ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Ανάπτυξη εφαρμογής δημιουργίας και παρουσίασης ψηφιακών ιστοριών πολιτιστικού περιεχομένου για κινητές συσκευές

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΕΥΑΓΓΕΛΟΥ ΚΟΥΤΚΙΑ

Επιβλέπων : Στέφανος Κόλλιας

Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2015

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Ανάπτυξη εφαρμογής δημιουργίας και παρουσίασης ψηφιακών ιστοριών πολιτιστικού περιεχομένου για κινητές συσκευές

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΕΥΑΓΓΕΛΟΥ ΚΟΥΤΚΙΑ

Επιβλέπων : Στέφανος Κόλλιας
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 19 η Μαρτίου 2015.

.....	.....	.....
Στέφανος Κόλλιας	Γεώργιος Στάμου	Ανδρέας Σταφυλοπάτης
Καθηγητής Ε.Μ.Π.	Επίκουρος Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2015

.....

ΕΥΑΓΓΕΛΟΣ ΚΟΥΤΚΙΑΣ

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Σκοπός αυτής της διπλωματικής είναι η ανάπτυξη μιας σύγχρονης εφαρμογής για το iOS που θα επιτρέπει στο χρήστη να δημιουργήσει και να μοιραστεί τη δική του, προσωπική ιστορία. Πέρα όμως από τη σαφή παρότρυνση για δημιουργία νέου υλικού, η εφαρμογή αυτή αποσκοπεί και τον εφοδιασμό του χρήστη με πλούσιο υλικό από τις ψηφιακές βιβλιοθήκες Europeana και DPLA, για τη διάνθιση των ιστοριών του.

Μέσω της εφαρμογής, ο χρήστης δύναται να κατασκευάσει την ιστορία του εισάγοντας προσωπικό υλικό, όπως φωτογραφίες από τη συσκευή του, ηχογραφημένα αρχεία και κείμενο που έχει συντάξει, αλλά και να περιπλανηθεί στα ψηφιακές συλλογές του DPLA και της Europeana, εκτελώντας αναζητήσεις στα αντίστοιχα Apis. Μπορεί να δει τα σχετικά μεταδεδομένα για το εκάστοτε αντικείμενο και να σώσει τοπικά όποια εικόνα του ελκύει το ενδιαφέρον. Έτσι, γίνεται εφικτή η επαναχρησιμοποίηση της σε όσες ιστορίες ο ίδιος επιθυμεί. Αφού συνθέσει την ιστορία του, μπορεί είτε να τη σώσει τοπικά, είτε να τη δημοσιεύσει, οπότε και γίνεται ορατή από τους υπόλοιπους χρήστες.

Στην εφαρμογή αυτή συναντώνται τα κύρια στοιχεία κάθε σύγχρονης εφαρμογής για κινητές συσκευές, όπως το κομμάτι του δικτύου, της αποθήκευσης δεδομένων τοπικά, της παρουσίασης περιεχομένου και της απόκρισης σε επαφές του χρήστη. Επιπλέον, για να καταστεί δυνατός ο διαμοιρασμός ιστοριών, χρησιμοποιήθηκε το Parse για τη δημιουργία του backend, συμπληρώνοντας έτσι τη σφαιρική προσέγγιση για την ανάπτυξη μιας τέτοιας εφαρμογής.

Λέξεις κλειδιά: Digital storytelling, iOS, Europeana, DPLA

Abstract

The goal of this diploma thesis is to develop an app for the iOS which will offer users the ability to create and share their own personal stories. Apart from cultivating user creativity, it also aims at supplying the aspiring users with rich material from the digital libraries of Europeana and DPLA in order to enhance their stories.

Through the app, users are able to insert text, record sound and add photos from their device, but can also wander through the digital collections of DPLA and Europeana by executing searches to the respective Apis. They can expand the metadata of every object that draws their attention and save any image for further use in their own stories. Thus, reusability of content is at main interest. Once a story has been created, the user can either publish it, or save it locally for further edit.

This app encapsulates the main aspects of any modern mobile app such as networking, saving data locally, content layout and responding to user touches and gestures. Additionally, to make story sharing feasible, apart from the client app, a backend was also created using Parse framework, concluding in an all round approach of mobile application development.

Keywords: Digital storytelling, iOS, Europeana, DPLA

Ευχαριστήριο σημείωμα

Θα ήθελα να ευχαριστήσω θερμά τον καθηγητή Ε.Μ.Π και επιβλέποντα της εργασίας κ. Στέφανο Κόλλια για την ανάθεση της διπλωματικής εργασίας, καθώς και για την πολύτιμη καθοδήγηση του καθ' όλη τη διάρκεια εκπόνησης της. Επίσης, θα ήθελα να ευχαριστήσω τον ερευνητή Βασίλη Τζουβάρα για την άριστη συνεργασία που είχαμε και τις συμβουλές που μου παρείχε. Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια μου για τη συνεχή υποστήριξη κατά τη διάρκεια των προπτυχιακών μου σπουδών.

Περιεχόμενα

1 Εισαγωγή 1

1.1 Digital Storytelling 1

1.2 Κίνητρο για τη συγκεκριμένη εφαρμογή 2

1.3 Europeana 3

1.4 DPLA (Digital Public Library of America) 3

1.5 Οργάνωση διπλωματικής 5

2 Σχετικές εφαρμογές 6

2.1 Storehouse (εφαρμογή iOS – ιστοσελίδα) 6

2.2 Storybird (ιστοσελίδα) 7

2.3 Storify (ιστοσελίδα) 7

3 Θεωρητικό - τεχνολογικό υπόβαθρο 9

3.1 REST και RESTful web services 9

3.2 JSON 10

3.3 Παρουσίαση των Controllers 11

4 Ανάλυση απαιτήσεων συστήματος 15

4.1 Απαιτήσεις συστήματος 15

4.1.1 Λειτουργικές απαιτήσεις 15

4.1.2 Μη λειτουργικές απαιτήσεις 16

4.2 Αρχιτεκτονική του συστήματος 17

4.3 Σχεδίαση εφαρμογής 18

4.3.1 Σχεδιαστικό μοτίβο MVC 18

4.3.2 Domain Class Diagram 20

4.4 Σύντομη Περιγραφή Βασικών κλάσεων 21

4.4.1 HomeViewController 21

4.4.2 StoryViewController 21

4.4.3 LoginViewController 21

4.4.4 CreateStoryViewController 21

4.4.5 CoverPickerController 22

4.4.6 SavedPhotosGalleryViewController 22

4.4.7 StorySaveHelper 22

4.4.8 CollectionGalleryViewController 22

4.4.9 MyStoriesViewController 23

4.4.10 Query Details – Query Handler 23

4.4.11 SearchItem 23
4.4.12 ImageDetailViewController 23
4.5 Περιπτώσεις χρήσης της εφαρμογής 24
5 Περιγραφή λειτουργικότητας εφαρμογής 30
5.1 Αρχική οθόνη εφαρμογής 30
5.1.1 Αναζήτηση σε δημοσιευμένες ιστορίες 32
5.1.2 Άνοιγμα ιστορίας 32
5.1.3 Σύνδεση - Εγγραφή χρήστη 35
5.2 Σύνταξη Ιστορίας 38
5.2.1 Συνοπτική περιγραφή 38
5.2.2 Εισαγωγή φωτογραφίας 40
5.2.3 Εισαγωγή κειμένου 41
5.2.4 Ηχογράφηση μηνύματος 43
5.2.5 Επιλογή εξωφύλλου ιστορίας 43
5.3 Αναζήτηση σε ψηφιακές βιβλιοθήκες 44
5.3.1 CollectionGalleryViewController 44
5.3.2 Λεπτομερής παρουσίαση αντικειμένου αναζήτησης 47
5.4 Πρόχειρα 49
5.4.1 Συνοπτική περιγραφή 49
5.4.2 MyStoriesViewController 49
6 Λεπτομέρειες υλοποίησης 51
6.1 Εργαλεία 51
6.2 Restkit 52
6.3 MBProgressHud 55
6.4 SDWebImage 55
6.5 Parse Backend 56
6.5.1 Διαδικασία εγγραφής – ταυτοποίησης χρήστη – αποσύνδεση 57
6.5.2 Ανέβασμα αρχείων στη βάση του Parse 58
6.5.3 Ανέβασμα ιστορίας και σύνδεση της με τον εκάστοτε χρήστη 59
6.6 CHTCollectionViewWaterfallLayout – Δυναμικό ύψος κελιών 61
6.7 Cocoapods – Σύνδεση βιβλιοθηκών 61
6.8 Αποθήκευση ιστοριών 63
6.9 Gesture Recognizers - Touch Events – Responder Chain 66
6.10 EUROPEANA και DPLA APIS 72
6.10.1 QueryDetails 72
6.10.2 SearchItem 73

7 Επίλογος 75
7.1 Σύνοψη 75
7.2 Μελλοντικές επεκτάσεις 75
8 Βιβλιογραφία 77

1

Εισαγωγή

1.1 Digital Storytelling

Το Digital Storytelling [1] αποτελεί τη μοντέρνα έκφραση της αρχαίας παράδοσης της διήγησης ιστοριών. Κατά το πέρασμα των αιώνων η αφήγηση ιστοριών χρησιμοποιήθηκε από την ανθρωπότητα για τη μετάδοση αξιών, σοφίας και γνώσης. Οι ιστορίες αυτές μπορεί να έχουν πολλές διαφορετικές μορφές. Πάντα προσαρμόζονταν στα εκάστοτε μέσα που αναδύονταν κατά καιρούς και έδιναν μεγαλύτερη δύναμη και εμβέλεια στον αφηγητή. Ξεκινώντας από απλές συζητήσεις γύρω από τη φωτιά και τη μετάδοση από στόμα σε στόμα φτάσαμε στην ισχυρή «φωνή» των υπολογιστών και των κινητών συσκευών.

Ο όρος Digital storytelling αναφέρεται, λοιπόν, στη χρησιμοποίηση ψηφιακών μέσων με τρόπο που να επιτρέπει στους ανθρώπους να δημιουργούν και να μοιράζονται ιστορίες από διάφορες εκφάνσεις τις ζωής τους.

Για τη δημιουργία των ιστοριών αυτών, μπορεί να χρησιμοποιηθεί ποικιλία πρώτων υλών, όπως εικόνες, αρχεία ήχου, κείμενο, βίντεο, αφήγηση, animations καθώς και οποιαδήποτε άλλη μορφή μη υλικών μέσων (υλικά δηλαδή που υπάρχουν μόνο σε ηλεκτρονική μορφή και όχι τα ίδια τα εκθέματα αυτά καθαυτά). Ο χαρακτήρας τους μπορεί να είναι ιστορικός, διδακτικός, καθοδηγητικός, φανταστικός ή και βιωματικός. Το μεγάλο προτέρημα σε σχέση με τις συμβατικές μορφές διήγησης, είναι ότι λόγω της ποικιλίας των μέσων που μπορεί κανείς να χρησιμοποιήσει, βελτιώνει την εμπειρία, τόσο του χρήστη όσο και του συγγραφέα και προωθεί τη διαδραστικότητα και την εκφραστικότητα.

Τα γεγονότα δείχνουν πως όταν ο κόσμος έχει την ευκαιρία να δημιουργήσει υλικό για δημόσια προβολή το κάνει και μάλιστα με ενθουσιώδη τρόπο, κάτι που φαίνεται και από το εξής παράδειγμα. Ενδεικτικό είναι ότι το υλικό που παράχθηκε στο YouTube σε ένα

διάστημα έξι μηνών, ήταν μεγαλύτερο από ότι παρήγαγαν και τα τρία μεγαλύτερα κανάλια της Αμερικής σε 60 χρόνια μετάδοσης [2]. Οι άνθρωποί δημιουργούν συνεχώς και δείχνουν μεγάλη διάθεση να συμμετάσχουν σε αυτή τη μαζική και όλο νόημα «συζήτηση» που λαμβάνει χώρο στα ψηφιακά μέσα.

Πριν από λίγα χρόνια η παραγωγή οπτικοακουστικού υλικού είχε πολύ μεγάλο κόστος, ωστόσο κάτι τέτοιο άλλαξε άρδην με τις τεχνολογικές εξελίξεις και κυρίως με την ανάπτυξη του διαδικτύου. Πέραν του ιστού, που αποτέλεσε τη βασική πηγή του digital story telling, τώρα πια με τη ραγδαία ανάπτυξη των κινητών συσκευών οι χρήστες έχουν πολλά εργαλεία στα χέρια τους, που μπορούν να χρησιμοποιήσουν ανά πάσα στιγμή για τη δημιουργία των προσωπικών τους ιστοριών. Υπάρχουν, πλέον, πολλές εφαρμογές, άλλες δωρεάν και άλλες επί πληρωμή, που παρέχουν στον οποιονδήποτε χρήστη τη δυνατότητα με απλά βήματα και χωρίς τεχνικές γνώσεις να συντάξει και να δημοσιεύσει τη δική του ιστορία.

1.2 Κίνητρο για τη συγκεκριμένη εφαρμογή

Μέσα στα πλαίσια της ταχείας, αυτής, ανάπτυξης του digital story telling, θεωρήθηκε ότι υπάρχει πρόσφορο έδαφος για την δημιουργία μιας mobile εφαρμογής σε λειτουργικό σύστημα iOS που θα επιτρέπει στους χρήστες τη δημιουργία και τη δημοσιοποίηση των προσωπικών του ιστοριών. Η εφαρμογή αυτή αναπτύχθηκε για το iPad, καθώς αυτό το μέγεθος της συσκευής κρίθηκε καταλληλότερο για τη δημιουργία και παρουσίαση ιστοριών.

Στόχος της και βασική διαφοροποίησης της σε σχέση με άλλες προϋπάρχουσες, ήταν πέραν της δυνατότητας σύνταξης της ιστορίας, να δίνεται στο χρήστη και υλικό από το οποίο θα μπορεί να εμπλουτίσει την αφήγηση του. Το υλικό αυτό είναι πολιτισμικού χαρακτήρα, καθώς ο χρήστης μπορεί να ψάξει σε δύο από τις μεγαλύτερες ψηφιακές βιβλιοθήκες, τη Europeana και τη DPLA (Digital Public Library of America) και να επαναχρησιμοποιήσει υλικό της αρεσκείας του, αναμειγμένο με δικό του, προσωπικό υλικό.

Συνοπτικά παρουσιάζονται οι βασικές λειτουργικότητες:

- ένας editor από όπου ο χρήστης μπορεί να συντάξει τη δικιά του ιστορία εισάγοντας κείμενο, εικόνες και ηχογραφήσεις
- δυνατότητα αναζήτησης στα DPLA και Europeana, φιλτραρίσματος των αποτελεσμάτων, ανάπτυξη των μεταδεδομένων ενός αντικειμένου και αποθήκευση του για χρήση στη δική του ιστορία
- δημοσιοποίηση της ιστορίας, ώστε να είναι ορατή σε όλους τους χρήστες, είτε αποθήκευση της στη συσκευή του χρήστη για μετέπειτα επεξεργασία

Ακολουθεί μια σύντομη περιγραφή των δύο αυτών ψηφιακών βιβλιοθηκών.

1.3 Europeana

Η Europeana αποτελεί τη ψηφιακή βιβλιοθήκη της Ευρώπης. Άρχισε να λειτουργεί στις 20 Νοεμβρίου 2008 και δίνει πρόσβαση σε βιβλία, χάρτες, ηχογραφήσεις, φωτογραφίες, αρχειακά έγγραφα, πίνακες και ταινίες, υλικά προερχόμενα από εθνικές βιβλιοθήκες και πολιτιστικά ιδρύματα των 27 κρατών μελών της Ευρωπαϊκής Ένωσης. Τα ψηφιακά αντικείμενα δεν αποθηκεύονται σε κάποιο κεντρικό υπολογιστή, αλλά φιλοξενούνται στο εκάστοτε πάροχο. Η Europeana από τη μεριά της συλλέγει και αποθηκεύει σημασιολογική πληροφορία για τα αντικείμενα, δίνοντας τη δυνατότητα στους χρήστες να φιλτράρουν τις αναζητήσεις τους αλλά και να δουν τα συγκεντρωμένα μεταδεδομένα για τα εκθέματα.

Παρέχει πολλές λειτουργίες, οι πιο σημαντικές από τις οποίες είναι οι εξής:

- Το portal [3] που είναι παρέχει τη δυνατότητα αναζήτησης σε εκατομμύρια αντικείμενα της Ευρωπαϊκής πολιτιστικής κληρονομιάς. Εκτός από τις ψηφιακές απεικονίσεις των αντικειμένων η Europeana κρατάει πληθώρα μεταδεδομένων που σχετίζονται με το ψηφιακό έκθεμα.
- Τη σελίδα Europeana Professionals [4] όπου βιβλιοθηκάριοι, αρχειοθέτες και έφοροι μπορούν να ενημερωθούν και να συζητήσουν σχετικά με τη ψηφιακή διατήρηση του υλικού.
- Το Europeana Labs [5] που παρέχει APIs για τη χρήση του υλικού της βιβλιοθήκης σε εφαρμογές. Τα Apis που είναι διαθέσιμα είναι δύο:
    - Το πρώτο είναι ένα Restful Api που ενδείκνυται για δυναμική αναζήτηση και ανάκτηση δεδομένων και προσφέρει ακριβώς τα ίδια δεδομένα με αυτά που εμφανίζονται στο portal.
    - Το δεύτερο είναι πιο πολύ πειραματικό και υποστηρίζει το κατέβασμα ολόκληρων σετ δεδομένων και εξελιγμένη σημασιολογική αναζήτηση μέσω της SPARQL γλώσσας. Προς το παρόν περιλαμβάνει ένα υποσύνολο των δεδομένων της Europeana, περίπου 21 από τις 34 εκατομμύρια εγγραφές και γίνονται προσπάθειες για την ανάπτυξή του.

Για την εφαρμογή που αναπτύχθηκε στα πλαίσια της εφαρμογής αυτής χρησιμοποιήθηκε το REST Api, λόγω της πληρότητας του και τον τρόπο παροχής δεδομένων, που ενδείκνυται για κινητές συσκευές.

1.4 DPLA (Digital Public Library of America)

Το DPLA, όπως προϊδεάζει και το όνομά του, είναι η εθνική ψηφιακή βιβλιοθήκη της Αμερικής. Η οργάνωση του ξεκίνησε το 2010 από το Berkman Center for Internet & Society

του πανεπιστημίου Harvard. Το έργο άρχισε να υλοποιείται τον Οκτώβρη του 2011 και απαίτησε τη συνεργασία εκατοντάδων δημοσίων και ερευνητών βιβλιοθηκάριων, ειδικών σε θέματα ψηφιακής τεχνολογίας και εθελοντών.

Συνδυάζει το υλικό των βιβλιοθηκών, των αρχείων και των μουσείων της Αμερικής. Προσπαθεί να διατηρήσει τη συνολική έκφραση της ανθρώπινης δραστηριότητας, όπως το γραπτό λόγο και τα έργα τέχνης σε συνδυασμό με τις προσπάθειες και τα δεδομένα της επιστήμης. Στόχος του είναι να διευρύνει το ανοιχτά διαθέσιμο υλικό και αυτός ο πολιτισμικός πλούτος να είναι πιο εύκολα ανακαλύψιμος και χρησιμοποιήσιμος από τον καθένα. Δύο είναι τα βασικά στοιχεία λειτουργίας του:

- Το portal [6] που παρέχει σε μαθητές, δασκάλους , ερευνητές και το κοινό πληθώρα υλικού. Είναι κάτι πολύ περισσότερο από μια απλή μηχανή αναζήτησης, καθώς παρέχει καινοτόμους τρόπους αναζήτησης και φιλτραρίσματος αντικειμένων συμπεριλαμβανομένου τον χρόνο, τη γεωγραφική θέση, το εικονικό ράφι, τον τύπο απεικόνισης, το αντικείμενο και τον πάροχο του υλικού.
- Μια πλατφόρμα που επιτρέπει νέες και ανατρεπτικές χρήσεις της πολιτιστικής κληρονομιάς. Προσφέρει ένα ανοιχτό API [7] και ανοιχτά δεδομένα που μπορούν να χρησιμοποιηθούν από προγραμματιστές και ερευνητές για να αναπτυχθούν εργαλεία και εφαρμογές.

1.5 Οργάνωση διπλωματικής

Η δομή της διπλωματικής είναι η ακόλουθη:

Κεφάλαιο 1: Το κεφάλαιο αυτό αποτελεί την εισαγωγή στο αντικείμενο της διπλωματικής και εξηγεί το κίνητρο για την ανάπτυξη της σχετικής εφαρμογής.

Κεφάλαιο 2: Παρουσιάζει αντίστοιχες εφαρμογές που καταπιάνονται με το αντικείμενο και σε τι διαφέρουν από την παρούσα.

Κεφάλαιο 3: Επεξηγούνται κάποιος βασικές έννοιες που χρησιμοποιούνται κατά τη διάρκεια της διπλωματικής.

Κεφάλαιο 4: Εδώ γίνεται η ανάλυση των απαιτήσεων του συστήματος, περιγράφεται η αρχιτεκτονική και η δομή της εφαρμογής και αναλύεται η λειτουργία των επιμέρους συνιστωσών.

Κεφάλαιο 5: Παρουσιάζεται πλήρως η λειτουργικότητα της εφαρμογής.

Κεφάλαιο 6: Λεπτομέρειες για την υλοποίηση και τα εργαλεία που χρησιμοποιήθηκαν κατά την ανάπτυξη της εφαρμογής.

Κεφάλαιο 7: Επίλογος

Κεφάλαιο 8: Βιβλιογραφία

2

Σχετικές εφαρμογές

Το digital story telling είναι ένα αντικείμενο με ραγδαία εξέλιξη και υπάρχουν πολλές εφαρμογές διαθέσιμες στον ιστό αλλά και στο Apple Store που καταπιάνονται με τον τομέα αυτό. Όπως προαναφέρθηκε και κατά την εισαγωγή, αυτό που διαφοροποιεί κυρίως την εφαρμογή μας από τις περισσότερες, είναι ότι εκτός από τη δημιουργία της ιστορίας εστιάζουμε στη τροφοδότηση του χρήστη με υλικό και δει πολιτιστικό. Για λόγους πληρότητας, κρίνεται χρήσιμο να αναλύσουμε κάποιες από τις πιο ενδιαφέρουσες εφαρμογές digital storytelling που συναντήσαμε κατά την έρευνα του χώρου και αποτέλεσαν πηγή έμπνευσης αλλά και κίνητρο για σκέψη και διαφοροποίηση.

2.1 Storehouse (εφαρμογή iOS – ιστοσελίδα)

Το Storehouse [8] αποτελεί μια εφαρμογή που επιτρέπει στο χρήστη τη δημιουργία και το διαμοιρασμό ιστοριών. Είναι διαθέσιμη δωρεάν για το iOS και για τον ιστό, ωστόσο η έκδοση που ξεχωρίζει είναι αυτή για τις κινητές συσκευές. Διακρίνεται για το εξαιρετικό της UI, όπως αποδεικνύει και το Apple Design award που κέρδισε το 2014, γεγονός που αναφέρεται και στον πρώτο σύνδεσμο. Χρησιμοποιεί εξαιρετικά τα animations για να αποτελέσει μια πολύ όμορφη εμπειρία για το χρήστη, κάτι που αποτελεί έντονη τάση από το iOS 7 και μετέπειτα. Παρέχει τη δυνατότητα της δημιουργίας ιστορίας και το διαμοιρασμό της, όπως θα περιμέναμε, ωστόσο για την άντληση υλικού βασίζεται σε ότι υπάρχει στη συσκευή του χρήστη. Συγκεκριμένα, επιτρέπει την εισαγωγή φωτογραφιών και βίντεο από τη βιβλιοθήκη του και κείμενο που μπορεί να συντάξει ο ίδιος. Επιπλέον, δεν είναι εφικτή η ηχογράφηση, ενώ ενδιαφέρον παρουσιάζει και ο τρόπος που πραγματοποιείται το συντακτικό κομμάτι των ιστοριών. Τα αντικείμενα τοποθετούνται «έξυπνα» από μόνα τους, προσπαθώντας να υπολογίσουν τη θέση με έναν σειριακό τρόπο. Βέβαια, ο χρήστης έχει

επιλογές να κάνει αλλαγές, αλλά όχι με απόλυτη ελευθερία με σκοπό την ταχύτερη ανάπτυξη τις ιστορίας και τη διατήρηση της ποιότητας εικόνας και βίντεο. Στην εφαρμογή μας επιλέχθηκε άλλη μέθοδος, όπου ο χρήστης έχει απόλυτο έλεγχο σχετικά με το που θέλει να τοποθετήσει την εκάστοτε εικόνα ή κείμενο και τι μέγεθος επιθυμεί.

2.2 Storybird (ιστοσελίδα)

Μια άλλη πολύ ενδιαφέρουσα δωρεάν εφαρμογή, η οποία είναι μόνο για τον ιστό είναι το Storybird [9]. Το ιδιαίτερο γνώρισμά της είναι ότι ασχολείται με το reverse story telling, δηλαδή προτρέπει το χρήστη να ακολουθήσει την αντίστροφη διαδικασία από το συνηθισμένο. Του δείχνει διάφορα σχέδια καλλιτεχνών και τον προτρέπει να εμπνευστεί από εκεί για να ξεκινήσει την δική του ιστορία, το δικό του κεφάλαιο, το δικό του ποίημα. Πρόκειται για έναν ιστότοπο με μοντέρνο σχεδιασμό, που εστιάζει στην καλλιτεχνική φύση του story telling και η οργάνωση του περιστρέφεται κυρίως γύρω από τους καλλιτέχνες και τα δημιουργήματα τους. Παρέχει έναν editor ο οποίος έχει διαφορετική λογική (διαθέτει και ένα μικρό preview των σελίδων που φτιάχνουμε) από το Storehouse αλλά και την εφαρμογή μας, καθώς στα websites όπως είναι φυσιολογικό δεν υπάρχει η δέσμευση του χώρου. Η δυνατότητα για αλληλεπίδραση του χρήστη στην τοποθέτηση των αντικειμένων στις σελίδες τις ιστορίες δεν είναι πολύ μεγάλη και μάλλον δεν είναι κάτι στο οποίο θέλει να δώσει τόσο έμφαση ο συγκεκριμένος ιστότοπος. Παρατηρούμε κάποια κοινή λογική με την εφαρμογή μας, στο ότι προσπαθεί να δώσει υλικό για την ιστορία, ωστόσο υπάρχει μια βασική διαφορά. Στην εφαρμογή μας ο χρήστης ξέρει για τι θέλει και πως θα το ψάξει, ενώ εδώ ο στόχος είναι ότι από την περιήγηση στις εικόνες ο ενδιαφερόμενος θα πάρει την ιδέα για την ιστορία του. Εκτός αυτού ο χαρακτήρας του είναι πιο καλλιτεχνικός και όχι πολιτιστικός, όπως της εφαρμογής μας.

2.3 Storify (ιστοσελίδα)

Τέλος, ιδιαίτερη είναι και η προσέγγιση της ιστοσελίδας Storify [10] προς το story telling. Στόχος της είναι να αξιοποιήσει την τεράστια εξάπλωση των μέσων κοινωνικής δικτύωσης και την πληθώρα των δημοσιεύσεων που γίνονται σε αυτά. Υποστηρίζει, ότι υπάρχει πολύ σημαντική πληροφορία σε αυτές, αρκεί να μπορεί κάποιος να ξεχωρίσει αυτό που αξίζει. Για το σκοπό αυτό, επιτρέπει στους χρήστες κατά το κομμάτι δημιουργίας της ιστορίας να συνδεθούν σε μία πληθώρα μέσων κοινωνικής δικτύωσης όπως είναι το twitter, το facebook, το google+, το youtube, το instagram, το flickr (κ.α.) και να σώσουν στην ιστορίας τους τις δημοσιεύσεις, τις φωτογραφίες και τα βίντεο που επιθυμούν, δημιουργώντας την δική τους ιστορία. Το γραφικό περιβάλλον είναι καλοσχεδιασμένο και το κομμάτι της δημιουργίας της ιστορίας αρκετά χρηστικό. Επίσης, παρέχει και ένα πλάνο επί πληρωμή για εταιρίες με

κάποιες εξτρά λειτουργικότητες, όπως ζωντανή ανανέωση των ιστοριών και σχεδιαστική προσαρμογή σύμφωνα με τις απαιτήσεις του πελάτη και βλέπουμε ότι χρησιμοποιείται ήδη από πολλές μεγάλες εταιρίες όπως το BBC και το Yahoo [11].

3

Θεωρητικό - τεχνολογικό υπόβαθρο

Στο κεφάλαιο αυτό θα αναλύσουμε κάποιες βασικές έννοιες που χρησιμοποιούνται συχνά κατά τη διάρκεια της εφαρμογής, ώστε ακόμα και κάποιος αναγνώστης που δεν έχει άμεση επαφή με το περιβάλλον ανάπτυξης εφαρμογής για το iOS να μπορεί να καταλάβει περί τίνος πρόκειται.

3.1 REST και RESTful web services

Τα αρχικά REST σημαίνουν Representational State Transfer. Βασίζεται σε ένα πρωτόκολλο επικοινωνίας client-server, που δεν διατηρεί κατάσταση, που αξιοποιεί τη μνήμη cache του server και σχεδόν πάντα χρησιμοποιεί το πρωτόκολλο HTTP.

Είναι ένα στυλ αρχιτεκτονικής για το σχεδιασμό εφαρμογών που κάνουν διαδικτυακές κλήσεις. Η βασική ιδέα είναι ότι αντί να χρησιμοποιεί κανείς πολύπολοκους μηχανισμούς, όπως το SOAP ή το RPC για την επικοινωνία μεταξύ μηχανημάτων, είναι προτιμότερο να χρησιμοποιεί το απλό HTTP.

Οι Restful εφαρμογές χρησιμοποιούν HTTP requests για να ζητήσουν, να δημιουργήσουν, να ανενεώσουν ή και να διαγράψουν δεδομένα.

Κατά την υλοποίηση των services από πλεύρα server ακολουθούνται κάποιοι βασικοί κανόνες:

- GET αιτήσεις πρέπει να χρησιμοποιούνται μόνο για την αίτηση δεδομένων από την εφαρμογή πελάτη προς το server και όχι για την τροποποίηση της βάσης του server.

- POST αιτήσεις πρέπει να χρησιμοποιούνται όταν η εφαρμογή-πελάτης θέλει να ανεβάσει δεδομένα στη βάση του server.
- DELETE αιτήσεις χρησιμοποιούνται όταν η εφαρμογή θέλει να διαγράψει κάτι.
- UPDATE αιτήσεις όταν θέλει να ανανεώσει κάτι.

Πατατηρούμε λοιπόν ότι υπάρχει μια πλήρης αντιστοιχία ανάμεσα στις μεθόδους του HTTP (GET, POST, DELETE, UPDATE) και στη λογική σχεδιασμού των webservices.

Άρα, με την έννοια RESTful, δεν εννοούμε κάποια συγκεκριμένη τεχνολολογία, αλλά ένα στυλ σχεδιασμού των services. Δεν παρέχεται κάποιο εργαλέιο, αλλά κάποια λογική που απλοποιεί την επικοινωνία. Δεν πρέπει να ξεχνάμε, ότι στην ουσία χρησιμοποιούμε το απλό και γνωστό HTTP.

Παραπάνω είδαμε, πως ανάλογα με τη μέθοδο του HTTP στέλνονται δεδομένα, ανεβαίνουν, διαγράφονται κτλ. Τι μορφή έχουν όμως αυτά τα δεδομένα όταν αποστέλλονται; Υπάρχει μια ποικιλία μορφών που χρησιμοποιούνται όπως HTML, XML, JSON και άλλες. Η μορφή μπορεί να διαφοροποιείται, ανάλογα με τον τύπο της εφαρμογής που θα σχεδιάσουμε και στις mobile εφαρμογές συνήθως χρησιμοποιούνταν XML. Η τάση, όμως, τα τελευταία χρόνια είναι η χρησιμοποίηση της μορφής JSON, κάτι που απλοποιεί πολύ τα πράγματα.

3.2 JSON

Το Json [12] είναι ένα ελεύθερο standard που χρησιμοποιεί κείμενο για να μεταδώσει αντικείμενα δεδομένων, αποτελούμενα από ζεύγη όνομα πεδίου-τιμή πεδίου. Χρησιμοποιείται κυρίως για την επικοινωνία μεταξύ server και web εφαρμογών σαν εναλλακτική του XML, παρουσιάζοντας τα εξής πλεονεκτήματα. Έχει απλούστερη δομή από το XML και επίσης αντιστοιχίζεται αρκετά εύκολα στις δομές δεδομένων που χρησιμοποιούν οι σύγχρονες γλώσσες προγραμματισμού (κάτι που ισχύει και στη περίπτωση της Objective C που χρησιμοποιούμε).

Οι βασικοί τύποι που χρησιμοποιούνται στο JSON είναι οι εξής:

- Number. Ένας προσημασμένος πραγματικός αριθμός που μπορεί να χρησιμοποιήσει την εκθετική απεικόνιση. Δεν επιτρέπει μη αριθμητικές έννοιες όπως το Nan.
- String. Μια αλληλουχία από κανέναν ή περισσότερους χαρακτήρες Unicode. Πρέπει να περιβάλλονται από το χαρακτήρα «“» και υποστηρίζουν τη χρήση του «\» ως χαρακτήρα αποφυγλης.
- Boolean. Επιτρέπονται οι τιμές true ή false.

- Array. Μια αριθμημένη λίστα από καμία ή περισσότερες τιμές οποιαδήποτε επιτρεπτού τύπου. Χρησιμοποιούν τους χαρακτήρες «[», «]» για δήλωση έναρξης και λήξης του πίνακα και οι τιμές που περιέχονται διαχωρίζονται με κόμμα.
- Object. Ένας μη αριθμημένος πίνακας με ζεύγη ονομα πεδίου-τιμή πεδίου.Για την έναρξη ή λήξη τους χρησιμοποιούνται οι χαρακτήρες «[», «]» και τα κλειδιά πρέπει να είναι τύπου string, ενώ οι τιμές οποιουδήποτε επιτρεπτού τύπου.
- Null. Αντιπροσωπεύει την κενή τιμή.

Ακολουθεί ένα παράδειγμα ενός Json αρχείου για να γίνουν κατανοητά τα παραπάνω:

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [],
  "spouse": null
}
```

3.3 Παρουσίαση των Controllers

Η κλάση UIViewController είναι μια πολύ βασική κλάση που ελέγχει την αλληλεπίδραση μεταξύ των δεδομένων της εφαρμογής και του τρόπου παρουσίασης τους, όπως θα δούμε πιο αναλυτικά παρακάτω (βάλε κεφάλαιο) στο περιβάλλον iOS. Στο σημείο αυτό θα εστιάσουμε στην επεξήγηση κάποιων βασικών υποκλάσεων της βασικής αυτής κλάσης, που

χρησιμοποιούνται κυρίως για την περιήγηση και την προβολή δεδομένων, ώστε να καταλαβαίνει ο χρήστης σε τι αναφερόμαστε κατά την παρουσίαση των οθονών της εφαρμογής.

- ***UITabBarController***

Αυτή είναι η κλάση υλοποιεί έναν ιδιαίτερο controller, τύπου radio button που ελέγχει συνήθως τη περιήγηση στις βασικές λειτουργικότητες μιας εφαρμογής. Για κάθε λειτουργικότητα που ελέγχει ο controller αυτός, υπάρχει το αντίστοιχο tab με χαρακτηριστικό εικονίδιο και τίτλο. Εμφανίζεται στο κάτω μέρος της οθόνης και συνήθως είναι εμφανής κατά τις περισσότερες λειτουργίες της εφαρμογής.

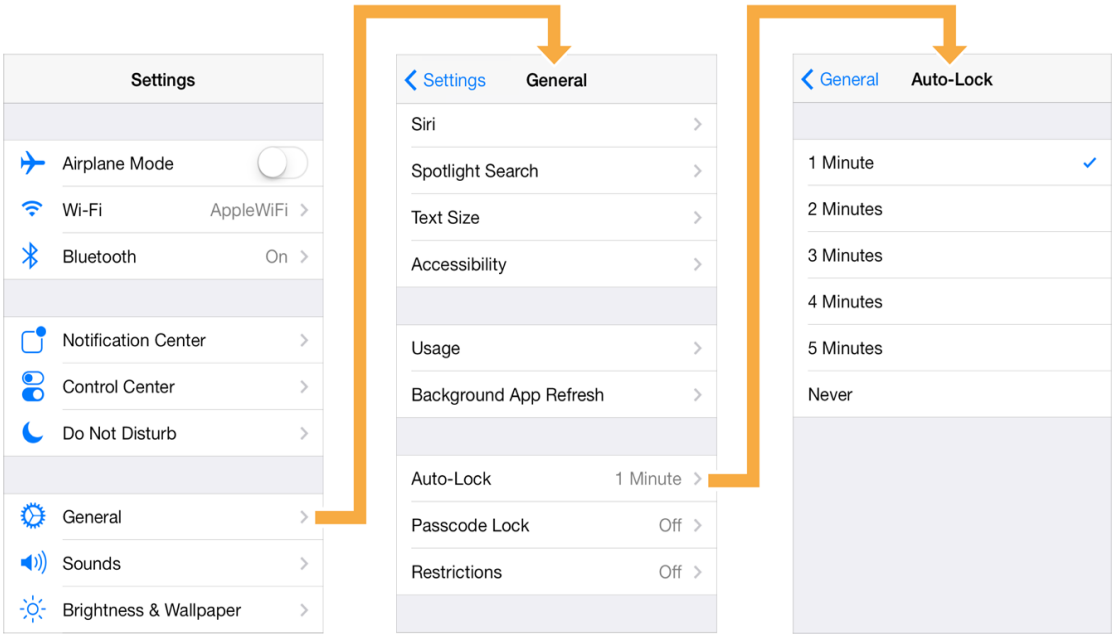
Εικόνα 3.3-1:UITabBarController

Στην ουσία, είναι ένας controller «διαχειριστής» που αποφασίζει ποιος controller θα εμφανίζεται. Για κάθε tab που περιέχει, κρατάει αναφορά σε έναν controller που είναι υπεύθυνος για την παρουσίαση της εκάστοτε λειτουργικότητας – οθόνης. Μόλις ο χρήστης πατήσει σε κάποιο tab ο UITabBarController επιλέγει τον σωστό controller και τον φορτώνει.

- ***UINavigationController***

Και αυτός, όπως και ο UITabBarController, είναι ένας controller περιήγησης, δηλαδή δεν είναι υπεύθυνος για την παρουσίαση μιας οθόνης, όπως οι περισσότεροι controllers, αλλά για την εναλλαγή τους. Σε αντίθεση με τον προαναφερθέντα controller που είναι υπεύθυνος για την μετάβαση από μία βασική λειτουργία σε άλλη, ο UINavigationController είναι συνήθως υπεύθυνος για την εσωτερική περιήγηση μέσα σε μια βασική λειτουργία. Στην πραγματικότητα, λειτουργεί σαν μια στοίβα, στην οποία μπορούμε να κάνουμε push έναν controller όταν θέλουμε να μεταβούμε σε αυτόν και pop όταν θέλουμε να επιστρέψουμε στον προηγούμενο. Κρατάει έναν πίνακα με όλους τους controllers που έχουμε κάνει push, ώστε να είναι εφικτή η προς τα πίσω μετάβαση. Αυτό που φαίνεται από τον UINavigationController είναι μια μπάρα στο πάνω μέρος της οθόνης, με τον τίτλο του

εκάστοτε controller που είναι εμφανής εκείνη τη στιγμή και άρα βρίσκεται στην κορυφή της στοίβας. Επιπλέον, στην μπάρα έχουμε συνήθως ένα κουμπί για να μεταβούμε στον προηγούμενο controller της στοίβας, καθώς και διάφορα άλλα κουμπιά της επιλογής μας, που αφορούν λειτουργίες που επιτρέπει ο τρέχων controller.

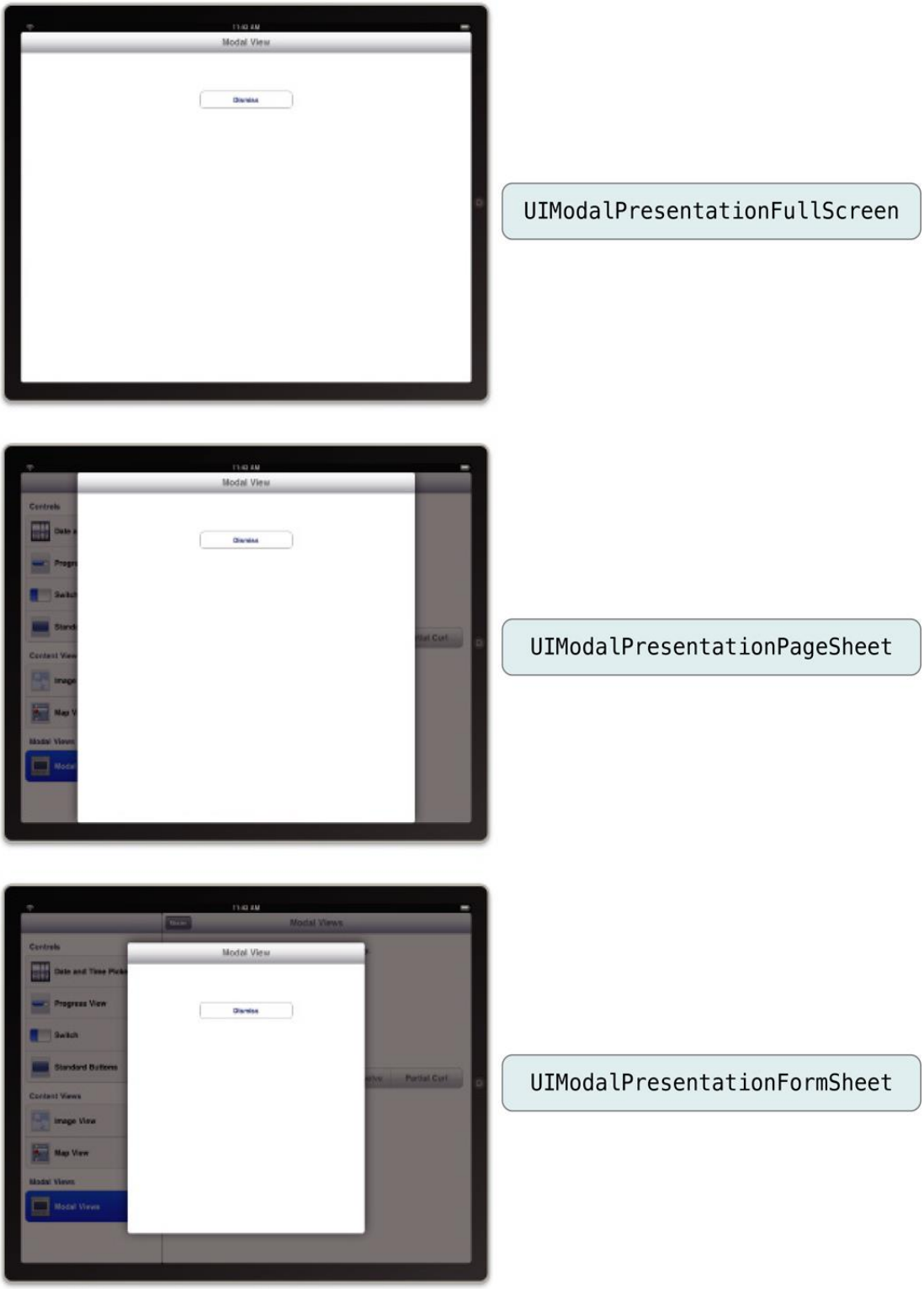


Εικόνα 3.3-2: UINavigationController

Στην αριστερή εικόνα βλέπουμε την μπάρα του UINavigationController στο πάνω μέρος με τον τίτλο Settings και καταλαβαίνουμε ότι έχει φορτώσει τον controller των ρυθμίσεων . Όταν πατήσουμε σε κάποια επιλογή ο UINavigationController φορτώνει κάποιον άλλο controller, τον υπεύθυνο για τις γενικές ρυθμίσεις, ανανεώνεται ο τίτλος στην μπάρα και εμφανίζεται αυτόματα και το κουμπί προς τα πίσω.

- ***Modal Controller***

Με τον όρο modal controller δεν αναφερόμαστε σε κάποια κλάση controller, όπως στις προηγούμενες δύο περιπτώσεις, αλλά σε έναν τρόπο παρουσίασης των controllers. Είδαμε ότι για να μεταβούμε από έναν σε έναν άλλο controller συνήθως διαμεσολαβεί είτε ο UINavigationController, είτε ο UITabBarController. Ωστόσο και κάποια απλή υποκλάση του UIViewController μπορεί να παρουσιάσει έναν άλλο controller «modally». Μπορεί, δηλαδή, να εμφανίσει έναν άλλο controller, επιλέγοντας μάλιστα ανάμεσα σε διαφορετικά στυλ παρουσίασης:



Εικόνα 3.3-3: Διαφορετικά στυλ παρουσίασης ενός Modal Controller

Στη παραπάνω εικόνα βλέπουμε τους διαφορετικούς τρόπους παρουσίασης και παρατηρούμε ότι όταν το στυλ δεν είναι πλήρης οθόνη, η όψη του πίσω controller παρουσιάζεται σκιασμένη.

4

Ανάλυση απαιτήσεων συστήματος

Ο χρήστης μέσω της εφαρμογής αυτής έχει τη δυνατότητα να δημιουργήσει τη δική του ιστορία, να τη μοιραστεί αλλά και να διαβάσει τις ιστορίες που άλλοι χρήστες έχουν δημοσιεύσει. Σημειώνεται, ότι για τις περισσότερες λειτουργίες απαιτείται ο χρήστης αφ' ενός να είναι εγγεγραμμένος και αφ' εταίρου να έχει συνδεθεί με τα διαπιστευτήρια του. Αυτό, γιατί είναι απαραίτητο για συγκεκριμένες διαδικασίες να γνωρίζουμε την ταυτότητα του, ώστε να σωθούν δεδομένα στον προσωπικό του φάκελο και να ανεβούν εκ μέρους του ιστορίες στο server.

Ακολουθεί ακριβής ανάλυση των λειτουργικών και μη λειτουργικών απαιτήσεων του συστήματος.

4.1 Απαιτήσεις συστήματος

4.1.1 Λειτουργικές απαιτήσεις

Οι ακριβείς λειτουργικές απαιτήσεις του συστήματος είναι οι εξής:

- Ο χρήστης μπορεί εύκολα να δει το υλικό των άλλων χρηστών (σύνδεση λογαριασμού μη απαραίτητη)

  Μπορεί:

  - να δει στον τοίχο του τα εξώφυλλα που έχουν δημοσιεύσει οι υπόλοιποι χρήστες.
  - να ανοίξει την ιστορία και να δει το περιεχόμενο της
  - να εκτελέσει αναζήτηση ανάμεσα στις ιστορίες αυτές, με βάση το τίτλο

- Ο χρήστης έχει στη διάθεση του έναν editor για να φτιάξει τις δικές του ιστορίες (σύνδεση λογαριασμού απαραίτητη).

  Συγκεκριμένα δύναται:

  - να εισαγάγει εικόνες από τις αποθηκευμένες τη βιβλιοθήκη φωτογραφιών της συσκευής
  - να εισαγάγει εικόνες, που έχει σώσει ο χρήστης από τις αναζητήσεις του στις ψηφιακές βιβλιοθήκες στο σύστημα αρχείων της εφαρμογής (app's sandbox)
  - να εισαγάγει κείμενο και να το μορφοποιήσει
  - να ηχογραφήσει κάτι και να το συμπεριλάβει και αυτό στην ιστορία του
  - να μεταφέρει και να κάνει resize όλα τα παραπάνω, ώστε να δώσει στην ιστορία του τη μορφή που θέλει
- Ο χρήστης μπορεί να ψάξει για υλικό στις ψηφιακές βιβλιοθήκες DPLA και Europeana. (σύνδεση λογαριασμού απαραίτητη)

  Του δίνεται η δυνατότητα:

  - Να ψάξει ταυτόχρονα και στις δύο βιβλιοθήκες, ή σε μία από τις δύο.
  - Να φιλτράρει την αναζήτηση με βάση τα δικαιώματα επαναχρησιμοποίησης των αντικειμένων, τον τύπο των αποτελεσμάτων και τη χρονική τους τοποθέτηση.
  - Να ακολουθήσει σύνδεσμο από τα αποτελέσματα της αναζήτησης στον ιστότοπο που το φιλοξενεί, καθώς και να σώσει την εικόνα του τοπικά για μετέπειτα χρήση της σε δική του ιστορία.
  - Να δει κάποια μεταδεδομένα με την πρώτη ματιά και να αναπτύξει με ένα πάτημα περαιτέρω όποιο αντικείμενο επιθυμεί.
- Ο χρήστης μπορεί είτε να αποθηκεύσει την ιστορία του τοπικά (drafts) και να την επεξεργάζεται όποτε θέλει, είτε να την ανεβάσει ανά πάσα στιγμή, οπότε και θα γίνει εμφανής στον τοίχο όλων των χρηστών (σύνδεση λογαριασμού απαραίτητη).

4.1.2 Μη λειτουργικές απαιτήσεις

Οι μη λειτουργικές απαιτήσεις είναι οι εξής:

- Για την παράθεση τόσο των ιστοριών, όσο και των αποτελεσμάτων αναζήτησης στις βιβλιοθήκες προτιμήθηκε το μοντέλο του καταρράκτη (waterfall layout), ώστε να επιτευχθεί δυναμική σε μέγεθος παρουσίαση των αντικειμένων, κάτι που επιτρέπει την οπτική προσαρμογή στην πληροφορία, σε αντίθεση με κάποια άλλη στατική όψη(card view, table view), που μένει ίδια για κάθε αντικείμενο.

- Ο χρήστης ανά πάσα στιγμή έχει τη δυνατότητα να περιηγηθεί στις βασικές λειτουργικότητες της εφαρμογής, με τη χρήση του UITabBar, το οποίο είναι από παντού προσβάσιμο.
- Ανάλογα με τις χειρονομίες του χρήστη (scroll down) εξαφανίζονται οι μπάρες καθοδήγησης (navigation bars) και το UITabBar, ώστε να αξιοποιείται στο έπακρο η οθόνη της συσκευής για να δείξει περιεχόμενο. Βέβαια, με την αντίθετη κίνηση ξαναεμφανίζονται, ώστε να μη στερείται τη δυνατότητα της πλοήγησης.

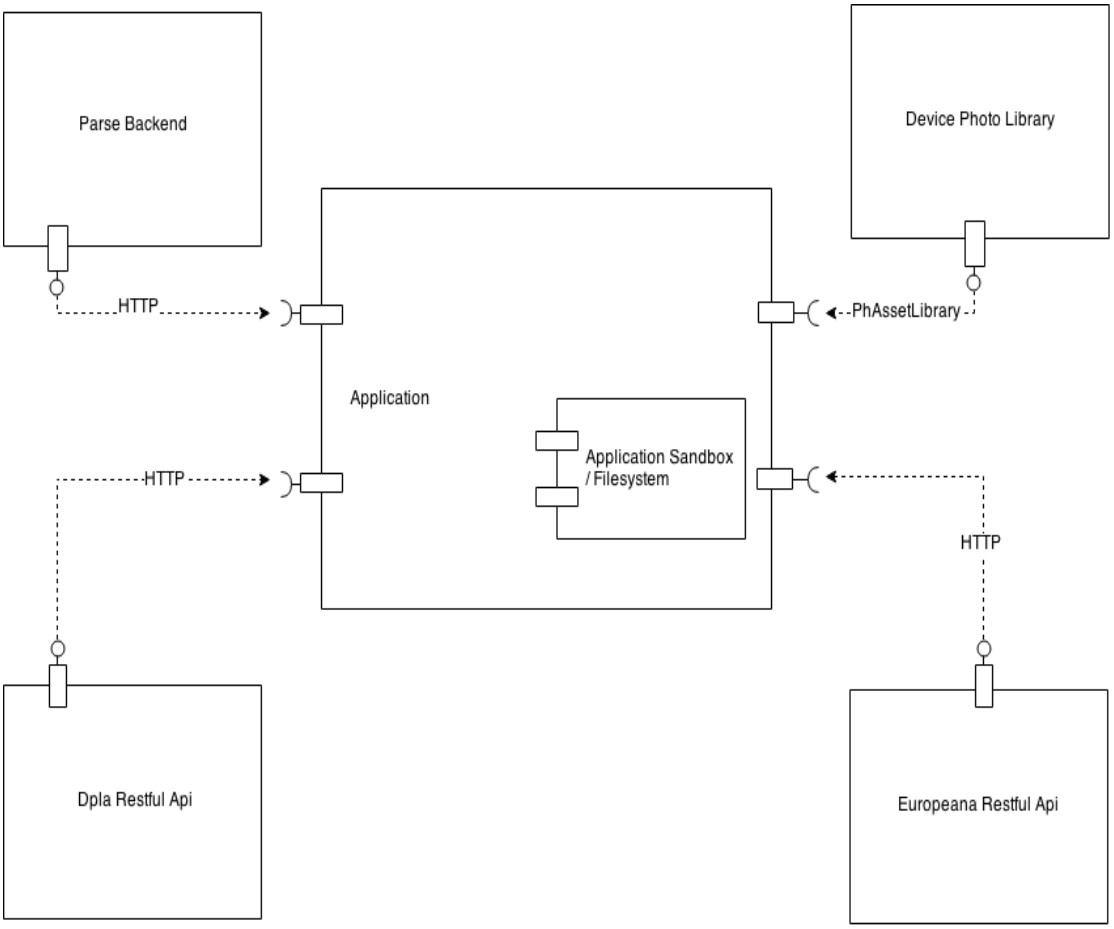
4.2 Αρχιτεκτονική του συστήματος

Για την ανάπτυξη της εφαρμογής αυτής χρησιμοποιήθηκαν οι εξής βασικές δομικές μονάδες:

- Ένα backend (Parse) στο οποίο ανεβάζονται και αποθηκεύονται οι ιστορίες των χρηστών, και από το οποίο ο κάθε χρήστης κατεβάζει όλες τις ιστορίες που εμφανίζονται στο τοίχο του.
- Το τοπικό σύστημα αρχείων της εφαρμογής, όπου σώζονται τόσο εικόνες από τα αποτελέσματα αναζήτησης που ο χρήστης θέλει να αποθηκεύσει, όσο και τα drafts των ιστοριών.
- Τα Restful Apis της Europeana και του DPLA, στα οποία γίνονται κλήσεις για την αναζήτηση πολιτιστικού υλικού από την εφαρμογή.
- Η βιβλιοθήκη εικόνων της συσκευής του χρήστη, από όπου μπορεί να αντλήσει υλικό για την ιστορία του(και βρίσκεται έξω από το sandbox της εφαρμογής)

Και φυσικά η ίδια η εφαρμογή, που αλληλεπιδρά με όλα τα παραπάνω.

Ακολουθεί ένα component diagram για να γίνουν πιο κατανοητά τα παραπάνω:



Εικόνα 4.2-1: Component Diagram

4.3 Σχεδίαση εφαρμογής

Για τη σχεδίαση της εφαρμογής χρησιμοποιήθηκε το προγραμματιστικό μοτίβο MVC (Model View Controller).

4.3.1 Σχεδιαστικό μοτίβο MVC

Το MVC είναι το βασικό σχεδιαστικό μοτίβο που επικρατεί στο iOS, είναι το προτεινόμενο για την αρχιτεκτονική σχεδίαση και τα frameworks της Apple είναι κατασκευασμένα με βάση αυτό [13]. Τα οφέλη από τη χρησιμοποίησή του είναι πολλαπλά, καθώς οδηγεί σε πιο καθαρό και επαναχρησιμοποιήσιμο κώδικα με ξεκάθαρες αρμοδιότητες. Κατατάσσει τα αντικείμενα μιας εφαρμογής σε τρεις κατηγορίες: model, view ή controller. Κάθε κατηγορία έχει σαφή όρια και επικοινωνεί με τις άλλες με σεβασμό ως προς τα όρια αυτά.

Πιο αναλυτικά:

- Model:

Τα αντικείμενα που ανήκουν στην κατηγορία Model απεικονίζουν τα δεδομένα που αφορουν την κύρια λογική της εφαρμογής, καθώς και την επεξεργασία/υπολογισμό τους. Για παράδειγμα, σε ένα παιχνίδι ένας χαρακτήρας μπορεί να αποτελεί ένα αντικείμενο του Model, ή σε μια εφαρμογή τηλεφωνικού καταλόγου, μια επαφή. Τέτοια αντικείμενα μπορούν να έχουν σχέσεις 1-1 ή και 1-N μεταξύ τους και συνήθως η απεικόνιση τους σχηματίζει γράφο. Επειδή συγκεντρώνουν τη λογική μιας συγκεκριμένης περιοχής, μπορούν να επαναχρησιμοποιηθούν σε άλλα αντίστοιχα προβλήματα. Ιδανικά, δεν πρέπει να έχουν διαδυνδέσεις με αντικείμενα της κατηγορίας View και δεν πρέπει να εμπεριέχουν καθόλου πληροφορία για τον τρόπο απεικόνισης ή παρουσίασης των δεδομένων στο χρήστη.

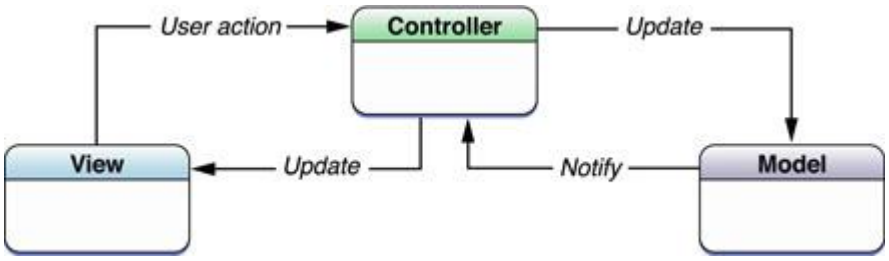
- View

Τα αντικείμενα της κατηγορίας αυτής είναι αυτά ουσιαστικά που βλέπει ο χρήστης. Ξέρουν να «ζωγραφίζουν» τον εαυτό τους και μπορούν να ανταποκριθούν σε δράσεις των χρηστών. Ο κύριος σκοπός τους είναι να απεικονίζουν δεδομένα από το Model και να καθιστούν δυνατή την επεξεργασία τους από το χρήστη, χωρίς όμως να έχουν άμεση διασύνδεση με αυτά. Ο ενδιάμεσος που βοηθά την επικοινωνία View – Model είναι ο Contoller.

- Controller

Τα αντικείμενα controller είναι αυτό που προαναφέραμε, ο ενδιάμεσος μεταξύ View και Model. Είναι ένα μέσο που ενημερώνει ποια αντικείμενα του View πρεπει να ενημερωθούν λόγω αλλαγής στο Model, αλλά και το αντίστροφο. Συντονίζουν διάφορα κομμάτια της εφαρμογής και διευθύνουν τον κύκλο ζωής των αντικειμένων.

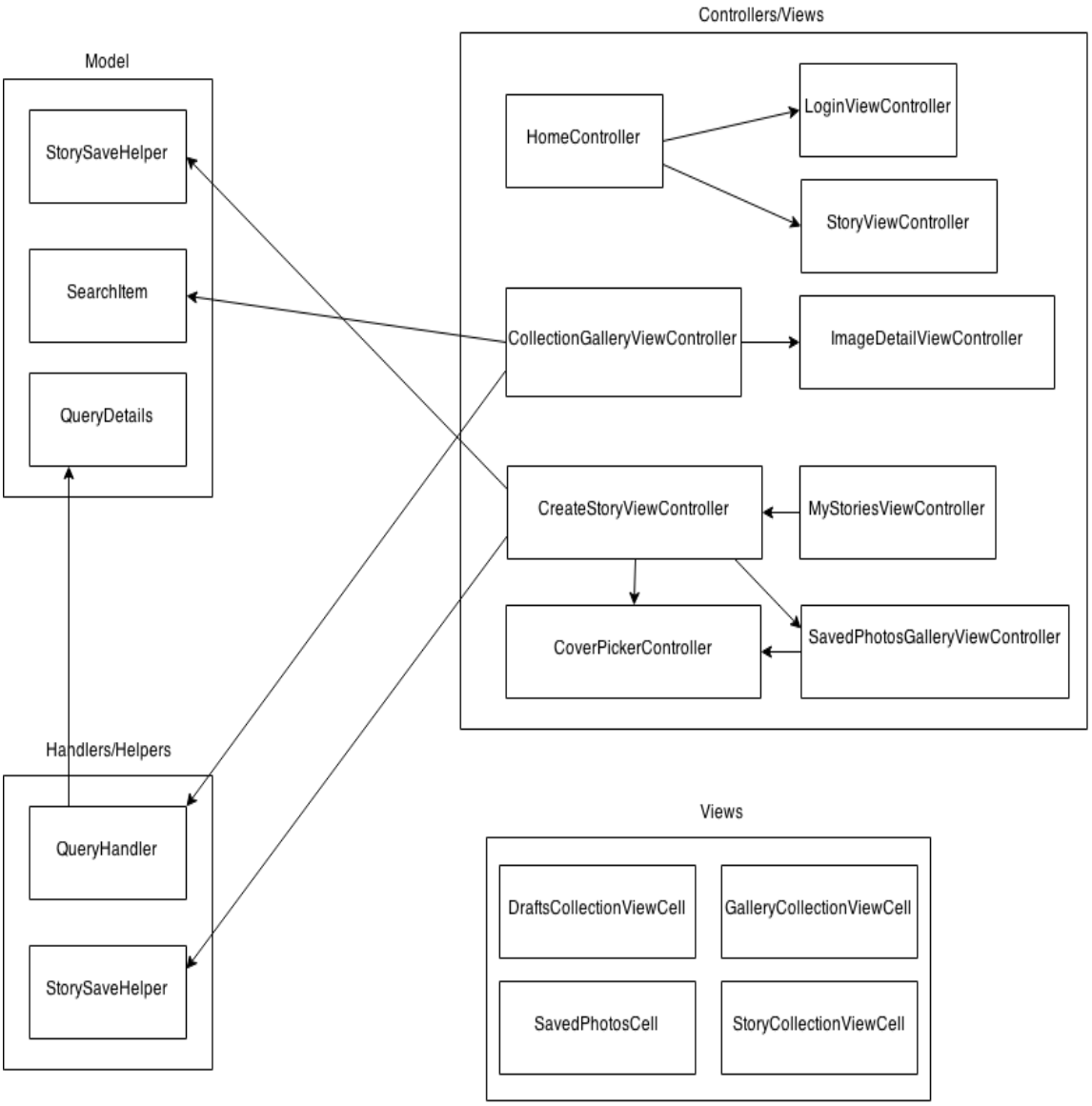
Το παρακάτω σχήμα είναι ενδεικτικό του τρόπου λειτουργίας του μοτίβου αυτού.



Εικόνα 4.3-1: MVC μοτίβο

4.3.2 Domain Class Diagram

Όπως προαναφέρθηκε για την σχεδίαση της εφαρμογής αυτής χρησιμοποιήθηκε το μοντέλο MVC. Ακολουθεί ένα Domain Class διάγραμμα που συνοψίζει τις βασικές κλάσεις της εφαρμογής (δεν εμφανίζονται όλες) και μια συνοπτική περιγραφή της λειτουργίας αυτών.



Εικόνα 4.3-2: Domain Class Diagram

Παρατηρήσεις για το διάγραμμα:

Παρότι όπως είδαμε στο μοτίβο MVC, το View και το Controller είναι διαφορετικές οντότητες, στο διάγραμμα βλέπουμε ότι το κουτί των Controllers εμπεριέχει και τη λέξη View στο τίτλο του. Αυτό γιατί στο iOS οι controllers (που άλλωστε by default ονομάζοντα ViewControllers) πάνε «πακέτο» με το View που τους ανήκει. Δηλαδή, έχουν από δημιουργίας τους reference στο view που ελέχγουν. Έτσι, συνηθίζεται στην κλάση των

controllers να υπάρχει και κώδικας που διαμορφώνει τον τρόπο αναπαράστασης του view τους, αρκεί να μην πρόκειται για πολύπλοκες ιεραρχίες όψεων, όπου και θα ήταν προτιμότερο να τοποθετείται ο κώδικας παρουσίασης σε ξεχωριστές κλάσεις που ανήκουν στην κατηγορία View.

4.4 Σύντομη Περιγραφή Βασικών κλάσεων

4.4.1 HomeViewController

Είναι η αρχική σελίδα και το πρώτο tab που βλέπει ο χρήστης. Είναι ο "τοίχος" με τις ιστορίες όλων των χρηστών. Αφού φορτώσει ο controller κάνει μια κλήση στο Parse για να φέρει όλες τις ιστορίες. Παρουσιάζει τα εξώφυλλα των ιστοριών, τα οποία έχουν δυναμικό μέγεθος που καθορίζεται από την εικόνα τους και την περιγραφή τους, υλοποιώντας το waterfall layout. Επιπλέον, δίνει τη δυνατότητα για αναζήτηση στις ιστορίες και εάν ο χρήστης πατήσει σε κάποια από αυτές φορτώνει τον StoryViewController.

Αρχικά η μόνη λειτουργικότητα που έχει ο επισκέπτης χρήστης στη διάθεση του είναι αυτό το tab. Για να ξεκλειδώσει και τα υπόλοιπα (create Story, Drafts) πρέπει να συνδεθεί με το λογαριασμό του. Μπορεί να πατήσει λοιπόν το κουμπί του sign in και να καλέσει τον LoginViewController που αναλαμβάνει τη σύνδεση του χρήστη

4.4.2 StoryViewController

Φορτώνεται από τον HomeViewController για να δείξει ολήκληρη την ιστορία που έχει δημοσιεύσει ο χρήστης. Παρέχει επίσης τη δυνατότητα να παίξει τα αρχεία ήχου που περιέχει η ιστορία καθώς και και για κάθε εικόνα να οδηγήσει το χρήστη στον ιστότοπο που φιλοξενεί το συγκεκριμένο αντικείμενο (εφόσον πρόκειται για αντικείμενο που προέρχεται από τις ψηφιακές βιβλιοθήκες).

4.4.3 LoginViewController

Είναι ο controller που αναλαμβάνει τη σύνδεση ή εγγραφή του χρήστη. Ο χρήστης εισάγει τον κωδικό και το όνομα χρήστη που επιθυμεί και ο controller κάνει κλήση στο Parse προκειμένου να εγγράψει τον χρήστη ή να δεί αν τα διαπιστευτήρια του είναι σωστά.

4.4.4 CreateStoryViewController

Είναι το δεύτερο tab της εφαρμογής και στην ουσία αποτελεί τον editor που επιτρέπει στο χρήστη να φτιάξει τη δική του ιστορία. Του δίνει τη δυνατότητα να εισάγει κείμενο, εικόνες καθώς και να ηχογραφήσει το δικό του μήνυμα. (Για την εισαγωγή εικόνων φορτώνεται ο SavedPhotosGalleryViewController). Όλα τα παραπάνω μπορεί να τα μετακινήσει σε όποιο σημείο της ιστορίας θέλει και να τα μεγενθύνει ή μικραίνει με απλές επαφές του χρήστη.

Επιπλέον, του δίνει τη δυνατότητα να αναζητήσει υλικό για την ιστορία του στις ψηφιακές βιβλιοθήκες πατώντας στο εικονίδιο της αναζήτησης και εμφανίζοντας τον CollectionGalleryViewController. Μόλις τελείωσει την ιστορία μπορεί, είτε να την σώσει τοπικά, είτε να την ανεβάσει στο server (Parse). Και στις δύο περιπτώσεις φορτώνεται ο CoverPickerController για να φτιάξει ο χρήστης το εξώφυλλο της ιστορίας του, που αποτελεί και το τελικό στάδιο της δημιουργίας της ιστορίας του. Μόλις ολοκληρωθεί το εξώφυλλο, καλείται ο StorySaveHelper για να σώσει την ιστορία είτε τοπικά, είτε να την ανεβάσει στο server.

4.4.5 CoverPickerController

Εμφανίζεται όταν ο χρήστης έχει ολοκληρώσει τη δημιουργία της ιστορίας του και τον καθοδηγεί στο να φτιάξει το εξώφυλλο της ιστορίας του. Του ζητά έναν τίτλο και μια σύντομη περιγραφή και τον παροτρύνει να διαλέξει μια φωτογραφία για το εξώφυλλο. Για την επιλογή της φωτογραφίας καλεί τον SavedPhotosGalleryViewController.

4.4.6 SavedPhotosGalleryViewController

Παρουσίαζει σε ένα UICollectionView με μορφή πλέγματος τις φωτογραφίες από τη συσκευή του χρήστη, καθώς και τις φωτογραφίες από αντικείμενα αναζήτησης που θέλησε να σώσει ο χρήστης. Του επιτρέπει να επιλέξει όποια επιθυμεί, είτε για εξώφυλλο, είτε για να την εισάγει στην ιστορία του.

4.4.7 StorySaveHelper

Όταν ο χρήστης τελείωσει με τη δημιουργία της προσωπικής του ιστορίας έχει δύο επιλογές, είτε να τη σώσει τοπικά, είτε να την ανεβάσει. Η κλάση αυτή υλοποιεί αυτές τις δύο δυνατότητες και παρέχει στην ουσία το απαραίτητο mapping, ώστε η ιστορία του χρήστη να μεταβεί από την οπτική της μορφή σε αναπαράσταση κατάλληλη για αποθήκευση ή ανέβασμα.

4.4.8 CollectionGalleryViewController

Αυτός είναι ο controller που εμφανίζεται όταν θέλουμε να κάνουμε αναζήτηση στις ψηφιακές βιβλιοθήκες. Διαθέτει πεδία για αναζήτηση, καθώς και διάφορα φίλτρα με τα οποία μπορεί ο χρήστης να καθοδηγήσει την αναζήτηση. Περιέχει ένα collection view, στο οποίο εμφανίζει τα αποτελέσματα της αναζήτησης με κάποια εικόνα και ορισμένα μεταδεδομένα. Η παρουσίαση των αποτελεσμάτων γίνεται με ένα δυναμικό layout, γνωστό ως waterfall layout, κάτι που κρίθηκε απαραίτητο ώστε η παρουσίας να προσαρμόζεται στις ανάγκες του κάθε αντικειμένου. Χρησιμοποιεί την κλάση QueryHandler για τις κλήσεις της αναζήτησης και των ImageDetailController σε περίπτωση που ο χρήστης πατήσει πάνω σε κάποια αντικείμενο του collectionView.

4.4.9 MyStoriesViewController

Αυτός ο controller αποτελεί το τρίτο tab της εφαρμογής και περιέχει τις αποθηκευμένες τοπικά ιστορίες του χρήστη. Αποτελείται από ένα UICollectionView που παρουσιάζει σε πλέγμα τα εξώφυλλα των ιστοριών του χρήστη και του επιτρέπει να τις επεξεργαστεί ξανά πατώντας πάνω σε όποια ιστορία επιθυμεί. Με το που πατά σε κάποια ιστορία ο χρήστης φορτώνεται ο CreateStoryViewController και αρχικοποιείται με την ιστορία του.

4.4.10 Query Details – Query Handler

Το API της Europeana και αυτό του DPLA έχουν σχετικό Documentation για το πώς πρέπει να γίνονται οι κλήσεις, με τι παραμέτρους, τι φίλτρα και διάφορα tokens για αυθεντικοποίηση του χρήστη. Οι δύο αυτές κλάσεις, λοιπόν. υλοποιούν το παραπάνω και είναι άρρηκτα συνδεδεμένες.

Η κλάση Query Details αναλαμβάνει να φέρει τις παραμέτρους και τα φίλτρα αναζήτησης σε μορφή κατάλληλη, ώστε να χρησιμοποιηθεί από το κάθε API.

Η κλάση Query Handler είναι αυτή που κάνει την κλήση στα APIs χρησιμοποιώντας την προαναφερθείσα κλάση για να κατασκευάσει το HTTP Request, να αναλάβει το paging των αντικειμένων και την αυθεντικοποίηση που ζητούν τα APIs. Αυτή είναι η κλάση που βλέπει και χρησιμοποιεί ο controller για την αναζήτηση στις ψηφιακές βιβλιοθήκες. (CollectionGalleryViewController)

4.4.11 SearchItem

Αυτή η κλάση αποτελεί το αντικείμενο της αναζήτησης που επιστρέφεται από τις ψηφιακές βιβλιοθήκες. Έχει κάποια γενικά πεδία που συναντώνται και στις δύο περιπτώσεις αναζήτησης (DPLA και Europeana) και κάποια ειδικά, ανάλογα με της ιδιαιτερότητες της καθ μίας. Για κάθε ψηφιακή βιβλιοθήκη γνωρίζει πώς από τη μορφή που παρέχουν τα APIs να πάρει τα στοιχεία που ενδιαφέρουν την εφαρμογή.

4.4.12 ImageDetailViewController

Είναι ο controller που παρουσιάζει το εκάστοτε αποτέλεσμα αναζήτησης με περισσότερες λεπτομέριες αφού ο χρήστης το επιλέξει. Δείχνει ένα μόνο αντικείμενο και είναι modal, με την έννοια ότι εμφανίζεται πάνω από τον CollectionGalleryViewController και δεν καταλαμβάνει ολόκληρο της οθονής. Στην ουσία αποκαλύπτει περισσότερα μεταδεδομένα στο χρήστη, από ό,τι μπορεί να δει στο collectionView όπου παρουσιάζονται όλα τα αντικείμενα.

4.5 Περιπτώσεις χρήσης της εφαρμογής

Περίπτωση χρήσης 1: ***Εγγραφή***

Αυτή δεν αποτελεί μια κύρια χρήση της εφαρμογής, αλλά αποτελεί απαραίτητο βήμα για άλλες, οπότε και αναλύεται εδώ προς αποφυγήν επανάληψης.

Κύριοι Δράστες:

Χρήστης, Parse backend

Προϋποθέσεις:

-

Βασική ροή:

1. Ο χρήστης πατάει το κουμπί για τη σύνδεση με το λογαριασμό του,

2. Εμφανίζεται modal controller στον οποίο ο χρήστης εισάγει το κωδικό και το όνομα του και πατά το κουμπί για να συνδεθεί.

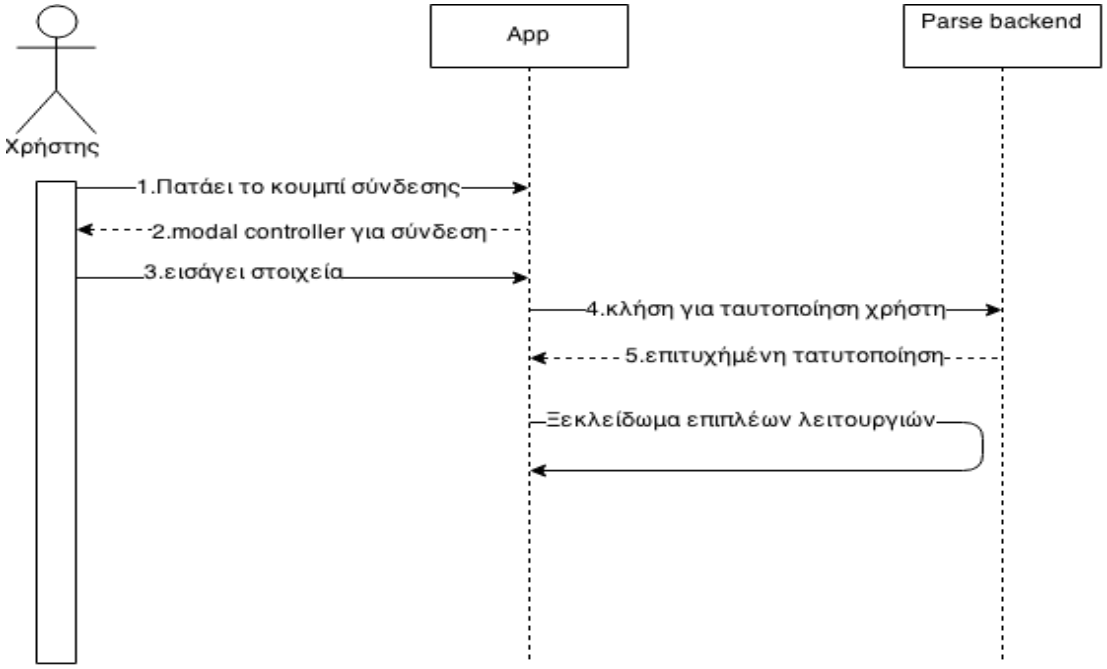
3. Ο χρήστης εισάγει τα στοιχεία του και πατάει το κουμπί της σύνδεσης.

4. Γίνεται κλήση στο Parse backend για να επαληθευτούν τα στοιχεία του χρήστη.

5. Κλήση επιτυχής (χρήστης εγγεγραμμένος και σωστά διαπιστευτήρια). Η εφαρμογή ξεκλειδώνει τις επιπλέον λειτουργίες (δημιουργία ιστορίας και πρόχειρα).

Εναλλακτική ροή:

5α. Κλήση ανεπιτυχής. Ο χρήστης ενημερώνεται με κατάλληλο μήνυμα και προτρέπεται να ξαναδοκιμάσει ή να εγγραφεί.



Εικόνα 4.5-1: Εγγραφή χρήστη

Περίπτωση χρήσης 2: ***Περιήγηση / αναζήτηση στις ιστορίες των άλλων χρηστών***

Κύριοι Δράστες:

Χρήστης, Parse backend

Προϋποθέσεις:

-

Βασική ροή:

1. Ο χρήστης εκκινεί την εφαρμογή και βρίσκεται στο πρώτο tab με όνομα home.

2. Η εφαρμογή κάνει κλήση στο Parse για να πάρει τις δημοσιευμένες ιστορίες των χρηστών.

3. Το Parse στέλνει πίσω την απάντηση με τις ιστορίες των χρηστών.

4. Η εφαρμογή προβάλλει τις ιστορίες.

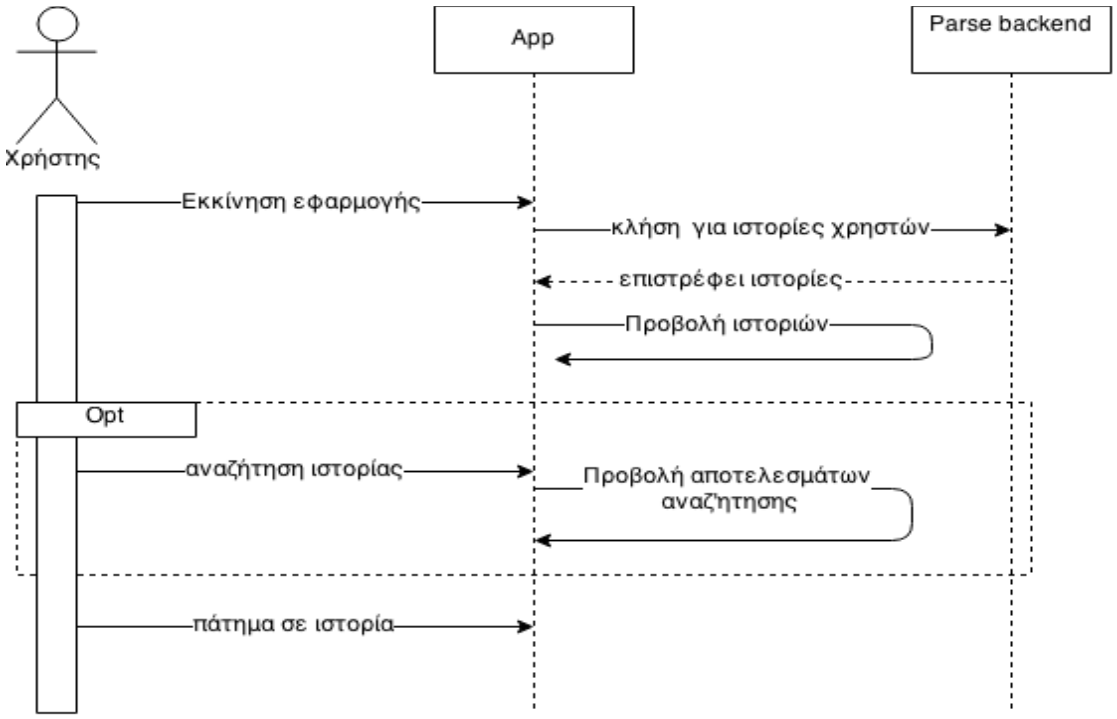
5. Ο χρήστης επιλέγει μια ιστορία που επιθυμεί και την ανοίγει.

6. Περιηγείται στις λεπτομέριες της ιστορίας.

Εναλλακτική ροή:

5α1. Ο χρήστης θέλει να εκτελέσει κάποια ιστορία, οπότε εισάγει κάποια λέξη κλειδί και στο πεδίο και πατάει το κουμπί search.

5α2. Η εφαρμογή φιλτράρει τα δεδομένα με βάσει τις λέξεις που εισήγαγε ο χρήστης και εμφανίζει τα αντίστοιχα αποτελέσματα.



Εικόνα 4.5-2: Περιήγηση / αναζήτηση στις ιστορίες των άλλων χρηστών

Περίπτωση χρήσης 3: Αναζήτηση σε ψηφιακές βιβλιοθήκες για υλικό

Κύριοι Δράστες:

Εγγεγραμμένος χρήστης, Europeana API, DPLA API, Parse backend , Filesystem εφαρμογής, Browser εφαρμογής

Προϋποθέσεις:

Ο χρήστης να έχει εγγραφεί στην εφαρμογή και να έχει συνδεθεί με τα διαπιστευτήρια της εγγραφής του. Βλέπε περίπτωση χρήσης 1.

Βασική ροή:

1. Ο χρήστης επιλέγει το tab “create story” από τον tabBarController που βρίσκεται στο κάτω μέρος της οθόνης και στη συνέχεια πατάει το κουμπί με το εικονίδιο της αναζήτησης

2. Διαλέγει τα φίλτρα αναζήτησης που επιθυμεί και εκτελεί την αναζήτηση.

3. Γίνεται η αντίστοιχη κλήση στο Europeana και/ή Dpla Api (ανάλογα με το φίλτρο αναζήτησης που έχει επιλέξει)

4. Η εφαρμογή παρουσιάζει τα δεδομένα στο χρήστη.

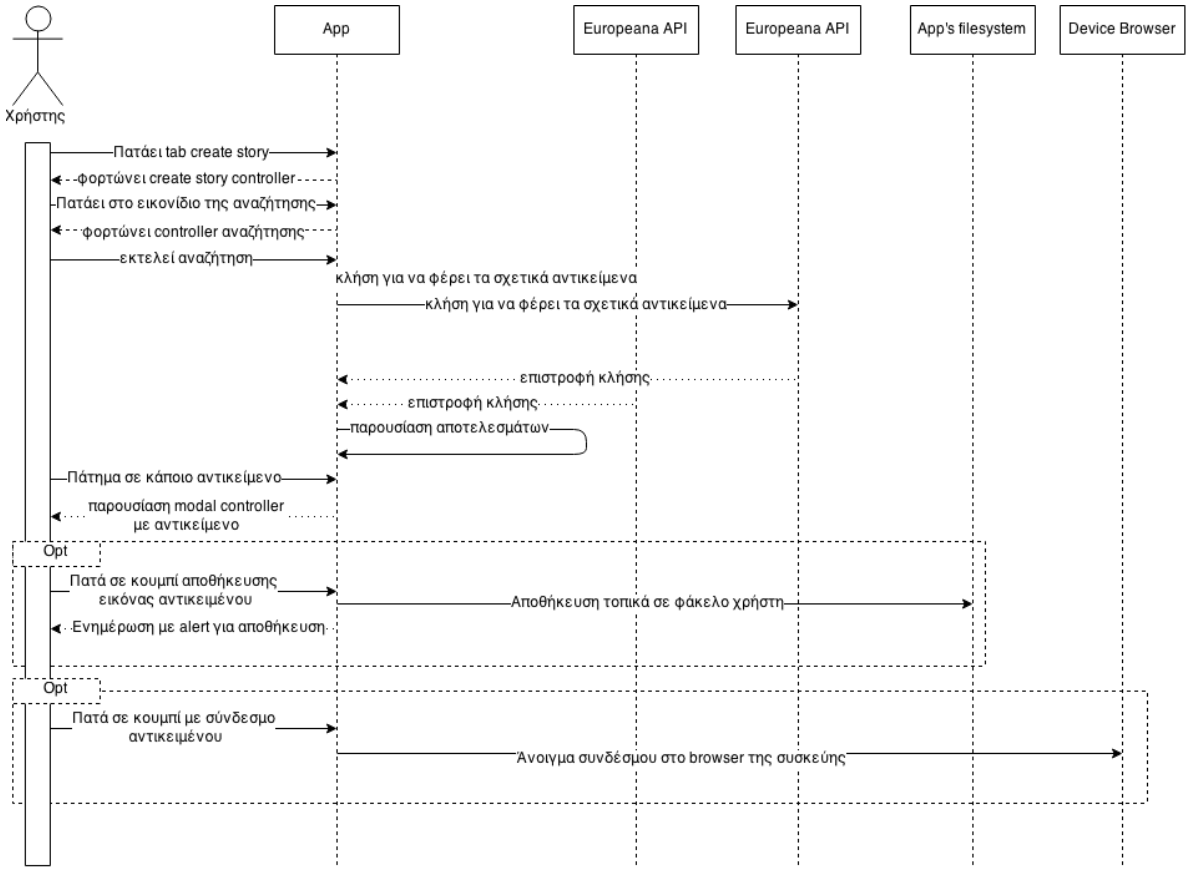
5. Ο χρήστης πατάει σε όποια αντικείμενο του προκαλέι ενδιαφέρον και εμφανίζεται ένας modal controller με περισσότερες λεπτομέριες για το συγκεκριμένο αντικείμενο. Στη συνέχεια αν θέλει μπορεί:

6.1. Να αποθηκεύσει τη φωρογραφία του τοπικά, οπότε πατά το κουμπί της αποθήκευσης.

6.1α. Να ανοίξει στο browser τη σελίδα του ιστότοπου που φιλοξενεί το αντικείμενο , οπότε και πατά το κουμπί με το link.

6.2. Η εφαρμογή σώζει την φωτογραφία στο φάκελο του χρήστη και ενημερώνει το χρήστη για την επιτυχημένη αποθήκευση.

6.2α. Η εφαρμογή ανοίγει τη σελίδα στο browser της συσκευής.



Εικόνα 4.5-3: Αναζήτηση σε ψηφιακές βιβλιοθήκες για υλικό

Περίπτωση χρήσης 4: Δημιουργία *ιστορίας*

Κύριοι Δράστες:

Εγγεγραμμένος χρήστης, Filesystem εφαρμογής

Προϋποθέσεις:

Ο χρήστης να έχει εγγραφεί στην εφαρμογή και να έχει συνδεθεί με τα διαπιστευτήρια της εγγραφής του. Βλέπε περίπτωση χρήσης 1.

Βασική ροή:

1. Ο χρήστης επιλέγει το tab “create story” από τον tabBarController και εμφανίζεται ο controller του editor.

2. Ο χρήστης βάζει υλικό στην ιστορία του.

2.1 Επιλέγει να εισάγει εικόνες, οπότε και πατά το εικονίδιο του gallery.

2.2 Εμφανίζεται ένας controller με τις αποθηκευμένες εικόνες από τις αναζητήσεις του χρήστη, καθώς και από τις εικόνες της συσκευής του.

2.3 Επιλέγει την εικόνα που επιθυμεί και την εισάγει στην ιστορία του. Μπορεί να την κάνει drag και resize όπου αυτός επιθυμεί.

Εναλλακτικές διαδρομές 2.1

2.1α1. Επιλέγει να εισάγει κείμενο, οπότε και πατά το εικονίδιο του κειμένου.

2.1α2. Εμφανίζεται ένα παραλληλόγραμο με κείμενο στην οθόνη του.

2.1α3. Εισάγει το κείμενο που θέλει, το μορφοποιεί και το κάνει drag και resize για να το τοποθετήσει όπως θέλει.

2.1β1. Επιλέγει να εισάγει μια προσωπική ηχογράφηση και πατά το εικονίδιο της ηχογράφησης και η ηχογράφηση ξεκινά.

2.1β2. Μιλά για όσο θέλει και στη συνέχεια πατάει το stop.

2.1β3. Η εφαρμογή σώζει την ηχογράφηση του τοπικά και του εμφανίζει ένα εικονίδιο με την ηχογράφηση του, το οποίο μπορεί να τοποθετήσει σε όποιο σημεί της ιστορίας επιθυμεί.

3. Ο χρήστης πατά το κουμπί save για να σώσει την ιστορία του .

3.1. Εμφανίζεται ένας modal controller και προτρέπει το χρήστη να εισάγει τίτλο και μια σύντομη περιγραφή για την ιστορία του

3.2. Η εφαρμογή σώζει την ιστορία του τοπικά στο φάκελο του χρήστη και καθαρίζει τον editor.

Εναλλακτική ροή:

3.α. Ο χρήστης επιλέγει να ανεβάσει την ιστορία του, οπότε πατά το κουμπί με το εικονίδιο του upload.

3.α1. Εμφανίζεται ένας modal controller και προτρέπει το χρήστη να εισάγει τίτλο και μια σύντομη περιγραφή για την ιστορία του

3.α2. Η εφαρμογή κάνει μια κλήση στο Parse και ανεβάζει τις πληροφορίες σχετικά με τη δομή της ιστορίας. Στη συνέχεια για όλα τα αρχεία του χρήστη (εικόνες από τη συσκευή του και ηχογραφήσεις) πραγματοποιεί ισάριθμες κλήσεις για να ανεβάσει και το υλικό αυτό στο server.

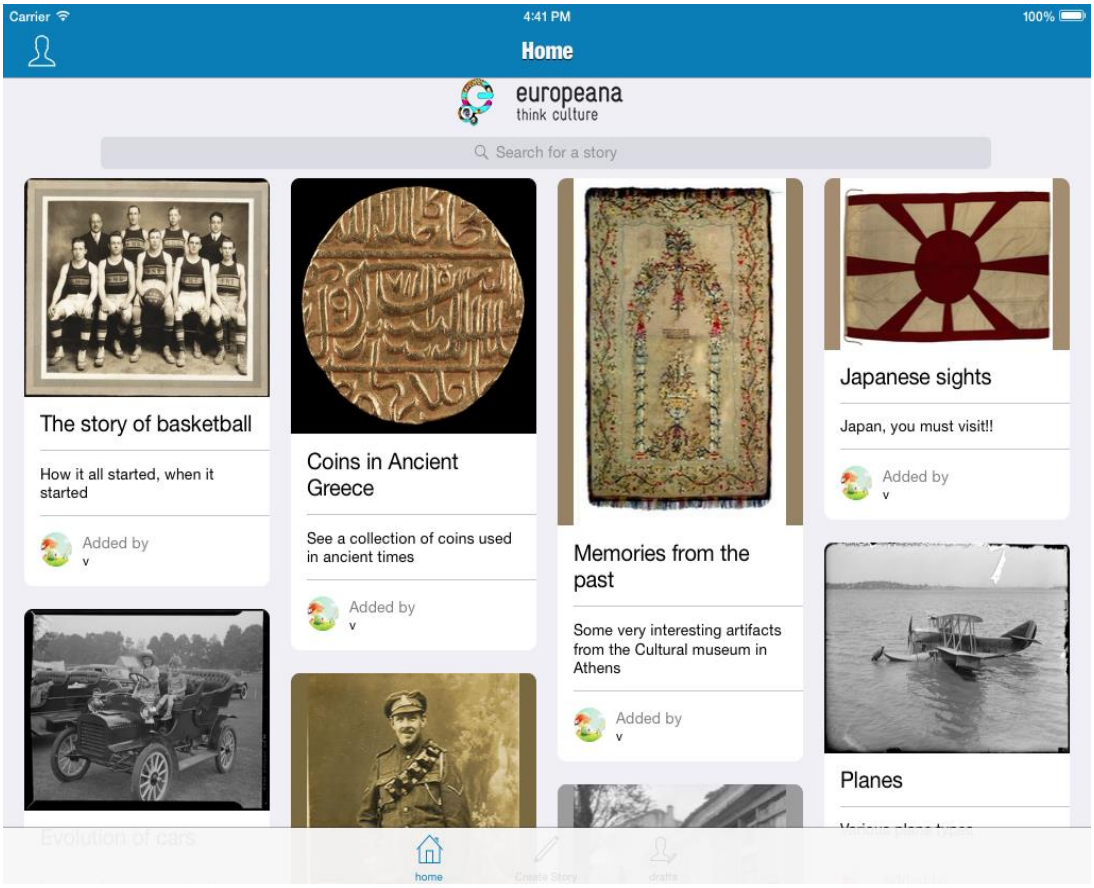
5

Περιγραφή λειτουργικότητας εφαρμογής

Εδώ θα παρουσιαστούν όλες οι λειτουργίες τις εφαρμογής, μαζί με τα ακριβή βήματα που απαιτούνται σε κάθε περίπτωση και τις απαραίτητες συνοδευτικές εικόνες. Οι εικόνες έχουν ληφθεί από συσκευή iPad 3rd generation. Σημειώνεται ότι η εφαρμογή τρέχει σε λειτουργικό iOS 8 και τρέχει σε landscape mode.

5.1 Αρχική οθόνη εφαρμογής

Αυτή είναι η πρώτη οθόνη που βλέπει ο χρήστης και στην ουσία αποτελεί και το πρώτο tab από τα τρία της εφαρμογής.



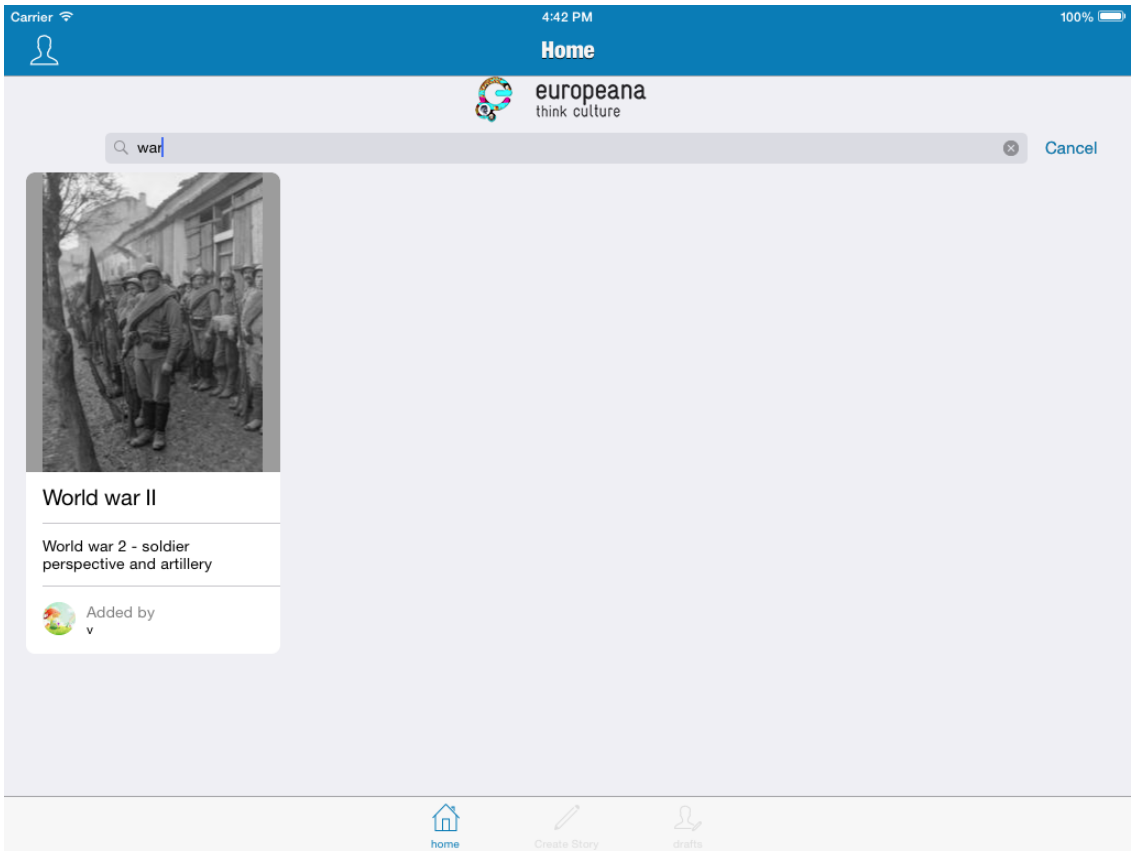
Εικόνα 5.1-1: Αρχική οθόνη

Στο πάνω μέρος βλέπουμε μια μπάρα για αναζήτηση στις δημοσιευμένες ιστορίες. Πιο κάτω είναι οι δημοσιευμένες ιστορίες των χρηστών και είναι μη scrollable επιφάνεια, καθώς το πλήθος των ιστοριών είναι δυναμικό και μπορεί να είναι πολύ μεγάλο. Κάτω υπάρχει ο UITabBarController, ο οποίος είναι εμφανείς σε όλες τις οθόνες της εφαρμογής, ώστε να επιτρέπει πάντα τη μετάβαση σε μία από τις βασικές λειτουργίες της εφαρμογής (home, create story, drafts). Όπως φαίνεται εδώ μόνο το tab με τίτλο home είναι ενεργοποιημένο, καθότι ο χρήστης δεν έχει συνδεθεί από το λογαριασμό του. Πάνω αριστερά βρίσκεται το κουμπί που πρέπει να πατήσει ο επισκέπτης για να συνδεθεί.

Κατά την εμφάνισή του, ο controller αυτός φορτώνει όλες τις δημοσιευμένες ιστορίες των χρηστών από το Parse backend και προβάλλει τα εξώφυλλά τους, που αποτελούνται από την εικόνα, τον τίτλο, τη περιγραφή και το χρήστη που την ανέβασε. Παρουσιάζει τις εικόνες με ένα waterfall layout, που στην ουσία είναι ένα layout όπου το πλήθος των στηλών και το πλάτος τους είναι προκαθορισμένο, αλλά το ύψος του κάθε αντικειμένου είναι δυναμικό. Καθορίζεται από τις διαστάσεις της εκάστοτε εικόνας, αλλά και τον όγκο του κειμένου σε τίτλο και περιγραφή.

5.1.1 *Αναζήτηση σε δημοσιευμένες ιστορίες*

Ο χρήστης μπορεί να αναζητήσει ιστορίες με λέξεις κλειδιά που θα εισάγει στην μπάρα αναζήτησης. Αφού εισαγάγει τη λέξη που θέλει και επιστρέψει από το πληκτρολόγιο της συσκευής εμφανίζονται τα αποτελέσματα της αναζήτησης.

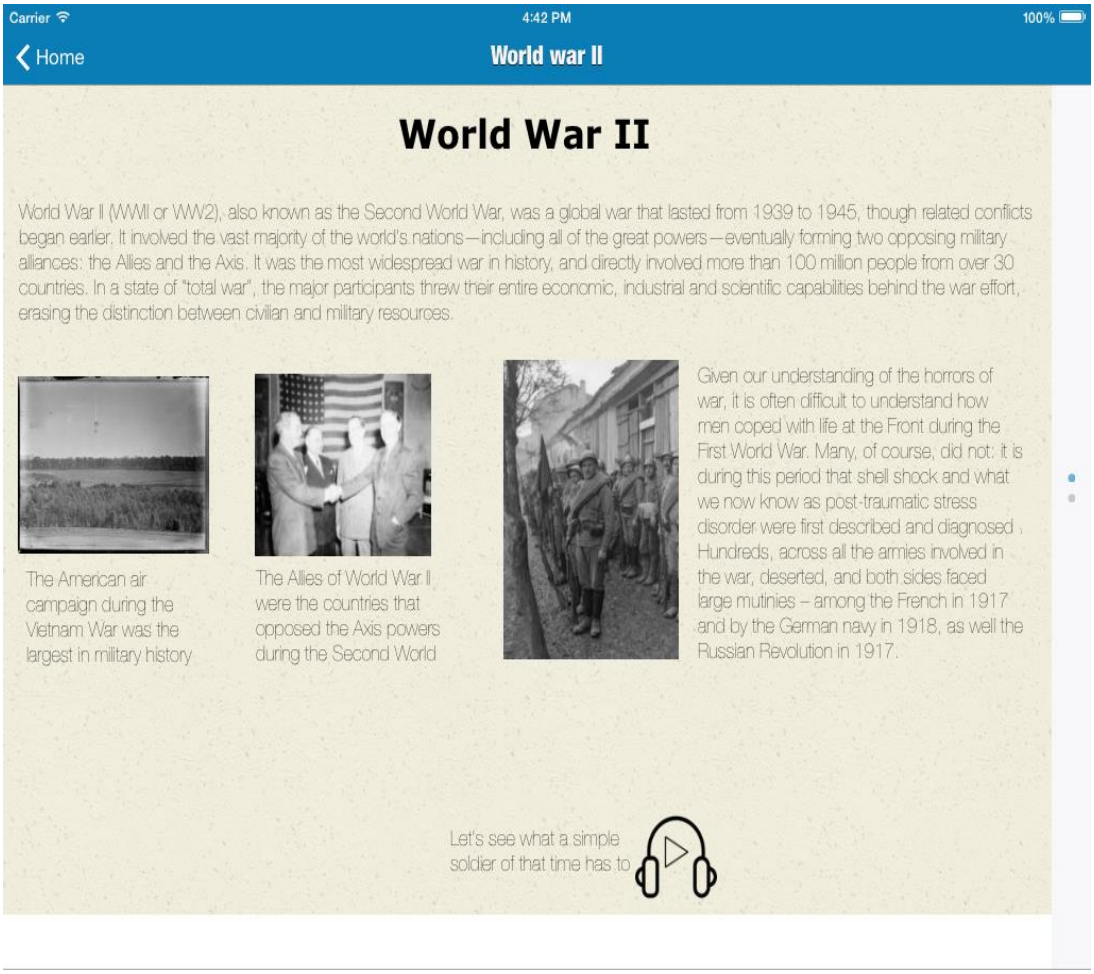


Εικόνα 5.1-2: Αναζήτηση σε δημοσιευμένες ιστορίες

Με το πάτημα του κουμπιού cancel επανέρχεται στην οθόνη με όλες τις ιστορίες.

5.1.2 *Άνοιγμα ιστορίας*

Σε οποιαδήποτε ιστορία (είτε έχοντας κάνει αναζήτηση είτε όχι) πατήσει ο χρήστης οδηγείται στην λεπτομερή παρουσίασή της. Υπεύθυνος για αυτό το κομμάτι είναι ο StoryViewController, ο οποίος φορτώνεται μετά από το πάτημα.



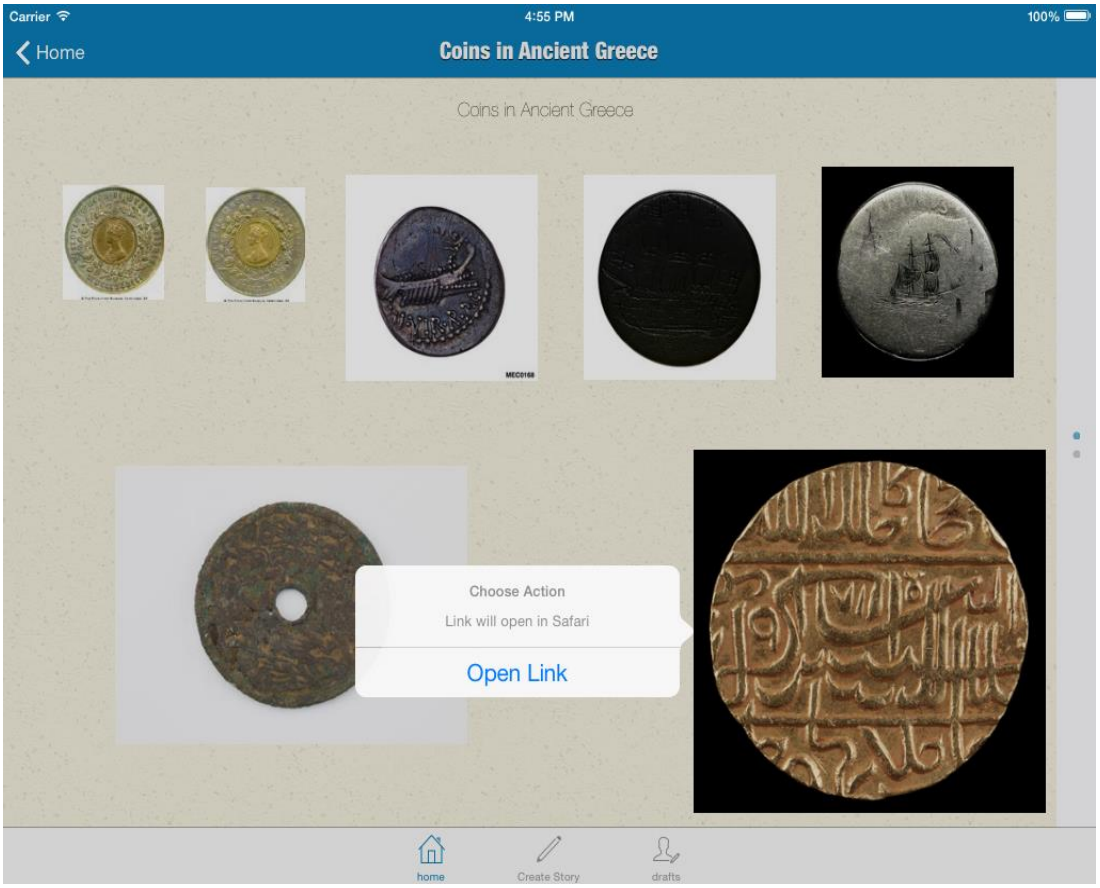
Εικόνα 5.1-3: StoryViewController

Εδώ ο χρήστης μπορεί να περιηγηθεί στην συγκεκριμένη ιστορία.

Δεξιά υπάρχει ένα UIPageControl το οποίο δείχνει από πόσες σελίδες αποτελείται η ιστορία. Για να μεταβεί στην επόμενη σελίδα μπορεί να κάνει scroll προς το κάτω.

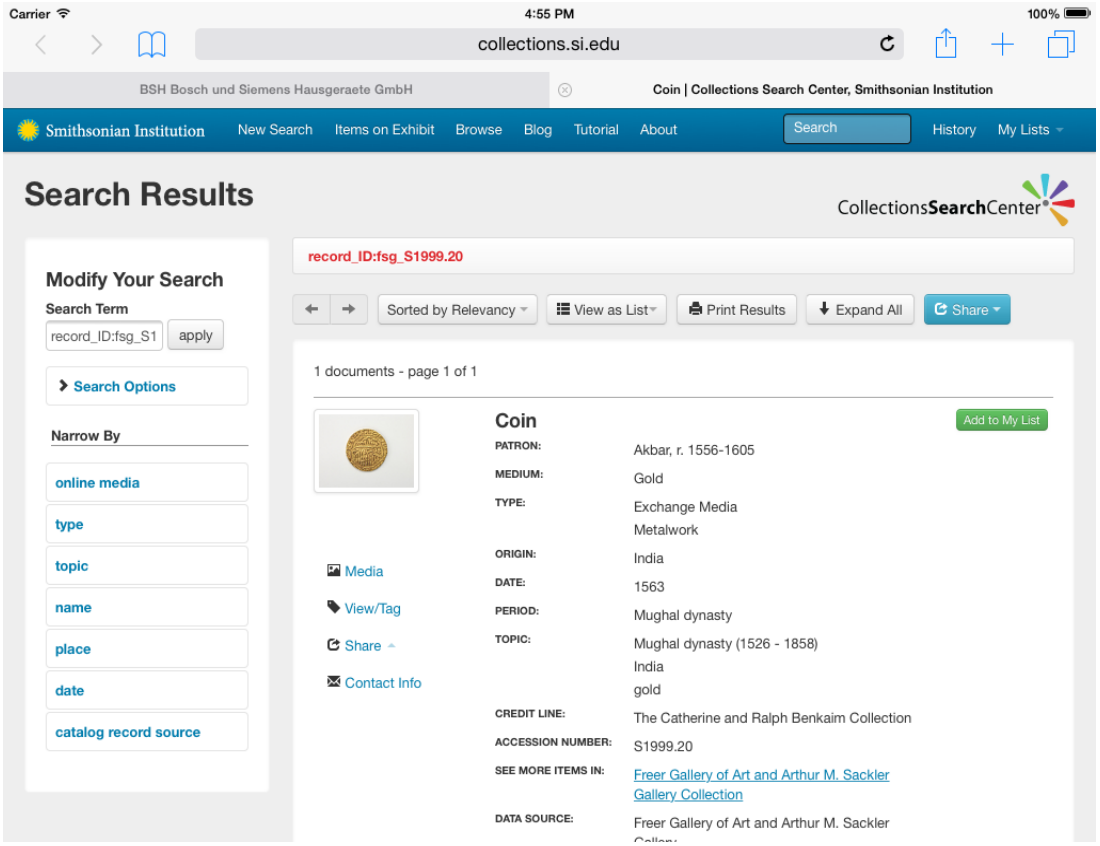
Εικονίδια όπως αυτά με τα ακουστικά υποδηλώνουν ηχογραφημένα αρχεία, τα οποία μπορεί να αναπαράγει πατώντας πάνω στο play.

Επιπλέον, ο χρήστης μπορεί να πατήσει πάνω στις εικόνες της ιστορίας και εάν οι εικόνες αυτές προέρχονται από τις ψηφιακές βιβλιοθήκες (και όχι από τη συσκευή του εκάστοτε χρήστη) θα ανοίξει ένα action panel, που τον ρωτά εάν θέλει να οδηγηθεί στον ιστότοπο που φιλοξενεί το συγκεκριμένο αντικείμενο.



Εικόνα 5.1-4: Action Panel για άνοιγμα αντικειμένου στον ιστότοπο

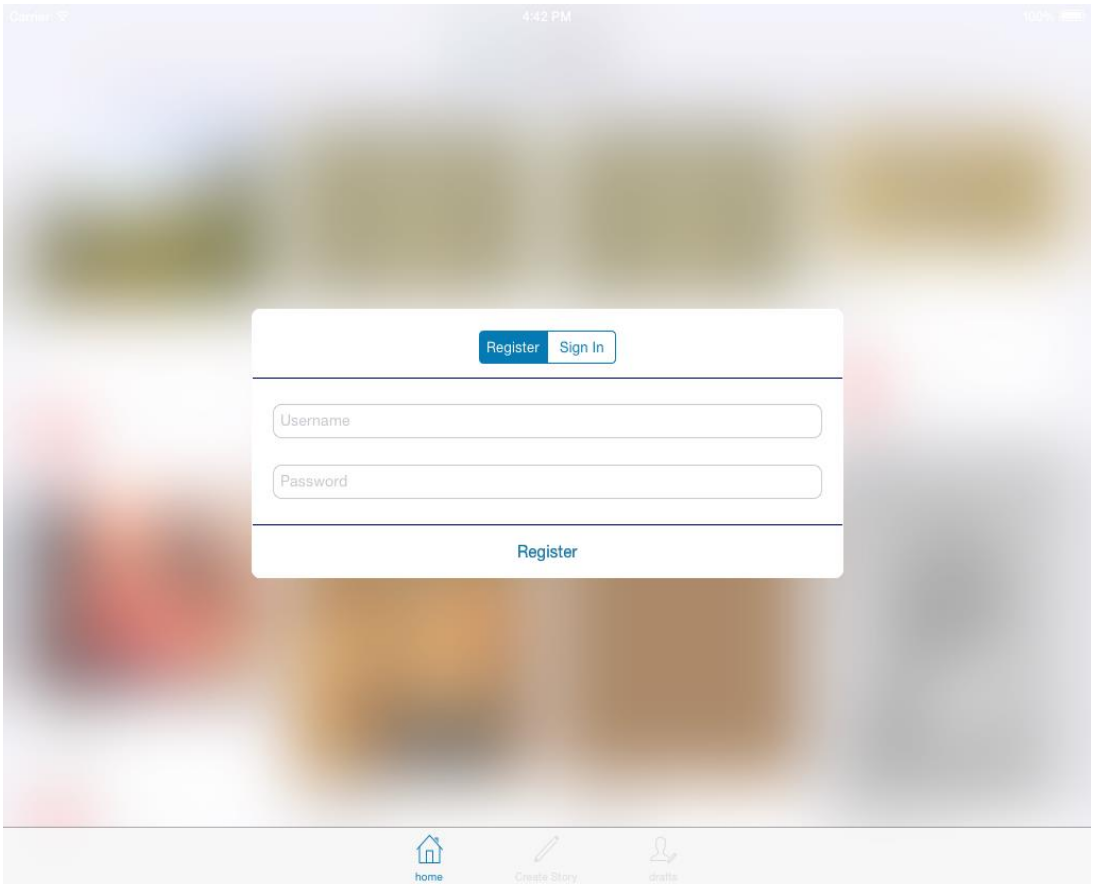
Και εάν πατήσει το κουμπί Open Link, η εφαρμογή θα ανοίξει την εικόνα αυτή στο browser της συσκευής.



Εικόνα 5.1-5:Σελίδα ιστοτόπου που φιλοξενεί το αντικείμενο

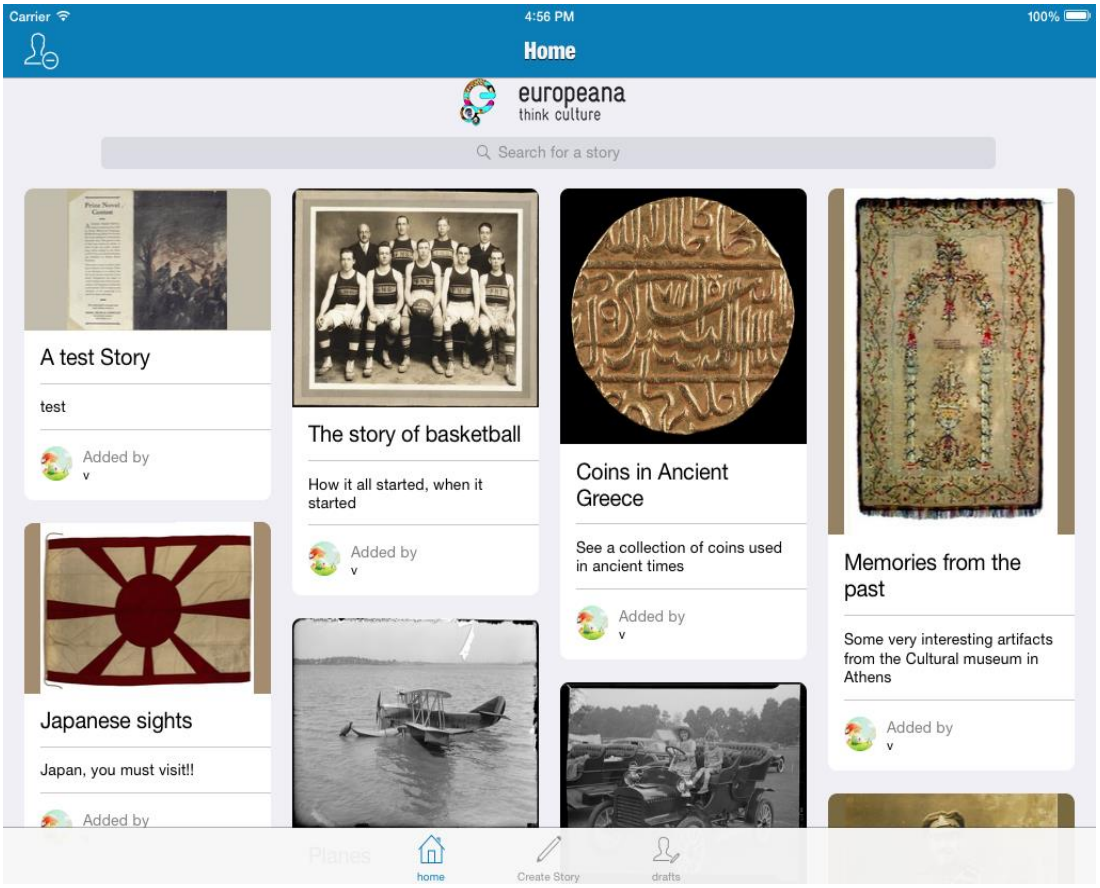
5.1.3 Σύνδεση - Εγγραφή χρήστη

Προκειμένου να ξεκλειδώσει ο χρήστης τα υπόλοιπα δύο tabs πρέπει να συνδεθεί με το λογαριασμό του. Πατάει, λοιπόν το κουμπί της σύνδεσης και εμφανίζεται ο LoginViewController.



Εικόνα 5.1-6: Σύνδεση – εγγραφή χρήστη

Ο Controller αυτός είναι modal και από πίσω του εμφανίζεται «θολωμένος» ο προηγούμενος controller, κάτι που είναι πολύ σύνηθες από το iOS 7 και μετά, προκειμένου να δοθεί έμφαση σε σημαντική πληροφορία. Παρέχει στο χρήστη δυνατότητα τόσο εγγραφής, όσο και σύνδεσης (εφόσον έχει εγγραφεί παλαιότερα). Μόλις η εκάστοτε διαδικασία ολοκληρωθεί επιτυχώς, επιστρέφουμε στην αρχική οθόνη με τις ιστορίες και ενεργοποιούνται και τα άλλα δύο tabs.



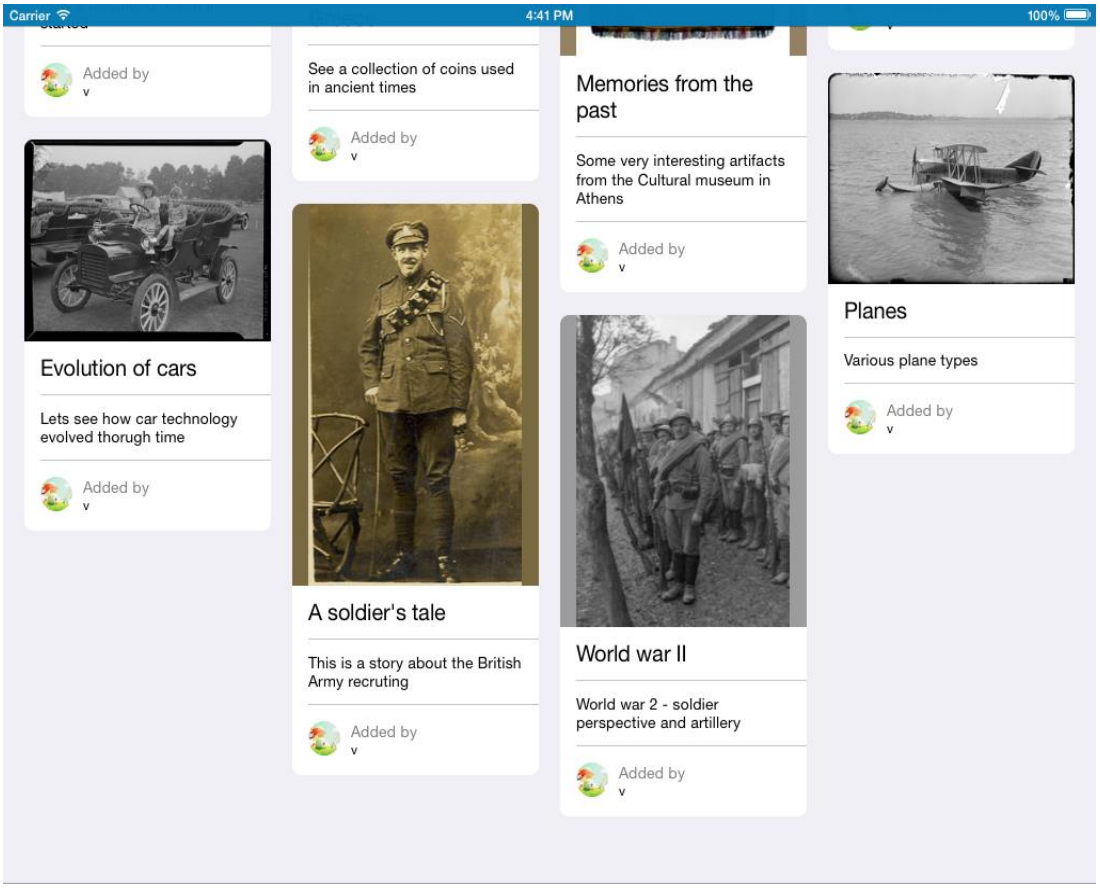
Εικόνα 5.1-7: Αρχική οθόνη – συνδεδεμένος χρήστης

Βλέπουμε ότι τώρα πια είναι ενεργοποιημένα και τα άλλα δύο tabs (Create Story, drafts).

Προτού αναλύσουμε τις υπόλοιπες λειτουργίες, ας δούμε μια επιπλέον λειτουργικότητα του πρώτου tab. Βλέπουμε ότι στη σελίδα αυτή φαίνεται ο navigation controller, η μπάρα αναζήτησης με το logo της Europeana στο πάνω μέρος και ο tabBarController στο κάτω. Όσο και αν κάτι τέτοιο είναι απαραίτητο για λόγους λειτουργικότητας και πλοήγησης, δεν αφήνει πολύ χώρο για την παρουσίαση των ιστοριών. Για αυτό και υλοποιήθηκε το εξής: Όταν ο χρήστης κάνει scroll προς τα κάτω τόσο ο UITabBarController όσο και η μπάρα αναζήτησης μαζί με το logo και τον UINavigationController εξαφανίζονται με κάποιο animation, για να δοθεί χώρος και έμφαση στις ιστορίες. Βέβαια, με λίγο scroll προς την αντίθετη κατεύθυνση ξαναεμφανίζονται τα στοιχεία αυτά, γιατί σε καμία περίπτωση ο χρήστης δεν πρέπει να στερείται λειτουργικότητας. Αυτή η τακτική είναι μια τεχνική που χρησιμοποιείται σε κινητές συσκευές για την εκμετάλλευση του χώρου και είναι γνώριμη στους χρήστες του iOS.

Παρατίθεται παρακάτω μια φωτογραφία όπου φαίνεται ακριβώς αυτό. Ο χρήστης έχει κάνει scroll, τα προαναφερθείσα στοιχεία εξαφανίζονται και δίνεται έμφαση στις ιστορίες. Το μόνο

που παραμένει, είναι το status bar το οποίο, συνήθως, είναι κακή πρακτική να μην είναι εμφανές, γιατί παρέχει πολύτιμη πληροφορία για το χρήστη.

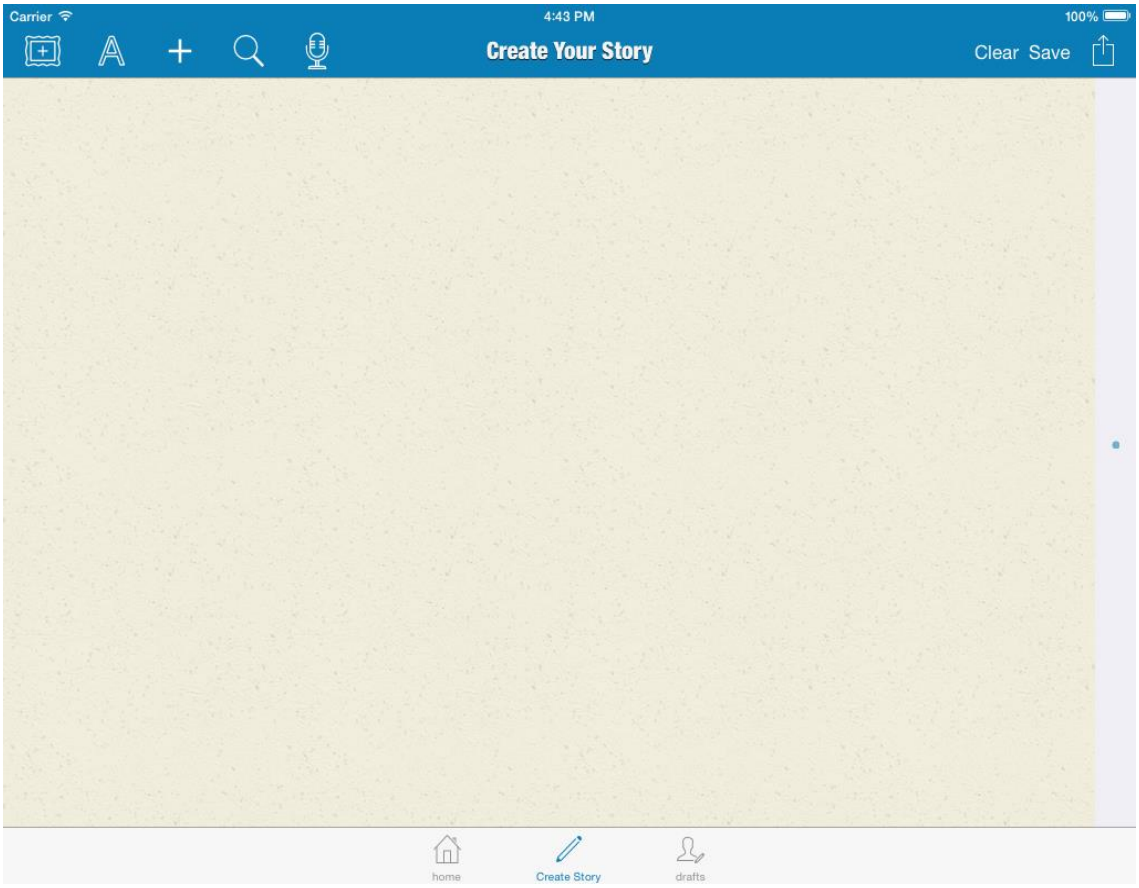


Εικόνα 5.1-8:Απόκρυψη status και navigation bars

5.2 Σύνταξη Ιστορίας

5.2.1 Συνοπτική περιγραφή

Αποτελεί το δεύτερο tab της εφαρμογής και περιλαμβάνει ότι έχει να κάνει με τη δημιουργία της ιστορίας του χρήστη. Ο πρώτος controller που φορτώνεται με το που πατά ο χρήστης το tab αυτό, είναι ο CreateStoryViewController.



Εικόνα 5.2-1: CreateStoryViewController

Πάνω στο navigation bar υπάρχουν όλες οι δυνατές λειτουργικότητες του editor.

Στην αριστερή πλευρά της μπάρας με σειρά από αριστερά προς τα δεξιά ο χρήστης μπορεί να:

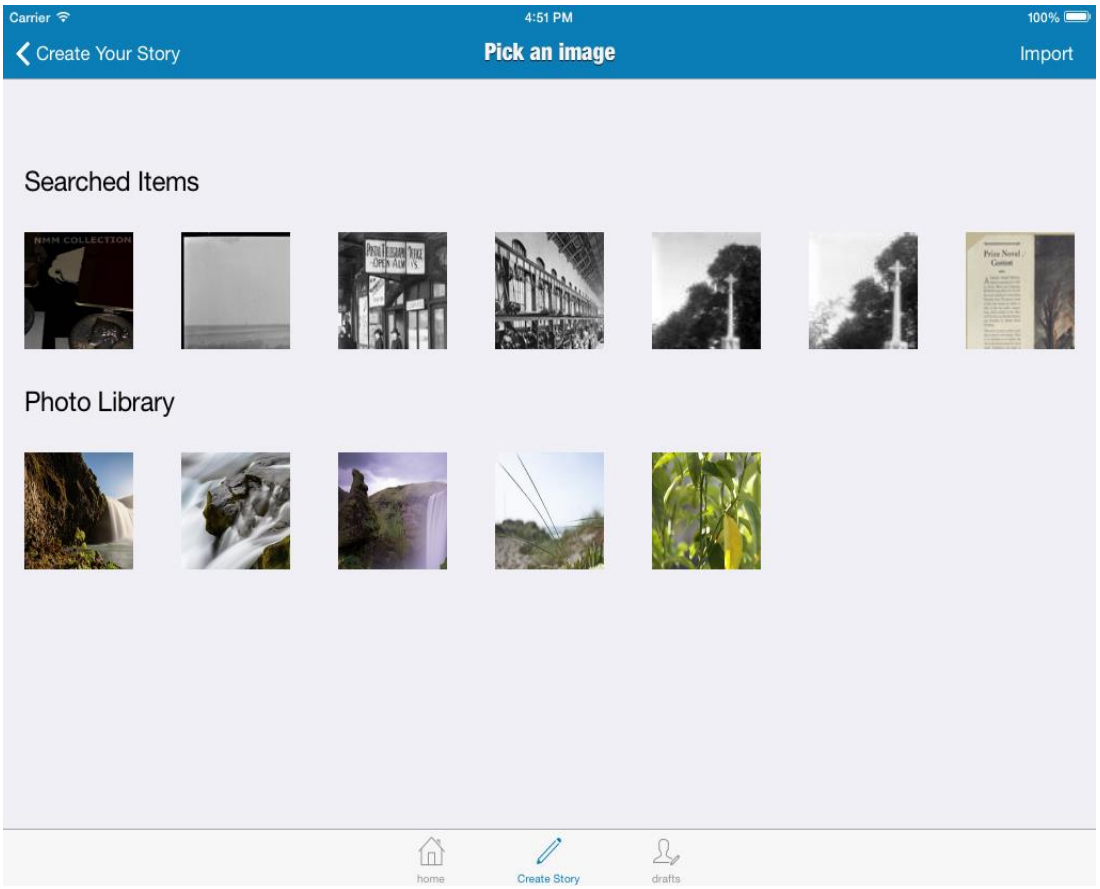
- εισαγάγει φωτογραφία
- εισαγάγει κείμενο
- προσθέσει σελίδα στην ιστορία
- να αναζητήσει υλικό στις ψηφιακές βιβλιοθήκες
- να ηχογραφήσει ένα μήνυμα

Και με τα κουμπιά στη δεξιά πλευρά μπορεί να:

- καθαρίσει τον editor από οτιδήποτε έχει εισάγει μέχρι στιγμής
- σώσει την ιστορία του τοπικά στα drafts
- να την ανεβάσει στο server προκειμένου να δημοσιευτεί στη Home οθόνη όλων των χρηστών.

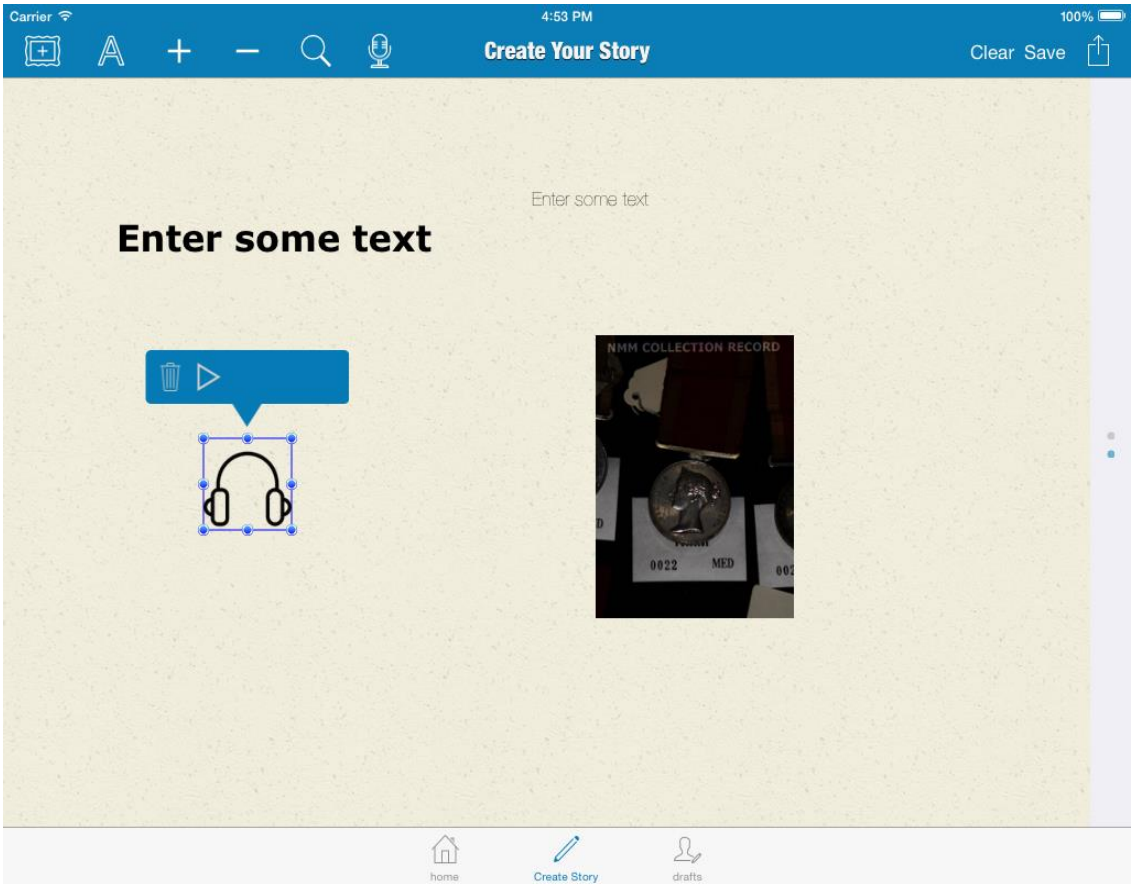
5.2.2 Εισαγωγή φωτογραφίας

Μόλις ο χρήστης πατήσει το σχετικό κουμπί, φορτώνεται ο SavedPhotosGallerViewController για να δώσει στο χρήστη τη δυνατότητα να επιλέξει τη φωτογραφία που θέλει να εισαγάγει στην ιστορία.



Εικόνα 5.2-2: Επιλογή φωτογραφίας

Παρατηρούμε ότι υπάρχουν δύο τμήματα. Το πρώτο (searched items) περιέχει φωτογραφίες από αντικείμενα αναζήτησης στις βιβλιοθήκες που επέλεξε να σώσει ο χρήστης και το δεύτερο φωτογραφίες από τη συσκευή του χρήστη. Εκεί, ο χρήστης μπορεί να επιλέξει μια φωτογραφία και να την εισάγει πατώντας το κουμπί import στην ιστορία του. Μόλις το πατήσει, επιστρέφει στην οθόνη του editor και η φωτογραφία εμφανίζεται επιλεγμένη και έτοιμη για τον χρήστη να τη μετακινήσει ή να της αλλάξει το μέγεθος, όπως αυτός επιθυμεί.



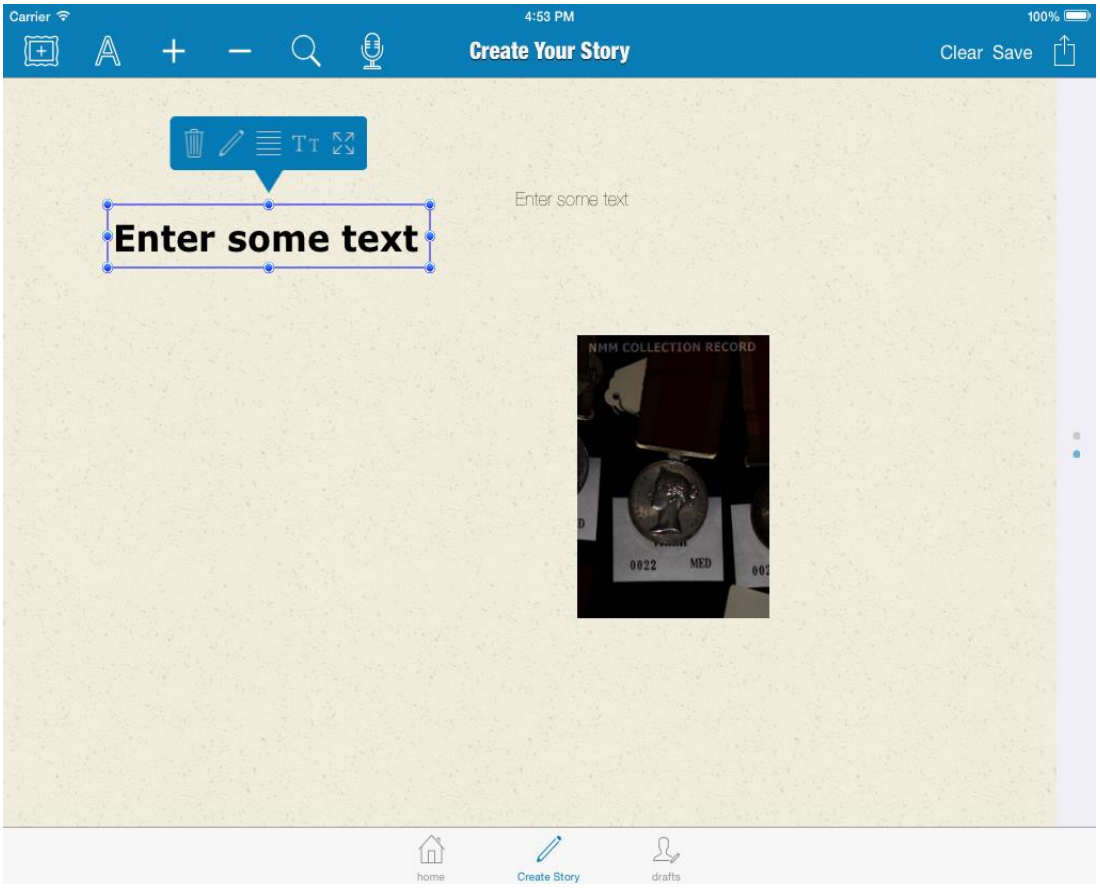
Εικόνα 5.2-3: Εισαγωγή φωτογραφίας

Η επιλεγμένη φωτογραφία εμφανίζεται με μπλε πλαίσιο.

Προκειμένου να της αλλάξει το μέγεθος δεν έχει παρά να πατήσει πάνω στο πλαίσιο και να επεκτείνει την εικόνα προς οποιαδήποτε κατεύθυνση. Για να την μετακινήσει πρέπει να πατήσει στο κέντρο της εικόνας να μετακινήσει την εικόνα όπου αυτός επιθυμεί και να απομακρύνει το δάχτυλό του από την οθόνη.

5.2.3 Εισαγωγή κειμένου

Αν ο χρήστης πατήσει το κουμπί εισαγωγής κειμένου εισάγεται στην οθόνη του editor ένα placeholder κείμενο, το οποίο μπορεί ο χρήστης να μορφοποιήσει, να μετακινήσει και να του δώσει το μέγεθος που προτιμά.



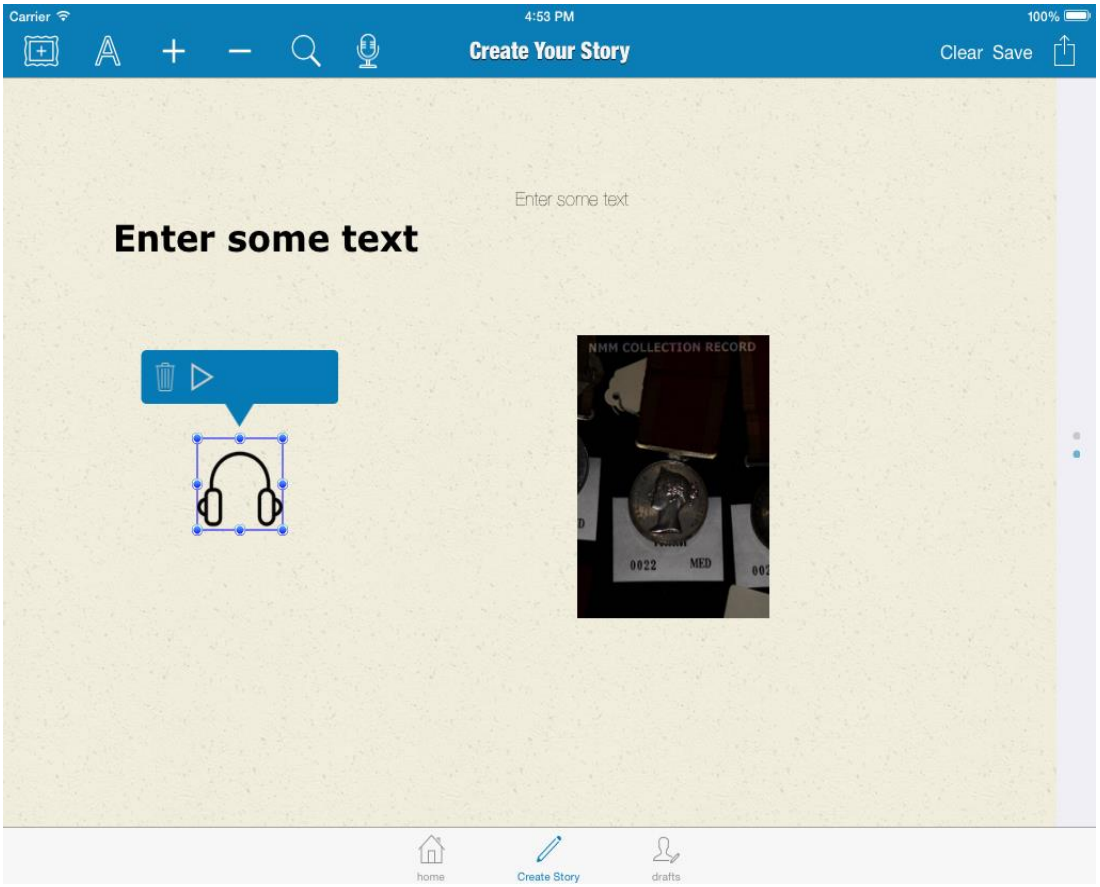
Εικόνα 5.2-4: Εισαγωγή – Επιλογές σύνταξης κειμένου

Η μετακίνηση και η αλλαγή μεγέθους λειτουργεί ακριβώς με τον ίδιο τρόπο όπως στις εικόνες, σημειώνοντας ότι το κείμενο "ακολουθεί" τις αλλαγές αυτές του μεγέθους. Για τις δυνατότητες μορφοποίησης, ο χρήστης μπορεί να χρησιμοποιήσει το popover που εμφανίζεται. Με σειρά από αριστερά προς τα δεξιά επεξηγούμε τη λειτουργία των κουμπιών:

- διαγραφή κειμένου
- επεξεργασία του κειμένου. Εμφανίζει το πληκτρολόγιο της συσκευής.
- επιλογή στοίχισης κειμένου(αριστερά, κέντρο, δεξιά). Επαναφορτώνει στο popover τις τρεις αυτές επιλογές με τα αντίστοιχα εικονίδια της στοίχισης.
- επιλογή τύπου κειμένου. Δίνονται δύο επιλογές, απλό κείμενο και κείμενο επικεφαλίδας.
- συρρίκνωση του μπλε πλαισίου ώστε να χωρέσει το κείμενο ακριβώς με βάση με το περιεχόμενο του. Αυτό είναι χρήσιμο σε περίπτωση που ο χρήστης αλλάζει το μέγεθος του πλαισίου και θέλει να το επαναφέρει ώστε να χωράει ακριβώς το κείμενο.

5.2.4 *Ηχογράφηση μηνύματος*

Με πάτημα στο κουμπί του μικροφώνου, ενεργοποιείται το μικρόφωνο της συσκευής και το εικονίδιο του κουμπιού αντικαθίσταται από ένα εικονίδιο stop.

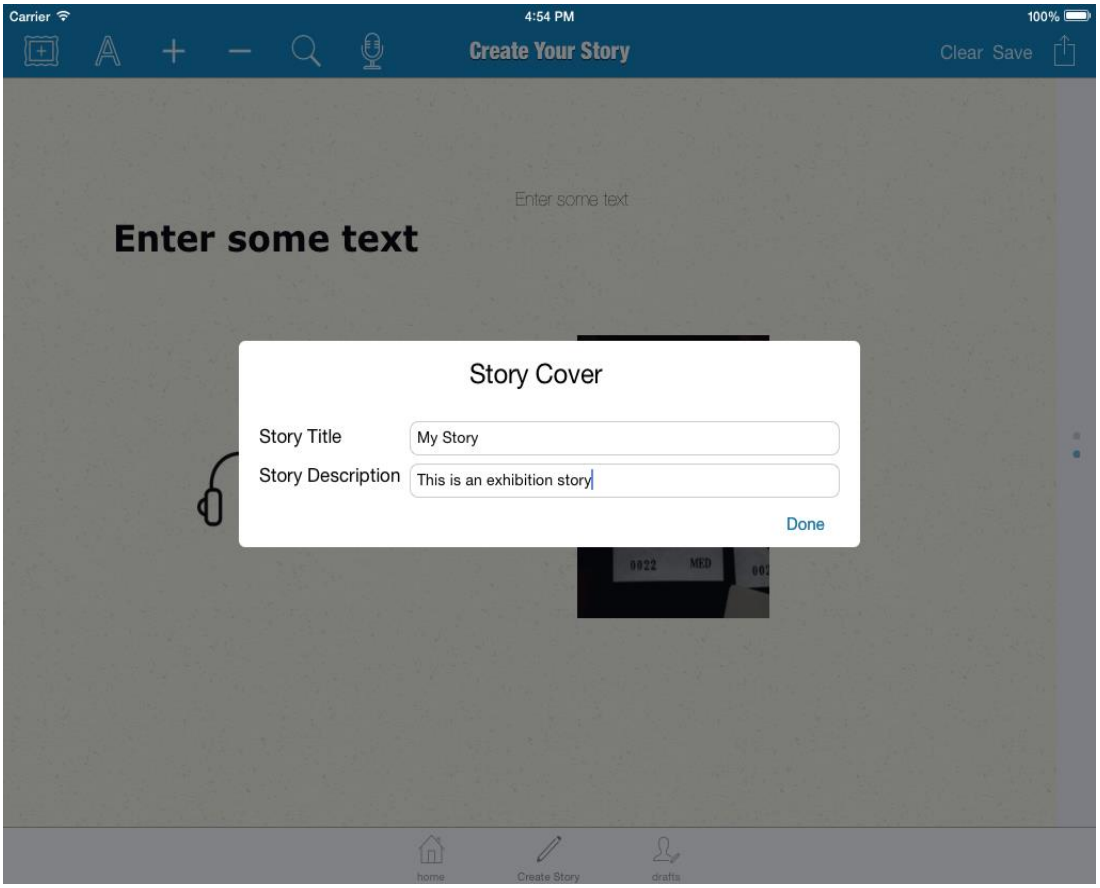


Εικόνα 5.2-5: Εισαγωγή ηχογράφησης

Μόλις το πατήσει ο χρήστης σταματά η ηχογράφηση και εμφανίζεται το ηχογραφημένο μήνυμα με το εικονίδιο των ακουστικών. Και αυτό μπορεί να το μετακινήσει όπου θέλει, αλλά όχι να αλλάξει το μέγεθος του εικονιδίου. Επιπλέον στο popover εμφανίζεται το κουμπί play, το οποίο μπορεί να πατήσει για να ακούσει το μήνυμα του.

5.2.5 *Επιλογή εξωφύλλου ιστορίας*

Όταν ο χρήστης τελειώσει την ιστορία του έχει δύο επιλογές. Να τη σώσει τοπικά, είτε να την ανεβάσει στο server. Και στις δύο περιπτώσεις, είτε δηλαδή ο χρήστης πατήσει το κουμπί save, είτε αυτό που συμβολίζει το ανέβασμα, παραπέμπεται στην επιλογή εξωφύλλου. Για την επιλογή φορτώνεται ο CoverPickerController.



Εικόνα 5.2-6: Επιλογή εξωφύλλου

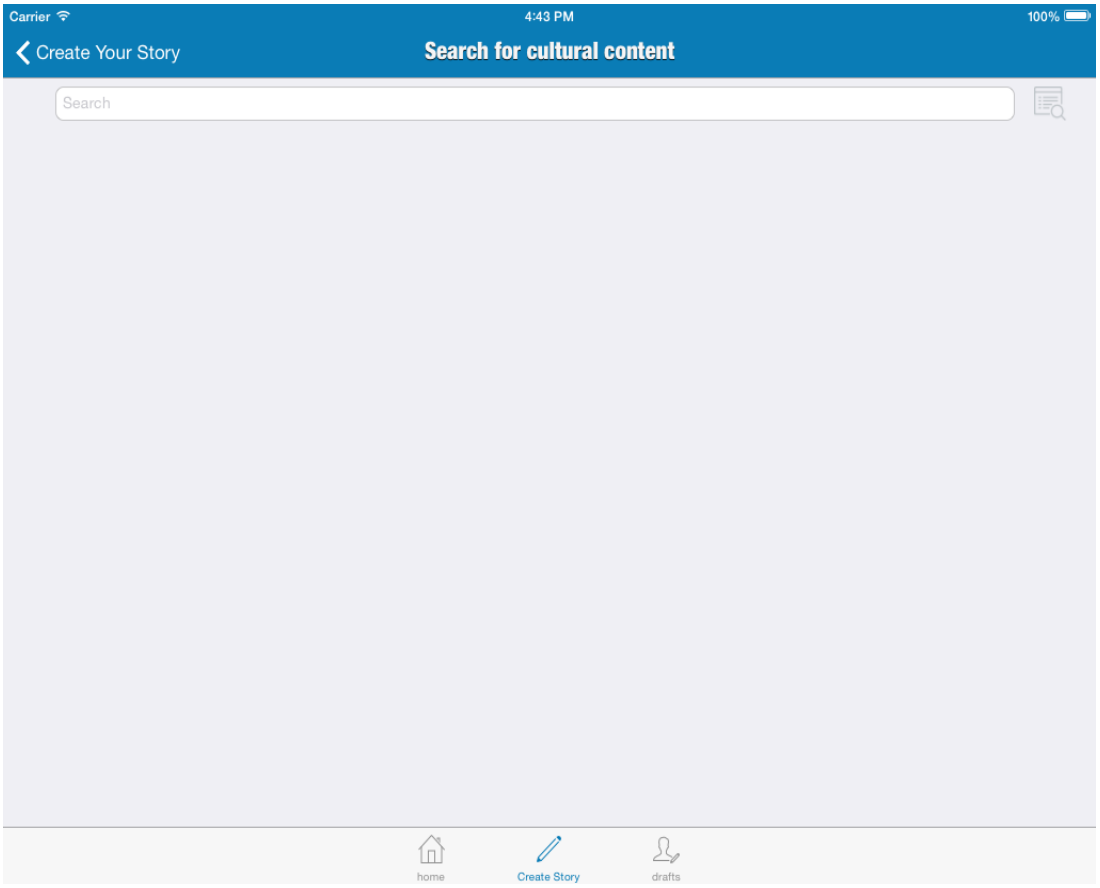
Εδώ ο χρήστης εισάγει τον τίτλο και την περιγραφή της ιστορίας του και πατά Done. Στη συνέχεια, εμφανίζεται πάλι ο controller για την επιλογή φωτογραφίας. Η διαδικασία είναι ακριβώς η ίδια με αυτή για την εισαγωγή φωτογραφίας, οπότε δεν κρίνεται απαραίτητο να παραθετούν οι σχετικές φωτογραφίες. Αφού ο χρήστης επιλέξει και φωτογραφία εξωφύλλου, η ιστορία σώζεται τοπικά ή ανεβάζεται, αναλογα με την αρχική δραστηριότητα του.

5.3 Αναζήτηση σε ψηφιακές βιβλιοθήκες

Εδώ θα εξετάσουμε πως γίνεται η αναζήτηση στις ψηφιακές βιβλιοθήκες

5.3.1 CollectionGalleryViewController

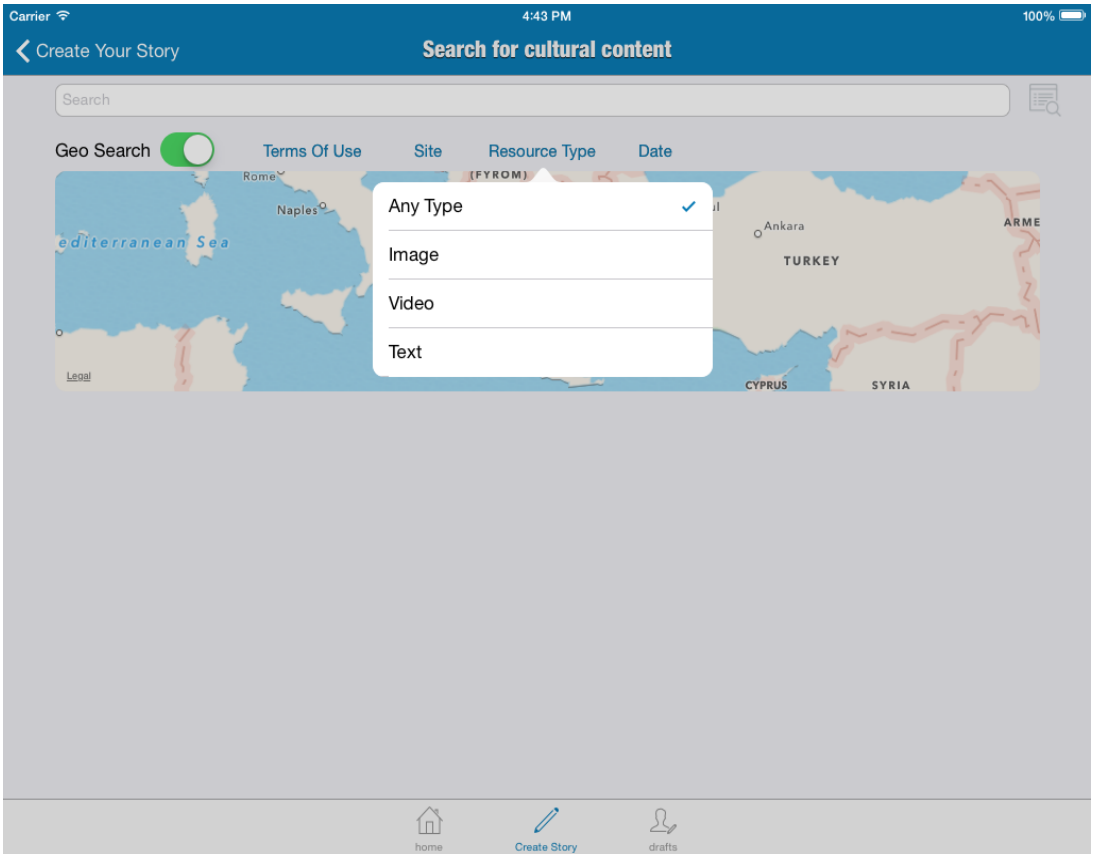
Αν ο χρήστης πατήσει το εικονίδιο με το μεγενθυντικό φακό στο tab Create Story φορτώνεται ο controller που υλοποιεί την αναζήτηση στις ψηφιακές βιβλιοθήκες.



Εικόνα 5.3-1: Αναζήτηση σε ψηφιακές βιβλιοθήκες

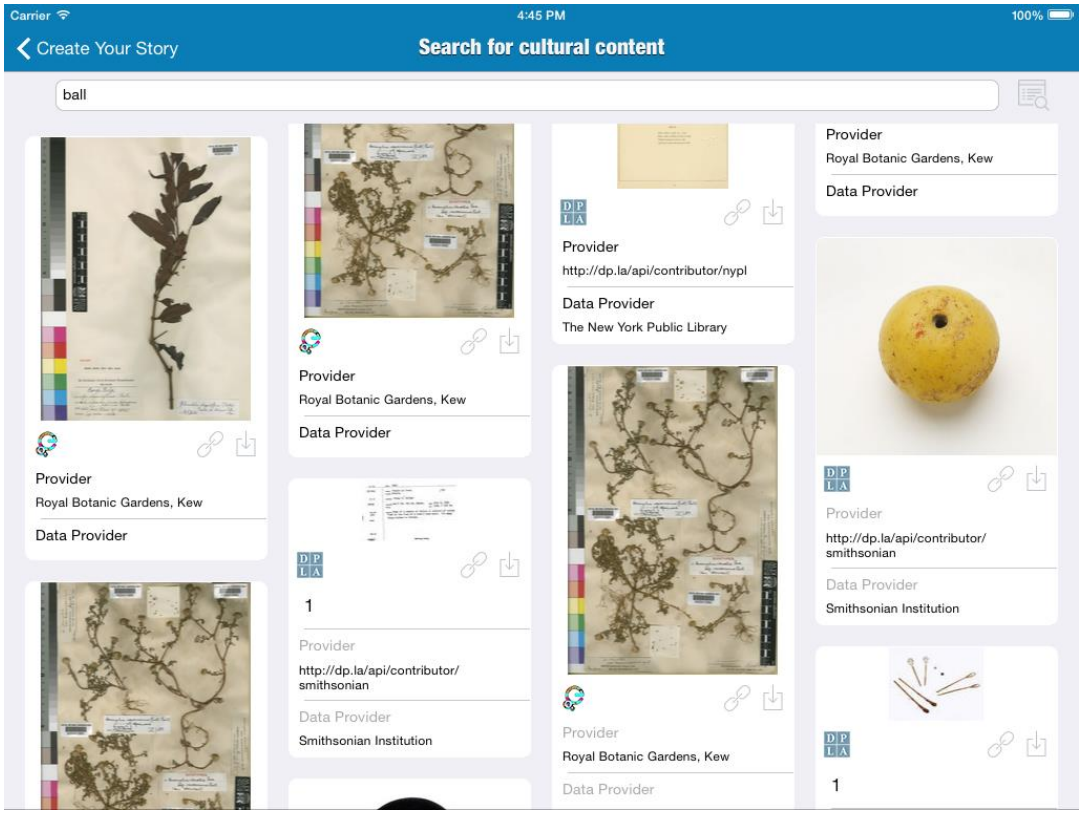
Διαθέτει μια μπάρα αναζήτησης, καθώς και ορισμένα φίλτρα, τα οποία ο χρήστης μπορεί να φέρει στο προσκήνιο πατώντας στο κουμπί που βρίσκεται δεξιά από τη μπάρα.

Για να θέσει κάποιο από τα φίλτρα αυτά δεν έχει παρά να πατήσει σε όποιο κουμπί επιθυμεί. Με το πάτημα εμφανίζεται ένα popover με τις δυνατές επιλογές του εκάστοτε φίλτρου. Ο χρήστης πατά σε όποια επιλογή του φίλτρου επιθυμει και το popover κλείνει και αποθηκεύει την επιλογή του. Μπορεί να βάλει όσα φίλτρα επιθυμεί και όλα θα συμπεριληφθούν στην αναζήτηση. Τα δυνατά φίλτρα αφορούν τα δικαιώματα επαναχρησιμοποίησης των αντικειμένων, το είδος του μέσου, το/τα site/s που θέλουμε να ψάξουμε, καθώς και χρονολογικό φίλτρο. Πέρα από τα φίλτρα αυτά, δίνεται και η δυνατότητα για γεωγραφική αναζήτηση, ενεργοποιώντας την επιλογή Geo Search και βάζοντας στο χάρτη την περιοχή που ο χρήστης επιθυμεί. Ακολουθεί η χαρακτηριστική εικόνα.



Εικόνα 5.3-2: Φίλτρα αναζήτησης

Εφόσον ο χρήστης εκτελέσει την αναζήτηση εμφανίζονται τα αποτελέσματα:

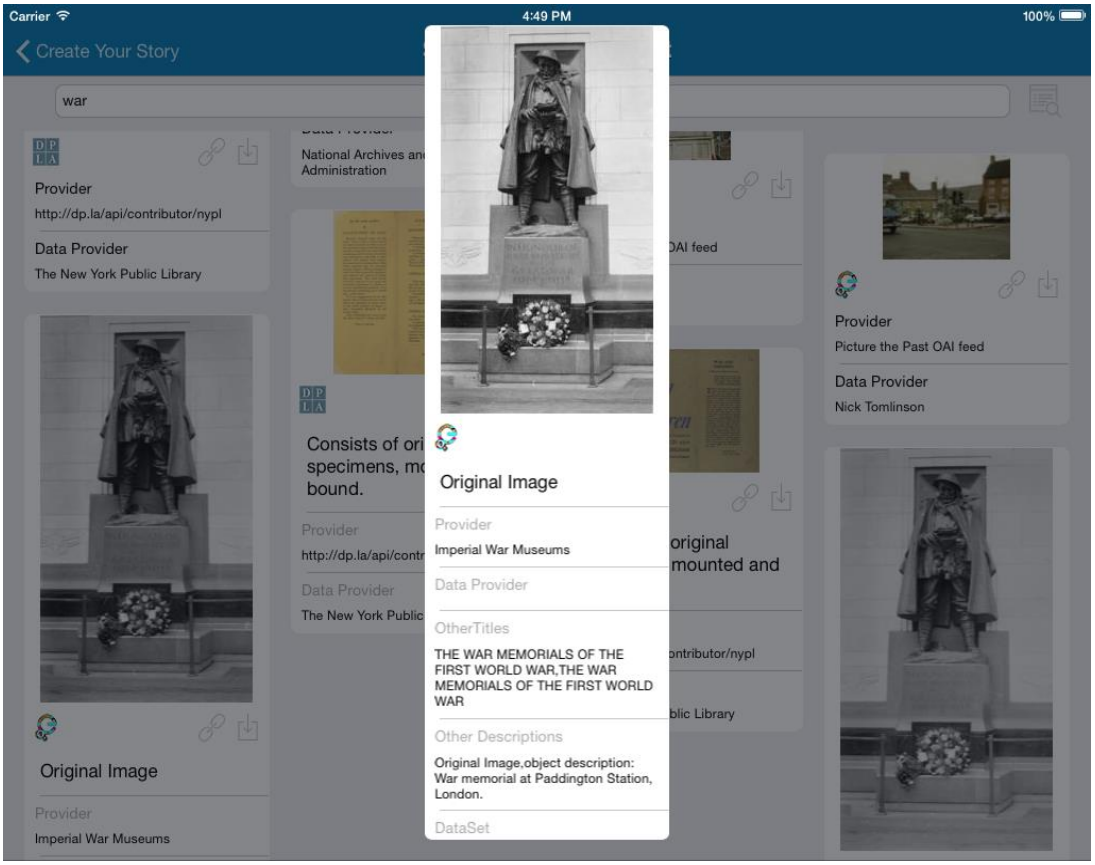


Εικόνα 5.3-3: Αποτελέσματα αναζήτησης

Για την παρουσίαση των αποτελεσμάτων χρησιμοποιήθηκε και πάλι το waterfall layout λόγω της δυναμικότητας των αποτελεσμάτων εικόνας (διαφορετικά μεγέθη εικόνας και διαφορετικός όγκος κειμένου μεταδεδομένων). Πάνω σε κάθε αντικείμενο παρατηρούμε δύο κουμπιά με γκρίζο χρώμα. Το ένα που μοιάζει με link οδηγεί στη σελίδα του ιστοτόπου που φιλοξενεί το συγκεκριμένο αντικείμενο, ενώ το δεύτερο αποθηκεύει την εικόνα τοπικά στο φάκελο του χρήστη, ώστε να τη χρησιμοποιήσει αργότερα είτε ως υλικό για την ιστορία του, είτε ως εξώφυλλο. Είναι οι φωτογραφίες που εμφανίζονται στον controller που επιλέγουμε φωτογραφίες που είδαμε νωρίτερα.

5.3.2 Λεπτομερής παρουσίαση αντικειμένου αναζήτησης

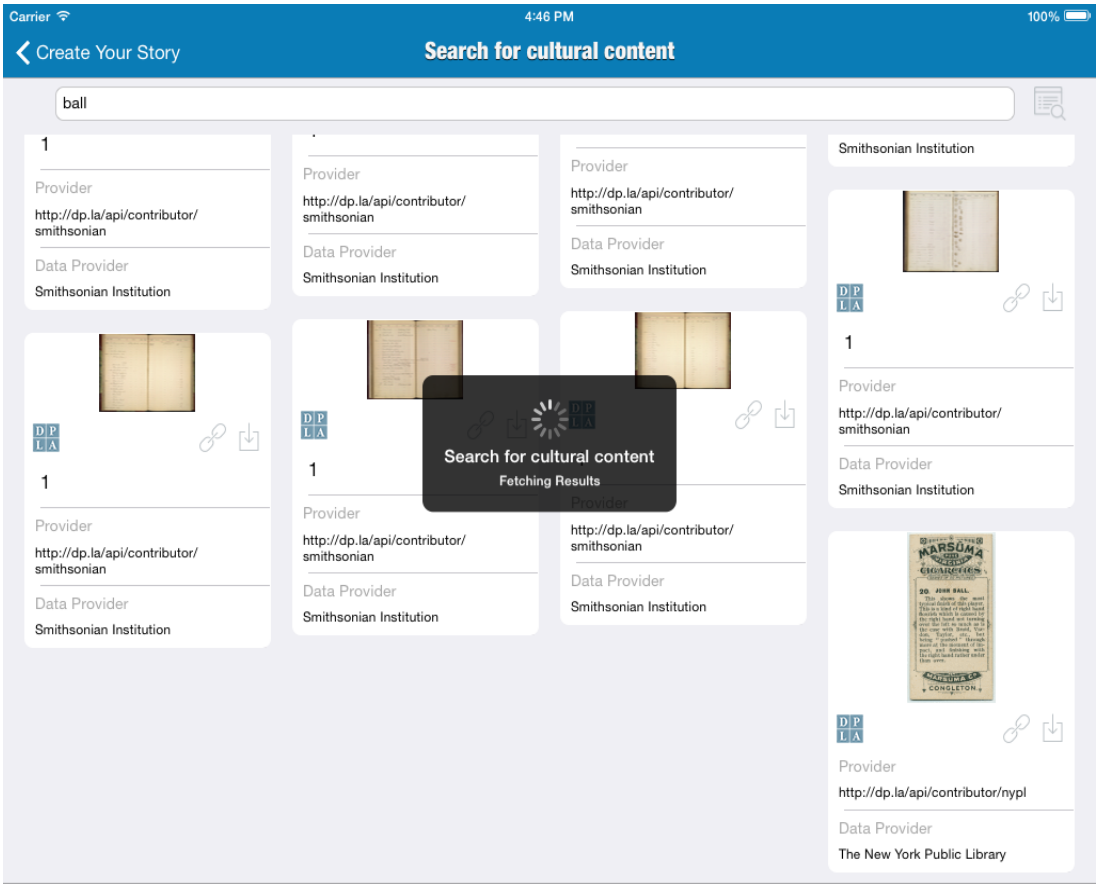
Εφόσον πατήσουμε σε κάποιο αντικείμενο, τότε εμφανίζεται ένας modal controller (ImageDetailViewController) με περισσότερα μεταδεδομένα για το αντικείμενο αυτό. Κάτω από την εικόνα υπάρχει ένας πίνακας τον οποίο μπορούμε να κάνουμε scroll σε περίπτωση που δε φτάνει ο χώρος για την εμφάνιση των μεταδεδομένων.



Εικόνα 5.3-4: Ανάπτυξη περαιτέρω μεταδεδομένων

Και πάλι εφαρμόζεται η τακτική που είδαμε προτύτερα για την εκμετάλλευση του χώρου της συσκευής. Δηλάδη σε περίπτωση που ο χρήστης κάνει scroll προς τα κάτω εξαφανίζουμε τον UITabBarController και το navigationBar για να δώσουμε έμφαση στα αντικείμενα που παίρνουμε από την αναζήτηση.

Επιπλέον, η αναζήτηση υποστηρίζει το paging, δηλαδή κάθε φορά ζητάμε από τα APIs σταθερή ποσότητα αντικειμένων και όταν ο χρήστης κάνει scroll και πλησιάζει προς το τέλος των αποτελεσμάτων ζητάμε την επόμενη «σελίδα» αποτελεσμάτων και τα φορτώνουμε δείχνοντας κατάλληλο μήνυμα.



Εικόνα 5.3-5: Φόρτωση επόμενης σελίδας αποτελεσμάτων

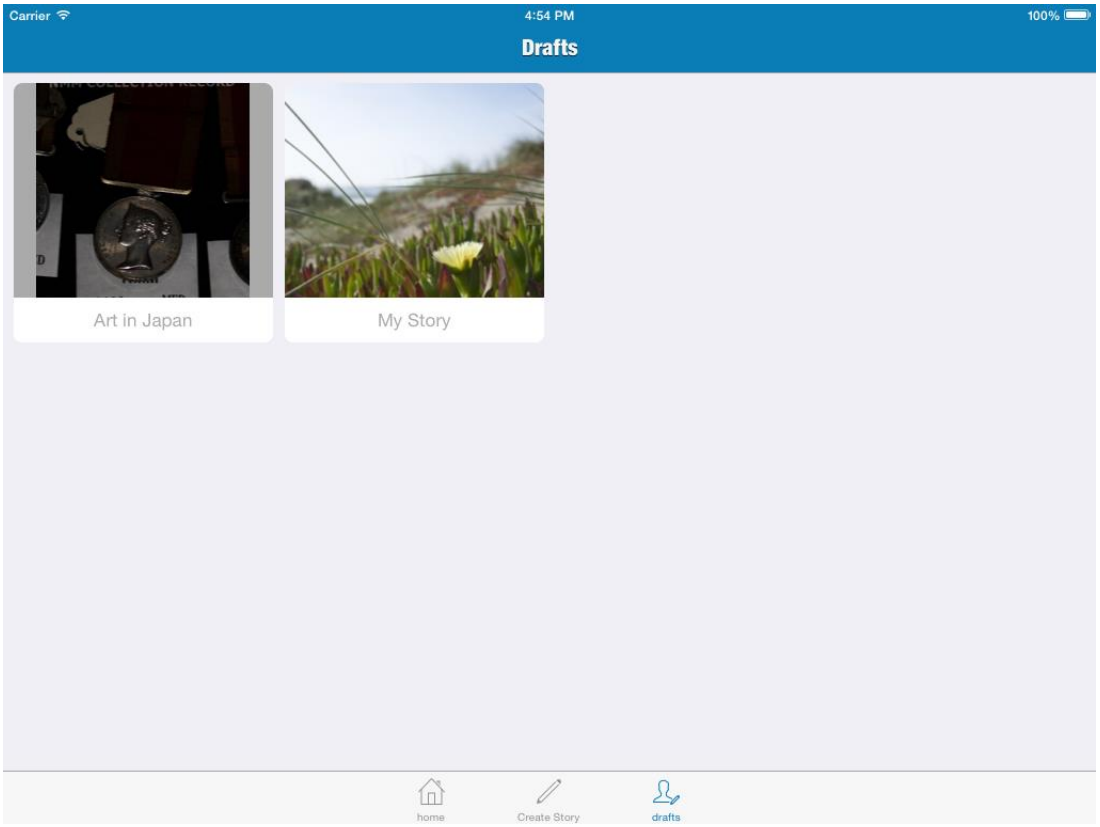
5.4 Πρόχειρα

5.4.1 Συνοπτική περιγραφή

Αποτελεί το τρίτο και τελευταίο tab της εφαρμογής. Εδώ ο χρήστης μπορεί να δεί όλες τις ιστορίες που αποφάσισε να σώσει τοπικά και να τις επεξεργαστεί ξανά.

5.4.2 MyStoriesViewController

Αυτός είναι στην ουσία ο controller που φορτώνεται όταν ο χρήστης επιλέξει το τρίτο tab. Παρουσιάζει τις ιστορίες που έχει σώσει ο χρήστης σε ένα πλέγμα, το λεγόμενο Grid layout με ίδιο μέγεθος για κάθε ιστορία, δείχνοντας την εικόνα και τον τίτλο της ιστορίας.



Εικόνα 5.4-1: Σελίδα με πρόχειρες ιστορίες

Μπορούμε να πατήσουμε πάνω σε όποια ιστορία επιθυμούμε και τοτε θα φορτωθεί ο CreateStoryController αρχικοποιημένος με την ιστορία που διαλέξαμε ώστε να την επεξεργαστούμε και να κάνουμε ό,τι αλλαγές θέλουμε. Αν κατά την επεξεργασία ο χρήστης ανεβάσει επιτυχώς την ιστορία, αυτή σβήνεται από τα drafts.

6

Λεπτομέρειες υλοποίησης

Θα ήταν δύσκολο και θα απαιτούσε πολύ μεγάλο όγκο κειμένου να περιγράψουμε την επακριβή λειτουργία της εφαρμογής και της κάθε κλάσης. Για αυτό κρίνεται προτιμότερο, στο κεφάλαιο αυτό, να περιγράψουμε κάποια μη τετριμμένα ζητήματα, σημεία που προβλημάτησαν, καθώς και να αναλύσουμε τεχνολογίες, εξωτερικά frameworks και libraries που χρησιμοποιήθηκαν.

6.1 Εργαλεία

Τα εργαλεία που χρησιμοποιήσαμε για την ανάπτυξη της εφαρμογής ήταν τα εξής:

- Τη γλώσσα προγραμματισμού Objective C.
- Το XCode, το οποίο είναι το ολοκληρωμένο περιβάλλον ανάπτυξης εφαρμογών για iOS και είναι διαθέσιμο δωρεάν για το OS X από το Mac App Store.Η έκδοση που χρησιμοποιήθηκε ήταν η 6.1.1.
- Τη Restkit library, μια βιβλιοθήκη για iOS που διευκολύνει την επικοινωνία με RestFul Apis.
- Τη βιβλιοθήκη MBProgressHud για τις όψεις προόδου που εμφανίζονται κατά τη διάρκεια του φορτώματος ή του ανεβάσματος δεδομένων από το server.
- Το Parse framework που χρησιμοποιήθηκε για το backend της εφαρμογής.
- Τη βιβλιοθήκη SDWebImage για το ασύγχρονο κατέβασμα φωτογραφιών και το κασάρισμα τους
- Τη βιβλιοθήκη CHTWaterfall για την υλοποίηση του Waterfall layout.
- Τα CocoaPods, που είναι ένας manager που ελέγχει τις εξαρτήσεις μιας εφαρμογής και ανάμεσα σε άλλα επιτρέπει την εύκολη σύνδεση βιβλιοθηκών.

- Το DHC Chrome App το οποίο είναι μια πολύ χρήσιμη εφαρμογή του Chrome που μας επιτρέπει να κάνουμε HTTP/ HTTPS κλήσεις με ευελιξία στη ρύθμιση των παραμέτρων και μας δίνει έτσι τη δυνατότητα να τεστάρουμε οποιαδήποτε κλήση κάνουμε και να βλέπουμε τα αποτελέσματα που γυρνά

6.2 Restkit

Για την εύκολη δημουργία των HTTP μυνημάτων προκειμένου να γίνουν οι κλήσεις στα APIs των ψηφιακών βιβλιοθηκών χρησιμοποιήθηκε η βιβλιοθήκη RestKit [14].

Αποτελεί μια μοντέρνα βιβλιοθήκη που μπορεί να χρησιμοποιηθεί τόσο για το iOS όσο και για το OS X και προσφέρει ευκολία στη δημιουργία κλήσεων προς RESTful Apis (εξ ου και το όνομα) και βασίζεται στην ευρέως διαδεδομένη βιβλιοθήκη AFNetworking [15]. Πέραν της ευκολίας ως προς την δημιουργία κλήσεων και των παραμέτρων τους, παρέχει και ένα πολύ ισχυρό μηχανισμό με τον οποίο μπορούμε να αυτοματοποιήσουμε την αντιστοίχιση μεταξύ των κλάσεων της εφαρμογή μας και στα αντικείμενα, με τη μορφή που μας τα δίνουν τα APIs. Υποστηρίζει πολύ καλά την άρρηκτη λειτουργία και με τα Core Data (το κύριο framework που χρησιμοποιείται για τη διατήρηση δεδομένων στο iOS και OS X και βασίζεται σε γράφους αντικειμένων για τη διαχείρηση των αντικειμένων) κάτι που την καθιστά πολύ χρήσιμη.

Ας δούμε λίγο πώς χρησιμοποιείται η βιβλιοθήκη αυτή για τη δημιουργία και αποστολή HTTP Requests και κάποια άλλα χρήσιμα στοιχεία:

Τα δύο βασικά αντικείμενα της βιβλιοθήκης αυτής είναι δύο:

1) ο RKObjectManager

2) ο AFHTTPClient

Ο RKObjectManager είναι ένα singleton αντικείμενο, στο οποίο μπορούμε να έχουμε πρόσβαση από οποιοδήποτε μέρος της εφαρμογής μας. Αυτός όμως μας είναι άχρηστος αν δεν του δοθεί ένας AFHTTClient ο οποίος στην ουσία μας επιτρέπει να διαμορφώσουμε τις HTTP κλήσεις όπως θέλουμε.

Έτσι για την αρχικοποίηση χρησιμοποιούμε τον εξής κώδικα:

```
NSURL *baseURL = [NSURL URLWithString:@"http://europeana.eu/api/v2"];

AFHTTPClient *client = [[AFHTTPClient alloc] initWithBaseURL:baseURL];
```

```
RKObjectManager *manager = [[RKObjectManager alloc] initWithHTTPClient:client];

[client setDefaultHeader:@"Accept" value:RKMIMETypeJSON];
```

Το BaseUrl είναι το "βασικό" url που θα χρησιμοποιήσουμε στις κλήσεις μας. Ορίζουμε συνήθως το μέγιστο κοινό κομμάτι του URL των HTTP requests ώστε να καλούμε τα διάφορα services χρησιμοποιώντας μετά το σχετικό path που προστίθεται κάθε φορά στο baseURL.

Πολύ προσοχή θέλει το ότι για να χρησιμοποιήσουμε κάποια άλλο baseURL(π.χ. του DPLA) δε φτάνει να αλλάξουμε το BaseURL property του AFHTTPClient, αλλά πρέπει να τον κατασκευάσουμε από την αρχή:

```
NSURL *baseURL = [NSURL URLWithString:@"http://api.dp.la/v2"];

AFHTTPClient* client = [[AFHTTPClient alloc] initWithBaseURL:baseURL];

[client setDefaultHeader:@"Accept" value:RKMIMETypeJSON];

RKObjectManager *managerWithNewBaseURL = [RKObjectManager sharedManager];

managerWithNewBaseURL.HTTPClient = client;
```

Αυτά όσον αφορά την αρχικοποιήση και την αλλαγή client. Για την εκτέλεση μιας κλήσης παίρνουμε το instance του objectManager και καλούμε τη συνάρτηση του HTTPClient με το relative Path του service και τις παραμέτρους που θέλουμε. Τις παραμέτρους τις περνάμε με τη μορφή NSDictionay και το Restkit γνωρίζει πώς να τις περάσει (είτε στο σώμα είτε στην επικεφαλίδα) ανάλογα με το αν θέλουμε να στείλουμε GET ή POST κτλ. Στην εφαρμογή μας, επειδή οι παράμετροι ήταν πολύπλοκες τις περάσαμε manually στο relativePath kai για αυτό και βλέπουμε να είναι nil το όρισμα parameters παρακάτω.

Το σημαντικό είναι ότι οι κλήσεις αυτές γίνοντα ασύγχρονα, χωρίς να μπλοκάρει το main thread εκτελέσης της εφαρμογής. Μόλις λάβουμε την απάντηση από το server, εκτελείται το αντίστοιχο block, είτε το success, είτε το failure ανάλογα με την έκβαση της απάντησης.

```
RKObjectManager *objectManager = [RKObjectManager sharedManager];

relPath = @"search.json";

 NSString *relPathWithParams = [relPath stringByAppendingString:[self
```

```
mm_addParametersFromParameterDic:generalParams]]; //εδώ καλούμε μια συνάρτηση για να φτιάξει manually το path με τις παραμέτρους μέσα

[objectManager.HTTPClient getPath:relPathWithParams

                          parameters:nil

                             success:^(AFHTTPRequestOperation *operation, id responseObject) {

                                 NSError *error;

                                 NSDictionary* json = [NSJSONSerialization JSONObjectWithData:operation.responseData


options:NSJSONReadingMutableContainers


error:&error];

                                 int responseCode = (int) operation.response.statusCode;

                                 if(responseCode == 200) {

                                     if (objQueryDetails.p_eSite == eeQuerySiteEuropeana) {

                                     block(json[@"items"]);

                                     }

                                     else if(objQueryDetails.p_eSite == eeQuerySiteDPLA) {

                                         block(json[@"docs"]);

                                     }

                                 }

                                 else {

                                     block(nil);

                                 }

                             }
```

```
                                failure:^(AFHTTPRequestOperation
*operation, NSError *error) {

                                        block(nil);

                                }

    ];
```

6.3 MBProgressHud

Στο σημείο αυτό φαίνεται κατάλληλο να αναφερθούμε και στη χρήση μιας άλλης βιβλιοθήκης την MBProgressHud [16] που είναι και αυτή ευρέως γνωστή. Είναι χρήσιμη για όταν κάνουμε κλήσεις στο server, ή και γενικά για δραστηριότητες που μπορεί κρατήσουν πολύ και δε θέλουμε κατά τη διάρκειά τους να αλληλεπιδρά ο χρήστης με την εφαρμογή. Μας παρέχει μια όψη που δείχνει ότι κάποια διεργασία λαμβάνει χώρο και προβάλλει κάποιο ενημερωτικό μήνυμα στο χρήστη. Αυτή η όψη προστίθεται, συνήθως, πάνω στο view κάποιου controller και παράλληλα απενεργοποιεί οποιαδήποτε αλληλεπίδραση του χρήστη με το view αυτό. Μόλις τελειώσει η διεργασία μας, την αφαιρούμε και ο χρηστής μπορεί και πάλι να χρησιμοποιήσει το view αυτό. Η χρήση της είναι απλή και γίνεται ως εξής:

6.4 SDWebImage

Κάτι που είναι πολύ σημαντικό στις mobile εφαρμογές είναι να μεταθέτουμε κατα το δυνατόν "βαριές" εργασίες σε background threads ώστε το main thread να είναι ελεύθερο να ανταποκρίνεται στις κινήσεις του χρήστη. Σε αντίθετη περίπτωση, όταν εκτελούμε κάτι χρονοβόρο στο κύριο thread η εφαρμογή δίνει την αίσθηση ότι δεν ανταποκρίνεται και ότι έχει κολλήσει. Όλες οι κλήσεις που προαναφέραμε τόσο μέσω RESTKIT όσο και μέσω PARSE είναι ασύγχρονες και πληρούν την προϋπόθεση αυτή. Μια χρήσιμη βιβλιοθήκη που βοηθά στο ασύγχρονο κατέβασμα εικόνων είναι η SDWebImage [17]. Όχι μόνο παρέχει ευκολία στην ασύγχρονη υλοποίηση, αλλά προσφέρει επιπλέον έναν εξειδικευμένο μηχανισμό caching τόσο στη μνήμη όσο και στο δίσκο για πιο αποδοτική επαναχρησιμοποίηση των εικόνων. Έτσι, λοιπόν, caching και κατέβασμα εικόνων απλοποιείται στην εξής γραμμή κώδικα :

```
[cell.p_imvImage    setImageWithUrl:[NSURL    URLWithString:urlString]
inFrame:cell.frame];
```

Να τονίσουμε ότι για όταν γίνεται cached μια εικόνα χρησιμοποιείται ως κλειδί το url της. Έτσι, όταν καλούμε την παραπάνω μέθοδο και εάν η εικόνα έχει ήδη κατέβει, τότε η

βιβλιοθήκη αυτή φροντίζει ώστε να πάρουμε το τοπικό αντίγραφο και όχι δεν εκκινεί λήψη από το δίκτυο.

Επιπλέον, προσοχή θέλει όταν γίνεται χρήση της βιβλιοθήκης αυτής για το φόρτωμα εικόνων μέσα σε κελιά ενός UITableView ή ενός UICollectionView. Λόγω του ότι τα κελιά αυτά επαναχρησιμοποιούνται κατά το δυνατόν περισσότερο, πρέπει να έχουμε στο νου μας να ακυρώσουμε το φόρτωμα μιας παλιάς εικόνας σε ένα κελί που επρόκειτο να επαναχρησιμοποιηθεί. Αυτός ο κώδικας πρέπει να τοποθετηθεί στις επεκτάσεις των κλάσων UICollectionViewCell που χρησιμοποιούμε:

```
-(void)prepareForReuse {

    [super prepareForReuse];
    [self.p_imvImage cancelCurrentImageLoad];
}
```

Εδώ καλούμε τη μέθοδο της βιβλιοθήκης η οποία ακυρώνει τη συγκεκριμένη λήψη.

6.5 Parse Backend

Κατά την ανάπτυξη μιας web εφαρμογής, συχνά υπάρχει άμεση εξάρτηση από το backend και την υλοποίηση του. Είναι σχετικά σύνηθες μάλιστα, η καθυστέρηση στην υλοποίηση του backend να καθυστερεί και την ανάπτυξη της εφαρμογής. Κάτι που αποτελεί μια πολύ μεγαλη ευκολία, είναι ότι υπάρχουν πλατφόρμες όπως το Parse [18] που μπορούν να μας δώσουν ένα πολύ εύκολα υλοποιήσιμο backend και μερικές ακόμα πολύ σημαντικές λειτουργικότητες για την ανάπτυξη iOS και άλλων mobile ή desktop εφαρμογών. Υπάρχουν και άλλες πλατφόρμες με αντίστοιχη λειτουργία, αλλά το Parse αναδείχτηκε το πιο βολικό και εύχρηστο για τους σκοπούς της διπλωματικής.

Το Parse διαθέτει native SDKs για όλα τα παρακάτω:

- iOS
- OS X
- Android
- Windows Phone
- Windows
- Javascript
- NET
- Unity
- PHP
- Xamarin

κάτι που σημαίνει ότι μπορεί να χρησιμοποιηθεί από μια πολύ μεγάλη γκάμα τεχνολογιών.

Εστιάζει στην παροχή τριών προϊόντων:

1) Το Parse Core: συνοψίζει τον πυρήνα των λειτουργιών της πλατφόρμας, όπως οι αποθήκευση των δεδομένων στο backend μέσω του Parse cloud, η αλληλεπίδραση με τα μέσα δικτύωσης και ένα εύχρηστο dashboard

2) Το Parse Analytics: μια υπηρεσία που παρέχει χρήσιμη πληροφορία για τη χρησιμοποίηση της βάσης από την εφαρμογή (π.χ πόσοι χρήστες την χρησινοποιούν, ποιοι είναι οι καινούριοι χρήστες, πόσες κλήσεις έγιναν προς τη βάση κτλ)

3) Το Parse Push: μια λετουργικότητα που επιτρεπεί την αποστολή notifications προς τις συσκευές των χρηστών, κάτι που δε χρησιμοποιήθηκε από την εφαρμογή μας

Στο σημείο αυτό πρέπει να επισημάνουμε πως η χρήση του Parse από μια εφαρμογή δεν είναι δωρεάν. Μας παρέχει όμως δωρεάν 1.000.000 κλήσεις με όριο 30 κλήσεων / δευτέλοπτο, 20 GB για αποθήκευση αρχείων και 20GB για την αποθήκευση της βάσης, κάτι που κρίθηκε υπεραρκετό για τους σκοπούς της εφαρμογής. Για περαιτέρω χρήση υπάρχει κλιμακωτή χρέωση και οι αναλυτικές χρεώσεις βρίσκονται στον ιστότοπο της πλατφόρμας.

Αυτό που είναι πολύ ενδιαφέρον, είναι ότι δε χρειάζεται να στήσουμε τη βάση και τη μορφή της προτού τη χρησιμοποιήσουμε. Αντίθετα, το μόνο που έχουμε να κάνουμε είναι να φτιάξουμε native αντικείμενα και να τα σώσουμε με κάποιο όνομα. Από εκεί και πέρα τα πάντα αναλαμβόνται από το Parse:η μετατροπή των τύπων, η σειριοποίηση τους, η αποστολή τους, η δημιουργία του αντίστοιχου πίνακα στη βάση στο backend, η αποστολή της απάντησης και της πληροφορίας εάν είναι επιτυχημένη ή όχι. Έτσι, ξεφεύγουμε από το HTTP και τα όσα αναφέραμε περί REST και πάμε σε κάτι ακόμα πιο απλό, καθώς όλα αυτά τα αναλαμβάνει το Parse χωρίς να κάνουμε εμείς κάτι. Καταλαβαίνουμε, λοπόν, ότι είναι ένα πολύ χρήσιμο εργαλείο, φτιαγμένο με μια αφαιρετική λογική προς διευκόλυνση του developer .

Ας δούμε τα βασικά στοιχεία χρήσης του με παραδείγματα κώδικα.

6.5.1 Διαδικασία εγγραφής – ταυτοποίησης χρήστη – αποσύνδεση

Για την εγγραφή κάλούμε την παρακάτω μέθοδο και στο μπλοκ έχουμε το αποτελέσμα της επιτυχίας της κλήσης:

```
[user signUpInBackgroundWithBlock:^(BOOL succeeded, NSError *error) {
    if (!error) {
        //The registration was successful, go to the wall
```

```
        [self mm_dismiss];
        [self.delegate mp_loginSucceededWithUsername:strUsername];
    } else {
        //Something bad has occurred
        NSString      *errorString      =      [[error      userInfo]
objectForKey:@"error"];
        UIAlertView    *errorAlertView    =    [[UIAlertView    alloc]
initWithTitle:@"Erro"        message:errorString        delegate:nil
cancelButtonTitle:@"Ok" otherButtonTitles:nil, nil];
        [errorAlertView show];
    }
}];
```

Για την ταυτοποίηση δημιουργούμε ένα αντικείμενο PFUser, θέτουμε το όνομα και τον κωδικό χρήστη και κάνουμε την αντίστοιχη κλήση.

```
PFUser *user = [PFUser user];
user.username = strUsername;
user.password = strPassword;

[PFUser                     logInWithUsernameInBackground:strUsername
password:strPassword block:^(PFUser *user, NSError *error) {
    if (user) {
        //Open the wall
        [self mm_dismiss];
        [self.delegate mp_loginSucceededWithUsername:strUsername];
    } else {
        //Something bad has ocurred
        NSString      *errorString      =      [[error      userInfo]
objectForKey:@"error"];
        UIAlertView    *errorAlertView    =    [[UIAlertView    alloc]
initWithTitle:@"Error"        message:errorString        delegate:nil
cancelButtonTitle:@"Ok" otherButtonTitles:nil, nil];
        [errorAlertView show];
    }
}];
```

Για αποσύνδεση καλούμε απλά:

```
[PFUser logOut];
```

6.5.2 Ανέβασμα αρχείων στη βάση του Parse

Μετατρέπουμε το επιθυμητό αρχείο (εικόνα, ήχος) σε NSData και χρησιμοποιούμε τη κλάση PFFile.

```
UIImage     *scaledImage                                                =
UIGraphicsGetImageFromCurrentImageContext();
 NSData       *scaledImageData                                          =
UIImagePNGRepresentation(scaledImage);
PFFile *file;
file = [PFFile fileWithName:@"img" data:scaledImageData];
```

Ανεβάζουμε το αρχείο καλώντας τη μέθοδο save. Αν η κλήση είναι επιτυχής, το file θα περιέχει την πληροφορία για το url που αντιστοιχήθηκε από το Parse σε αυτό που μόλις ανεβάσαμε. Πληροφορία που χρειαζόμαστε για να φτιάξουμε το δομή που περιγράφει μια ιστορία.

```
if ([file save]) {

    mdicStory[@"CoverImageUrl"]     = file.url;
    mdicStory[@"CoverImageSize"]                          =      [self
mm_getStringSizeFromSize:CGSizeMake(231, newHeight)] ;
    //check for color
    LEColorPicker *colorPicker = [[LEColorPicker alloc] init];
    LEColorScheme         *colorScheme         =         [colorPicker
colorSchemeFromImage:scaledImage];
    UIColor *mainColor = [colorScheme backgroundColor];
    mdicStory[@"CoverColor"] = [mainColor stringValue];
    //
}
```

Αυτές οι κλήσεις επιλέχθηκε να είναι σύγχρονες γιατι θέλουμε να δούμε ότι το υλικό της ιστορίας θα ανέβει επιτυχώς προτού ανεβάσουμε την δομή της ιστορίας.

6.5.3 Ανέβασμα ιστορίας και σύνδεση της με τον εκάστοτε χρήστη

Αρχικά, κατασκευάζουμε το NSDictionary που περιέχει όλη την πληροφορία για το περιεχόμενο της ιστορίας. Στη συνέχεια φτιάχνουμε ένα PFObject με αυτό και το όνομα του πίνακα της βάσης του Parse στην οποία αντιστοιχεί το NSDictionary αυτό.

```
NSDictionary *dicStory;
PFObject  *objStory        =  [PFObject  objectWithClassName:@"Story"
dictionary:dicStory];
```

Το ενδιαφέρον είναι ότι δε χρειάζεται να προετοιμάσουμε κάτι από τη πλευρά του server για να ανεβάσουμε ένα αντικείμενο. Το Parse κοιτάει αν υπάρχει κάποιος πίνακας με το τίτλο Story. Αν δεν υπάρχει, τον φτιαχνει και κάνει αυτόματα αντιστοίχιση των πεδίων με τα κλειδιά του NSDictionary που ανεβάσαμε. Αν υπάρχει απλά κάνει την αντιστοίχιση καιτο προσθέτει στον ήδη υπάρχοντα πίνακα.

Μετά, το συνδέουμε με τον χρήστη στο οποίο ανήκει:

```
[objStory setObject:[PFUser currentUser].username forKey:@"user"];
```

Ανεβάζουμε το αρχείο. Χρησιμοποιούμε τη μέθοδο saveInBackgroundWithBlock για να εξασφαλίσουμε ότι το ανέβασμα θα γίνει ασύγχρονα. Στο μπλοκ επιστροφής, βλέπουμε αν το ανέβασμα ήταν επιτυχές, ή αν υπήρξε κάποιο πρόβλημα.

```
[objStory saveInBackgroundWithBlock:^(BOOL succeeded, NSError *error)
{

   [MBProgressHUD hideHUDForView:self.view animated:YES];
    if (succeeded){

        NSLog(@"Object Uploaded!");
        FileManager *manager = [FileManager
mm_defaultManagerForUser:self.p_strUsername];
        [manager mm_deleteUserStorywithName:self.p_strTitle];
        [self.navigationController popViewControllerAnimated:YES];
    }
    else{

        NSString *errorString = [[error userInfo]
objectForKey:@"error"];
        NSLog(@"Error: %@", errorString);
    }
}];
```

6.6 CHTCollectionViewWaterfallLayout – Δυναμικό ύψος κελιών

Για τη δημιουργία του waterfall layout που προαναφέρθηκε χρησιμοποιήθηκε η βιβλιοθήκη CHTCollectionViewWaterfallLayout [19]. Η βιβλιοθήκη αυτή χρησιμοποιεί τη γνωστή κλάση UICollectionView [20]. Η κλάση αυτή μας προσφέρει τη δυνατότητα να φτιάχνουμε πολύ εύκολα grid layouts, αλλά και τελείως ιδιαίτερες υλοποιήσεις όπως αυτή που παρέχει η CHTCollectionViewWaterfallLayout. Η χρήση της παραπάνω βιβλιοθήκης είναι εύκολη, καθώς βασίζεται στη λειτουργία του βασικού UICollectionView. Αυτό που διαφοροποιείται είναι ότι πρέπει να ορίσουμε το σταθερό πλάτος των στηλών που επιθυμούμε να έχουμε και το μεταβλητό ύψος για το εκάστοτε κελί. Στη συγκεκριμένη εφαρμογή, αυτό που ήθελε ιδιαίτερη προσοχή είναι ότι το ύψος κάθε κελιού δεν ήταν γνωστό εκ των προτέρων. Αντίθετα, έπρεπε να καθορίζεται δυναμικά, με βάση το περιεχόμενο που ήταν η εικόνα και τα διάφορα πεδία κειμένου.

Για να επιτευχθεί αυτό χρησιμοποιήθηκε η εξής τεχνική. Το κελί του UICollectionView αποτελείται από δύο πράγματα. Ένα UITableview που περιέχει τα πεδία κειμένου και την εκάστοτε εικόνα. Για την εικόνα υπολογίζουμε το απαιτούμενο ύψος με βάση το σταθερό πλάτος και με προυπόθεση να με χαλάει η ανάλυση κρατώντας σταθερό το λόγο πλάτος/ύψους. Για τον υπολογισμό του ύψους του UITableview αρχικοποιούμε ένα κελί του UITableview το οποίο δεν εμφανίζεται κάπου στην οθόνη αλλά χρησίμευε ως μέτρο και του δίνουμε τόσες γραμμές όσες και τα πεδία κειμένου που πρόκειται να δείξει. Στο σημείο αυτό βασιστήκαμε στη νέα δυνατότητα που δίνει το iOS 8 και το Autolayout και επιτρέπει σε ένα κελί του UITableview να υπολογίζει δυναμικά το ύψος του με βάση το περιεχόμενο του και κάποιους περιορισμούς που έχουμε θέσει. Έτσι, το UITableview υπολογίζει το ύψος του και αθροίζοντάς το με το ύψος της εικόνας προκύπτει το μεταβλητό ύψος για το waterfall layout που επιθυμούμε.

6.7 Cocoapods – Σύνδεση βιβλιοθηκών

Παραπάνω είδαμε διάφορες εξωτερικές βιβλιοθήκες που χρησιμοποιήθηκαν από την εφαρμογή. Για τη σύνδεση βιβλιοθηκών συνήθως υπάρχουν δύο επιλογές. Πρώτον, να πάρουμε τα αρχεία της βιβλιοθήκης, να τα βάλουμε στο φάκελο του project μας και να ρυθμίσουμε ό,τι απαιτείται για τη σύνδεση τους κατόπην υποδείξεως του δημιουργού . Δεύτερον, μπορούμε να χρησμοποιήσουμε κάτι πιο απλό με μεγαλύτερη ευελιξία, τα CocoaPods [21]. Όπως προαναφέρθηκε κατά την εισαγωγή, τα Cocoapods είναι ένας manager που ελέγχει τις εξαρτήσεις μιας εφαρμογής. Μπορεί δηλαδή κανείς εύκολα να δηλώσει ότι η εφαρμογή «εξαρτάται», δηλαδή χρησιμοποιεί αυτές τις βιβλιοθήκες και

μάλιστα να δηλώσει και ποιες εκδόσεις τους προτιμά. Δε χρειάζεται να μετακινήσει καθόλου κώδικα στο project της εφαρμογής και η όλη σύνδεση αναλαμβάνεται από τα cocoapods. Βασικό τους πλεονέκτημα είναι ότι αφ' ενός μας γλυτώνουν από την συχνα χρονοβόρα διαδικασία ρύθμισης των παραμέτρων σύνδεσης και αφ΄ετέρου εξ αιτίας της επιλογής δήλωσης της έκδοσης της κάθε βιβλοθήκης μπορούμε εύκολα να αναβαθμίσουμε τις εξαρτήσεις μας, χωρίς να σβήνουμε και να μετακινούμε αρχεία. Για να δηλώσουμε τις εξαρτήσεις της εφαρμογής μας ακολουθούμε την εξής διαδικασία:

1. Πρώτα πρέπει να τα εγκαταστήσουμε:

Τρέχουμε στο terminal την εντολή: `$ sudo gem install cocoapods`

2. Στη συνέχεια φτιάχνουμε ένα απλό αρχείο κειμένου, το ονομάζουμε Podfile και το σώζουμε μέσα στο φάκελο του project.

Αυτό θα περιέχει την πληροφορία για τις εξαρτήσεις της εφαρμογής μας. Τα περιεχόμενα του είναι κάπως έτσι:

```
platform :ios, '8.0'

pod 'AFNetworking', '~> 2.0'

pod 'ARAnalytics', '~> 2.7'
```

Με τη λέξη platform δηλώνουμε το είδος (iOS, OS X) της πλατφόρμας και τη μικρότερη δυνατή έκδοση που στοχεύει η εφαρμογή μας

Στη συνέχεια με τη λέξη pod δηλώνουμε όποια βιβλιοθήκη θέλουμε να συμπεριλάβουμε και την έκδοση που θέλουμε να χρησιμοποιήσουμε.

3.Εγκαθιστούμε τις εξαρτήσεις στο project:

Μετακινούμαστε(στο terminal) στο φάκελο που βρίσκεται το Podfile και τρέχουμε την εντολή: `$ pod install`

Παρατηρούμε ότι παράγεται ένα αρχείο με την κατάληξη .xcworkspace. Από εδώ και πέρα χρησιμοποιούμε αυτό το αρχείο για να ανοίγουμε το project στο Xcode.

Σημειώνεται ότι στην εφαρμογή μας για τη σύνδεση της βιβλιοθήκης RestKit (που είναι και καμία φορά είναι δύστροπη στη σύνδεση της) χρησιμοποιήθηκαν τα CocoaPods, ενώ για τις υπόλοιπες απλή αντιγραφή αρχείων, για ευκολότερη διερεύνηση των αρχείων τους και δυνατή τροποποίηση τους.

6.8 *Αποθήκευση ιστοριών*

Για την αποθήκευση ιστοριών υπήρξαν δύο διαφορετικές προσεγγίσεις που λήφθησαν υπ' όψη. Η μία ήταν η χρήση βάσης δεδομένων και η δεύτερη η αποθήκευση των ιστοριών σε μορφή .plist στο filesystem. Προτιμήθηκε η δεύτερη περίπτωση, καθώς η κάθε ιστορία αποτελούσε ένα NSDictionary με κλειδιά τις διάφορες παραμέτρους της ιστορίας (π.χ. τίτλο, περιγραφή, url εικόνων, url αρχείων ήχου κτλ) και στην ουσία ένα αρχείο .plist είναι ένα σειριοποιημένο NSDictionary. Επιπλέον, διατηρώντας τα αρχεία στη μορφή αυτή, ήταν πολύ εύκολο το ανέβασμά τους στη βάση του Parse, αλλά και η λήψη τους γλυτώνοντας τη διαδικασία της μετατροπής από NSDictionary σε δομή κατάλληλη για τη βάση δεδομένων και το αντίστροφο. Έτσι, φροντίσαμε για τον εκάστοτε χρήστη να δημιουργούνται οι κατάλληλοι φάκελοι όπου σώζονται οι ιστορίες του, αλλά και το υλικό των ιστοριών που πρέπει να διατηρηθεί τοπικά, όπως οι φωτογραφίες και τα αρχείου ήχου.

Παραθέτουμε ένα τέτοιο ενδεικτικό αρχείο που περιγράφει μια ιστορία:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
     <key>ContentHeight</key>
     <real>1320</real>
     <key>ContentWidth</key>
     <real>987</real>
     <key>CoverImageLocalPath</key>
     <string>id=2A456415-B1AE-424A-9795-A0625A768EBDVk!.png</string>
     <key>Description</key>
     <string>This is an exhibition story</string>
     <key>Images</key>
     <array>
       <dict>
             <key>ImageUrl</key>

     <string>http://europeanastatic.eu/api/image?uri=http%3A%2F%2Fcol
lections.rmg.co.uk%2FmediaLib%2F1598%2Fmedia-
1598391%2Flarge.jpg&amp;size=LARGE&amp;type=IMAGE</string>
             <key>Link</key>

     <string>http://www.europeana.eu/portal/record/2022362/_Royal_Mus
eums_Greenwich__http___collections_rmg_co_uk_collections_objects_4048
```

```
3.html?utm_source=api&amp;utm_medium=api&amp;utm_campaign=d5tFQz8Yx</
string>
            <key>LocalImagePath</key>

    <string>:##europeanastatic.eu#api#image?uri=%3A%2F%2Fcollections
.rmg.co.uk%2FmediaLib%2F1598%2Fmedia-
1598391%2Flarge.jpg&amp;size=LARGE&amp;type=IMAGE.png</string>
            <key>Page</key>
            <integer>1</integer>
            <key>frame</key>
            <string>528.000000-215.000000-200.000000-
267.000000</string>
      </dict>
    </array>
    <key>Recordings</key>
    <array>
      <dict>
            <key>LocalRecordingUrl</key>
            <string>MyFilename_FA120A26-E56E-42B8-ADCD-16B19F0F8BF3-
625-0000021BD106B22B.m4a</string>
            <key>Page</key>
            <integer>1</integer>
            <key>frame</key>
            <string>172.000000-305.000000-100.000000-
100.000000</string>
      </dict>
    </array>
    <key>Text</key>
    <array>
      <dict>
            <key>Page</key>
            <integer>1</integer>
            <key>alignment</key>
            <integer>0</integer>
            <key>frame</key>
           <string>465.000000-78.000000-137.000000-56.000000</string>
            <key>isHeader</key>
            <false/>
            <key>text</key>
            <string>Enter some text</string>
      </dict>
```

```
        <dict>
              <key>Page</key>
              <integer>1</integer>
              <key>alignment</key>
              <integer>0</integer>
              <key>frame</key>
             <string>88.000000-101.000000-322.000000-75.000000</string>
              <key>isHeader</key>
              <true/>
              <key>text</key>
              <string>Enter some text</string>
        </dict>
      </array>
      <key>Title</key>
      <string>My Story</string>
</dict>
</plist>
```

Κύριο ρόλο σε αυτή τη διαδικασία έπαιξαν τρείς κλάσεις:

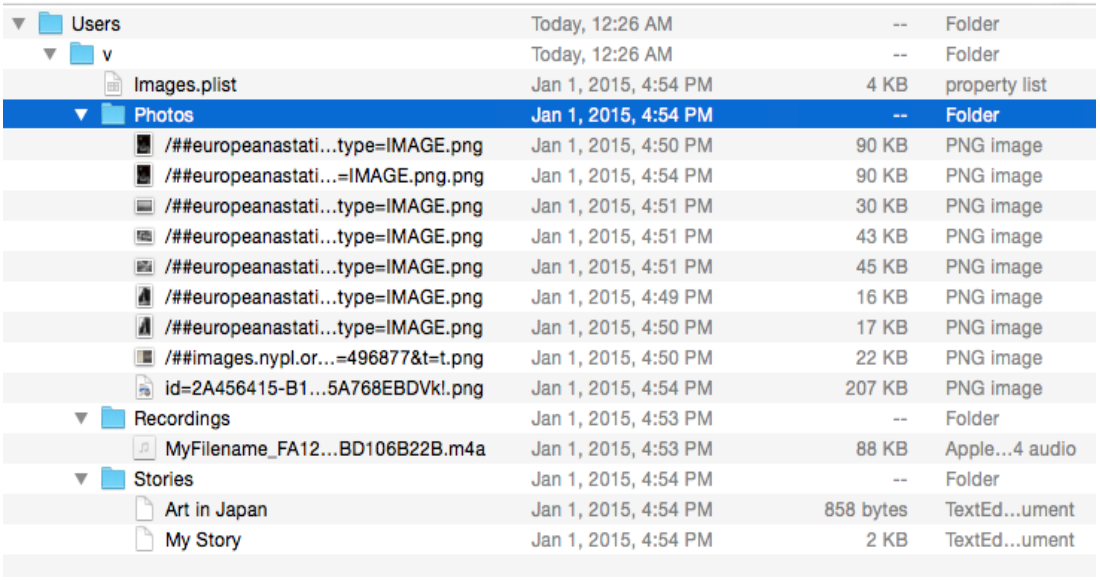
- Filesystem
- FileManager
- StorySaveHelper

Η κλάση Filesystem που ήταν υπεύθυνη για την κατασκευή όλων των paths για τους φάκελους της εφαρμογής. Είναι αυτή που κρατά την πληροφορία για το που πρέπει να σωθεί τι και για ποιον χρήστη.

Η κλάση FileManager είναι η κλάση που προσφέρει όλη τη λειτουργικότητα που έχει να κάνει με τη δημιουργία, τη διαγραφή και των έλεγχο αρχείων στο filesystem. Όπως είναι προφανές η λειτουργία της, είναι άρρηκτα δεμένη με τη κλάση FileSystem.

Τέλος, η κλάση StorySaveHelper είναι αυτή που γνωρίζει το πως πρέπει να σώσει μια ιστορία και τα αρχεία που τη συνοδεύουν, κάνοντας χρήση του FileManager.

Παραθέτουμε κάποιες ενδεικτικές εικόνες της δομής του filesystem με του φακέλους των χρηστών, των ιστοριών και των αρχείων τους.



Εικόνα 6.8-1: Δομή filesystem

6.9 Gesture Recognizers - Touch Events – Responder Chain

Μία πολύ χρήσιμη βιβλιοθήκη που χρησιμοποιήθηκε για τη δημιοργία των ιστοριών είναι η SPUserResizableView [22]. Η κλάση αυτή μας δίνει τη δυνατότητα για τη δημιουργία views τα οποία μπορείς να μεταφέρεις και να τα τους δώσεις το μέγεθος που επιθυμείς με απλές χειρονομίες. Βέβαια, απαιτήθηκαν πολλές τροποποιήσεις για να επιτύχουμε τη λειτουργικότητα που βλέπυμε στην εφαρμογή. Το κύριο πρόβλημα ήταν ότι για τη δημιουργία του editor αυτό που χρησιμοποιήθηκε, βάση της ιστορίας ήταν ένα UIScrollView, πάνω στο οποία ο χρήστης τοποθετεί όλα τα υλικά της ιστορίας. Αυτό αποτέλεσε πρόβλημα, γιατί έπρεπε η εφαρμογή να διαχειριστεί παρόμοιες χειρονομίες σε πολλά διαφορετικά επίπεδα. Για παράδειμα, ο χρήστης μπορεί να κάνει scroll το εξωτερικό UIScrollView, το οποιοδήποτε κείμενο έχει εισαγάγει μέσα σε ένα SPUserResizableView, αλλά και να μετακινήσει το ίδιο το SPUserResizableView. Οπότε καταλαβαίνουμε ότι πρέπει να δοθεί προσοχή στην αλληλουχία με την οποία πρέπει να γίνει η απόκριση στις χειρονομίες του χρήστη από την ιεαραρχία των views του CreateStoryController. Για να επιτευχθεί αυτό έπρεπε να δοθεί προσοχή στο πώς λειτουργεί η απόκριση αλλά και η μετάδοση των γεγονότων στο iOS. Υπάρχουν δύο βασικοί τρόποι για την απόκριση σε γεγονότα:

1) UIGestureRecognizers [23]

Οι UIGestureRecognizers βοηθούν στο να αναγνωρίσουμε μία χειρονομία του χρήστη και να εκτελέσουμε μία μέθοδο με βάση τη χειρονομία αυτή. Υπάρχουν προκαθορισμένοι UIGestureRecognizers που περιλαμβάνουν τις βασικότερες χειρονομίες όπως το άγγιγμα, το σύρσιμο κτλ. Αυτός είναι ο πίνακας με τους προκαθορισμένους recognizers.

Gesture	UIKit class
Tapping (any number of taps)	`UITapGestureRecognizer`
Pinching in and out (for zooming a view)	`UIPinchGestureRecognizer`
Panning or dragging	`UIPanGestureRecognizer`
Swiping (in any direction)	`UISwipeGestureRecognizer`
Rotating (fingers moving in opposite directions)	`UIRotationGestureRecognizer`
Long press (also known as "touch and hold")	`UILongPressGestureRecognizer`

Εικόνα 6.9-1: Προκαθορισμένοι UIGestureRecognizers

Εκτός αυτών, υπάρχει και η δυνατότητα να ορίσουμε εμείς το δικό μας UIGestureRecognizer και να ορίσουμε ακριβώς ποια χειρονομία αναγνωρίζει.

2)Touch events

Αυτά είναι τα γεγονότα που συλλαμβάνουν πιο απλουστευμένα τις επαφές πάνω σε κάποιο view. Μπορούμε, δηλαδή, να λάβουμε κλήσεις για το πότε ξεκίνησε μια επαφή πάνω σε ένα view, κάθε πότε μετακινείται και πότε σταμάτησε. Είναι πιο χαμηλού επιπέδου από ότι οι UIGestureRecognisers και εμείς πρέπει να ορίσουμε τι γίνεται σε κάθε περίπτωση, αλλά παρέχουν μεγαλύτερη ελευθερία. Πολύ σημαντικό στο σημείο αυτό, είναι να δούμε το πως μεταδίδονται απο το iOS τα γεγονότα αυτά, όταν δεν έχουμε μόνο μια όψη, αλλά μια ολόκληρη αλληλουχία όψεων. Σε γενικές γραμμές, αυτό που συμβαίνει είναι ότι όταν ο χρήστης προκαλεί κάποιο γεγονός (π.χ. ενα άγγιγμα), το σύστημα ψάχνει μέσα στην ιεραρχία των όψεων ποια είναι αυτή η όψη που θα αποκριθεί στο γεγονός αυτό. Ξεκινά λοιπόν και "ρωτά" την πιο εδική όψη, δηλαδή την τελευταία στην ιεραρχία αν μπορεί να αποκριθεί. Αν ναι, τότε αποκρίνεται αυτή και η μετάδοση σταματά εκεί, αλλιώς ρωτά τον πατέρα της και πάει λέγοντας. Δημιουργείται έτσι μια αλυσίδα (responder chain [24]) η οποία διασχίζεται κάθε φορά από κάτω προς τα πάνω μέχρι να βρεθεί η όψη που θα ανταποκριθεί στο γεγονός (αν υπάρχει βέβαιο κάποια κατάλληλη για την εκάστοτε δράση του χρήστη).

Εικόνα 6.9-2: Responder chain

Για παράδειγμα, στην παραπάνω εικόνα αν το γεγονός συμβεί στην όψη E τότε η E θα ερωτηθεί αν μπορεί να ανταποκριθεί στο γεγονός. Αν δεν υπάρχει σχετικός κώδικας, το γεγονός θα περαστεί στην όψη C και σε περίπτωση που ούτε αυτή μπορεί να ανταποκριθεί, στην όψη A. Κατά τη διάσχιση αυτής της αλυσίδας, όταν βρεθεί η πρώτη όψη που μπορεί να ανταποκριθεί σταματάει η μετάδοση των γεγονότων.

Επίσης, στο σημείο αυτό πρέπει να αναφέρουμε τι συμβαίνει όταν έχουμε και κάποιον UIGestureRecognizer στην ιεραρχία των όψεων. Οι UIGestureRecognizers δεν συμμετέχουν στην αλυσίδα μετάδοσης που αναφέραμε παραπάνω. Αντίθετα, είναι οι πρώτοι που λαμβάνουν τα γεγονότα και εφόσον αναγνωρίσουν την χειρονομία που τους αντιστοιχεί, δε στέλνονται τα υπόλοιπα γεγονότα στις όψεις.

Αυτή είναι η κλασσική λειτουργία μετάδοσης γεγονότων στο iOS, ωστόσο μπορεί να υπάρχουν στιγμές που εμείς να θέλουμε να παρακάμψουμε αυτό τον τρόπο διάδοσης και να ορίσουμε ότι σε κάποιες ειδικές καταστάσεις πρέπει κάποιος άλλος να παρέμβει. Ή, για παράδειγμα, να μη θέλουμε μερικές φορές κάποιο γεγονός να το αναγνωρίσει ένας UIGestureRecognizer αλλά να το πιάσει κάποιο view και να χρησιμοποιήσουμε τα touch

events. Κάτι τέτοιο χρειάστηκε να γίνει και στον CreateStoryController. Ακολουθούν μερικά ενδεικτικά παραδείγματα:

Ας δούμε πως είναι μια τυπική ιεραρχία των όψεων στον CreateStoryController:

- Controller's View(1ο επίπεδο)
  - UIScrollView(2ο)
    - SPUserResizableView(3ο)
      - UITextView(4ο)
    - SPUserResizableView(3ο)
      - UIImageView(4ο)

Ακολουθεί και μια εικόνα από το νέο feature του Xcode 6 που δείχνει "ζωντανά" μια τέτοια ιεραρχία:

Εικόνα 6.9-3: Ιεραρχία όψεων

Ας δούμε τις κινήσεις του χρήστη που θέλουμε να αντιληφθούμε:

- Σε οποιδήποτε SPUserResizableView αν πατήσει ο χρήστης πρέπει να το επιλέξουμε(πλαίσιο επιλογής) και στη συνέχει να το μετακινήσουμε ή να του αλλάξουμε μέγεθος (Touch Event).
- Στο Controller's View έχουμε έναν UITapGestureRecognizer για να ξέρουμε πως όταν πατήσει ο χρήστης κάπου πάνω στην ιστορία όπου δεν υπάρχει κάποια υλικό

της ιστορίας (π.χ κείμενο ή φωτογραφία) πρέπει να αφαιρέσουμε τα πλαίσιο (UIGestureRecognizer) .

- Όταν ο χρήστης πατήσει σε κάποιο UITextView θέλουμε αν είναι σε editing mode να εμφανιστεί το πληκτρολόγιο αλλίως να ακυρώσουμε την επαφή (Touch event).

1ο Θέμα: Πώς θα θα κάνουμε το UITapGestureRecognizer του view να μην "απορροφά" τα touch events όταν πατήσουμε σε κάποιο SPUserResizableView, αλλά να αναγνωρίζει μόνο όταν ο χρήστης πατά στα κενά σημεία της ιστορίας;

Για τη λύση τέτοιων προβλημάτων υπάρχει η μέθοδος `(BOOL)gestureRecognizer:(UIGestureRecognizer*)gestureRecognizer shouldReceiveTouch:(UITouch *)touch` στο UIGestureRecognizerDelegate που ορίζει πότε ένας recognizer πρέπει να ξεκινά να αναγνωρίζει μια αλληλουχία touch events. Έτσι, εμείς ελέγχουμε και λέμε ότι αν το touch είναι μέσα σε ένα SPUserResizableView τότε δεν πρέπει να το αναγνωρίσεις, ώστε να μεταδοθεί το γεγονός στην αλυσίδα και να αναγνωριστεί από το SPUserResizableView. Ακολουθεί ο αντίστοιχος κώδικας:

```
(BOOL)gestureRecognizer:(UIGestureRecognizer *)gestureRecognizer shouldReceiveTouch:(UITouch *)touch {

    if ([currentlyEditingView hitTest:[touch locationInView:currentlyEditingView] withEvent:nil]) {//if it's inside no handling from the controller

        [self mm_showActionPopUpForView:currentlyEditingView];
        [currentlyEditingView showEditingHandles];
        return NO;
    }//else hide handles from last edited view
    return YES;
}
```

2ο Θέμα: Πως το UITextView δεν θα αποκρίνεται σε κάποιο touch όταν δε βρισκόμαστε σε editing mode, αλλά θα πιάνει το touch το SPUserResizableView (παρότι βρίσκεται πιο πάνω στο responder chain);

Για να λυθεί το θέμα αυτό, επεκτείναμε την κλάση UITextView σε MyTextView και φροντίσαμε, όταν δεν είναι σε edit mode το MyTextView να μεταδίδει τα touch events στο SPUserResizableView που είναι και "πατέρας" του στην ιεραρχία των όψεων. Παραθέτουμε και τον κώδικα:

```
-(void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event {

    if (self.editable) {

        [super touchesBegan:touches withEvent:event];
    }
    else {

        [self.superview touchesBegan:touches withEvent:event];
    }
}

-(void)touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event {

    if (self.editable) {

        [super touchesEnded:touches withEvent:event];
    }
    else {

        [self.superview touchesEnded:touches withEvent:event];
    }
}

-  (void)touchesCancelled:(NSSet *)touches withEvent:(UIEvent *)event
{
    // Notify the delegate we've ended our editing session.
    if (self.editable) {

        [super touchesCancelled:touches withEvent:event];
    }
    else {

        [self.superview touchesCancelled:touches withEvent:event];
    }
}

-(void)touchesMoved:(NSSet *)touches withEvent:(UIEvent *)event {


    if (self.editable) {
```

```
        [super touchesMoved:touches withEvent:event];
    }
    else {

        [self.superview touchesMoved:touches withEvent:event];
    }
}
```

6.10EUROPEANA και DPLA APIS

Για την εκτέλεση των αναζητήσεων για το πολιτισμικό υλικό χρησιμοποιήθηκαν τα Apis της Europeana και της DPLA [5] [7].

Δύο κλάσεις είναι αυτές που συνοψίζουν τη χρήση των APIs αυτών:

- QueryDetails
- SearchItem

6.10.1 QueryDetails

Όλες οι λεπτομέριες για τη διαμόρφωση των κλήσεων προς τα δύο APIs περιέχονται στην κλάση QueryDetails. Εκεί φαίνεται πώς διαμορφώνεται μια κλήση για το κάθε API αναλυτικά. Προκειμένου να πάρουμε τα αποτελεσμάτα που θέλουμε και στη μορφή που επιθυμούμε, πρέπει να θέσουμε τις κατάλληλες url παραμέτρους στην κλήση που κάνουμε προς το εκάστοτε API.

Πολύ χρήσιμο για να δοκιμάσουμε πώς πρέπει να είναι οι κλήσεις μας στο API της Europeana είναι και η κονσόλα που μας παρέχεται, όπου γραφικά βάζουμε τα φίλτρα και βλέπουμε το url που παράγεται. Ας δούμε τις κύριες παραμέτρους τις οποίες θέτουμε στις κλήσεις για τα δύο APIs:

1) Europeana Api

- wskey : Για να χρησιμοποιήσουμε το API πρέπει να εγγραφούμε σε αυτό, ώστε να μας αποσταλεί ένα κλειδί. Στην παράμετρο αυτή, λοιπόν, θέτουμε αυτό ακριβώς το κλειδί που λαμβάνουμε κατά την εγγραφή στο API.
- query : Ο όρος για τον οποίο θέλουμε να ψάξουμε στο API.
- profile : Τι είδους μορφή θέλουμε να έχουν τα αποτελέσματα που θέλουμε να πάρουμε από το API και πόσο αναλυτκά θέλουμε να είναι. Εμείς χρησιμοποιήσαμε την τιμή rich.

- rows : Πόσα αποτελέσματα θέλουμε να πάρουμε πίσω από μια κλήση. Θέσαμε την τιμή 24. Αυτό το πεδίο είναι χρήσιμο για να δηλώσουμε πόσα ακριβώς αποτελέσματα επιθυμούμε.
- start : Η σελίδα την οποία θέλουμε να πάρουμε. Χρησιμοποιείται σε συνάρτηση με το παραπάνω πεδίο για την υλοποίηση του paging ώστε να φορτώνουμε τα αποτελέσματα σταδιακά, καθώς ο χρήστης κάνει scroll προς τα κάτω.
- qf : Χρησιμοποείται για να φιλτράρουμε τα αποτελέσματα της αναζήτησης. Αυτό είναι το πεδίο που χρησιμοποιούμε για να περιορίσουμε την αναζήτηση σε εικόνες ή να κάνουμε χρονική αναζήτηση. Σε περίπτωση που θέλουμε παραπάνω από ένα φίτλρα χρησιμοποιούμε και το qf1 Π.χ. qf = YEAR:[1980+TO+1987], qf1 = TYPE:IMAGE
- reusability : Φίλτρο σχετικά με τα δικαιώματα των επιστρεφόμενων αποτελεσμάτων. Αν είναι ελεύθερα για επαναχρησιμοποίηση, αν δεσμεύονται από κάποιους όρους κτλ.

2) DPLA Api

Ας δούμε και τις αντίστοιχες παραμέτρους με τη μορφή που απαιτεί το API του DPLA:

- api_key : το αποκτούμε αφού κάνουμε εγγραφή
- q : αντίστοιχα με το query
- page_size : αντίστοιχα με το rows
- page : αντίστοιχα με το start
- fields : ένα πεδίο που ορίζουμε ποια πεδία από τα δυνατά πεδία των αποτελεσμάτων επιστροφής μας ενδιαφέρουν ώστε να πάρουμε ακριβώς τα πεδία που μας ενδιαφέρουν και τίποτα περιττό.
- sourceResource.* = χρησιμοποιείται για να φιλτράρουμε τα αποτελέσματα με βάση τα φίλτρα αναζήτησης, π.χ. sourceResource.type = image, sourceResource.date.before ,sourceResource.date.after

6.10.2 SearchItem

Η κλάση SearchItem είναι αυτή που παρέχει στην εφαρμογή τα αποτελέσματα των αναζητήσεων σε μια ενιαία μορφή και γνωρίζει πώς να πάρει την πληροφορία που χρείαζεται, (τίτλος, εικόνα, δικαιώματα, περιγραφή κτλ.) από τα αποτελέσματα που επιστρέφει μια κλήση σε καθένα από τα παραπάνω APIs. Στην κλάση αυτή φαίνεται ακριβώς από ποια πεδία των αποτελεσμάτων επιστροφής παίρνουμε την εκάστοτε πληροφορία που δείχνουμε στην εφαρμογή. Τα αποτελέσματα από τα APIs όπως προαναφέραμε είναι σε μορφή JSON, οπότε

για να πάρουμε την ζητούμενη πληροφρία πρέπει να χρησιμοποιήσουμε το σωστό κλειδί. Ας παρουσιάσουμε τα κλειδιά αυτά:

- Europeana
    - URL εικόνας :edmIsShownBy ή edmPreview
    - τίτλοι : title
    - περιγραφές : dcDescription
    - δημιουργός : dcCreator
    - πάροχος : dataProvider
    - πάροχος δεδομένων: provider
    - σκορ : score
    - όνομα συλλογής : edmDatasetName
- DPLA
    - URL εικόνας :hasView.@id
    - τίτλοι : sourceResource.title
    - περιγραφές : sourceResource.description
    - δημιουργός : dcCreator
    - πάροχος : provider.@id
    - πάροχος δεδομένων: provider.name
    - όνομα συλλογής : isShownAt

7

Επίλογος

7.1 Σύνοψη

Η εφαρμογή που αναπτύχθηκε στο πλαίσιο αυτής της διπλωματικής είχε ως σκοπό και κύρια διαφοροποίηση από προϋπάρχουσες, να δώσει στο χρήστη τη δυνατότητα να δημιουργεί ιστορίες όχι μόνο από δικό του, προσωπικό υλικό, αλλά και από την πληθώρα ιστοριών που βρίσκονται ανοικτά διαθέσιμες στις ψηφιακές βιβλιοθήκες, όπως η Europeana και η DPLA. Θελήσαμε να προωθήσουμε την επαναχρησιμοποίηση του υλικού αυτού, αλλά και να συστήσουμε στα άτομα που δεν γνωρίζουν, την ύπαρξη όλου αυτού του ψηφιακού πλούτου.

Στην παρούσα εργασία έγινε μια περιήγηση στο λειτουργικό σύστημα iOS και προσπαθήσαμε, ώστε ακόμα και κάποιος που δεν έχει ασχοληθεί με το συγκεκριμένο λειτουργικό να πάρει μια βασική ιδέα για τον τρόπο δόμησης μιας τέτοιας εφαρμογής. Είδαμε θέματα που αποτελούν κύριο άξονα σε κάθε σύγχρονη εφαρμογή για κινητές συσκευές, όπως το κομμάτι του networking, της αποθήκευσης δεδομένων τοπικά, της παρουσίασης περιεχομένου και της απόκρισης σε επαφές του χρήστη. Παρουσιάστηκε αναλυτικά η αρχιτεκτονική και η δομή της εφαρμογής, καθώς και τα όποια εργαλεία, ή βιβλιοθήκες που χρησιμοποιήθηκαν. Τέλος, αναλύσαμε και κάποια πιο λεπτά σημεία που είτε αποτέλεσαν τροχοπέδη στη διάρκεια της ανάπτυξης, είτε θεωρήθηκαν πιο πολύπλοκα και απευθύνονται σε άτομα εξοικειωμένα με το περιβάλλον αυτό

7.2 Μελλοντικές επεκτάσεις

Κάποιες ιδέες για μελλοντικές επεκτάσεις είναι οι παρακάτω:

- Η προσθήκη και άλλων APIs για αναζήτηση υλικού.

- Η δυνατότητα εισαγωγής και αποθήκευσης αρχείων βίντεο στη δημιουργία ιστορίας.
- Ακόμα μεγαλύτερη ευελιξία στον τρόπο δημιουργίας μιας ιστορίας, π.χ. , με διάφορους μετρητές χρόνου να δίνεται και χρονικός άξονας στην προβολή των εκάστοτε συστατικών μιας ιστορίας.
- Η προσθήκη push notifications προκειμένου να ενημερώνεται κάποιος χρήστης ότι μια καινούρια ιστορία έχει δημοσιευθεί, ακόμα και όταν έχει κλειστή την εφαρμογή.

8

Βιβλιογραφία

[1] C. Roland, "Digital stories in the classroom," *School Art*, (2006).

[2] M. Wesch, "The Anthropology of YouTube," in *Library of Congres*, Washington, 2008.

[3] «Europeana,» [Ηλεκτρονικό]. Available: http://www.europeana.eu/.

[4] «Europeana Professional,» [Ηλεκτρονικό]. Available: http://pro.europeana.eu/.

[5] «Europeana Labs,» [Ηλεκτρονικό]. Available: http://labs.europeana.eu/api/.

[6] «Digital Public Library of America,» [Ηλεκτρονικό]. Available: http://dp.la/.

[7] «DPLA API Codex,» [Ηλεκτρονικό]. Available: http://dp.la/info/developers/codex/.

[8] «Storehouse,» [Ηλεκτρονικό]. Available: https://www.storehouse.co/.

[9] «Storybird,» [Ηλεκτρονικό]. Available: http://storybird.com/.

[10] «Storify,» [Ηλεκτρονικό]. Available: https://storify.com.

[11] «Storify Enterprise,» [Ηλεκτρονικό]. Available: http://web.livefyre.com/storify-enterprise/.

[12] «JSON,» [Ηλεκτρονικό]. Available: http://www.json.org/.

[13] «Model-View-Controller,» Apple Developer, [Ηλεκτρονικό]. Available: https://developer.apple.com/library/ios/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html.

[14] «Restkit,» [Ηλεκτρονικό]. Available: http://restkit.org/.

[15] «AFNetworking,» [Ηλεκτρονικό]. Available: http://afnetworking.com/.

[16] «MBProgressHUD,» [Ηλεκτρονικό]. Available: https://github.com/jdg/MBProgressHUD.

[17] «SDWebImage,» [Ηλεκτρονικό]. Available: https://github.com/rs/SDWebImage.

[18] «Parse,» [Ηλεκτρονικό]. Available: https://parse.com/.

[19] «CHTCollectionViewWaterfallLayout,» Github, [Ηλεκτρονικό]. Available: https://github.com/chiahsien/CHTCollectionViewWaterfallLayout.

[20] «UICollectionView,» Apple Developer, [Ηλεκτρονικό]. Available: https://developer.apple.com/library/ios/documentation/UIKit/Reference/UICollectionView_class/index.html.

[21] «CocoaPods,» [Ηλεκτρονικό]. Available: http://cocoapods.org/.

[22] «SPUserResizableView,» Github, [Ηλεκτρονικό]. Available: https://github.com/spoletto/SPUserResizableView.

[23] «Gesture Recognizer Basics,» Apple Developer, [Ηλεκτρονικό]. Available: https://developer.apple.com/library/ios/documentation/EventHandling/Conceptual/EventHandlingiPhoneOS/GestureRecognizer_basics/GestureRecognizer_basics.html.

[24] «Event Delivery Responder Chain,» Apple Developer, [Ηλεκτρονικό]. Available: https://developer.apple.com/library/ios/documentation/EventHandling/Conceptual/EventHandlingiPhoneOS/event_delivery_responder_chain/event_delivery_responder_chain.html.

[25] Europeana, «Europeana Data Model Primer,» 14 7 2013. [Ηλεκτρονικό]. Available: http://pro.europeana.eu/files/Europeana_Professional/Share_your_data/Technical_requirements/EDM_Documentation/EDM_Primer_130714.pdf. [Πρόσβαση 11 2014].

[26] Europeana, «EDM Definition,» 17 12 2014. [Ηλεκτρονικό]. Available: http://pro.europeana.eu/files/Europeana_Professional/Share_your_data/Technical_requirements/EDM_Documentation//EDM%20Definition%20v5.2.6_01032015.pdf. [Πρόσβαση 12 2014].