|  |  |  |
| --- | --- | --- |
| **Διπλωματική**  **Εργασία** | |  |
|  | Εξόρυξη δεδομένων από κοινωνικά δίκτυα και τον Παγκόσμιο Ιστό με σκοπό την δημιουργία εφαρμογής με θέμα την περιήγηση στις παραλίες της Ελλάδας | |

Καψάλης Παναγιώτης

**Α.Μ. : 4999**

|  |  |
| --- | --- |
| *Η παρούσα εργασία εκπονήθηκε σε συνεργασία με το Εργαστήριο Γραφικών πολυμέσων και Γεωγραφικών Συστημάτων του Τμήματος Μηχανικών Η/Υ & Πληροφορικής.* | Lab_Logo_Trans |

Πίνακας Περιεχομένων

[Πίνακας Περιεχομένων 4](#_Toc460594356)

[Πίνακας Σχημάτων 8](#_Toc460594357)

[1. Εισαγωγή 10](#_Toc460594358)

[1.1. Περιγραφή 10](#_Toc460594359)

[1.2. Παγκόσμιος Ιστός 10](#_Toc460594360)

[1.3. Κοινωνικά Δίκτυα 11](#_Toc460594361)

[1.3.1. Ορισμός 11](#_Toc460594362)

[1.3.2. Foursquare 11](#_Toc460594363)

[1.4. Mobile Εφαρμογες 12](#_Toc460594364)

[1.4.1. Native υλοποίηση 13](#_Toc460594365)

* [Πλεονεκτήματα 13](#_Toc460594366)
* [Μειονεκτήματα 14](#_Toc460594367)

[1.4.2. Cross-Platform υλοποίηση 14](#_Toc460594368)

* [Πλεονεκτήματα 15](#_Toc460594369)
* [Μειονεκτήματα 15](#_Toc460594370)

[2. Crawlers 17](#_Toc460594371)

[2.1. Τι είναι 17](#_Toc460594372)

[2.2. Πολιτικές Ιχνηλάτησης 19](#_Toc460594373)

[2.3. Πολιτική Επιλογής Περιεχομένου 20](#_Toc460594374)

[2.4. Πολιτική Καλής Συμπεριφοράς 20](#_Toc460594375)

[2.5. Πολιτική Παραλληλισμού 21](#_Toc460594376)

* [Πολιτική Δυναμικής Εκχώρησης 21](#_Toc460594377)
* [Πολιτική Στατικής Εκχώρησης 21](#_Toc460594378)

[2.6. Προβλήματα 21](#_Toc460594379)

[3. Tools & Frameworks 22](#_Toc460594380)

[3.1. Scrapy 22](#_Toc460594381)

[3.1.1. Αρχιτεκτονική του Scrapy 22](#_Toc460594382)

* [ScrapyEngine 22](#_Toc460594383)
* [Scheduler 23](#_Toc460594384)
* [Downloader 23](#_Toc460594385)
* [Spiders 23](#_Toc460594386)
* [Item Pipeline 23](#_Toc460594387)
* [Spider Middleware 24](#_Toc460594388)
* [Downloader Middleware 24](#_Toc460594389)

[3.1.2. Δημιουργία ενός Scrapy πρότζεκτ 25](#_Toc460594390)

* [Καθορισμός του αντικειμένου ιχνηλάτευσης 25](#_Toc460594391)
* [Spider 26](#_Toc460594392)

[3.2. Portia Scrapinghub 28](#_Toc460594393)

* [Scrapinghub API 33](#_Toc460594394)

[3.3. Δημιουργία εφαρμογής με χρήση του Xamarin 34](#_Toc460594395)

[3.3.1. Shared Projects 35](#_Toc460594396)

* [Πλεονεκτήματα 36](#_Toc460594397)
* [Μειονεκτήματα 36](#_Toc460594398)

[3.3.2. Portable Class Libraries 36](#_Toc460594399)

* [Πλεονεκτήματα 37](#_Toc460594400)
* [ Δημιουργία και έλεγχος πρότζεκτ το οποίο μπορεί να χρησιμοποιηθεί από άλλες βιβλιοθήκες και εφαρμογές 37](#_Toc460594401)
* [Μειονεκτήματα 37](#_Toc460594402)

[3.3.3. Cross-Platform Data Access 38](#_Toc460594403)

* [Πότε χρησιμοποιείται μια βάση δεδομένων 38](#_Toc460594404)
* [Πλεονεκτήματα χρήσης Βάσης Δεδομένων 39](#_Toc460594405)
* [Μηχανισμός SQLite 39](#_Toc460594406)
* [Χρησιμοποιώντας το SQLite.NET 39](#_Toc460594407)
* [SQLiteΧαρακτηριστικά 41](#_Toc460594408)
* [Ερωτήματα σε μια SQLite βάση 41](#_Toc460594409)

[4. Σχεδιασμός και Υλοποίηση της βάσης γνώσης 42](#_Toc460594410)

[4.1. Βάση Δεδομένων 42](#_Toc460594411)

[4.1.1. Διάγραμμα E-R και πίνακες 42](#_Toc460594412)

* [Πίνακας beach 43](#_Toc460594413)
* [Πίνακας beachplus 43](#_Toc460594414)
* [Πίνακας description 44](#_Toc460594415)
* [Πίνακας photos 45](#_Toc460594416)
* [Πίνακας tsek 45](#_Toc460594417)

[4.1.2. Εξαγωγή δεδομένων από Ιστοσελίδες 45](#_Toc460594418)

* [Wondergreece.gr 46](#_Toc460594419)
* [terrabook.com 48](#_Toc460594420)
* [Foursquare 50](#_Toc460594421)
* [Geonames.org 52](#_Toc460594422)
* [Υπόλοιποι Ιστότοποι 53](#_Toc460594423)

[4.1.3. Ταξινόμηση της Πληροφορίας 54](#_Toc460594424)

* [Συνάρτηση similar 55](#_Toc460594425)

[5. Υλοποίηση Πληροφοριακού Συστήματος 57](#_Toc460594426)

[5.1. Αρχιτεκτονική Πληροφοριακού Συστήματος 57](#_Toc460594427)

* [get\_beach 58](#_Toc460594428)
* [GetAllItems 58](#_Toc460594429)
* [getCoordinates 59](#_Toc460594430)
* [getAlternateNames 59](#_Toc460594431)
* [getAddress 60](#_Toc460594432)
* [getDetails 60](#_Toc460594433)
* [getPhotos 61](#_Toc460594434)
* [getRegionBeaches 61](#_Toc460594435)
* [getNearBeaches 62](#_Toc460594436)

[5.2. Cross-Platform Mobile Application 62](#_Toc460594437)

[5.2.1. Παράμετροι της Εφαρμογής 62](#_Toc460594438)

[5.2.2. Αρχική Οθόνη της εφαρμογής 63](#_Toc460594439)

[5.2.3. NEAR TO ME 66](#_Toc460594440)

[5.2.4. SEARCH 67](#_Toc460594441)

[5.2.5. REGIONS 68](#_Toc460594442)

[5.3. Δεύτερη Οθόνη της Εφαρμογής 71](#_Toc460594443)

[5.3.1. PHOTOS 72](#_Toc460594444)

[5.3.2. DESCRIPTION 74](#_Toc460594445)

[5.3.3. COMMENTS 77](#_Toc460594446)

[5.3.4. MAP 79](#_Toc460594447)

[5.3.5. Offline χρήση της εφαρμογής 80](#_Toc460594448)

* [beach.cs 80](#_Toc460594449)
* [PhotoSQLite.cs 81](#_Toc460594450)
* [RegionSQLite.cs 82](#_Toc460594451)
* [NearSQLite.cs 83](#_Toc460594452)
* [LatLngSQLite.cs 84](#_Toc460594453)
* [Αρχείο κώδικα LatLngSQLite.cs 85](#_Toc460594454)
* [AlterNamesSQLite.cs 85](#_Toc460594455)
* [AddressSQLite.cs 86](#_Toc460594456)
* [CommentsSQLite.cs 87](#_Toc460594457)
* [descriptionSQLite.cs 87](#_Toc460594458)

[6. Εγχειρίδιο Εφαρμογής 90](#_Toc460594459)

[7. Παράρτημα 97](#_Toc460594460)

* [App7 /MyClass.cs 97](#_Toc460594461)
* [App7.Droid/ MainActivity.cs 103](#_Toc460594462)
* [App7.Droid/ ItemContext.cs 105](#_Toc460594463)
* [App7. Droid/Fragment1.cs 106](#_Toc460594464)
* [App7.Droid/ Fragmen2.cs 108](#_Toc460594465)
* [App7.Droid/ RegionActivity.cs 111](#_Toc460594466)
* [App7.Droid/ ShowMap.cs 112](#_Toc460594467)
* [App7.Droid/ Class1.cs 113](#_Toc460594468)
* [App7.Droid/ CustomAdapter.cs 114](#_Toc460594469)
* [App7.Droid/ NearAdapter.cs 115](#_Toc460594470)
* [App7.Droid/ RegionAdapter.cs 116](#_Toc460594471)
* [Patmosplus.py 117](#_Toc460594472)
* [skiathosplus.py 118](#_Toc460594473)
* [zanteplus.py 119](#_Toc460594474)
* [thasosplus.py 120](#_Toc460594475)
* [cleanplus.py 121](#_Toc460594476)
* [run\_me.py 122](#_Toc460594477)
* [ids.py 122](#_Toc460594478)
* [beach\_search.py 123](#_Toc460594479)
* [geo.py 124](#_Toc460594480)
* [distance.py 125](#_Toc460594481)
* [para.py 125](#_Toc460594482)
* [detect\_latin.py 125](#_Toc460594483)
* [similar\_names.py 126](#_Toc460594484)
* [api.py 129](#_Toc460594485)
* [fillInfo.py 132](#_Toc460594486)
* [passenger\_wsgi.py 133](#_Toc460594487)

[8. Βιβλιογραφία 136](#_Toc460594488)

Πίνακας Σχημάτων

[Εικόνα 1 Χρόνος που καταναλώνεται ανά πλατφόρμα 12](#_Toc461117859)

[Εικόνα 2 Χρόνος που καταναλώνεται ανά πλατφόρμα 13](#_Toc461117860)

[Εικόνα 3 Native vs Cross Platform 16](#_Toc461117861)

[Εικόνα 4 Η δομή ενός φυλλομετρητή 18](#_Toc461117862)

[Εικόνα 5 Αρχιτεκτονική web φυλλομετρητή 18](#_Toc461117863)

[Εικόνα 6 Αρχιτεκτονική του Scrapy 24](#_Toc461117864)

[Εικόνα 7 Εγκατάσταση του Scrapy 25](#_Toc461117865)

[Εικόνα 8 Δημιουργία Scrapy project 25](#_Toc461117866)

[Εικόνα 9 Δημιουργία του nettuts project 25](#_Toc461117867)

[Εικόνα 10 Δομή φακέλου nettuts project 25](#_Toc461117868)

[Εικόνα 11 items.py 26](#_Toc461117869)

[Εικόνα 12 Πεδία του NettutsItem 26](#_Toc461117870)

[Εικόνα 13 Δηλώσεις των κλάσεων 26](#_Toc461117871)

[Εικόνα 14 name, allowed\_domains, start\_urls 27](#_Toc461117872)

[Εικόνα 15 class MySpider 27](#_Toc461117873)

[Εικόνα 16 Δημιουργία λογαριασμού στο Scrapinghub 28](#_Toc461117874)

[Εικόνα 17 Δημιουργία πρότζεκτ στο Portia Scrapinghub 29](#_Toc461117875)

[Εικόνα 18 portia new spider 29](#_Toc461117876)

[Εικόνα 19 Crawling follow patterns 30](#_Toc461117877)

[Εικόνα 20 Annotate this page 30](#_Toc461117878)

[Εικόνα 21 Extracted Items 31](#_Toc461117879)

[Εικόνα 22 Επιλογή Ονόματος 31](#_Toc461117880)

[Εικόνα 23 Επιλογή Περιγραφής 32](#_Toc461117881)

[Εικόνα 24 Extracted Sample 32](#_Toc461117882)

[Εικόνα 25 Running – Completed Jobs 33](#_Toc461117883)

[Εικόνα 26 Install shub 33](#_Toc461117884)

[Εικόνα 27 Διαμοιρασμός κώδικα μεταξύ πλατφόρμων 34](#_Toc461117885)

[Εικόνα 28 Tasky solution file 35](#_Toc461117886)

[Εικόνα 29 Αρχιτεκτονική Tasky project 35](#_Toc461117887)

[Εικόνα 30 Επιλογή χαρακτηριστικών PCL 37](#_Toc461117888)

[Εικόνα 31 Αρχιτεκτονική cross platform εφαρμογής που χρησιμοποιεί PCL για τον διαμοιρασμό κώδικα 38](#_Toc461117889)

[Εικόνα 32 SQLite.net PCL NuGet package 40](#_Toc461117890)

[Εικόνα 33 SQLite.net Xamarin Component Store 40](#_Toc461117891)

[Εικόνα 34 Using statement 40](#_Toc461117892)

[Εικόνα 35 Database file path 40](#_Toc461117893)

[Εικόνα 36 Store Data 41](#_Toc461117894)

[Εικόνα 37 Retrieve Data 41](#_Toc461117895)

[Εικόνα 38 Διάγραμμα Συσχχετίσεων - Οντοτήτων 42](#_Toc461117896)

[Εικόνα 39 Πίνακας beach 43](#_Toc461117897)

[Εικόνα 40 Πίνακας beachplus 43](#_Toc461117898)

[Εικόνα 41 Πίνακας descrption 44](#_Toc461117899)

[Εικόνα 42 Πίνακας photos 45](#_Toc461117900)

[Εικόνα 43 Πίνακας tsek 45](#_Toc461117901)

[Εικόνα 44 Δημιουργία Εφαρμογής Foursquare 50](#_Toc461117902)

[Εικόνα 45 Στοιχεία της Εφαρμογής 50](#_Toc461117903)

[Εικόνα 46 Client secret – Client id 50](#_Toc461117904)

[Εικόνα 47 Δημιουργία λογαριασμού στο geonames.org 52](#_Toc461117905)

[Εικόνα 48 Scrapinghub API key 54](#_Toc461117906)

[Εικόνα 49 Αρχιτεκτονική Πληροφοριακού Συστήματος 57](#_Toc461117907)

[Εικόνα 50 virtual environment 63](#_Toc461117908)

[Εικόνα 51 activate virtual enviroment 63](#_Toc461117909)

[Εικόνα 52 restart server 63](#_Toc461117910)

[Εικόνα 53 Αρχική οθόνη της εφαρμογής 65](#_Toc461117911)

[Εικόνα 54 NEAR TO ME tag 68](#_Toc461117912)

[Εικόνα 55 SEARCH (a) 69](#_Toc461117913)

[Εικόνα 56 REGIONS 70](#_Toc461117914)

[Εικόνα 57 Δεύτερη οθόνη της εφαρμογής 72](#_Toc461117915)

[Εικόνα 58 DESCRIPTION tag 75](#_Toc461117916)

[Εικόνα 59 COMMENTS tag 76](#_Toc461117917)

[Εικόνα 60 MAP tag 78](#_Toc461117918)

[Εικόνα 61 Εκκίνηση της Εφαρμογής 89](#_Toc461117919)

[Εικόνα 62 NEAR TO ME (a) 90](#_Toc461117920)

[Εικόνα 63 NEAR TO ME (b) 90](#_Toc461117921)

[Εικόνα 64 SEARCH 91](#_Toc461117922)

[Εικόνα 65 REGIONS 92](#_Toc461117923)

[Εικόνα 66 Οθόνη Παραλίας 93](#_Toc461117924)

[Εικόνα 67 Πληροφορίες 94](#_Toc461117925)

[Εικόνα 68 Σχόλια 95](#_Toc461117926)

[Εικόνα 69 Χάρτης 95](#_Toc461117927)

1. Εισαγωγή

# Περιγραφή

Τα τελευταία χρόνια μεγάλη ανάπτυξη και δημοτικότητα έχουν γνωρίσει τα smartphones τα οποία ενσωματώνουν τόσο τις δυνατότητες ενός απλού τηλεφώνου, δηλαδή την πραγματοποίηση κλήσεων, αποστολή γραπτών μηνυμάτων, αλλά και δυνατότητες που μέχρι πρότινος ήταν χαρακτηριστικά καθαρά και μόνο των ηλεκτρονικών υπολογιστών, όπως πρόσβαση στο διαδίκτυο, γραφικά περιβάλλοντα εργασίας, αναπαραγωγή πολυμέσων όπως αναπαραγωγή ήχου, αναπαραγωγή ταινιών. Έτσι παρατηρούμε πλήθος εφαρμογών που ενσωματώνουν εικόνα, ήχο, video, χρήση διαδικτύου. Με αυτόν τρόπο ενισχύεται η ανάγκη του ανθρώπου για πληροφορία καθώς και την ανάγκη της διαχείρισης της πληροφορίας από τον άνθρωπο εύκολα και γρήγορα.

Στα πλαίσια της διπλωματικής αυτής αναπτύχθηκε μια εφαρμογή , που συγκεντρώνει πληροφορία από το κοινωνικό δίκτυο Foursquare καθώς και από άλλες σελίδες του Διαδικτύου. Εφόσον γίνει η εξόρυξη της πληροφόρίας, έπεται η αποθήκευσή της σε μία MySQL βάση δεδομένων, τα στοιχεία που εξάγωνται είναι το όνομα της παραλίας, το γεωγραφικό μήκος και πλάτος της τοποθεσίας που βρίσκεται η παραλία, η διεύθυνση, η περιοχή της παραλίας καθώς και περιγραφές και σχόλια των χρηστών των σελίδων του διαδικτύου και του Foursquare. Web-services επικοινωνούν με την βάση με σκοπό την διοχέτευση των πληροφοριών που συλλέξαμε στην εφαρμογή. Αρχικά παρουσιάζεται μια λίστα των παραλιών της Ελλάδας, στην συνέχεια με την ενργοποίηση του GPS του κινητού παρουσιάζονται οι 40 πιο κοντινές στον χρήστη παραλίες. Επίσης υπάρχει μπάρα αναζήτησης των παραλιών, καθώς και χάρτης που δείχνει τις πιο κοντινές στον χρήστη παραλίες.

Πατώντας πάνω σε μία παραλία εμφανίζονται εικόνες της παραλίας, χάρτης , η διεύθυνση της σχόλια από χρήστες του Foursquare ή άλλων σελίδων και περιγραφή για αυτή.

# Παγκόσμιος Ιστός

Ο Παγκόσμιος Ιστός είναι ένα ανοιχτό σύστημα διασυνδεδεμένων πληροφοριών και πολυμεσικού περιεχομένου, που επιτρέπει στους χρήστες του Διαδικτύου να αναζητήσουν πληροφορίες μεταβαίνοντας από ένα έγγραφο σε ένα άλλο.

Κάθε δίκτυο-δομική μονάδα του διαδικτύου αποτελείται από συνδεδεμένους υπολογιστές σε τοπικό επίπεδο. Αυτά τα δίκτυα με την σειρά τους συνδέονται με ευρύτερα δίκτυα. Το ευρύτερο δίκτυο στον κόσμο λέγεται Παγκόσμιος Ιστός το οποίο είναι μοναδικό και συμπεριλαμβάνεται τόσο τα γήινα δίκτυα όσο και τα δίκτυα δορυφόρων που είναι συνδεδεμένα σε αυτό. Η τεχνολογία του Ιστού καθιστά δυνατή την δημιουργία «υπερκειμένων», μια διασύνδεση δηλαδή πάρα πολλών μη ιεραρχημένων στοιχείων.

Η Επιστήμη του Παγκόσμιου Ιστού αποτελεί μια ερευνητική πρωτοβουλία του SirTimothyBerners-Lee.

# Κοινωνικά Δίκτυα

## Ορισμός

Ένα κοινωνικό δίκτυο είναι μια κοινωνική δομή, που συμπεριλαμβάνει κόμβους (φυσικά πρόσωπα ή οργανισμοί) οι οποίοι συνδέονται με έναν ή περισσότρους παρόμοιου τύπου κόμβους.

## Foursquare

Το Foursquare είναι μια mobile social εφαρμογή, η οποία προτείνει στους χρήστες να επισκευφθούν τοποθεσίες (**venues**) βάσει των **γεύσεών** τους και των **check-in** τους. Το Foursquareδιαχωρίζεται σε 2 ανεξάρτητες εφαρμογές το Foursquare και το Swarm η οποία χρησιμοποιείται για check-in.

Με πάνω από 50 εκατομμύρια χρήστες και 6 δισεκατομμύρια check-in ο συνδιασμός Foursquare και Swarm μπορεί να προσφέρει στους χρήστες τη δυνατότητα να δηλώνουν τις τοποθεσίες που επισκέφθηκαν μέσω check-in, να βλέπουν την δραστηριότητα των φίλων τους, να δέχονται προτάσεις για επίσκεψη σε νέες τοποθεσίες να διαβάζουν tips χρηστών να αξιολογούν, να βλέπουν φωτογραφίες των τοποθεσιών και άλλες χρήσιμες πληροφορίες.

Κάθε χρήστης έχει το δικό του προφίλ, στο οποίο μπορεί να δηλώνει τις **γεύσεις** του, δηλαδή όσα του αρέσουν (φαγητά, διασκέδαση, κ.λ.π). Με τον τρόπο αυτό η εφαρμογή μπορεί να του προτείνει καλύτερα τοποθεσίες οι οποίες θα του αρέσουν. Επίσης κάθε χρήστης έχει την δυνατότητα να ακολολουθεί άλλους χρήστες για να διαβάζει **tips** (σχόλια των χρηστών τα οποία αφήνουν σε μέρη που επισκέπτονται).

Το Foursquare έχει σχεδιαστεί με τέτοιον τρόπο, ώστε από την αρχική κιόλας σελίδα να μπορεί ο χρήστης να αναζητήσει τοποθεσίες ανάλογα με την ώρα, έχοντας έτοιμες επιλογές προς αναζήτηση και πεδίο αναζήτησης. Η αναζήτηση περιέχει πολλά φίλτρα όπως τιμές ωράριο λειτουργίας κ.λ.π. Επίσης η εφαρμογή μπορεί να προτείνει στους χρήστες τοποθεσίες οι οποίες βρίσκονται τριγύρω από την τοποθεσία τους.

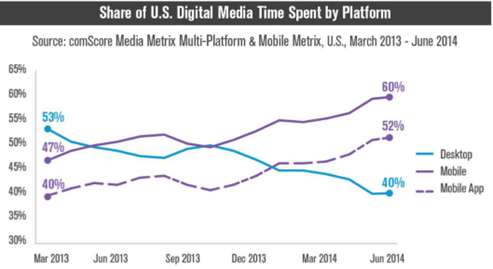
Περνώντας στην εφαρμογή Swarm, ο χρήστης μπορεί να έχιε πάλι το δικό του προφίλ, μέσω του οποίου μπορεί να βλέπει που βρίσκονται οι φίλοι του βάσει των check-inτους ή της λειτουργίας **neighborhood sharing** η οποία δείχνει στο περίπου που βρίσκονται οι χρήστες χωρίς check-in. Η εφαρμογή Swarm δίνει δυνατότητα στους χρήστες να κάνουν check-in, και μάλιστα η λειτουργία check-in εμφανίζεται και στην εφαρμογή Foursquare με εγκατάσταση του Swarm, να διαμοιράζονται την τοποθεσία τους να σχεδιάζουν εξόδους και να βλέπουν το ιστορικό τους.

Το Foursquare δίνει την δυνατότητα για αλληλεπίδραση με την παγκόσμιας κλάσης βάση δεδομένων τοποθεσιών. Μέσω των endpoints του μπορούμε να κάνουμε αιτήσεις προς την βάση δεδομένων του Foursquareέχοντας ως εισόδους τις παραμέτρους **clientID** και **clientsecret**. Μπορούμε να κάνουμε αναζήτηση για μέρη (venues) σε μια περιοχή καθώς και να δούμε τις φωτογραφίες που έχουν ανεβάσει οι χρήστες για αυτό το μέρος , τα σχόλια (tips) που έχουν κάνει κ.λ.π. Κάθε μέρος venue συνδέεται με ένα url , χρησιμοποιώντας τα endpoints που μας παρέχει το Foursquare Api μπορούμε να λάβουμε στοιχεία για αυτό το μέρος δίνοντας στο url την παράμετρο του αντίστοιχου endpoint που θέλουμε ως έξοδο.

Για να γίνει αναζήτηση τοποθεσιών (venues) σε μια περιοχή χρησιμοποιούνται οι μέθοδοι **explore** και **search**. Η μέθοδος explore είναι καλύτερη στο να απαντήσει σε ερωτήσεις του τύπου ”ποιες είναι οι πιο δημοφηλείς καφετέριες?” ενώ η search είναι καλύτερη στο να απαντήσει ερωτήσεις του τύπου “ποιο είναι το κοντινότερο CoffeeIsland?”. Κάθε χρήστης έχει δικαίωμα να κάνει μέχρι 5000 αναζητήσειςτην ώρα.

Για παροχή μιας πιο πλούσιας εμπειρίας, είναι απαραίτητη η σύνδεση των χρηστών μιας εφαρμογής που κάνει αιτήσεις στο Foursquare Api, με τους προσωπικούς λογαριασμούς τους στο Foursquare .Για να γίνει αυτή είναι απαραίτητο ένα διακριτικό πρόσβασης (accesstoken) για κάθενα χρήστη.

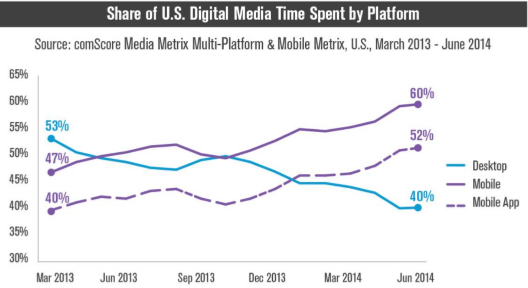
# Mobile Εφαρμογες



Εικόνα 1 Χρόνος που καταναλώνεται ανά πλατφόρμα

Τον Αύγουστο του 2014 διεξήχθη στις Ηνωμένες Πολιτείες της Αμερικής μια έρευνα η οποία έδειξε ότι οι Αμερικάνοι πολίτες ξοδεύουν το μεγαλύτερος μέρος του χρόνου τους καταναλώνοντας ψηφιακές υπηρεσίες μέσω mobile εφαρμογών. Αυτό σημαίνει ότι οι mobile εφαρμογές μας τρώνε πάνω από το 52% του χρόνου που καταναλώνουμε σε ψηφιακά μέσα, περισσότερο ακόμα και από το χρόνο που ξοδεύουμε σε desktop εφαρμογές αλλά και σε mobile websurfing. Σε συνδυασμό με το mobile web, η χρήση κινητών συσκευών στο σύνολο της αγγίζει το 60% του χρόνου που δαπανάται σε ψηφιακά μέσα, αφήνοντας το 40% στη χρήση desktop-based εφαρμογών.

Όπως παρατηρούμε από την εικόνα 1, η χρήση mobile εφαρμογών έχει ανοδική πορεία εις βάρος των desktop εφαρμογών. Είναι φυσικό η ανάγκη για mobile development ολοένα και να αυξάνεται. Με τόσες όμως διαφορετικές συσκευές που κυκλοφορούν κάθε μία με το δικό της λειτουργικό, αρχιτεκτονική, απαιτήσεις και προφανώς με το δικό της μερίδιο χρηστών, ο δημιουργός μιας εφαρμογής καλείται να αντιμετωπίσει ένα δίλλημα να ακολουθήσει native ή cross-platform υλοποίηση.



Εικόνα 2 Χρόνος που καταναλώνεται ανά πλατφόρμα

## Native υλοποίηση

Υπάρχουν πάρα πολλές συσκευές στις οποίες μπορεί μια εφαρμογή να τρέξει όπως Android, WindowsPhone , Blackberryκαι πολλές άλλες κάθε μία με τα δικά της χαρακτηριστικά, αρχιτεκτονική, λειτουργικό, επιδόσεις, γραφικά στοιχεία, μέγεθος οθόνης κ.λ.π. Αν μια εφαρμογή είναι nativeσημαίνει ότι έχει δημιουργηθεί με συγκεκριμένα εργαλέια και γλώσσα και μπορεί να εκτελεστεί μόνο στην πλατφόρμα αυτή που αρχικά έχει σχεδιαστεί..Αναγκαστικά λοιπόν απαιτείται η δημιουργία μιας δεύτερης εφαρμογής από την αρχή ώστε να δουλέψει σε άλλη πλατφόρμα.

### Πλεονεκτήματα

* Βελτιστοποίηση: Αναπτύσσοντας μια native mobile εφαρμογή για το λειτουργικό στο οποίο πρόκειται να χρησιμοποιηθεί επιτυγχάνονται αυτόματα βέλτιστες αποδόσεις σε ταχύτητα, απρόσκοπη λειτουργία όλων των αισθητήρων, των λειτουργιών της συσκευής κλπ.
* Πρόσβαση σε ενημερώσεις: Μέρα με τη μέρα κάθε πλατφόρμα παρουσιάζει και ένα καινούργιο χαρακτηριστικό που δεν υπήρχε μέχρι πρότινος. Ακολουθώντας native στρατηγική υπάρχψει η δυνατότητα να γίνει χρήση όλων των τελευταίων χαρακτηριστικών του εκάστοτε λειτουργικού συστήματος.
* Υποστήριξη: Ακόμη και όταν υπάρξουν σφάλματα, υπάρχει η δυνατότητα υποστήριξης μέσα από το επίσημο support team της κατασκευάστριας εταιρείας.
* Μεγαλύτερη Ευελιξία: Το γεγονός ότι το native design δίνει σε κάποιον την δυνατότητα να εκμεταλευτεί στο έπακρο όλα τα χαρακτηριστικά της συσκευής του και του λειτουργικού της συστήματος , του επιτρέπει να μπορέσει να καλύψει ακόμα και τις πλέον πρωσοποιημένες ανάγκες του.
* Γρήγορος Κώδικας: Ο κώδικας τρέχει πιο γρήγορα σε μια native εφαρμογή από ότι η JavaScript που γενικά είναι πιο αργή γλώσσα.
* Documentation: Υπάρχουν πάνω από 2500 βιβλία μόνο για τον προγραμματισμό σε Android και πολυ περισσότερα topics σε φορουμ όπως το StackOverflow. Αυτό καθιστά την λύση ενός προβλήματος πιο εύκολη.
* Ευκολία στην χρήση: Η Native πλατφόρμα είναι αυτό που το κοινό έχει συνηθίσει, οπότε το τελικό προιόν καθιστά μια πιο εύχρηστη εφαρμογή.

### Μειονεκτήματα

* Μεγαλύτερο Κόστος: Στο native design, κάθε έκδοση της εφαρμογής που αντοιστοιχεί σε διαφορετικό λειτουργικό , θεωρείται μια καινούργια εφαρμογή. Αυτό σημαίνει ότι αν θέλετε η εφαρμογή σας να υλοποιηθεί σε Android και σε Windows Phone πρακτικά ζητάτε τη δημιουργία 2 διαφορετικών εφαρμογών.
* Μεγαλύτερος Χρόνος παράδοσης: Όπως και με το κόστος , ο χρόνος παράδοσης σε native design είναι πολλαπλάσιος του αριθμού των λειτουργικών που θέλετε να υποστηρίζει η εφαρμογή σας.
* Δύσκολο Development: Το επίπεδο των γνώσεων και της εμπειρίας που χρειάζεται για να φτιαχτεί μια native εφαρμογή είναι αρκετά υψηλά. Αν δε το τελικό προιόν πρέπει να λειτουργεί σε πολλαπλές πλατφόρμες, επόμενο είναι το τεχνολογικό υπόβαθρο που χρειάζεται να είναι αρκετά απαιτητικό, ειδικά για κάποιον νέο προγραμματιστή.
* Εξάρτηση από τα tools: Για να κατασκευάσει κανείς μια native εφαρμογή δεν έχει πολλά περιθώρια να αποκλίνει όσον αφορά τα tools και τη γλώσσα που θα χρησιμοποιήσει. Αναγκστικά θα χρησιμοποιήσει τα IDEs που κάθε πλατφόρμα προσφέρει. Για παράδειγμα για να υλοποιήσει κάποιος εφαρμογή για smartphone της Apple είναι αναγκασμένος να το κάνει αποκλειστικά με χρήση υπολογιστή που έχει κατασκευάσει η Apple.

## Cross-Platform υλοποίηση

Με τον όρο cross-platform εννοούμε κατά βάση web-based εφαρμογές βασισμένες κυρίως σε τεχνολογίες όπως HTML5, JavaScript, CSS3, οι οποίες είναι συμβατές με την πλειοψηφία των ηλεκτρονικών συσκευών που κυκλοφορούν. Υπάρχουν 2 είδη τέτοιων εφαρμογών:

Mobilewebsites: Πρόκειται για websites που η δομή και η εμφάνιση της εφαρμογής υλοποιείται με την χρήση ειδικών browserwindows αντί για τα native Uiframework της κάθε πλαφόρμας. Επίσης γίνεται χρήση responsive CSS εργαλείων ώστε το interface να αλλάζει δυναμικά, ανάλογα με το μέγεθος της οθόνης. Επιτυγχάνεται έτσι ομοιόμορφη αίσθηση και εμφάνιση ανεξαρτήτως μεγέθους συσκευής. Το κακό με αυτή την υλοποίηση είναι ότι δεν μπορεί η εφαρμογή να αποκτήσει πρόσβαση σε επιμέρους στοιχεία της συσκευής όπως η κάμερα, το gps, η δόνηση, η εσωτερική μνήμη της συσκευής κλπ. Η πρόσβαση όμως στην εφαρμογή γίνεται μέσω του browser που διαθέτει ο χρήστης.

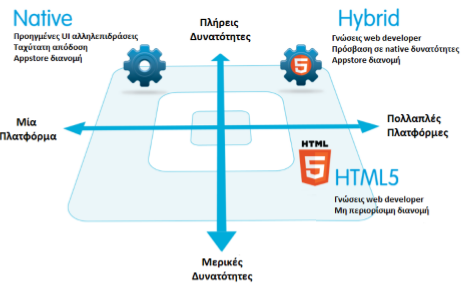
Υβριδικές web εφαρμογές: Οι εφαρμογές αυτές είναι ένας συνδυάσμος των native εφαρμογών και των mobile websites που αναλύσαμε παραπάνω. Γίνεται χρήση των Native UI εργαλείων μαζί με WebViews και ταυτόχρονα υπάρχει και πρόσβαση σε χρήσιμα device level APIs μέσω δίαφορων plugins. Επίση επειδή διατηρούνται τα χαρακτηριστικά ενός website διατηρείται και το responsiveness, άρα η εναλλαγή από μικρές σε μεαγαλύτερες οθόνες/συσκευές δεν αποτελεί πρόβλημα. Η πρόσβαση στην εφαρμογή γίνεται από τον χρήστη κανονικά αφού εγκαταστήσει στην συσκευή του το εκάστοτε appstore.

### Πλεονεκτήματα

* Μια υλοποίηση – μεγαλύτερη κάλυψη: Αναπτύσοντας μια φορά την εφαρμογή μπορούμε να την δημοσιεύσουμε σε πολλαπλά marketplaces. Αυτό κάνει τη διάδοση της εφαρμογής μας ευκολότερη με αποτέλεσμα παράλληλα να επιτυγχάνουμε καλύτερο marketing.
* Μεγαλύτερο πελατολόγιο: Όπως είναι φυσικό με περισσότερες πλατφόρμες συμβατές με την εφαρμογή σου καταφέρνεις να απευθυνθείς σε μεγαλύτερο μερίδιο αγοράς. Αν δε οι πλατφόρμες αυτές είναι και οι δημοφιλέστερες Android και Ios τότε σου δίνουν ένα πρόσθετο πλεονέκτημα.
* Χρήση γνωστών εργαλείων – μικρότερη εξειδίκευση: Τα εργαλεία για cross platform development βασίζονται σε ταχνολογίες όπως HTML5, JavaScript, CSS3 κλπ, τις ίδιες τεχνολογίες δηλαδή που χρησιμοποιούνται στο webdevelopment. Η γκάμα επαγγελματιών που μπορούν να υποστηρίξουν το project διευρύνεται σημαντικά. Ταυτόχρονα είναι πιο εύκολο για κάποιον να ξεκινήσει μιαweb-basedεφαρμογή αντί για μια native, μιας και χρειάζεται λιγότερη εξειδίκευση.
* Χαμηλό Κόστος: Όπως είπαμε παραπάνω , με το cross platform development υλοποιούμε μια φορά. Η ύπαρξη μιας μονο έκδοσης της εφαρμογής μας που καλύπτει πολλά διαφορετικα marketplaces μειώνει σημαντικά το κόστος υλοποίησης.
* Μικρότερος χρόνος παράδοσης: Με μια εφαρμογή αντί για πολλές, ο φόρτος εργασίας μειώνεται, επομένως το έργο υλοποιείται πιο γρήγορα.
* Ευκολότεροε debugging: Το debugging είναι μια εξαιρετικά επίπονη διαδικασία οπότε όταν έχεις μια εφαρμογή αντί για πολλές να επιδιορθώσεις ή να υποστηρίξεις μετά την παράδοση προφανώς τα πράγματα γίνονται πιο εύκολα.
* Επικοινωνία μεταξύ των developers: Είναι ευκολότερο να συγχρονιστεί μια ομάδα ανθρώπων που δουλεύουν πάνω σε μια πλατφόρμα αντί για 2 ή παραπάνω που δουλεύουν ξεχωριστά.

### Μειονεκτήματα

* Περιορισμοί στην λειτουργικότητα: Στην προσπάθεια τους να καλύψουν κατά το δυνατόν περισσότερα λειτουργικά συστήματα, εκδόσεις και συσκευές, οι πλατφόρμες cross platform ανάπτυξης εφαρμογών κινητών συσκευών υλοποιούν επιλεκτικά λειτουργικότητες. Αυτό σημαίνει ότι παρόλο που έχουμε τη δυνατότητα να κάνουμε χρήση του μεγαλύτερου μέρους των εργαλείων που μας δίνει μια κινητή συσκευή , κάνοντας χρήση cross platform development ποτέ δεν θα έχουμε πρόσβαση σε όλα τα στατιστικά που μας δίνει το εκάστοτε λειτουργικό σύστημα.
* Περιορισμοί στην διεπαφή χρήστη: Διαφορετικά λειτουργικά συστήματα κινητών τηλεφώνων χρησιμοποιούν διαφορετικά μεγέθη οθονών. Τα αντίστοιχα marketplaces έχουν συντάξει μια σειρά από κανόνες που πρέπει να τηρούνται προκειμένου οι εφαρμογές να γίνουν διαθέσιμες στους χρήστες μέσω του store. Tocrossplatformdesign, προσπαθώντας να καλύψει όλες τις διαφορετικές ανάγκες, πολύ συχνά αποτυγχάνει με αποτέλεσμα να μεγαλώνει ο χρόνος υλοποίησης ώστε να μπορέσουν να καλυφθούν όλες οι ιδιαιτερότητες των διάφορων λειτουργικών συστημάτων.
* Διαφορετικές πλατφόρμες – διαφορετικές δυνατότητες: Δεν είναι μόνο το UI που διαφέρει μεταξύ των διαφορετικών ειδών από πλατφόρμες. Ακόμα και το να είναι συμβατή η εφαρμογή με τις ρυθμίσεις κάθε συσκευής , το σύστημα αρχείων που υιοθετείται και ο τρόπος με τον οποίο λειτουργούν τα διάφορα στοιχεία της συσκευής (camera, gps, accelatorsensor) τα οποία διαφέρουν επίσης, δυσχεραίνουν ακόμα περισσότερο το έργο του προγραμματιστή.
* Απώλεια ευελιξίας: Κάθε πλατφόρμα προσφέρει τις δικιές της ευελιξίες. Όταν προσπαθεί κανείς να σχεδιάσει μια τέτοια εφαρμογή αναγκάζεται να ψάχνει για τις ομοιότητες, χάνοντας έτσι τις ευελιξίες που θα είχε στη διάθεση του. Προσπαθώντας λοιπόν να φτιάξει μαι εφαρμογή που θα είχε ομοιόμορφη εμφάνιση και λειτουργίες σε κάθε πλατφόρμα καταλήγει να κάνει συμβιβασμούς εκεμταλλευόμενος μόνο τα κοινά χαρακτηριστικά και όχι τις ιδιαιτερότητες της κάθε πλατφόρμας.



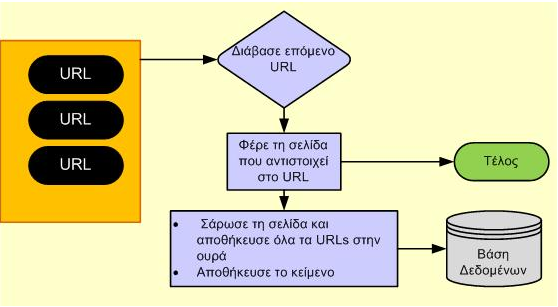
Εικόνα 3 Native vs Cross Platform

1. Crawlers

# Τι είναι

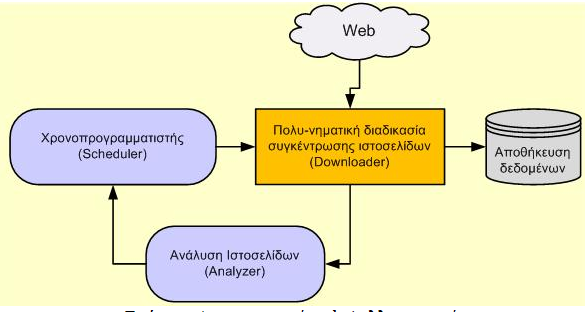
Τα δεδομένα που θα χειρίζεται η εφαρμογή τα βρίσκουμε σε διάφορα sites ή κοινωνικά δίκτυα,σε αυτή την περίπτωση το Foursquare. Το Foursquare παρέχει api έτσι ώστε να υπάρχει μια ευκολία για εξόρυξη δεδομένων από αυτό. Άλλες όμως ιστοσελίδες που δεν διαθέτουν api, καθιστούν αναγκαία την χρήση των crawlers έτσι ώστε να γίνει εξόρυξη πληροφορίας από αυτές. Για την περάτωση της διπλωματικής επιλέχθηκε ο open source crawler Scrapy ο οποίος υλοποιείται με την γλώσσα προγραμματισμού Python, είναι πιο γρηγορος σε σχέση με άλλους ιχνηλάτες καθώς εκτελεί ασύγχρονες πράξεις.

Ένας crawler (ιχνηλάτης) είναι ένα πρόγραμμα το οποίο «φυλλομετρά» το web με ένα αυτόματο και συστηματικό τρόπο. Η διαδικασία αυτή αναφέρεται και ως crawling. Άλλοι συνώνυμοι όροι που χρησιμοποιούνται στην βιβλιογραφία είναι bots, Webspiders, Webrobots. O ιχνηλάτης αποτελεί βασικό στοιχείο μηχανών αναζήτησης, και πολλών άλλων web εφαρμογών, καθόσον από αυτόν εξαρτάται η συγκέντρωση, επικαιροποίηση αλλά και η ποιότητα των δεδομένων. Επιπλέον οι ιχνηλάτες μπορεί να χρησιμοποιηθούν για την αυτόματη συντήρηση ενός ιστότοπου , όπως για παράδειγμα για τον έλεγχο των συνδέσμων ή για την επιβεβαίωση HTML κώδικα, καθώς επίσης μπορούν να χρησιμοποιηθούν για την συλλογή συγκεκριμένου τύπου πληροφορίας από το web, όπως για παράδειγμα την συλλογή e-mail διευθύνσεων (spamming, spambot). Ένας web crawler είναι ένα είδος πράκτορα λογισμικού (bot). Γενικά ο crawler ξεκινά από μια λίστα με URL (UniformResourceLocator) για να επισκεφτεί. Αυτά τα URL ονομάζονται σπόροι (seeds). Καθώς ο crawler επισκέπτεται αυτά τα URL, εντοπίζει όλους τους υπερσυνδέσμους (hyperlinks) σε μια ιστοσελίδα και τους προσθέτει στη λίστα των URL για να τα επισκεφτεί στη συνέχεια με μια συγκεκριμένη πολιτική. Επομένως οι crawlers είναι προγράμματα που διασχίζουν τον γράφο του web ακολουθώντας τους συνδέσμους και αυτόματα κατεβάζουν τοπικά το περιεχόμενο αυτών των σελίδων. Η βασική δομή ενός crawler δίνεται στην εικόνα 3. Όπως βλεπουμε ο crawler (ιχνηλάτης) εφοδιάζεται με μια λίστα από URLs. Επισκέπτεται το πρώτο URL κατεβάζει το περιεχόμενο της σελίδας και τυχόν νέους συνδέσμους που υπάρχουν στη σελίδα οι οποίοι και προστίθενται στην αρχική ουρά των URL. Η διαδικασία αυτή επαναλαμβάνεται έως ότου εξαντληθεί η λίστα των URLs. Μια περισσότερο γενική εικόνα ενός ιχνηλάτη του web παρουσιάζεται στην εικόνα 4.



Εικόνα Η δομή ενός φυλλομετρητή

Όπως βλέπουμε και από το σχήμα ο ιχνηλάτης (crawler) αποτελείται από 4 βασικά στοιχεία: τη μονάδα ανάκτησης ιστοσελίδων μέσω του πρωτοκόλλου HTTP, (downloader), την μονάδα ανάλυσης (parsing)των ιστοσελίδων και εξαγωγή των συνδέσμων που περιέχει, (analyzer), τον χρονοπρογραμματιστή (scheduler), μονάδα διαχείρισης των URLs, και τέλος την αποθήκη που συγκεντρώνεται το περιεχόμενο των ιστοσελίδων.



Εικόνα Αρχιτεκτονική web φυλλομετρητή

Ο downloader εκτός από την ανάκτηση των ιστοσελίδων δημιουργεί και ένα ευρετήριο με τα URLs των σελίδων. Η επιλογή της σειράς με την οποία επισκέπτονται οι κόμβοι του web καθορίζεται από την μέθοδο Dequeue του χρονοπρογραμματιστή. Συνήθως η επίσκεψη γίνεται με διάσχιση κατά πλάτος (Breadth First Search). Συνήθως ο ιχνηλάτης χρησιμοποιεί πολλές διαδιακασίες καταβάσματος ιστοσελίδων ταυτόχρονα οι οποίες εκτελούνται σε διαφορετικούς επεξεργαστές. Λεπτομέρειες σχετικά με ιχνηλάτες δεν δημοσιεύονται (αποτελούν μυστικά των εταιρειών) ώστε να περιορίζονται οι κακόβουλοι χρήστες (spammers). Ένας ιχνηλάτης μπορεί να είναι εξειδικευμένος (focused) να συγκεντρώνει σελίδες που είναι σχετικές με ένα προ-καθορισμένο θεματικό πεδίο. Για παράδειγμα το CiteSeeX , μια μηχανή αναζήτησης και ηλεκτρονική βιβλιοθήκη για άρθρα στην επιστήμη των υπολογιστών χρησιμοποιεί εξειδικευμένο crawlingγια την συγκέτρωση επιστημονικών άρθρων. Τέλος ένας crawlerμπορεί να χρησιμοποιηθεί ως φίλτρο για την συγκέντρωση σελίδων κάποιου συγκεκριμένου τύπου, για παράδειγμα μόνο για HTMLσελίδες. Αυτό μπορεί να γίνει όταν ο ιχνηλάτης εξετάζει το URLκαι μόνο όταν το URLέχει κατάληξη, για παράδειγμα .html, .htm, να επιτρέπει να κατέβει το αρχείο. Η επίδοση ενός ιχνηλάτη αξιολογείται από τον ρυθμό ανάκτησης σελίδων από το web. Προφανώς ο ρυθμός αυτός έχει περιορισμούς, όπως για παράδειγμα , το εύρος των δικτύων. Συνήθως η διαδικασία αυτή γίνεται παράλληλα από πολλούς εξυπηρέτες ωστέ να μεγιστοποιηθεί ο ρυθμός της συγκέντρωσης του περιεχομένου. Η φύση και το μέγεθος του webκάνουν το έργο του ιχνηλάτη, πολύπλοκο, πολύ δύσκολο και όπως θα δούμε, και ο σχεδιασμός ενός συστήματος ιχνηλάτησης του webαποτελεί μια πρόκληση για πολλά προβλήματα.

# Πολιτικές Ιχνηλάτησης

Κάποια χαρακτηριστικά του webκάνουν την ιχνηλάτηση μια δύσκολη διαδικάσία, όπως για παράδειγμα :

Ο τεράστιος όγκος του web,

Η πολύ μεγάλη συχνότητα αλλαγών στο web,

Η δυναμική παραγωγή webσελίδων.

Ένα τεράστιο μέρος των ιστοσελίδων δεν είναι ορατές, αλλά ανήκουν στο λεγόμενο βαθύ ή μη-επισκέψιμο web (π.χ βάσεις δεδομένων). Οι σελίδες αυτές είναι επισκέψιμες μόνο όταν υποβληθεί ένα ερώτημα στην βάση δεδομένων. Συνεπώς οι σελίδες αυτές δεν μπορούν να επισκεφθούν με αυτόματο τρόπο. Ο τεράστιος όγκος του ορατού web σημαίνει ότι οι ιχνηλάτες μπορούν να συγκεντρώσουν ένα μόνο μέρος του web και για το λόγο αυτό θα πρέπει να μπουν κάποιες προτεραιότητες. Η μεγάλη συχνότητα μεταβολών στο web σημαίνει ότι έως ότου ένας ιχνηλάτης κατεβάσει την τελευταία σελίδα από έναν ιστότοπο είναι πολύ πιθανόν μια νέα σελίδα να έχει προστεθεί ή κάποιες σελίδες που είχαν ενημερωθεί να διαγραφούν. Επειδή υπάρχει ένας τεράστιος αριθμών διευθύνσεων και επειδή το εύρος των δικτύων δεν είναι άπειρο, θέλουμε να έχουμε κλιμακωτούς και αποτελεσματικούς αλγόριθμους συγκέντρωσης του περιεχομένου του web. Για το λόγο αυτό είναι πολύ σημαντική η σειρά με την οποία ένας ιχνηλάτης θα επισκεφθεί τις σελίδες του web. Η αποτελεσματικότητα και κατά συνέπεια και η επίδοση ενός crawler εξαρτάται από:

* Την πολιτική επιλογής: που καθορίζει ποια σελίδα θα ανακτηθεί κάθε φορά,
* Την πολιτική επίσκεψης: μιας σελίδας που καθορίζει αν έχουν γίνει αλλαγές στην σελίδα
* Μια πολιτική καλής συμπεριφοράς (politenesspolicy) για την αποφυγή συνωστισμού στους ιστότοπους
* Μια πολιτική παραλληλισμού της διαδικασίας ιχνηλάτησης η οποία καθορίζει πως πρέπει να συντονιστούν οι ιχνηλάτες που θα χρησιμοποιηθούν για την συγκέντρωση του περιεχομένου του web.

# Πολιτική Επιλογής Περιεχομένου

Λόγω του τεράστιου όγκου του web όλες οι μηχανές αναζήτησης συγκεντρώνουν ένα μόνο ποσοστό από τα περιεχόμενα όλου του web, περίπου το 60%. Επομένως είναι πολύ σημαντικό αυτό το ποσοστό του web που θα συγκεντρωθεί να είναι όσο το δυνατό περισσότερο αντιπροσωπευτικό και όχι συγκεντρωμένο τυχαία. Αυτό σημαίνει ότι χρειάζεται ένας μηχανισμός καθορισμού της προτεραιότητας των ιστοσελίδων ανάλογα την σημαντικότητά τους. Η σημαντικότητα μιας ιστοσελίδας εξαρτάται από πολλά πράγματα, για παράδειγμα, τη δημοτικότητα των συνδέσμων που δείχνουν προς την σελίδα. Ο σχεδιασμός μιας αποτελεσματικής πολιτικής επιλογής του περιεχομένου του web δεν είναι καθόλου απλή διαδικασία καθόσον βασίζεται σε μερική πληροφορία. Πράγματι το σύνολο των webσελίδων δεν είναι γνωστό κατά την διάρκεια της ιχνηλάτησης. Για το πρόβλημα αυτό έχουν χρησιμοποιηθεί διάφορες πολιτικές όπως για παράδειγμα , η τεχνική κατά πλάτος-πρώτα (breadth-first), το πλήθος των συνδέσμων που δείχνουν σε μια ιστοσελίδα και η τιμή του Pagerankτ ης σελίδας που ενδεχομένως είναι γνωστή από προηγούμενη ιχνηλάτηση. Πολλές φορές 2 διαφορετικά URLs δείχνουν στο ίδιο περιεχόμενο. Οι ιχνηλάτες συνήθως κάνουν μια μορφή κανονικοποίησης στα URL για να αποφύγουν την ανάκτηση της ίδιας σελίδας πολλές φορές. Με την κανονικοποίηση αλλάζει το URL και παίρνει μια πρότυπη και συμβατή μορφή. Το webαλλάζει από στιγμή σε στιγμή. Εκατομμύρια νέες σελίδες δημιουργούνται καθημερινά και άλλες τόσες αποσύρονται ημερησίως. Είναι δυνατόν όσπου να τελείωσει ένας ιχνηλάτης την συγκέντρωση του περιεχομένου ενός ιστότοπου, νέες σελίδες να έχουν εμφανιστεί και άλλες να λεχουν αποσυρθεί. Προφανώς μια μηχανλη αναζήτησης στο web πρέπει να εντοπίσει εγκαίρως τέτοιες μεταβολές αι να διαθέτει πάντα επικαραιροποιημένο περιεχόμενο. Στόχος μιας μηχανής αναζήτησης είναι να κρατήσει τη «φρεσκάδα» των σελίδων όσο πιο ψηλά. Για την επανα–επίσκεψη των σελίδων έχουν δοκιμαστεί 2 πολιτικές (re-visitingpolicy).

* Ομοιόμορφη πολιτική, όλες οι σελίδες επανα-επισκέπτονται με την ίδια συχνότητα ανεξάρτητα από το πόσο συχνά αλλάζουν,
* Αναλογική πολιτική, πιο τακτικά επανα-επισκέπτονται οι σελίδες που αλλάζουν συχνά

Από τις 2 αυτές πολιτικές στη πράξη πιο αποτελεσματική έχει αποδειχθεί η πρώτη. Και στις 2 αυτές πολιτικές δεν λαμάνεται υπόψη πόσο σημαντική είναι μια σελίδα, δηλαδή όλες οι σελίδες θεωρούνται ποιοτικά ίσες.

# Πολιτική Καλής Συμπεριφοράς

Όταν ένας crawler εκτελεί πολλαπλά αιτήματα ανά δευτερόλεπτο και κατεβάζει μεγάλα αρχεία μπορεί να απασχολεί τον εξυπηρέτη του οποίου η επίδοση μπορεί να πέσει δραματικά. Αυτό είναι ένα αρνητικό στοιχείο για πολλές εφαρμογές γιατί :

Απασχολείται το δίκτυο του οποίου το εύρος είναι πεπερασμένο

Υπερφορτώνεται ο εξυπηρετητής

Μια αραχνοπαγίδα έχει ως αποτέλεσμα ο crawler να πέσει σε ατέρμονο loop με αποτέλεσμα να καταρεύσει. Μια μερική λύση των προβλημάτων αυτών βρίσκεται στο πρωτόκολλο εξαιρέσεων των ρομπότ (robot exclusion protocol), γνωστό ως πρωτόκολλο robots.txt το οποίο χρησιμοποιείται από τους διαχειριστές για να επισημάνουν τα μέρη εκείνα του εξυπηρετητή που δεν πρέπει να πειράξει ο ιχνηλάτης. Ένας «ευγενικός» crawler εναλλάσει διαδοχικές αιτήσεις μεταξύ 2 ξενιστών (hosts) καθώς επίσης διαδοχικές αιτήσεις στον ίδιο εξυπηρετητή γίνοτναι εφόσον περάσουν κάποια δευτερόλεπτα. Οι εμπορικές μηχανές αναζήτησης έχουν την δυνατότητα να δηλώσουν ένα αριθμό που δηλώνει τα δευτερόλεπτα που πρέπει να περάσουν μεταξύ 2 διαδοχικών αιτημάτων στον ίδιο εξυπηρετητή. Η τεχνική αυτή λύνει το πρόβλημα που προκύπτει στη περίπτωση που ένας εξυπηρετητής δεν είναι διαθέσιμος οπότε κατά την προσπάθεια σύνδεσης δημιουργείται σφάλμα (timeout). Επομένως ο ιχνηλάτης θα πρέπει να επισκέφτεται πολλές φορές τις σελίδες ενός τέτοιου εξυπηρετητή έως ότου τον βρει διαθέσιμη.

# Πολιτική Παραλληλισμού

Κατανεμημένη συγκέντρωση του web σημαίνει κατανεμημένη επεξεργασία κατά την οποία χρησιμοποιούνται ταυτόχρονα πολλές μηχανές για να συγκεντρώσουν και να ευρετηριάσουν το περιεχόμενο του web. Σκοπός μας είναι η μεγιστοποίηση της συχνότητας κατεβάσματος web περιεχομένου και ταυτόχρονα η ελαχιστοποίηση του κόστους επικοινωνίας (overhead) από τον παραλληλισμό. Προφανώς θα πρέπει να αποφεύγουμε να κατεβασουμε τις ίδιες σελίδες πολλές φορές. Για τον σκοπό αυτό το σύστημα ιχνηλάτησης απαιτεί μια πολιτική για την εκχώρηση νέων URL που προκύπτουν κατά την διάρκεια ιχνηλάτησης σε διαφορετικούς υπολογιστές, για να μη βρεθούν τα ίδια URLσε διαφορετικές διεργασίες ιχνηλάτησης. Συνήθως υπάρχουν 2 πολιτικές εκχώρησης του φόρτου στους επεξεργαστές :

### Πολιτική Δυναμικής Εκχώρησης

Με την πολιτική αυτή ένας κεντρικός εξυπηρετητής εκχωρεί δυναμικά νέα URLσε διαφορετικούς ιχνηλάτες. Αυτό επιτρέπει στον κεντρικό εξυπηρετητή να ισοζυγίσει το φόρτο σε κάθε ιχνηλάτη. Με την πολιτική αυτή ωστόσο ο κεντρικός εξυπηρετητής μπορεί να δημιουργεί πρόβλημα καθυστέρησης (bottleneck), για αυτό η περισσότερη δουλειά πρέπει να μεταφερθεί στις κατανεμημένες διεργασίες ιχνηλάτησης.

### Πολιτική Στατικής Εκχώρησης

Με αυτή την πολιτική, υπάρχει ένας σταθερός κανόνας ο οποίος ορίζει πως θα γίνει η εκχώρηση των URLστους crawlers.

# Προβλήματα

* Αραχνοπαγίδες (spidertraps). Μια αραχνοπαγίδα είναι ένα σύνολο από ιστοσελίδες οι οποίες έχουν δημιουργηθεί είτε σκόπιμα είτε όχι με σκοπό να αναγκάσουν τον ιχνηλάτη να κάνει ένα άπειρο πλήθος αιτήσεων και έτσι στο τέλος να καταρρεύσει. Αραχνοπαγίδες μπορεί να δημιουργηθούν για να πιάσουν κακόβουλο λογισμικό (spambots) τα οποία δεσμεύουν ένα μέρος του έυρους του δικτύου.
* Πλαίσια (frames). Ορισμένες σελίδες αποτελούνται από υποσελίδες. Ο χρήστης βλέπει μια μοναδική σελίδα χωρισμένη σε τμήματα που το καθένα όμως αντιστοιχεί σε διαφορετικό πλαίσιο. Ο φυλλομετρητής κατεβάζοντας των HTML κώδικα της αρχικής σελίδας θα πρέπει να εντοπίσει τα URLs των διαφορετικών πλαισίων να ανακτήσει το περιεχόμενό τους και να το θεωρήσει ως περιεχόμενο της αρχικής σελίδας.

1. Tools & Frameworks

# Scrapy

Το Scrapy είναι ένα framework της Python για ιχνηλάτευση (crawling) ιστοσελίδων και εξαγωγή δομημένων δεδομένων από αυτές, τα οποία δεδομένα μπορούν να χρησιμοποιηθούν για ένα ευρύ φάσμα εφαρμογών, όπως η εξόρυξη δεδομένων, επεξεργασία πληροφοριών ή για την δημιουργία αρχείων. Ακόμα και αν αρχικά είχε σχεδιαστεί για webscraping , το Scrapy μπορεί να χρησιμοποιηθεί για την εξαγωγή δεδομένων χρησιμοποιώντας APIιστοσελίδων ή ως ένα crawlerγενικής χρήσης.

Η αρχιτεκτονική ενός Scrapy πρότζεκτ βασίζεται στις λεγόμενες «αράχνες» (spiders). Οι αράχνες είναι αυτόνομα προγράμματα ανίχνευσης στα οποία έχουν δοθεί ένα σύνολο οδηγιών. Πρόκειται για ένα framework το οποίο δεν επαναλαμβάνει τον εαυτό του πράγμα που καθιστά ευκολότερη την χρήση του Scrapy για την δημιουργία μεγάλης κλίμακας crawling πρότζεκτ. Επιπλέον το Scrapy παρέχει και κέλυφος (shell), δίνοντας έτσι την δυνατότητα στους προγραμματιστές να δοκιμάσουν την συμπεριφορά ενός ιστότοπου.

Το Scrapy αποτελεί έναν ιχνηλάτη ευρείας χρήσης καθώς είναι απλό, γρήγορο, επεκτάσιμο. Είναι ένα framework ανοιχτού κώδικα, δηλαδή οποιοςδήποτε μπορεί να κάνει δοκιμές και επεκτάσεις σε αυτό, και για αυτόν τον λόγο πολλές εταιρίες έχουν αναπτύξει εφαρμογές και άλλα έργα που αφορούν την εξόρυξη δεδομένων πάνω στο Scrapy. Επίσης έχει εκτενές documentation και υπάρχουν πάρα πολλά ερωτήματα και topics σε φόρουμ ευρέως αναγνωρισμένα όπως το StackOverFlow.

## Αρχιτεκτονική του Scrapy

Το Scrapy αποτελείται από τα εξής συστατικά:

### ScrapyEngine

Η μηχανή του Scrapy είναι υπεύθυνη για τον έλεγχο της ροής των δεδομένων μεταξύ όλων των συστατικών του συστήματος, και την ενεργοποίηση ενεργειών όταν συμβούν συγκεκριμένα γεγονότα. Η ροή των δεδομένων στο Scrapy ελέγχεται από την μηχανή εκτέλεσης (execution engine) και έχει ως εξής :

Η μηχανή δέχεται τα πρώτα URLs για να να ανιχνεύσει από την αράχνη και τα στέλνει στον Scheduler με την σειρά που αιτούνται.

Η μηχανή ρωτά τον Scheduler για τα επόμενα προς ανίχνευση URLs.

Schedulerεπιστρέφει τα επόμενα URLs προς ανίχνευση στην μηχανή, η οποία με την σειρά της τα στέλνει στον Downloader, μέσω του Downloader Middleware.

Μόλις η σελίδα κατέβει ο Downloader δημιουργεί μια ανταπόκριση (Response) με αυτή την σελίδα και την στέλνει στην μηχανή, μέσω του Downloader Middleware.

Η μηχανή δέχεται την ανταπόκριση από τον Downloader και την αποστέλει στην Αράχνη για επεξεργασία, μέσω του Spider Middleware.

Η Αράχνη επεξεργάζεται την ανταπόκριση και επιστρέφει τα στοιχεία που προήρθαν από την εξόρυξη των ιστοσελίδων και νέα αιτήματα προς την μηχανή.

Η μηχανή στέλνει τα στοιχεία που προήρθαν από τα URLs καθώς και τα νέα αιτήματα στην Αράχνη μέσω του Spider Middleware, η οποία με την σειρά της στέλνει τα αποτελέσματα στον αγωγό στοιχείων (Item Pipelines)και τις επεξεργασμένες αιτήσεις στον Scheduler.

Τα παραπάνω βήματα επαναλαμβάνονται μέχρις ότου να μην υπάρχουν και άλλα αιτήματα προς τον Scheduler.

### Scheduler

Ο Scheduler λαμβάνει αιτήσεις από την μηχανή οι οποίες μπαίνουν σε μια ουρά έτσι ώστε να τροφοδοτήσουν την μηχανή μόλις αυτή τις ζητήσει.

### Downloader

Είναι υπεύθυνος για την προσκόμιση ιστοσελίδων και να τις τροφοδοτεί στην μηχανή, η οποία με την σειρά της να τις στέλνει στις Αράχνες.

### Spiders

Πρόκειται για custom κλάσεις γραμμένες από τους χρήστες του Scrapy για την ανάλυση των αιτημάτων και την εξαγωγή των στοιχείων από τα URLs. Ουσιαστικά πρόκειται για κλάσεις οι οποίες καθορίζουν πως ένα συγκεκριμένο site (η ομάδα ιστοσελίδων) θα αναλυθεί, συμπεριλαμβανομένου πως πραγματοποιείται το crawling, και ποια θα είναι τα εξαγώμενα στοιχεία από τα sites. Ο κύκλος εξόρυξης για τις αράχνες πάει ως εξής :

Η αρχή γίνεται με την γενίκευση των αρχικών αιτημάτων για να αναλυθούν τα πρώτα URLs, και να γίνει καθορισμός της συνάρτησης η οποία κατεβάζει την ανταπόκριση από τα συγκεκριμένα αιτήματα.

Στην συνάρτηση αυτή (callback function) γίνεται ανάλυση της απόκρισης (webpage) και επιστρέφονται λεξικά (dicts) με τα δεδομένα που εξάγαμε

Στις callback functions γίνεται ανάλυση των περιεχομένων μια σελίδας με την χρήση των Selectors και γενικέυονται αντικείμενα με τα εξαγώγιμα στοιχεία.

Τελικά τα στοιχεία επιστρέφονται από την Αράχνη σε μια βάση δεδομένων ή γράφονται σε ένα αρχείο μέσω του αγωγου στοιχείων.

### Item Pipeline

Όταν ένα στοιχείο έχει εξαχθεί από μια αράχνη στέλνεται στον αγωγό αντικειμένων (Item Pipeline) ο οποίος με την σειρά του επεξεργάζεται αυτό το στοιχείο μέσα από πολλές συνιστώσες, που εκτελούνται διαδοχικά. Κάθε συνιστώσα του αγωγού στοιχείων είναι μια pythonκλάση που υλοποιεί μια απλή μέθοδο. Λαμβάνουν ένα αντικείμενο και εκτελούν μια ενέργεια σε αυτό. Για παράδειγμα ο αγωγός στοιχείων κάνει τα εξής:

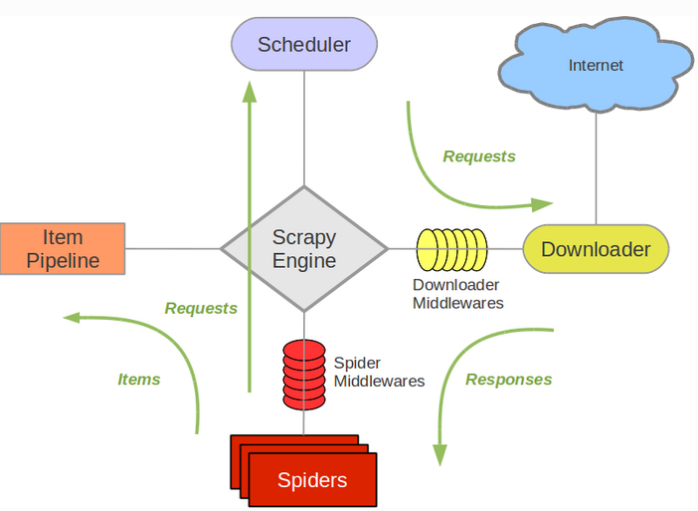
* Καθαρισμό δεδομένων
* Εξετάζει αν τα δεδομένα περιέχουν συγκεκριμένα πεδία
* Έλεγχος για διπλότυπα αντίγραφα
* Αποθήκευση των αντικειμένων σε μια βάση δεδομένων

### Spider Middleware

Πρόκειται για συγκεκριμένα προγράμματα που κάθονται μεταξύ της μηχανής και των αραχνών και είναι ικανά να επεξεργαστούν την είσοδο και την έξοδο της αράχνης.

### Downloader Middleware

Πρόκειται για προγράμματα που κάθονται μεταξύ της μηχανής και του Downloader και επεξεργάζονται τα αιτήματα όταν αυτά είναι να περαστούν από την μηχανή στον Downloaderκαι αποκρίσεις από τον Downloaderστην μηχανή.



Εικόνα Αρχιτεκτονική του Scrapy

## Δημιουργία ενός Scrapy πρότζεκτ

Πρίν γίνει η εγκατάσταση του Scrapy, πρέπει πρώτα να έχουμε εγκαταστήσει στον προσωπικό μας υπολογιστή μια έκδοση της Python. Έπειτα το μόνο που απομένει είναι η εγκατάσταση του Scrapy. Ο πιο εύκολος τρόπος είναι μέσω της βιβλιοθήκης **pip**της Python.



Εικόνα Εγκατάσταση του Scrapy

Στην συνέχεια ανοίγουμε το terminalπηγαίνουμε στον φάκελο όπου θέλουμε να είναι το πρότζεκτ και εκτελούμε την ακόλουθη εντολή :



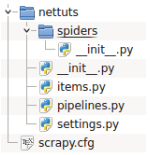
Εικόνα Δημιουργία Scrapy project

Σε αυτό το παράδειγμα θα δείξουμε πως να εξάγουμε όλους τους τίτλους των ποστ στο <http://net.tutsplus.com/>. Ονομάζουμε το πρότζεκτ “nettuts”:



Εικόνα Δημιουργία του nettuts project

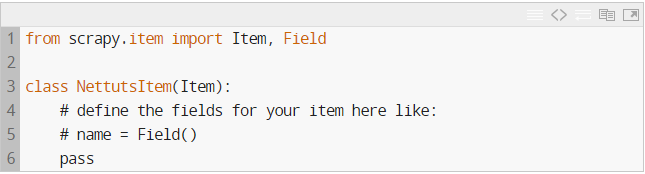
Μετά την δημιουργία του πρότζεκτ έχουμε ένα φάκελο ο οποίος αποθηκεύει σχεδόν όλα αυτά που χρειάζονται



Εικόνα Δομή φακέλου nettuts project

### Καθορισμός του αντικειμένου ιχνηλάτευσης

Όταν γίνεται χρήση του Scrapy , πρέπει να γίνει καθορισμός του τι θα έχουμε ως αποτέλεσμα μετά το crawling, το οποίο ονομάζεται αντικείμενο (item). Για να γίνει αυτό ανοίγουμε το **items.py** αρχείο :



Εικόνα items.py

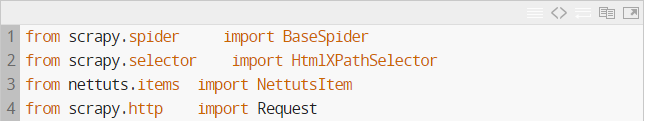
Η κλάση NettutsItem προέρχεται από την κλάση Item και αποθηκεύει τα δεδομένα που θέλουμε να εξάγουμε. Κάθε πεδίο στην κλάση πρέπει να έιναι ένα αντικείμενο της κλάσης Field. Στο παραδειγμά μας, θέλουμε να εξάγουμε τον τίτλο ενός πόστ, έτσι λοιπόν υπάρχει μόνο ένα πεδίο:



Εικόνα Πεδία του NettutsItem

### Spider

Εφόσον δημιουργήσαμε το μοντέλο (τα αντικείμενα που θέλουμε να εξορύξουμε), το μόνο που έχει μείνει είναι η δημιουργία της αράχνης (spider) η οποία ελέγχει το crawling. Στον φάκελο spiders δημιουργούμε ένα αρχείο με το όνομα ntspiders.py. Στο αρχείο αυτό δηλώνουμε κάποιες απαραίτητες κλάσεις:



Εικόνα Δηλώσεις των κλάσεων

Η κλάση BaseSpider είναι η κλάση στην οποία βασίζεται όλο το crawling. Η κλάση HtmlXPathSelector είναι μια βοηθητική κλάση η οποία μας βοηθαέι να εξάγουμε τα δεδομένα που επιθυμούμε. Το επόμενο πράγμα είναι η καρδιά του crawler η κλάση της αράχνης. Πρόκειται για μια κλάση που κληρονομεί από την BaseSpider κλάση και έχει 3 πεδία και μια μέθοδο:

* name: το όνομα της αράχνης, το οποίο χρησιμοποιείται για την εκκίνηση της αράχνης. Μπορούμε να δώσουμε ότι όνομα επιθημούμε. Στην συγκεκριμένη περίπτωση θα την ονομάσουμε “nettuts”.
* allowed\_domains: Πρόκειται για μια λίστα των domains στα οποία ο crawler εκτελεί crawling. Οποιοδήποτε domain δεν βρίσκεται στην λίστα δεν είναι διαθέσιμο για crawling.
* start\_urls: Πρόκειται για μια λίστα από URLs, τα οποία είναι οι ρίζες για το crawling.
* parse(self, response): Είναι η κύρια μέθοδος η οποία κληρονομεί από την BaseSpider και περιέχει την κύρια λογική του crawler.



Εικόνα 14 name, allowed\_domains, start\_urls

Στο επόμενο βήμα πρέπει να καθορίσουμε τα δεδομένα που θελουμε να εξάγουμε από την ιστοσελίδα. Σε αυτή την περίπτωση είναι ο τίτλος των πόστ. Για την εξαγωγή του λοιπόν χρήσιμοπούμε XPathSelectors. Συνεπώς ο πλήρης κώδικας της αράχνης είναι:



Εικόνα class MySpider

Για να γίνει εκτέλεση της αράχνης πάμε στο command prompt των Windows, πηγαίνουμε στο φάκελο του πρότζεκτ και εκτελούμε την εντολή:

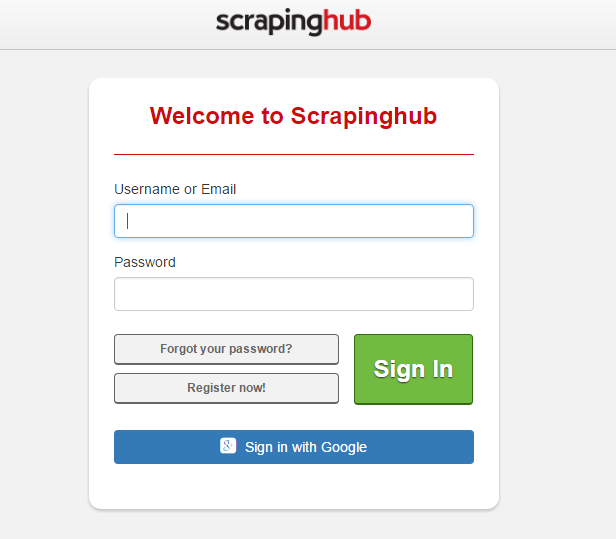


Με την παραπάνω εντολή γίνεται εκκίνηση της αράχνης που φτιάξαμε και αποθηκεύουμε τα δεδομένα σε ένα CSV αρχείο.

# Portia Scrapinghub

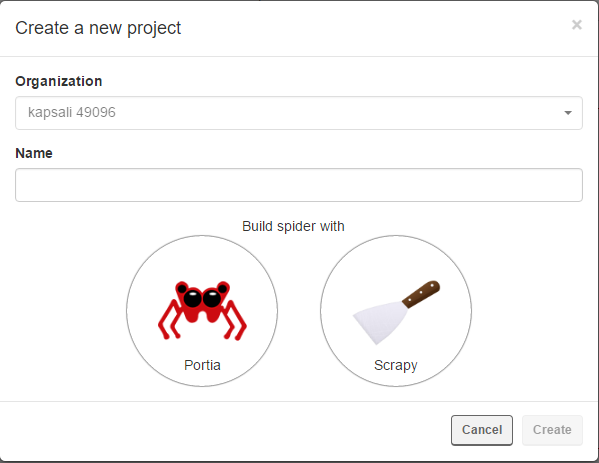
Για την υλοποίηση της βάσης γνώσης στηριχθήκαμε σε πληροφορία συνολικά από 10 ιστοσελίδες. Οι πιο αντιπροσωπευτικές και αυτές που φέραν την περισσότερη πληροφορία αναλύθηκαν με την χρήση του Scrapy Framework. Τα υπόλοιπα sites τα οποία φέραν μικρότερη πληροφορία αναλύθηκαν με την χρήση του εργαλείου Portia Scrapinhub.

Πρόκειται για ένα online open-source εργαλείο το οποίο μέσω επιλογής των στοιχείων σε ενα website κάνει extractτα αντίστοιχα δεδομένα. Για την χρήση του portia scrapinghub πρέπει αρχικά να δημιουργήσει λογαριασμό στο Scrapinghub, όπου συμπληρώνεις τα στοιχεία σου.



Εικόνα Δημιουργία λογαριασμού στο Scrapinghub

Εφόσον ολοκληρωθούν τα απαραίτητα βήματα, ο χρήστης οδηγείται στο overview του λογαριασμού του από όπου μπορεί να δημιουργήσει ένα πρότζεκτ. Πατώντας το σχετικό κουμπί (CreateProject) μπορούμε να δημιουργήσουμε ενα έργο και οδηγούμαστε στην φόρμα όπου ακολουθεί στην παρακάτω εικόνα.



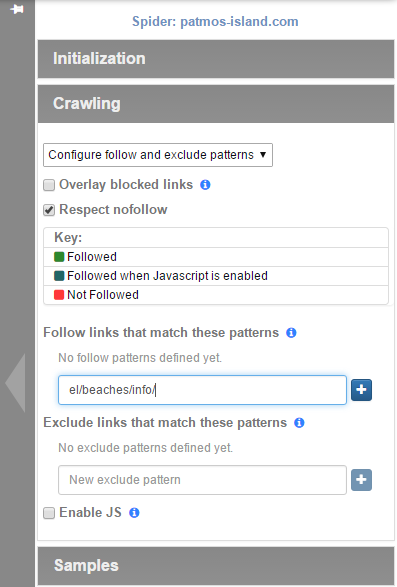
Εικόνα Δημιουργία πρότζεκτ στο Portia Scrapinghub

Στην παραπάνω φόρμα συμπληρώνουμε το όνομα του πρότζεκτ καθώς και το εικονίδιο Portia. Και επιλέγουμε το κουμπί create. Στην συνέχεια μεταφερόμαστε σε μια σελίδα που θέτουμε το URLτου ιστότοπου που θελουμε να εξάγουμε τα δεδομένα του.Σε αυτό το παράδειγμα θα επεξεργαστούμε το <http://patmos-island.com/el/beaches/browse> και πατάμε την επιλογή newspider.



Εικόνα portia new spider

Αφού ολοκληρωθεί αυτό το βήμα, το portia εξομειώνει τον ιστότοπο. Στον εξομειωμένο ιστότοπο πατάμε πάνω σε ένα από τα λινκ των παραλιών. Παρατηρούμε ότι όλα τα λινκ που αναφέρονται στις σελίδες που μας δίνουν λεπτομερείς πληροφορίες για τις παραλίες στα URLs τους έχουν όλα το συγκεκριμένο pattern «**el/beaches/info**». Οπότε για να κάνει crawling αυτές τις ιστοσελίδες το portia scrapinghub θα πρέπει να τις ορίσουμε, αυτό γίνεται πηγαίνοντας στο αναδυόμενο παράθυρο του εργαλείου, επιλέγουμε το tab **Crawling**, στο αναδυόμενο μενού επιλέγουμε το **configure follow and exclude patterns**, και στο editbox βάζουμε το pattern που βρίσκουμε στα λινκ που θέλουμε να ακολουθήσει το portia (Βλέπε Εικόνα 18).



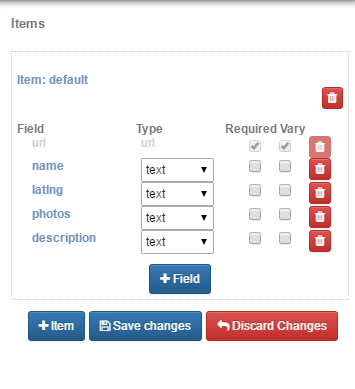
Εικόνα Crawling follow patterns

Αφού ολοκληρωθεί αυτό το βήμα επιλέγουμε το κόκκινο κουμπί που γράφει “Annotate Page”, έτσι ώστε να μπορέσει να γίνει η επιλογή των στοιχείων που θέλουμε να εξαχθούν από τις σελίδες που φέρουν το pattern που τέθηκε στην επιλογή Crawling του Portia.



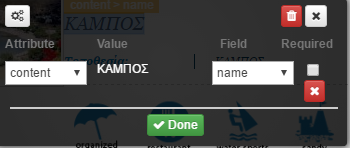
Εικόνα Annotate this page

Τα αντικείμενα που επιθυμούμε να εξαχθούν είναι: το όνομα της παραλίας, οι φωτογραφίες της παραλίας, περιγραφή της παραλίας και το γεωγραφικό μήκος και πλάτος. Εφόσον έχουμε πατήσει το κουμπί “Annotate this page”, πηγαίνοντας στο αναδυόμενο παράθυρο παρατηρούμε ότι οι επιλογές έχουν αλλάξει. Επιλέγουμε την επιλογή **Extracted Items**, όπου και θέτουμε τα αντικείμενα που θέλουμε να εξαχθούν καθώς και τον τύπο τους.

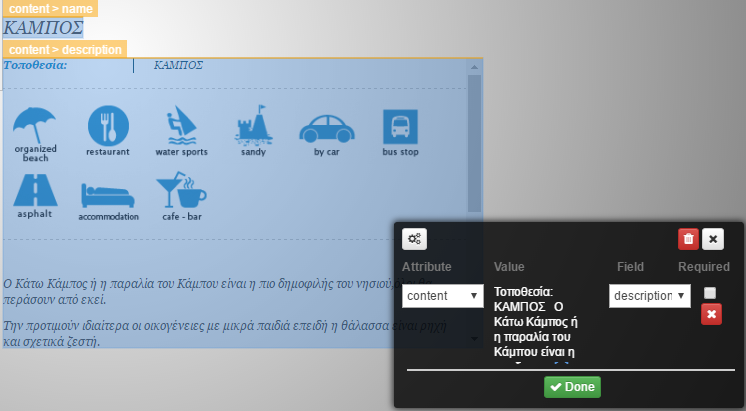


Εικόνα Extracted Items

Η εικόνα 20 παρουσιάζει τα αντικείμενα καθώς και τους τύπους των αντικειμένων, για την αποθήκευση των αλλαγών επιλέγουμε το κουμπί “**Save changes**”. Για την επιλογή των αντικειμένων μαρκάρουμε τα στοιχεία που θέλουμε για παράδειγμα, για να εξαχθέι το όνομα της παραλίας το μαρκάρουμε και στο drop down menu επιλέγουμε το στοιχείο που οριοθετεί, δηλαδή το name. Ομοίως και για τα υπόλοιπα στοιχεία.

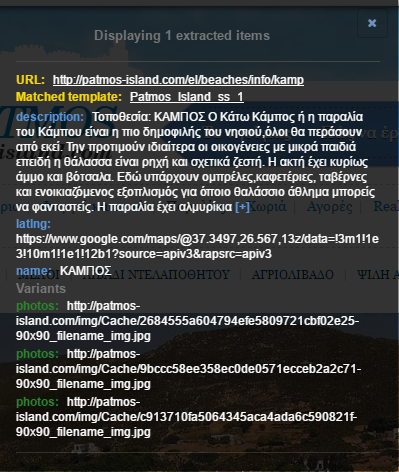


Εικόνα Επιλογή Ονόματος



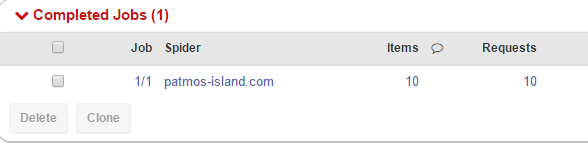
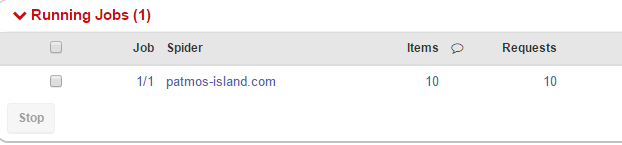
Εικόνα Επιλογή Περιγραφής

Αφού ολοκληρωθεί η επιλογή όλων των αντικειμένων, επιλέγουμε στην κορυφή του παραθύρου το κουμπί “**Save Sample**”, ουσιαστικά με την επιλογή αυτή αποθηκεύονται τα στοιχεία που αναλύθηκαν από την σελίδα, δηλαδή αποθηκεύονται πληροφορίες για ένα λινκ.



Εικόνα Extracted Sample

Για να γίνει η εξόρυξη των πληροφοριών από όλες τις σελίδες πηγαίνουμε στην αρχική σελίδα του πρότζεκτ και πατάμε το κουμπί “Publish Changes”. Μετά από αυτό το βήμα οδηγούμαστε στο dashboard του πρότζεκτ, επιλέγουμε το κουμπί “**Run Spider**”. Τότε η αράχνη που παράγει το portia μπαίνει στην λίστα Running Jobs. Όταν κατέβει όλη η πληροφορία τότε η αράχνη εισάγεται στην λίστα Completed Jobsκαι τότε μέσω του Scrapinghub APIμπορούμε να προσπελάυνουμε τα δεδομένα που κατέβασε η αράχνη.



Εικόνα Running – Completed Jobs

Πατώντας στην επιλογή Items του dashboard μπορούμε να δούμε όλες τις πληροφορίες που κατέβηκαν από τις ιστοσελίδες του ιστότοπου που ακολουθούσαν το pattern “**el/beaches/info**”.

### Scrapinghub API

Για να μπορέσουμε να αποθηκεύσουμε όλη αυτή την πληροφορία στην βάση δεδομένων πρέπει να εγκαταστήσουμε το **shub,** πρόκειται για τον command line client του Scrapinghub. Υπάρχει και για την γλώσσα προγραμματισμού Python. Για να το εγκαταστήσουμε πηγαίνουμε στο command line των Windowsκαι εκτελούμε την παρακάτω εντολή ( Βλέπε εικόνα 25 ).



Εικόνα Install shub

Και με την χρήση των εντολών του Scrapinghub API είναι δυνατή η προσκόμιση των αντικειμένων που εξήχθησαν σε μορφή JSON και με την κατάλληλη μορφοποίηση είναι δυνατή η αποθηκευσή τους στην βάση δεδομένων.

# Δημιουργία εφαρμογής με χρήση του Xamarin

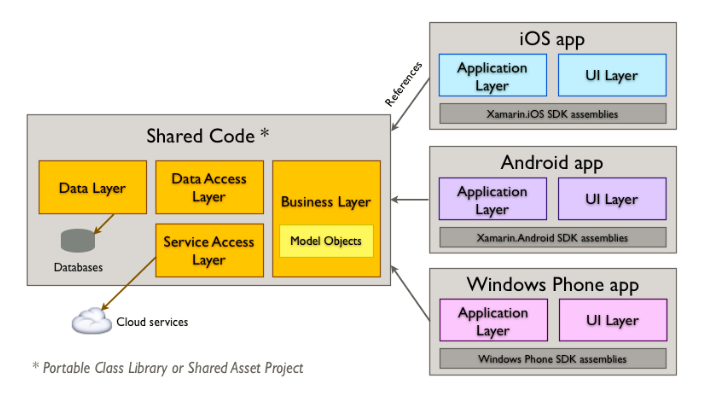
Το Xamarin είναι ένα από τα βασικά εργαλεία που χρησιμοποιήθηκαν σε αυτή την διπλωματική. Μετά την συλλογή πληροφοριών για τις παραλίες της Ελλάδας, έγκειται η ανάγκη παρουσίασης όλης αυτής της πληροφορίας σε κινητά. Το Xamarin μας επιτρέπει την δημιουργία εφαρμογών σε όλες τις τεχνολογίες (IOS, WindowsPhone, Android). H Xamarin είναι μια εταιρεία που ανήκει στην Microsoft και ιδρύθηκε από τους δημιουργούς των πρότζεκτ Mono, MonoforAndroid και MonoTouch. Αυτά τα πρότζεκτ είναι cross-platform υλοποιήσεις του CLI (Common Language Infracture) και του .NET framework. Με χρήση της γλώσσας προγραμματισμού C#, οι προγραμματιστές μπορούν να χρησιμοποιήσουν τα εργαλεία της Xamarin για την δημιουργία native αλλά και cross-platform εφαρμογών για IOS, Windows phone, Android, καθώς και να γίνεται διαμοιρασμός κώδικα σε πολλές πλατφόρμες.

Το μεγάλο ερώτημα είναι ο διαμοιρασμός κώδικα μεταξύ των διαφορετικών τεχνολογιών. Υπάρχουν δύο διαφορετικές μέθοδοι για τον διαμοιρασμό κώδικα μεταξύ των cross-platform apps:

Shared Projects: Γίνεται χρήση του Shared Asset Project για την οργάνωση του κώδικα, και χρήση του #if compiler για την διαχείρηση συγκεκριμένων απαιτήσεων της κάθε πλατφόρμας.

Portable Class Libraries: Δημιουργία μιας Portable Class Library (PCL), με στόχο τις πλατφόρμες που επιθυμούμε να υποστηρίξουμε, γίνεται χρήση interfaces για την εξυπηρέτηση των απαιτήσεων κάθρε πλατφόρμας.

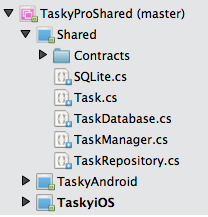
Ο στόχος της στρατηγικής διαμοιρασμού κώδικα είναι η υποστήριξη της αρχιτεκτονικής που φαίνετα στην παρακάτω εικόνα, όπου ένα σύνολο κώδικα είναι δυνατό να διαχειρίζεται από πολλές πλατφόρμες.



Εικόνα Διαμοιρασμός κώδικα μεταξύ πλατφόρμων

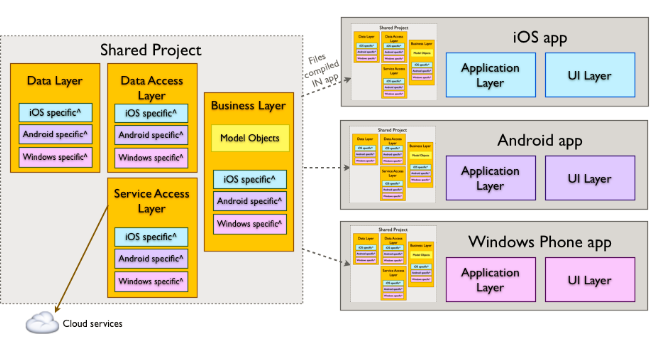
## Shared Projects

Η απλούστερη προσέγγιση για τον διαμοιρασμό αρχείων κώδικα είναι η χρήση ενός Shared Project (αυτή η μέθοδος εισάγεται στο XamarinStudio 5 και στο VisualStudio 2013). Η παρακάτω εικόνα δείχνει ένα solution file που περιέχει 3 πρότζεκτ εφαρμογών (για Android, IOS, Windows Phone). Επίσης περιέχει και ένα Shared Project που περιέχει κοινό C# κώδικα και για τα τρία πρότζεκ εφαρμογών.



Εικόνα Tasky solution file

Η σχετική αρχιτεκτονική φαίνεται από την παρακάτω εικόνα. Κάθε πρότζεκτ περιλαμβάνει όλα τα διαμοιραζόμενα αρχεία κώδικα.



Εικόνα Αρχιτεκτονική Tasky project

Μια cross-platform εφαρμογή που η οποία υποστηρίζει IOS, Android και Windows Phone απαιτεί ένα πρότζεκ εφαρμογής για κάθε πλατφόρμα. Ο κοινός κώδικας «ζει» στο Shared Project.

Για παράδειγμα ένα solution file μιας cross-platform εφαμοργής περιλαμβάνει τα εξής:

Shared: Περιλαμβάνει το κοινό κώδικα για όλα τα πρότζεκτ.

AppAndroid: Xamarin.Android πρότζεκτ εφαρμογής

AppIOS: Xamarin.IOS πρότζεκτ εφαρμογής

AppWinPhone: Windows Phone πρότζεκ εφαρμογής

Με αυτόν τον τρόπο τα τρία πρότζεκ εφαρμογών μοιράζονται τον κοινό κώδικα, τα C# αρχεία στον φάκελο Shared. Κάθε αλλαγή στον κοινό κώδικα θα διαμοιράζεται μεταξύ και των τριών πρότζεκτ.

### Πλεονεκτήματα

* Διαμοιρασμός κώδικα μεταξύ πολλών διαφορετικών πρότζεκτ.
* Ο κοινός κώδικας μπορεί να γίνει branched σε κάθε πλατφόρμα κάνοντας χρήση τις ντιρεκτίβες κάθε compiler, για παράδειγμα κάνοντας χρήση του **# if \_ANDROID\_** ,έτσι ώστε ο κοινός κώδικας να έχει διαφορετικές εκδοχές για κάθε πλατφόρμα, δίνοντας έτσι ευελιξία.
* Τα πρότζεκτ των εφαρμογών μπορούν να περιλμβάνουν συγκεκριμένα χαρακτηριστικά για κάθε πλατφόρμα, όπου ο διαμοιρασμένος κώδικας μπορεί να χειριστεί.

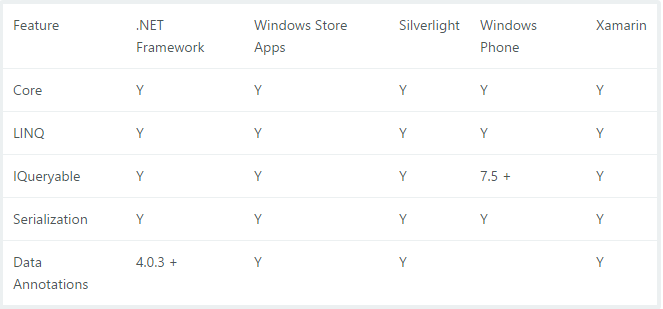
### Μειονεκτήματα

* Σε σχέση μετα άλλα πρότζεκτ του solution file μιας cross-platform εφαρμογής ένα Shared Project δεν έχει έξοδο ένα dll αρχείο. Κατά την διάρκεια του compilation τα αρχεία κοινού κώδικα χειρίζονται ως μέρος του σχετικού πρότζεκτ εφαρμογής και γίνονται compile στο dll αρχείο αυτού του πρότζεκτ. Αν υπάρχει ανάγη διαμοιρασμού κώδικα μέσω DLL αρχείων τότε οι Portable Class Libraries.

## Portable Class Libraries

Όταν δημιουργείται ένα πρότζεκτ εφαρμογής ή ένα Library project, το DLLαρχείο που παράγεται περιορίζεται στο να δουλεύει για την συγκεκριμένη πλατφόρμα για την οποία αναφέρεται το πρότζεκτ. Αυτό αποτρέπει την δημιουργία από τον προγραμματιστή , για κάθε πρότζεκτ DLL αρχείων.

Με την δημιουργία μιας PCL, υπάρχει η δυνατότητα επιλογής σε ποιες πλατφόρμες να λειτουργεί ο κώδικας. Οι επιλογές αυτές περιγράφουν ποιες πλατφόρμες υποστηρίζει η βιβλιοθήκη. Η παρακάτω εικόνα δείχνει κάποια από τα χαρακτηριστικά της .NET πλατφόρμας. Για την δημιουργία ενός PCL, το οποίο θα εκτελείται σε συγκεκριμένες συσκευές/πλατφόρμες, πρέπει να γίνει η επιλογή των συγκεκριμένων χαρακτηριστικών κατά την δημιουργία του πρότζεκτ.



Εικόνα Επιλογή χαρακτηριστικών PCL

Η στήλη του Xamarin, όπως παρατηρείται, αντανακλά το γεγονός ότι το Xamarin.Android και το Xamarin.IOS υποστηρίζουν όλα τα χαρακτηριστικά που συνδέονται με το VisualStudio 2013, και την διαθεσιμότητα των χαρακτηριστικών σε κάθε βιβλιοθήκες που δημιουργούνται θα περιορίζονται μόνο από τις άλλες πλατφόρμες που δεν είναι συμβατές, για παράδειγμα .NET 4, .NET 4.5, Silverlight 5, Windows Phone 8, Windows Store Apps.

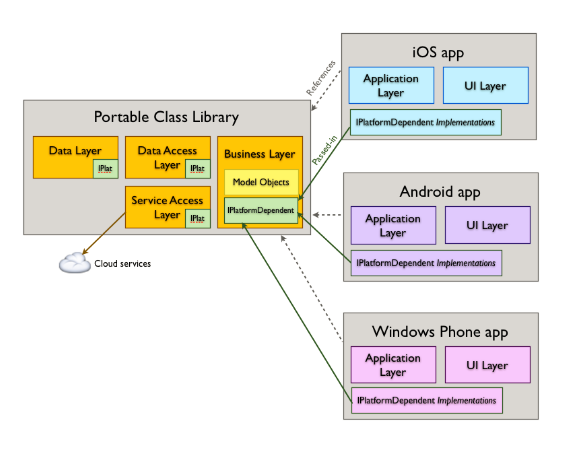
### Πλεονεκτήματα

### Δημιουργία και έλεγχος πρότζεκτ το οποίο μπορεί να χρησιμοποιηθεί από άλλες βιβλιοθήκες και εφαρμογές

* Αλλαγές στον κώδικα θα επηρεάσουν όλον τον κώδικα που υπάρχει σε μια solution, δηλαδή την PCL και τα πρότζεκ εφαρμογών για κάθε πλατφόρμα.
* Ένα PCL προτζεκτ μπορεί να χρησιμοποιηθεί από άλλα πρότζεκτ σε μια solution ή το DLL αρχείο του μπορεί να χρησιμοποηθεί ως αναφορά σε άλλα προτζεκτ σε ένα solution.

### Μειονεκτήματα

* Επειδή το ίδιο PCL πρότζεκτ διαμοιράζεται μεταξύ πολλών εφαρμογών, οι συγκεκριμένες βιβλιοθήκες κάθε πλατφόρμας δεν μπορούν να χρησιμοποιηθούν σε ένα PCL πρότζεκτ.
* Ένα PCL πρότζεκτ μπορεί να μην περιέχει κλάσεις οι οποίες μπορεί να είναι διαθέσιμες στα MonoTouch ή MonoforAndroid.



Εικόνα Αρχιτεκτονική cross platform εφαρμογής που χρησιμοποιεί PCL για τον διαμοιρασμό κώδικα

## Cross-Platform Data Access

Οι περισσότερες εφαρμογές έχουν απαίτηση να αποθηκεύουν τα δεδομένα στην συσκευή τοπικά, εκτός αν η ποσότητα των δεδομένων είναι μικρή. Αυτό συνήθως απαιτεί μια βάση δεδομένων και μια διαστρωμάτωση στα δεδομένα έτσι ώστε να γίνεται σωστά η επικοινωνία με την βάση. Οι τεχνολογίες Android και IOS χρησιμοποιούν τον μηχανισμό διαχείρισης δεδομένων SQLite και η πρόσβαση γίνεται πιο απλή μέσω του παρόχου SQLite δεδομένων του Xamarin.

### Πότε χρησιμοποιείται μια βάση δεδομένων

Ενώ οι αποθηκευτικές και επεξεργαστικές δυνατότητες των κινητών συσκευών αυξάνονται, τα τηλέφωνα και τα tablets εξακολουθούν να υστερούν σε σχέση με τα λάπτοπ και desktop. Για τον λόγο αυτό αξίζει να θησιαστεί λίγος χρόνος για τον σχεδιασμό της αρχιτεκτονικής της επεξεργασίας και της αποθήκευσης των δεδομένων για μια εφαρμογή. Υπάρχουν πολλές επιλογές που καθεμία ταιριάζει και σε διαφορετικές απαιτησεις.

* Προτιμήσεις: Τόσο το IOS όσο και το Android προσφέρουν ένα build-in μηχανισμό για την αποθήκευση απλών ζευγών χαρακτηριστικών τιμών. Αν πρόκειται για αποθήκευση ρυθμίσεων χρήστη ή μικρό όγκο δεδομένων τότε συνίσταται χρήση των δυνατοτήτων κάθε πλατφόρμας για αποθήκευση. Στο IOSγια παράδειγμα χρησιμοποιείται το iCloud για backup και συγχρονισμό των δεδομένων για χρήστες που χρησιμοποιούν παραπάνψ από μια συσκευές.
* Αρχεία κειμένου: Η είσοδος του χρήστη ή η κρυφή μνήμη κατεβασμένου περιεχομένου μπορεί να αποθηκευτεί στο σύστημα αρχείων. Είναι απαραίτητο να γίνεται χρήση κατάλληλης ονόμασίας των αρχείων για καλύτερη οργάνωση και εύρεσης των δεδομένων.
* Serialized Data Files: Τα αντικέιμενα μπορούν να παραμείνουν ως XML ή JSON μορφή στο σύστημα αρχείων. Το πλαίσιο .NET περιέχει βιβλιοθήκες που μπορούν να κάνουν serializing και de-serializing αντικειμένων εύκολα. Έγκειται η ανάγκη κατάλληλης οματοδοσίας στα αρχεία δεδομένων
* Βάση Δεδομένων: Ο μηχανισμός SQLite βάσης δεδομένων είναι διαθέσιμος σε IOS και Android και είναι χρήσιμος για αποθήκευση δεδομένων που χρειαζόμαστε να τροφοδοτήσουν μια εφαρμογή.
* Αρχεία Εικόνων: Αν και είναι δυνατή η αποθήκευση των εικόνων ως δυαδικά δεδομένα στην βάση δεδομένων της συσκευής, συνίσταται η απευθείας αποθήκευση των εικόνων στο σύστημα αρχείων της συσκευής. Αν είναι απαραίτητο είναι δυνατή η αποθήκευση των ονομάτων των εικόνων στη βάση δεδομένων έτσι ώστε να συσχετιστούν με άλλα δεδομένα. Όταν υπάρχει μεγάλος όγκος δεδομένων εικόνας έγκειται ο σχεδιασμός μιας στρατηγικής κρυφής μνήμης όπου διαγράφει αρχεία που δεν χρειάζονται πλεόν έτσι ώστε να απελευθερόνεται χώρος αποθήκευσης.

### Πλεονεκτήματα χρήσης Βάσης Δεδομένων

* Οι SQL βάσεις δεδομένων καθιστούν δυνατή την αποτελεσματική φύλαξη των δομημένων δεδομένων.
* Συγκεκριμένα δεδομένα μπορούν να εξαχθούν από την βάση μέσω ερωτημάτων SQL.
* Τα αποτελέσματα των SQL ερωτημάτων μπορούν να ταξινομηθούν.
* Προγραμματιστές με γνώσεις στις βάσεις δεδομένων μπορούν να σχεδιάσουν μια βάση δεδομένων και την πρόσβαση στα δεδομένα της.
* Το μοντέλο δεδομένων ενός διακομιστή μιας συνδεδεμένης εφαρμογής μπορεί να επαναχρησιμοποιηθεί σε μια mobile εφαρμογή.

### Μηχανισμός SQLite

Ο SQLite μηχανισμός είναι μια ανοιχτού κώδικα μηχανισμός βάση δεδομένων που υιοθετήθηκε από την Googleκαι την Apple για τις κινητές εφαρμογές. O μηχανισμός SQLite είναι ενσωματωμένος και στις δύο τεχνολογίες που αναφέρθηκαν πριν. O SQLite μηχανισμός είναι κατάλληλος για υλοποίηση cross-platformεφαρμογών γιατί:

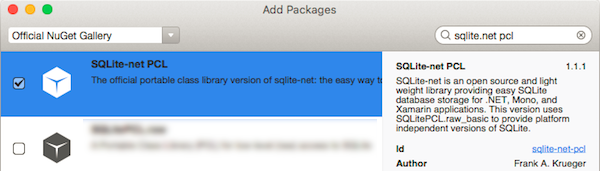
* Είναι μικρός, γρήγορος και εύκολα αποδεκτός από τις κινητές συσκευές.
* Η βάση αποθηκεύεται σε ένα αρχείο, πράγμα που διευκολύνει την διαχείρηση της βάσης από mobileσυσκευές.
* Η μορφή των αρχείων καθιστά την χρήση της βάσης σε όλες τις πλατφόρμες.

### Χρησιμοποιώντας το SQLite.NET

Η βιβλιοθήκη SQLite.NET είναι η βιβλιοθήκη που προτείνει το Xamarinγια την αποθήκευση και την προσέγγιση αντικειμένων στην τοπική SQLite βάση δεδομένων σε μία IOS ή Android συσκευή.

Για να κατεβάσουμε την βιβλιοθήκη αυτή στην εφαρμογή έχουμε 2 επιλογές:

* NuGet: Ο κώδικας είναι διαθέσιμος ως SQLite.net PCL NuGet package που υποστηρίζει μια ποικιλία τεχνολογιών



Εικόνα 32 SQLite.net PCL NuGet package

* Component Store: Η βιβλιοθήκη SQLite.NET είναι διαθέσιμη για IOS και Android από το Xamarin Component Store όπως φαίνεται και από την παρακάτω εικόνα



Εικόνα 33 SQLite.net Xamarin Component Store

Ανεξάρτητα της μεθόδου που χρησιμοποιείται για την χρήση της SQLite.NET βιβλιοθήκης σε μια εφαρμογή, ο κώδικας που χρησιμοποιείται είναι ο ίδιος.

Για να συμπεριληφθεί η SQLite.NET ακολουθούνται τα εξής βήματα:

* Η εντολή δήλωσης της βιβλιοθήκης



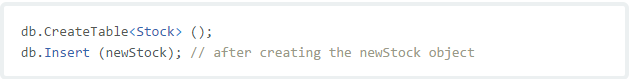
Εικόνα Using statement

* Δημιουργία Βάσης: Η αναφορά σε μια βάση δεδομένων μπορεί να δημιουργηθεί περνόντας το μονοπάτι του αρχείου που θα είναι το αρχείο της βάσης δεδομένων στην κλάση SQLiteConnection constructor.



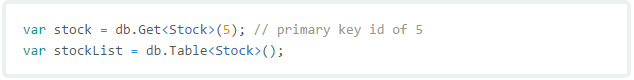
Εικόνα Database file path

* Αποθήκευση Δεδομένων: Εφόσον έχει δημιουργηθεί η σύνδεση με την SQLite βάση, είναι αναγκαία η δημιουργία μιας κλάσης που θα περιέχει το μοντέλο δεδομένων, οπότε για την δημιουργία ενός πίνακα υπάρχει η εντολή **CreateTable** και για την εισαγωγή αντικειμένων στην βάση η εντολή **Insert**. Έστω Stock η κλάση του μοντέλου δεδομένων.



Εικόνα Store Data

* Ανάκτηση Δεδομένων: Για την ανάκτηση ενός αντικειμένου ή λίστας αντικειμένων ακολουθούμε τον παρακάτω κανόνα:



Εικόνα Retrieve Data

### SQLiteΧαρακτηριστικά

Πρόκειται για χαρακτηριστικά που μπορούν να χρησιμοποιηθούν με σκοπό τον έλεγχο πως αποθηκεύονται τα δεδομένα σε μια SQLite βάση δεδομένων.

* **[PrimaryKey]**: Αυτό το χαρακτηριστικό μπορεί να κληθεί για μια integer ιδιότητα με σκοπό να καθορίσει το πρωτεύον κλειδί ενός πίνακα της βάσης.
* **[AutoIncreament]**: Το χαρακτηριστικό αυτό έχει σκοπό την δημιουργία μιας αυτο-αυξανόμενης ιδιότητας, συνήθως πρόκειται για το πρωτεύον κλειδί.
* **[Column(name)]**: Πρόκειται χαρακτηριστικό που είναι υπεύθυνο για την ονοματοδοσία των στηλών ενός πίνακα.
* **[Table(name)]**: Σηματοδοτεί την κλάση μοντέλου για τα αντικείμενα που πρόκειται να αποθηκευτούν σε έναν SQLite πίνακα.
* **[MaxLength(value)]**: Περιορίζει το μέγεθος μιας text ιδιότητας, όταν επιχειρείται εισαγωγή αντικειμένου σε έναν SQLite πίνακα.
* **[Ignore]**: Με την χρήση αυτού του χαρακτηριστικού επεξαιρείται μια ιδιότητα από την βάση δεδομένων.
* **[Unique]**: Μέσω αυτού του χαρακτηριστικού εξασφαλίζουμε ότι μια τιμή όταν εισάγεται σε στήλη ενός πίνακα είναι μοναδική.

### Ερωτήματα σε μια SQLite βάση

* **Insert**: Εισάγει ένα νέο αντικείμενο στην βάση.
* **Get<T>**: Επιχειρεί να ανακτήσει ένα αντικείμενο χρησιμοποιώντας το πωτεύον κλειδί.
* **Table<T>**: Επιστρέφει όλα τα αντικείμενα ενός πίνακα.
* **Query<T>**: Εκτελεί ένα SQL ερώτημα προς την βάση δεδομένων και επιστρέφει έναν αριθμό γραμμών.
* **Execute**: Με την χρήση αυτής της μεθόδου γίνεται εκτέλεση ερωτημάτων που δεν αναμένονται

1. Σχεδιασμός και Υλοποίηση της βάσης γνώσης

# Βάση Δεδομένων

## Διάγραμμα E-R και πίνακες

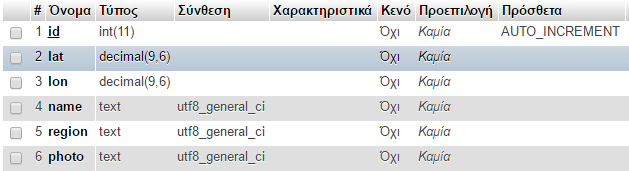
Παρουσιάζεται ένα απλοποημένο διάγραμμα Οντοτήτων – Συσχετίσεων (E-R) της βάσης δεδομένων.



Εικόνα Διάγραμμα Συσχχετίσεων - Οντοτήτων

Η βάση δεδομένων αποτελείται από τους πίνακες beach, beachplus, description, photos, tsek.

### Πίνακας beach



Εικόνα Πίνακας beach

Ο πίνακας beach διατηρεί το γεωγραφικό μήκος, το γεωγραφικό πλάτος, το όνομα, την περιοχή και μια φωτογραφία της παραλίας. Επίσης αποθηκεύουμε το πεδίο id το οποίο είναι το πρωτεύον μοναδικό κλειδί του πίνακα και φέρει το χαρακτηριστικό auto-increment.

Ο MySQL κώδικας για την δημιουργία του πίνακα είναι:

Αρχείο Κώδικα MySQL κώδικας πίνακα beach

**CREATETABLE** `beach` **(**

`id` **int(**11**)NOTNULL** AUTO\_INCREMENT**,**

`lat` **decimal(**9**,**6**)NOTNULL,**

`lon` **decimal(**9**,**6**)NOTNULL,**

`name` text **NOTNULL,**

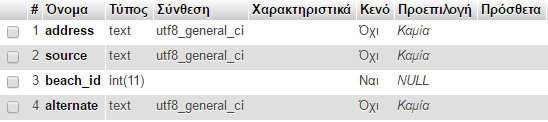
`region` text **NOTNULL,**

`photo` text **NOTNULL,**

**PRIMARYKEY(**`id`**)**

**)** ENGINE**=**InnoDB AUTO\_INCREMENT**=**3274**DEFAULT** CHARSET**=**utf8

### Πίνακας beachplus



Εικόνα Πίνακας beachplus

Ο πίνακας beachplus φέρει επιπλέον πληροφορίες για κάθε παραλία όπως η διεύθυνση, η πηγή όπου πήραμε τις εκάστοτε πληροφορίες διαφορετικά ονόματα της παραλίας. Σε αυτή την περίπτωση είναι δυνατή ύπαρξη παραπάνω εγγραφών με το ίδιο beach\_id. Το πεδίο beach\_id είναι ξένο κλειδί το οποίο υποδηλώνει την σύνδεση του beach πίνακαμε τον πίνακα beachplus. Η ύπαρξη παραπάνω εγγραφών δικαιολογείται από το γεγονός ότι μπορεί να έχουμε εξάγει πληροφορίες για την ίδια παραλία από διαφορετικούς ιστότοπους.

Ο MySQL κώδικας για την δημιουργία του πίνακα beachplus είναι ο παρακάτω:

Αρχείο Κώδικα MySQL κώδικας πίνακα beachplus

**CREATETABLE** `beachplus` **(**

`address` text **NOTNULL,**

`source` text **NOTNULL,**

`beach\_id` **int(**11**)DEFAULTNULL,**

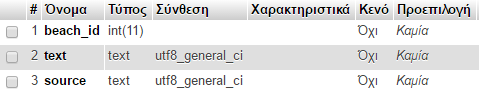
`alternate` text **NOTNULL,**

**KEY** `beach\_id` **(**`beach\_id`**),**

**CONSTRAINT** `beachplus\_ibfk\_1` **FOREIGNKEY(**`beach\_id`**)REFERENCES** `beach` **(**`id`**)ONDELETECASCADEONUPDATECASCADE**

**)** ENGINE**=**InnoDB **DEFAULT** CHARSET**=**utf8

### Πίνακας description



Εικόνα Πίνακας descrption

Ο πίνακας description φέρει πληροφορία περιγραφής από διάφορους ιστότοπους αλλά και σχόλια των χρηστών του Foursquare. Περιέχει διπλότυπα καθώς είναι δυνατή η εξαγωγή πληροφοριών από διαφορετικούς ιστότοπους. Το πεδίο beach\_id είναι ξένο κλειδί και υποδηλώνει την σχέση με τον πίνακα beach. Ουσιαστικά το πεδίο beach\_id δηλώνει ότι ο πίνακας φέρει πληροφορία για την αντίστοιχη παραλία στον πίνακα beach, όπου η τιμή τουid ισούται με την τιμή του beach\_id. Περιέχει τα πεδία text, source, beach\_id.

Ο κώδικας MySQL για την δημιουργία του πίνακα description είναι ο παρακάτω:

Αρχείο Κώδικα MySQL κώδικας πίνακα description

**CREATETABLE** `description` **(**

`beach\_id` **int(**11**)NOTNULL,**

`text` text **NOTNULL,**

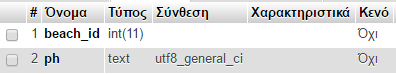
`source` text **NOTNULL,**

**KEY** `beach\_id` **(**`beach\_id`**),**

**CONSTRAINT** `description\_ibfk\_1` **FOREIGNKEY(**`beach\_id`**)REFERENCES** `beach` **(**`id`**)**

**)** ENGINE**=**InnoDB **DEFAULT** CHARSET**=**utf8

### Πίνακας photos



Εικόνα Πίνακας photos

Ο πίνακας photos περιέχει τα URLs των φωτογραφιών σχετικών με κάθε παραλία. Το πεδίο ph υποδηλώνει το URL το πεδίο beach\_id υποδηλώνει το id του πίνακα beach, σχετίζοντας με αυτό τον τρόπο τις φωτογραφίες κάθε γραμμής με μια παραλία (ξεχωριστό id) του πίνακα beach.

Ο κώδικας MySQL για την δημιουργία του πίνακα είναι:

Αρχείο Κώδικα MySQL κώδικας πίνακα photos

**CREATETABLE** `photos` **(**

`beach\_id` **int(**11**)NOTNULL,**

`ph` text **NOTNULL,**

**KEY** `beach\_id` **(**`beach\_id`**),**

**KEY** `beach\_id\_2` **(**`beach\_id`**),**

**CONSTRAINT** `photos\_ibfk\_1` **FOREIGNKEY(**`beach\_id`**)REFERENCES** `beach` **(**`id`**)**

**)** ENGINE**=**InnoDB **DEFAULT** CHARSET**=**utf8

### Πίνακας tsek



Εικόνα Πίνακας tsek

Ο πίνακας αυτός περιέχει το πεδίο pid όπου και αποθηκεύουμε τα id των σελίδων του Foursquare. Πρόκειται για έναν βοηθητικό πίνακα, που με την αποθήκευση των id των σελίδων του Foursquare αποφεύγουμε την αποθήκευση των διπλωτύπων στην βάση δεδομένων.

Αρχείο Κώδικα MySQL κώδικας πίνακα tsek

**CREATETABLE** `tsek` **(**

`pid` text **CHARACTERSET** utf8 **NOTNULL**

**)** ENGINE**=**InnoDB **DEFAULT** CHARSET**=**latin1

## Εξαγωγή δεδομένων από Ιστοσελίδες

Για την εξαγωγή των δεδομένων από ιστοτόπους χρησιμοποιείται το Scrapy Framework ή το online εργαλείο Portia Scrapinghub και λαμβάνουμε τα δεδομένα μέσω του Scrapinghub API. Σε αυτή την ενότητα θα εξετάσουμε πως παίρνουμε πληροφορία από κάθε website.

### Wondergreece.gr

Από αυτόν τον ιστότοπο αναζητούμε πληροφορίες όπως το όνομα παραλίας, το γεωγραφικό μήκος και πλάτος, την περιγραφή, φωρογραφίες, την διεύθυνση και περιοχή της παραλίας, από ιστοσελίδες που περιέχουν στο URLτους το μοτίβο **/Fysi/Paralies**. Στον συγκεκριμένο ιστότοπο για την εξαγωγή των πληροφοριών που επιθυμούμε, έγινε χρήση του ScrapyFramework.

Το αρχείο items.py περιέχει την Python κλάση των αντικειμένων που θα εξαχθούν από τον ιστότοπο. Η κλάση αυτή κληρονομεί από την κλάση scrapy.item το property field. Έτσι λοιπόν δηλώνουμε σε αυτή την κλάση τα αντικείμενα που επιθυμούμε να εξαχθούν από τον ιστότοπο. Τα html πεδία που θα εξαχθούν φαίνονται στο αρχείο items.py του πρότζεκτ.

Αρχείο Κώδικα wondergreece/items.py

**from** scrapy**.**item **import** Item**,** Field

**classWondergreeceItem(**Item**):**

beach **=** Field**()**

description **=** Field**()**

photos **=** Field**()**

lat **=** Field**()**

lon **=** Field**()**

address **=** Field**()**

region **=** Field**()**

**pass**

Αρχείο Κώδικα 7

Τα παραπάνω πεδία σηματοδοτούν τα στοιχεία που ο crawler, θα αναζητήσει στις σελίδες του ιστοτόπου, που ακολοθούν ένα συγκεκριμένο μοτίβο. Ο ορισμός των σελίδων που θα ακολουθηθούν καθώς και τα xpaths των σελίδων που βρίσκονται τα αντικείμενα πρός εξόρυξη ορίζονται στο αρχείο αράχνης.

Στην συνέχεια πηγαίνουμε στον φάκελο spidersτου πρότζεκτ και δημιουργούμε το αρχείο που περιέχει τον κώδικα της αράχνης (spider). Η αράχνη πηγαίνει σε κάθε ιστοσελίδα του ιστοτόπου που ικανοποιεί το μοτίβο **/Fysi/Paralies**. Πρόκειται για το αρχείο wonderspider.py, είναι μια κλάση ουσιαστικά που κληρονομεί από την κλάση CrawlSpider, έτσι ώστε να εκτελείται recursively crawling, δηλαδή να ακολουθούνται περισσότερα από ένα link σε ένα ιστότοπο, αλλά και από αυτά τα links να ακολουθούνται εξερχόμενα links και να γίνονται crawling. Το αναγνωριστικό κάθε αράχνης είναι το όνομα της και καλείται μέσω αυτού. Για την εξαγωγή των ζητούμενων στοιχείων του ιστοτόπου χρησιμοποιούμε Xpaths και selectors για την προσπέλαση των απαραίτητων HTML συστατικών. Ακολουθεί ο κώδικας ο οπίος περιγράφει τα HTML μονοπάτια, που ακολουθεί ο crawler έτσι ώστε να εξάγει πληροφορία από συγκεκριμένα sites του ιστότοπου.

Η παραπάνω κλάση κληρονομεί από την κλάση CrawlSpider, ακολουθεί την αρχική λίστα των λινκ που έχουν ένα συγκεκριμένο μοτίβο, και σε αυτά τα λινκ ακολουθούνται άλλα εσωτερικά λινκς τα οποία μπορεί να πληρούν ένα άλλο μοτίβο. Αυτήν την δουλειά την αναλαμβάνει η συνάρτηση rules που κληρονομείται από την κλάση CrawlSpider, όπου ορίζονται σε ποια links πρέπει να περιοριστεί ο crawler. Ο crawler επιστρέφει ένα dictionary της κλάσης WondergreeceItem, το οποίο περιέχει όλα τα στοιχεία που βρήκε ο crawler στον ιστότοπο.

Αρχείο Κώδικα 7 wonderspider.py

**classMySpider(**CrawlSpider**):**

name **=**'wonder'

allowed\_domains **=[**"wondergreece.gr"**]**

start\_urls **=[**"http://www.wondergreece.gr/v1/el/Perioxes/"**]**

rules **=(**

Rule**(**LinkExtractor**(** allow**=(),** restrict\_xpaths**=(**'//div[@class="row-fluid regionbox"]//div[@class="rclass"]//h4/a'**)),** follow**=True),**

Rule**(**LinkExtractor**(**allow**=(**r'Perioxes/.\*/Fysi/Paralies/.\*'**,)),** callback**=**'parse\_items'**,**follow**=True),**

**)**

**def**parse\_items**(**self**,**response**):**

sel **=** Selector**(**response**)**

item **=** WondergreeceItem**()**

item**[**'beach'**]=** sel**.**xpath**(**'//div[@class="title span6 pull-left"]/h3/text()'**).**extract**()[**0**]**

item**[**'description'**]=** sel**.**xpath**(**'//section[@class="entry-content media clearfix"]/p[@class="text-justify"]/text() | //section[@class="entry-content media clearfix"]/p[2]/text()'**).**extract**()**

**if** item**[**'description'**]!=[]:**

item**[**'description'**]=** item**[**'description'**][**0**]**

**else:**

item**[**'description'**]=**u'\u0394\u03b5\u03bd \u03c5\u03c0\u03ac\u03c1\u03c7\u03b5\u03b9 \u03c0\u03b5\u03c1\u03b9\u03b3\u03c1\u03b1\u03c6\u03ae'

ph **=** sel**.**xpath**(**'//ul[@class="slides"]//li//img'**)**

fot **=[]**

**for** f **in** ph**:**

photo **=** f**.**xpath**(**'@src'**).**extract**()**

fot**.**append**(**photo**)**

item**[**'photos'**]=** fot

**print** item**[**'photos'**]**

latlon **=** sel**.**xpath**(**'//script'**).**re**(**r'"latlng":(.\*)'**)**

Για την εκτέλεση του spider αρχείου ανοίγουμε το cmd πηγαίνουμε στο φάκελο του πρότζεκτ και εκτελούμε την εντολή **scrapy crawl wonder**.

Τα δεδομένα όμως που εξάγονται από τον crawler πρέπει να αποθηκευτούν στην βάση δεδομένων με σκοπό την επαναχρησιμοποίησή τους, από την εφαρμογή. Για αυτό τον λόγο υπάρχει το αρχείο pipelines.py του πρότζεκτ. Το αρχείο αυτό είναι ο αγωγός σύνδεσης των δεδομένων με την βάση. Πρόκειται για μια κλάση όπου στον δημιουργό της αρχικοποιούνται τα στοιχεία της βάσης δεδομένων και έχει μια μέθοδο στην οποία εκτελούνται SQL εντολές INSERT με σκοπό την εισαγωγή των δεδομένων που εξάγει η αράχνη από τον ιστότοπο στην βάση δεδομένων.

Αρχείο Κώδικα wondergreece/pipelines.py

**classWondergreecePipeline(**object**):**

**def**\_\_init\_\_**(**self**):**

self**.**conn **=** MySQLdb**.**connect**(**user**=**'root'**,** passwd**=**''**,** db**=**'database'**,** host**=**'localhost'**,** charset**=**"utf8"**,** use\_unicode**=True)**

self**.**cursor **=** self**.**conn**.**cursor**()**

#log data to json file

**def** process\_item**(**self**,** item**,** spider**):**

**if** elegxos\_paralias**(**item**[**'beach'**],** item**[**'lat'**],** item**[**'lon'**],** item**[**'address'**],** item**[**'description'**],**'wondergreece.gr'**,** item**[**'photos'**])!=**0**:**

**print**'Inserted to beachplus'

**else:**

**try:**

sql **=**"""INSERT INTO beach (name,lat,lon,region) VALUES (%s,%s,%s,%s)"""

self**.**cursor**.**execute**(**sql**,(**item**[**'beach'**],** item**[**'lat'**],** item**[**'lon'**],** item**[**'region'**]))**

sql3 **=**"""SELECT MAX(id) FROM beach"""

self**.**cursor**.**execute**(**sql3**)**

result **=** self**.**cursor**.**fetchone**()**

sql2 **=**"""INSERT INTO beachplus(address ,source, beach\_id, alternate) VALUES (%s, %s, %s, %s)"""

self**.**cursor**.**execute**(**sql2**,(**item**[**'address'**],**'wondergreece.gr'**,** result**,** item**[**'beach'**]))**

sql4 **=**"""INSERT INTO description(beach\_id, text, source) VALUES(%s, %s, %s)"""

self**.**cursor**.**execute**(**sql4**,(**result**,** item**[**'description'**],**"wondergreece.gr"**))**

**for** item **in** item**[**'photos'**]:**

sql5 **=**"""INSERT INTO photos(beach\_id, ph) VALUES(%s, %s)"""

self**.**cursor**.**execute**(**sql5**,(**result**,** item**,))**

self**.**conn**.**commit**()**

**except** MySQLdb**.**Error**,** e**:**

**print**"Error %d: %s"**%(**e**.**args**[**0**],** e**.**args**[**1**])**

**return** item

Τα δεδομένα όμως που εξάγονται από την αράχνη πρέπει να αποθηκευτούν στην βάση δεδομένων έτσι ώστε να είναι δυνατή η χρησιμοποίηση τους από την εφαρμογή. Για αυτό τον λόγο υπάρχει το αρχείο pipelines.py του πρότζεκτ. Το αρχείο αυτό είναι ο αγωγός σύνδεσης των δεδομένων με την βάση. Πρόκειται για μια κλάση όπου στον δημιουργό της αρχικοποιούνται τα στοιχεία της βάσης δεδομένων και έχει μια μέθοδο στην οποία εκτελούνται SQL εντολές INSERT με σκοπό την εισαγωγή των δεδομένων που εξάγει η αράχνη από τον ιστότοπο στην βάση δεδομένων.

### terrabook.com

Πρόκειται για έναν ιστότοπο ο οποίος μας δίνει σημαντικές πληροφορίες για ελληνικές παραλίες, πάλι και σε αυτή την περίπτωση ζητούμενα είναι το όνομα της παραλίας, το γεωγραφικό μήκος και πλάτος της παραλίας. Περιγραφή και σχόλια σχετικά με την παραλία καθώς και φωτογραφίες. Το αρχείο items.py περιέχει την κλάση – μοντέλο όπου κάθε αντικείμενο που εξάγεται από τον ιστότοπο έχει ως ιδιότητα τα ζητούμενα χαρακτηριστικά που θέλουμε να εξάγουμε.

**from** scrapy**.**item **import** Item**,** Field

**classTerrabookItem(**Item**):**

beach\_name **=** Field**()**

description **=** Field**()**

lng **=** Field**()**

lat **=** Field**()**

region **=** Field**()**

address **=** Field**()**

photos **=** Field**()**

**pass**

Αρχείο Κώδικα terrabook/items.py

Για την εξόρυξη της πληροφορίας από τον ιστότοπο, είναι αναγκαία η δημιουργία ενός αρχείου τύπου spider. Το αρχείο αυτό αποθηκεύεται στoν φάκελο spiders και ονομάζεται terraspy.py. Παρατηρείται ότι οι ιστοσελίδες δεν ακολουθουν κάποιο μοτίβο. Συνεπώς παρατηρούμε στην δομή του ιστότοπου. Η κλάση της αράχνης κληρονομεί από την κλάση CrawlSpider. Κάνοντας χρήση του χαρακτηριστικού **rules** ορίζεται ποια links που πρέπει να ακολουθηθούν από την αράχνη. Στην συγκεκριμένη περίπτωση ακολουθούνται τα Links που υπάρχουν σε συγκεριμένα στοιχεία της HTML δομής του ιστοτόπου. Το χαρακτηριστικό **name** αποτελεί μοναδικό χαρακτηριστικό, ταυτότητα δηλαδή της αράχνης, η οποία καλείται μέσω αυτού. Για την εξόρυξη των συγκεκριμένων χαρακτηριστικών χρησιμοποιούνται xpath selectors με σκοπό να γίνει επιλογή των htm lχαρακτηριστικών που μας δίνουν την ζητούμενη πληροφορία. Παρακάτω παρατήθεται μέρος του αρχείου terraspy.py.

Αρχείο Κώδικα teraspy.py

name **=**"terra"

allowed\_domains **=[**"terrabook.com"**]**

start\_urls **=[**"https://www.terrabook.com/el/"**]**

rules **=(**

Rule**(**LinkExtractor**(** allow**=(),** restrict\_xpaths**=(**'//div[@class="ggclCtnr ggclp120"]//div[@class="ggclImgCntr ggclImg"]/a'**)),** follow**=True),**

Rule**(**LinkExtractor**(** allow**=(),** restrict\_xpaths**=(**'//ul[@id="megaUber"]/li[9]/a'**)),** follow**=True),**

Rule**(**LinkExtractor**(** allow**=(),** restrict\_xpaths**=(**'//li//h4[@class="hidden"]/a'**)),** callback**=**"parse\_items"**,**follow**=True)**

**)**

**def**parse\_items**(**self**,**response**):**

sel **=** Selector**(**response**)**

item **=** TerrabookItem**()**

beach **=** sel**.**xpath**(**"//h1[@id='title']/text()"**).**extract**()[**0**].**strip**()**

item**[**'beach\_name'**]=**''**.**join**(**beach**)**

**print** item**[**'beach\_name'**]**

item**[**'lat'**]=** sel**.**xpath**(**"//script"**).**re**(**r'centerlat =\ ([0-9.]+)'**)**

item**[**'lat'**]=[**str**(**x**)for** x **in** item**[**'lat'**]]**

**print** item**[**'lat'**]**

item**[**'lng'**]=** sel**.**xpath**(**"//script"**).**re**(**r'centerlng =\ ([0-9.]+)'**)**

item**[**'lng'**]=[**str**(**x**)for** x **in** item**[**'lng'**]]**

**print** item**[**'lng'**]**

photos **=** sel**.**xpath**(**"//div[@class='rsTmb']//img/@src"**).**extract**()**

item**[**'photos'**]=** photos

#item['description'] = sel.xpath("//div[@id='description']//p//text()").extract()

#print item['description']

ps **=** sel**.**xpath**(**"//div[@id='description']//p"**)**

Όπως παρατηρείται στο παραπάνω κομμάτι κώδικα, η συνάρτηση rules περιορίζει τον crawler σε ιστοσελίδες που υπάρχουν σε συγκεκριμένα xpaths. Στην συνέχεια η συνάρτηση parse\_items αποθηκέυει στην λίστα items για κάθε ιστότοπο ένα dictionary με τα στοιχεία που βρήκε ο crawler σε αυτούς.Για την εκκίνηση της όλης διαδικασίας ανοίγουμε το cmd και εκτελούμε την εντολή **scrapy crawl terra**.

Το items dictionary πρέπει να αποθηκευτεί σε μια βάση δεδομένων, ετσι λοιπόν αναλαμβάνει το αρχείο pipelines.py είναι υπέυθυνο για αυτή την εκχώρηση.

Αρχείο Κώδικα terrabook/pipelines.py

**def**\_\_init\_\_**(**self**):**

self**.**conn **=** MySQLdb**.**connect**(**user**=**'root'**,** passwd**=**''**,** db**=**'database'**,** host**=**'localhost'**,** charset**=**"utf8"**,** use\_unicode**=True)**

self**.**cursor **=** self**.**conn**.**cursor**()**

#log data to json file

**def**process\_item**(**self**,** item**,** spider**):**

**if** elegxos\_paralias**(**item**[**'beach\_name'**],** float**(**item**[**'lat'**][**0**]),** float**(**item**[**'lng'**][**0**]),** item**[**'address'**],** item**[**'description'**],**'terrabook.com'**,** item**[**'photos'**])!=**0**:**

**print**'Inserted to beachplus'

**else:**

**try:**

sql **=**"""INSERT INTO beach (name,lat,lon,region) VALUES (%s,%s,%s,%s)"""

self**.**cursor**.**execute**(**sql**,(**item**[**'beach\_name'**],** item**[**'lat'**],** item**[**'lng'**],** item**[**'region'**]))**

sql3 **=**"""SELECT MAX(id) FROM beach"""

self**.**cursor**.**execute**(**sql3**)**

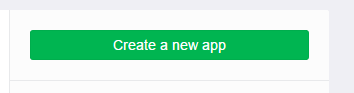
result **=** self**.**cursor**.**fetchone**()**

sql2 **=**"""INSERT INTO beachplus(address ,source, beach\_id, alternate) VALUES (%s, %s, %s, %s)"""

self**.**cursor**.**execute**(**sql2**,(**item**[**'address'**],**'terrabook.com'**,** result**,** item**[**'beach\_name'**]))**

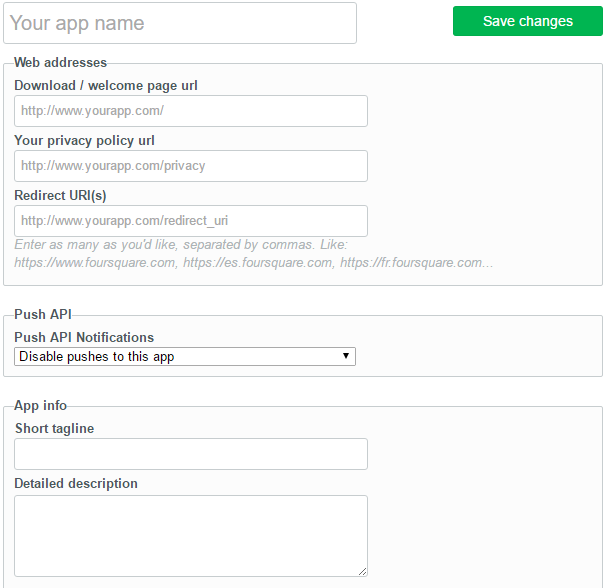
### Foursquare

Το Foursquare διαθέτει API οπότε δεν χρειάζεται να γίνει χρήση του Scrapy framework ή άλλου είδους εργαλείου crawling. Το API του Foursquare διαθέτει βιβλιοθήκη για την Python. Για την εγκαταστασή της το μόνο που αρκεί είναι να ανοίξουμε το command window και εκτελούμε την εντολή **pip install foursquare**. Αυτό που απομένει είναι η απόκτηση των μυστικών κλειδιών χρήστη (client\_id, client\_secret). Έτσι λοιπόν αυτό που μένει είναι να πάμε στην επίσημη σελίδα του Foursquare, να μεταβούμε στην καρτέλα developers και πάλι να πατήσουμε την καρτέλα **MyApps**. Επιλέγουμε το κουμπί **Create new app**.



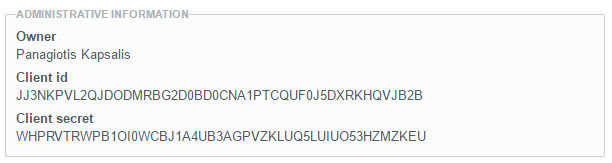
Εικόνα Δημιουργία Εφαρμογής Foursquare

Στην συνέχεια οδηγούμαστε σε μια φόρμα όπου πρέπει να συμπληρώσουμε τα στοιχεία της εφαρμογής. Εφόσον ολοκληρώσουμε την συμπλήρωση των στοιχείων πατάμε το κουμπί **Save changes**.



Εικόνα Στοιχεία της Εφαρμογής

Μετά την αποθήκευση των στοιχείων της εφαρμογής έχουμε διαθέσιμα τα **client\_secret**και **client\_id** της εφαρμογής. Τα στοιχεία αυτά είναι απαραίτητα για το authentication προς το Foursquare.



Εικόνα 46 Client secret – Client id

Έχοντας διαθέσιμα λοιπόν τα στοιχεία της εφαρμογής Foursquare μπορούμε να συνδεθούμε με το API του. Το αρχείο beach\_search.py είναι το αρχείο που περιέχει τον κώδικα που προσπελάυνει την πληροφορία του Foursquare για τις παραλίες της Ελλάδας. Πιο συγκεκριμένα περιέχει την συνάρτηση extract(p\_city) η οποία δέχεται ως όρισμα το όνομα μιας πόλης και δίνει ως αποτέλεσμα τις παραλίες που θεωρεί το Foursquare API ως κοντινές της. Αρχικά πρέπει να γίνει η σύνδεση με το Foursquare:

client **=** foursquare**.**Foursquare**(**client\_id**=**'0NKJU5YU3205AZXBGQWLDWRLREZRNBGWPDHYR2PXNFOFOGIY'**,**client\_secret**=**'DIWIFQPLO0HLBBKOCYLAO1FRK5KA0DDGS1ZTVLN2RBSQST3E'**)**

Στην συνέχεια εκτελείται το ερώτημα προς το FoursquareAPI. Παρατηρούμε ότι η εντολή που φέρει το ερώτημα έχει ως παραμέτρους τις λεξεις “Beach”, το όνομα της πόλης που δίνεται ως όρισμα στην συνάρτηση extract και το id της κατηγορίας που ανήκουν τα αποτελέσματα που επιθυμούμε, δηλαδή παραλίες. Τα category ids μπορούμε να τα βρούμε στο documentation του Foursquare. Το Foursquare δίνει ως αποτέλεσμα ένα JSON αντικείμενο, έτσι λοιπόν για να διαχωρίσουμε την πληροφορία το μόνο που αρκεί είναι να προσπελαύνουμε το JSON object μέσω βρόχων και να αποθηκεύσουμε τις πληροφορίες που επιθυμούμε στην βάση δεδομένων. Από το Foursquare εξάγουμε το όνομα παραλίας, συντεταγμένες, φωτογραφίες των χρηστών του Foursquare, καθώς και σχόλια των χρηστών για την παραλία. Επειδή υπάρχει περίπτωση παραπάνω από μια πόλη να δίνει ως αποτέλεσμα την ίδια παραλία υπάρχει ο κίνδυνος των διπλότυπων. Έτσι λοιπόν για να αποφύγουμε την αποθήκευση πληροφορίας για την ίδια παραλία παραπάνω από μία φορές , χρησιμοποιούμε τον βοηθητικό πίνακα tsek. Σε αυτόν τον πίνακα εγγράφουμε τα ids των σελίδων, έτσι με αυτόν τον τρόπο κάθε φορά που έχουμε ένα id που υπάρχει στον πίνακα tsek, δεν γίνεται αποθήκευση πληροφορίας στην βάση δεδομένων.

Για να έχουμε πρόσβαση στα σχόλια των χρηστών για κάθε αποτέλεσμα του Foursquare API, σύμφωνα με το ερώτημα που τέθηκε, χρησιμοποιούμε τις συναρτήσεις του Foursquare **client.venues.tips(id)** με όρισμα το αναγνωριστικό της παραλίας στο Foursquare, όμοια για να πάρουμε τις φωτογραφίες χρησιμοποιούμε την συνάρτηση **client.venues.photos(id, params={'group': 'venue'})**. Τα οποία αντίστοιχα αποθηκεύονται στην βάση δεδομένων.

Αρχείο Κώδικα Απόσπασμα beach\_search.py

**if** venues**[**"stats"**][**"tipCount"**]!=**0**:**

#tip\_details(venues["id"],plaz\_name,result)

tip **=** client**.**venues**.**tips**(**venues**[**"id"**])**

**for** item **in** tip**[**"tips"**][**"items"**]:**

tip\_text **=** item**[**"text"**]**

sql6 **=**"""INSERT INTO description(beach\_id, text, source) VALUES(%s, %s, %s)"""

args6 **=(**result**,** tip\_text**,** source**)**

cursor**.**execute**(**sql6**,** args6**)**

db**.**commit**()**

**print**'End\n'

ph**=**client**.**venues**.**photos**(**plaz\_id**,** params**={**'group'**:**'venue'**})**

**for** photo **in** ph**[**"photos"**][**"items"**]:**

photo\_url**=**photo**[**"prefix"**]+**"width500"**+**photo**[**"suffix"**]**

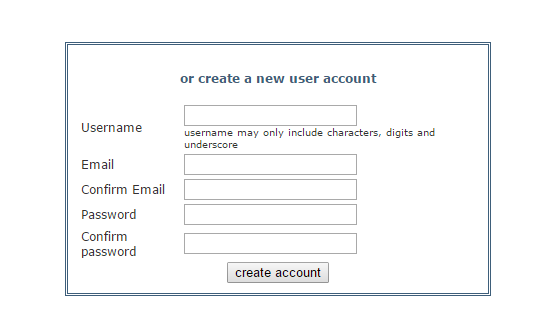
sqlph **=**"""INSERT INTO photos (beach\_id, ph) VALUES (%s, %s)"""

cursor**.**execute**(**sqlph**,(**result**,** photo\_url**))**

db**.**commit**()**

### Geonames.org

Ομοίως και ο ιστότοπος geonames.org διαθέτει API, οπότε δεν χρειάζεται κάποιος crawlerγια να προσπελαύνει πληροφορία από αυτό. Για να μπορεί κάποιος να συνδεθεί με το APIτου geonames.org χρειάζεται να δημιουργήσει ένα account. Για να γίνει αυτό μεταβαίνουμε στον σύνδεσμο [**http://www.geonames.org/login**](http://www.geonames.org/login) και συμπληρώνουμε την ακόλουθη φόρμα στην παρακάτω εικόνα. Εφόσον γίνει η συμπλήρωση των στοιχείων και γίνει η εξακρύβωση του χρήστη μέσω του email σύνδεσης, για να γίνει προσπέλαση πληροφορίας με το geonames webservice αρκεί να δοθεί ως παράμετρος στο URL το username. Ανάλογα με την παράμετρο που τιθεται στο URL, η απόκριση του webservice του ιστότοπου είναι σε μορφή JSONή σε μορφή XML. Πιο συγκεκριμένα θέτουμε την παράμετρο **searchJSON** για JSON μορφή και την παράμετρο **searchXML** για XML μορφή.



Εικόνα Δημιουργία λογαριασμού στο geonames.org

Το geonames API μας τροφοδοτεί με μια πληθώρα πληροφοριών όπως το όνομα μιας παραλίας σε ποια χώρα είναι, σε ποιον νομό, συντεταγμένες τον κωδικό της χώρας, τον κωδικό του νομού της χώρας, καθώς και διαφορετικές ονομασίες των παραλιών. Στην συγκεκριμένη περίπτωση χρειαζόμαστε το όνομα τις συντεταγμένες, καθώς και διαφορετικές ονομασίες μιας παραλίας. Το geonames webservice παράγει τα αποτελέσματα σε JSON αλλά και σε XML format. Πατώντας στον browser το URL [**http://api.geonames.org/searchJSON?q=beach&country=GR&maxRows=1000&lang=el&username=kapsali**](http://api.geonames.org/searchJSON?q=beach&country=GR&maxRows=1000&lang=el&username=kapsali), παρατηρούμε ο browser να δίνει έξοδο ένα JSON αντικείμενο. Πιο συγκεκριμένα θέτουμε στο geonamesνα ψάξει για παραλίες στην χώρα με κωδικό το GR, το JSON objectνα έχει αποτέλεσμα ως 1000 γραμμές να είναι στα ελληνικά για τον χρήστη kapsali. Με χρήση της βιβλιοθήκης **urllib2** της Python είναι δυνατή η επεξεργασία του JSON αντικειμένου και η αποθήκευση του στην βάση δεδομένων.

Αρχείο Κώδικα Απόσπασμα από το geo.py

response **=** urllib2**.**urlopen**(**' http://api.geonames.org/searchJSON?q=beach&country=GR&maxRows=1000&lang=el&username=kapsali'**)**

json\_object **=** json**.**load**(**response**)**

**for** item **in** json\_object**[**u'geonames'**]:**

**if** item**[**u'countryCode'**]==**'GR'**:**

**print**'BEGIN'

**try:**

**print** item**[**u'toponymName'**]**

**except:**

**print** remove\_accents**(**item**[**u'toponymName'**])**

**print** item**[**u'lat'**]**

**print** item**[**u'lng'**]**

**print** item**[**u'adminName1'**]**

**try:**

name **=** item**[**u'name'**]**

**print** name

**except:**

name **=** remove\_accents**(**item**[**u'name'**])**

**print** name

sql1 **=**"""INSERT INTO beach(name, lat, lon, region) VALUES(%s, %s, %s, %s) """

Υπάρχει περιορισμός στον αριθμό των γραμμών του JSON object στις 1000, καθώς για περισσότερα αποτελέσματα πρέπει κάποιος να δημιουργήσει λογαριασμό επί πληρωμής.

Το geonames webservice είναι χρήσιμο, εκτός από την συμπλήρωση της βάσης γνώσης, για την τροφοδότηση της συνάρτησης extract(p\_city) με τοπονύματα της Ελλάδας. Σε αυτή την περίπτωση ακολουθείται διαφορετικός τρόπος. Εγκαθίσταται η βιβλιοθήκη της Python για το geonames webservice, εκτελώντας την εντολή **pip install geonames\_rdf**.

Αρχείο Κώδικα Απόσπασμα από το run\_me.py

items **=[]**

result **=** sa**.**query**(**''**).**country**(**'gr'**).**max\_rows**(**1000**).**execute**()**

**for** id\_**,** name **in** result**.**get\_flat\_results**():**

item **=** geonames**.**compat**.**make\_unicode**(**" [{1}]"**).**format**(**id\_**,** name**)**

items**.**append**(**item**)**

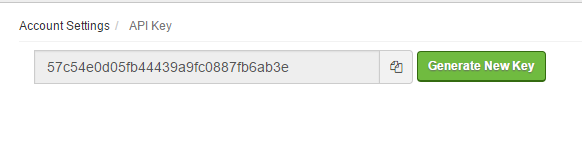
items **=[**unicodedata**.**normalize**(**'NFKD'**,** x**).**encode**(**'latin-1'**,**'ignore'**)for** x **in** items**]**

Στο παραπάνω κομμάτι κώδικα αποστέλεται ερώτημα προς το geonames webservicesγια όλα τα τοπονύματα, το ‘’ σημαίνει όλα τα τοπονύματα, της χώρας με κωδικό GR. Τα αποτελέσματα αποθηκεύονται στην λίστα items, της οποίας τα στοιχεία τροφοδοτούν ητν συνάρτηση extract.

### Υπόλοιποι Ιστότοποι

Επίσης εκτός από τους προαναφερόμενους ιστότοπους, εξάγαμε πληροφορία και από άλλους ιστότοπους όπως το e-zakynthos.gr, patmos-island.com, destinationskiathos.com, cleanbeaches.gr, Thasos-view.com. Πρόκειται για ιστότοπους που δεν περιέχουν μεγάλο όγκο πληροφορίας και για την εξαγωγή της πληροφορίας χρησιμοποιήθηκε το PortiaScrapinghub. Εφόσον επιλεχθούν τα html στοιχεία των ιστότοπων, αποθηκεύονται στον λογαριασμό. Για να είναι δυνατή η αξιοποίηση της πληροφορίας χρησιμοποιούνται οι εντολές του Scrapinghub API.

Για να γίνει σύνδεση με το Scrapinghub API χρησιμοποιούμε την συνάρτηση Connection της βιβλιοθήκης Scrapinghub, η οποία πέρνει ως όρισμα το Scrapinghub API key, το οποίο παίρνουμε από το προφίλ στο Scrapinghub.



Εικόνα Scrapinghub API key

Η σύνδεση με το Scrapinghub APIεπιτυγχάνεται με τον παρακάτω κώδικα.

Αρχείο Κώδικα Σύνδεση με το Scrapinghub API

**from** scrapinghub **import** Connection

**import** time

conn **=** Connection**(**'57c54e0d05fb44439a9fc0887fb6ab3e'**)**

**print** conn

**print** conn**.**project\_ids**()**

project **=** conn**[**77645**]**

jobs **=** project**.**jobs**(**state**=**'finished'**)**

jobs\_id **=[**x**.**id **for** x **in** jobs**]**

**print** jobs\_id

job **=** project**.**job**(**u'77645/1/1'**)**

Αρχικά για να γίνει η σύνδεση χρησιμοποιείται η συνάρτηση Connection, στην συνέχεια γίνεται εκτύπωση όλων των ids των πρότζεκτ ενός λογαριασμού. Για να επιλεγεί ένα πρότζεκτ χρησιμοποιείται το id του. Κάθε πρότζεκτ αποτελείται από jobs (εργασίες). Κάθε εργασία σηματοδοτείται από μια ετικέτα, οι εργασίες που φέρουν την ετικέτα “finished” περιέχουν άμεσα προσπελάσιμη πληροφορία. Η πληροφορία είναι σε μορφή JSON (default) και προσπελαύνεται με την χρήση της συνάρτησης jobs.items(). Στην συνέχεια με χρήση βρόχων επανάληψης στο JSON object αποθηκεύουμε την πληροφορία στην βάση δεδομένων.

## Ταξινόμηση της Πληροφορίας

Υπάρχει η δυνατότητα διαφορετικές ιστοσελίδες να αναφέρουν την ίδια παραλία με διαφορετικές ονομασίες. Για παράδειγμα ένας ιστότοπος να αναγράφει την παραλία ως Kalogria Beach, ένας άλλος ως Καλόγρια, άλλος ως Καλόγρια (KalogriaBeach) και ενας άλλος ιστότοπος να την αναφέρει ως παραλία Κάτω Αχαίας. Αυτές οι ονομασίες δεν πρέπει να καταχωρηθούν στην βάση δεδομένων ως διαφορετικές παραλίες, για τον λόγο αυτό δημιουργήθηκε ο πίνακας beachplus της βάσης δεδομένων και πιο συγκεκριμένα τα πεδίο του πίνακα alternate. Γίνεται έλεγχος, μέσω βιβλιοθηκών της Python, των ονομάτων των παραλιών. Γίνεται χρήση της βιβλιοθήκης transliterate, για μετατροπή των λατινικών χαρακτήρων σε ελληνικους. Για παράδειγμα αν ο ιστότοπος terrabook.com αναγράφει την παραλία ως Kalogria Beach και ο ιστότοπος Wondergreece.g rως Παραλία Καλόγρια, γίνεται χρήση της βιβλιοθήκης transliterate για μετατροπή των λατινικών χαρακτήρων σε ελληνικούς και στην συνέχεια γίνεται χρήση της συνάρτησης SequenceMatcher() της βιβλιοθήκης difflib η οποία επιστρέφει έναν δεκαδικό αριθμό. Όσο αυτός ο αριθμός προσεγγίζει το 1 τόσο πιο πολύ μοιάζουν τα ονόματα. Επειδή όμως υπάρχει περίπτωση 2 παραλίες να έχουν το ίδιο όνομα αλλά να είναι σε διαφορετικές τοποθεσίες. Για παράδειγμα υπάρχει όνομα Χρυσή ΄Αμμος στην Άνδρο και στην Μυτιλήνη αντίστοιχα πρέπει να ληφθούν υπόψην και οι γεωγραφικές συντεταγμένες των παραλιών. Συνεπώς όταν βρίσκουμε πληροφορίες σχετικά με το όνομα μιας παραλίας, ελέγχουμε αν υπάρχει ένα συναφές όνομα ήδη στην βάση δεδομένων, σε περίπτωση που η χιλιομετρική απόσταση που φέρουν τα 2 ονόματα (υπολογίζεται μέσω των συντεταγμένων) είναι μικρότερη του 1 χιλιομέτρου τότε το νέο όνομα καταχωρείται στον πίνακα beachplus με το id του ονόματος στο οποίο μοίαζει και ήδη υπάρχει στον πίνακα beach, το πεδίο alternate παίρνει το νέο όνομα και οι φωτογραφίες, περιγραφές, σχόλια χρηστών εισάγονται στους αντίστοιχους πίνακες με το id του ονόματος παραλίας που ήδη υπάρχει καταχωρημένο. Επίσης πρέπει να σημειωθεί ότι παραλίες που έχουν χιλιομετρική απόσταση μικρότερη ή ίση των 100 μέτρων και ας έχουν διαφορετικά ονόματα θεωρούνται ίδια παραλία.

### Συνάρτηση similar

Ο εντοπισμός όμοιων τοπονυμάτων γίνεται από την συνάρτηση **similar(a,b)**. Η συνάρτηση αυτή δέχεται ως όρισμα 2 τοπονύματα. Απομονώνει αλφαρηθμητικά από τα ονόματα παραλιών που εμφανίζονται πολλές φορές σε αυτά όπως τα αλφαριθμητικά παραλία, Παραλία, Beach, paralia, κτλπ. Επίσης απογυμνόνονται από χαρακτήρες όπως οι παρενθέσεις, οι παύλες. Με την χρήση της βιβλιοθήκης dfflib υπολογίζει τον δείκτη ομοιότητας μεταξύ των δύο λέξεων αν αυτός ο δείκτης είναι μεγαλύτερος η ίσος του 0.80, προσεγγιστικά, τα ονόματα θεωρούνται όμοια. Για παράδειγμα αν έχουμε 2 ονόματα παραλιών δηλαδή, Kalogria (Παραλία Καλόγρια) και Καλογρια (Kalogr. Beach). Η συνάρτηση similar μόλις κληθεί ξεχωρίζει τα αλφαριθμητικά από τις παρενθέσεις, δηλαδή πλέον δημιουργούνται 4 ονόματα τα Καλόγρια, Παραλία Καλόγρια , από το πρώτο τοπόνυμο, και το Καλόγρια, Kalogr. Beach. Έτσι λοιπόν η συνάρτηση similar, αναλαμβάνει να κάνει συγκρίσεις μεταξύ αυτών των 2 ζευγών και επιστρέφει τον μέγιστο δείκτη ομοιότητας.

Αρχείο Κώδικα Απόσπασμα από τον κώδικα της συνάρτησης similar

a **=** a**.**lower**()**

b **=** b**.**lower**()**

p **=**u'\u03c0\u03b1\u03c1\u03b1\u03bb\u03af\u03b1'

p1 **=**u'\u03c0\u03b1\u03c1\u03b1\u03bb\u03b9\u03b1'

a **=** a**.**replace**(**'beach'**,**''**)**

a **=** a**.**replace**(**'paralia'**,**''**)**

a **=** a**.**replace**(**p**,**''**)**

a **=** a**.**replace**(**p1**,**''**)**

b **=** b**.**replace**(**'beach'**,**''**)**

b **=** b**.**replace**(**'paralia'**,**''**)**

b **=** b**.**replace**(**p**,**''**)**

b **=** b**.**replace**(**p1**,**''**)**

a**=**a**.**strip**()**

b**=**b**.**strip**()**

**print** a

**print** b

seq **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

Στο παραπάνω κομμάτι κώδικα φαίνεται αυτό που αναφέραμε και πιο πριν, ότι απομονώνουμε τα αλφαριθμητικά – λέξεις που εμφανίζονται πιο συχνά στο όνομα μιας παραλίας, όπως παραλία, beach, paralia, κτλπ. Οι μεταβλητές p και p1 φέρουν Unicode τιμές, είναι οι αντιστοιχες λέξεις παραλία και Παραλία. Είναι σε αυτή την μορφή καθώς ο υπολογιστής διαβάζει τους ελληνικούς χαρακτήρες σε αυτή την μορφή οπότε για να γίνει η αντικατάσταση της λέξης παραλίας χρησιμοποιούμε την συνάρτηση replace της Python με ορίσματα την Unicode τιμή των λέξεων παραλία και Παραλία. Στην συνέχεια εφόσον έχουμε απογυμνώσει τα ονόματα των παραλιών από τις πιο συχνα εμφανιζόμενες λέξεις, η σύγκριση ομοιότητας των ονομάτων γίνεται με χρήση της συνάρτησης SequenceMatcher της dfflib βιβλιοθήκης.

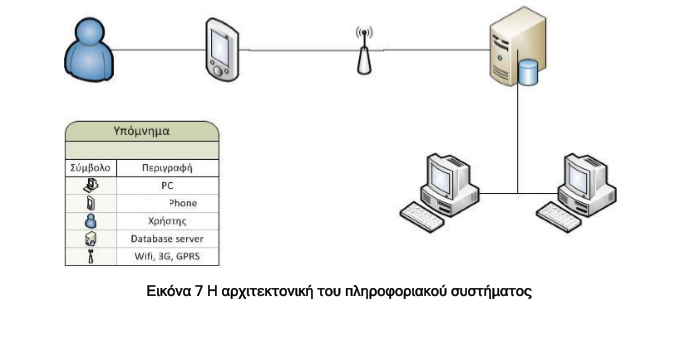
Η όλη λογική της σύγκρισης των ονομάτων βασίστηκε στα κοινά γράμματα που έχουν 2 λέξεις αναμεταξύ τους και κατά συνέπεια ο υπολογισμός του ποσοστού ομοιότητας. Προκειται ουσιαστικά για την χρήση του αλγορίθμου **Ratcliff/ Obershelp**. Ο αλγόριθμος αυτός επιστρέφει 1 στην απόλυτη ομοιότητα 2 αλφαριθμητικών. Πιο συγκεκριμένα. Έστω 2 λέξεις, ο αλγόριθμος συνενώνει τις 2 αυτές λέξεις, σε σειρά, σε μια, εντοπίζει το σύνολο των όμοιων χαρακτήρων και τους εναποθέτει στην μέση του κοινού αλφαριθμητικού. Αριστερά και δεξιά είναι οι χαρακτήρες που δεν είναι όμοιοι. Στο επόμενο βήμα ο αλγόριθμος, παίρνει τους χαρακτήες αριστερά και δεξιά και τους εναποθέτει στην σειρά και αναζητά πάλι για όμοιο σύνολο χαρακτήρων. Πρόκειται για έναν ευέλικτο αλγόριθμος ο οποίος υπολογίζει τον μέγιστο αριθμό όμοιων χαρακτήρων, συνεπώς λέξεις όπως kalogria, kalogr, kaloggria έχουν μεγάλο δείκτη ομοιότητας. Ο αγόριθμος Ratcliff/ Obershelp εκφράζεται από τον παρακάτω τύπο

Όπου **K** είναι ο αριθμός των όμοιων χαρακτήρων και α,b τα μήκη των αλφαρηθμητικών. Το σύνολο των όμοιων χαρακτήρων ονομάζεται **anchor** , η τιμή του K αυξάνεται βάσει του μήκους του anchor. Τότε το πλήθος των χαρακτήρων αριστερά και δεξιά του anchor πρέπει να εξετασταστούν σαν ένα αλφαριθμητικό και να βρεθεί ένα νέο anchor για αυτούς. Η διαδικασία επαναλαμβάνεται μέχρις ότου εξεταστούν οι χαρακτήρες και των 2 αλαφαριθμητικών.

1. Υλοποίηση Πληροφοριακού Συστήματος

# Αρχιτεκτονική Πληροφοριακού Συστήματος

Τα δεδομένα που υπάρχουν στην βάση δεδομένων πρέπει να συνδεθούν με την εφαρμογή. Για την δουλειά αυτή γίνεται η χρήση webservices. Για την υλοποίηση των webservices χρησιμοποιείται το Flask framework της Python.



Εικόνα Αρχιτεκτονική Πληροφοριακού Συστήματος

Οι RESTful webservices που χρησιμοποιήθηκαν για την επικοινωνία του Database serverμε την εφαρμογή είναι οι παρακάτω:

### get\_beach

Η κλάση get\_beach, υλοποιεί το ομόνυμο webservice, περιέχει την συνάρτηση get(idd) η οποία παίρνει ως όρισμα το id της παραλίας της οποία θέλουμε να επιστραφεί το όνομα. Η get(idd) μέσω SQL ερωτημάτων κάνει SELECT το όνομα της παραλίας που έχει id το όρισμα της συνάρτησης. Με την χρήση της συνάρτησης

Αρχείο Κώδικα κλάση get\_beach

**class get\_beach(**Resource**):**

*@app.route***(**'/get\_beach/<idd>'**,** methods**=[**'GET'**])**

**def**get**(**idd**):**

servername **=**"localhost"

username **=**"root"

password **=**""

dbname **=**"database"

q\_list\_one **=**"SELECT name FROM beach WHERE id = %s"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cur **=** db**.**cursor**()**

json\_return **={}**

**try:**

cur**.**execute**(**q\_list\_one**,(**idd**,))**

r\_list\_one **=** cur**.**fetchall**()**

items **=[]**

**for** data **in** r\_list\_one**:**

i **={**'name'**:**data**[**0**]}**

items**.**append**(**i**)**

**except** Exception **as** error**:**

**return{**'error'**:**str**(**error**)}**

**finally:**

cur**.**close**()**

db**.**close**()**

**return** jsonify**(**Items **=** items**)**

jsonify το αποτέλεσμα των SQL ερωτημάτων μετατρέπονται σε JSON μορφή, έτσι

ώστε να επιστραφούν από το webservice προς την εφαρμογή.

### GetAllItems

Η κλάση GetAllItems, υλοποιεί το webservice GetAllItems, περιέχει την getAll(). Η συνάρτηση αυτή δεν δέχεται ορίσματα, συνδέεται με την βάση δεδομένων. Μέσω SQL ερωτημάτων επιστρέφει όλα τα δεδομένα του πίνακα beach. Δηλαδή πιο συγκεκριμένα για κάθε id, επιστρέφει το όνομα της παραλίας, το γεωγραφικό μήκος και πλάτος και μια φωτογραφία της παραλίας. Τα δεδομένα αυτά επιστρέφονται σε μορφή JSON.

Αρχείο Κώδικα κλάση GetAllItems

**classGetAllItems(**Resource**):**

*@app.route***(**'/GetAllItems'**,** methods**=[**'GET'**])**

**def**getAll**():**

servername **=**"localhost"

username **=**"root"

password **=**""

dbname **=**"database"

queryAll **=**"SELECT name, id, photo FROM beach"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cur **=** db**.**cursor**()**

json\_return **={}**

**try:**

cur**.**execute**(**queryAll**)**

listall **=** cur**.**fetchall**()**

items **=[]**

**for** data **in** listall**:**

j**={**'name'**:**data**[**0**],**'id'**:**data**[**1**],**'photo'**:**data**[**2**]}**

items**.**append**(**j**)**

**except** Exception **as** error**:**

**return{**'error'**:**str**(**error**)}**

**finally:**

cur**.**close**()**

db**.**close**()**

**return** jsonify**(**Items **=** items**)**

### getCoordinates

Η κλάση getCoordinates υλοποιεί το getCoordinates webservice, περιέχει την συνάρτηση getlatlon(idd). Η συνάρτηση αυτή παίρνει ως όρισμα το id της παραλίας της οποία επιθυμούμε να μάθουμε το γεωγραφικό μήκος και πλάτος, με την χρήση SQL ερωτημάτων επιστρέφει το όνομα τις συντεταγμένες και το id της παραλίας.Τα δεδομένα του webservice στέλνονται σε JSON format.

Αρχείο Κώδικα κλάση getCoordinates

**classgetCoordinates(**Resource**):**

*@app.route***(**'/getCoordinates/<idd>'**,** methods**=[**'GET'**])**

**def**getlatlon**(**idd**):**

servername **=**"localhost"

username **=**"root"

password **=**""

dbname **=**"database"

querylatlon **=**"SELECT lat, lon, name, id FROM beach WHERE id = %s"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cur **=** db**.**cursor**()**

json\_return **={}**

**try:**

cur**.**execute**(**querylatlon**,(**idd**,))**

listcoords **=** cur**.**fetchall**()**

items **=[]**

**for** data **in** listcoords**:**

jc **={**'latitude'**:**float**(**data**[**0**]),**'longitude'**:**float**(**data**[**1**]),**'name'**:**data**[**2**],**'id'**:** data**[**3**]}**

items**.**append**(**jc**)**

**except** Exception **as** error**:**

**return{**'Error'**:**str**(**error**)}**

**finally:**

cur**.**close**()**

db**.**close**()**

**return** jsonify**(**Items **=** items**)**

### getAlternateNames

Ηκλάση getAlternateNames, υλοποιεί το ομόνυμο webservice, περιέχει την συνάρτηση getAltNames(idd), η οποία παίρνει ως όρισμα το id της παραλίας της SQL ερωτημάτων προς τον πίνακα beachplus, η συνάρτηση επιστρέφει σε JSON μορφή τα διαφορετικά ονόματα μιας παραλίας καθώς και την πηγή (ιστότοπος) κάθε ονόματος.

Αρχείο Κώδικα κλάση getAlternateNames

**class getAlternateNames(**Resource**):**

*@app.route***(**'/getAlternateNames/<idd>'**,** methods **=[**'GET'**])**

**def**getAltNames**(**idd**):**

servername **=**"localhost"

username **=**"root"

password **=**""

dbname **=**"database"

queryalt **=**"SELECT alternate, source, beach\_id FROM beachplus WHERE beach\_id = %s"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cur **=** db**.**cursor**()**

json\_return **={}**

**try:**

cur**.**execute**(**queryalt**,(**idd**,))**

listalt **=** cur**.**fetchall**()**

items **=[]**

**for** data **in** listalt**:**

jalt **={**'Names'**:**data**[**0**],**'Source'**:** data**[**1**],**'Id'**:** data**[**2**]}**

items**.**append**(**jalt**)**

**except** Exception **as** error**:**

**return{**'Error'**:**str**(**error**)}**

**finally:**

cur**.**close**()**

db**.**close**()**

**return** jsonify**(**Items **=** items**)**

### getAddress

Η κλάση getAddress, υλοποιεί το ομόνυμο webservice, περιέχει την συνάρτηση getAd(idd), η συνάρτηση αυτή παίρνει ως όρισμα το idτης παραλίας της οποία θέλουμε να επιστραφεί η διεύθυνση.Η διεύθυνση είναι πεδίο του πίνακα beach. Για τον προσδιορισμό των διευθύνσεων χρησιμοποιείται η βιβλιοθήκη geopy. Η βιβλιοθήκη αυτή περιέχει πολλά geocoding webservices. Το default webserviceπου χρησιμοποιείται είναι το OpenStreetMap Nominatim. Το αποτέλεσμα του webservice είναι σε JSONμορφή.

Αρχείο Κώδικα κλάση getAddress

**classgetAddress(**Resource**):**

*@app.route***(**'/getAddress/<idd>'**,** methods **=[**'GET'**])**

**def**getAd**(**idd**):**

servername **=**"localhost"

username **=**"root"

password **=**""

dbname **=**"database"

queryforadd **=**"SELECT address, alternate, beach\_id FROM beachplus WHERE beach\_id = %s"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cur **=** db**.**cursor**()**

json\_return **={}**

**try:**

cur**.**execute**(**queryforadd**,(**idd**,))**

listad **=** cur**.**fetchone**()**

items **=[]**

#for data in listad:

jad **={**'address'**:**listad**[**0**],**'Names'**:**listad**[**1**],**'Id'**:**listad**[**2**]}**

items**.**append**(**jad**)**

**except** Exception **as** error**:**

**return{**'Error'**:**str**(**error**)}**

**finally:**

cur**.**close**()**

db**.**close**()**

**return** jsonify**(**Items **=** items**)**

### getDetails

Η κλάση getDetails, υλοποιεί το ομόνυμο webservice, περιέχει την συνάρτηση getD(idd), παίρνει ως όρισμα το id της παραλίας για την οποία ζητούνται περιγραφές και σχόλια χρηστών ιστοτόπων. Η συνάρτηση μέσω SQL ερωτημάτων συνδέεται με τον πίνακα description και επιστρέφει όλες τις εγγραφές του πίνακα με id το όρισμα της συνάρτησης. Τα αποτελέσματα επιστρέφονται σε JSON format.

Αρχείο Κώδικα Απόσπασμα κλάσης getDetails

queryfordesc **=**"SELECT text, source, beach\_id FROM description WHERE beach\_id = %s"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cur **=** db**.**cursor**()**

json\_return **={}**

**try:**

cur**.**execute**(**queryfordesc**,(**idd**,))**

listdesc **=** cur**.**fetchall**()**

items **=[]**

**for** data **in** listdesc**:**

jdesc **={**'description'**:**data**[**0**],**'source'**:**data**[**1**],**'id'**:**data**[**2**]}**

items**.**append**(**jdesc**)**

Αρχείο Κώδικα 23Αρχείο Κώδικα 24

### getPhotos

Η κλάση getPhotos, υλοποιεί το ομόνυμο webservice, περιέχει την συνάρτηση, getPh(idd), παίρνει ως όρισμα το id της παραλίας για την οποία ζητούνται οι φωτογραφίες. Η συνάρτηση μέσω SQL ερωτημάτων, επιστρέφει όλες τις εγγραφές του πίνακα photos σχετικές με το id που δίνεται ως όρισμα στην συνάρτηση σε μορφή JSON.

Αρχείο Κώδικα κλάση getDetails

**classgetPhotos(**Resource**):**

*@app.route***(**'/getPhotos/<idd>'**,** methods **=[**'GET'**])**

**def**getPh**(**idd**):**

servername **=**'localhost'

username **=**'root'

password **=**''

dbname **=**'database'

queryPhoto **=**"SELECT ph, beach\_id FROM photos WHERE beach\_id = %s"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cur **=** db**.**cursor**()**

json\_return **={}**

**try:**

cur**.**execute**(**queryPhoto**,(**idd**,))**

listph **=** cur**.**fetchall**()**

items **=[]**

**for** data **in** listph**:**

jph **={**'photo'**:**data**[**0**],**'id'**:**data**[**1**]}**

items**.**append**(**jph**)**

**except** Exception **as** error**:**

**return{**'Error'**:**str**(**error**)}**

**finally:**

cur**.**close**()**

db**.**close**()**

**return** jsonify**(**Items **=** items**)**

### getRegionBeaches

Η κλάση getRegionBeaches, υλοποιείτο ομόνυμο webservice, περιέχει την συνάρτηση getRbeaches(pointer), παίρνει ως όρισμα έναν ακέραιο. Αυτός ο ακέραιος δείχνει στην λίστας region που περιέχει όλες τις περιφέρειες της Ελλάδας. Μέσω SQL ερωτημάτων η συνάρτηση επιστρέφει όλες τις εγγραφές του πίνακα beach όπου το πεδίο region ισούται με την τιμή της λίστας region στην θέση pointer. Τα αποτελεσματα επιστρέφονται σε μορφή JSON.

Αρχείο Κώδικα Απόσπασμα κλάσης getRegionBeaches

region **=[**"Περιφέρεια Πελοποννήσου"**,**"Περιφέρεια Κρήτης"**,**"Περιφέρεια Ιόνιων νησιών"**,**"Περιφέρεια Νοτίου Αιγαίου"**,**"Περιφέρεια Ανατολικής Μακεδονίας και Θράκης"**,**"Περιφέρεια Στερεάς Ελλάδας"**,**"Περιφέρεια Βόρειου Αιγαίου"**,**"Περιφέρεια Αττικής"**,**"Περιφέρεια Θεσσαλίας"**,**"Περιφέρεια Κεντρικής Μακεδονίας"**,**"Περιφέρεια Ηπείρου"**,**"Περιφέρεια Δυτικής Ελλάδας"**]**

queryAll **=**"SELECT name, id, photo FROM beach WHERE region = %s"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cur **=** db**.**cursor**()**

i **=** int**(**pointer**)**

json\_return **={}**

**try:**

cur**.**execute**(**queryAll**,(**region**[**i**],))**

listall **=** cur**.**fetchall**()**

items **=[]**

**for** data **in** listall**:**

j**={**'name'**:**data**[**0**],**'id'**:**data**[**1**],**'photo'**:**data**[**2**]}**

items**.**append**(**j**)**

### getNearBeaches

Η κλάση getNearBeaches, υλοποιεί το ομώνυμο webservice, περιέχει την συνάρτηση getNear(mylat, mylon). Η συνάρτηση παίρνει ως ορίσματα τις συντεταγμένες τις συσκευής του χρήστη της εφαρμογής. Η συνάρτηση μέσω SQL ερωτημάτων συλλέγει όλα τα στοιχεία του πίνακα beach για κάθε id. Στην συνέχεια ελέγχει ποιες παραλίες έχουν μικρότερη χιλιομετρική απόσταση από 15 χιλιόμετρα και τις επιστρέφει ως κοντινές με την τοποθεσία του χρήστη. Η χιλιομετρική απόσταση 2 σημείων υπολογίζεται από τις συντεταγμένες του. Η συνάρτηση haversine αναλαμβάνει την μετατροπή των αποστάσεων των συντεταγμένων σε χιλιόμετρα, μεσω υπολογισμών συνημιτόνων των συντεταγμένων.

Αρχείο Κώδικα απόσπασμα κώδικα από την κλάση getNearBeaches

results **=** cur**.**fetchall**()**

**for** item **in** results**:**

**if(**haversine**(**float**(**item**[**3**]),** float**(**item**[**2**]),** float**(**mylong**),** float**(**mylat**))<=**15**):**

names**.**append**(**item**[**0**])**

ids**.**append**(**int**(**item**[**1**]))**

lats**.**append**(**float**(**item**[**2**]))**

lons**.**append**(**float**(**item**[**3**]))**

havs**.**append**(**haversine**(**float**(**item**[**3**]),** float**(**item**[**2**]),** float**(**mylong**),** float**(**mylat**)))**

phs**.**append**(**item**[**4**])**

i **=** len**(**names**)-**1

**while** i **>=**0**:**

myjson **={**'name'**:**names**[**i**],**'id'**:**ids**[**i**],**'latitude'**:**lats**[**i**],**'longitude'**:**lons**[**i**],**'distance'**:**float**(**havs**[**i**]),**'photo'**:**phs**[**i**]}**

items**.**append**(**myjson**)**

# Upload δεδομένων και webservices στον Server

Στόχος είναι η κλήση των webservices από περισσότερους από έναν πόρους. Καλόντας τα webservices με την χρήση της ip του υπολογιστή, γίνεται τοπική χρήση των webservices με αποτέλεσμα οι αποκρίσεις τους, να είναι εμφανές μόνο στον υπολογίστή στον οποίο ανήκει το webservice και τον emulator. Για τον λόγο αυτό γίνεται χρήση του server, γίνονται upload τα κατάλλληλα αρχεία και χρήση του url του server με σκοπό την κλήση των υπρεσιών από όλους τους πόρους που φέρουν την εφαρμογή. Επειδή τα webservices είναι υλοποιημένα με την χρήση της γλώσσας προγραμματισμού Python, χρειαζόμαστε έναν server ο οποίος να είναι συμβατός στο Flask framework. Γίνεται χρήση του dreamhost, αρχικά πρέπει να γίνει setup του domain που θα χρησιμοποιηθεί για την χρήση και την κλήση των webservices. Στο πάνελ του dreamhost, επιλέγουμε **Manage Domains,** προσθέτουμε ένα καινούργιο domain, πατώντας την επιλογή **Domain/ Sub-Domain**. Συνήθως τα ονόματα που θα δίνονται στα subdomains, είναι ανγνωριστικά του ονόματος της εφαρμογής που θα καλεί τα webservice. Κάνουμε τσέκ την επιλογή remove WWW, και επειδή χρειαζόμαστε Python υπηρεσίες επιλέγουμε το box **Passenger (Ruby/ Python apps only)**. Στην συνέχεια παίρνουμε username και password για πρόσβαση στο domain που δημιουργήθηκε, με την χρήση του προγράμματος PuTTy συνδεόμαστε και εμφανίζεται ένας φάκελος με το όνομα του domain που δημιουργήσαμε. Μεταβαίνουμε σε αυτόν τον φάκελο. Αρχικά πρέπει να δημιουργηθεί ένα virtual environment. Αυτό επιτυγχάνεται με την χρήση της εντολής virtualenv. Εκτελόντας την εντολής ls πρέπει να εμφανιστούν οι φάκελοι, bin, include, lib, public



Εικόνα virtual environment

Στην συνέχεια πρέπει να γίνει η ενεργοποίηση του εικονικού περιβάλλοντος με την παρακάτω εντολή.



Εικόνα 51 activate virtual enviroment

Αποτέλεσμα της εντολής αυτής είναι το command line να μοίαζει κάπως έτσι **(myappname.domain.com)myappname.domain.com$**

Στην συνέχεια πρέπει να γίνει εγκατάσταση του Flask framework , η εγκατάσταση του framework γίνεται με την χρήση της εντολής **easy\_install flask.**

Εφόσον έχουν ολοκληρωθεί τα παραπάνω βήματα πρέπει να δημιουργηθεί ένα αρχείο με το όνομα passenger\_wsgi.py τον κωδικα του αρχείου που περιέχει την flask application των webservices και τις παρακάτω εντολές

INTERP = os.path.join(os.environ['HOME'], 'bin', 'python')

if sys.executable != INTERP:

os.execl(INTERP, INTERP, \*sys.argv)

sys.path.append(os.getcwd())

#**enter your flask app here**

Για να γίνει εκκίνηση του server είναι αναγκαία η δημιουργία ενός φακέλου tmp και στον φάκελο αυτό να δημιουργηθεί ένα αρχείο με το όνομα restart.txt. Κάθε φορά που γίνονται αλλαγές στην flask app και να είναι ορατές προς όλες πρέπει να ανανεόνεται το αρχείο restart.txt. Αυτό επιτυγχάνεται με την παρακάτω εντολή:



Εικόνα restart server

Έτσι λοιπόν τα webservices μπορούν να κληθούν από οποίαδήποτε συσκευή που φέρει την εφαρμογή, με το domain που ορίστηκε στο web panel του dreamhost και το app route που ορίστηκε από την flask application, πιο συγκεκριμένα το webservice που επιστρέφει όλα τα αντικείμενα του πίνακα beach καλείται από το url **myappname.domain.com/GetAllItems**.

# Cross-Platform Mobile Application

## Παράμετροι της Εφαρμογής

Το αρχείο AndroidManifest.xml είναι το αρχείο που περιέχει όλα τα user permissions της εφαρμογής.

Αρχείο Κώδικα AndroidManifest.xml

<?xmlversion=**"1.0"**encoding=**"utf-8"**?>

<manifestxmlns:android=**"http://schemas.android.com/apk/res/android"**package=**"app7.app7"**android:installLocation=**"auto"**android:versionCode=**"2"**android:versionName=**"2.0"**>

<uses-sdk/>

<uses-permissionandroid:name=**"android.permission.ACCESS\_WIFI\_STATE"**/>

<uses-permissionandroid:name=**"android.permission.ACCESS\_NETWORK\_STATE"**/>

<uses-permissionandroid:name=**"android.permission.INTERNET"**/>

<uses-permissionandroid:name=**"android.permission.ACCESS\_COARSE\_LOCATION"**/>

<uses-permissionandroid:name=**"android.permission.ACCESS\_FINE\_LOCATION"**/>

<uses-permissionandroid:name=**"android.permission.ACCESS\_MOCK\_LOCATION"**/>

<uses-permissionandroid:name=**"android.permission.WRITE\_EXTERNAL\_STORAGE"**/>

<uses-permissionandroid:name=**"android.permission.GET\_ACCOUNTS"**/>

<uses-permissionandroid:name=**"android.permission.USE\_CREDENTIALS"**/>

<applicationandroid:label=**"App7.Droid"**android:icon=**"@drawable/Icon"**>

<meta-dataandroid:name=**"com.google.android.maps.v2.API\_KEY"**android:value=**"AIzaSyAdhD1tzxhOm3yV-j9XLP8eqZG6\_lOdiio"**/>

<meta-dataandroid:name=**"com.google.android.gms.version"**android:value=**"@integer/google\_play\_services\_version"**/>

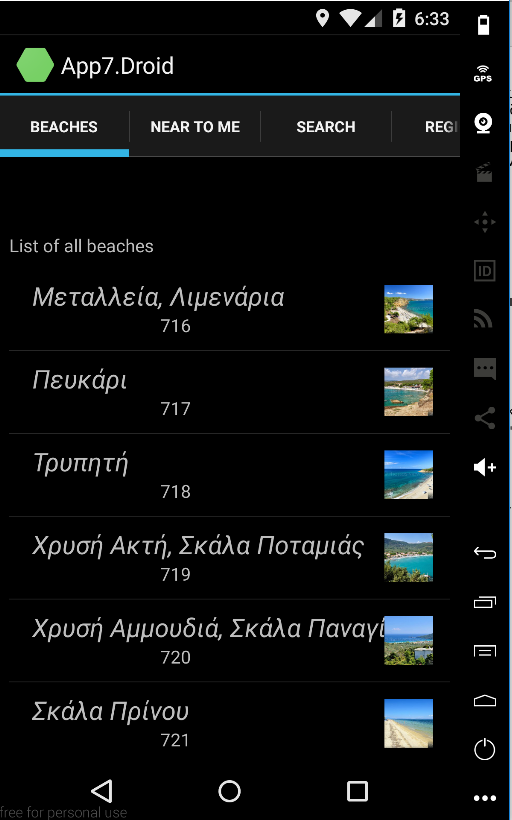
</application>

</manifest>

Παρατηρούμε ότι το παραπάνω αρχείο αναφέρει ιδιότητες που έχει η εφαρμογή έπειτα από έγκριση του χρήστη. Μερικές από αυτές είναι η πρόσβαση στην κατάσταση του WIFI, Internet, να έχει πρόσβαση στις τοποθεσίες που βρίσκεται ο χρήστης καθώς και τις συντεταγμένες της συσκευής, και την εγγραφή δεδομένων σε εξωτερικές μνήμες. Επίσης σε αυτό το αρχείο τοποθετούμε τα API keysγια την χρήση google maps στην εφαρμογή. Τα permissions Που οριοθετούνται είναι πάρα πολύ σημαντικά καθώς χωρίς αυτά δεν μπορέι η εφαρμογή να χρησιμοποιήσει τους αντίστοιχους πόρους της εφαρμογής. Πιο συγκεκριμένα η εφαρμογή για να χρησιμοποιήσει τις συντεταγμένες του χρήστη πρέπει να έχιε πρόσβαση στο gps του κινητού, δηλώνοντας τα αντίστοιχα permissions, για να μπορεί να συνδεθεί με το google maps api να μπορεί να συνδεθεί σε wifi αλλά και να χρησιμοποιεί internet, για να γίνεται χρήση από την εφαρμογή χαρτών της google πρέπει να δηλωθεί το αντίστοιχο κλειδί της υπηρεσίας καθώς και η έκδοση των google play services.

## Αρχική Οθόνη της εφαρμογής

Όταν ο χρήστης ανοίγει την εφαρμογή μεταφέρεται στην αρχική της οθόνη



Εικόνα Αρχική οθόνη της εφαρμογής

Παρατηρούμε ότι η αρχική οθόνη αποτελείται από 4 καρτέλες για την υλοποίηση των καρτελών χρησιμοποιήθηκαν ActionBars tags. Πιο συγκεκριμένα χρησιμοποιείται ένα fragment και κάθε φορά που επιλέγεται ένα διαφορετικό tag ενεργοποιείται και μια διαφορετική κλάση. Η συνάρτηση AddTab προσθετει tags, κάθε tag πρέπει όμως να συνδέεται με μια κλάση τύπου fragment. Κάθε tag υλοποιεί διαφορετικές λειτουργίες για αυτό όσα tags έχουμε, έχουμε και τόσες κλάσεις. Το default tag που εμφανίζεται είναι το tag **BEACHES**, περιέχει την λίστα με τις παραλίες της χώρας που έχουμε συγκεντρώσει από τους ιστότοπους. Η λίστα αυτή είναι διαδραστική, δηλαδή αν πατήσουμε σε ένα αντικείμενο της λίστας μπορούμε να βρεθούμε στην οθόνη της παραλίας.

Αρχείο Κώδικα AddTab συνάρτηση

void AddTab**(**string tabText**,** Fragment fragment**)**

**{**

var tab **=this.**ActionBar**.**NewTab**();**

tab**.**SetText**(**tabText**);**

// must set event handler for replacing tabs tab

tab**.**TabSelected **+=delegate(object** sender**,** ActionBar**.**TabEventArgs e**){**

e**.**FragmentTransaction**.**Replace**(**Resource**.**Id**.**frameLayout1**,** fragment**);**

**};**

**this.**ActionBar**.**AddTab**(**tab**);**

**}**

Παρατηρούμε ότι η λίστα των παραλιών προήρθε από την κλήση του webservice GetAllItems, είναι το webservice που επιστρέφει όλα τα δεδομένα του πίνακα beach. Για την κλήση ενός webservice χρησιμοποιείται η βιβλιοθήκη resthsharp. Η βιβλιοθήκη restsharp είναι αρκετά ευέλικτη καθώς κάνοντας χρήση μιας κλάσης μοντέλου, δηλαδή της κλάσης που αναφέρει τις ιδιότητες των JSON αντικειμένων που λαμβάνουμε από το webservice, κάνει αυτόματα deserialization του JSON object. Συνεπώς η συνάρτηση readitems αναλαμβάνει την κλήση του GetAllItems webservice και μέσω χρήσης της βιβλιοθήκης restsharp την διάσπαση των properties από το JSON object. Όπως αναφέραμε και σε προηγούμενα κεφάλαια, πρόκειται για μια cross-platform εφαρμογή οπότε πρέπει το Data Access κομμάτι να είναι κοινό για κάθε τεχνολογία, Android, Windows Phone, IOS, συνεπώς πρέπει η συνάρτηση readItems όπως και όλες οι υπόλοιπες κλάσεις που ανήκουν στο data access κομμάτι να προστεθούν σε μια Public κλάση έτσι ώστε να είναι ορατές από όλα τα πρότζεκτ, καθώς για κάθε τεχνολογία στο solution υλοποιείται και διαφορετικό πρότζεκτ. Αυτό γίνεται γιατί μπορεί το data access κομμάτι να είναι κοινό, το user interface κομμάτι της εφαρμογής αλλάζει για κάθε τεχνολογία. Συνοψίζοντας πατώντας το tag BEACHES καλείται η κλάση ALLItems η οποία με την σειρά της καλεί την συνάρτηση readItems η οποία επιστρέφει τα δεδομένα του πίνακα beach, τα οποία με την σειρά τους τροφοδοτούν τον CustomAdapter που είναι υπεύθυνος για την τοποθέτηση των δεδομένων σε ένα listview. Αυτό το listview περιέχει την φωτό, το όνομα της παραλίας και το id της παραλίας που έχει στον πίνακα beach.

Αρχείο Κώδικα κλάση AllItems

**public** class AllItems **:** ListFragment

**{**

**publicoverride**void OnCreate**(**Bundle savedInstanceState**)**

**{**

**base.**OnCreate**(**savedInstanceState**);**

// Create your fragment here

**}**

**publicoverride** View OnCreateView**(**LayoutInflater inflater**,** ViewGroup container**,** Bundle savedInstanceState**)**

**{**

**base.**OnCreateView**(**inflater**,** container**,** savedInstanceState**);**

var view **=** inflater**.**Inflate**(**Resource**.**Layout**.**AllLayout**,** container**,false);**

var sampleTextView **=** view**.**FindViewById**<**TextView**>(**Resource**.**Id**.**textView1**);**

checkMode**.**mode **=**0**;**

var myActivity **=(**MainActivity**)this.**Activity**;**

myActivity**.**clearTextLoc**();**

sampleTextView**.**Text **=**"List of all beaches"**;**

ListAdapter **=null;**

MyClass myclass **=new** MyClass**();**

myclass**.**readItems**();**

myActivity**.**RunOnUiThread**(()=>**

**{**

myActivity**.**ListAdapter **=new** myCustomAdapter**(**myActivity**,** FromAllItemClass**.**allItems**);**

**});**

myActivity**.**ListView**.**ItemClick**+=** myActivity**.**ListView\_ItemClick**;**

**return** view**;**

**}**

**}**

Το webservice GetAllItems επιστρέφει το παραπάνω JSON object, του οποίο την διεργασία deserialization αναλαμβάνει η restsharp βιβλιοθήκη με σκοπό την πιο εύκολη χρήση των δεδομένων από τον χρήστη. Στην συνέχεια υπέυθυνος για την εμφάνιση των δεδομένων είναι ο CustomAdapter.

Αρχείο Κώδικα GetAllItems webservice response

**{**

"Items"**:[**

**{**

"id"**:**716**,**

"name"**:**"Μεταλλεία, Λιμενάρια"**,**

"photo"**:**"http://www.thassos-view.com/cache/preview/d0a4648c7840523445af51647a8ec279.jpg"

**},**

**{**

"id"**:**717**,**

"name"**:**"Πευκάρι"**,**

"photo"**:**"http://www.thassos-view.com/cache/preview/53dbcce80e8a4f5c72c7cdc8f53b705b.jpg"

**},**

**{**

"id"**:**718**,**

"name"**:**"Τρυπητή"**,**

"photo"**:**"http://www.thassos-view.com/cache/preview/f177d864918ee89e056db9604fa5a27d.jpg"

**},**

**{**

"id"**:**719**,**

"name"**:**"ΧρυσήΑκτή, ΣκάλαΠοταμιάς"**,**

"photo"**:**"http://www.thassos-view.com/cache/preview/d87fb41e71d1024f436be235c9e57e99.jpg"

**},**

**{**

"id"**:**720**,**

"name"**:**"Χρυσή Αμμουδιά, Σκάλα Παναγίας"**,**

"photo"**:**"http://www.thassos-view.com/cache/preview/6138bee1017f912fc661d84502a5e164.jpg"

**},**

**{**

"id"**:**721**,**

"name"**:**"ΣκάλαΠρίνου"**,**

"photo"**:**"http://www.thassos-view.com/cache/preview/b8c9168cc8a45a1e0ab6a3311a22a301.jpg"

**},**

**{**

"id"**:**722**,**

"name"**:**"ΣκάλαΜαριών"**,**

"photo"**:**"http://www.thassos-view.com/cache/preview/b1c80abab2314ac21c6817a08afe6667.jpg"

**},**

**{**

"id"**:**723**,**

"name"**:**"Γλυφάδα"**,**

"photo"**:**"http://www.thassos-view.com/cache/preview/b4e6ed1ff8a77cf33af0ed9b59da15ea.jpg"

**},**

**{**

"id"**:**724**,**

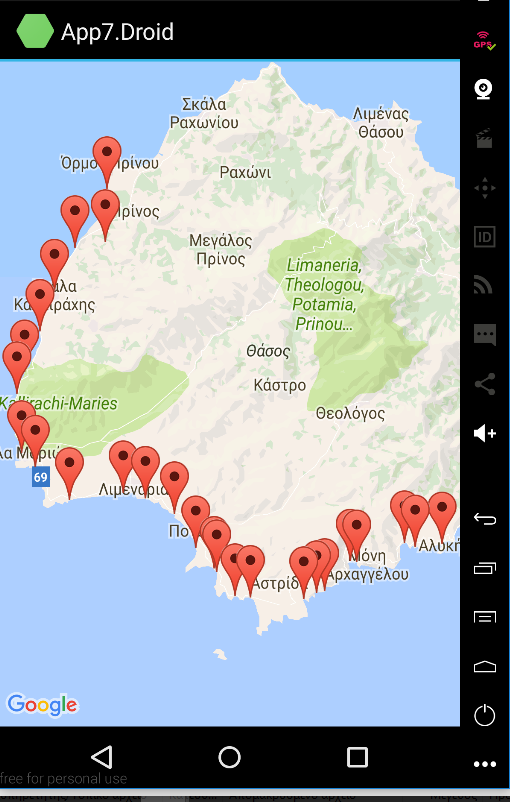
"name"**:**"Νιστέρι"**,**

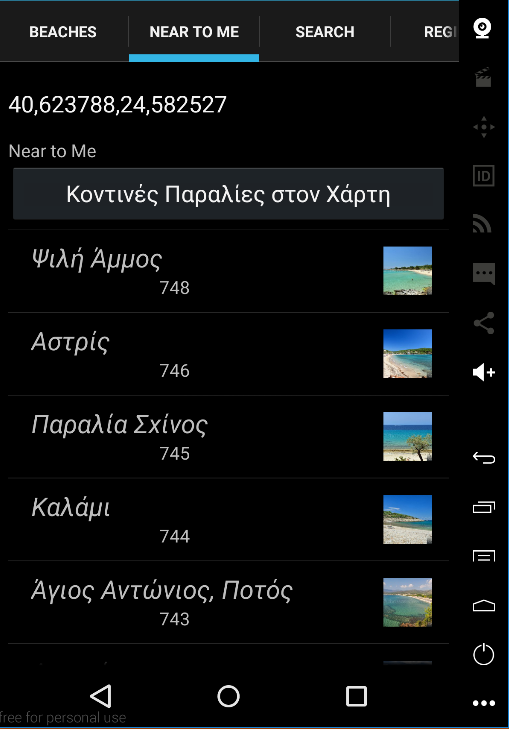
"photo"**:**"http://www.thassos-view.com/cache/preview/755a6f7f989d8d0dca84de4ac5c4a730.jpg"

**},**

## NEAR TO ME

Πατώντας το δεύτερο tag **NEARTO ME** οδηγούμαστε σε μια άλλη οθόνη, η οποία είναι η παρακάτω.





Εικόνα 54 NEAR TO ME tag

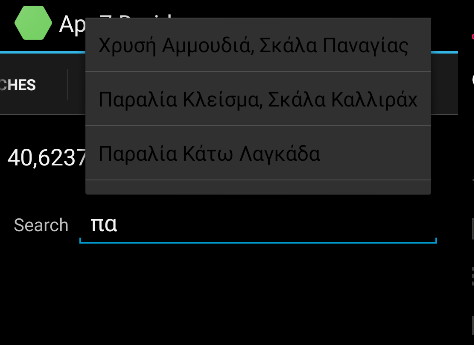
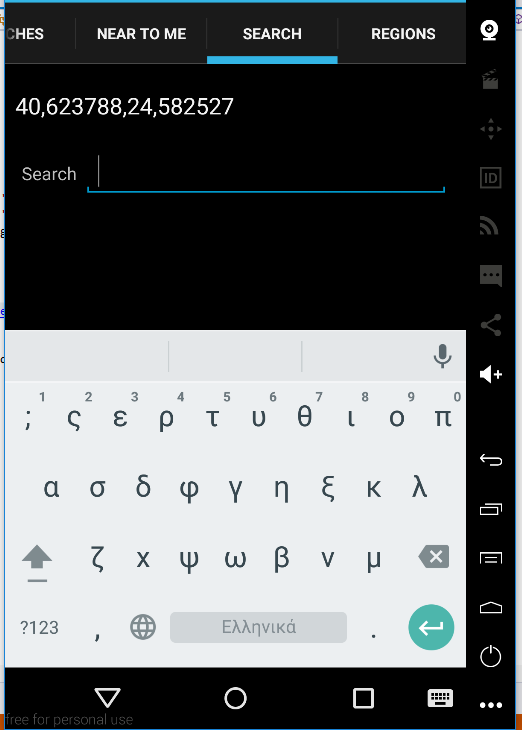
Σε αυτήν την οθόνη πατώντας το κουμπί εμφανίζεται ένας χάρτης που έχει tags όλες τι κοντινες παραλίες στον χρήστη. Ενεργοποιούμε το gps για τον υπολογισμό της χιλιομετρικής απόστασης μεταξύ της τοποθεσίας του χρήστη και των παραλιών που υπάρχουν στην βάση δεδομένων. Πατώντας λοιπόν το κουμπί «Κοντινές Παραλίες στον Χάρτη» έχουμε ως εξής:

Πιο συγκεκριμένα όταν πατάμε το tag NEAR TO ME καλείται η κλάση NearItems είναι μια κλάση η οποία κληρονομεί από την κλάση ListFragment και καλεί την συνάρτηση getNearBeaches. Η συνάρτηση αυτή καλεί το webservice getNearBeaches, συλλέγει με αυτόν τον τρόπο όλα τα δεδομένα του πίνακα beach. Ο χρήστης πρέπει να έχει ενεργοποιήσει το gps στην εφαρμογή του έτσι ώστε το webservice getNearBeaches να μπορέσει να υπολογίσει την χιλιομετρική απόσταση της συσκευής από τις παραλίες. Επιστρέφονται οι παραλίες με χιλιομετρική απόσταση από τον χρήστη των 15 χιλιομέτρων.

Η συνάρτηση resthsharp παράγει μια λίστα η οποία τροφοδοτείΟυσιαστικά με την κλήση της συνάρτησης getNearItems αυτό που μας επιστρέφει ο server είναι μια απόκριση την οποία ο server την παίρνει και με την σειρά της restsharp και διασπά το JSON object στις πληροφορίες που θέλουμε. Στην συνέχεια με τα δεδομένα αυτά τροφοδοτείται ο CustomAdapter NearAdapter με σκοπό την ένταξη των δεδομένων στο Listview της κλάσης ListFragment. Παρατηρούμε ότι από την απόκριση παίρνουμε για κάθε εγγραφή αρκετά δεδομένα όπως την απόσταση, το id των κοντινών παραλιών , το γεωγραφικό μήκος και πλάτος, το όνομα της παραλίας και μια φωτογραφία κάθε παραλίας.

## SEARCH

Πατώντας το tag **SEARCH** εμφανίζεται η παρακάτω εικόνα στην οθόνη της συσκευής του χρήστη:



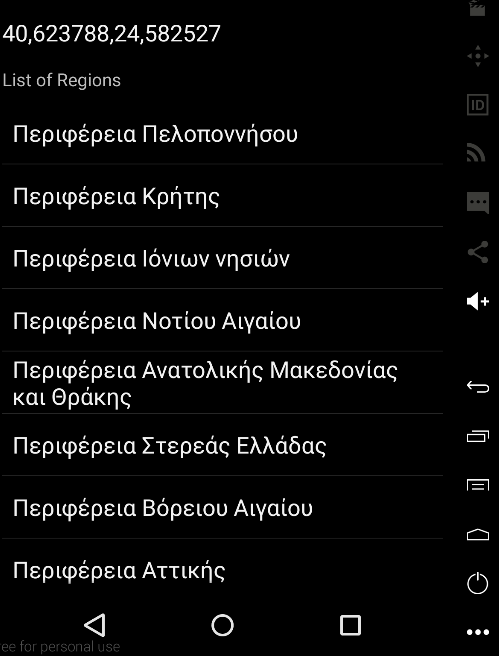
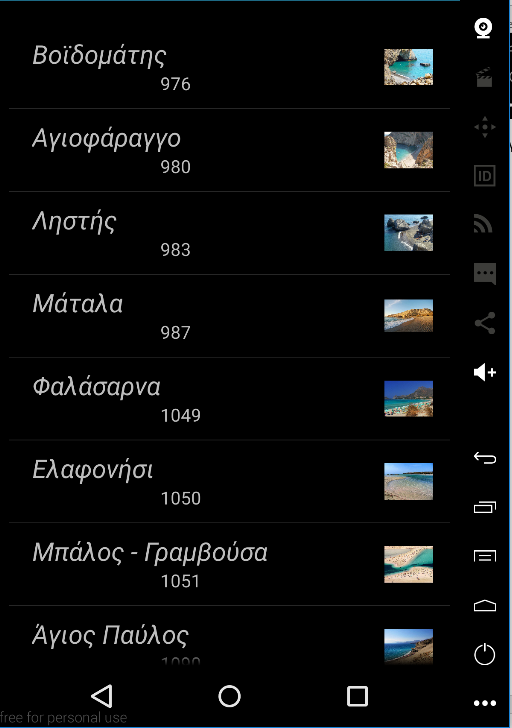
Εικόνα SEARCH (a)

Πληκτρολογόντας το όνομα μιας παραλίας παρουσιάζεται ενα γκρι φόντο που έχει τα αποτελέσματα της αναζήτησης για παράδειγμα

Πιο συγκεκριμένα, επιλέγοντας το tag SEARCH ενεργοποιείται η κλάση SEARCHME, η οποία κλάση κληρονομεί από την κλάση fragment. Χρησιμοποιούμε ένα autocomplete textview για την δημιουργία του autocomplete search. Από το textview αυτό πληκτρολογόντας οποιοδήποτε αλφαριθμητικά, το textview προσπαθεί να τα συμπληρώσει με την βοήθεια ενός ArrayAdapter που τροφοδοτείται από τα ονόματα που λάβαμε από το GetAllItems webservice. Με την υλοποίηση της συνάρτησης TextView\_ItemClick λίστα συμπλήρωσης που παράγεται μπορεί να γίνει interactive καθώς ο χρήστης πατώντας πάνω σε μια από τις προτιμόμενες παραλίες μπορούμε να μεταφερθούμε σε οθόνη με πιο λεπτομερή χαρακτηριστικά για αυτήν.

## REGIONS

Πατώντας το tag **REGIONS** οδηγούμαστε σε μια λίστα με τις περιφέρειες της Ελλάδας, πατώντας για παράδειγμα στην Περιφέρεια Κρήτης οδηγούμαστε σε ένα καινούργιο Litview, όπου έχουμε της παραλίες που βρίσκονται στην περιφέρεια Κρήτης.

Εικόνα REGIONS

Πατώντας στο tag REGIONS ενεργοποιείται η κλάση regionItems, η οποία κλάση καλεί την συνάρτηση getRegionBeaches , πρόκειται για μια dataaccess συνάρτηση η οποία καλεί το webservice **getRegionBeaches**. Ανάλογα με ποια περιφέρεια επιλεχθεί στέλνεται προς το webserviceένας ακέραιος, ο οποίος ακέραιος είναι η θέση του ονόματος της περιφέρειας σε μια λίστα με όνομα regions. To webservice αναλαμβάνει να προσκομίσει τις παραλίες που βρίσκονται σε μια περιφέρεια. Πιο συγκεκριμένα το αποτέλεσμα του webserviceείναι σε μορφή JSONκαι φαίνεται παρακάτω. Πατώντας πάνω σε ένα αντικέιμενο της λίστας μεταφερόμαστε στην οθόνη της παραλίας στην οποία πατήσαμε. Παρατηρούμε ότι τα δεδομλενα που παίρνουμε από το server είναι το id της παραλίας , το όνομα και μια φωτό. Αυτά τα δεδομένα τα δέχεται ο RegionAdapter έτσι ώστε να τα τοποθετήσει σε ένα listview. Κάθε παραλία στο listview παρουσιάζεται με το όνομα της, το id και μια φωτογραφία.

**{**

"Items"**:[**

**{**

"id"**:**976**,**

"name"**:**"Βοϊδομάτης"**,**

"photo"**:**"http://www.wondergreece.gr/public/images/2013/03/24/heraklion\_voidomatis/20130324173530heraklion\_voidomatis1.jpg"

**},**

**{**

"id"**:**980**,**

"name"**:**"Αγιοφάραγγο"**,**

"photo"**:**"http://www.wondergreece.gr/public/images/2013/03/24/heraklion\_b\_agiofaraggo/20130324173512heraklion\_b\_agiofaraggo5.jpg"

**},**

**{**

"id"**:**983**,**

"name"**:**"Ληστής"**,**

"photo"**:**"http://www.wondergreece.gr/public/images/2013/03/24/heraklion\_b\_listis/20130324173516heraklion\_b\_listis2.jpg"

**},**

**{**

"id"**:**987**,**

"name"**:**"Μάταλα"**,**

"photo"**:**"http://www.wondergreece.gr/public/images/2013/03/24/heraklion\_b\_matala/20130324173520heraklion\_b\_matala1.jpg"

**},**

**{**

"id"**:**1049**,**

"name"**:**"Φαλάσαρνα"**,**

"photo"**:**"http://www.wondergreece.gr/public/images/2013/04/01/falasarna/20130401184551falasarna2.jpg"

**},**

**{**

"id"**:**1050**,**

"name"**:**"Ελαφονήσι"**,**

"photo"**:**"http://www.wondergreece.gr/public/images/2013/04/01/elafonisi/20130401184550elafonisi7.jpg"

**},**

**{**

"id"**:**1051**,**

"name"**:**"Μπάλος - Γραμβούσα"**,**

"photo"**:**"http://www.wondergreece.gr/public/images/2013/04/01/balos.gramvousa/20130401184549balos.gramvousa5.jpg"

**},**

**{**

"id"**:**1090**,**

"name"**:**"ΆγιοςΠαύλος "**,**

"photo"**:**"http://www.wondergreece.gr/public/images/2013/04/01/agiospavlos/20130401184549agiospavlos2.jpg"

**},**

Αρχείο Κώδικα getRegionBeaches webservice response

Αρχείο Κώδικα συνάρτηση getRegionBeaches

**Public** void getRegionBeaches**()**

**{**

SQLiteConnection db **=new** SQLiteConnection**(**dbreg**);**

db**.**CreateTable**<**region**>();**

var table **=** db**.**Table**<**region**>();**

var rrr **=new** getRegionItems**();**

var regList **=** rrr**.**getRegionFSQL**(**dbreg**,** fromRegionFrag**.**regionPointer**);**

**if(**regList**.**Count **==**0**)**

**{**

var url **=**"http://192.168.1.2:5000/getRegionBeaches/"**+** fromRegionFrag**.**regionPointer**.**ToString**();**

var client **=new** RestClient**(**url**);**

var request **=new** RestRequest**(**Method**.**GET**);**

var response4 **=** client**.**Execute**<**tRootObject**>(**request**);**

rrr**.**fillTableregion**(**dbreg**,** response4**.**Data**.**Items**,** table**,** fromRegionFrag**.**regionPointer**);**

regionCall**(**response4**.**Data**.**Items**);**

**}**

**else{**

var result **=new** List**<**tItem**>();**

**foreach(**var s **in** regList**)**

**{**

var data **=new** tItem

**{**

id **=** s**.**beach\_id**,**

name **=** s**.**name**,**

photo **=** s**.**photo

**};**

result**.**Add**(**data**);**

**}**

regionCall**(**result**);**

**}**

**}**

**public**void regionCall**(**List**<**tItem**>** d**)**

**{**

FromAllItemClass**.**regionItems **=** d**;**

var beachArray **=** d**.**Select**(**item **=>** item**.**name**).**ToArray**();**

RegionActivity**.**regAr **=** beachArray**;**

var beachIdsArray **=** d**.**Select**(**item **=>** item**.**id**).**ToArray**();**

RegionActivity**.**regId **=** beachIdsArray**;**

**}**

# Δεύτερη Οθόνη της Εφαρμογής

Πατώντας πάνω στο όνομα μιας παραλίας μεταβαίνουμε στην δεύτερη οθόνη της εφαρμογής. Πρόκειται για μια οθόνη που αποτελείται από tabs. Τα tabs αυτά δημιουργήθηκαν με την χρήση actionbars, κάθε tab συνδέεται με μια fragment κλάση και κάθε φορά που πατάμε ένα tag ενεργοποιείται η αντίστοιχη κλάση. Πιο συγκεκριμένα μια σύνοψη της δεύτερης οθόνης της εφαρμογής είναι η παρακάτω εικόνα.



Εικόνα Δεύτερη οθόνη της εφαρμογής

Παρατηρούμε ότι τα tags είναι τα εξής PHOTOS, DESCRIPTION, COMMENTS, MAP. Καθένα ξεχωριστό tag συσχετίζεται με μια κλάση τύπου fragment. Στις παρακάτω υποενότητες αναλύονται τα tags της δεύτερης οθόνης της εφαρμογής. Το πρώτο tag Που εμφανίζεται default είναι το tag PHOTOS. Πρόκειται για την συνάρτηση

## PHOTOS

Πατώντας το tag PHOTOS ενεργοποιείται η κλάση showPhotos, η κλάση αυτή κληρονομεί από την κλάση fragment. Ο κώδικας της παρατίθεται παρακάτω:

**public** class showPhotos **:** Fragment

**{**

**public** static MultiImageView image**;**

**public** **override** void OnCreate**(**Bundle savedInstanceState**)**

**{**

**base.**OnCreate**(**savedInstanceState**);**

// Create your fragment here

**}**

**public** **override** View OnCreateView**(**LayoutInflater inflater**,** ViewGroup container**,** Bundle savedInstanceState**)**

**{**

var view **=** inflater**.**Inflate**(**Resource**.**Layout**.**ph**,** container**,** **false);**

image **=** view**.**FindViewById**<**MultiImageView**>(**Resource**.**Id**.**imageView1**);**

var myActivity **=** **(**ItemContext**)this.**Activity**;**

MyClass mm **=** **new** MyClass**();**

var myli **=** mm**.**getPhotos**(**SharedObjects**.**mainId**);**

photocall**(**myli**);**

**return** view**;**

**}**

**public** void photocall**(**List**<**photoItem**>** li**)** **{**

var photoarray **=** li**.**Select**(**item **=>** item**.**photo**).**ToArray**();**

image**.**LoadImageList**(**photoarray**);**

image**.**SliderSelectedIcon **=** BitmapFactory**.**DecodeResource**(**Resources**,** Resource**.**Drawable**.**slider\_blt\_grn**);**

image**.**SliderUnselectedIcon **=** BitmapFactory**.**DecodeResource**(**Resources**,** Resource**.**Drawable**.**slider\_blt\_trans**);**

image**.**SetSliderIconDimensions**(**50**,** 50**);**

image**.**DownloadedImageSampleSize **=** 1**;**

var myActivity **=** **(**ItemContext**)this.**Activity**;**

image**.**ImagesLoaded **+=** **(**sender**,** e**)** **=>**

**{** // Loads the first image in the list

myActivity**.**RunOnUiThread**(**image**.**LoadImage**);**

**};**

**}**

**}**

Αρχείο Κώδικα κλάση showPhotos

Η κλάση showPhotos είναι η κλάση η οποία καλεί την συνάρτηση getPhotos. H συνάρτηση αυτή με την σειρά της καλεί το webservice getPhotos, το οποίο επιστρέφει τα URLs των φωτογραφιών που σχετίζονται με το id που παίρνει ως όρισμα η συνάρτηση getPhotos. Στην συνέχεια η συνάρτηση photocall της κλάσης παίρνει την λίστα των URLs των φωτογραφιών και με την βοήθεια της συνάρτησης MultiImageView γίνεται εμφάνιση των φωτογραφιών στην οθόνη του κινητού. Το webservice getPhotos συνδέεται με την βάση δεδομένων και επιστρέφει τις εγγραφές του πίνακα photos που σχετίζονται με το id από το οποίο καλείται το webservice. Η απόκριση του webservice είναι σε JSON format. Η βιβλιοθήκη restsharp χρησιμοποιείται από την συνάρτηση getPhotos έτσι ώστε να καλέσει το ομώνυμο webservice να λάβει την απόκριση του και να κάνει deserialization το JSON αντικείμενο, με σκοπό την χρήση των δεδομένων με πιο εύκολο τρόπο από την εφαρμογή.

**{**

"Items"**:** **[**

**{**

"id"**:** 717**,**

"photo"**:** "http://www.thassos-view.com/cache/preview/53dbcce80e8a4f5c72c7cdc8f53b705b.jpg"

**},**

**{**

"id"**:** 717**,**

"photo"**:** "http://www.thassos-view.com/cache/preview/6c5796c28cb481abf52c45d97a9c998b.jpg"

**},**

**{**

"id"**:** 717**,**

"photo"**:** "http://www.thassos-view.com/cache/preview/57723093b639205279bb5fb791032e7e.jpg"

**},**

**{**

"id"**:** 717**,**

"photo"**:** "http://www.thassos-view.com/cache/preview/a446c30ecf91f9c637a670ecc5fd5640.jpg"

**},**

**{**

"id"**:** 717**,**

"photo"**:** "http://www.thassos-view.com/cache/preview/6d8eb7fa75b81eb27f87d66ffd82dc41.jpg"

**},**

**{**

"id"**:** 717**,**

"photo"**:** "http://www.thassos-view.com/cache/preview/747094fa49ec0b491641ebe7edfab7a1.jpg"

**},**

**{**

"id"**:** 717**,**

"photo"**:** "http://www.thassos-view.com/cache/preview/065dc8f71b8ff5eee41a061ef6e16e1c.jpg"

**},**

**{**

"id"**:** 717**,**

"photo"**:** "http://www.thassos-view.com/cache/preview/723bd91ec794fc1c8e0a214255a1b331.jpg"

**},**

**{**

"id"**:** 717**,**

"photo"**:** "https://irs2.4sqi.net/img/general/width500/36325345\_nub83BrB21EmC1\_4BzEpQE79IK-SjEf4rOU9o9uWe78.jpg"

**},**

**{**

"id"**:** 717**,**

"photo"**:** "https://irs2.4sqi.net/img/general/width500/10467185\_1eZcoysUGg2shKnrvdRHWQSobNPHDkQqN7ZdSjnyXzo.jpg"

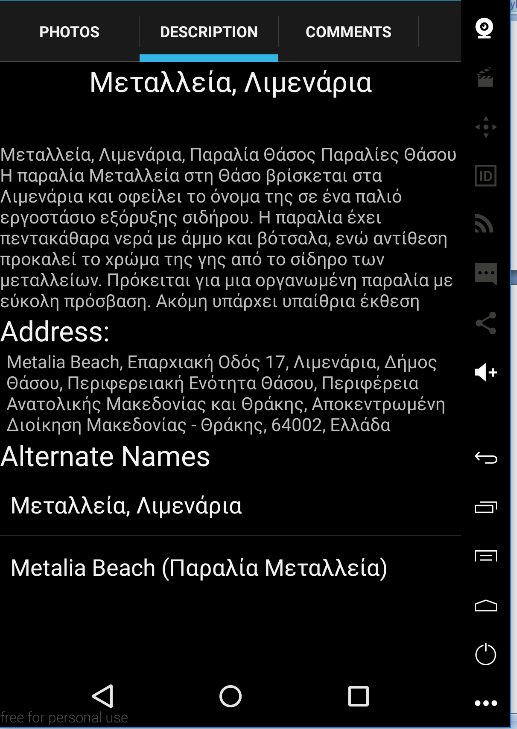
**},**

**Αρχείο Κώδικα 35 getPhotos webservice response**

Παρατηρούμε ότι το JSON αντικείμενο έχει ως ιδιότητες το id της παραλίας για την οποία ζητούνται φωτογραφίες, και το url των φωτογραφιών. Από εκεί και έπειτα η συνάρτηση photocall μέσω της βιβλιοθήκης MultiImageView αναλαμβάνει την εμφάνιση των φωτογραφιών μέσω των URLs τους. Πιο συγκεκριμένα πρόκειται για την βιβλιοθήκη Restsharp, η οποία προσκομίζει το JSON αντικείμενο παίρνει τα URLs των φωτογραφιών και στην συνέχεια τα επεξεργάζεται η βιβλιοθήκη MultiImageView, έχοντας μόνο τα URLs δεν μπορούμε να τα προβάλουμε αυτό που χρειάζεται είναι η ψηφιακή απεικόνιση των φωτογραφιών, έτσι με την χρήση της συνάρτησης LoadImageList δίδεται ως όρισμα η λίστα με τα URLs και κατεβαίνουν οι φωτογραφίες.

## DESCRIPTION

Πατώντας το tag DESCRIPTION μεταβαίνουμε στην παρακάτω οθόνη

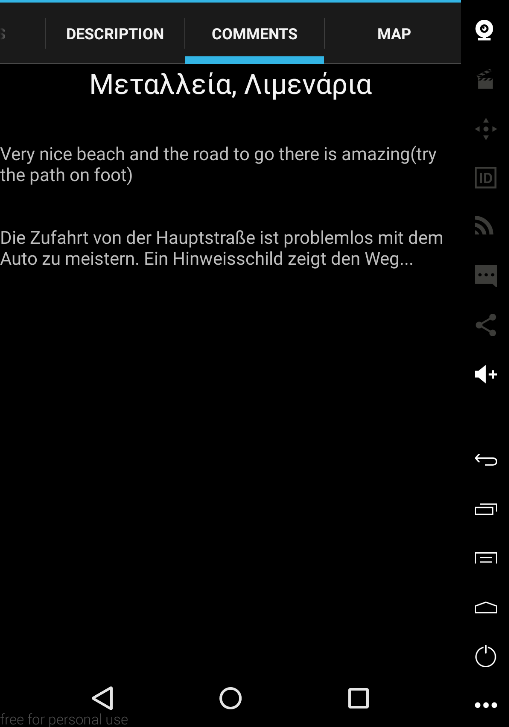


Εικόνα DESCRIPTION tag

Παρατηρούμε ότι μεταφερόμαστε σε μία οθόνη όπου αποτελείται από την περιγραφή της παραλίας, περιέχει την διεύθυνση της παραλίας καθώς και διαφορετικά ονόματα της παραλίας. Όταν πατάμε το tag αυτό ενεργοποιείται η κλάση τύπου fragment beachDesc. Η κλάση αυτή καλεί τις data access συναρτήσεις readDesc, showAddress, getAlter, οι οποίες συναρτήσεις με την σειρά τους καλούν τα webservices getDetails, getAddress, getAlterNames. Τα webservice επιστρέφουν JSON αντικείμενα τα οποία με χρήση της restsharp βιβλιοθήκης υπόκεινται σε deserialization Με σκοπό την πιο εύκολη διαχείρηση των δεδομένων από την εφαρμογή. Πάλι η λογική είναι ίδια, τα webservices είναι υπεύθυνα για την προσκόμιση των δεδομένων από την βάση γνώσης στην εφαρμογή, έτσι ώστε να προβληθούν στο UI κομμάτι της. Η διαχείριση των αποκρίσεων των webservices γίνεται από την βιβλιοθήκη restsharp. Οι περιγραφές προβάλλονται από ένα textView, όπως και η διεύθυνση της παραλίας. Και τα ονόματα της παραλίας από ένα listview. Τα JSON objects που παράγoνται περιέχουν τις περιγραφές για το μέρος επιστρέφουν τις περιγραφές και το σχόλια που πάρθηκαν από διάφορους ιστοτόπους, την πήγη κάθε κειμένου, καθώς και το id της παραλίας για την οποία ζητήθηκαν οι περιγραφές. Επειδή το tag αφορά περιγραφή της παραλίας και όχι τα σχόλια των χρηστών, υπάρχει ξεχωριστό tag, αποκλείουμε τα κείμενα με πηγή το Foursquare.

## COMMENTS

Αυτό το tag όταν πατηθεί, εμφανίζεται στην οθόνη του κινητού το εξής:



Εικόνα COMMENTS tag

Με το πάτημα του tag COMMENTS ενεργοποιείται η κλάση beachComments, η οποία κληρονομεί από την κλάση Fragment. Η κλάση αυτή καλεί την συνάρτηση data access readComments, η οποία με την σειρά της καλεί το getDetails webservice, το οποίο έχει ως έξοδο ένα JSON object με κείμενα (είτε σχόλια χρηστών είτε περιγραφές για την παραλία). Περιέχει και την πηγή κάθε κειμένου. Η πηγή σχολίων του tag COMMENTS είναι το Foursquare, συνεπως η συνάρτηση readComments που καλεί το webservice αποκλείει όλα τα κείμενα με πηγές διαφορετικές του Foursquare. H βιβλιοθήκη restsharp είναι υπεύθυνη για το deserialization του JSON object με σκοπό την ευκολότερη επεξεργασία του αντικειμένου από τις data access συναρτήσεις.

Αρχείο Κώδικα getDetails webservice response

**{**

"Items"**:** **[**

**{**

"description"**:** "Πευκάρι, Παραλία Θάσος Παραλίες Θάσου Το Πευκάρι βρίσκεται στη νότια Θάσο, μόλις 1 χλμ από τον Ποτό. Η απόστασή του από τον Λιμένα είναι 39χλμ. Πρόκειται για μία οργανωμένη παραλία, με άμμο και βότσαλα, η οποία έχει βραβευτεί με μπλε σημαία. Η παραλία πήρε το όνομα της από το πευκοδάσος της περιοχής. Το Πευκάρι είναι δημοφιλές για τα θαλάσσια σπορ του ενώ κατά μήκος της παραλίας θα βρείτε πολλές ταβέρνες και εστιατόρια. "**,**

"id"**:** 717**,**

"source"**:** "thassos-view.com"

**},**

**{**

"description"**:** "Great beach and polite stuff but didn't like so much stones in the water."**,**

"id"**:** 717**,**

"source"**:** "Foursquare"

**},**

**{**

"description"**:** "It's nowhere near as beautiful as other beaches. The pictures in the Internet are deceptive :D . It's fraught with stones and sea-urchins, so bring special shoes with you."**,**

"id"**:** 717**,**

"source"**:** "Foursquare"

**},**

**{**

"description"**:** "It was ince a great place ... However it's becoming more and more populated. The food has suffeed the tastes of the mass. We will give it another chance."**,**

"id"**:** 717**,**

"source"**:** "Foursquare"

**},**

**{**

"description"**:** "Nice, quiet and clean. Sand doesn't stick on you ☺️ You better go deeper to see the water turn blue-green"**,**

"id"**:** 717**,**

"source"**:** "Foursquare"

**},**

**{**

"description"**:** "Sığ deniz su sporları müthiş"**,**

"id"**:** 717**,**

"source"**:** "Foursquare"

**},**

**{**

"description"**:** "Burada 2 saat geçirilir,guzel bir koy."**,**

"id"**:** 717**,**

"source"**:** "Foursquare"

**},**

**{**

"description"**:** "Excellent beach, delicious food"**,**

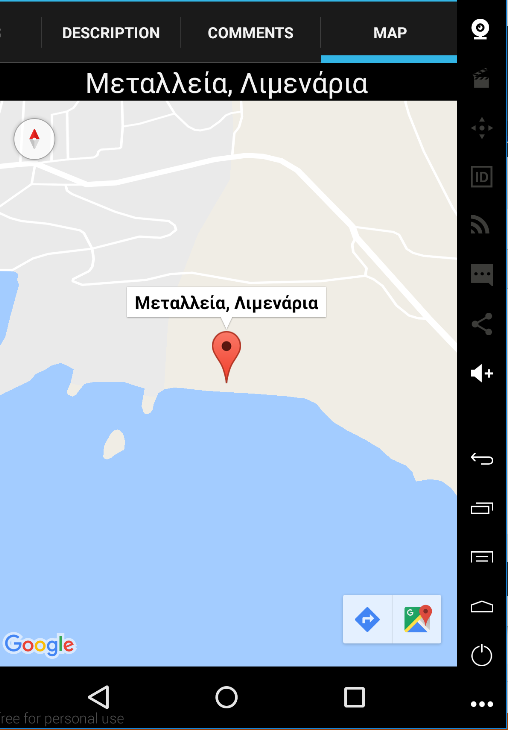
"id"**:** 717**,**

"source"**:** "Foursquare"

**},**

## MAP

Πατώντας το tag MAP εμφανίζεται στην οθόνη του κινητού η παρακάτω εικόνα:



Εικόνα MAP tag

Ενεργοποιείται η κλάση myMAP, η οποία καλεί το webservice getCoordinates, το οποίο webservice παίρνει τις συντεταγμένες από τον πίνακα beach, πεδία του πίνακα lat και lon, και με την χρήση του google maps api δημιουργούμε google maps για την εφαρμογή. Η απόκριση του getCoordinates webservice είναι η παρακάτω:

Αρχείο Κώδικα getCoordinates webservice response

**{**

"Items"**:** **[**

**{**

"id"**:** 716**,**

"latitude"**:** 40.62379**,**

"longitude"**:** 24.582528**,**

"name"**:** "Μεταλλεία, Λιμενάρια"

**}**

**]**

**}**

**JSON object από το getCoordinates webservice**

Παρατηρούμε ότι το JSON αντικείμενο περιέχει, το id της παραλίας για την οποία ζητήθηκαν οι συντεταγμένες, το γεωγραφηκός μήκος και πλάτος και το όνομα της παραλίας. Με την χρήση της restsharp βιβλιοθήκης, η οποία εκτελεί deserialization στα δεδομένα, έχουμε μεγαλύτερη ευελιξία στην χρήση των δεδομένων.

## Offline χρήση της εφαρμογής

Όταν το κινητό είναι συνδεδεμένο στο ίντερνετ,, μπορούν να κληθούν τα webservices με σκοπό την διασύνδεση των δεδομένων με την εφαρμογή. Τα δεδομένα που προσκομίζουν τα webservices στην εφαρμογή, αποθηκεύονται στην τοπική μνήμη του κινητού, έτσι ώστε την επόμενη φορά που χρειαστούν να μην γίνει κλήση των webservices αλλά να προσπελαστούν από την τοπική μνήμη του κινητού. Αυτό επιτυγχάνεται με την χρήση της SQLite βιβλιοθήκης.

Κάθε webservice το συνδέουμε και με μια διαφορετική κλάση μοντέλο που είναι ο πίνακας για την SQLite database. Πιο συγκεκριμένα ελέγχουμε αν υπάρχουν τα δεδομένα στην SQLite database της συσκευής στην περίπτωση που δεν υπάρχουν τότε γίνεται κλήση των webservices.

Επειδή σχεδόν όλα τα webservices που υλοποιήθηκαν για την εφαρμογή είναι ατομικά, δηλαδή επιστρέφουν δεδομένα μόνο για ένα id, εκτός από το webservice GetAllItems το οποίο επιστρέφει όλα τα δεδομένα του πίνακα beach. Σε offline χρήση της εφαρμογής θα εμφανιστούν αποτελέσματα μόνο για πεδία που είχε διατρέξει η εφαρμογή κατά την λειτουργία της με χρήση ίντερνετ. Παρακάτω, παρατίθενται αποσπάσματα κώδικα SQLite που χρησιμοποιήθηκαν για την υλοποίηση της εφαμοργής.

### beach.cs

Πρόκειται για το αρχείο κώδικα που περιέχει τις κλάσεις που είναι υπεύθυνες για την καταχώρηση των δεδομένων από τα webservices στον SQLite πίνακα beach. Στον πίνακα αυτόν καταχωρούνται το όνομα της παραλίας, το id της παραλίας, και μια φωτογραφία σχετική με την παραλία. Για να γίνει η διασύνδεση με την SQLite βάση δεδομένων χρησιμοποιείται το path προς το αρχείο το οποίο αντιπροσωπεύει την τοπική βάση δεδομένων του κινητού. Για την αποθήκευση δεδομένων στην βάση χρησιμοποιείται η κλάση μοντέλο. Δηλαδή κάθε φορά στην SQLite βάση δεδομένων αποθηκεύεται ένα αντικείμενο της κλάσης μοντέλου. Πιο συγκεκριμένα το beach.cs είναι υπέυθυνο για την αποθήκευση των δεδομένων του πίνακα beach της βάσης γνώσης στον τοπικό πίνακα της συσκευής beach, με σκοπό την offline εμφάνιση όλης της λίστας των παραλιών της ελλάδας στην οθόνη του κινητού, δηλαδή χωρίς να κληθεί το webservice GetAllItems. Παρακάτω παρατίθεται ο κώδικας του αρχείου beach.cs

Αρχείο Κώδικα beach.cs

**[**Table**(**"beach"**)]**

**public** class beach

**{**

**public** int beach\_id **{** get**;** set**;** **}**

**public** string name **{** get**;** set**;** **}**

**public** string photo **{** get**;** set**;** **}**

**}**

**public** class ToDoAllItems

**{**

**public** void DoGetAllBeaches**(**string dbPath**,** string**[]** names**,**int**[]** idds**,** string**[]** phs**)** **{**

var db **=** **new** SQLiteConnection**(**dbPath**);**

**for** **(**var i **=** 0**;** i **<** names**.**Length**;** i**++)** **{**

var newBeach **=** **new** beach**();**

newBeach**.**beach\_id **=** idds**[**i**];**

newBeach**.**name **=** names**[**i**];**

newBeach**.**photo **=** phs**[**i**];**

db**.**Insert**(**newBeach**);**

**}**

var table **=** db**.**Table**<**beach**>();**

**}**

**}**

**Αρχείο κώδικα beach.cs**

### PhotoSQLite.cs

Το αρχείο αυτό περιέχει 2 κλάσεις, την κλάση photos που αντιπροσωπεύει τον πίνακα photos ο οποίος θα αποθηκευτεί στην τοπική μνήμη του κινητού και την κλάση getPhotos η οποία διαθέτει συναρτήσεις για την αποθήκευση δεδομένων στην SQLite βάση δεδομένων αλλά και συνάρτηση για την διαχείριση των δεδομένων. Πιο συγκεκριμένα έχουμε την κλάση photos, αντιπροσωπεύει τον πίνακα photos της τοπικής βάσης της συσκευής, περιέχει τα πεδία photos, το id της παραλίας και ένα flag. Η κλάση getPhotos περιέχει την συνάρτηση accessPhotos η οποία είναι υπεύθυνη για την αποθήκευση των URLs των φωτογραφιών που επέστρεψε το getPhotos webservice για συγκεκριμένο id. Στον πίνακα photos αποθηκεύονται αντικείμενα της κλάσης photos. Η συνάρτηση getPhotoyId , ελέγχει για την ύπαρξη φωτογραφιών που είναι καταχωρημένες στον πίνακα photos με το id το οποίο τέθηκε ως είσοδος στην συνάρτηση. Αν ικανοποιείται η συνάρτηση τότε δεν καλείται το webservice getPhotos. Παρακάτω παρατίθεται ο κώδικας του αρχείου PhotoSQLite.cs

Αρχείο Κώδικα PhotoSQLite.cs

**[**Table**(**"photos"**)]**

**public** class photos

**{**

**public** string photo **{** get**;** set**;** **}**

**public** int ph\_id **{** get**;** set**;** **}**

**public** int flag **{** get**;** set**;** **}**

**}**

**public** class getPhotos

**{**

**public** void accessPhoto**(**string dbPhPath**,** List**<**photoItem**>** dph**,** TableQuery**<**photos**>** table**)**

**{**

var dbph **=** **new** SQLiteConnection**(**dbPhPath**);**

**if** **(**dph**.**Count **==** 0**)**

**{**

var new1 **=** **new** photos**();**

new1**.**flag **=** 1**;**

dbph**.**Insert**(**new1**);**

**}**

**else**

**{**

var photo\_id **=** dph**.**Select**(**item **=>** item**.**id**).**ToArray**()[**0**];**

var photos **=** dph**.**Select**(**item **=>** item**.**photo**).**ToArray**();**

var newphoto **=** **new** photos**();**

**foreach** **(**var i **in** photos**)**

**{**

newphoto**.**photo **=** i**;**

newphoto**.**ph\_id **=** photo\_id**;**

newphoto**.**flag **=** 0**;**

dbph**.**Insert**(**newphoto**);**

**}**

**}**

**}**

**public** List**<**photos**>** getPhotoById**(**int idd**,** string dbPhPath**)**

**{**

var dbph **=** **new** SQLiteConnection**(**dbPhPath**);**

**return** dbph**.**Query**<**photos**>(**"select \* from photos where ph\_id = ? and flag = ?"**,** idd**,** 0**);**

**}**

**}**

**Αρχείο κώδικα PhotoSQLite.cs**

### RegionSQLite.cs

Το αρχείο αυτό περιέχει τις κλάσεις region και getRegionItems. Η κλάση region είναι η κλάση μοντέλο και αντιπροσωπεύει τον πίνακα region της τοπικής βάσης του κινητού. Τα πεδία της κλάσης και συνεπώς του πίνακα είναι το id της παραλίας, το όνομα, η φωτογραφία, και το region id. Η κλάση getRegionItems περιέχει την συνάρτηση filltableRegion, η οποία αναλαμβάνει την αποθήκευση των δεδομένων που προέρχονται από το getRegionBeaches webservice στην τοπική βάση δεδομένων, έτσι ώστε το webservice να μην καλείται για ήδη αποθηκευμένα δεδομένα στην τοπική μνήμη του κινητού. Η συνάρτηση getRegionFSQL επιστρέφει τις εγγραφές του πίνακα region της SQLite database για το αντίστοιχο region id. Παρακάτω παρατίθεται ο κώδικας του αρχείου RegionSQLite.cs

Αρχείο Κώδικα RegionSQLite.cs

**[**Table**(**"region"**)]**

**public** class region

**{**

**public** int beach\_id **{** get**;** set**;** **}**

**public** string name **{** get**;** set**;** **}**

**public** string photo **{** get**;** set**;** **}**

**public** int regid **{** get**;** set**;** **}**

**}**

**public** class getRegionItems **{**

**public** void fillTableregion**(**string dbreg**,** List**<**tItem**>** dr**,** TableQuery**<**region**>** table**,** int point**)**

**{**

var db **=** **new** SQLiteConnection**(**dbreg**);**

var ids **=** dr**.**Select**(**item **=>** item**.**id**).**ToArray**();**

var names **=** dr**.**Select**(**item **=>** item**.**name**).**ToArray**();**

var photos **=** dr**.**Select**(**item **=>** item**.**photo**).**ToArray**();**

**for** **(**var i **=** 0**;** i **<** ids**.**Length**;** i**++)** **{**

var item **=** **new** region**();**

item**.**beach\_id **=** ids**[**i**];**

item**.**name **=** names**[**i**];**

item**.**photo **=** photos**[**i**];**

item**.**regid **=** point**;**

db**.**Insert**(**item**);**

**}**

**}**

**public** List**<**region**>** getRegionFSQL**(**string dbreg**,** int point**)** **{**

var db **=** **new** SQLiteConnection**(**dbreg**);**

**return** db**.**Query**<**region**>(**"select \* from region where regid = ?"**,** point**);**

**}**

**}**

**Αρχείο κώδικα RegionSQLite.cs**

### NearSQLite.cs

Το αρχείο αυτό περιέχει τις κλάσεις near, getNearFSQL. Η κλάση near κρατάει τα δεδομένα που προέρχονται από το webservice getNearBeaches. Πρόκειται για τον πίνακα δηλαδή της τοπικής βάσης δεδομένων της συσκευής. Έχει τις εξλης ιδιότητες που αποτελούν στήλες του πίνακα. Το όνομα της παραλίας, μια φωτογραφία της παραλίας, το Id της παραλίας, την απόσταση της παραλίας από την τοποθεσία που βρίσκεται ο χρήστης, το γεωγραφικό μήκος και πλάτος της παραλίας και τις συντεταγμένες της συσκευής. Στον πίνακα αποθηκεύονται αντικείμενα της κλάσης region. Η κλάση getNearFSQL, περιέχει την συνάρτηση fillTableNear η οποία είναι υπεύθυνη για την αποθήκευση των δεδομένων που προέρχονται από το webservice getNearBeaches. Τα δεδομένα απουηκεύονται με βάση τις συντεταγμένες του χρήστη. Δηλαδή αν ζητήθούν οι παραλίες που είναι κοντινές στην τοποθεσία 40.623790, 24.582528 ελέγχεται ο πίνακας near αν υπάρχουν εγγραφές που περιέχουν στα πεδία mylat = 40.623790 και mylon= 24.582528. Συνεπώς αν ικανοποιύνται οι παραπάνω συνθήκες δεν υπάρχει ανάγκη για κλήση του webservice getNearBeaches. Η συνάρτηση reachNear παίρνει ως ορίσματα τις συντεγμένες του κινητού και επιστρέφει τις κοντινές παραλίες που είναι αποθηκευμένες στην τοπική βάση του κινητού. Παρακάτω παρατίθεται ο κώδικας του αρχείου NearSQLite.cs

Αρχείο Κώδικα NearSQLite.cs

**[**Table**(**"near"**)]**

**public** class near

**{**

**public** string name **{** get**;** set**;** **}**

**public** string photo **{** get**;** set**;** **}**

**public** int beach\_id **{** get**;** set**;** **}**

**public** double distance **{** get**;** set**;** **}**

**public** double lon **{** get**;** set**;** **}**

**public** double lat **{** get**;** set**;** **}**

**public** string mylat **{** get**;** set**;** **}**

**public** string mylon **{** get**;** set**;** **}**

**}**

**public** class getNearFSQL

**{**

**public** void fillTableNear**(**string dbnear**,** List**<**myItem**>** nl**,** TableQuery**<**near**>** table**,** string mla**,** string ml**)**

**{**

var db **=** **new** SQLiteConnection**(**dbnear**);**

var names **=** nl**.**Select**(**item **=>** item**.**name**).**ToArray**();**

var photos **=** nl**.**Select**(**item **=>** item**.**photo**).**ToArray**();**

var ids **=** nl**.**Select**(**item **=>** item**.**id**).**ToArray**();**

var lons **=** nl**.**Select**(**item **=>** item**.**longitude**).**ToArray**();**

var lats **=** nl**.**Select**(**item **=>** item**.**latitude**).**ToArray**();**

var all\_dist **=** nl**.**Select**(**item **=>** item**.**distance**).**ToArray**();**

**for** **(**var i **=** 0**;** i **<** ids**.**Length**;** i**++)** **{**

near nn **=** **new** near**();**

nn**.**beach\_id **=** ids**[**i**];**

nn**.**distance **=** all\_dist**[**i**];**

nn**.**lat **=** lats**[**i**];**

nn**.**lon **=** lons**[**i**];**

nn**.**name **=** names**[**i**];**

nn**.**photo **=** photos**[**i**];**

nn**.**mylat **=** mla**;**

nn**.**mylon **=** ml**;**

db**.**Insert**(**nn**);**

**}**

**}**

**public** List**<**near**>** reachNear**(** string dbnear**,** string mla**,** string ml**)** **{**

var db **=** **new** SQLiteConnection**(**dbnear**);**

**return** db**.**Query**<**near**>(**"select \* from near where mylat = ? and mylon = ?"**,** mla**,** ml**);**

**}**

**}**

**Αρχείο κώδικα NearSQLite.cs**

### LatLngSQLite.cs

Το αρχείο αυτό περιέχει τις κλάσεις latlng και getCoordsFSQL. Η κλάση latlng είναι η κλάση που προσδιορίζει τον πίνακα της τοπικής βάσης τηε συσκευής. Έχει ως ιδιότητες το όνομα, το id , το γεωγραφικό μήκος και πλάτος της παραλίας. Η κλάση getCoordsFSQL περιέχει την συνάρτηση filltablelatlng, ο οποίος εκτελεί την εξής λειτουργία, παίρνει τα δεδομένα που επιστρέφει το getCoordinates webservice και τα αποθηκεύει τοπικά στην συσκευή. Αποθηκεύονται αντικείμενα της κλάσης latlng στην τοπική βάση. Η συνάρτηση getlatlngFromTable ελέγχει αν υπάρχουν οι συντεγμένες για το ζητούμενο id της παραλίας, σε περίπτωση που υπάρχουν δεν υπάρχει ανάγκη κλήσης του getCoordinates webservice.

Αρχείο Κώδικα LatLng.cs

**[**Table**(**"latlng"**)]**

**public** class latlng

**{**

**public** float lat **{** get**;** set**;** **}**

**public** float lon **{** get**;** set**;** **}**

**public** string name **{** get**;** set**;** **}**

**public** int beach\_id **{** get**;** set**;** **}**

**}**

**public** class getCoordsFSQL **{**

**public** void fillTablelatlng**(**string dblatlng**,** string json**,** TableQuery**<**latlng**>** table**)** **{**

var db **=** **new** SQLiteConnection**(**dblatlng**);**

var lll **=** **new** latlng**();**

var jsonItem **=** JObject**.**Parse**(**json**);**

lll**.**lat **=** **(**float**)**jsonItem**[**"Items"**][**0**][**"latitude"**];**

lll**.**lon **=** **(**float**)**jsonItem**[**"Items"**][**0**][**"longitude"**];**

lll**.**name **=** jsonItem**[**"Items"**][**0**][**"name"**].**ToString**();**

lll**.**beach\_id **=** **(**int**)**jsonItem**[**"Items"**][**0**][**"id"**];**

db**.**Insert**(**lll**);**

**}**

**public** List**<**latlng**>** getlatlngFromTable**(**int idd**,** string dblatlng**)**

**{**

var db **=** **new** SQLiteConnection**(**dblatlng**);**

**return** db**.**Query**<**latlng**>(**"select \* from latlng where beach\_id = ?"**,** idd**);**

**}**

**}**

### Αρχείο κώδικα LatLngSQLite.cs

### AlterNamesSQLite.cs

Αυτό το αρχείο περιέχει τις κλάσεις alter και getAlterNames. Η κλάση alter περιέχει τις στήλες του πίνακα της τοπικής βάσης της συσκευής. Έχει λοιπόν ως ιδιότητες τα όνοματα μιας παραλίας, το id της και την πηγή από την οποία προέρχεται κάθε όνομα. Η κλάση getAlterNames περιέχει την συνάρτηση fillTableNames η οποία δέχεται τα δεδομένα που προκύπτουν από το getAlternateNames webservice και τα αποθηκεύει στον πίνακα alter της τοπικής μνήμης της συσκευής. Η συνάρτηση getAltById ελέγχει αν υπάρχουν ονόματα στην τοπική μνήμη του κινητού καταχωρημένα με το id που τέθηκε σαν όρισμα στην συνάρτηση. Σε περίπτωση που υπάρχουν δεν καλείται το webservice getAlternateNames. Παρακάτω ακολουθεί ο κώδικας του αρχείου AlterNamesSQLite.cs.

Αρχείο Κώδικα AlterNameSQLite.cs

**public** class alter

**{**

**public** string names **{** get**;** set**;** **}**

**public** string source **{** get**;** set**;** **}**

**public** int beach\_id **{** get**;** set**;** **}**

**}**

**public** class getAlterNames

**{**

**public** void fillTableNames**(**string dbalter**,** List**<**nItem**>** alt**,** TableQuery**<**alter**>** table**)**

**{**

var db **=** **new** SQLiteConnection**(**dbalter**);**

var names\_id **=** alt**.**Select**(**item **=>** item**.**Id**).**ToArray**()[**0**];**

var altnames **=** alt**.**Select**(**item **=>** item**.**Names**).**ToArray**();**

var source **=** alt**.**Select**(**item **=>** item**.**Source**).**ToArray**();**

**for** **(**var i **=** 0**;** i **<** altnames**.**Length**;** i**++)**

**{**

alter al **=** **new** alter**();**

al**.**beach\_id **=** names\_id**;**

al**.**names **=** altnames**[**i**];**

al**.**source **=** source**[**i**];**

db**.**Insert**(**al**);**

**}**

**}**

**public** List**<**alter**>** getAltById**(**int idd**,** string dbalter**)**

**{**

var db **=** **new** SQLiteConnection**(**dbalter**);**

**return** db**.**Query**<**alter**>(**"select \* from alttable where beach\_id = ?"**,** idd**);**

**}**

**}**

**Αρχείο κώδικα AlterNamesSQLite.cs**

### AddressSQLite.cs

Το αρχείο αυτό περιέχει τις κλάσεις address και getAdSQL. Η κλάση address είναι η κλάση μοντέλου του ομώνυμου πίνακα στην τοπική βάση της συσκευής. Έχει ως ιδιότητες το όνομα , την διεύθυνση και το id της παραλίας. Η κλάση getAdSQL περιέχει τις συναρτήσεις fillTableAd, η οποία είναι υπεύθυνη να πάρει τα δεδομένα που παράγει το webservice getAddress και να τα αποθηκεύση στην τοπική μνήμη της συσκευής και την συνάρτηση getAddressFromTable η οποία ελέγχει αν υπάρχει η ζητούμενη διελυθυνση βάσει του id που δίνεται σαν είσοδος στην συνάρτηση. Σε περίπτωση που υπάρχει δεν καλείται το webservice getAddress.

Αρχείο Κώδικα AddresSQLite.cs

**[**Table**(**"address"**)]**

**public** class address

**{**

**public** string names **{** get**;** set**;** **}**

**public** string ad **{** get**;** set**;** **}**

**public** int beach\_id **{** get**;** set**;** **}**

**}**

**public** class getAdSQL

**{**

**public** void fillTableAd**(**string dbad**,** string json**,** TableQuery**<**address**>** table**)**

**{**

var db **=** **new** SQLiteConnection**(**dbad**);**

var addd **=** **new** address**();**

var jsonItem **=** JObject**.**Parse**(**json**);**

addd**.**names **=** jsonItem**[**"Items"**][**0**][**"Names"**].**ToString**();**

addd**.**ad **=** jsonItem**[**"Items"**][**0**][**"address"**].**ToString**();**

addd**.**beach\_id **=** **(**int**)**jsonItem**[**"Items"**][**0**][**"Id"**];**

db**.**Insert**(**addd**);**

**}**

**public** List**<**address**>** getaddressFromTable**(**int idd**,** string dbad**)**

**{**

var db **=** **new** SQLiteConnection**(**dbad**);**

**return** db**.**Query**<**address**>(**"select \* from address where beach\_id = ?"**,** idd**);**

**}**

**}**

**Αρχείο κώδικα AddressSQLite.cs**

### CommentsSQLite.cs

Το αρχείο αυτό περιέχει τις κλάσεις comments και getComments. Η κλάση comments είναι η κλάση μοντέλο που σχετίζεται με τον πίνακα της τοπικής μνήμης της συσκευής. Έχει ως ιδιότητες τα σχόλια, το Id και την πηγή από την οποία προέρχονται τα σχόλια. Η κλάση getComments περιέχει την συνάρτηση fillComments η οποία είναι υπεύθυνη για να πάρει τα δεδομένα που προέρχονται από το getDetails webservice, να απομονώσει τα κείμενα που έχουν πηγή το Foursquare, πρόκειται για την κύρια πηγή σχολιών χρηστών, και να τα αποθηκεύσει τοπικά στην μνήμη του κινητού. Η συνάρτηση getComById είναι υπεύθυνη στο να επιστρέψει τα σχόλια τα οποία είναι καταχωρημένα στην τοπική μνήμη του κινητού. Σε περίπτωση που δεν υπάρχουν τοπικά καλείται το webservice getDetails.

### descriptionSQLite.cs

Το αρχείο αυτό περιέχει τις κλάσεις desc και getDescription. Η κλάση desc είναι η κλάση μοντέλο για τον πίνακα τοπικής μνήμης της συσκευής , περιέχει τα κείμενα, το id της παραλίας και την πηγή των περιγραφών. Η κλάση getDescription περιέχει τις συναρτήσεις fillTable, η οποία παίρνει τα δεδομένα από το getDetails webservice και αποθηκεύει μόνο τα κείμενα που έχουν πηγή διαφορετική του Foursquare τοπικά στην συσκευή. Η συνάρτηση getDescById επιστρέφει τις περιγραφές που έχουν καταχωρηθεί τοπικά βάσει του id που δίνεται σαν είσοδος στην συνάρτηση. Παρακάτω παρατίθεται το αρχείο κώδικα descriptionSQLite.cs

Αρχείο Κώδικα CommentsSQLite.cs

**[**Table**(**"comments"**)]**

**public** class comments

**{**

**public** string description **{** get**;** set**;** **}**

**public** string source **{** get**;** set**;** **}**

**public** int beach\_id **{** get**;** set**;** **}**

**}**

**public** class getComments **{**

**public** void fillComments**(**string dbcom**,** List**<**dItem**>** dc**,** TableQuery**<**comments**>** table**)**

**{**

var db **=** **new** SQLiteConnection**(**dbcom**);**

var com **=** dc**.**Select**(**item **=>** item**.**description**).**ToArray**();**

var com\_id **=** dc**.**Select**(**item **=>** item**.**id**).**ToArray**()[**0**];**

var source **=** dc**.**Select**(**item **=>** item**.**source**).**ToArray**();**

**for** **(**var i **=** 0**;** i **<** com**.**Length**;** i**++)** **{**

comments ccc **=** **new** comments**();**

ccc**.**beach\_id **=** com\_id**;**

ccc**.**description **=** com**[**i**];**

ccc**.**source **=** source**[**i**];**

db**.**Insert**(**ccc**);**

**}**

**}**

**public** List**<**comments**>** getComById**(**int idd**,** string dbcom**)**

**{**

var db **=** **new** SQLiteConnection**(**dbcom**);**

**return** db**.**Query**<**comments**>(**"select \* from comments where beach\_id = ?"**,** idd**);**

**}**

**}**

**Αρχείο κώδικα CommentsSQLite.cs**

Αρχείο Κώδικα descriptionSQLite.cs

**[**Table**(**"desc"**)]**

**public** class desc

**{**

**public** string description **{** get**;** set**;** **}**

**public** string source **{** get**;** set**;** **}**

**public** int beach\_id **{** get**;** set**;** **}**

**}**

**public** class getDescription

**{**

**public** void fillTable**(**string dbdesc**,** List**<**dItem**>** details**,** TableQuery**<**desc**>** table**)**

**{**

var db **=** **new** SQLiteConnection**(**dbdesc**);**

var desc\_id **=** details**.**Select**(**item **=>** item**.**id**).**ToArray**()[**0**];**

var info **=** details**.**Select**(**item **=>** item**.**description**).**ToArray**();**

var src **=** details**.**Select**(**item **=>** item**.**source**).**ToArray**();**

**for** **(**var i **=**0**;** i**<**info**.**Length**;** i**++)**

**{**

desc d **=** **new** desc**();**

d**.**description **=** info**[**i**];**

d**.**beach\_id **=** desc\_id**;**

d**.**source **=** src**[**i**];**

db**.**Insert**(**d**);**

**}**

**}**

**public** List**<**desc**>** getDescById**(**int idd**,** string dbdesc**)**

**{**

var db **=** **new** SQLiteConnection**(**dbdesc**);**

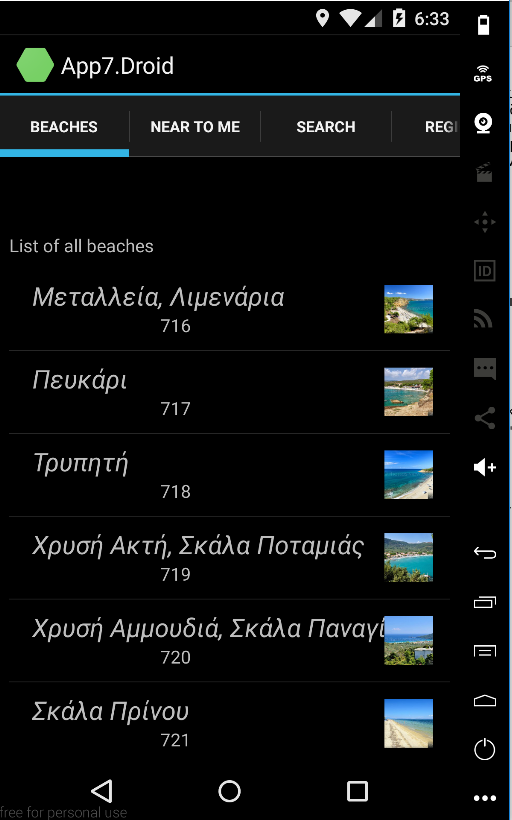
**return** db**.**Query**<**desc**>(**"select \* from desc where beach\_id = ?"**,** idd**);**

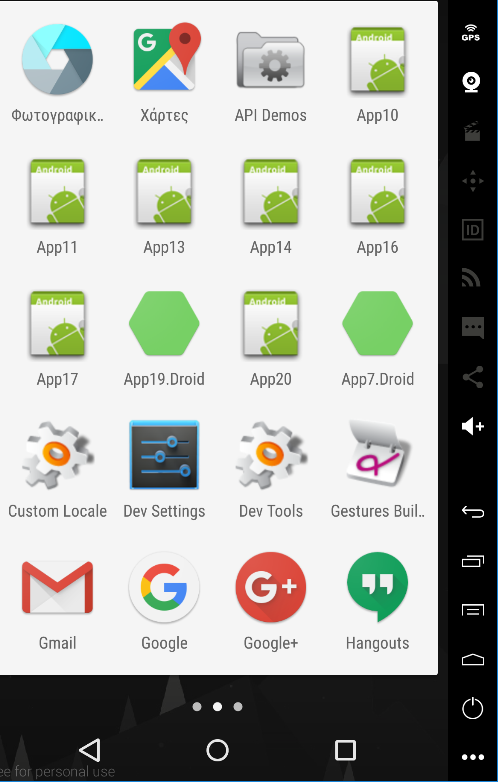
**}**

**}**

1. Εγχειρίδιο Εφαρμογής

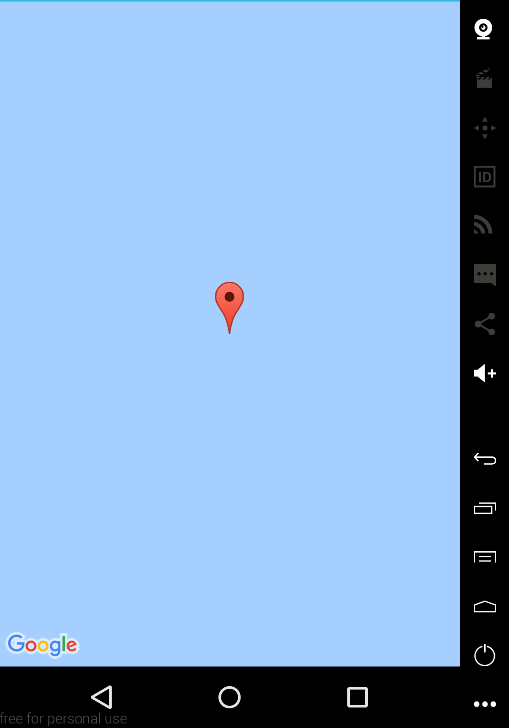
Ο χρήστης πατάει στο εικονίδιο της εφαρμογής και μεταβαίνουμε στην αρχική της οθόνη:

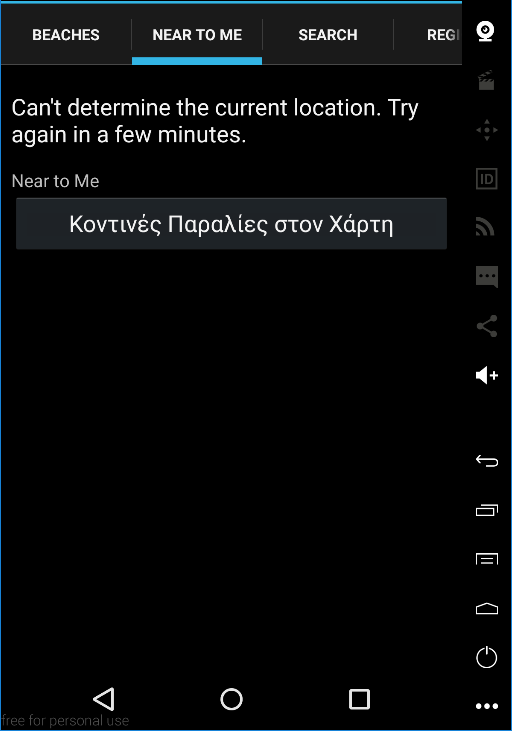




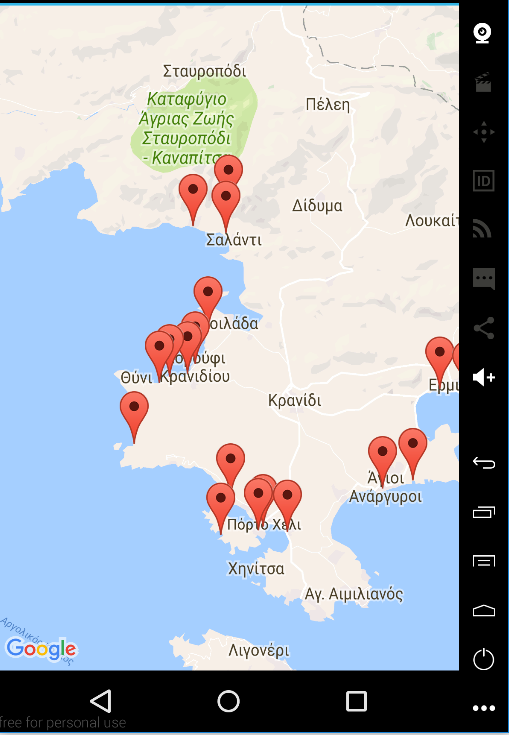
Εικόνα Εκκίνηση της Εφαρμογής

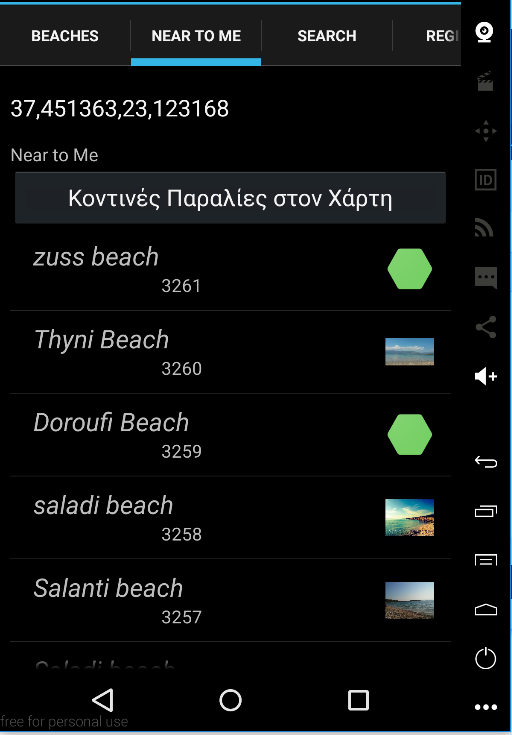
Με την εκκίνηση της εφαρμογής μεταβαίνουμε στην λίστα με όλες τις παραλίες της Ελλάδας. Παρατηρούμε ότι η αρχική οθόνη της εφαρμογή αποτελείται από tags. Στην εικόνα 58 βρισκόμαστε στο tag BEACHES που όπως αναφέρθηκε είναι λίστα με όλες τις παραλίες της Ελλάδας. Πατώντας το tag NEAR TO ME έχουμε την οθόνη της εικόνας 59 και πατώντας στο κουμπί κοντινές παραλίες στον χάρτη παίρνουμε ένα χάρτη που δείχνει στο πουθενά. Πρέπει να γίνει ενεργοποίηση του gps της συσκευής για να επιστραφούν στην οθόνη οι κοντινές παραλίες. Το λογισμικό της εφαρμογής τροφοδοτείται από τις συντεταγμένες. Ενεργοποιώντας λοιπόν το gps έχουμε το αποτέλεσμα της εικόνας 60. Εμφανίζεται η λίστα των κοντινών παραλιών του χρήστη και πατώντας το κουμπί «Κοντινές Παραλίες στον Χάρτη» εμφανίζονται στον χάρτη markers με τις τοποθεσίες παραλιών. Αν πατήσουμε πάνω στου markers εμφανίζονται τα ονόματα των παραλιών.





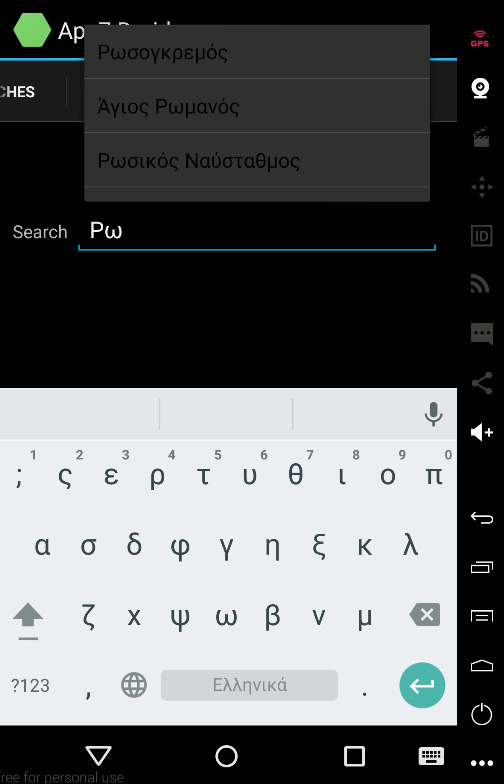
Εικόνα 62 NEAR TO ME (a)

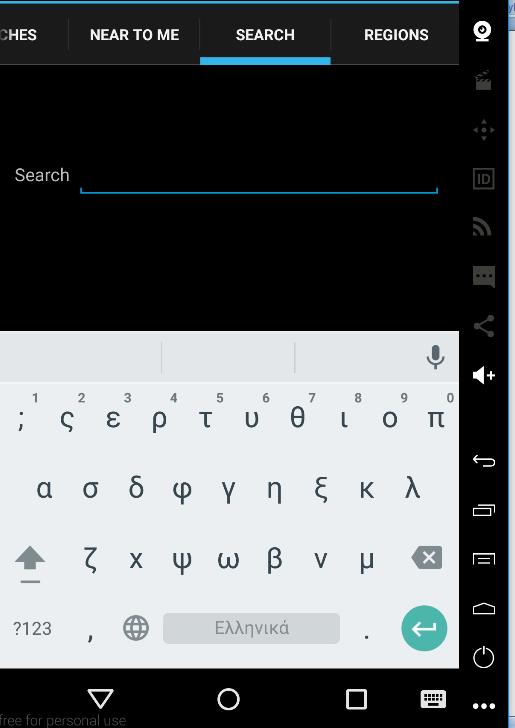




Εικόνα 63 NEAR TO ME (b)

Πατώντας στο tag SEARCH οδηγούμαστε στην παρακάτω οθόνη:

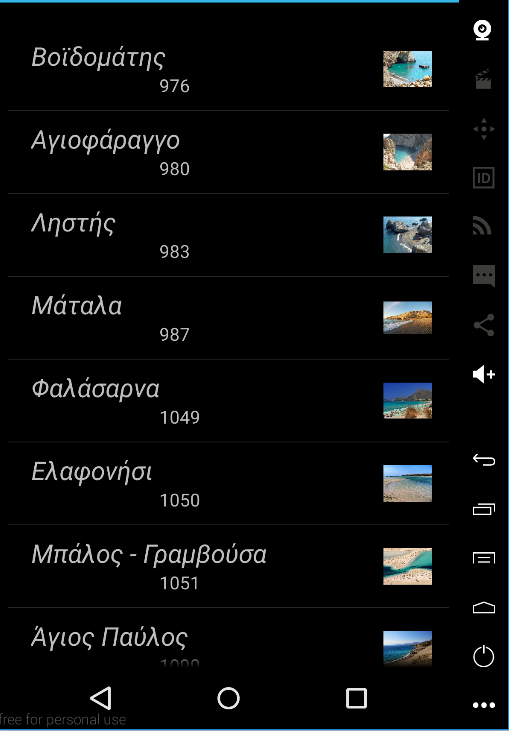


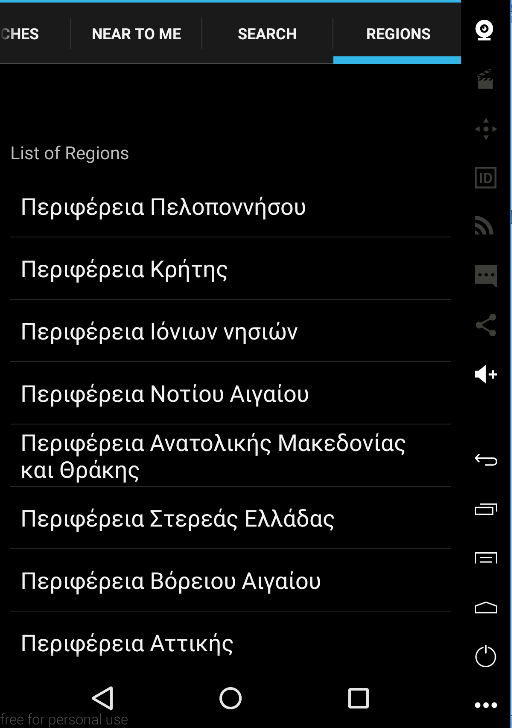


Εικόνα SEARCH

Στην μπάρα αναζήτησης αν γράψουμε οτιδήποτε , εμφανίζονται πιθανά αποτελέσματα του τι γράψαμε, πατώντας στην αναδυόμενη λίστα μπορούμε να επιλέξουμε και να μεταβούμε στην οθόνη της αντίστοιχης παραλίας.

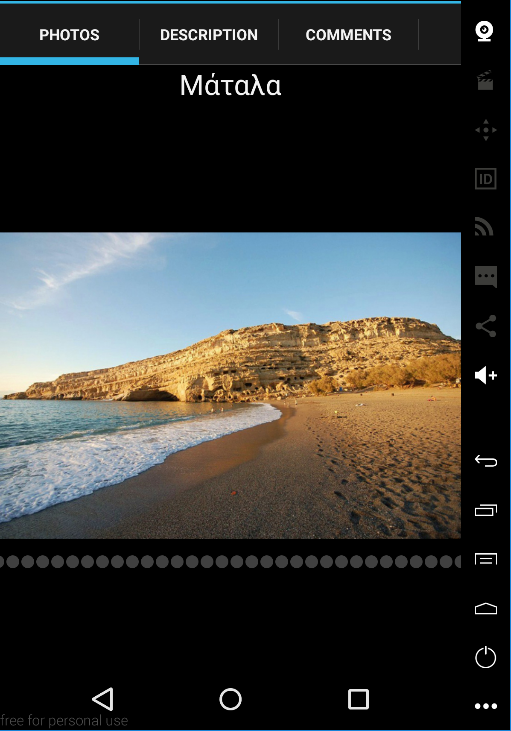
Στην συνέχεια πατώντας στο tag REGIONS μεταφερόμαστε στην οθόνη της εικόνας 62. Πρόκειται για μια λίστα με τις περιοχές της Ελλάδας. Οι παραλίες που φέρει η εφαρμογή στο tag BEACHES, τώρα έχουν ταξινομηθεί ανά περιφέρεια. Πατώντας τώρα πάνω σε μια Περιφέρεια για παράδειγμα Περιφέρεια Κρήτης μεταβαίνουμε σε μια νέα οθόνη όπου υπάρχει μια λίστα με τις παραλίες της Περιφέρειας Κρήτης. Πατώντας στο όνομα μιας παραλίας μπορούμε να μεταβούμε στην οθόνη της αντίστοιχης παραλίας.





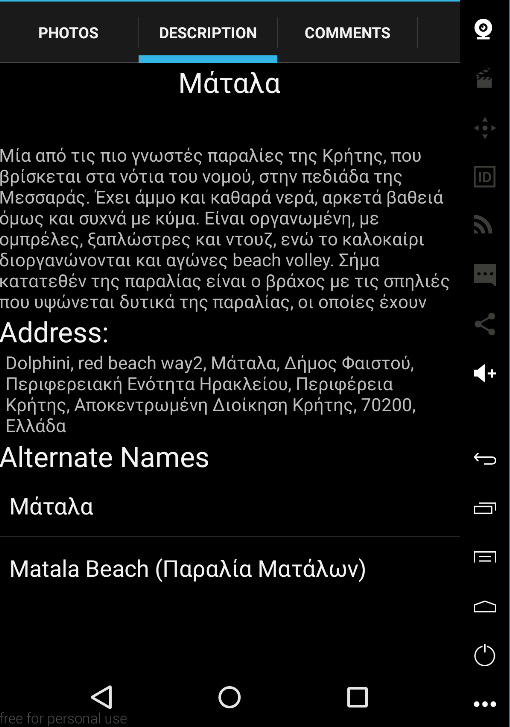
Εικόνα REGIONS

Πατώντας τυχαία στο όνομα μιας παραλίας μεταβαίνουμε σε οθόνη με πιο λεπτομερή στοιχεία που αφορούν την παραλία. Πατώντας στην παραλία Μάταλα έχουμε ως εξής:



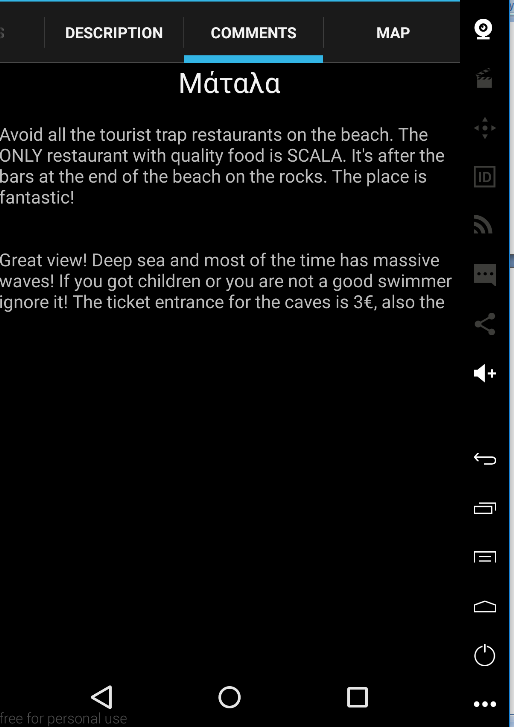
Εικόνα Οθόνη Παραλίας

Παρατηρούμε ότι η οθόνη μιας παραλίας αποτελείται από τα tags PHOTOS, DESCRIPTION, COMMENTS και MAP. Το πρώτο tag είναι το PHOTOS είναι οι φωτογραφίες που έχει συγκεντρώσει η εφαρμογή για την παραλία Μάταλα, Πατώντας στο tag DESCRIPTION έχουμε περιγραφές για την παραλία αλλά και την διεύθυνσή της καθώς και τα διαφορετικά ονόματα της παραλίας.

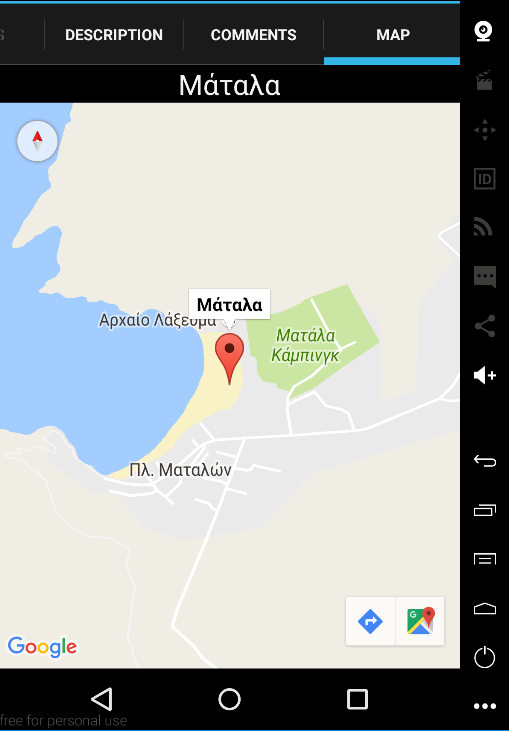


Εικόνα Πληροφορίες

Πατώντας στο tag COMMENTS μεταβαίνουμε στα σχόλια χρηστών του Foursquare που έχει συγκεντρώσει η εφαρμογή. Πρόκειται για μια dropdown λίστα που περιέχει τα σχόλια που αφορούν την συγκεκριμένη παραλία. Η οθόνη του tag COMMENTS φαίνεται στην εικόνα 65. Πατώντας το tag MAP βλέπουμε την τοποθεσία της παραλίας στον χάρτη. Πατώντας πάνω στον marker εμφανίζεται το όνομα της παραλίας. Η οθόνη του tag MAP φαίνεται στην εικόνα 66.



Εικόνα Σχόλια



Εικόνα Χάρτης

1. Παράρτημα

### App7 /MyClass.cs

**using** App7**.**Droid**;**

**using** App7**.**SQLite**;**

**using** Newtonsoft**.**Json**.**Linq**;**

**using** RestSharp**;**

**using** SQLite**;**

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**IO**;**

**using** System**.**Linq**;**

**namespace** App7

**{**

**public** class MyClass

**{**

**public** string dbPath **=** Path**.**Combine**(**System**.**Environment**.**GetFolderPath**(**System**.**Environment**.**SpecialFolder**.**Personal**),** "allbeach15.db3"**);**

**public** string dbPhPath **=** Path**.**Combine**(**System**.**Environment**.**GetFolderPath**(**System**.**Environment**.**SpecialFolder**.**Personal**),** "photosbyid2.db3"**);**

**public** string dbdesc **=** Path**.**Combine**(**System**.**Environment**.**GetFolderPath**(**System**.**Environment**.**SpecialFolder**.**Personal**),** "descbyid12.db3"**);**

**public** string dbalter **=** Path**.**Combine**(**System**.**Environment**.**GetFolderPath**(**System**.**Environment**.**SpecialFolder**.**Personal**),** "alternamesbyid12.db3"**);**

**public** string dbnear **=** Path**.**Combine**(**System**.**Environment**.**GetFolderPath**(**System**.**Environment**.**SpecialFolder**.**Personal**),** "neardb2.db3"**);**

**public** string dbreg **=** Path**.**Combine**(**System**.**Environment**.**GetFolderPath**(**System**.**Environment**.**SpecialFolder**.**Personal**),** "region11.db3"**);**

**public** string dbad **=** Path**.**Combine**(**System**.**Environment**.**GetFolderPath**(**System**.**Environment**.**SpecialFolder**.**Personal**),** "ad1.db3"**);**

**public** string dblatlng **=** Path**.**Combine**(**System**.**Environment**.**GetFolderPath**(**System**.**Environment**.**SpecialFolder**.**Personal**),** "latlng1.db3"**);**

**public** string dbcom **=** Path**.**Combine**(**System**.**Environment**.**GetFolderPath**(**System**.**Environment**.**SpecialFolder**.**Personal**),** "comments1.db3"**);**

**public** MyClass **()**

**{**

**}**

**public** void readItems**()**

**{**

SQLiteConnection db **=** **new** SQLiteConnection**(**dbPath**);**

db**.**CreateTable**<**beach**>();**

var table **=** db**.**Table**<**beach**>();**

**if** **(**table**.**Count**()** **==** 0**)**

**{**

var client **=** **new** RestClient**(**"http://192.168.1.3:5000/GetAllItems"**);**

var request **=** **new** RestRequest**(**Method**.**GET**);**

var response2 **=** client**.**Execute**<**RootObject**>(**request**);**

callback**(**response2**.**Data**.**Items**,** 0**);**

**}**

**else** **{**

var mylist **=** **new** List**<**Item**>();**

**foreach** **(**var s **in** table**)** **{**

var data **=** **new** Item

**{**

id **=** s**.**beach\_id**,**

name **=** s**.**name**,**

photo **=** s**.**photo

**};**

mylist**.**Add**(**data**);**

**}**

callback**(**mylist**,** 1**);**

**}**

**}**

var mylist **=** **new** List**<**Item**>();**

**foreach** **(**var s **in** table**)** **{**

var data **=** **new** Item

**{**

id **=** s**.**beach\_id**,**

name **=** s**.**name**,**

photo **=** s**.**photo

**};**

mylist**.**Add**(**data**);**

**}**

callback**(**mylist**,** 1**);**

**}**

**}**

**public** void callback**(**List**<**Item**>** d**,** int checkStock**)**

**{**

FromAllItemClass**.**allItems **=** d**;**

MainActivity**.**names **=** d**.**Select**(**item **=>** item**.**name**).**ToArray**();**

MainActivity**.**idds **=** d**.**Select**(**item **=>** item**.**id**).**ToArray**();**

MainActivity**.**phs **=** d**.**Select**(**item **=>** item**.**photo**).**ToArray**();**

toNear**.**mynames **=** MainActivity**.**names**;**

toNear**.**myids **=** MainActivity**.**idds**;**

**if** **(**checkStock **==** 0**)**

**{**

ToDoAllItems myallItems **=** **new** ToDoAllItems**();**

myallItems**.**DoGetAllBeaches**(**dbPath**,** MainActivity**.**names**,** MainActivity**.**idds**,** MainActivity**.**phs**);**

**}**

**}**

**public** void getNearBeaches**()**

**{**

SQLiteConnection db **=** **new** SQLiteConnection**(**dbnear**);**

db**.**CreateTable**<**near**>();**

var table **=** db**.**Table**<**near**>();**

var nnn **=** **new** getNearFSQL**();**

var string1 **=** toNear**.**mylat**.**ToString**().**Replace**(**','**,** '.'**);**

var string2 **=** toNear**.**mylon**.**ToString**().**Replace**(**','**,** '.'**);**

var nearList **=** nnn**.**reachNear**(**dbnear**,** string1**,** string2**);**

**if** **(**nearList**.**Count **==** 0**)**

**{**

var c **=** toNear**.**mylon**;**

var url **=** "http://192.168.1.3:5000/getNearBeaches/" **+** string1 **+** "/" **+** string2**;**

var client **=** **new** RestClient**(**url**);**

var request **=** **new** RestRequest**(**Method**.**GET**);**

var response3 **=** client**.**Execute**<**myRootObject**>(**request**);**

**if** **(**response3**.**Data **==** **null)**

**{**

var para\_data **=** **new** List**<**myItem**>();**

var data1 **=** **new** myItem

**{**

id **=** 0**,**

distance **=** 0.0**,**

latitude **=** 0.0**,**

longitude **=** 0.0**,**

name **=** "unknown"**,**

photo **=** ""

**};**

para\_data**.**Add**(**data1**);**

near**(**para\_data**);**

**}**

**else**

**{**

nnn**.**fillTableNear**(**dbnear**,** response3**.**Data**.**Items**,** table**,** string1**,** string2**);**

near**(**response3**.**Data**.**Items**);**

**}**

**}**

**else** **{**

var result **=** **new** List**<**myItem**>();**

**foreach** **(**var s **in** nearList**)** **{**

var data **=** **new** myItem

**{**

id **=** s**.**beach\_id**,**

distance **=** s**.**distance**,**

latitude **=** s**.**lat**,**

longitude **=** s**.**lon**,**

name **=** s**.**name**,**

photo **=** s**.**photo

**};**

result**.**Add**(**data**);**

**}**

near**(**result**);**

**}**

**}**

**else** **{**

var result **=** **new** List**<**myItem**>();**

**foreach** **(**var s **in** nearList**)** **{**

var data **=** **new** myItem

**{**

id **=** s**.**beach\_id**,**

distance **=** s**.**distance**,**

latitude **=** s**.**lat**,**

longitude **=** s**.**lon**,**

name **=** s**.**name**,**

photo **=** s**.**photo

**};**

result**.**Add**(**data**);**

**}**

near**(**result**);**

**}**

**}**

**public** void near**(**List**<**myItem**>** li**)**

**{**

FromAllItemClass**.**nearItems **=** li**;**

var myArray2 **=** li**.**Select**(**item **=>** item**.**name**).**ToArray**();**

var myArray3 **=** li**.**Select**(**item **=>** item**.**id**).**ToArray**();**

var mylats **=** li**.**Select**(**item **=>** item**.**latitude**).**ToArray**();**

var mylons **=** li**.**Select**(**item **=>** item**.**longitude**).**ToArray**();**

MainActivity**.**lats **=** mylats**;**

MainActivity**.**lons **=** mylons**;**

MainActivity**.**names **=** myArray2**;**

MainActivity**.**idds **=** myArray3**;**

helloMap**.**hnames **=** MainActivity**.**names**;**

helloMap**.**hellolons **=** MainActivity**.**lons**;**

helloMap**.**hellolats **=** MainActivity**.**lats**;**

**}**

**public** void getRegionBeaches**()**

**{**

SQLiteConnection db **=** **new** SQLiteConnection**(**dbreg**);**

db**.**CreateTable**<**region**>();**

var table **=** db**.**Table**<**region**>();**

var rrr **=** **new** getRegionItems**();**

var regList **=** rrr**.**getRegionFSQL**(**dbreg**,** fromRegionFrag**.**regionPointer**);**

**if** **(**regList**.**Count **==** 0**)**

**{**

var url **=** "http://192.168.1.3:5000/getRegionBeaches/" **+** fromRegionFrag**.**regionPointer**.**ToString**();**

var client **=** **new** RestClient**(**url**);**

var request **=** **new** RestRequest**(**Method**.**GET**);**

var response4 **=** client**.**Execute**<**tRootObject**>(**request**);**

rrr**.**fillTableregion**(**dbreg**,** response4**.**Data**.**Items**,** table**,** fromRegionFrag**.**regionPointer**);**

regionCall**(**response4**.**Data**.**Items**);**

**}**

**else** **{**

var result **=** **new** List**<**tItem**>();**

**foreach(**var s **in** regList**)**

**{**

var data **=** **new** tItem

**{**

id **=** s**.**beach\_id**,**

name **=** s**.**name**,**

photo **=** s**.**photo

**};**

result**.**Add**(**data**);**

**}**

regionCall**(**result**);**

**}**

**}**

**public** void regionCall**(**List**<**tItem**>** d**)**

**{**

FromAllItemClass**.**regionItems **=** d**;**

var beachArray **=** d**.**Select**(**item **=>** item**.**name**).**ToArray**();**

RegionActivity**.**regAr **=** beachArray**;**

var beachIdsArray **=** d**.**Select**(**item **=>** item**.**id**).**ToArray**();**

RegionActivity**.**regId **=** beachIdsArray**;**

**}**

**public** void readLatLng**(**int idd**)**

**{**

SQLiteConnection db **=** **new** SQLiteConnection**(**dblatlng**);**

db**.**CreateTable**<**latlng**>();**

var table **=** db**.**Table**<**latlng**>();**

var ltln **=** **new** getCoordsFSQL**();**

var latlngList **=** ltln**.**getlatlngFromTable**(**idd**,** dblatlng**);**

**if** **(**latlngList**.**Count**()** **==** 0**)**

**{**

var url **=** "http://192.168.1.3:5000/getCoordinates/" **+** idd**.**ToString**();**

var client **=** **new** RestClient**(**url**);**

var request **=** **new** RestRequest**(**Method**.**GET**);**

var response5 **=** client**.**Execute**<**RootLatLong**>(**request**);**

ltln**.**fillTablelatlng**(**dblatlng**,** response5**.**Content**,** table**);**

DrawMap**(**response5**.**Content**);**

**}**

**else**

**{**

var list **=** **new** List**<**LatlongItem**>();**

**foreach** **(**var s **in** latlngList**)**

**{**

var data **=** **new** LatlongItem

**{**

id **=** s**.**beach\_id**,**

latitude **=** s**.**lat**,**

longitude **=** s**.**lon**,**

name **=** s**.**name

**};**

list**.**Add**(**data**);**

**}**

var item **=** **new** RootLatLong

**{**

Items **=** list

**};**

string json **=** Newtonsoft**.**Json**.**JsonConvert**.**SerializeObject**(**item**);**

DrawMap**(**json**);**

**}**

**}**

**public** void DrawMap**(**string jsonString**)**

**{**

var jsonItem **=** JObject**.**Parse**(**jsonString**);**

var mylat **=** jsonItem**[**"Items"**][**0**][**"latitude"**].**ToString**();**

ItemContext**.**lat **=** mylat**;**

var mylon **=** jsonItem**[**"Items"**][**0**][**"longitude"**].**ToString**();**

ItemContext**.**lon **=** mylon**;**

var myuri **=** "geo:" **+** ItemContext**.**lat **+** "," **+** ItemContext**.**lon**;**

helloMap**.**helloLat **=** ItemContext**.**lat**;**

helloMap**.**helloLng **=** ItemContext**.**lon**;**

helloMap**.**hellobeachId **=** SharedObjects**.**mainId**;**

**}**

**public** List**<**photoItem**>** getPhotos**(**int idd**)**

**{**

SQLiteConnection dbph **=** **new** SQLiteConnection**(**dbPhPath**);**

dbph**.**CreateTable**<**photos**>();**

var table **=** dbph**.**Table**<**photos**>();**

var gg **=** **new** getPhotos**();**

var phlist **=** gg**.**getPhotoById**(**idd**,** dbPhPath**);**

**if** **(**phlist**.**Count**()** **==** 0**)**

**{**

var url **=** "http://192.168.1.3:5000/getPhotos/" **+** idd**.**ToString**();**

var client **=** **new** RestClient**(**url**);**

var request **=** **new** RestRequest**(**Method**.**GET**);**

var response6 **=** client**.**Execute**<**PhObject**>(**request**);**

gg**.**accessPhoto**(**dbPhPath**,** response6**.**Data**.**Items**,** table**);**

**return** response6**.**Data**.**Items**;**

**}**

**else** **{**

var result **=** **new** List**<**photoItem**>();**

**foreach** **(**var s **in** phlist**)** **{**

var data **=** **new** photoItem

**{**

id **=** s**.**ph\_id**,**

photo **=** s**.**photo

**};**

result**.**Add**(**data**);**

**}**

**return** result**;**

**}**

**}**

**public** List**<**dItem**>** readDesc**(**int idd**)**

**{**

SQLiteConnection db **=** **new** SQLiteConnection**(**dbdesc**);**

db**.**CreateTable**<**desc**>();**

var table **=** db**.**Table**<**desc**>();**

var ddd **=** **new** getDescription**();**

var desclist **=** ddd**.**getDescById**(**idd**,** dbdesc**);**

**if** **(**desclist**.**Count**()** **==** 0**)**

**{**

var url **=** "http://192.168.1.3:5000/getDetails/" **+** idd**.**ToString**();**

var client **=** **new** RestClient**(**url**);**

var request **=** **new** RestRequest**(**Method**.**GET**);**

var response7 **=** client**.**Execute**<**dRootObject**>(**request**);**

ddd**.**fillTable**(**dbdesc**,** response7**.**Data**.**Items**,** table**);**

**return** response7**.**Data**.**Items**;**

**}**

**else**

**{**

var result **=** **new** List**<**dItem**>();**

**foreach** **(**var s **in** desclist**)**

**{**

var data **=** **new** dItem

**{**

id **=** s**.**beach\_id**,**

description **=** s**.**description**,**

source **=** s**.**source

**};**

result**.**Add**(**data**);**

**}**

**return** result**;**

**}**

**}**

**public** string showAddress**(**int idd**)**

**{**

SQLiteConnection db **=** **new** SQLiteConnection**(**dbad**);**

db**.**CreateTable**<**address**>();**

var table **=** db**.**Table**<**address**>();**

var aaaa **=** **new** getAdSQL**();**

var adList **=** aaaa**.**getaddressFromTable**(**idd**,** dbad**);**

**if** **(**adList**.**Count**()** **==** 0**)**

**{**

var url **=** "http://192.168.1.3:5000/getAddress/" **+** idd**.**ToString**();**

var client **=** **new** RestClient**(**url**);**

var request **=** **new** RestRequest**(**Method**.**GET**);**

var response8 **=** client**.**Execute**<**adRootObject**>(**request**);**

aaaa**.**fillTableAd**(**dbad**,** response8**.**Content**,** table**);**

**return** response8**.**Content**;**

**}**

**else** **{**

var items **=** **new** List**<**adItem**>();**

**foreach** **(**var s **in** adList**)**

**{**

var data **=** **new** adItem

**{**

Id **=** s**.**beach\_id**,**

Names **=** s**.**names**,**

address **=** s**.**ad

**};**

items**.**Add**(**data**);**

**}**

var item **=** **new** adRootObject

**{**

Items **=** items

**};**

string json **=** Newtonsoft**.**Json**.**JsonConvert**.**SerializeObject**(**item**);**

**return** json**;**

**}**

**}**

**public** List**<**nItem**>** getAlter**(**int idd**)**

**{**

SQLiteConnection db **=** **new** SQLiteConnection**(**dbalter**);**

db**.**CreateTable**<**alter**>();**

var table **=** db**.**Table**<**alter**>();**

var alll **=** **new** getAlterNames**();**

var alterlist **=** alll**.**getAltById**(**idd**,** dbalter**);**

**if** **(**alterlist**.**Count**()** **==** 0**)**

**{**

var url **=** "http://192.168.1.3:5000/getAlternateNames/" **+** idd**.**ToString**();**

var client **=** **new** RestClient**(**url**);**

var request **=** **new** RestRequest**(**Method**.**GET**);**

var response9 **=** client**.**Execute**<**nRootObject**>(**request**);**

alll**.**fillTableNames**(**dbalter**,** response9**.**Data**.**Items**,** table**);**

**return** response9**.**Data**.**Items**;**

**}**

**else**

**{**

var result **=** **new** List**<**nItem**>();**

**foreach** **(**var s **in** alterlist**)** **{**

var data **=** **new** nItem

**{**

Id **=** s**.**beach\_id**,**

Names **=** s**.**names**,**

Source **=** s**.**source

**};**

result**.**Add**(**data**);**

**}**

**return** result**;**

**}**

**}**

**public** List**<**dItem**>** readComments**(**int idd**)**

**{**

SQLiteConnection db **=** **new** SQLiteConnection**(**dbcom**);**

db**.**CreateTable**<**comments**>();**

var table **=** db**.**Table**<**comments**>();**

var cocc **=** **new** getComments**();**

var commentsList **=** cocc**.**getComById**(**idd**,** dbcom**);**

**if** **(**commentsList**.**Count **==** 0**)**

**{**

var url **=** "http://192.168.1.3:5000/getDetails/" **+** idd**.**ToString**();**

var client **=** **new** RestClient**(**url**);**

var request **=** **new** RestRequest**(**Method**.**GET**);**

var response10 **=** client**.**Execute**<**dRootObject**>(**request**);**

cocc**.**fillComments**(**dbcom**,** response10**.**Data**.**Items**,** table**);**

**return** response10**.**Data**.**Items**;**

**}**

**else** **{**

var result **=** **new** List**<**dItem**>();**

**foreach** **(**var s **in** commentsList**)**

**{**

var data **=** **new** dItem

**{**

id **=** s**.**beach\_id**,**

description **=** s**.**description**,**

source = s.sourceΑρχείο Κώδικα 50

**};**

result**.**Add**(**data**);**

**}**

**return** result**;**

**}**

**}**

Αρχείο Κώδικα App7/MyClass.cs

### App7.Droid/ MainActivity.cs

**using** System**;**

**using** Android**.**Locations**;**

**using** Android**.**App**;**

**using** Android**.**Content**;**

**using** Android**.**Runtime**;**

**using** Android**.**Views**;**

**using** Android**.**Widget**;**

**using** Android**.**OS**;**

**using** System**.**Collections**.**Generic**;**

**using** RestSharp**;**

**using** System**.**Linq**;**

**using** Android**.**Util**;**

**namespace** App7**.**Droid

**{**

**[**Activity**(**Label **=** "App7.Droid"**,** MainLauncher **=** **true,** Icon **=** "@drawable/icon"**)]**

**public** class MainActivity **:** ListActivity**,** ILocationListener

**{**

static **readonly** string TAG **=** "X:" **+** **typeof(**MainActivity**).**Name**;**

Location \_currentLocation**;**

LocationManager \_locationManager**;**

string \_locationProvider**;**

TextView \_locationText**;**

**public** static string**[]** names**;**

**public** static int**[]** idds**;**

**public** static string**[]** phs**;**

**public** static double**[]** lats**;**

**public** static double**[]** lons**;**

**public** void OnLocationChanged**(**Location location**)**

**{**

\_currentLocation **=** location**;**

**if** **(**\_currentLocation **==** **null)**

**{**

\_locationText**.**Text **=** "Unable to determine your location. Try again in a short while."**;**

**}**

**else**

**{**

toNear**.**mylat **=** \_currentLocation**.**Latitude**.**ToString**();**

toNear**.**mylon **=** \_currentLocation**.**Longitude**.**ToString**();**

\_locationText**.**Text **=** string**.**Format**(**"{0:f6},{1:f6}"**,** \_currentLocation**.**Latitude**,** \_currentLocation**.**Longitude**);**

**}**

**}**

**public** void OnProviderDisabled**(**string provider**)** **{** **}**

**public** void OnProviderEnabled**(**string provider**)** **{** **}**

**public** void OnStatusChanged**(**string provider**,** Availability status**,** Bundle extras**)**

**{**

Log**.**Debug**(**TAG**,** "{0}, {1}"**,** provider**,** status**);**

**}**

**protected** **override** void OnCreate**(**Bundle bundle**)**

**{**

**base.**OnCreate**(**bundle**);**

SetContentView**(**Resource**.**Layout**.**Main**);**

**this.**ActionBar**.**NavigationMode **=** ActionBarNavigationMode**.**Tabs**;**

\_locationText **=** FindViewById**<**TextView**>(**Resource**.**Id**.**location\_text**);**

//textView = FindViewById<AutoCompleteTextView>(Resource.Id.autoCompleteTextView1);

AddTab**(**"Beaches"**,** **new** AllItems**());**

AddTab**(**"Near To Me"**,** **new** NearItems**());**

AddTab**(**"Search"**,** **new** SearchMe**());**

AddTab**(**"Regions"**,** **new** regionItems**());**

InitializeLocationManager**();**

**}**

**public** void clearTextLoc**()** **{**

\_locationText**.**Text **=** ""**;**

**}**

void InitializeLocationManager**()**

**{**

\_locationManager **=** **(**LocationManager**)**GetSystemService**(**LocationService**);**

Criteria criteriaForLocationService **=** **new** Criteria

**{**

Accuracy **=** Accuracy**.**Fine

**};**

IList**<**string**>** acceptableLocationProviders **=** \_locationManager**.**GetProviders**(**criteriaForLocationService**,** **true);**

**if** **(**acceptableLocationProviders**.**Any**())**

**{**

\_locationProvider **=** acceptableLocationProviders**.**First**();**

**}**

**else**

**{**

\_locationProvider **=** string**.**Empty**;**

**}**

Log**.**Debug**(**TAG**,** "Using " **+** \_locationProvider **+** "."**);**

**}**

**protected** **override** void OnResume**()**

**{**

**base.**OnResume**();**

\_locationManager**.**RequestLocationUpdates**(**\_locationProvider**,** 0**,** 0**,** **this);**

Log**.**Debug**(**TAG**,** "Listening for location updates using " **+** \_locationProvider **+** "."**);**

**}**

**protected** **override** void OnPause**()**

**{**

**base.**OnPause**();**

\_locationManager**.**RemoveUpdates**(this);**

Log**.**Debug**(**TAG**,** "No longer listening for location updates."**);**

**}**

**public** void AddressButton\_OnClick**()**

**{**

**if** **(**\_currentLocation **==** **null)**

**{**

\_locationText**.**Text **=** "Can't determine the current location. Try again in a few minutes."**;**

**return;**

**}**

toNear**.**mylat **=** \_currentLocation**.**Latitude**.**ToString**();**

toNear**.**mylon **=** \_currentLocation**.**Longitude**.**ToString**();**

MyClass myclass **=** **new** MyClass**();**

myclass**.**getNearBeaches**();**

RunOnUiThread**(()** **=>**

**{**

ListAdapter **=** **new** NearAdapter**(this,** FromAllItemClass**.**nearItems**);**

**});**

**}**

void AddTab**(**string tabText**,** Fragment fragment**)**

**{**

var tab **=** **this.**ActionBar**.**NewTab**();**

tab**.**SetText**(**tabText**);**

// must set event handler for replacing tabs tab

tab**.**TabSelected **+=** **delegate** **(object** sender**,** ActionBar**.**TabEventArgs e**)** **{**

e**.**FragmentTransaction**.**Replace**(**Resource**.**Id**.**frameLayout1**,** fragment**);**

**};**

**this.**ActionBar**.**AddTab**(**tab**);**

**}**

**public** void TextView\_ItemClick**(object** sender**,** AdapterView**.**ItemClickEventArgs e**)**

**{**

var activity2 **=** **new** Intent**(this,** **typeof(**ItemContext**));**

var beachName **=** names**[**e**.**Position**];**

var beachId **=** idds**[**e**.**Position**];**

SharedObjects**.**mainName **=** beachName**;**

SharedObjects**.**mainId **=** beachId**;**

StartActivity**(typeof(**ItemContext**));**

**}**

AdapterView**.**ItemClickEventArgs e**)**

**{**

**if** **(**checkMode**.**mode **==** 1**)**

**{**

var pointer **=** e**.**Position**;**

fromRegionFrag**.**regionPointer **=** pointer**;**

var intent **=** **new** Intent**(this,** **typeof(**RegionActivity**))**

**.**SetFlags**(**ActivityFlags**.**ReorderToFront**);**

StartActivity**(**intent**);**

**}**

**else** **{**

var activity2 **=** **new** Intent**(this,** **typeof(**ItemContext**));**

var beachName **=** names**[**e**.**Position**];**

var beachId **=** idds**[**e**.**Position**];**

SharedObjects**.**mainName **=** beachName**;**

SharedObjects**.**mainId **=** beachId**;**

var intent **=** **new** Intent**(this,** **typeof(**ItemContext**))**

**.**SetFlags**(**ActivityFlags**.**ReorderToFront**);**

StartActivity**(**intent**);**

**}**

**}**

Αρχείο Κώδικα 52 App7.Droid/MainActivity.cs

### App7.Droid/ ItemContext.cs

Αρχείο Κώδικα App7.Droid / ItemContext.cs

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** RestSharp**;**

**using** Android**.**App**;**

**using** Android**.**Content**;**

**using** Android**.**OS**;**

**using** Android**.**Runtime**;**

**using** Android**.**Views**;**

**using** Android**.**Widget**;**

**using** Newtonsoft**.**Json**.**Linq**;**

**using** Android**.**Gms**.**Maps**;**

**using** Android**.**Gms**.**Maps**.**Model**;**

**using** Macaw**.**UIComponents**;**

**using** Android**.**Graphics**;**

**using** Android**.**Support**.**V4**.**App**;**

**namespace** App7**.**Droid

**{**

**[**Activity**(**Label **=** "App7.Droid"**,** Icon **=** "@drawable/icon"**)]**

**public** class ItemContext **:** FragmentActivity

**{**

**public** static string lat**;**

**public** static string lon**;**

//Button mapsButton;

**protected** **override** void OnCreate**(**Bundle savedInstanceState**)**

**{**

**base.**OnCreate**(**savedInstanceState**);**

SetContentView**(**Resource**.**Layout**.**ItemContext**);**

**this.**ActionBar**.**NavigationMode **=** ActionBarNavigationMode**.**Tabs**;**

//imageView = FindViewById<MultiImageView>(Resource.Id.imageView1);

//mapsButton = FindViewById<Button>(Resource.Id.button1);

var myTextView **=** FindViewById**<**TextView**>(**Resource**.**Id**.**textView1**);**

var mybeachId **=** SharedObjects**.**mainId**;**

MyClass mycc **=** **new** MyClass**();**

mycc**.**readLatLng**(**SharedObjects**.**mainId**);**

myTextView**.**Text **=** SharedObjects**.**mainName**;**

AddTab**(**"Photos"**,** **new** showPhotos**());**

AddTab**(**"Description"**,** **new** beachDesc**());**

AddTab**(**"Comments"**,** **new** beachComments**());**

AddTab**(**"Map"**,** **new** myMap**());**

// readLatLng(mybeachId);

//getPhotos(mybeachId);

**}**

**public** void OnMapReady**(**GoogleMap googleMap**)**

**{**

googleMap**.**AddMarker**(new** MarkerOptions**()**

**.**SetPosition**(new** LatLng**(**50.897778**,** 3.013333**))**

**.**SetTitle**(**"Marker"**));**

**}**

void AddTab**(**string tabText**,** Android**.**App**.**Fragment view**)**

**{**

var tab **=** **this.**ActionBar**.**NewTab**();**

tab**.**SetText**(**tabText**);**

tab**.**TabSelected **+=** **delegate** **(object** sender**,** ActionBar**.**TabEventArgs e**)** **{**

var fragment **=** **this.**FragmentManager**.**FindFragmentById**(**Resource**.**Id**.**frameLayout1**);**

**if** **(**fragment **!=** **null)**

e**.**FragmentTransaction**.**Remove**(**fragment**);**

e**.**FragmentTransaction**.**Add**(**Resource**.**Id**.**frameLayout1**,** view**);**

**};**

tab**.**TabUnselected **+=** **delegate** **(object** sender**,** ActionBar**.**TabEventArgs e**)** **{**

e**.**FragmentTransaction**.**Remove**(**view**);**

**};**

**this.**ActionBar**.**AddTab**(**tab**);**

**}**

### App7. Droid/Fragment1.cs

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** Android**.**App**;**

**using** Android**.**Content**;**

**using** Android**.**OS**;**

**using** Android**.**Runtime**;**

**using** Android**.**Util**;**

**using** Android**.**Views**;**

**using** Android**.**Widget**;**

**namespace** App7**.**Droid

**{**

**public** class NearItems **:** ListFragment

**{**

**public** **override** void OnCreate**(**Bundle savedInstanceState**)**

**{**

**base.**OnCreate**(**savedInstanceState**);**

// Create your fragment here

**}**

**public** **override** View OnCreateView**(**LayoutInflater inflater**,** ViewGroup container**,** Bundle savedInstanceState**)**

**{**

**base.**OnCreateView**(**inflater**,** container**,** savedInstanceState**);**

var view **=** inflater**.**Inflate**(**Resource**.**Layout**.**AllLayout**,** container**,** **false);**

var sampleTextView **=** view**.**FindViewById**<**TextView**>(**Resource**.**Id**.**textView1**);**

sampleTextView**.**Text **=** "Near to Me"**;**

checkMode**.**mode **=** 0**;**

var myActivity **=** **(**MainActivity**)this.**Activity**;**

myActivity**.**clearList**();**

myActivity**.**AddressButton\_OnClick**();**

myActivity**.**ListView**.**ItemClick **+=** myActivity**.**ListView\_ItemClick**;**

Button aButton **=** **new** Button**(**myActivity**);**

aButton**.**Text **=** "Κοντινές Παραλίες στον Χάρτη"**;**

LinearLayout ll **=** view**.**FindViewById**<**LinearLayout**>(**Resource**.**Id**.**linearLayout1**);**

ll**.**AddView**(**aButton**);**

aButton**.**Click **+=** **(**sender**,** e**)** **=>**

**{**

Intent intent **=** **new** Intent**(this.**Activity**,** **typeof(**ShowMap**));**

StartActivity**(**intent**);**

**};**

**return** view**;**

**}**

**}**

**public** class regionItems **:** ListFragment

**{**

**public** string**[]** regionArray **=** **{** "Περιφέρεια Πελοποννήσου"**,** "Περιφέρεια Κρήτης"**,** "Περιφέρεια Ιόνιων νησιών"**,** "Περιφέρεια Νοτίου Αιγαίου"**,** "Περιφέρεια Ανατολικής Μακεδονίας και Θράκης"**,** "Περιφέρεια Στερεάς Ελλάδας"**,** "Περιφέρεια Βόρειου Αιγαίου"**,** "Περιφέρεια Αττικής"**,** "Περιφέρεια Θεσσαλίας"**,** "Περιφέρεια Κεντρικής Μακεδονίας"**,** "Περιφέρεια Ηπείρου"**,** "Περιφέρεια Δυτικής Ελλάδας" **};**

**public** **override** void OnCreate**(**Bundle savedInstanceState**)**

**{**

**base.**OnCreate**(**savedInstanceState**);**

// Create your fragment here

**}**

**public** **override** View OnCreateView**(**LayoutInflater inflater**,** ViewGroup container**,** Bundle savedInstanceState**)**

**{**

**base.**OnCreateView**(**inflater**,** container**,** savedInstanceState**);**

var view **=** inflater**.**Inflate**(**Resource**.**Layout**.**AllLayout**,** container**,** **false);**

var sampleTextView **=** view**.**FindViewById**<**TextView**>(**Resource**.**Id**.**textView1**);**

sampleTextView**.**Text **=** "List of Regions"**;**

var myActivity **=** **(**MainActivity**)this.**Activity**;**

myActivity**.**clearList**();**

myActivity**.**ListAdapter **=** **new** ArrayAdapter**<**String**>((**MainActivity**)this.**Activity**,** Android**.**Resource**.**Layout**.**SimpleListItem1**,** regionArray**);**

checkMode**.**mode **=** 1**;**

myActivity**.**ListView**.**ItemClick **+=** myActivity**.**ListView\_ItemClick**;**

**return** view**;**

**}**

**public** void ListView\_ItemClick1**(object** sender**,** AdapterView**.**ItemClickEventArgs e**)**

**{**

Intent intent **=** **new** Intent**(this.**Activity**,** **typeof(**RegionActivity**));**

var pointer **=** e**.**Position**;**

fromRegionFrag**.**regionPointer **=** pointer**;**

StartActivity**(**intent**);**

**}**

**}**

**public** class AllItems **:** ListFragment

**{**

**public** **override** void OnCreate**(**Bundle savedInstanceState**)**

**{**

**base.**OnCreate**(**savedInstanceState**);**

// Create your fragment here

**}**

**public** **override** View OnCreateView**(**LayoutInflater inflater**,** ViewGroup container**,** Bundle savedInstanceState**)**

**{**

**base.**OnCreateView**(**inflater**,** container**,** savedInstanceState**);**

var view **=** inflater**.**Inflate**(**Resource**.**Layout**.**AllLayout**,** container**,** **false);**

var sampleTextView **=** view**.**FindViewById**<**TextView**>(**Resource**.**Id**.**textView1**);**

checkMode**.**mode **=** 0**;**

var myActivity **=** **(**MainActivity**)this.**Activity**;**

myActivity**.**clearTextLoc**();**

sampleTextView**.**Text **=** "List of all beaches"**;**

ListAdapter **=** **null;**

MyClass myclass **=** **new** MyClass**();**

myclass**.**readItems**();**

myActivity**.**RunOnUiThread**(()** **=>**

**{**

myActivity**.**ListAdapter **=** **new** myCustomAdapter**(**myActivity**,** FromAllItemClass**.**allItems**);**

**});**

myActivity**.**ListView**.**ItemClick **+=** myActivity**.**ListView\_ItemClick**;**

**return** view**;**

**}**

**}**

**public** class SearchMe **:** Fragment

**{**

**public** **override** void OnCreate**(**Bundle savedInstanceState**)**

**{**

**base.**OnCreate**(**savedInstanceState**);**

// Create your fragment here

**}**

**public** **override** View OnCreateView**(**LayoutInflater inflater**,** ViewGroup container**,** Bundle savedInstanceState**)**

**{**

**base.**OnCreateView**(**inflater**,** container**,** savedInstanceState**);**

var view **=** inflater**.**Inflate**(**Resource**.**Layout**.**SearchLayout**,** container**,** **false);**

AutoCompleteTextView textView **=** view**.**FindViewById**<**AutoCompleteTextView**>(**Resource**.**Id**.**autocomplete\_country**);**

var adapter **=** **new** ArrayAdapter**<**String**>(this.**Activity**,** Resource**.**Layout**.**list\_item**,** toNear**.**mynames**);**

checkMode**.**mode **=** 0**;**

textView**.**Adapter **=** adapter**;**

var myActivity **=** **(**MainActivity**)this.**Activity**;**

myActivity**.**clearList**();**

textView**.**ItemClick **+=** TextView\_ItemClick**;**

**return** view**;**

**}**

**public** void TextView\_ItemClick**(object** sender**,** AdapterView**.**ItemClickEventArgs e**)**

**{**

var activity2 **=** **new** Intent**(this.**Activity**,** **typeof(**ItemContext**));**

AutoCompleteTextView autoText **=** **(**AutoCompleteTextView**)**sender**;**

var myText **=** autoText**.**Text**;**

int index **=** Array**.**IndexOf**(**toNear**.**mynames**,** myText**);**

var beachName **=** toNear**.**mynames**[**index**];**

var beachId **=** toNear**.**myids**[**index**];**

SharedObjects**.**mainName **=** beachName**;**

SharedObjects**.**mainId **=** beachId**;**

Intent intent **=** **new** Intent**(this.**Activity**,** **typeof(**ItemContext**)).**SetFlags**(**ActivityFlags**.**ReorderToFront**);**

StartActivity**(**intent**);**

**}**

**}**

Αρχείο Κώδικα 54 App7.Droid/ Fragmament1.cs

### App7.Droid/ Fragmen2.cs

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** Android**.**App**;**

**using** Android**.**Content**;**

**using** Android**.**OS**;**

**using** Android**.**Runtime**;**

**using** Android**.**Util**;**

**using** Android**.**Views**;**

**using** Android**.**Widget**;**

**using** Macaw**.**UIComponents**;**

**using** Android**.**Graphics**;**

**using** RestSharp**;**

**using** Android**.**Gms**.**Maps**;**

**using** Newtonsoft**.**Json**.**Linq**;**

**using** Newtonsoft**.**Json**;**

**using** Android**.**Gms**.**Maps**.**Model**;**

**using** Android**.**Locations**;**

**namespace** App7**.**Droid

**{**

**public** class myMap **:** Fragment**,** IOnMapReadyCallback

**{**

LatLng location**;**

CameraUpdate cameraUpdate**;**

MapFragment mapFragment**;**

**public** **override** void OnCreate**(**Bundle savedInstanceState**)**

**{**

**base.**OnCreate**(**savedInstanceState**);**

**}**

**public** **override** View OnCreateView**(**LayoutInflater inflater**,** ViewGroup container**,** Bundle savedInstanceState**)**

**{**

var view **=** inflater**.**Inflate**(**Resource**.**Layout**.**basic**,** container**,** **false);**

InitMap**();**

**return** view**;**

**}**

**public** **override** void OnPause**()**

**{**

**base.**OnPause**();**

**if** **(**mapFragment **!=** **null)**

**{**

FragmentManager fragmentManager **=** FragmentManager**;**

fragmentManager**.**BeginTransaction**().**Remove**(**mapFragment**).**Commit**();**

**}**

**}**

**public** void OnMapReady**(**GoogleMap googleMap**)**

**{**

googleMap**.**MapType **=** GoogleMap**.**MapTypeNormal**;**

googleMap**.**AddMarker**(new** MarkerOptions**()**

**.**SetPosition**(**location**)**

**.**SetTitle**(**SharedObjects**.**mainName**));**

googleMap**.**MoveCamera**(**cameraUpdate**);**

**}**

**public** void InitMap**()** **{**

var myActivity **=** **(**ItemContext**)this.**Activity**;**

mapFragment **=** **(**MapFragment**)**FragmentManager**.**FindFragmentById**(**Resource**.**Id**.**map**);**

mapFragment**.**GetMapAsync**(this);**

var mylat **=** double**.**Parse**(**helloMap**.**helloLat**);**

var mylong **=** double**.**Parse**(**helloMap**.**helloLng**);**

var loc **=** **new** LatLng**(**mylat**,** mylong**);**

location **=** loc**;**

CameraPosition**.**Builder builder **=** CameraPosition**.**InvokeBuilder**();**

builder**.**Target**(**location**);**

builder**.**Zoom**(**16**);**

builder**.**Tilt**(**30**);**

CameraPosition cameraPosition **=** builder**.**Build**();**

var mycamera **=** CameraUpdateFactory**.**NewCameraPosition**(**cameraPosition**);**

cameraUpdate **=** mycamera**;**

**}**

**}**

**public** class showPhotos **:** Fragment

**{**

**public** static MultiImageView image**;**

**public** **override** void OnCreate**(**Bundle savedInstanceState**)**

**{**

**base.**OnCreate**(**savedInstanceState**);**

// Create your fragment here

**}**

**public** **override** View OnCreateView**(**LayoutInflater inflater**,** ViewGroup container**,** Bundle savedInstanceState**)**

**{**

var view **=** inflater**.**Inflate**(**Resource**.**Layout**.**ph**,** container**,** **false);**

image **=** view**.**FindViewById**<**MultiImageView**>(**Resource**.**Id**.**imageView1**);**

var myActivity **=** **(**ItemContext**)this.**Activity**;**

MyClass mm **=** **new** MyClass**();**

var myli **=** mm**.**getPhotos**(**SharedObjects**.**mainId**);**

photocall**(**myli**);**

**return** view**;**

**}**

//public void getPhotos(int idd) {

// var url = "http://192.168.17.1:5000/getPhotos/" + idd.ToString();

//var client = new RestClient(url);

//var request = new RestRequest(Method.GET);

//client.ExecuteAsync<PhObject>(request, response =>

//{

// var ph = response.Data.Items;

//photocall(response.Data.Items);

// });

//}

**public** void photocall**(**List**<**photoItem**>** li**)** **{**

var photoarray **=** li**.**Select**(**item **=>** item**.**photo**).**ToArray**();**

image**.**LoadImageList**(**photoarray**);**

image**.**SliderSelectedIcon **=** BitmapFactory**.**DecodeResource**(**Resources**,** Resource**.**Drawable**.**slider\_blt\_grn**);**

image**.**SliderUnselectedIcon **=** BitmapFactory**.**DecodeResource**(**Resources**,** Resource**.**Drawable**.**slider\_blt\_trans**);**

image**.**SetSliderIconDimensions**(**50**,** 50**);**

image**.**DownloadedImageSampleSize **=** 1**;**

var myActivity **=** **(**ItemContext**)this.**Activity**;**

image**.**ImagesLoaded **+=** **(**sender**,** e**)** **=>**

**{** // Loads the first image in the list

myActivity**.**RunOnUiThread**(**image**.**LoadImage**);**

**};**

**}**

**}**

var mycamera **=** CameraUpdateFactory**.**NewCameraPosition**(**cameraPosition**);**

cameraUpdate **=** mycamera**;**

**}**

**}**

**public** class showPhotos **:** Fragment

**{**

**public** static MultiImageView image**;**

**public** **override** void OnCreate**(**Bundle savedInstanceState**)**

**{**

**base.**OnCreate**(**savedInstanceState**);**

**}**

**public** **override** View OnCreateView**(**LayoutInflater inflater**,** ViewGroup container**,** Bundle savedInstanceState**)**

**{**

var view **=** inflater**.**Inflate**(**Resource**.**Layout**.**ph**,** container**,** **false);**

image **=** view**.**FindViewById**<**MultiImageView**>(**Resource**.**Id**.**imageView1**);**

var myActivity **=** **(**ItemContext**)this.**Activity**;**

MyClass mm **=** **new** MyClass**();**

var myli **=** mm**.**getPhotos**(**SharedObjects**.**mainId**);**

photocall**(**myli**);**

**return** view**;**

**}**

**public** void photocall**(**List**<**photoItem**>** li**)** **{**

var photoarray **=** li**.**Select**(**item **=>** item**.**photo**).**ToArray**();**

image**.**LoadImageList**(**photoarray**);**

image**.**SliderSelectedIcon **=** BitmapFactory**.**DecodeResource**(**Resources**,** Resource**.**Drawable**.**slider\_blt\_grn**);**

image**.**SliderUnselectedIcon **=** BitmapFactory**.**DecodeResource**(**Resources**,** Resource**.**Drawable**.**slider\_blt\_trans**);**

image**.**SetSliderIconDimensions**(**50**,** 50**);**

image**.**DownloadedImageSampleSize **=** 1**;**

var myActivity **=** **(**ItemContext**)this.**Activity**;**

image**.**ImagesLoaded **+=** **(**sender**,** e**)** **=>**

**{** // Loads the first image in the list

myActivity**.**RunOnUiThread**(**image**.**LoadImage**);**

**};**

**}**

**}**

**public** class beachDesc **:** ListFragment

**{**

TextView text**;**

TextView adText**;**

TextView namest**;**

**public** **override** void OnCreate**(**Bundle savedInstanceState**)**

**{**

**base.**OnCreate**(**savedInstanceState**);**

// Create your fragment here

**}**

**public** **override** View OnCreateView**(**LayoutInflater inflater**,** ViewGroup container**,** Bundle savedInstanceState**)**

**{**

**base.**OnCreateView**(**inflater**,** container**,** savedInstanceState**);**

var view **=** inflater**.**Inflate**(**Resource**.**Layout**.**desc**,** container**,** **false);**

namest **=** view**.**FindViewById**<**TextView**>(**Resource**.**Id**.**textView1**);**

adText **=** view**.**FindViewById**<**TextView**>(**Resource**.**Id**.**sss**);**

text **=** view**.**FindViewById**<**TextView**>(**Resource**.**Id**.**message\_scroll**);**

text**.**MovementMethod **=** **new** Android**.**Text**.**Method**.**ScrollingMovementMethod**();**

MyClass mmm **=** **new** MyClass**();**

var mmml **=** mmm**.**readDesc**(**SharedObjects**.**mainId**);**

desc**(**mmml**);**

var jsm **=** mmm**.**showAddress**(**SharedObjects**.**mainId**);**

getAd**(**jsm**);**

var mynames **=** mmm**.**getAlter**(**SharedObjects**.**mainId**);**

getNames**(**mynames**);**

**return** view**;**

**}**

**public** void desc**(**List**<**dItem**>** li**)**

**{**

var darray **=** li**.**Select**(**item **=>** item**.**description**).**ToArray**();**

var dsource **=** li**.**Select**(**item **=>** item**.**source**).**ToArray**();**

var perigrafh **=** ""**;**

var comments **=** ""**;**

var myActivity **=** **(**ItemContext**)this.**Activity**;**

**for** **(**var i **=** 0**;** i **<** darray**.**Length**;** i**++)**

**{**

**if** **(**dsource**[**i**]** **==** "Foursquare"**)**

**{**

comments **=** comments **+** "\n" **+** "\n" **+** "\n" **+** darray**[**i**];**

**}**

**else**

**{**

perigrafh **=** perigrafh **+** "\n" **+** "\n" **+** darray**[**i**];**

**}**

**}**

myActivity**.**RunOnUiThread**(()** **=>** **{**

text**.**Text **=** perigrafh**;**

**});**

**}**

**public** void getAd**(**string jsonString**)**

**{**

var jsonItem **=** JObject**.**Parse**(**jsonString**);**

var myAddress **=** jsonItem**[**"Items"**][**0**][**"address"**].**ToString**();**

var myActivity **=** **(**ItemContext**)this.**Activity**;**

myActivity**.**RunOnUiThread**(()** **=>** **{**

adText**.**Text **=** myAddress**;**

**});**

**}**

**public** void getNames**(**List**<**nItem**>** d**)**

**{**

var namesArray **=** d**.**Select**(**item **=>** item**.**Names**).**ToArray**();**

var myActivity **=** **(**ItemContext**)this.**Activity**;**

var myAdapter **=** **new** ArrayAdapter**<**String**>(**myActivity**,** Android**.**Resource**.**Layout**.**SimpleListItem1**,** namesArray**);**

myActivity**.**RunOnUiThread**(()** **=>**

**{**

namest**.**Text **=** "Alternate Names"**;**

ListAdapter **=** myAdapter**;**

**});**

**}**

**}**

**public** class beachComments **:** Fragment

**{**

TextView comments**;**

**public** **override** void OnCreate**(**Bundle savedInstanceState**)**

**{**

**base.**OnCreate**(**savedInstanceState**);**

// Create your fragment here

**}**

**public** **override** View OnCreateView**(**LayoutInflater inflater**,** ViewGroup container**,** Bundle savedInstanceState**)**

**{**

**base.**OnCreateView**(**inflater**,** container**,** savedInstanceState**);**

var view **=** inflater**.**Inflate**(**Resource**.**Layout**.**beachComments**,** container**,** **false);**

comments **=** view**.**FindViewById**<**TextView**>(**Resource**.**Id**.**me\_scroll**);**

comments**.**MovementMethod **=** **new** Android**.**Text**.**Method**.**ScrollingMovementMethod**();**

MyClass mn **=** **new** MyClass**();**

beachComm**(**mn**.**readComments**(**SharedObjects**.**mainId**));**

**return** view**;**

**}**

Αρχείο Κώδικα 55 App7.Droid/ Fragment2.cs

### App7.Droid/ RegionActivity.cs

Αρχείο Κώδικα 56 App7.Droid/ RegionActivity.cs

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** RestSharp**;**

**using** Android**.**App**;**

**using** Android**.**Content**;**

**using** Android**.**OS**;**

**using** Android**.**Runtime**;**

**using** Android**.**Views**;**

**using** Android**.**Widget**;**

**namespace** App7**.**Droid

**{**

**[**Activity**(**Label **=** "App7.Droid"**,** Icon **=** "@drawable/icon"**)]**

**public** class RegionActivity **:** ListActivity

**{**

**public** static string**[]** regAr**;**

**public** static int**[]** regId**;**

**protected** **override** void OnCreate**(**Bundle savedInstanceState**)**

**{**

**base.**OnCreate**(**savedInstanceState**);**

SetContentView**(**Resource**.**Layout**.**region**);**

MyClass my **=** **new** MyClass**();**

my**.**getRegionBeaches**();**

RunOnUiThread**(()** **=>**

**{**

ListAdapter **=** **new** RegionAdapter**(this,** FromAllItemClass**.**regionItems**);**

**});**

ListView**.**ItemClick **+=** ListView\_ItemClick**;**

**}**

**private** void ListView\_ItemClick**(object** sender**,** AdapterView**.**ItemClickEventArgs e**)**

**{**

var beachName **=** regAr**[**e**.**Position**];**

var beachId **=** regId**[**e**.**Position**];**

SharedObjects**.**mainName **=** beachName**;**

SharedObjects**.**mainId **=** beachId**;**

var intent **=** **new** Intent**(this,** **typeof(**ItemContext**))**

**.**SetFlags**(**ActivityFlags**.**ReorderToFront**);**

StartActivity**(**intent**);**

**}**

**}**

**public** class fromRegionFrag

**{**

**public** static int regionPointer**;**

**}**

**}**

### App7.Droid/ ShowMap.cs

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** RestSharp**;**

**using** Android**.**App**;**

**using** Android**.**Content**;**

**using** Android**.**OS**;**

**using** Android**.**Runtime**;**

**using** Android**.**Views**;**

**using** Android**.**Widget**;**

**using** Android**.**Support**.**V4**.**App**;**

**using** Android**.**Gms**.**Maps**;**

**using** Android**.**Gms**.**Maps**.**Model**;**

**namespace** App7**.**Droid

**{**

**[**Activity**(**Label **=** "App7.Droid"**,** Icon **=** "@drawable/icon"**)]**

**public** class ShowMap **:** FragmentActivity**,** IOnMapReadyCallback

**{**

CameraUpdate cu**;**

LatLng location**;**

LatLngBounds bounds**;**

**protected** **override** void OnCreate**(**Bundle savedInstanceState**)**

**{**

**base.**OnCreate**(**savedInstanceState**);**

SetContentView**(**Resource**.**Layout**.**showMap**);**

var mapFragment **=(**SupportMapFragment**)**SupportFragmentManager**.**FindFragmentById**(**Resource**.**Id**.**map**);**

mapFragment**.**GetMapAsync**(this);**

LatLngBounds**.**Builder b **=** **new** LatLngBounds**.**Builder**();**

**for** **(**var j**=**0**;** j**<** helloMap**.**hellolats**.**Length**;** j**++)**

**{**

b**.**Include**(new** LatLng**(**helloMap**.**hellolats**[**j**],** helloMap**.**hellolons**[**j**]));**

**}**

bounds **=** b**.**Build**();**

cu **=** CameraUpdateFactory**.**NewLatLngBounds**(**bounds**,** 50**,** 50**,** 10**);**

**}**

**public** void OnMapReady**(**GoogleMap googleMap**)**

**{**

googleMap**.**MapType **=** GoogleMap**.**MapTypeNormal**;**

**if** **(**helloMap**.**hellolats**.**Length **==** 0**)**

**{**

googleMap**.**AddMarker**(new** MarkerOptions**()**

**.**SetPosition**(new** LatLng**(**50.897778**,** 3.013333**)));**

**}**

**else**

**{**

**for** **(**var i **=** 0**;** i **<** helloMap**.**hellolats**.**Length**;** i**++)**

**{**

googleMap**.**AddMarker**(new** MarkerOptions**()**

**.**SetPosition**(new** LatLng**(**helloMap**.**hellolats**[**i**],** helloMap**.**hellolons**[**i**]))**

**.**SetTitle**(**helloMap**.**hnames**[**i**]));**

**}**

googleMap**.**AnimateCamera**(**CameraUpdateFactory**.**NewLatLngZoom**(**bounds**.**Center**,** 11**));**

**}**

**}**

**}**

**public** class helloMap **{**

**public** static string helloLat**;**

**public** static string helloLng**;**

**public** static int hellobeachId**;**

**public** static double**[]** hellolats**;**

**public** static double**[]** hellolons**;**

**public** static string**[]** hnames**;**

**}**

**}**

Αρχείο Κώδικα App7.Droid/ ShowMap.cs

### App7.Droid/ Class1.cs

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** Android**.**App**;**

**using** Android**.**Content**;**

**using** Android**.**OS**;**

**using** Android**.**Runtime**;**

**using** Android**.**Views**;**

**using** Android**.**Widget**;**

**using** Square**.**Picasso**;**

**namespace** App7**.**Droid

**{**

**public** class myCustomAdapter **:** BaseAdapter**<**Item**>**

**{**

ImageView image**;**

List**<**Item**>** items**;**

Activity context**;**

**public** myCustomAdapter**(**Activity context**,** List**<**Item**>** items**)** **:** **base()**

**{**

**this.**context **=** context**;**

**this.**items **=** items**;**

**}**

**public** **override** long GetItemId**(**int position**)**

**{**

**return** position**;**

**}**

**public** **override** Item **this[**int position**]**

**{**

get **{** **return** items**[**position**];** **}**

**}**

**public** **override** int Count

**{**

get **{** **return** items**.**Count**;** **}**

**}**

**public** **override** View GetView**(**int position**,** View convertView**,** ViewGroup parent**)**

**{**

var item **=** items**[**position**];**

View view **=** convertView**;**

var idds **=** items**.**Select**(**data **=>** data**.**id**).**ToArray**();**

**for** **(**var i **=** 0**;** i **<** idds**.**Length**;** i**++)**

**{**

//getPhotoForListview(idds[i]);

**}**

var dds **=** items**;**

**if** **(**view **==** **null)** // no view to re-use, create new

view **=** context**.**LayoutInflater**.**Inflate**(**Resource**.**Layout**.**CustomView**,** **null);**

image **=** view**.**FindViewById**<**ImageView**>(**Resource**.**Id**.**Image**);**

//getPhotoForListview(item.id);

view**.**FindViewById**<**TextView**>(**Resource**.**Id**.**Text1**).**Text **=** item**.**name**;**

view**.**FindViewById**<**TextView**>(**Resource**.**Id**.**Text2**).**Text **=** item**.**id**.**ToString**();**

**if** **(**item**.**photo **!=** ""**)**

**{**

Picasso**.**With**(**context**).**Load**(**item**.**photo**).**Into**(**image**);**

**}**

**else**

**{**

image**.**SetImageResource**(**Resource**.**Drawable**.**Icon**);**

**}**

**return** view**;**

**}**

**}**

**}**

Αρχείο Κώδικα 58 App7.Droid/ Class1.cs

### App7.Droid/ CustomAdapter.cs

Αρχείο Κώδικα 59 App7.Droid/ CustomAdapter.cs

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** Android**.**App**;**

**using** Android**.**Content**;**

**using** Android**.**OS**;**

**using** Android**.**Runtime**;**

**using** Android**.**Views**;**

**using** Android**.**Widget**;**

**namespace** App7**.**Droid

**{**

**public** class myCustomAdapter **:** BaseAdapter**<**Item**>**

**{**

ImageView image**;**

List**<**Item**>** items**;**

Activity context**;**

**public** myCustomAdapter**(**Activity context**,** List**<**Item**>** items**)** **:** **base()**

**{**

**this.**context **=** context**;**

**this.**items **=** items**;**

**}**

**public** **override** long GetItemId**(**int position**)**

**{**

**return** position**;**

**}**

**public** **override** Item **this[**int position**]**

**{**

get **{** **return** items**[**position**];** **}**

**}**

**public** **override** int Count

**{**

get **{** **return** items**.**Count**;** **}**

**}**

**public** **override** View GetView**(**int position**,** View convertView**,** ViewGroup parent**)**

**{**

var item **=** items**[**position**];**

View view **=** convertView**;**

var idds **=** items**.**Select**(**data **=>** data**.**id**).**ToArray**();**

**for** **(**var i **=** 0**;** i **<** idds**.**Length**;** i**++)**

**{**

//getPhotoForListview(idds[i]);

**}**

var dds **=** items**;**

**if** **(**view **==** **null)** // no view to re-use, create new

view **=** context**.**LayoutInflater**.**Inflate**(**Resource**.**Layout**.**CustomView**,** **null);**

image **=** view**.**FindViewById**<**ImageView**>(**Resource**.**Id**.**Image**);**

//getPhotoForListview(item.id);

view**.**FindViewById**<**TextView**>(**Resource**.**Id**.**Text1**).**Text **=** item**.**name**;**

view**.**FindViewById**<**TextView**>(**Resource**.**Id**.**Text2**).**Text **=** item**.**id**.**ToString**();**

**if** **(**item**.**photo **!=** ""**)**

**{**

Picasso**.**With**(**context**).**Load**(**item**.**photo**).**Into**(**image**);**

**}**

**else**

**{**

image**.**SetImageResource**(**Resource**.**Drawable**.**Icon**);**

**}**

**return** view**;**

**}**

**}**

**}**

### App7.Droid/ NearAdapter.cs

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** Android**.**App**;**

**using** Android**.**Content**;**

**using** Android**.**OS**;**

**using** Android**.**Runtime**;**

**using** Android**.**Views**;**

**using** Android**.**Widget**;**

**using** Square**.**Picasso**;**

**namespace** App7**.**Droid

**{**

**public** class NearAdapter**:** BaseAdapter**<**myItem**>**

**{**

ImageView image**;**

List**<**myItem**>** items**;**

Activity context**;**

**public** **override** int Count

**{**

get **{** **return** items**.**Count**;** **}**

**}**

**public** **override** myItem **this[**int position**]**

**{**

get **{** **return** items**[**position**];** **}**

**}**

**public** NearAdapter**(**Activity context**,** List**<**myItem**>** items**)** **:** **base()**

**{**

**this.**context **=** context**;**

**this.**items **=** items**;**

**}**

**public** **override** long GetItemId**(**int position**)**

**{**

**return** position**;**

**}**

**public** **override** View GetView**(**int position**,** View convertView**,** ViewGroup parent**)**

**{**

var item **=** items**[**position**];**

View view **=** convertView**;**

var idds **=** items**.**Select**(**data **=>** data**.**id**).**ToArray**();**

**for** **(**var i **=** 0**;** i **<** idds**.**Length**;** i**++)**

{

//getPhotoForListview(idds[i]);

}

var dds = items;

if (view == null) // no view to re-use, create new

view = context.LayoutInflater.Inflate(Resource.Layout.CustomView, null);

image = view.FindViewById<ImageView>(Resource.Id.Image);

//getPhotoForListview(item.id);

view.FindViewById<TextView>(Resource.Id.Text1).Text = item.name;

view.FindViewById<TextView>(Resource.Id.Text2).Text = item.id.ToString();

if (item.photo != "")

{

Picasso.With(context).Load(item.photo).Into(image);

}

else

{

image.SetImageResource(Resource.Drawable.Icon);

}

return view;

}

}

}

Αρχείο Κώδικα App7.Droid/ NearAdapter.cs

### App7.Droid/ RegionAdapter.cs

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** Android**.**App**;**

**using** Android**.**Content**;**

**using** Android**.**OS**;**

**using** Android**.**Runtime**;**

**using** Android**.**Views**;**

**using** Android**.**Widget**;**

**using** Square**.**Picasso**;**

**namespace** App7**.**Droid

**{**

**public** class RegionAdapter **:** BaseAdapter**<**tItem**>**

**{**

List**<**tItem**>** items**;**

ImageView image**;**

Activity context**;**

**public** RegionAdapter**(**Activity context**,** List**<**tItem**>** items**)** **:** **base()**

**{**

**this.**context **=** context**;**

**this.**items **=** items**;**

**}**

**public** **override** tItem **this[**int position**]**

**{**

get **{** **return** items**[**position**];** **}**

**}**

**public** **override** int Count

**{**

get **{** **return** items**.**Count**;** **}**

**}**

**public** **override** long GetItemId**(**int position**)**

**{**

**return** position**;**

**}**

**public** **override** View GetView**(**int position**,** View convertView**,** ViewGroup parent**)**

**{**

var item **=** items**[**position**];**

View view **=** convertView**;**

var idds **=** items**.**Select**(**data **=>** data**.**id**).**ToArray**();**

**for** **(**var i **=** 0**;** i **<** idds**.**Length**;** i**++)**

**{**

//getPhotoForListview(idds[i]);

}

var dds = items;

if (view == null) // no view to re-use, create new

view = context.LayoutInflater.Inflate(Resource.Layout.CustomView, null);

image = view.FindViewById<ImageView>(Resource.Id.Image);

//getPhotoForListview(item.id);

view.FindViewById<TextView>(Resource.Id.Text1).Text = item.name;

view.FindViewById<TextView>(Resource.Id.Text2).Text = item.id.ToString();

if (item.photo != "")

{

Picasso.With(context).Load(item.photo).Into(image);

}

else

{

image.SetImageResource(Resource.Drawable.Icon);

}

return view;

}

}

}

Αρχείο Κώδικα 61 App7.Droid/ RegionAdapter.cs

### Patmosplus.py

# !/usr/bin/python

# -\*- coding: utf-8 -\*-

**import** MySQLdb

**import** sys

**from** geopy**.**exc **import** GeocoderTimedOut

sys**.**path**.**insert**(**0**,** 'C:\Users\home\greekbeachesapp'**)**

**from** sunthesis **import** elegxos\_paralias

**from** scrapinghub **import** Connection

**from** geopy**.**geocoders **import** Nominatim

**import** time

conn **=** Connection**(**'57c54e0d05fb44439a9fc0887fb6ab3e'**)**

**print** conn

**print** conn**.**project\_ids**()**

project **=** conn**[**77645**]**

jobs **=** project**.**jobs**(**state**=**'finished'**)**

jobs\_id **=** **[**x**.**id **for** x **in** jobs**]**

**print** jobs\_id

job **=** project**.**job**(**u'77645/1/1'**)**

servername **=** "localhost"

username **=** "root"

password **=** ""

dbname **=** "database"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cursor **=** db**.**cursor**()**

**for** item **in** job**.**items**():**

**if** u'latlon' **not** **in** item**.**keys**():**

**continue**

**else:**

**print** str**(**item**[**u'latlon'**][**0**])**

inp **=** str**(**item**[**u'latlon'**][**0**])**

startlat **=** inp**.**find**(**'@'**)**

endlat **=** inp**.**find**(**','**)**

lat **=** float**(**inp**[**startlat**+**1**:**endlat**])**

**print** lat

endlng **=** inp**.**find**(**'z'**)**

lng **=** float**(**inp**[**endlat**+**1**:**endlng**].**replace**(**',13'**,**''**))**

**print** lng

f **=** **[]**

**if** u'variants' **in** item**.**keys**():**

**print** item**[**u'variants'**]**

**for** data **in** item**[**u'variants'**]:**

**for** ph **in** data**[**u'photos'**]:**

f**.**append**(**ph**)**

**else:**

f **=** **[]**

#r = u"Πάτμος, Δωδεκάνησα"

geolocator **=** Nominatim**()**

location **=** geolocator**.**reverse**([**lat**,** lng**],** timeout**=**10**)**

ad **=** location**.**address

**if** 'state\_district' **in** location**.**raw**[**'address'**].**keys**():**

state\_district **=** location**.**raw**[**'address'**][**'state\_district'**]**

**else:**

state\_district **=** ' '

r **=** state\_district

source **=** 'patmos-island.com'

**if** elegxos\_paralias**(**item**[**u'name'**][**0**],** lat**,** lng**,** ad**,** item**[**u'description'**],** source**,** f**)** **!=** 0**:**

**print** 'Inserted to beachplus'

**else:**

sql1 **=** """INSERT INTO beach (name,lat,lon,region) VALUES (%s,%s,%s,%s)"""

args1 **=** **(**item**[**u'name'**][**0**],** lat**,** lng**,** r**)**

cursor**.**execute**(**sql1**,** args1**)**

sql3 **=** """SELECT MAX(id) FROM beach"""

cursor**.**execute**(**sql3**)**

result **=** cursor**.**fetchone**()**

sql2 **=** """INSERT INTO beachplus(address ,source, beach\_id, alternate) VALUES (%s, %s, %s, %s)"""

args2 **=** **(**ad**,**source**,** result**,** item**[**u'name'**][**0**],** **)**

cursor**.**execute**(**sql2**,** args2**)**

sql4 **=** """INSERT INTO description(beach\_id, text, source) VALUES(%s, %s, %s)"""

args4 **=** **(**result**,** item**[**u'description'**],** source**)**

cursor**.**execute**(**sql4**,** args4**)**

sqlp **=** """INSERT INTO photos(beach\_id, ph) VALUES(%s, %s)"""

**for** string **in** f**:**

cursor**.**execute**(**sqlp**,** **(**result**,** string**))**

db**.**commit**()**

**print** 'end'

Αρχείο Κώδικα patmosplus.py

### skiathosplus.py

# !/usr/bin/python

# -\*- coding: utf-8 -\*-

**import** MySQLdb

**import** sys

**from** geopy**.**exc **import** GeocoderTimedOut

sys**.**path**.**insert**(**0**,** 'C:\Users\home\greekbeachesapp'**)**

**from** sunthesis **import** elegxos\_paralias

**from** scrapinghub **import** Connection

**from** geopy**.**geocoders **import** Nominatim

**import** time

conn **=** Connection**(**'57c54e0d05fb44439a9fc0887fb6ab3e'**)**

**print** conn

**print** conn**.**project\_ids**()**

project **=** conn**[**77659**]**

jobs **=** project**.**jobs**(**state**=**'finished'**)**

jobs\_id **=** **[**x**.**id **for** x **in** jobs**]**

**print** jobs\_id

job **=** project**.**job**(**u'77659/1/1'**)**

servername **=** "localhost"

username **=** "root"

password **=** ""

dbname **=** "database"

source **=** 'destinationskiathos.com'

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cursor **=** db**.**cursor**()**

**for** item **in** job**.**items**():**

**if** u'latlon' **not** **in** item**.**keys**():**

**continue**

**else:**

**if** u'description' **not** **in** item**.**keys**():**

item**[**u'description'**]** **=** u"Δεν υπάρχουν διαθέσιμες πληροφορίες (ακόμη!)"

**print** str**(**item**[**u'latlon'**][**0**])**

inp **=** str**(**item**[**u'latlon'**][**0**])**

startlat **=** inp**.**find**(**'@'**)**

endlat **=** inp**.**find**(**','**)**

lat **=** float**(**inp**[**startlat**+**1**:**endlat**])**

**print** lat

endlng **=** inp**.**find**(**'z'**)**

lng **=** float**(**inp**[**endlat**+**1**:**endlng**].**replace**(**',15'**,**''**))**

**print** lng

#r = u"Σκιάθος, Σποράδες"

geolocator **=** Nominatim**()**

location **=** geolocator**.**reverse**([**lat**,** lng**],** timeout**=**10**)**

ad **=** location**.**address

**if** 'state\_district' **in** location**.**raw**[**'address'**].**keys**():**

state\_district **=** location**.**raw**[**'address'**][**'state\_district'**]**

**else:**

state\_district **=** ' '

r **=** state\_district

time**.**sleep**(**1**)**

**if** elegxos\_paralias**(**item**[**u'name'**][**0**],** lat**,** lng**,** ad**,** item**[**u'description'**],** source**,** item**[**u'photo'**])** **!=** 0**:**

**print** 'Inserted to beachplus'

**else:**

sql1 **=** """INSERT INTO beach (name,lat,lon,region) VALUES (%s,%s,%s,%s)"""

args1 **=** **(**item**[**u'name'**][**0**],** lat**,** lng**,** r**)**

cursor**.**execute**(**sql1**,** args1**)**

sql3 **=** """SELECT MAX(id) FROM beach"""

cursor**.**execute**(**sql3**)**

result **=** cursor**.**fetchone**()**

sql2 **=** """INSERT INTO beachplus(address ,source, beach\_id, alternate) VALUES (%s, %s, %s, %s)"""

args2 **=** **(**ad**,**source**,** result**,** item**[**u'name'**][**0**])**

cursor**.**execute**(**sql2**,** args2**)**

sql4 **=** """INSERT INTO description(beach\_id, text, source) VALUES(%s, %s, %s)"""

args4 **=** **(**result**,** item**[**u'description'**],** source**)**

cursor**.**execute**(**sql4**,** args4**)**

sqlp **=** """INSERT INTO photos(beach\_id, ph) VALUES(%s, %s)"""

cursor**.**execute**(**sqlp**,** **(**result**,** item**[**u'photo'**]))**

db**.**commit**()**

**print** 'end'

Αρχείο Κώδικα skiathosplus.py

### zanteplus.py

Αρχείο Κώδικα zanteplus.py

# !/usr/bin/python

# -\*- coding: utf-8 -\*-

**from** scrapinghub **import** Connection

**from** geopy**.**geocoders **import** Nominatim

**import** time

conn **=** Connection**(**'57c54e0d05fb44439a9fc0887fb6ab3e'**)**

**print** conn

**print** conn**.**project\_ids**()**

project **=** conn**[**77608**]**

jobs **=** project**.**jobs**(**state**=**'finished'**)**

jobs\_id **=** **[**x**.**id **for** x **in** jobs**]**

**print** jobs\_id

job **=** project**.**job**(**u'77608/1/1'**)**

source **=** 'e-zakynthos.com'

servername **=** "localhost"

username **=** "root"

password **=** ""

dbname **=** "database"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cursor **=** db**.**cursor**()**

z **=** u'\u0396\u03ac\u03ba\u03c5\u03bd\u03b8\u03bf\u03c2'

**for** item **in** job**.**items**():**

**if** u'latlon' **not** **in** item**.**keys**():**

**continue**

**else:**

item**[**u'name'**]** **=** ''**.**join**(**item**[**u'name'**]).**replace**(**z**,**''**)**

item**[**u'name'**]** **=** item**[**u'name'**].**replace**(**'e-'**,** ''**)**

item**[**u'name'**]** **=** item**[**u'name'**].**strip**(**" "**)**

inp **=** str**(**item**[**u'latlon'**][**0**])**

startlat **=** inp**.**find**(**'@'**)**

endlat **=** inp**.**find**(**','**)**

lat **=** float**(**inp**[**startlat**+**1**:**endlat**])**

**print** lat

endlng **=** inp**.**find**(**'z'**)**

**if** '15z' **in** str**(**item**[**u'latlon'**][**0**]):**

lng **=** inp**[**endlat**+**1**:**endlng**].**replace**(**',15'**,**''**)**

**elif** '16z' **in** str**(**item**[**u'latlon'**][**0**]):**

lng **=** inp**[**endlat**+**1**:**endlng**].**replace**(**',16'**,**''**)**

**elif** '14z' **in** str**(**item**[**u'latlon'**][**0**]):**

lng **=** inp**[**endlat**+**1**:**endlng**].**replace**(**',14'**,**''**)**

**else:**

lng **=** inp**[**endlat**+**1**:**endlng**].**replace**(**',13'**,**''**)**

lng **=** float**(**lng**)**

**print** lng

f **=** **[]**

**if** u'variants' **in** item**.**keys**():**

**print** item**[**u'variants'**]**

**for** data **in** item**[**u'variants'**]:**

**for** ph **in** data**[**u'photos'**]:**

f**.**append**(**ph**)**

**else:**

f **=** **[]**

geolocator **=** Nominatim**()**

location **=** geolocator**.**reverse**([**lat**,** lng**],** timeout**=**10**)**

ad **=** location**.**address

**if** 'state\_district' **in** location**.**raw**[**'address'**].**keys**():**

state\_district **=** location**.**raw**[**'address'**][**'state\_district'**]**

**else:**

state\_district **=** ' '

r **=** state\_district

time**.**sleep**(**2**)**

**if** elegxos\_paralias**(**item**[**u'name'**],** lat**,** lng**,** ad**,** item**[**u'description'**],** source**,** f**)** **!=** 0**:**

**print** 'Inserted to beachplus'

**else:**

sql1 **=** """INSERT INTO beach (name,lat,lon,region) VALUES (%s,%s,%s,%s)"""

args1 **=** **(**item**[**u'name'**],** lat**,** lng**,** r**)**

cursor**.**execute**(**sql1**,** args1**)**

sql3 **=** """SELECT MAX(id) FROM beach"""

cursor**.**execute**(**sql3**)**

result **=** cursor**.**fetchone**()**

sql2 **=** """INSERT INTO beachplus(address ,source, beach\_id, alternate) VALUES (%s, %s, %s, %s)"""

args2 **=** **(**ad**,**source**,** result**,** item**[**u'name'**])**

cursor**.**execute**(**sql2**,** args2**)**

sql4 **=** """INSERT INTO description(beach\_id, text, source) VALUES(%s, %s, %s)"""

args4 **=** **(**result**,** item**[**u'description'**],** source**)**

cursor**.**execute**(**sql4**,** args4**)**

sqlp **=** """INSERT INTO photos(beach\_id, ph) VALUES(%s, %s)"""

**for** string **in** f**:**

cursor**.**execute**(**sqlp**,** **(**result**,** string**))**

db**.**commit**()**

### thasosplus.py

# !/usr/bin/python

# -\*- coding: utf-8 -\*-

conn **=** Connection**(**'42e697f0d35348e9b6c70d6e74dcbc93'**)**

**print** conn

**print** conn**.**project\_ids**()**

project **=** conn**[**78127**]**

jobs **=** project**.**jobs**(**state**=**'finished'**)**

jobs\_id **=** **[**x**.**id **for** x **in** jobs**]**

**print** jobs\_id

job **=** project**.**job**(**u'78127/1/1'**)**

source **=** 'thassos-view.com'

servername **=** "localhost"

username **=** "root"

password **=** ""

dbname **=** "database"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cursor **=** db**.**cursor**()**

**for** item **in** job**.**items**():**

**if** u'latlon' **not** **in** item**.**keys**():**

**continue**

**else:**

item**[**u'name'**]** **=** item**[**u'name'**][**0**].**replace**(**u', \u03a0\u03b1\u03c1\u03b1\u03bb\u03af\u03b1 \u0398\u03ac\u03c3\u03bf\u03c2'**,** ''**)**

item**[**u'name'**]** **=** item**[**u'name'**].**replace**(**u', \u0398\u03ac\u03c3\u03bf\u03c2'**,** ''**)**

**print** str**(**item**[**u'latlon'**][**0**])**

inp **=** str**(**item**[**u'latlon'**][**0**])**

startlat **=** inp**.**find**(**'@'**)**

endlat **=** inp**.**find**(**','**)**

lat **=** float**(**inp**[**startlat**+**1**:**endlat**])**

**print** lat

endlng **=** inp**.**find**(**'z'**)**

**if** '15z' **in** str**(**item**[**u'latlon'**][**0**]):**

lng **=** inp**[**endlat**+**1**:**endlng**].**replace**(**',15'**,**''**)**

**elif** '12z' **in** str**(**item**[**u'latlon'**][**0**]):**

lng **=** inp**[**endlat**+**1**:**endlng**].**replace**(**',12'**,**''**)**

**elif** '17z' **in** str**(**item**[**u'latlon'**][**0**]):**

lng **=** inp**[**endlat**+**1**:**endlng**].**replace**(**',17'**,**''**)**

**else:**

lng **=** inp**[**endlat**+**1**:**endlng**].**replace**(**',16'**,**''**)**

lng **=** float**(**lng**)**

**print** lng

z **=** u'\u03a6\u03c9\u03c4\u03bf\u03b3\u03c1\u03b1\u03c6\u03af\u03b5\u03c2'

item**[**u'description'**]** **=** item**[**u'description'**][**0**].**replace**(**z**,** '@'**)**

item**[**u'description'**]** **=** item**[**u'description'**].**replace**(**'Labels'**,** '#'**)**

thisStart **=** item**[**u'description'**].**find**(**'@'**)**

thisEnd **=** item**[**u'description'**].**find**(**'#'**)**

removed **=** item**[**u'description'**][**thisStart**+**1**:**thisEnd**]**

item**[**u'description'**]** **=** item**[**u'description'**].**replace**(**removed**,** ''**)**

item**[**u'description'**]** **=** item**[**u'description'**].**replace**(**'@'**,** ''**)**

item**[**u'description'**]** **=** item**[**u'description'**].**replace**(**'#'**,** ''**)**

f **=** **[]**

**if** u'variants' **in** item**.**keys**():**

**print** item**[**u'variants'**]**

**for** data **in** item**[**u'variants'**]:**

**for** ph **in** data**[**u'images'**]:**

f**.**append**(**ph**)**

**else:**

f **=** **[]**

#r = u"Θάσος, Βόρειο Αιγαίο"

geolocator **=** Nominatim**()**

location **=** geolocator**.**reverse**([**lat**,** lng**],** timeout**=**10**)**

ad **=** location**.**address

**if** 'state\_district' **in** location**.**raw**[**'address'**].**keys**():**

state\_district **=** location**.**raw**[**'address'**][**'state\_district'**]**

**else:**

state\_district **=** ' '

r **=** state\_district

time**.**sleep**(**2**)**

**if** elegxos\_paralias**(**item**[**u'name'**],** lat**,** lng**,** ad**,** item**[**u'description'**],** source**,** f**)** **!=** 0**:**

**print** 'Inserted to beachplus'

**else:**

sql1 **=** """INSERT INTO beach (name,lat,lon,region) VALUES (%s,%s,%s,%s)"""

args1 **=** **(**item**[**u'name'**],** lat**,** lng**,** r**)**

cursor**.**execute**(**sql1**,** args1**)**

sql3 **=** """SELECT MAX(id) FROM beach"""

cursor**.**execute**(**sql3**)**

result **=** cursor**.**fetchone**()**

sql2 **=** """INSERT INTO beachplus(address ,source, beach\_id, alternate) VALUES (%s, %s, %s, %s)"""

args2 **=** **(**ad**,**source**,** result**,** item**[**u'name'**])**

cursor**.**execute**(**sql2**,** args2**)**

sql4 **=** """INSERT INTO description(beach\_id, text, source) VALUES(%s, %s, %s)"""

args4 **=** **(**result**,** item**[**u'description'**],** source**)**

cursor**.**execute**(**sql4**,** args4**)**

sqlp **=** """INSERT INTO photos(beach\_id, ph) VALUES(%s, %s)"""

**for** string **in** f**:**

cursor**.**execute**(**sqlp**,** **(**result**,** string**))**

db**.**commit**()**

Αρχείο Κώδικα thasosplus.py

### cleanplus.py

Αρχείο Κώδικα cleanplus.py

# !/usr/bin/python

# -\*- coding: utf-8 -\*-

**import** MySQLdb

**import** sys

**from** geopy**.**exc **import** GeocoderTimedOut

sys**.**path**.**insert**(**0**,** 'C:\Users\home\greekbeachesapp'**)**

**from** sunthesis **import** elegxos\_paralias

**from** scrapinghub **import** Connection

**from** geopy**.**geocoders **import** Nominatim

**import** time

conn **=** Connection**(**'57c54e0d05fb44439a9fc0887fb6ab3e'**)**

**print** conn

**print** conn**.**project\_ids**()**

project **=** conn**[**77669**]**

jobs **=** project**.**jobs**(**state**=**'finished'**)**

jobs\_id **=** **[**x**.**id **for** x **in** jobs**]**

**print** jobs\_id

job **=** project**.**job**(**u'77669/1/1'**)**

servername **=** "localhost"

username **=** "root"

password **=** ""

dbname **=** "database"

source **=** 'cleanbeaches.gr'

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cursor **=** db**.**cursor**()**

**for** item **in** job**.**items**():**

**if** item**[**u'url'**]** **==** 'http://www.cleanbeaches.gr/'**:**

**continue**

**else:**

inp **=** str**(**item**[**u'latlon'**][**0**])**

startlat **=** inp**.**find**(**'@'**)**

endlat **=** inp**.**find**(**','**)**

lat **=** float**(**inp**[**startlat**+**1**:**endlat**])**

**print** lat

endlng **=** inp**.**find**(**'z'**)**

lng **=** float**(**inp**[**endlat**+**1**:**endlng**].**replace**(**',16'**,**''**))**

**print** lng

photos **=** **[]**

#r = u"Αττική"

geolocator **=** Nominatim**()**

location **=** geolocator**.**reverse**([**lat**,** lng**],** timeout**=**10**)**

ad **=** location**.**address

**if** 'state\_district' **in** location**.**raw**[**'address'**].**keys**():**

state\_district **=** location**.**raw**[**'address'**][**'state\_district'**]**

**else:**

state\_district **=** ' '

r **=** state\_district

time**.**sleep**(**1**)**

**if** u'sugkrish' **in** item**.**keys**():**

desc **=** item**[**u'katalhlothta'**][**0**]** **+** ' ' **+** item**[**u'metrhseis'**][**0**]** **+** ' ' **+** item**[**u'sugkrish'**][**0**]**

**else:**

desc **=** item**[**u'katalhlothta'**][**0**]** **+** ' ' **+** item**[**u'metrhseis'**][**0**]**

**if** elegxos\_paralias**(**item**[**u'name'**][**0**],** lat**,** lng**,** ad**,** desc**,** source**,** photos**)** **!=** 0**:**

**print** 'Inserted to beachplus'

**else:**

sql1 **=** """INSERT INTO beach (name,lat,lon,region) VALUES (%s,%s,%s,%s)"""

args1 **=** **(**item**[**u'name'**][**0**],** lat**,** lng**,** r**)**

cursor**.**execute**(**sql1**,** args1**)**

sql3 **=** """SELECT MAX(id) FROM beach"""

cursor**.**execute**(**sql3**)**

result **=** cursor**.**fetchone**()**

sql2 **=** """INSERT INTO beachplus(address ,source, beach\_id, alternate) VALUES (%s, %s, %s, %s)"""

args2 **=** **(**ad**,**source**,** result**,** item**[**u'name'**][**0**])**

cursor**.**execute**(**sql2**,** args2**)**

sql4 **=** """INSERT INTO description(beach\_id, text, source) VALUES(%s, %s, %s)"""

args4 **=** **(**result**,** desc**,** source**)**

cursor**.**execute**(**sql4**,** args4**)**

db**.**commit**()**

### run\_me.py

Αρχείο Κώδικα run\_me.py

# !/usr/bin/python

# -\*- coding: utf-8 -\*-

**import** sys

reload**(**sys**)**

sys**.**setdefaultencoding**(**'utf-8'**)**

**import** unicodedata

**from** beach\_search **import** extract

**import** geonames**.**adapters**.**search

**from** django**.**utils**.**encoding **import** smart\_str**,** smart\_unicode

\_USERNAME **=** 'kapsali'

sa **=** geonames**.**adapters**.**search**.**Search**(**\_USERNAME**)**

items **=** **[]**

result **=** sa**.**query**(**''**).**country**(**'gr'**).**max\_rows**(**1000**).**execute**()**

**for** id\_**,** name **in** result**.**get\_flat\_results**():**

item **=** geonames**.**compat**.**make\_unicode**(**" [{1}]"**).**format**(**id\_**,** name**)**

items**.**append**(**item**)**

items **=** **[**unicodedata**.**normalize**(**'NFKD'**,** x**).**encode**(**'latin-1'**,**'ignore'**)** **for** x **in** items**]**

**print** items

**for** item **in** items**:**

**print** item

**print** '\n'

**try:**

extract**(**item**)**

**except:**

sep **=** ','

text **=** ''**.**join**(**item**)**

rest **=** text**.**split**(**sep**,**1**)[**0**]**

**try:**

extract**(**rest**)**

**except:**

**continue**

### ids.py

# !/usr/bin/python

# -\*- coding: utf-8 -\*-

**import** sys

reload**(**sys**)**

sys**.**setdefaultencoding**(**'utf-8'**)**

**import** unicodedata

**from** beach\_search **import** extract

**import** geonames**.**adapters**.**search

**from** django**.**utils**.**encoding **import** smart\_str**,** smart\_unicode

\_USERNAME **=** 'kapsali'

sa **=** geonames**.**adapters**.**search**.**Search**(**\_USERNAME**)**

items **=** **[]**

result **=** sa**.**query**(**''**).**country**(**'gr'**).**max\_rows**(**1000**).**execute**()**

**for** id\_**,** name **in** result**.**get\_flat\_results**():**

item **=** geonames**.**compat**.**make\_unicode**(**" [{1}]"**).**format**(**id\_**,** name**)**

items**.**append**(**item**)**

items **=** **[**unicodedata**.**normalize**(**'NFKD'**,** x**).**encode**(**'latin-1'**,**'ignore'**)** **for** x **in** items**]**

**print** items

**for** item **in** items**:**

**print** item

**print** '\n'

**try:**

extract**(**item**)**

**except:**

sep **=** ','

text **=** ''**.**join**(**item**)**

rest **=** text**.**split**(**sep**,**1**)[**0**]**

**try:**

extract**(**rest**)**

**except:**

**continue**

Αρχείο Κώδικα ids.py

### beach\_search.py

Αρχείο Κώδικα beach\_search.py

**def** extract**(**p\_city**):**

# !/usr/bin/python

# -\*- coding: utf-8 -\*-

**import** MySQLdb

**import** foursquare

**import** json

**import** time

**import** os

**import** sys

**import** codecs**,** locale

**from** tips **import** tip\_details

**import** unicodedata

**import** sys**,** nltk**,** shutil

**import** string

**import** unicodedata

**import** chardet

**from** listing\_ad **import** it\_li

**from** unidecode **import** unidecode

**from** venue\_info **import** get\_venue\_info

**from** photo\_info **import** get\_venue\_photo

**from** check\_entry **import** check\_double\_entry

**from** ids **import** idd

**from** geopy**.**geocoders **import** Nominatim

reload**(**sys**)**

sys**.**setdefaultencoding**(**'utf-8'**)**

**import** sys

sys**.**path**.**insert**(**0**,** 'C:\Users\home\greekbeachesapp'**)**

**from** sunthesis **import** elegxos\_paralias

#import greek\_alphabet

client **=** foursquare**.**Foursquare**(**client\_id**=**'0NKJU5YU3205AZXBGQWLDWRLREZRNBGWPDHYR2PXNFOFOGIY'**,**

client\_secret**=**'DIWIFQPLO0HLBBKOCYLAO1FRK5KA0DDGS1ZTVLN2RBSQST3E'**)**

plaz **=** client**.**venues**.**search**(**params**={**'query'**:** 'Beach'**,** 'near'**:** p\_city**,** 'categoryId'**:** '4bf58dd8d48988d1e2941735'**})**

servername **=** "localhost"

username **=** "root"

password **=** ""

dbname **=** "database"

**for** venues **in** plaz**[**"venues"**]:**

plaz\_id **=** str**(**venues**[**"id"**])**

**print** type**(**plaz\_id**)**

plaz\_name **=** venues**[**"name"**]**

plaz\_lat **=** venues**[**"location"**][**"lat"**]**

**print** plaz\_lat

plaz\_lng **=** venues**[**"location"**][**"lng"**]**

**print** plaz\_lng

**print** "Rating & Url\n"

get\_venue\_info**(**plaz\_id**)**

**print** "Location\n"

plaz\_lat **=** float**(**venues**[**"location"**][**"lat"**])**

**print** plaz\_lat

plaz\_lng **=** float**(**venues**[**"location"**][**"lng"**])**

**print** plaz\_lng

plaz\_cc **=** str**(**venues**[**"location"**][**"cc"**])**

**print** plaz\_cc

**print** type**(**plaz\_cc**)**

ccs **=** **[**'gr'**,**'GR'**]**

ap **=** plaz\_cc **in** ccs

**if** ap **==** **True:**

geolocator **=** Nominatim**()**

location **=** geolocator**.**reverse**([**plaz\_lat**,** plaz\_lng**],** timeout**=**10**)**

**if** 'state\_district' **in** location**.**raw**[**'address'**].**keys**():**

state\_district **=** location**.**raw**[**'address'**][**'state\_district'**]**

**else:**

state\_district **=** ' '

reg **=** state\_district

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cursor **=** db**.**cursor**()**

source **=** "Foursquare"

re **=** idd**(**plaz\_id**)**

**print** re

**if** re **==** **True:**

**print** "this entry exists"

**continue**

**else:**

**if** elegxos\_paralias**(**plaz\_name**,** plaz\_lat**,** plaz\_lng**,** location**.**address**,** plaz\_id**,** 'Foursquare'**,** plaz\_id**)** **!=**0**:**

**print** 'Inserted to beachplus'

sql0 **=** """INSERT INTO tsek(pid) VALUES(%s)"""

args0 **=** **([**plaz\_id**])**

cursor**.**execute**(**sql0**,** args0**)**

db**.**commit**()**

**else:**

sql0 **=** """INSERT INTO tsek(pid) VALUES(%s)"""

args0 **=** **([**plaz\_id**])**

cursor**.**execute**(**sql0**,** args0**)**

db**.**commit**()**

sql1 **=** """INSERT INTO beach(name, lat, lon, region) VALUES(%s, %s, %s, %s) """

region\_b **=** reg

args1 **=** **(**plaz\_name**,** plaz\_lat**,** plaz\_lng**,** region\_b**)**

cursor**.**execute**(**sql1**,** args1**)**

db**.**commit**()**

sql2 **=** """SELECT MAX(id) FROM beach"""

cursor**.**execute**(**sql2**)**

db**.**commit**()**

result **=** cursor**.**fetchone**()**

sql3 **=** """INSERT INTO beachplus(address ,source, beach\_id, alternate) VALUES (%s, %s, %s, %s)"""

args3 **=** **(**location**.**address**,** "Foursquare"**,** result**,** plaz\_name**)**

cursor**.**execute**(**sql3**,** args3**)**

db**.**commit**()**

#print 'Tips\n'

**if** venues**[**"stats"**][**"tipCount"**]** **!=** 0**:**

#tip\_details(venues["id"],plaz\_name,result)

tip **=** client**.**venues**.**tips**(**venues**[**"id"**])**

**for** item **in** tip**[**"tips"**][**"items"**]:**

tip\_text **=** item**[**"text"**]**

sql6 **=** """INSERT INTO description(beach\_id, text, source) VALUES(%s, %s, %s)"""

args6 **=** **(**result**,** tip\_text**,** source**)**

cursor**.**execute**(**sql6**,** args6**)**

db**.**commit**()**

**print** 'End\n'

ph**=**client**.**venues**.**photos**(**plaz\_id**,** params**={**'group'**:** 'venue'**})**

**for** photo **in** ph**[**"photos"**][**"items"**]:**

photo\_url**=**photo**[**"prefix"**]+**"width500"**+**photo**[**"suffix"**]**

sqlph **=** """INSERT INTO photos (beach\_id, ph) VALUES (%s, %s)"""

cursor**.**execute**(**sqlph**,** **(**result**,** photo\_url**))**

db**.**commit**()**

# disconnect from server

db**.**close**()**

### geo.py

Αρχείο Κώδικα geo.py

#!/usr/bin/python

#-\*- coding: iso-8859-7 -\*-

**import** json

**import** urllib2

**import** urllib

**import** string

**import** sys

**import** MySQLdb

**from** geopy**.**geocoders **import** Nominatim

reload**(**sys**)**

sys**.**setdefaultencoding**(**'utf8'**)**

**from** converting **import** remove\_accents

**from** geopy**.**geocoders **import** Nominatim

response **=** urllib2**.**urlopen**(**' http://api.geonames.org/searchJSON?q=beach&country=GR&maxRows=1000&lang=el&username=kapsali'**)**

json\_object **=** json**.**load**(**response**)**

servername **=** "localhost"

username **=** "root"

password **=** ""

dbname **=** "database"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cursor **=** db**.**cursor**()**

#print json\_object

#print type(json\_object)

#print json\_object[ u'geonames']

**for** item **in** json\_object**[** u'geonames'**]:**

**if** item**[**u'countryCode'**]** **==** 'GR'**:**

**print** 'BEGIN'

**try:**

**print** item**[** u'toponymName'**]**

**except:**

**print** remove\_accents**(**item**[** u'toponymName'**])**

**print** item**[**u'lat'**]**

**print** item**[**u'lng'**]**

**print** item**[**u'adminName1'**]**

**try:**

name **=** item**[**u'name'**]**

**print** name

**except:**

name **=** remove\_accents**(**item**[**u'name'**])**

**print** name

sql1 **=** """INSERT INTO beach(name, lat, lon, region) VALUES(%s, %s, %s, %s) """

args1 **=** **(**name**,** item**[**u'lat'**],** item**[**u'lng'**],** item**[**u'adminName1'**])**

cursor**.**execute**(**sql1**,** args1**)**

geolocator **=** Nominatim**()**

coord**=(**float**(**item**[**u'lat'**]),**float**(**item**[**u'lng'**]))**

location **=** geolocator**.**reverse**(**coord**)**

sql3 **=** """SELECT MAX(id) FROM beach"""

cursor**.**execute**(**sql3**)**

result **=** cursor**.**fetchone**()**

sql2 **=** """INSERT INTO beachplus(address ,source, beach\_id, alternate) VALUES (%s, %s, %s, %s)"""

cursor**.**execute**(**sql2**,** **(**location**.**address**,** 'geonames.org'**,** result**,** item**[**u'name'**]))**

db**.**commit**()**

### distance.py

Αρχείο Κώδικα distance.py

**from** math **import** radians**,** cos**,** sin**,** asin**,** sqrt

**def** haversine**(**lon1**,** lat1**,** lon2**,** lat2**):**

"""

Calculate the great circle distance between two points

on the earth (specified in decimal degrees)

"""

# convert decimal degrees to radians

lon1**,** lat1**,** lon2**,** lat2 **=** map**(**radians**,** **[**lon1**,** lat1**,** lon2**,** lat2**])**

# haversine formula

dlon **=** lon2 **-** lon1

dlat **=** lat2 **-** lat1

a **=** sin**(**dlat**/**2**)\*\***2 **+** cos**(**lat1**)** **\*** cos**(**lat2**)** **\*** sin**(**dlon**/**2**)\*\***2

c **=** 2 **\*** asin**(**sqrt**(**a**))**

km **=** 6367 **\*** c

**return** km

### para.py

Αρχείο Κώδικα para.py

**def** krithrio\_omoiothtas**(**name\_a**,** name\_b**):**

# -\*- coding: utf-8 -\*-

**from** detect\_latin **import** isEnglish

**from** transliterate **import** translit**,** get\_available\_language\_codes

**from** similar\_names **import** similar

**print** isEnglish**(**name\_b**)**

**print** isEnglish**(**name\_a**)**

**if** isEnglish**(**name\_b**)==True:**

name\_a **=** translit**(**name\_a**,** 'el'**,**reversed**=True)**

**print** name\_a

**print** similar**(**name\_a**,** name\_b**)**

**elif** isEnglish**(**name\_a**)==True:**

name\_b **=** translit**(**name\_b**,** 'el'**,**reversed**=True)**

**print** name\_b

**print** similar**(**name\_a**,** name\_b**)**

**else:**

**print** "the names are greek chars"

**print** name\_a

**print** name\_b

**print** similar**(**name\_a**,** name\_b**)**

**return** similar**(**name\_a**,** name\_b**)**

### detect\_latin.py

Αρχείο Κώδικα detect\_latin.py

# -\*- coding: utf-8 -\*-

**def** isEnglish**(**s**):**

**try:**

s**.**decode**(**'ascii'**)**

**except:**

**return** **False**

**else:**

**return** **True**

### similar\_names.py

# -\*- coding: utf-8 -\*-

**def** isEnglish**(**s**):**

**try:**

s**.**decode**(**'ascii'**)**

**except:**

**return** **False**

**else:**

**return** **True**# !/usr/bin/python

# -\*- coding: utf-8 -\*-

**def** similar**(**a**,**b**):**

**import** difflib

a **=** a**.**lower**()**

b **=** b**.**lower**()**

p **=** u'\u03c0\u03b1\u03c1\u03b1\u03bb\u03af\u03b1'

p1 **=** u'\u03c0\u03b1\u03c1\u03b1\u03bb\u03b9\u03b1'

a **=** a**.**replace**(**'beach'**,**''**)**

a **=** a**.**replace**(**'paralia'**,**''**)**

a **=** a**.**replace**(**p**,**''**)**

a **=** a**.**replace**(**p1**,**''**)**

b **=** b**.**replace**(**'beach'**,**''**)**

b **=** b**.**replace**(**'paralia'**,**''**)**

b **=** b**.**replace**(**p**,**''**)**

b **=** b**.**replace**(**p1**,**''**)**

a**=**a**.**strip**()**

b**=**b**.**strip**()**

**print** a

**print** b

seq **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

a\_init **=** a

b\_init **=** b

**print** 'first comparison'

#print a

#print b

**print** seq**.**ratio**()**

starta **=** a**.**find**(**'('**)**

enda **=** a**.**find**(**')'**)**

**if** starta **!=** **-**1 **and** enda **!=** **-**1 **:**

resulta **=** a**[**starta**+**1**:**enda**]**

resulta **=** resulta**.**strip**(**" "**)**

tempa **=** a\_init**.**replace**(**resulta**,** ''**)**

tempa **=** tempa**.**replace**(**'('**,**''**)**

tempa **=** tempa**.**replace**(**')'**,**''**)**

tempa **=** tempa**.**strip**(**" "**)**

**elif** a**.**find**(**","**)** **!=-**1 **:**

koma **=** a**.**find**(**","**)**

apartA **=** a**[**0**:**koma**].**strip**(**' '**)**

apartB **=** a**[**koma**+**1**:**len**(**a**)].**strip**(**' '**)**

**print** 'the result is:'

**print** a

starb **=** b**.**find**(**'('**)**

endb **=** b**.**find**(**')'**)**

**if** starb **!=** **-**1 **and** endb **!=** **-**1**:**

resultb **=** b**[**starb**+**1**:**endb**]**

resultb **=** resultb**.**strip**(**" "**)**

tempb **=** b\_init**.**replace**(**resultb**,** ''**)**

tempb **=** tempb**.**replace**(**'('**,**''**)**

tempb **=** tempb**.**replace**(**')'**,**''**)**

tempb **=** tempb**.**strip**(**" "**)**

**elif** b**.**find**(**","**)** **!=-**1 **:**

koma **=** b**.**find**(**","**)**

bpartA **=** b**[**0**:**koma**].**strip**(**' '**)**

bpartB **=** b**[**koma**+**1**:**len**(**b**)].**strip**(**' '**)**

**print** bpartB

seq1 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

**print** 'second comparison'

**print** a

**print** b

**print** seq1**.**ratio**()**

**if** starb **!=** **-**1 **and** endb **!=** **-**1 **and** starta **!=** **-**1 **and** enda **!=** **-**1**:**

a **=** tempa

b **=** tempb

seqtatb **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

**print** seqtatb**.**ratio**()**

rate1 **=** seqtatb**.**ratio**()**

b **=** resultb

seqtaresb **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate8 **=** seqtaresb**.**ratio**()**

**print** rate8

a **=** resulta

b **=** tempb

seqresatb **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

**print** seqresatb**.**ratio**()**

rate2 **=** seqresatb**.**ratio**()**

a **=** resulta

b **=** resultb

seqresaresb **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate9 **=** seqresaresb**.**ratio**()**

**print** rate9

**else:**

rate8 **=** 0.0

rate1 **=** 0.0

rate2 **=** 0.0

rate9 **=** 0.0

**if** starta **!=** **-**1 **and** enda **!=** **-**1 **and** b**.**find**(**','**)** **!=** **-**1**:**

**print** '1-1'

a **=** tempa

b **=** bpartA

**print** a

**print** b

seq12 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate12 **=** seq12**.**ratio**()**

**print** rate12

**print** '1-1'

a **=** resulta

b **=** bpartB

**print** a

**print** b

seq13 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate13 **=** seq13**.**ratio**()**

**print** rate13

a **=** tempa

b **=** bpartB

**print** a

**print** b

seq14 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate14 **=** seq13**.**ratio**()**

**print** rate14

a **=** resulta

b **=** bpartA

**print** a

**print** b

seq15 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate15 **=** seq13**.**ratio**()**

**print** rate15

**else:**

rate12 **=** 0.0

rate15 **=** 0.0

rate14 **=** 0.0

rate13 **=** 0.0

**if** starb **!=** **-**1 **and** endb **!=** **-**1 **and** a**.**find**(**','**)** **!=** **-**1**:**

a **=** apartA

b **=** tempb

seq16 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate16 **=** seq16**.**ratio**()**

**print** rate16

a **=** apartB

b **=** resultb

seq17 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate17 **=** seq17**.**ratio**()**

**print** rate17

b **=** tempb

a **=** apartB

seq18 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate18 **=** seq18**.**ratio**()**

**print** rate18

a **=** apartA

b **=** resultb

seq19 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate19 **=** seq19**.**ratio**()**

**print** rate19

**else:**

rate18 **=** 0.0

rate19 **=** 0.0

rate16 **=** 0.0

rate17 **=** 0.0

**if** a**.**find**(**','**)** **!=-**1 **and** b**.**find**(**','**)** **!=-**1**:**

a **=** apartA

b **=** bpartA

seq20 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate20 **=** seq20**.**ratio**()**

**print** rate20

a **=** apartB

b **=** bpartB

seq21 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate21 **=** seq21**.**ratio**()**

**print** rate21

a **=** apartB

b **=** bpartA

seq22 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate22 **=** seq22**.**ratio**()**

**print** rate22

a **=** apartA

b **=** bpartB

seq23 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate23 **=** seq23**.**ratio**()**

**print** rate23

**else:**

rate23 **=** 0.0

rate22 **=** 0.0

rate21 **=** 0.0

rate20 **=** 0.0

**if** a**.**find**(**','**)** **!=-**1 **and** **(**starb **==** **-**1 **and** endb **==** **-**1 **or** b**.**find**(**','**)** **==** **-**1**):**

a **=** apartA

b **=** b\_init

seq24 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate24 **=** seq24**.**ratio**()**

**print** rate24

a **=** apartB

seq25 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate25 **=** seq25**.**ratio**()**

**print** rate25

**else:**

rate25 **=** 0.0

rate24 **=** 0.0

**if** b**.**find**(**','**)** **!=-**1 **and** **(**starta **==** **-**1 **and** enda **==** **-**1 **or** a**.**find**(**','**)** **==-**1**):**

a **=** a\_init

b **=** bpartA

seq26 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate26 **=** seq26**.**ratio**()**

**print** rate26

b **=** bpartB

seq27 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate27 **=** seq27**.**ratio**()**

**print** rate27

**else:**

rate26 **=** 0.0

rate27 **=** 0.0

**if** starb **!=** **-**1 **and** endb **!=** **-**1 **and** **(**starta **==** **-**1 **and** enda **==** **-**1 **or** a**.**find**(**','**)** **==-**1**):**

b **=** tempb

a **=** a\_init

seqab **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate3 **=** seqab**.**ratio**()**

**print** rate3

b **=** resultb

seq30 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate30 **=** seq30**.**ratio**()**

**print** rate30

**else:**

rate3 **=** 0.0

rate30 **=** 0.0

**if** **(**starb **==** **-**1 **and** endb **==** **-**1 **or** b**.**find**(**','**)** **==** **-**1**)** **and** starta **!=** **-**1 **and** enda **!=** **-**1**:**

b **=** b\_init

a **=** tempa

seqba **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate4 **=** seqba**.**ratio**()**

**print** rate4

a **=** resulta

seq40 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate40 **=** seq40**.**ratio**()**

**print** rate40

**else:**

rate4 **=** 0.0

rate40 **=** 0.0

**return** max**(**seq**.**ratio**(),** seq1**.**ratio**(),** rate1**,**rate2**,** rate13**,** rate15**,** rate12**,** rate14**,** rate18**,** rate19**,** rate16**,** rate17**,** rate22**,** rate23**,** rate20**,** rate21**,** rate24**,** rate25**,** rate26**,** rate27**,** rate3**,** rate30**,** rate4**,** rate40**)**

**print** a

**print** b

**print** seq1**.**ratio**()**

**if** starb **!=** **-**1 **and** endb **!=** **-**1 **and** starta **!=** **-**1 **and** enda **!=** **-**1**:**

a **=** tempa

b **=** tempb

seqtatb **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

**print** seqtatb**.**ratio**()**

rate1 **=** seqtatb**.**ratio**()**

b **=** resultb

seqtaresb **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate8 **=** seqtaresb**.**ratio**()**

**print** rate8

a **=** resulta

b **=** tempb

seqresatb **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

**print** seqresatb**.**ratio**()**

rate2 **=** seqresatb**.**ratio**()**

a **=** resulta

b **=** resultb

seqresaresb **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate9 **=** seqresaresb**.**ratio**()**

**print** rate9

**else:**

rate8 **=** 0.0

rate1 **=** 0.0

rate2 **=** 0.0

rate9 **=** 0.0

**if** starta **!=** **-**1 **and** enda **!=** **-**1 **and** b**.**find**(**','**)** **!=** **-**1**:**

**print** '1-1'

a **=** tempa

b **=** bpartA

**print** a

**print** b

seq12 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate12 **=** seq12**.**ratio**()**

**print** rate12

**print** '1-1'

a **=** resulta

b **=** bpartB

**print** a

**print** b

seq13 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate13 **=** seq13**.**ratio**()**

**print** rate13

a **=** tempa

b **=** bpartB

**print** a

**print** b

seq14 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate14 **=** seq13**.**ratio**()**

**print** rate14

a **=** resulta

b **=** bpartA

**print** a

**print** b

seq15 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate15 **=** seq13**.**ratio**()**

**print** rate15

**else:**

rate12 **=** 0.0

rate15 **=** 0.0

rate14 **=** 0.0

rate13 **=** 0.0

**if** starb **!=** **-**1 **and** endb **!=** **-**1 **and** a**.**find**(**','**)** **!=** **-**1**:**

a **=** apartA

b **=** tempb

seq16 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate16 **=** seq16**.**ratio**()**

**print** rate16

a **=** apartB

b **=** resultb

seq17 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate17 **=** seq17**.**ratio**()**

**print** rate17

b **=** tempb

a **=** apartB

seq18 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate18 **=** seq18**.**ratio**()**

**print** rate18

a **=** apartA

b **=** resultb

seq19 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate19 **=** seq19**.**ratio**()**

**print** rate19

**else:**

rate18 **=** 0.0

rate19 **=** 0.0

rate16 **=** 0.0

rate17 **=** 0.0

**if** a**.**find**(**','**)** **!=-**1 **and** b**.**find**(**','**)** **!=-**1**:**

a **=** apartA

b **=** bpartA

seq20 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate20 **=** seq20**.**ratio**()**

**print** rate20

a **=** apartB

b **=** bpartB

seq21 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate21 **=** seq21**.**ratio**()**

**print** rate21

a **=** apartB

b **=** bpartA

seq22 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate22 **=** seq22**.**ratio**()**

**print** rate22

a **=** apartA

b **=** bpartB

seq23 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate23 **=** seq23**.**ratio**()**

**print** rate23

**else:**

rate23 **=** 0.0

rate22 **=** 0.0

rate21 **=** 0.0

rate20 **=** 0.0

**if** a**.**find**(**','**)** **!=-**1 **and** **(**starb **==** **-**1 **and** endb **==** **-**1 **or** b**.**find**(**','**)** **==** **-**1**):**

a **=** apartA

b **=** b\_init

seq24 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate24 **=** seq24**.**ratio**()**

**print** rate24

a **=** apartB

seq25 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate25 **=** seq25**.**ratio**()**

**print** rate25

**else:**

rate25 **=** 0.0

rate24 **=** 0.0

**if** b**.**find**(**','**)** **!=-**1 **and** **(**starta **==** **-**1 **and** enda **==** **-**1 **or** a**.**find**(**','**)** **==-**1**):**

a **=** a\_init

b **=** bpartA

seq26 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate26 **=** seq26**.**ratio**()**

**print** rate26

b **=** bpartB

seq27 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate27 **=** seq27**.**ratio**()**

**print** rate27

**else:**

rate26 **=** 0.0

rate27 **=** 0.0

**if** starb **!=** **-**1 **and** endb **!=** **-**1 **and** **(**starta **==** **-**1 **and** enda **==** **-**1 **or** a**.**find**(**','**)** **==-**1**):**

b **=** tempb

a **=** a\_init

seqab **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate3 **=** seqab**.**ratio**()**

**print** rate3

b **=** resultb

seq30 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate30 **=** seq30**.**ratio**()**

**print** rate30

**else:**

rate3 **=** 0.0

rate30 **=** 0.0

**if** **(**starb **==** **-**1 **and** endb **==** **-**1 **or** b**.**find**(**','**)** **==** **-**1**)** **and** starta **!=** **-**1 **and** enda **!=** **-**1**:**

b **=** b\_init

a **=** tempa

seqba **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate4 **=** seqba**.**ratio**()**

**print** rate4

a **=** resulta

seq40 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate40 **=** seq40**.**ratio**()**

**print** rate40

**else:**

rate4 **=** 0.0

rate40 **=** 0.0

**return** max**(**seq**.**ratio**(),** seq1**.**ratio**(),** rate1**,**rate2**,** rate13**,** rate15**,** rate12**,** rate14**,** rate18**,** rate19**,** rate16**,** rate17**,** rate22**,** rate23**,** rate20**,** rate21**,** rate24**,** rate25**,** rate26**,** rate27**,** rate3**,** rate30**,** rate4**,** rate40**)**

Αρχείο Κώδικα similar\_names.py

rate18 **=** seq18**.**ratio**()**

a **=** apartA

b **=** resultb

seq19 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate19 **=** seq19**.**ratio**()**

**else:**

rate18 **=** 0.0

rate19 **=** 0.0

rate16 **=** 0.0

rate17 **=** 0.0

**if** a**.**find**(**','**)** **!=-**1 **and** b**.**find**(**','**)** **!=-**1**:**

a **=** apartA

b **=** bpartA

seq20 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate20 **=** seq20**.**ratio**()** a **=** apartB

b **=** bpartB

seq21 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate21 **=** seq21**.**ratio**()**

**print** rate21

a **=** apartB

b **=** bpartA

seq22 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate22 **=** seq22**.**ratio**()**

a **=** apartA

b **=** bpartB

seq23 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate23 **=** seq23**.**ratio**()**

**else:**

rate23 **=** 0.0

rate22 **=** 0.0

rate21 **=** 0.0

rate20 **=** 0.0

**if** a**.**find**(**','**)** **!=-**1 **and** **(**starb **==** **-**1 **and** endb **==** **-**1 **or** b**.**find**(**','**)** **==** **-**1**):**

a **=** apartA

b **=** b\_init

seq24 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate24 **=** seq24**.**ratio**()**

**print** rate24

a **=** apartB

seq25 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate25 **=** seq25**.**ratio**()**

**print** rate25

**else:**

rate25 **=** 0.0

rate24 **=** 0.0

**if** b**.**find**(**','**)** **!=-**1 **and** **(**starta **==** **-**1 **and** enda **==** **-**1 **or** a**.**find**(**','**)** **==-**1**):**

a **=** a\_init

b **=** bpartA

seq26 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate26 **=** seq26**.**ratio**()**

b **=** bpartB

seq27 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate27 **=** seq27**.**ratio**()**

**else:**

rate26 **=** 0.0

rate27 **=** 0.0

**if** starb **!=** **-**1 **and** endb **!=** **-**1 **and** **(**starta **==** **-**1 **and** enda **==** **-**1 **or** a**.**find**(**','**)** **==-**1**):**

b **=** tempb

a **=** a\_init

seqab **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate3 **=** seqab**.**ratio**()**

b **=** resultb

seq30 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate30 **=** seq30**.**ratio**()**

**else:**

rate3 **=** 0.0

rate30 **=** 0.0

**if** **(**starb **==** **-**1 **and** endb **==** **-**1 **or** b**.**find**(**','**)** **==** **-**1**)** **and** starta **!=** **-**1 **and** enda **!=** **-**1**:**

b **=** b\_init

a **=** tempa

seqba **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate4 **=** seqba**.**ratio**()**

a **=** resulta

seq40 **=** difflib**.**SequenceMatcher**(**a**=**a**.**lower**(),** b**=**b**.**lower**())**

rate40 **=** seq40**.**ratio**()**

**else:**

rate4 **=** 0.0

rate40 **=** 0.0

**return** max**(**seq**.**ratio**(),** seq1**.**ratio**(),** rate1**,**rate2**,** rate13**,** rate15**,** rate12**,** rate14**,** rate18**,** rate19**,** rate16**,** rate17**,** rate22**,** rate23**,** rate20**,** rate21**,** rate24**,** rate25**,** rate26**,** rate27**,** rate3**,** rate30**,** rate4**,** rate40**)**

### api.py

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

**from** flask **import** Flask

**from** flask **import** Response

**import** MySQLdb

**import** urllib

**from** flask **import** jsonify

**from** flask **import** request

**from** flask**.**ext**.**responses **import** json\_response**,** xml\_response**,** auto\_response

**from** flask\_restful **import** Resource**,** Api

**from** flask\_restful **import** reqparse

**from** flask**.**ext**.**mysql **import** MySQL

**from** distance **import** haversine

**import** json

**import** sys

sys**.**path**.**insert**(**0**,** 'C:\Users\home\greekbeachesapp'**)**

#from distance import haversine

mysql **=** MySQL**()**

app **=** Flask**(**\_\_name\_\_**)**

# MySQL configurations

app**.**config**[**'MYSQL\_DATABASE\_USER'**]** **=** 'root'

app**.**config**[**'MYSQL\_DATABASE\_PASSWORD'**]** **=** ''

app**.**config**[**'MYSQL\_DATABASE\_DB'**]** **=** 'database'

app**.**config**[**'MYSQL\_DATABASE\_HOST'**]** **=** 'localhost'

mysql**.**init\_app**(**app**)**

api **=** Api**(**app**)**

**class** **get\_beach(**Resource**):**

*@app.route***(**'/get\_beach/<idd>'**,** methods**=[**'GET'**])**

**def** get**(**idd**):**

servername **=** "localhost"

username **=** "root"

password **=** ""

dbname **=** "database"

q\_list\_one **=** "SELECT name FROM beach WHERE id = %s"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cur **=** db**.**cursor**()**

json\_return **=** **{}**

**try:**

cur**.**execute**(**q\_list\_one**,** **(**idd**,))**

r\_list\_one **=** cur**.**fetchall**()**

items **=** **[]**

**for** data **in** r\_list\_one**:**

i **=** **{**'name'**:**data**[**0**]}**

items**.**append**(**i**)**

**except** Exception **as** error**:**

**return** **{**'error'**:**str**(**error**)}**

**finally:**

cur**.**close**()**

db**.**close**()**

**return** jsonify**(**Items **=** items**)**

**class** **GetAllItems(**Resource**):**

*@app.route***(**'/GetAllItems'**,** methods**=[**'GET'**])**

**def** getAll**():**

servername **=** "localhost"

username **=** "root"

password **=** ""

dbname **=** "database"

queryAll **=** "SELECT name, id, photo FROM beach"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cur **=** db**.**cursor**()**

json\_return **=** **{}**

**try:**

cur**.**execute**(**queryAll**)**

listall **=** cur**.**fetchall**()**

items **=** **[]**

**for** data **in** listall**:**

j**={**'name'**:**data**[**0**],** 'id'**:**data**[**1**],** 'photo'**:**data**[**2**]}**

items**.**append**(**j**)**

**except** Exception **as** error**:**

**return** **{**'error'**:**str**(**error**)}**

**finally:**

cur**.**close**()**

db**.**close**()**

**return** jsonify**(**Items **=** items**)**

**class** **getCoordinates(**Resource**):**

*@app.route***(**'/getCoordinates/<idd>'**,** methods**=[**'GET'**])**

**def** getlatlon**(**idd**):**

servername **=** "localhost"

username **=** "root"

password **=** ""

dbname **=** "database"

querylatlon **=** "SELECT lat, lon, name, id FROM beach WHERE id = %s"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cur **=** db**.**cursor**()**

json\_return **=** **{}**

**try:**

cur**.**execute**(**querylatlon**,** **(**idd**,))**

listcoords **=** cur**.**fetchall**()**

items **=** **[]**

**for** data **in** listcoords**:**

jc **=** **{**'latitude'**:**float**(**data**[**0**]),** 'longitude'**:**float**(**data**[**1**]),** 'name'**:**data**[**2**],** 'id'**:** data**[**3**]}**

items**.**append**(**jc**)**

**except** Exception **as** error**:**

**return** **{**'Error'**:**str**(**error**)}**

**finally:**

cur**.**close**()**

db**.**close**()**

**return** jsonify**(**Items **=** items**)**

**class** **getAlternateNames(**Resource**):**

*@app.route***(**'/getAlternateNames/<idd>'**,** methods **=** **[**'GET'**])**

**def** getAltNames**(**idd**):**

servername **=** "localhost"

username **=** "root"

password **=** ""

dbname **=** "database"

queryalt **=** "SELECT alternate, source, beach\_id FROM beachplus WHERE beach\_id = %s"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cur **=** db**.**cursor**()**

json\_return **=** **{}**

**try:**

cur**.**execute**(**queryalt**,** **(**idd**,))**

listalt **=** cur**.**fetchall**()**

items **=** **[]**

**for** data **in** listalt**:**

jalt **=** **{**'Names'**:**data**[**0**],** 'Source'**:** data**[**1**],** 'Id'**:** data**[**2**]}**

items**.**append**(**jalt**)**

**except** Exception **as** error**:**

**return** **{**'Error'**:**str**(**error**)}**

**finally:**

cur**.**close**()**

db**.**close**()**

**return** jsonify**(**Items **=** items**)**

**class** **getAddress(**Resource**):**

*@app.route***(**'/getAddress/<idd>'**,** methods **=** **[**'GET'**])**

**def** getAd**(**idd**):**

servername **=** "localhost"

username **=** "root"

password **=** ""

dbname **=** "database"

queryforadd **=** "SELECT address, alternate, beach\_id FROM beachplus WHERE beach\_id = %s"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cur **=** db**.**cursor**()**

json\_return **=** **{}**

**try:**

cur**.**execute**(**queryforadd**,** **(**idd**,))**

listad **=** cur**.**fetchone**()**

items **=** **[]**

#for data in listad:

jad **=** **{**'address'**:**listad**[**0**],** 'Names'**:**listad**[**1**],** 'Id'**:**listad**[**2**]}**

items**.**append**(**jad**)**

**except** Exception **as** error**:**

**return** **{**'Error'**:**str**(**error**)}**

**finally:**

cur**.**close**()**

db**.**close**()**

**return** jsonify**(**Items **=** items**)**

**class** **getDetails(**Resource**):**

*@app.route***(**'/getDetails/<idd>'**,** methods **=** **[**'GET'**])**

**def** getD**(**idd**):**

servername **=** "localhost"

username **=** "root"

password **=** ""

dbname **=** "database"

queryfordesc **=** "SELECT text, source, beach\_id FROM description WHERE beach\_id = %s"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cur **=** db**.**cursor**()**

json\_return **=** **{}**

**try:**

cur**.**execute**(**queryfordesc**,** **(**idd**,))**

listdesc **=** cur**.**fetchall**()**

items **=** **[]**

**for** data **in** listdesc**:**

jdesc **=** **{**'description'**:**data**[**0**],** 'source'**:**data**[**1**],** 'id'**:**data**[**2**]}**

items**.**append**(**jdesc**)**

**except** Exception **as** error**:**

**return** **{**'Error'**:**str**(**error**)}**

**finally:**

cur**.**close**()**

db**.**close**()**

**return** jsonify**(**Items **=** items**)**

**class** **getPhotos(**Resource**):**

*@app.route***(**'/getPhotos/<idd>'**,** methods **=** **[**'GET'**])**

**def** getPh**(**idd**):**

servername **=** 'localhost'

username **=** 'root'

password **=** ''

dbname **=** 'database'

queryPhoto **=** "SELECT ph, beach\_id FROM photos WHERE beach\_id = %s"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cur **=** db**.**cursor**()**

json\_return **=** **{}**

**try:**

cur**.**execute**(**queryPhoto**,** **(**idd**,))**

listph **=** cur**.**fetchall**()**

items **=** **[]**

**for** data **in** listph**:**

jph **=** **{**'photo'**:**data**[**0**],** 'id'**:**data**[**1**]}**

items.append(jph)

except Exception as error:

return {'Error':str(error)}

finally:

cur.close()

db.close()

return jsonify(Items = items)

class getRegionBeaches(Resource):

@app.route('/getRegionBeaches/<pointer>', methods=['GET'])

def getRbeaches(pointer):

servername = "localhost"

username = "root"

password = ""

dbname = "database"

region = ["Περιφέρεια Πελοποννήσου", "Περιφέρεια Κρήτης", "Περιφέρεια Ιόνιων νησιών", "Περιφέρεια Νοτίου Αιγαίου", "Περιφέρεια Ανατολικής Μακεδονίας και Θράκης", "Περιφέρεια Στερεάς Ελλάδας", "Περιφέρεια Βόρειου Αιγαίου", "Περιφέρεια Αττικής", "Περιφέρεια Θεσσαλίας", "Περιφέρεια Κεντρικής Μακεδονίας", "Περιφέρεια Ηπείρου", "Περιφέρεια Δυτικής Ελλάδας" ]

queryAll = "SELECT name, id, photo FROM beach WHERE region = %s"

db = MySQLdb.connect(servername, username, password, dbname, charset="utf8", use\_unicode=True)

cur = db.cursor()

i = int(pointer)

json\_return = {}

try:

cur.execute(queryAll, (region[i],))

listall = cur.fetchall()

items = []

for data in listall:

j={'name':data[0], 'id':data[1], 'photo':data[2]}

items.append(j)

except Exception as error:

return {'error':str(error)}

finally:

cur.close()

db.close()

return jsonify(Items = items)

class getNearBeaches(Resource):

@app.route('/getNearBeaches/<mylat>/<mylong>', methods = ['GET'])

def getNear(mylat, mylong):

servername = "localhost"

username = "root"

password = ""

dbname = "database"

myquery = "SELECT name, id, lat, lon, photo FROM beach"

db = MySQLdb.connect(servername, username, password, dbname, charset="utf8", use\_unicode=True)

cur = db.cursor()

json\_return = {}

try:

cur.execute(myquery)

items = []

names = []

lats = []

lons = []

ids = []

havs = []

phs = []

results = cur.fetchall()

for item in results:

if (haversine(float(item[3]), float(item[2]), float(mylong), float(mylat)) <= 15):

names.append(item[0])

ids.append(int(item[1]))

lats.append(float(item[2]))

lons.append(float(item[3]))

havs.append(haversine(float(item[3]), float(item[2]), float(mylong), float(mylat)))

phs.append(item[4])

i = len(names) - 1

while i >=0:

myjson = {'name':names[i], 'id':ids[i], 'latitude':lats[i], 'longitude':lons[i], 'distance':float(havs[i]), 'photo':phs[i]}

items.append(myjson)

i = i -1;

except Exception as error:

return {'Error':str(error)}

finally:

cur.close()

db.close()

return jsonify(Items = items)

if \_\_name\_\_ == '\_\_main\_\_':

app.debug = True

app.run(host='0.0.0.0')

**class** **getCoordinates(**Resource**):**

*@app.route***(**'/getCoordinates/<idd>'**,** methods**=[**'GET'**])**

**def** getlatlon**(**idd**):**

servername **=** "localhost"

username **=** "root"

password **=** ""

dbname **=** "database"

querylatlon **=** "SELECT lat, lon, name, id FROM beach WHERE id = %s"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cur **=** db**.**cursor**()**

json\_return **=** **{}**

**try:**

cur**.**execute**(**querylatlon**,** **(**idd**,))**

listcoords **=** cur**.**fetchall**()**

items **=** **[]**

**for** data **in** listcoords**:**

jc **=** **{**'latitude'**:**float**(**data**[**0**]),** 'longitude'**:**float**(**data**[**1**]),** 'name'**:**data**[**2**],** 'id'**:** data**[**3**]}**

items**.**append**(**jc**)**

**except** Exception **as** error**:**

**return** **{**'Error'**:**str**(**error**)}**

**finally:**

cur**.**close**()**

db**.**close**()**

**return** jsonify**(**Items **=** items**)**

**class** **getAlternateNames(**Resource**):**

*@app.route***(**'/getAlternateNames/<idd>'**,** methods **=** **[**'GET'**])**

**def** getAltNames**(**idd**):**

servername **=** "localhost"

username **=** "root"

password **=** ""

dbname **=** "database"

queryalt **=** "SELECT alternate, source, beach\_id FROM beachplus WHERE beach\_id = %s"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cur **=** db**.**cursor**()**

json\_return **=** **{}**

**try:**

cur**.**execute**(**queryalt**,** **(**idd**,))**

listalt **=** cur**.**fetchall**()**

items **=** **[]**

**for** data **in** listalt**:**

jalt **=** **{**'Names'**:**data**[**0**],** 'Source'**:** data**[**1**],** 'Id'**:** data**[**2**]}**

items**.**append**(**jalt**)**

**except** Exception **as** error**:**

**return** **{**'Error'**:**str**(**error**)}**

**finally:**

cur**.**close**()**

db**.**close**()**

**return** jsonify**(**Items **=** items**)**

**class** **getAddress(**Resource**):**

*@app.route***(**'/getAddress/<idd>'**,** methods **=** **[**'GET'**])**

**def** getAd**(**idd**):**

servername **=** "localhost"

username **=** "root"

password **=** ""

dbname **=** "database"

queryforadd **=** "SELECT address, alternate, beach\_id FROM beachplus WHERE beach\_id = %s"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cur **=** db**.**cursor**()**

json\_return **=** **{}**

**try:**

cur**.**execute**(**queryforadd**,** **(**idd**,))**

listad **=** cur**.**fetchone**()**

items **=** **[]**

#for data in listad:

jad **=** **{**'address'**:**listad**[**0**],** 'Names'**:**listad**[**1**],** 'Id'**:**listad**[**2**]}**

items**.**append**(**jad**)**

**except** Exception **as** error**:**

**return** **{**'Error'**:**str**(**error**)}**

**finally:**

cur**.**close**()**

db**.**close**()**

**return** jsonify**(**Items **=** items**)**

**class** **getDetails(**Resource**):**

*@app.route***(**'/getDetails/<idd>'**,** methods **=** **[**'GET'**])**

**def** getD**(**idd**):**

servername **=** "localhost"

username **=** "root"

password **=** ""

dbname **=** "database"

queryfordesc **=** "SELECT text, source, beach\_id FROM description WHERE beach\_id = %s"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cur **=** db**.**cursor**()**

json\_return **=** **{}**

**try:**

cur**.**execute**(**queryfordesc**,** **(**idd**,))**

listdesc **=** cur**.**fetchall**()**

items **=** **[]**

**for** data **in** listdesc**:**

jdesc **=** **{**'description'**:**data**[**0**],** 'source'**:**data**[**1**],** 'id'**:**data**[**2**]}**

items**.**append**(**jdesc**)**

**except** Exception **as** error**:**

**return** **{**'Error'**:**str**(**error**)}**

**finally:**

cur**.**close**()**

db**.**close**()**

**return** jsonify**(**Items **=** items**)**

**try:**

cur**.**execute**(**queryfordesc**,** **(**idd**,))**

listdesc **=** cur**.**fetchall**()**

items **=** **[]**

**for** data **in** listdesc**:**

jdesc **=** **{**'description'**:**data**[**0**],** 'source'**:**data**[**1**],** 'id'**:**data**[**2**]}**

items**.**append**(**jdesc**)**

**except** Exception **as** error**:**

**return** **{**'Error'**:**str**(**error**)}**

**finally:**

cur**.**close**()**

db**.**close**()**

**return** jsonify**(**Items **=** items**)**

**class** **getPhotos(**Resource**):**

*@app.route***(**'/getPhotos/<idd>'**,** methods **=** **[**'GET'**])**

**def** getPh**(**idd**):**

servername **=** 'localhost'

username **=** 'root'

password **=** ''

dbname **=** 'database'

queryPhoto **=** "SELECT ph, beach\_id FROM photos WHERE beach\_id = %s"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cur **=** db**.**cursor**()**

json\_return **=** **{}**

**try:**

cur**.**execute**(**queryPhoto**,** **(**idd**,))**

listph **=** cur**.**fetchall**()**

items **=** **[]**

**for** data **in** listph**:**

jph **=** **{**'photo'**:**data**[**0**],** 'id'**:**data**[**1**]}**

items**.**append**(**jph**)**

**except** Exception **as** error**:**

**return** **{**'Error'**:**str**(**error**)}**

**finally:**

cur**.**close**()**

db**.**close**()**

**return** jsonify**(**Items **=** items**)**

**class** **getRegionBeaches(**Resource**):**

*@app.route***(**'/getRegionBeaches/<pointer>'**,** methods**=[**'GET'**])**

**def** getRbeaches**(**pointer**):**

servername **=** "localhost"

username **=** "root"

password **=** ""

dbname **=** "database"

region **=** **[**"Περιφέρεια Πελοποννήσου"**,** "Περιφέρεια Κρήτης"**,** "Περιφέρεια Ιόνιων νησιών"**,** "Περιφέρεια Νοτίου Αιγαίου"**,** "Περιφέρεια Ανατολικής Μακεδονίας και Θράκης"**,** "Περιφέρεια Στερεάς Ελλάδας"**,** "Περιφέρεια Βόρειου Αιγαίου"**,** "Περιφέρεια Αττικής"**,** "Περιφέρεια Θεσσαλίας"**,** "Περιφέρεια Κεντρικής Μακεδονίας"**,** "Περιφέρεια Ηπείρου"**,** "Περιφέρεια Δυτικής Ελλάδας" **]**

queryAll **=** "SELECT name, id, photo FROM beach WHERE region = %s"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cur **=** db**.**cursor**()**

i **=** int**(**pointer**)**

json\_return **=** **{}**

**try:**

cur**.**execute**(**queryAll**,** **(**region**[**i**],))**

listall **=** cur**.**fetchall**()**

items **=** **[]**

**for** data **in** listall**:**

j**={**'name'**:**data**[**0**],** 'id'**:**data**[**1**],** 'photo'**:**data**[**2**]}**

items**.**append**(**j**)**

**except** Exception **as** error**:**

**return** **{**'error'**:**str**(**error**)}**

**finally:**

cur**.**close**()**

db**.**close**()**

**return** jsonify**(**Items **=** items**)**

**class** **getNearBeaches(**Resource**):**

*@app.route***(**'/getNearBeaches/<mylat>/<mylong>'**,** methods **=** **[**'GET'**])**

**def** getNear**(**mylat**,** mylong**):**

servername **=** "localhost"

username **=** "root"

password **=** ""

dbname **=** "database"

myquery **=** "SELECT name, id, lat, lon, photo FROM beach"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cur **=** db**.**cursor**()**

json\_return **=** **{}**

**try:**

cur**.**execute**(**myquery**)**

items **=** **[]**

names **=** **[]**

lats **=** **[]**

lons **=** **[]**

ids **=** **[]**

havs **=** **[]**

phs **=** **[]**

results **=** cur**.**fetchall**()**

**for** item **in** results**:**

Αρχείο Κώδικα api.py

phs **=** **[]**

results **=** cur**.**fetchall**()**

**for** item **in** results**:**

**if** **(**haversine**(**float**(**item**[**3**]),** float**(**item**[**2**]),** float**(**mylong**),** float**(**mylat**))** **<=** 15**):**

names**.**append**(**item**[**0**])**

ids**.**append**(**int**(**item**[**1**]))**

lats**.**append**(**float**(**item**[**2**]))**

lons**.**append**(**float**(**item**[**3**]))**

havs**.**append**(**haversine**(**float**(**item**[**3**]),** float**(**item**[**2**]),** float**(**mylong**),** float**(**mylat**)))**

phs**.**append**(**item**[**4**])**

i **=** len**(**names**)** **-** 1

**while** i **>=**0**:**

myjson **=** **{**'name'**:**names**[**i**],** 'id'**:**ids**[**i**],** 'latitude'**:**lats**[**i**],** 'longitude'**:**lons**[**i**],** 'distance'**:**float**(**havs**[**i**]),** 'photo'**:**phs**[**i**]}**

items**.**append**(**myjson**)**

i **=** i **-**1**;**

**except** Exception **as** error**:**

**return** **{**'Error'**:**str**(**error**)}**

**finally:**

cur**.**close**()**

db**.**close**()**

**return** jsonify**(**Items **=** items**)**

**if** \_\_name\_\_ **==** '\_\_main\_\_'**:**

app**.**debug **=** **True**

app**.**run**(**host**=**'0.0.0.0'**)**

### fillInfo.py

Αρχείο Κώδικα fillInfo.py

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

**import** MySQLdb

servername **=** "localhost"

username **=** "root"

password **=** ""

dbname **=** "database"

myquery **=** "SELECT \*FROM beach WHERE region = %s"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cur **=** db**.**cursor**()**

fetchId **=** "SELECT id FROM beach"

cur**.**execute**(**fetchId**)**

idds **=** cur**.**fetchall**()**

ids **=** **[]**

fetchPhoto **=** "SELECT ph FROM photos WHERE beach\_id = %s LIMIT 1"

**for** iding **in** idds**:**

**print** iding**[**0**]**

cur**.**execute**(**fetchPhoto**,** **(**int**(**iding**[**0**]),))**

photo **=** cur**.**fetchall**()**

**if** photo **==** **():**

**print** ''

**else:**

**print** photo**[**0**][**0**]**

qup **=** "UPDATE beach SET photo = %s WHERE id = %s"

args **=** **(**photo**[**0**][**0**],** int**(**iding**[**0**]))**

cur**.**execute**(**qup**,** args**)**

db**.**commit**()**

### passenger\_wsgi.py

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

**import** sys**,** os

**import** MySQLdb

INTERP **=** os**.**path**.**join**(**os**.**environ**[**'HOME'**],** 'bin'**,** 'python'**)**

**if** sys**.**executable **!=** INTERP**:**

os**.**execl**(**INTERP**,** INTERP**,** **\***sys**.**argv**)**

sys**.**path**.**append**(**os**.**getcwd**())**

**from** flask **import** Flask**,** jsonify

**from** distance **import** haversine

application **=** Flask**(**\_\_name\_\_**)**

*@application.route***(**'/GetAllItems'**,** methods**=[**'GET'**])**

**def** getAll**():**

servername **=** "mysql.dream.mmlab.gr"

username **=** "mmlab\_user"

password **=** "mml@b#student"

dbname **=** "dreammmlab\_kapsali"

queryAll **=** "SELECT name, id, photo FROM beach"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cur **=** db**.**cursor**()**

json\_return **=** **{}**

**try:**

cur**.**execute**(**queryAll**)**

listall **=** cur**.**fetchall**()**

items **=** **[]**

**for** data **in** listall**:**

j**={**'name'**:**data**[**0**],** 'id'**:**data**[**1**],** 'photo'**:**data**[**2**]}**

items**.**append**(**j**)**

**except** Exception **as** error**:**

**return** **{**'error'**:**str**(**error**)}**

**finally:**

cur**.**close**()**

db**.**close**()**

**return** jsonify**(**Items **=** items**)**

*@application.route***(**'/get\_beach/<idd>'**,** methods**=[**'GET'**])**

**def** get**(**idd**):**

servername **=** "mysql.dream.mmlab.gr"

username **=** "mmlab\_user"

password **=** "mml@b#student"

dbname **=** "dreammmlab\_kapsali"

q\_list\_one **=** "SELECT name FROM beach WHERE id = %s"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cur **=** db**.**cursor**()**

json\_return **=** **{}**

**try:**

cur**.**execute**(**q\_list\_one**,** **(**idd**,))**

r\_list\_one **=** cur**.**fetchall**()**

items **=** **[]**

**for** data **in** r\_list\_one**:**

i **=** **{**'name'**:**data**[**0**]}**

items**.**append**(**i**)**

**except** Exception **as** error**:**

**return** **{**'error'**:**str**(**error**)}**

**finally:**

cur**.**close**()**

db**.**close**()**

**return** jsonify**(**Items **=** items**)**

*@application.route***(**'/getCoordinates/<idd>'**,** methods**=[**'GET'**])**

**def** getlatlon**(**idd**):**

servername **=** "mysql.dream.mmlab.gr"

username **=** "mmlab\_user"

password **=** "mml@b#student"

dbname **=** "dreammmlab\_kapsali"

querylatlon **=** "SELECT lat, lon, name, id FROM beach WHERE id = %s"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cur **=** db**.**cursor**()**

json\_return **=** **{}**

**try:**

cur**.**execute**(**querylatlon**,** **(**idd**,))**

listcoords **=** cur**.**fetchall**()**

items **=** **[]**

**for** data **in** listcoords**:**

jc **=** **{**'latitude'**:**float**(**data**[**0**]),** 'longitude'**:**float**(**data**[**1**]),** 'name'**:**data**[**2**],** 'id'**:** data**[**3**]}**

items**.**append**(**jc**)**

**except** Exception **as** error**:**

**return** **{**'Error'**:**str**(**error**)}**

**finally:**

cur**.**close**()**

db**.**close**()**

**return** jsonify**(**Items **=** items**)**

*@application.route***(**'/getAlternateNames/<idd>'**,** methods **=** **[**'GET'**])**

**def** getAltNames**(**idd**):**

servername **=** "mysql.dream.mmlab.gr"

username **=** "mmlab\_user"

password **=** "mml@b#student"

dbname **=** "dreammmlab\_kapsali"

queryalt **=** "SELECT alternate, source, beach\_id FROM beachplus WHERE beach\_id = %s"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cur **=** db**.**cursor**()**

json\_return **=** **{}**

**try:**

cur**.**execute**(**queryalt**,** **(**idd**,))**

listalt **=** cur**.**fetchall**()**

items **=** **[]**

**for** data **in** listalt**:**

jalt **=** **{**'Names'**:**data**[**0**],** 'Source'**:** data**[**1**],** 'Id'**:** data**[**2**]}**

items**.**append**(**jalt**)**

**except** Exception **as** error**:**

**return** **{**'Error'**:**str**(**error**)}**

**finally:**

cur**.**close**()**

db**.**close**()**

**return** jsonify**(**Items **=** items**)**

*@application.route***(**'/getAddress/<idd>'**,** methods **=** **[**'GET'**])**

**def** getAd**(**idd**):**

servername **=** "mysql.dream.mmlab.gr"

username **=** "mmlab\_user"

password **=** "mml@b#student"

dbname **=** "dreammmlab\_kapsali"

queryforadd **=** "SELECT address, alternate, beach\_id FROM beachplus WHERE beach\_id = %s"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cur **=** db**.**cursor**()**

json\_return **=** **{}**

**try:**

cur**.**execute**(**queryforadd**,** **(**idd**,))**

listad **=** cur**.**fetchone**()**

items **=** **[]**

#for data in listad:

jad **=** **{**'address'**:**listad**[**0**],** 'Names'**:**listad**[**1**],** 'Id'**:**listad**[**2**]}**

items**.**append**(**jad**)**

**except** Exception **as** error**:**

**return** **{**'Error'**:**str**(**error**)}**

**finally:**

cur**.**close**()**

db**.**close**()**

**return** jsonify**(**Items **=** items**)**

*@application.route***(**'/getDetails/<idd>'**,** methods **=** **[**'GET'**])**

**def** getD**(**idd**):**

servername **=** "mysql.dream.mmlab.gr"

username **=** "mmlab\_user"

password **=** "mml@b#student"

dbname **=** "dreammmlab\_kapsali"

queryfordesc **=** "SELECT text, source, beach\_id FROM description WHERE beach\_id = %s"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cur **=** db**.**cursor**()**

json\_return **=** **{}**

**try:**

cur**.**execute**(**queryfordesc**,** **(**idd**,))**

listdesc **=** cur**.**fetchall**()**

items **=** **[]**

**for** data **in** listdesc**:**

jdesc **=** **{**'description'**:**data**[**0**],** 'source'**:**data**[**1**],** 'id'**:**data**[**2**]}**

items**.**append**(**jdesc**)**

**except** Exception **as** error**:**

**return** **{**'Error'**:**str**(**error**)}**

**finally:**

cur**.**close**()**

db**.**close**()**

**return** jsonify**(**Items **=** items**)**

*@application.route***(**'/getPhotos/<idd>'**,** methods **=** **[**'GET'**])**

**def** getPh**(**idd**):**

servername **=** 'mysql.dream.mmlab.gr'

username **=** 'mmlab\_user'

password **=** 'mml@b#student'

dbname **=** 'dreammmlab\_kapsali'

queryPhoto **=** "SELECT ph, beach\_id FROM photos WHERE beach\_id = %s"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cur **=** db**.**cursor**()**

json\_return **=** **{}**

Αρχείο Κώδικα passenger\_wsgi.py

**try:**

cur**.**execute**(**queryPhoto**,** **(**idd**,))**

listph **=** cur**.**fetchall**()**

items **=** **[]**

**for** data **in** listph**:**

jph **=** **{**'photo'**:**data**[**0**],** 'id'**:**data**[**1**]}**

items**.**append**(**jph**)**

**except** Exception **as** error**:**

**return** **{**'Error'**:**str**(**error**)}**

**finally:**

cur**.**close**()**

db**.**close**()**

**return** jsonify**(**Items **=** items**)**

*@application.route***(**'/getRegionBeaches/<pointer>'**,** methods**=[**'GET'**])**

**def** getRbeaches**(**pointer**):**

servername **=** "mysql.dream.mmlab.gr"

username **=** "mmlab\_user"

password **=** "mml@b#student"

dbname **=** "dreammmlab\_kapsali"

region **=** **[**"Περιφέρεια Πελοποννήσου"**,** "Περιφέρεια Κρήτης"**,** "Περιφέρεια Ιόνιων νησιών"**,** "Περιφέρεια Νοτίου Αιγαίου"**,** "Περιφέρεια Ανατολικής Μακεδονίας και Θράκης"**,** "Περιφέρεια Στερεάς Ελλάδας"**,** "Περιφέρεια Βόρειου Αιγαίου"**,** "Περιφέρεια Αττικής"**,** "Περιφέρεια Θεσσαλίας"**,** "Περιφέρεια Κεντρικής Μακεδονίας"**,** "Περιφέρεια Ηπείρου"**,** "Περιφέρεια Δυτικής Ελλάδας" **]**

queryAll **=** "SELECT name, id, photo FROM beach WHERE region = %s"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cur **=** db**.**cursor**()**

i **=** int**(**pointer**)**

json\_return **=** **{}**

**try:**

cur**.**execute**(**queryAll**,** **(**region**[**i**],))**

listall **=** cur**.**fetchall**()**

items **=** **[]**

**for** data **in** listall**:**

j**={**'name'**:**data**[**0**],** 'id'**:**data**[**1**],** 'photo'**:**data**[**2**]}**

items**.**append**(**j**)**

**except** Exception **as** error**:**

**return** **{**'error'**:**str**(**error**)}**

**finally:**

cur**.**close**()**

db**.**close**()**

**return** jsonify**(**Items **=** items**)**

*@application.route***(**'/getNearBeaches/<mylat>/<mylong>'**,** methods **=** **[**'GET'**])**

**def** getNear**(**mylat**,** mylong**):**

servername **=** "mysql.dream.mmlab.gr"

username **=** "mmlab\_user"

password **=** "mml@b#student"

dbname **=** "dreammmlab\_kapsali"

myquery **=** "SELECT name, id, lat, lon, photo FROM beach"

db **=** MySQLdb**.**connect**(**servername**,** username**,** password**,** dbname**,** charset**=**"utf8"**,** use\_unicode**=True)**

cur **=** db**.**cursor**()**

json\_return **=** **{}**

**try:**

cur**.**execute**(**myquery**)**

items **=** **[]**

names **=** **[]**

lats **=** **[]**

lons **=** **[]**

ids **=** **[]**

havs **=** **[]**

phs **=** **[]**

results **=** cur**.**fetchall**()**

**for** item **in** results**:**

**if** **(**haversine**(**float**(**item**[**3**]),** float**(**item**[**2**]),** float**(**mylong**),** float**(**mylat**))** **<=** 15**):**

names**.**append**(**item**[**0**])**

ids**.**append**(**int**(**item**[**1**]))**

lats**.**append**(**float**(**item**[**2**]))**

lons**.**append**(**float**(**item**[**3**]))**

havs**.**append**(**haversine**(**float**(**item**[**3**]),** float**(**item**[**2**]),** float**(**mylong**),** float**(**mylat**)))**

phs**.**append**(**item**[**4**])**

i **=** len**(**names**)** **-** 1

**while** i **>=**0**:**

myjson **=** **{**'name'**:**names**[**i**],** 'id'**:**ids**[**i**],** 'latitude'**:**lats**[**i**],** 'longitude'**:**lons**[**i**],** 'distance'**:**float**(**havs**[**i**]),** 'photo'**:**phs**[**i**]}**

items**.**append**(**myjson**)**

i **=** i **-**1**;**

**except** Exception **as** error**:**

**return** **{**'Error'**:**str**(**error**)}**

**finally:**

cur**.**close**()**

db**.**close**()**

**return** jsonify**(**Items **=** items**)**

**if** \_\_name\_\_ **==** '\_\_main\_\_'**:**

application**.**debug **=** **True**

application**.**run**(**host**=**'0.0.0.0'**)**

1. Βιβλιογραφία

* Συστήματα Βάσεων Δεδομένων, 6η Έκδοση, Silberschatz, Korth, Sudarschan
* RESTful Web Services, Richardson, Ruby
* Learning Scrapy, Kouzis-Loukas
* Fundamentals of Computer Programming With C#, Nakov
* Mining the Social Web, Matthew A. Russel
* Building Web APIs with Flask, Miguel Grinberg
* <http://www.tutorialspoint.com/python/>
* http://www.tutorialspoint.com/csharp/
* <https://developer.xamarin.com/guides/cross-platform/getting_started/>
* <https://en.wikipedia.org/wiki/Web_crawler>
* <https://en.wikipedia.org/wiki/Information_system>
* https://en.wikipedia.org/wiki/Android\_(operating\_system)
* <http://json2csharp.com/>
* <http://blog.falafel.com/31-days-of-xamarin-android/>
* https://developer.xamarin.com/guides/cross-platform/application\_fundamentals/data/
* <http://flask.pocoo.org/>
* <https://github.com/codepath/android_guides/wiki/Genymotion-2.0-Emulators-with-Google-Play-support>
* <https://developer.xamarin.com/guides/cross-platform/application_fundamentals/data/part_3_using_sqlite_orm/>
* <https://developer.xamarin.com/recipes/android/other_ux/fragment/create_a_fragment/>
* <https://developer.xamarin.com/recipes/android/web_services/consuming_services/call_a_rest_web_service/>
* <https://phpmyadmin.dreamhost.com/index.php?token=15c4f1e97d20ef3009a86b97d8553ea7>
* <https://mattcarrier.com/flask-dreamhost-setup/>
* <http://scrapy.org/doc/>
* http://mherman.org/blog/2012/11/05/scraping-web-pages-with-scrapy/#.V8g7SyiLSM8
* <https://forums.xamarin.com/>
* <http://stackoverflow.com/>
* <http://restsharp.org/>
* <http://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world>
* <http://www.tutorialspoint.com/xpath/>
* <http://code.tutsplus.com/tutorials/an-introduction-to-pythons-flask-framework--net-28822>
* <http://blog.ostebaronen.dk/2014/03/filtering-listview-with-searchview-in.html>
* <http://pawel.sawicz.eu/restsharp/>
* <https://www.stum.de/2009/12/22/using-restsharp-to-consume-restful-web-services/>
* <https://bugzilla.xamarin.com/>