CHARACTER SET utf8 DEFAULT NULL,
  `lastName` varchar(32) COLLATE utf8_bin DEFAULT NULL,
  `radius` int(11) NOT NULL DEFAULT '10000' COMMENT 'The radius of
the current location, in meters',
  `eventsRequestInterval` int(11) DEFAULT '10' COMMENT 'in minutes',
  `nextEventsRequestUpdate` datetime NOT NULL DEFAULT
CURRENT_TIMESTAMP,
```

```
  `maxEventsPerEventsRequest` int(11) NOT NULL DEFAULT '30' COMMENT
'Max number of events the system returns to a user''s request to see
the events in his/her area',
  PRIMARY KEY (`id`),
  UNIQUE KEY `users_emailAddress_UNIQUE` (`emailAddress`),
  KEY `users_status_INDEX` (`status`),
  KEY `users_weight_INDEX` (`weight`)
) ENGINE=InnoDB AUTO_INCREMENT=34 DEFAULT CHARSET=utf8
COLLATE=utf8_bin;
/*!40101 SET character_set_client = @saved_cs_client */;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2017-07-23 14:35:57
user@snf-730179:20170723140757:~$
```

Παράρτημα Γ.1 – Απόσπασμα των ρυθμίσεων του Restful API server

```
################################################################################
# Copyright 2016, George Zeakis <zeageorge@gmail.com>
# All rights reserved.
#
# Redistribution and use in source and binary forms, with or without
# modification, are permitted provided that the following conditions are
# met:
#     * Redistributions of source code must retain the above copyright
# notice, this list of conditions and the following disclaimer.
#     * Redistributions in binary form must reproduce the above
# copyright notice, this list of conditions and the following disclaimer
# in the documentation and/or other materials provided with the
# distribution.
#     * Neither the name of George Zeakis <zeageorge@gmail.com> nor the
# names of its contributors may be used to endorse or promote products
# derived from this software without specific prior written permission.
#
# THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
# "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
# LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
# A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
# OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
# SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
# LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
# DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
# THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
# (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
# OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
#           ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Author George Zeakis <zeageorge@gmail.com>
# This is the configuration file of the xalerts application.
# You can pass this file as
#   mvn exec:java -Dexec.args="@/someFilesystemPathTo/application.config"
#
# Every option (or all options) in this file can be used as command line
# option as
#   mvn exec:java -Dexec.args="-v -d -wsp=80 -dbh=localhost -dbp=3306
#            -dbn=xalerts -dbu=xalerts_system -dbpw=xalerts_password"
# You must provide at least all the required options.
# No spaces allowed between the option name, the equal sign and the value,
# eg.
# --web-server-port =80        wrong
# --web-server-port= 80        wrong
# --web-server-port = 80       wrong
# --web-server-port=80         correct
#
# Some options such as -h, --help, --debug, -d, --verbose, -v, etc. work as a
# switch (flag). The existence of these turn on the corresponding functionality.
# for documentation on the following options go to
# https://www.xalerts.gr/docs/api/v1.0/
################################################################################
# GENERAL ======================================================================
# Prints out all the options with their descriptions
# This option is a switch
# Default: off
#--help
# or -h
# ------------------------------------------------------------------------------
# Debug mode
# This option is a switch
# Default: off
--debug
# or -d
# ------------------------------------------------------------------------------
#
# APPLICATION ==================================================================
```

```
# Application name.
# Type: String
# Required
# Default:
--app-name=XAlerts_API
# or -app-n=XAlerts_API
# ------------------------------------------------------------------------------
# Application author.
# Type: String
# Required
# Default:
--app-author=XAlerts (George Zeakis)
# or -app-a=XAlerts (George Zeakis)
# ------------------------------------------------------------------------------
# Application URL. For https use the --web-server-key-store-location option
# Type: URL
# Required
# Default:
--app-url=http://snf-730179.vm.okeanos.grnet.gr/api/v1.0/
# or -app-u=http://api.xalerts.gr
# ------------------------------------------------------------------------------
#
# SPARK::WEBSERVER ============================================================
# A TCP Port
# Type: integer [1-65535]
# Required
# Default:
--web-server-port=8081
# or -wsp=80
# ------------------------------------------------------------------------------
#
# DATABASE ====================================================================
# Host name or ip address
# Type: string
# Required
# Default:
--database-host=localhost
# or -dbh=localhost
# ------------------------------------------------------------------------------
# A TCP Port
# Type: integer [1-65535]
# Required
# Default:
--database-port=3307
# or -dbp=3306
# ------------------------------------------------------------------------------
# Database name
# Type: string
# Required
# Default:
--database-name=xalerts
# or -dbn=xalerts
# ------------------------------------------------------------------------------
# Database username
# Type: string
# Required
# Default:
--database-username=xalerts_system
# or -dbu=xalerts_system
# ------------------------------------------------------------------------------
# Database password
# Type: string
# Required
# Default:
--database-password=**********
# or -dbpw=xalerts_password
# ------------------------------------------------------------------------------
# Database table name prefix
# Type: string
# Optional
# Default:
--database-table-name-prefix=gzea_
# or -dbtnp=gzea_
# END OF FILE #################################################################
```

Παράρτημα Γ.2 – Απόσπασμα πηγαίου κώδικα της κύριας μεθόδου του Restful API server

```
/*
Copyright 2016, George Zeakis <zeageorge@gmail.com>
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

    * Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above
copyright notice, this list of conditions and the following disclaimer
in the documentation and/or other materials provided with the
distribution.
    * Neither the name of George Zeakis <zeageorge@gmail.com> nor the
names of its contributors may be used to endorse or promote products
derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */
package com.zeageorge.xalerts_api_slim;

import com.beust.jcommander.JCommander;
import com.google.gson.FieldNamingPolicy;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.zeageorge.xalerts_api_slim.components.alerts.AlertsController;
import com.zeageorge.xalerts_api_slim.components.alerts.comments.CommentsController;
import com.zeageorge.xalerts_api_slim.components.IndexHomePage.IndexController;
import com.zeageorge.xalerts_api_slim.components.alerts.categories.CategoriesController;
import com.zeageorge.xalerts_api_slim.lib.MySQLDataSource.MySQLDataSource;
import com.zeageorge.xalerts_api_slim.lib.Util;
import com.zeageorge.xalerts_api_slim.lib.View;
import
com.zeageorge.xalerts_api_slim.components.registrations.services.registration.RegistrationCont
roller;
import com.zeageorge.xalerts_api_slim.components.users.User;
import com.zeageorge.xalerts_api_slim.components.users.services.login.LoginController;
import com.zeageorge.xalerts_api_slim.lib.ApplicationResponse;
import freemarker.cache.ClassTemplateLoader;
import freemarker.template.Configuration;
import java.text.DateFormat;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import org.eclipse.jetty.http.HttpStatus;
import spark.Request;
import spark.Response;
import spark.Spark;
import spark.TemplateEngine;
import spark.template.freemarker.FreeMarkerEngine;

import static spark.Spark.exception;
import static spark.Spark.port;
import static spark.Spark.after;
import static spark.Spark.before;
import static spark.Spark.get;
```

```
import static spark.Spark.head;
import static spark.Spark.options;
import static spark.Spark.patch;
import static spark.Spark.post;
import static spark.Spark.put;
import static spark.Spark.trace;
import static spark.Spark.delete;
import static spark.Spark.connect;

/**
 *
 * @author George Zeakis <zeageorge@gmail.com>
 */
public final class Application {

  private static final String COPYRIGHT_HEADER
          = "*************************************************\n"
            + "* XAlerts_API                                   *\n"
            + "* Copyright 2016 Zeakis George <zeageorge@gmail.com>*\n"
            + "* All Rights Reserved.                          *\n"
            + "*************************************************\n";

  private static final Gson DEFAULT_GSON = new GsonBuilder()
          .excludeFieldsWithoutExposeAnnotation()
          .serializeNulls()
          .setDateFormat(DateFormat.LONG)
          .setFieldNamingPolicy(FieldNamingPolicy.IDENTITY)
          .create();

  private static final Util UTIL = Util.getInstance();

  private static final MySQLDataSource DATASOURCE = MySQLDataSource.getInstance();

  private static final CommandLineOptions OPTIONS = new CommandLineOptions();

  private static final FreeMarkerEngine FREEMARKER_TEMPLATE_ENGINE = new FreeMarkerEngine();

  private ApplicationResponse applicationResponse;

  private JCommander jCommander;

  private String[] args = null;

  private User currentUser;

  private Application() {
    System.out.print(COPYRIGHT_HEADER);
    System.out.println("Application started: " +
ZonedDateTime.now().format(DateTimeFormatter.RFC_1123_DATE_TIME) + "\n");
  }

  public Application setApplicationConfiguration(String[] parameters) {
    // only set once !!!
    if (args == null) {
      this.args = parameters;

      init();
    }

    return this;
  }

  private void init() {

    try {
      jCommander = new JCommander(OPTIONS, args);
    }
    catch (com.beust.jcommander.ParameterException e) {
      System.out.println(e.getMessage());

      System.out.println("Use -h or --help for list of options");

      System.exit(0);
    }

    jCommander.setProgramName(OPTIONS.getAppName());
```

```
    if (OPTIONS.isHelp()) {
      jCommander.usage();

      System.out.println("The @ syntax allows you to put all your options into a file and pass
this file as parameter: ");

      System.out.println("eg. java " + OPTIONS.getAppName().toLowerCase() + "
@/tmp/parameters");

      System.exit(0);
    }

    exception(Exception.class, (exception, request, response) -> {
        if (OPTIONS.isDebug()) {
          System.out.println("--------------------------------------------");
          exception.printStackTrace();
          System.out.println("--------------------------------------------");
        }
      });

    port(OPTIONS.getWebServerPort());

    // FreeMarker template Engine
    Configuration freeMarkerConfiguration = new Configuration();

    freeMarkerConfiguration.setNumberFormat("0.####");

    freeMarkerConfiguration.setTemplateLoader(new ClassTemplateLoader(Application.class,
"/"));

    FREEMARKER_TEMPLATE_ENGINE.setConfiguration(freeMarkerConfiguration);

    // Configure MySQL data source
    DATASOURCE
            .setHost(OPTIONS.getDatabaseHost())
            .setPort(OPTIONS.getDatabasePort())
            .setCharset("utf8")
            .setDatabaseName(OPTIONS.getDatabaseName())
            .setDatabaseTableNamePrefix(OPTIONS.getDatabaseTableNamePrefix())
            .setDatabaseUsername(OPTIONS.getDatabaseUsername())
            .setDatabasePassword(OPTIONS.getDatabasePassword());

    applicationResponse = new ApplicationResponse();
  }

  public void run() {
    before("*", Filters.init);
    before("*", Filters.addTrailingSlashes);
    before("*", Filters.validateAccessToken);

    post(Path.Web.API.V1_0.LOGIN, LoginController.login);
    post(Path.Web.API.V1_0.REGISTER, RegistrationController.register);

    get(Path.Web.API.V1_0.GET_ALERT_CATEGORIES, (Request request, Response response) -> {
      authorize(response);
      return CategoriesController.getAlertCategories.handle(request, response);
    });

    get(Path.Web.API.V1_0.GET_ALERTS_IN_LOCATION, (Request request, Response response) -> {
      authorize(response);
      return AlertsController.getAlertsInLocation.handle(request, response);
    });

    get(Path.Web.API.V1_0.GET_ALERT, (Request request, Response response) -> {
      authorize(response);
      return AlertsController.getAlert.handle(request, response);
    });

    post(Path.Web.API.V1_0.CREATE_NEW_ALERT, (Request request, Response response) -> {
      authorize(response);
      return AlertsController.createNewAlert.handle(request, response);
    });

    delete(Path.Web.API.V1_0.DELETE_ALERT, (Request request, Response response) -> {
      authorize(response);
      return AlertsController.deleteAlert.handle(request, response);
    });
```

```
//    patch(Path.Web.API.V1_0.EDIT_ALERT, (Request request, Response response) -> {
//      authorize(response);
//      return AlertsController.editAlert.handle(request, response);
//    });

    put(Path.Web.API.V1_0.EDIT_ALERT, (Request request, Response response) -> {
      authorize(response);
      return AlertsController.editAlert.handle(request, response);
    });

    get(Path.Web.API.V1_0.GET_ALERT_IMAGES, (Request request, Response response) -> {
      authorize(response);
      return AlertsController.getAlertImages.handle(request, response);
    });

    post(Path.Web.API.V1_0.CREATE_NEW_ALERT_IMAGES, (Request request, Response response) -> {
      authorize(response);
      return AlertsController.postImages.handle(request, response);
    });

    get(Path.Web.API.V1_0.GET_ALERT_COMMENTS, (Request request, Response response) -> {
      authorize(response);
      return CommentsController.getAlertComments.handle(request, response);
    });

    post(Path.Web.API.V1_0.CREATE_NEW_ALERT_COMMENT, (Request request, Response response) -> {
      authorize(response);
      return CommentsController.createNewAlertComment.handle(request, response);
    });

    get(Path.Web.API.V1_0.GET_ALERT_COMMENT_IMAGES, (Request request, Response response) -> {
      authorize(response);
      return CommentsController.getAlertCommentImages.handle(request, response);
    });

    post(Path.Web.API.V1_0.CREATE_NEW_ALERT_COMMENT_IMAGES, (Request request, Response
response) -> {
      authorize(response);
      return CommentsController.postAlertCommentImages.handle(request, response);
    });

    get(Path.Web.API.V1_0.INDEX, IndexController.serveIndexPage);

    get("*", (request, response) -> View.HTTPStatusCode501NotImplemented.handle(request,
response));
    post("*", (request, response) -> View.HTTPStatusCode501NotImplemented.handle(request,
response));
    head("*", (request, response) -> View.HTTPStatusCode501NotImplemented.handle(request,
response));
    options("*", (request, response) -> View.HTTPStatusCode501NotImplemented.handle(request,
response));
    trace("*", (request, response) -> View.HTTPStatusCode501NotImplemented.handle(request,
response));
    put("*", (request, response) -> View.HTTPStatusCode501NotImplemented.handle(request,
response));
    patch("*", (request, response) -> View.HTTPStatusCode501NotImplemented.handle(request,
response));
    delete("*", (request, response) -> View.HTTPStatusCode501NotImplemented.handle(request,
response));
    connect("*", (request, response) -> View.HTTPStatusCode501NotImplemented.handle(request,
response));

    after("*", Filters.addGzipHeader);
  }

  private void authorize(Response response) {
    if (currentUser == null || currentUser.getId() < 1) {
      response.header("WWW-Authenticate", "Bearer realm=\"\"");

      Spark.halt(HttpStatus.UNAUTHORIZED_401);
    }
  }

  public boolean isCurrentUserGuest() {
    return (currentUser == null || currentUser.getId() < 1);
  }
```

```
  public User getCurrentUser() {
    return currentUser;
  }

  public Application setCurrentUser(User currentUser) {
    this.currentUser = currentUser;

    return this;
  }

  public ApplicationResponse getApplicationResponse() {
    return applicationResponse;
  }

  public Application setApplicationResponse(ApplicationResponse applicationResponse) {
    this.applicationResponse = applicationResponse;

    return this;
  }

  public MySQLDataSource getDataSource() {
    return DATASOURCE;
  }

  public Util getUtil() {
    return UTIL;
  }

  /**
   * Default GSON
   *
   * GSON options for this method are
   * excludeFieldsWithoutExposeAnnotation()
   * serializeNulls()
   * setDateFormat(DateFormat.LONG)
   * setFieldNamingPolicy(FieldNamingPolicy.IDENTITY)
   *
   * @return
   */
  public Gson getDefaultJSONConverter() {
    return DEFAULT_GSON;
  }

  public CommandLineOptions getConfig() {
    return OPTIONS;
  }

  public TemplateEngine getTemplateEngine() {
    return FREEMARKER_TEMPLATE_ENGINE;
  }

  /**
   *
   * @param args
   */
  public static void main(String[] args) {
    Application app = Application.getInstance();

    app.setApplicationConfiguration(args).run();
  }

  public static Application getInstance() {
    return LazyHolder.INSTANCE;
  }

  private static class LazyHolder {

    private static final Application INSTANCE = new Application();

  }

}
```