ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Σχεδίαση και Υλοποίηση Συστήματος Προβολής Μουσειακών Χώρων

Ιωάννης Λαγός

Εισηγητής: Σπύρος Συρμακέσης, Καθηγητής

ΠΑΤΡΑ 2020

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή

Πάτρα,

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1.

2.

3.

Υπεύθυνη Δήλωση Φοιτητή

Βεβαιώνω ότι είμαι συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τη συγκεκριμένη εργασία. Η έγκριση της διπλωματικής εργασίας από το Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Πελοποννήσου δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα εκ μέρους του Τμήματος. Η παρούσα εργασία αποτελεί πνευματική ιδιοκτησία του φοιτητή Λαγού Ιωάννη του Προκοπίου που την εκπόνησε. Στο πλαίσιο της πολιτικής ανοικτής πρόσβασης ο συγγραφέας/δημιουργός εκχωρεί στο Πανεπιστήμιο Πελοποννήσου, μη αποκλειστική άδεια χρήσης του δικαιώματος αναπαραγωγής, προσαρμογής, δημόσιου δανεισμού, παρουσίασης στο κοινό και ψηφιακής διάχυσης τους διεθνώς, σε ηλεκτρονική μορφή και σε οποιοδήποτε μέσο, για διδακτικούς και ερευνητικούς σκοπούς, άνευ ανταλλάγματος και για όλο το χρόνο διάρκειας των δικαιωμάτων πνευματικής ιδιοκτησίας. Η ανοικτή πρόσβαση στο πλήρες κείμενο για μελέτη και ανάγνωση δεν σημαίνει καθ’ οιονδήποτε τρόπο παραχώρηση δικαιωμάτων διανοητικής ιδιοκτησίας του συγγραφέα/δημιουργού ούτε επιτρέπει την αναπαραγωγή, αναδημοσίευση, αντιγραφή, αποθήκευση, πώληση, εμπορική χρήση, μετάδοση, διανομή, έκδοση, εκτέλεση, «μεταφόρτωση» (downloading), «ανάρτηση» (uploading), μετάφραση, τροποποίηση με οποιονδήποτε τρόπο, τμηματικά ή περιληπτικά της εργασίας, χωρίς τη ρητή προηγούμενη

έγγραφη συναίνεση του συγγραφέα/δημιουργού. Ο συγγραφέας/δημιουργός διατηρεί το σύνολο των ηθικών και περιουσιακών του δικαιωμάτων.

ΠΕΡΙΛΗΨΗ

Σκοπός της παρούσας πτυχιακής εργασίας είναι η δημιουργία ενός συστήματος που επιτρέπει στο διαχειριστή ενός μουσειακού χώρου να δημιουργήσει ένα χάρτη του χώρου του, μέσω μιας εφαρμογής στο διαδίκτυο, ώστε αυτός στη συνέχεια να μπορεί να προβληθεί σε μια εφαρμογή για κινητές συσκευές .

Αρχικά θα αναφερθούμε στις τεχνολογίες που χρησιμοποιήθηκαν για τις υλοποιήσεις των εφαρμογών για το διαδίκτυο και για τις κινητές συσκευές.

Θα εξεταστεί ο τρόπος σχεδιασμού και υλοποίησης τέτοιων εφαρμογών μέσω των τεχνολογιών που έχουν αναφερθεί, εφιστώντας τη προσοχή στην επικοινωνία μεταξύ των εφαρμογών με τον εξυπηρετητή.

Στη συνέχεια θα γίνει παρουσίαση των προαναφερθέντων εφαρμογών, προβάλλοντας τη ροή των ενεργειών που μπορεί να κάνει ένας χρήστης τόσο στη διαδικτυακή εφαρμογή όσο και στο κινητό του τηλέφωνο.

Τελειώνοντας, θα παραθέσουμε συμπεράσματα που μπορεί να προήλθαν μέσω της ανάπτυξης του λογισμικού και μελλοντικές επεκτάσεις πάνω στο ίδιο.

ABSTRACT

The goal of this thesis is to create a system that allows a museum administrator to create a map of the museum, by using a web application, so that the map can be later viewed through amobile application.

At first there will be a mention in the technologies that were used in order to implement the web and mobile application

Next, we will examine the way of designing and implementing that kind of application via the technologies mentioned before, focusing on the map creation.

Then, there will be a presentation of the aforementioned application, projecting the flow of action that can be be taken by any user in both the web and the mobile application

Last, we will cite the conclusions that came from the software development process and display possible expansions for said software

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον επιβλέπων καθηγητή μου, Σπύρο Συρμακέση, για την πολύτιμη βοήθεια και καθοδήγηση που σκοπό είχε την υλοποίηση της παρούσας πτυχιακής. Επίσης θα ήθελα να ευχαριστήσω τους φίλους μου και την οικογένειά μου που μου στάθηκαν καθ' όλη τη διάρκεια των σπουδών μου.

Περιεχόμενα

Πίνακας εικόνων

ΚΕΦΑΛΑΙΟ 1: ΕΙΣΑΓΩΓΗ

1.1 Περιγραφή αντικειμένου

Σε πολλούς μουσειακούς χώρους, ένας επισκέπτης για να μπορέσει να περιηγηθεί μέσα σε αυτά, να δει και να κατανοήσει την ιστορία των εκθεμάτων θα πρέπει ή να πάει με έναν ειδικό πάνω στο θεματικό περιεχόμενο των εκθεμάτων του χώρου ή να έχει μαζί του ένα εγχειρίδιο που να περιέχει πληροφορίες σχετικές με τα εκθέματα. Και στις 2 περιπτώσεις το κόστος είναι αρκετό, καθώς η περιήγηση με κάποιον ειδικό θα είναι επί πληρωμή εις βάρος του επισκέπτη και για να συλλεχθούν οι πληροφορίες, να επεξεργαστούν και να εντυπωθούν σωστά σε ένα εγχειρίδιο περιλαμβάνει ένα κόστος εις βάρος της διοίκησης του μουσειακού χώρου.

Αυτό το κόστος έρχεται να μειώσει ή παρούσα πτυχιακή εργασία, καθώς δίνει τη δυνατότητα υλοποίησης ενός χάρτη του εκάστοτε χώρου, κατηγοριοποιώντας τον σε κτήρια, ορόφους, δωμάτια και εκθέματα ώστε να μπορεί ο κάθε επισκέπτης να εξερευνήσει εις βάθος τις πτυχές του χώρου μέσα από το κινητό του τηλέφωνο.

Η πτυχιακή εργασία χωρίζεται στο Web κομμάτι, είσοδο στο οποίο δίνεται μόνο στο διαχειριστή του μουσειακού χώρου μέσω ενός φυλλομετρητή(web browser), και περιλαμβάνει την δημιουργία και επεξεργασία διαφορετικών χώρων με τους ορόφους, δωμάτια και εκθέματα μέσα σε αυτό, και στο Mobile κομμάτι, στο οποίο έχει κυρίως πρόσβαση ο επισκέπτης, ώστε να μπορεί να δει το χάρτη του χώρου που έχει επιλέξει καθώς και πληροφορίες για τα εκθέματα που βρίσκονται μέσα στα διάφορα δωμάτια, αλλά και ο διαχειριστής, καθώς χρησιμοποιώντας τα στοιχεία λογαριασμού του μπορεί να επεξεργαστεί ή να προσθέσει εκθέματα στους ήδη διαμορφωμένους χάρτες που διαχειρίζεται από οποιαδήποτε συσκευή.

1.2 Δομή Πτυχιακής

Η πτυχιακή χωρίζεται σε 5 κεφάλαια:

- Στο κεφάλαιο 1 γίνεται μια εισαγωγή στο αντικείμενο της πτυχιακής καθώς και στη δομή που θα ακολουθήσει το συγκεκριμένο έγγραφο.
- Στο κεφάλαιο 2 θα γίνει αναφορά στις τεχνολογίες που χρησιμοποιήθηκαν για να υλοποιηθεί η πτυχιακή εργασία
- Στο κεφάλαιο 3 θα γίνει μια περιγραφή του σχεδιασμού και των μεθόδων υλοποίησης τόσο του Web μέρους όσο και του Mobile .
- Στο κεφάλαιο 4 θα γίνει παρουσίαση του πλήρες λογισμικού.
- Στο κεφάλαιο 5 παραπέμπονται τα συμπεράσματα μέσα από τη δημιουργία του λογισμικού καθώς και μελλοντικές επεκτάσεις πάνω σε αυτό.

ΚΕΦΑΛΑΙΟ 2: ΕΡΓΑΛΕΙΑ ΚΑΙ ΤΕΧΝΟΛΟΓΙΕΣ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΗΘΗΚΑΝ

2.1 Εισαγωγή

Οι τεχνολογίες που χρησιμοποιήθηκαν χωρίζονται στις κατηγορίες:

- Τεχνολογίες front-end διαδικτυακής εφαρμογής
- Τεχνολογίες front-end εφαρμογής για κινητά τηλέφωνα
- Τεχνολογίες back-end για την επικοινωνία των εφαρμογών με τη βάση δεδομένων

2.1.1 Τι είναι το front-end;

Το front-end κομμάτι μιας εφαρμογής είναι η τεχνική του να μετατρέπονται τα δεδομένα σε ένα γραφικό περιβάλλον φιλικό ως προς το χρήστη, ώστε να αποδοθεί με σωστό τρόπο η διαχείριση του back-end κομματιού της εφαρμογής.

2.1.2 Τι είναι το back-end;

Το back-end κομμάτι μιας εφαρμογής είναι το σημείο γύρω από το οποίο ‘χτίζεται’ η λειτουργικότητα μιας εφαρμογής και πρέπει πάντα να είναι σωστά διαμορφωμένο ώστε να εξασφαλίζεται η σταθερότητα της εφαρμογής. Αποτελείται συνήθως από την ίδια την εφαρμογή, τον εξυπηρετητή και τη βάση δεδομένων και στην ουσία γεφυρώνει το χάσμα μεταξύ των τριών στοιχείων.

2.2 Εργαλεία ανάπτυξης

2.2.1 Sublime Text 3

Για την υλοποίηση του κώδικα χρησιμοποιήθηκε το Sublime Text 3, το οποίο είναι ένα δωρεάν πρόγραμμα ανοιχτού κώδικα γραμμένο στη γλώσσα προγραμματισμού Python.

Υποστηρίζει πολλές προγραμματιστικές γλώσσες και επιτρέπει την εγκατάσταση και χρήση προσθέτων(plugins) όπως το xDebug που όταν εγκατασταθεί και παραμετροποιηθεί σωστά, επιτρέπει στο χρήστη να δει τη ροή των δεδομένων μέσα στον κώδικα σε πραγματικό χρόνο για να μπορέσει να αποσφαλματώσει τυχόν προβλήματα.

2.2.2 XAMPP

Το XAMPP είναι ένα πακέτο προγραμμάτων ελεύθερου λογισμικού και ανοιχτού κώδικα. Τα πιο βασικά προγράμματα που περιέχει είναι ο εξυπηρετητής ιστοσελίδων HTTP Apache, το σύστημα διαχείρισης βάσεων δεδομένων MySQL και έναν διερμηνέα για κώδικα γραμμένο σε γλώσσες προγραμματισμού PHP και Perl, δίνοντας το όνομα XAMPP από τα αρχικά κάθε εργαλείου με το X να είναι ο όρος «cross-platform» , δηλαδή ένα λογισμικού ανεξάρτητο πλατφόρμας αφού μπορεί να τρέξει και σε Microsoft Windows, Macintosh OS, Linux Και Solaris. Ο σκοπός δημιουργίας του ήταν η ανάπτυξη και δοκιμή ιστοσελίδων τοπικά σε έναν υπολογιστή χωρίς να χρειάζεται σύνδεση στο διαδίκτυο. Μπορεί όμως να χρησιμοποιηθεί και για τη φιλοξενία ιστοσελίδων και πρόσβαση αυτών μέσω του Παγκόσμιου Ιστού, ασφαλίζοντας ταυτόχρονα σημαντικά μέρη του XAMPP με ένα εργαλείο προστασίας με κωδικό.

2.2.2.1 Εξυπηρετητής Ιστοσελίδων Apache

Ο Apache HTTP ή γνωστός ως απλά Apache είναι ένας εξυπηρετητής του παγκόσμιου ιστού και είναι ένα λογισμικό ανοιχτού κώδικα. Αναπτύσσεται και συντηρείται από μια κοινότητα ανοιχτού κώδικα υπό την επίβλεψη του Ιδρύματος Λογισμικού Apache.

Η ανάπτυξη του Apache βασίστηκε στο NCSA HTTPd, και ξεκίνησε όταν ο κώδικας στο NCSA έπαψε να αναπτύσσεται. Έπαιξε ένα μεγάλο ρόλο στην εξάπλωση του Παγκόσμιου Ιστού, καθώς ήταν η πρώτη βιώσιμη εναλλακτική επιλογή απέναντι στον εξυπηρετητή της Netscape.

Από το 1996 θεωρείται το πιο δημοφιλές λογισμικό εξυπηρετητή, φτάνοντας τα 100 εκατομμύρια ιστοσελίδες το 2009 ενώ το 2019 υπολογίζεται πως εξυπηρετεί το 29% των ενεργών ιστοσελίδων.

2.2.2.2 Σύστημα διαχείρισης βάσεων δεδομένων MySQL

Η MySQL είναι ένα ανοιχτού κώδικα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων και διατίθεται δωρεάν προς χρήση. Το πρόγραμμα τρέχει σε έναν εξυπηρετητή παρέχοντας πρόσβαση πολλών χρηστών σε ένα σύνολο βάσεων δεδομένων. Η MySQL είναι μέρος του LAMP, του πιο δημοφιλούς συνδυασμού πλατφορμών εκτέλεσης ιστοσελίδων που βγαίνει από τα αρχικά των συστημάτων που το αποτελούν, Linux,Apache,MySQL,PHP/Perl/Python.

Η MySQL δημιουργήθηκε από την εταιρεία MySQL AB, που ιδρύθηκε από τους David Axmark, Allan Larson και Michael 'Monty' Widenius. Έλαβε το όνομα της από την κόρη του Widenius, My. Η ανάπτυξη της ξεκίνησε από τους Widenius και Axmark, το 1994, έχοντας μια πρώτη έκδοση το 1995. Αρχικά δημιουργήθηκε για να παρέχει μια αποδοτική και αξιόπιστη επιλογή διαχείρισης δεδομένων τόσο για επαγγελματική όσο και για προσωπική χρήση.

Λόγω του ότι αποτελεί μέρος του LAMP, πρόκειται για το πιο δημοφιλές σύστημα διαχείρισης βάσεων δεδομένων και χρησιμοποιείτε από εταιρείες όπως το Facebook, το YouTube και το WordPress.

2.2.3 Android Studio

Το Android Studio είναι ένα ολοκληρωμένο προγραμματιστικό περιβάλλον για ανάπτυξη εφαρμογών για το λογισμικό Android. Ανακοινώθηκε το Μάιο του 2013 και η πρώτη σταθερή έκδοσή του βγήκε το Δεκέμβριο του 2014. Η ανάπτυξη των εφαρμογών γίνεται στις γλώσσες Java και Kotlin, με την τελευταία να είναι από τη Google η επιθυμητή γλώσσα για το περιβάλλον αυτό.

2.3 Τεχνολογίες

2.3.1 Τεχνολογίες front-end διαδικτυακής εφαρμογής

2.3.1.1 HTML

Για τη δημιουργία κάθε σελίδας τον βασικό ρόλο έχει η γλώσσα σήμανσης HTML(Hyper Text Markup Language) καθώς είναι αυτή που καθορίζει τη βασική δομή της σελίδας. Για να κατανοηθεί πλήρως η HTML5, θα αναφερθούν κάποια πράγματα σχετικά την προέλευση της HTML. Η HTML ξεκίνησε, ως μια ιδέα του Tim Berners - Lee, ως ένα τρόπο επικοινωνίας και ανταλλαγής διαφόρων πληροφοριών μεταξύ ερευνητών από όλο τον κόσμο. Η πρώτη έκδοση της HTML αποτελούταν από 20 στοιχεία για τον αρχικό και σχετικά απλό σχεδιασμό της, πολλά απο τα οποία ήταν εμπνευσμένα από την γλώσσα SGML(Standard Generalized Markup Language), ένα μεγάλο σύστημα επεξεργασίας εγγράφων. Με τη πάροδο του χρόνου, η γλώσσα HTML εξελισσόταν εισάγοντας νέα στοιχεία και προσαρμογές στους κανόνες της αυξάνοντας έτσι και τον αριθμό της έκδοσης της φτάνοντας στη τελευταία έκδοση, την HTML5.

Η σημερινή έκδοση της HTML, η HTML5, αποτελώντας εξέλιξη των προηγούμενων εκδόσεων από αυτή, κληρονομεί ένα μεγάλο ποσοστό των στοιχείων των προκατόχων της γεγονός που σημαίνει πως ένα μεγάλο μέρος της HTML5 λειτουργεί και σε παλιότερες εκδόσεις φυλλομετρητών.

2.3.1.2 CSS3

Η CSS(Cascading Style Sheets) είναι μια γλώσσα που χρησιμοποιείται σε συνδυασμό με την HTML για να καθοριστεί η εμφάνιση μιας ιστοσελίδας.

Ξεκίνησε το 1996 από την W3C(World Wide Web Consortium) με στόχο να μπορούν να αποθηκευτούν όλοι οι κανόνες μορφοποίησης ξεχωριστά από το περιεχόμενο και τη δομή, τεχνική που εφαρμοζόταν μέχρι τότε, αυξάνοντας τον χώρο που καταλαμβάνει μια ιστοσελίδα σε έναν εξυπηρετητή, καθώς κάθε στοιχείο HTML είχε το δικό του κανόνα μορφοποίησης.

Με την εισαγωγή των αρχείων με κατάληξη .css και την υποστήριξή τους από την HTML4, ο σχεδιαστής ενός ιστοτόπου θα μπορούσε να εισάγει όλους τους κανόνες μορφοποίησης αυτού σε ένα αρχείο που στη συνέχεια θα τροφοδοτούσε τα στοιχεία της HTML με τους αντίστοιχους κανόνες χωρίς να χρειάζεται να τους επαναλάβει στον κώδικα για κάθε στοιχείο ξεχωριστά.

Όπως και η HTML5 έτσι και η CSS3 αποτελεί εξέλιξη της CSS και αυτή με τη σειρά της έχει κληρονομήσει ένα μεγάλο μέρος των κανόνων των προκατόχων της. Αυτό σημαίνει ότι και στην CSS3, σε ένα μεγάλο ποσοστό οι κανόνες μορφοποίησης θα λειτουργούν και σε παλιότερες εκδόσεις φυλλομετρητών.

2.3.1.3 JavaScript

Η JavaScript είναι μια γλώσσα προγραμματισμού που χρησιμοποιείτε για να κάνει το περιεχόμενο μιας ιστοσελίδας διαδραστικό και για τη ασύγχρονη επικοινωνία και ανταλλαγή δεδομένων μεταξύ χρήστη και εξυπηρετητή. Η JavaScript είναι μια γλώσσα προγραμματισμού της κατηγορίας client-side, που σημαίνει πως οποιαδήποτε λειτουργία της, εκτελείται από το φυλλομετρητή του χρήστη και όχι από τον εξυπηρετητή της σελίδας(server-side) όπως γίνεται στη γλώσσα PHP.

Η JavaScript δημιουργήθηκε αρχικά από τον Brendan Eich της εταιρείας Netscape με την ονομασία Mocha, μετονομάστηκε στη συνέχεια σε LiveScript και κατέληξε στην ονομασία JavaScript καθώς η ανάπτυξη της είχε μεγάλη επιρροή από τη γλώσσα προγραμματισμού Java. Ο Eich δημιούργησε τη τώρα JavaScript με στόχο τη δημιουργία δυναμικών στοιχείων HTML ανάλογα με τα δεδομένα που έχει στη διάθεση του το κομμάτι κώδικα και τις ενέργειες του χρήστη μέσα σε μία ιστοσελίδα.

Οι εντολές της μπορούν να ανταποκριθούν σε διάφορα συμβάντα ενός χρήστη, όπως για παράδειγμα το κλικ σε ένα κουμπί ή η είσοδος δεδομένων σε μια φόρμα και αποστολή αυτών σε έναν εξυπηρετητή, χωρίς να γίνει ανακατεύθυνση του χρήστη στην ίδια ή σε άλλη σελίδα.

2.3.1.4 jQuery

Η jQuery είναι μια δημοφιλή και πλούσια σε χαρακτηριστικά βιβλιοθήκη βασισμένη πάνω στη JavaScript. Η βιβλιοθήκη αυτή διευκολύνει την μετακίνηση και επεξεργασία HTML σελίδων, τη διαχείριση συμβάντων (event handling), τη δημιουργία εφέ και την ασύγχρονη επικοινωνία με έναν εξυπηρετητή.

Κυκλοφόρησε το 2006 από τον John Resig, όντας επηρεασμένος από την βιβλιοθήκη cssQuery, που ήταν και αυτή βασισμένη στη JavaScript και έδινε τη δυνατότητα στον προγραμματιστή να επιλέγει στοιχεία της HTML μέσω των ετικετών(tags) που είχαν χρησιμοποιηθεί σε μία σελίδα.

Όπως και στη JavaScript, οι εντολές της jQuery μπορούν να ανταποκριθούν σε ενέργειες που εκτελεί ένας χρήστης σε μια σελίδα.

2.3.1.5 jCanvas

Η βιβλιοθήκη του jCanvas, δημιουργήθηκε από τον Caleb Evans και βοηθάει στη δημιουργία σχημάτων πάνω στο στοιχείο canvas της HTML μέσω της βιβλιοθήκης jQuery.

2.3.1.6 AJAX

Η Ajax είναι ένα σύνολο προγραμματιστικών μεθόδων μου χρησιμοποιούνται για να υλοποιηθούν ασύγχρονες εφαρμογές διαδικτύου από τη πλευρά του χρήστη. Μέσω της Ajax μπορούν οι εφαρμογές να επικοινωνήσουν και να αποστείλουν ή να λάβουν δεδομένα από μια βάση δεδομένων ασύγχρονα. Η σύνηθες μορφή που έχουν τα δεδομένα είναι σε JSON, δηλαδή ένα αντικείμενο του οποίου οι πληροφορίες είναι οριοθετημένες με κόμματα.

2.3.2 Τεχνολογίες front-end εφαρμογής για κινητά τηλέφωνα

2.3.2.1 Java

Η Java είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού που σχεδιάστηκε από την Sun Microsystems. Το πιο βασικό χαρακτηριστικό και πλεονέκτημα της έναντι σε άλλες γλώσσες προγραμματισμού είναι η ανεξαρτησία του λειτουργικού συστήματος και πλατφόρμας. Τα προγράμματα που είναι φτιαγμένα σε Java τρέχουν το ίδιο και σε Windows, Macintosh, Linux και Unix χωρίς να χρειαστεί να γίνει εκ νέου η μεταγλώττιση ή να αλλάξει ο πηγαίος κώδικας για κάθε λειτουργικό σύστημα.

Η Java δημιουργήθηκε ως ένα εργαλείο ανάπτυξης λογισμικού σε μικρο-συσκευές από τη Sun. Ο James Gosling, που ήταν εργαζόμενος της Sun, μετά από πειραματισμούς στη γλώσσα C++, δημιούργησε την Oak. Παρόλο που η Oak ήταν βασισμένη στη C++, είχε πιο έντονο αντικειμενοστραφή χαρακτήρα από αυτή και χαρακτηριζόταν για την απλότητα της. Μετά τη μετονομασία της σε Java, αφού η ομάδα ανάπτυξη της ανακάλυψε πως δεν το όνομα Oak ήταν ήδη καταχωρημένο, έκανε τη πρώτη της εμφάνιση στη βιομηχανία της πληροφορικής το 1995 μέσω του συνεδρίου Sun World 1995. Το Δεκέμβριο του 1995, οι IMB, Borland, Mitsubishi Electronics Sybase και Symantec ανακοίνωσαν σχέδια να χρησιμοποιήσουν τη Java για τη δημιουργία λογισμικού. Έκτοτε, η Java, ακολουθεί μια ανοδική πορεία και είναι πλέον μια από τις πιο γνωστές γλώσσες στον κόσμο της πληροφορικής. Στις 13 Νοεμβρίου 2006 η Java έγινε μια γλώσσα ανοιχτού κώδικα, όσον αφορά το μεταγλωττιστή (javac) και το πακέτο ανάπτυξης (JDK, Java Development Kit)

2.3.3 Τεχνολογίες back-end για την επικοινωνία των εφαρμογών με τη βάση δεδομένων

2.3.3.1 PHP

Η PHP είναι μια γλώσσα προγραμματισμού που χρησιμοποιείται στη δημιουργία ιστοσελίδων με δυναμικό περιεχόμενο. Σε αντίθεση με τη JavaScript , η PHP είναι μια γλώσσα που εκτελείτε στον εξυπηρετητή (server-side), την επεξεργάζεται δηλαδή ένας

διακομιστής και στη συνέχεια φέρνει τα αποτελέσματα της επεξεργασίας στον περιηγητή του χρήστη.

Η PHP ξεκίνησε από τον Rasmus Lerdorf to 1994 όταν ο ίδιος, όντας φοιτητής, δημιούργησε ένα script στη γλώσσα προγραμματισμού C για προσωπική του χρήση και το ονόμασε php.cgi. Αφού μοιράστηκε το script σε φίλους του, αυτοί με τη σειρά τους ζητούσαν επεκτάσεις πάνω στις δυνατότητες του script, έχοντας ως αποτέλεσμα να δημιουργηθεί το 1997 μια δεύτερη έκδοση της γλώσσας, γνώση τότε ως PHP/FI (Personal Home Page/Form Interpreter), την οποία χρησιμοποιούσαν και αρκετές ιστοσελίδες. Την ίδια χρονιά οι Andi Gutmans και Zeev Suraski ξαναέγραψαν τη γλώσσα από την αρχή, βασισμένη στη προηγούμενη μορφή της, θέτοντας έτσι της βάσης για τη μέχρι τώρα εξέλιξη της, καθώς οι παραπάνω δημιούργησαν τη εταιρεία Zend ή οποία είναι και υπεύθυνη για την ανάπτυξη και εξέλιξη της γλώσσας.

Η PHP πλέον βρίσκεται στην έκδοση 7.3.8 και χρησιμοποιείται από πληθώρα εφαρμογών και ιστοσελίδων, κάνοντας την μία από τις πιο διαδεδομένες τεχνολογίες του Παγκοσμίου Ιστού.

ΚΕΦΑΛΑΙΟ 3: ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΥΛΟΠΟΙΗΣΗ

3.1 Εκδόσεις εφαρμογής

Η παρούσα πτυχιακή εργασία περιλαμβάνει υλοποιήσεις για δύο διαφορετικά συστήματα: Η μία υλοποίηση λογισμικού δημιουργήθηκε για πρόσβαση του διαχειριστή ενός μουσειακού χώρου μέσω του διαδικτύου, για να αρχικοποιήσει ένα χώρο και να σχεδιάσει το χάρτη ενώ η δεύτερη υλοποίηση δημιουργήθηκε αρχικά για να έχει πρόσβαση ο απλός χρήστης-επισκέπτης του μουσείου για τη περιήγηση του μέσα στο χώρο του μουσείου μέσω του κινητού του τηλεφώνου,αλλά και για τον διαχειριστή της διαδικτυακής εφαρμογής καθώς δίνεται η δυνατότητα να τροποποιήσει το περιεχόμενο των ήδη δημιουργημένων δωματίων.

3.2 Περιβάλλον ανάπτυξης

Η διαδικτυακή εφαρμογή αναπτύχτηκε αρχικά χρησιμοποιώντας το XAMPP και στη συνέχεια έγινε η μεταφορά της στην εταιρεία Web Hosting Hostinger. Ως πρόγραμμα επεξεργασίας κειμένου χρησιμοποιήθηκε το Sublime Text 3, ως εξυπηρετητής ο Apache/2.4, η έκδοση της PHP είναι 5.6 και ως βάση δεδομένων χρησιμοποιήθηκε η MariaDB έκδοσης 10.2.

Η εφαρμογή για κινητά τηλέφωνα αναπτύχτηκε στο Android Studio έκδοσης 3.1.3 με χρήση της γλώσσας Java και η εφαρμογή είναι για κινητά με επίπεδο API(Automatic Programming Interface) 22 και πάνω δηλαδή έκδοση λογισμικού 5.0(Lollipop) και μετά.

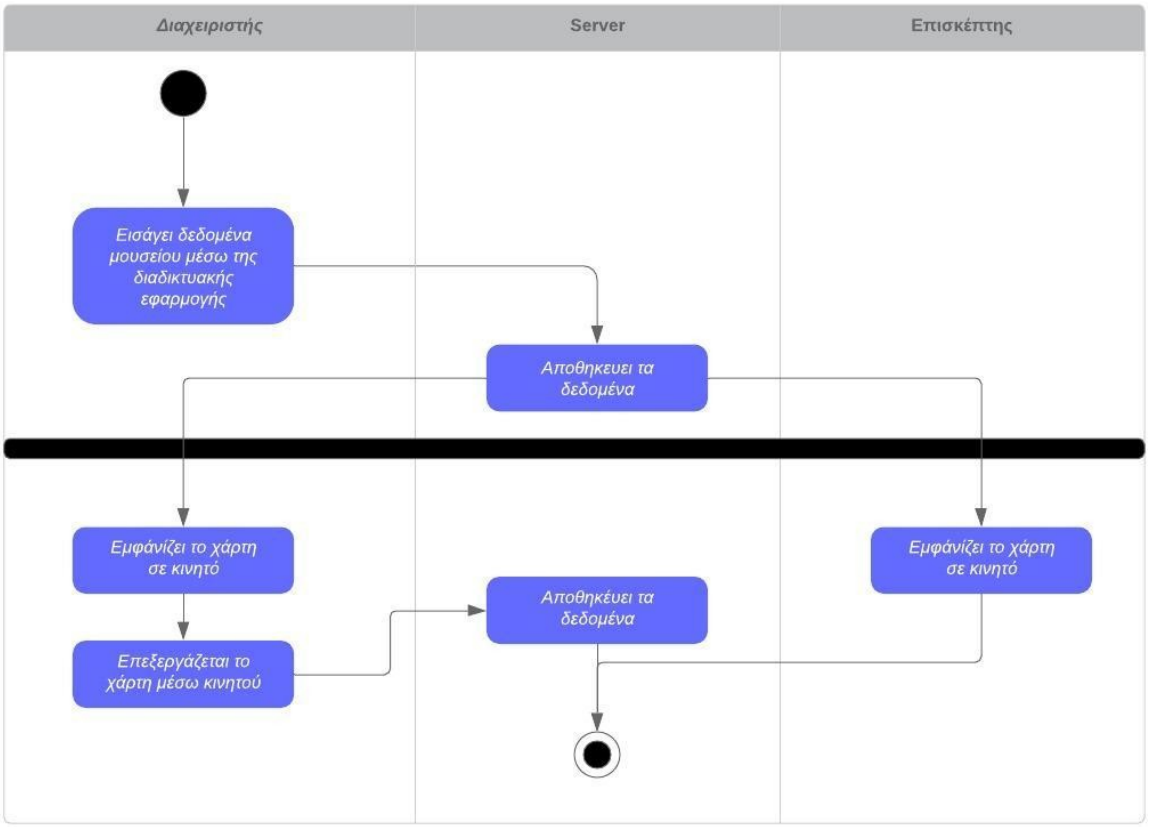
3.3 Σχεδιασμός υλοποιήσεων

Ο σχεδιασμός των υλοποιήσεων βασίστηκε στις δραστηριότητες που θα μπορούν να κάνουν οι χρήστες πάνω στο λογισμικό.

- Ο διαχειριστής ενός μουσείου μπορεί να καταχωρήσει , μέσω της διαδικτυακής εφαρμογής, ένα μουσείο, τους ορόφους του, να επεξεργαστεί τον χάρτη του ορόφου εισάγοντας του δωμάτια και να εισάγει εκθέματα μέσα στα δωμάτια.

- Ο επισκέπτης μπορεί, στο κινητό του τηλέφωνο να εμφανίσει το χάρτη του μουσείου που θέλει να δει, εισάγοντας το αναγνωριστικό του μουσείου (museumid)

Στο ρόλο του διαχειριστή δίνεται και η επιπλέον δυνατότητα να μπορεί να επεξεργαστεί τα ήδη υπάρχοντα δωμάτια, επεξεργάζοντας ή προσθέτοντας ένα έκθεμα, από οποιοδήποτε κινητό τηλέφωνο, εισάγοντας τα στοιχεία σύνδεσης του που χρησιμοποιεί στη διαδικτυακή εφαρμογή



Εικόνα 1: Διάγραμμα δραστηριοτήτων εφαρμογής

3.4 Περιγραφή βάσης δεδομένων

3.4.1 Περιγραφή πινάκων βάσης δεδομένων

Η βάση δεδομένων χρησιμοποιεί MySQL και τα δεδομένα αποθηκεύονται σε 5 πίνακες:

- userinfo: Στον συγκεκριμένο πίνακα αποθηκεύονται οι λογαριασμοί των χρηστών της διαδικτυακής εφαρμογής, οι χρήστες δηλαδή που είναι και οι διαχειριστές του μουσειακού χώρου. Ο πίνακας αποτελείται από τα πεδία

    1. id, που είναι το αναγνωριστικό του χρήστη μέσα στον πίνακα,
    2. fullname, που αποθηκεύει το ονοματεπώνυμο του χρήστη
    3. password, που αποθηκεύει το password του χρήστη

    - museum: Σε αυτόν τον πίνακα αποθηκεύονται τα δεδομένα για τους μουσειακούς χώρους. Ο πίνακας αποτελείται απο τα εξής πεδία:

        1. museumid, που το αναγνωριστικό του μουσείου μέσα στον πίνακα της βάσης δεδομένων
        2. museumname, που αποθηκεύει το όνομα του μουσείου
        3. userid, που πρόκειται για αναφορά στον πίνακα userinfo ώστε να υπάρχει σχέση μεταξύ διαχειριστή μουσείου και μουσείο

- floor: Σε αυτόν τον πίνακα αποθηκεύονται δεδομένα σχετικά με τους ορόφους κάθε μουσείου. Τα πεδία του πίνακα είναι:

    1. floorid, που είναι το αναγνωριστικό του ορόφου μέσα στον πίνακα της βάσης δεδομένων
    2. floorname, που αποθηκεύει το όνομα του ορόφου
    3. museumid, που πρόκειται για αναφορά στον πίνακα Museum ώστε να υπάρχει σχέση μεταξύ μουσείου και ορόφου

- room: Ο συγκεκριμένο πίνακας περιέχει πιο πολλά δεδομένα από ότι οι προηγούμενοί του ,καθώς αποθηκεύει και τις συντεταγμένες του κάθε δωματίου πάνω στον καμβά, το χρώμα του δωματίου και τον τύπο του. Πιο συγκεκριμένα ο πίνακας αποτελείται από:

    1. roomid, που είναι το αναγνωριστικό του δωματίου μέσα στον πίνακα της βάσης δεδομένων

    2. irlroomid, που είναι το αναγνωριστικό του δωματίου πάνω στον καμβά, ώστε να μπορεί ο χρήστης να ξεχωρίσει πιο είναι το δωμάτιο ένα όταν εισάγει κάποιο έκθεμα

    3. startX, πρόκειται για συντεταγμένη στον άξονα Χ και αποθηκεύει τη θέση του ποντικού στον άξονα Χ όταν ο χρήστης πατήσει το αριστερό κουμπί του ποντικιού του

    4. startY, πρόκειται για συντεταγμένη στον άξονα Υ και αποθηκεύει τη θέση του ποντικού στον άξονα Υ όταν ο χρήστης πατήσει το αριστερό κουμπί του ποντικιού του

    5. endX, πρόκειται για συντεταγμένη στον άξονα Χ και αποθηκεύει τη θέση του ποντικού στον άξονα Χ όταν ο χρήστης αφήσει το αριστερό κουμπί του ποντικιού του

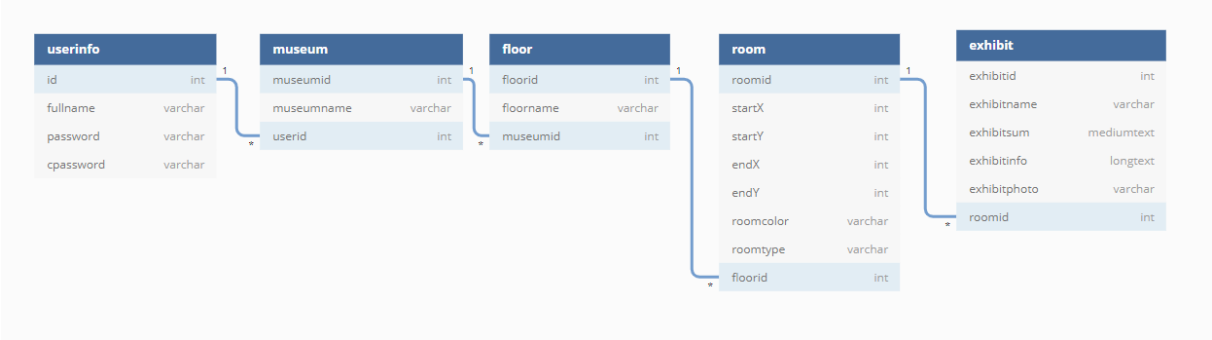
    6. endY, πρόκειται για συντεταγμένη στον άξονα Υ και αποθηκεύει τη θέση του ποντικού στον άξονα Υ όταν ο χρήστης αφήσει το αριστερό κουμπί του ποντικιού του

    7. floorid, που πρόκειται για αναφορά στον πίνακα floor ώστε να υπάρχει σχέση μεταξύ ορόφου και δωματίου

- exhibit: Ο πίνακας αυτός αποθηκεύει δεδομένα σχετικά με τα εκθέματα που υπάρχουν σε ένα δωμάτιο. Τα πεδία που τον αποτελούν είναι τα εξής:

    1. exhibitid, που είναι το αναγνωριστικό του εκθέματος μέσα στον πίνακα της βάσης δεδομένων

2. exhibitname, που αποθηκεύει ως δεδομένα το όνομα ενός εκθέματος
3. exhibitsum, που αποθηκεύει ως δεδομένα μια περίληψη σχετική με τις πληροφορίες ενός εκθέματος
4. exhibitinfo, που αποθηκεύει ως δεδομένα αρκετές πληροφορίες σχετικά με ένα έκθεμα
5. exhibitphoto, που χρησιμοποιείται ως αναγνωριστικό του εκθέματος σε σχέση με τον υποκατάλογο uploads ώστε να εμφανίζεται η σωστή φωτογραφία όταν κάποιος επιλέγει να δει τις πληροφορίες ενός εκθέματος
6. roomid, που πρόκειται για αναφορά στον πίνακα room ώστε να υπάρχει σχέση μεταξύ ορόφου και δωματίου



Εικόνα 2: Δομή βάσης δεδομένων

3.4.2 Σχέσεις μεταξύ πινάκων βάσης δεδομένων

Οι σχέσης μεταξύ των πινάκων πρόκειται για 1 προς πολλά. Πιο αναλυτικά:

- Η σχέση 1 προς πολλά μεταξύ των πινάκων userinfo και museum σημαίνει πως ένας χρήστης μπορεί να έχει πολλά μουσεία αλλά ένα μουσείο ανήκει μόνο σε ένα χρήστη

- Η σχέση 1 προς πολλά μεταξύ των πινάκων museum και floor σημαίνει πως ένα μουσείο μπορεί να έχει πολλούς ορόφους αλλά ένας όροφος μπορεί να ανήκει μόνο σε ένα μουσείο

- Η σχέση 1 προς πολλά μεταξύ των πινάκων floor και Room σημαίνει πως ένας όροφος μπορεί να έχει πολλά δωμάτια αλλά κάθε δωμάτιο ανήκει μόνο σε έναν όροφο

- Η σχέση 1 προς πολλά μεταξύ των πινάκων room και exhibit σημαίνει πως ένα δωμάτιο μπορεί να έχει πολλά εκθέματα άλλα κάθε έκθεμα ανήκει σε ένα μόνο δωμάτιο

Επίσης μεταξύ ξένων κλειδιών και πινάκων-γονέων έχουν το χαρακτηριστικό CASCADE DELETE που σημαίνει πως αν για παράδειγμα διαγραφτεί ένας όροφος, τότε χάνονται και ως δεδομένα και τα δωμάτια και τα εκθέματα που υπάγονται κάτω από αυτόν από τη βάση δεδομένων.

Παρακάτω φαίνεται ο κώδικας της MySQL που χρησιμοποιείται για να προστεθεί αυτό το χαρακτηριστικό. Δημιουργούμε μια αλυσίδα από CASCADE χαρακτηριστικά ώστε διαγράφοντας το ένα στοιχείο τις αλυσίδας, τότε διαγράφονται και όλα που ανήκουν σε αυτό.

```
ALTER TABLE `museum`
  ADD CONSTRAINT `museum_ibfk_1` FOREIGN KEY (`userid`) REFERENCES
`userinfo` (`userid`) ON DELETE CASCADE ;
ALTER TABLE `floor`
  ADD CONSTRAINT `floor_ibfk_1` FOREIGN KEY (`museumid`) REFERENCES
`museum` (`museumid`) ON DELETE CASCADE;
ALTER TABLE `room`
  ADD CONSTRAINT `room_ibfk_1` FOREIGN KEY (`floorid`) REFERENCES `floor`
(`floorid`) ON DELETE CASCADE;
ALTER TABLE `exhibit`
  ADD CONSTRAINT `exhibit_ibfk_1` FOREIGN KEY (`roomid`) REFERENCES `room`
(`roomid`) ON DELETE CASCADE;
```

3.5 Ιεραρχία καταλόγων διακομιστή

Όλα τα αρχεία που σχετίζονται με την διαδικτυακή εφαρμογή, την επικοινωνία αυτής με τη βάση δεδομένων καθώς και την επικοινωνία της εφαρμογής για κινητά τηλέφωνα με τη βάση δεδομένων βρίσκονται στον κατάλογο /public_html .

- Ο υποκατάλογος css περιέχει τον κώδικα css που χρησιμοποιείται για τη διαδικτυακή εφαρμογή.
- Ο υποκατάλογος dbconfig περιέχει το αρχείο σε γλώσσα php που χρησιμοποιείται για την σύνδεση ιστοσελίδας και βάσης δεδομένων.
- Ο υποκατάλογος img περιέχει της εικόνες με καταλήξεις .png και .ico που χρησιμοποιούνται μέσα στην ιστοσελίδα.
- Ο υποκατάλογος js περιέχει όλα τα αρχεία που δημιουργήθηκαν σε γλώσσα JavaScript και κάνουν χρήση της βιβλιοθήκης jQuery που χρησιμοποιούνται για να στείλουν δεδομένα προς τη βάση δεδομένων κάνοντας χρήση της Ajax επικοινωνίας αλλά και για τη δημιουργία διαφόρων εφέ, όπως την εμφάνιση μιας φόρμας modal με το πάτημα ενός κουμπιού.
- Ο υποκατάλογος painterscripts περιέχει όλα τα αρχεία της βιβλιοθήκης jCanvas για την υλοποίηση της σελίδας στην οποία δημιουργούμε τον χάρτη ενός ορόφου
- Ο υποκατάλογος PHPFiles περιέχει όλα τα php αρχεία που χρησιμοποιούνται για την επικοινωνία της διαδικτυακής ιστοσελίδας με τη βάση δεδομένων.
- Ο υποκατάλογος androidPHPFiles περιέχει όλα τα php αρχεία που χρησιμοποιούνται για την επικοινωνία της εφαρμογής για κινητά τηλέφωνα με τη βάση δεδομένων.
- Ο υποκατάλογος upload χρησιμοποιείται για να αποθηκεύονται οι εικόνες που έχει ανεβάσει ο διαχειριστής ενός μουσειακού χώρου κατά τη δημιουργία ή επεξεργασία ενός εκθέματος.

3.6 Χρησιμότητα Sessions

Οι συνεδρίες (sessions) επιτρέπουν την αποθήκευση μεταβλητών μέσω της PHP στην μεριά του εξυπηρετητή που χρειάζονται για τη περιήγηση του χρήστη σε μια σελίδα. Οι μεταβλητές που είναι αποθηκευμένες στις συνεδρίες παύουν να ισχύουν όταν ο χρήστης κλείσει το παράθυρο περιήγησης, ή όταν ο κώδικας διαγράψει τις μεταβλητές αυτές από μία συνεδρία. Στο πρόγραμμα μας χρησιμοποιούνται τέτοιου είδους μεταβλητές για να αποθηκεύσουμε το αναγνωριστικό του χρήστη (userid) και το αναγνωριστικό του ορόφου (floorid) που επεξεργάζεται κατά την περιήγηση του. Η μεταβλητή userid παύει να ισχύει όταν ο χρήστης αποσυνδεθεί από τη πλατφόρμα ενώ η μεταβλητή floorid παύει να ισχύει όταν ο χρήστης ολοκληρώσει την επεξεργασία ενός ορόφου.

3.7 Περιγραφή αρχείων διακομιστή

3.7.1 Περιγραφή αρχείου υποκατάλογου dbconfig

Το αρχείο config.php είναι το πιο βασικό αρχείο ώστε να μπορεί οποιοδήποτε αρχείο σε γλώσσα php να επικοινωνήσει με τη βάση δεδομένων. Οι μεταβλητές που περιέχει είναι τα διαπιστευτήρια που χρησιμοποιούνται για την επικοινωνία. Το συγκεκριμένο αρχείο καλείται από όλα τα αρχεία php χρησιμοποιώντας την εντολή require_once('../dbconfig/config.php');

```
<?php

$con=mysqli_connect ("localhost", "u940333051_edaz", "15sb5t") or die ('I
cannot connect to the database because: ' . mysql_error());
mysqli_select_db ($con,'u940333051_edaz');

?>
```

3.7.2 Περιγραφή αρχείων υποκατάλογου PHPFiles

Μέσα στον συγκεκριμένο κατάλογο βρίσκονται αρχεία που λειτουργούν ως συναρτήσεις του συστήματος από τη μεριά της διαδικτυακής εφαρμογής και χρησιμοποιούνται για τη μεταφορά δεδομένων από και προς τη βάση.

Οι οντότητες μουσείο(museum), όροφος(floor), δωμάτιο(room), έκθεμα(exhibit) έχουν από 2 αρχεία:

- ένα αρχείο addnew που χρησιμοποιείται για τη δημιουργία μιας οντότητας, λαμβάνοντας δεδομένα από μια φόρμα, πχ το όνομα, και από το SESSION το userid του χρήστη και τα εισάγει στον αντίστοιχο πίνακα. Αν ο χρήστης πατήσει το κουμπί επεξεργασίας μιας οντότητας, τότε χρησιμοποιείται το ίδιο αρχείο για να ενημερώσει τα δεδομένα στον πίνακα της συγκεκριμένης οντότητας.
- ένα αρχείο delete που χρησιμοποιείται για τη διαγραφή μιας οντότητας από τη βάση δεδομένων

- Τα αρχεία που υπάγονται στη κατηγορία addnew είναι τα εξής:
    1. addnewmuseum.php
    2. addnewfloor.php
    3. addnewroom.php
    4. addnewexhibit.php

και ο κώδικας είναι της μορφής που φαίνεται παρακάτω:

```
<?php
ob_start();
session_start();
require_once('../dbconfig/config.php');

$museumid=$_POST['museumid'];
$museumname= $_POST["museumname"];
$userid= $_SESSION['userid'];

$sql = "INSERT INTO museum(museumid,museumname,userid) VALUES
('$museumid','$museumname','$userid')ON DUPLICATE KEY UPDATE
museumid='$museumid',museumname='$museumname',userid='$userid'";

if ($con->query($sql) === TRUE) {
    header("Location: ../homepage.php");
} else {
```

```
    echo "Error: " . $sql . "<br>" . $con->error;
}

ob_end_flush();
?>
```

Παρόλο που όλα τα αρχεία είναι ίδια κάνοντας χρήση της INSERT συνάρτησης της SQL, το αρχείο addnewexhibit.php είναι λίγο διαφορετικό καθώς πρέπει να εισάγει και την εικόνα που εισάγει ο χρήστης μέσα στον φάκελο uploads όπως φαίνεται και παρακάτω:

```
<?php
      require_once('../dbconfig/config.php');
      if($_FILES['uploaded_file']['name']!='' && $_POST['photo']<>
$_FILES['uploaded_file']['name']  )
  {
    $path = "../uploads/";
    $path = $path . basename( $_FILES['uploaded_file']['name']);
    $photo=basename( $_FILES['uploaded_file']['name']);
     if(move_uploaded_file($_FILES['uploaded_file']['tmp_name'], $path)) {
     }
  }
      else if (empty($_FILES['uploaded_file']['name']))
      {
          $photo= $_POST['photo'];
      }

if($_POST['exhibitid']===0){
    $exhibitid='exhibitid';}
    else{
        $exhibitid=$_POST['exhibitid'];
    }
        $exhibitname= $_POST["exhibitname"];
        $exhibitsum=$_POST["exhibitsum"];
        $exhibitinfo=$_POST["exhibitinfo"];
        $exhibitphoto=$photo;
        $roomid=$_POST["roomid"];
```

```
    $sql = "INSERT INTO exhibit VALUES
('$exhibitid','$exhibitname','$exhibitsum','$exhibitinfo','$exhibitphoto','
$roomid')ON DUPLICATE KEY UPDATE
exhibitname='$exhibitname',exhibitsum='$exhibitsum',
exhibitinfo='$exhibitinfo', exhibitphoto='$exhibitphoto'";

        if ($con->query($sql) === TRUE) {
        header("Location: ../painterpoints1.php");
        } else {
            echo "Error: " . $sql . "<br>" . $con->error;
}
?>
```

Στην ουσία η λειτουργία του συγκεκριμένου αρχείου είναι να ελέγχει αρχικά αν ο χρήστης έχει εισάγει φωτογραφία στη φόρμα, και αν ναι να την μεταφέρει στο φάκελο uploads και να ελέγχει αν ο χρήστης κάνει επεξεργασία ενός εκθέματος, ελέγχοντας το exhibitid να μην είναι ίσο με το 0, ή αν εισάγει νέο έκθεμα, δηλαδή exhibitid ίσο με το 0. Στην περίπτωση της επεξεργασίας, αν ο χρήστης δεν θέλει να αλλάξει φωτογραφία, τότε ο πρώτος έλεγχος, απλά μεταφέρει το όνομα της φωτογραφίας πάλι μέσα στη βάση χωρίς να την αλλάζει.

- Τα αρχεία που υπάγονται στη κατηγορία delete είναι τα εξής:

    1. deletemuseum.php
    2. deletefloor.php
    3. deleteroom.php
    4. deleteexhibit.php

και ο κώδικας είναι της μορφής που φαίνεται παρακάτω:

```
<?php
```

```
 ob_start();
 session_start();
      require_once('../dbconfig/config.php');


    $id = $_GET["id"];
    $sql = "DELETE FROM museum WHERE museumid='". $id."' ";
       header("Location: ../homepage.php");
       $result = mysqli_query($con, $sql);
ob_end_flush();
?>
```

?>

- Το αρχείο floortohomepage.php καλείται απο τη σελίδα homepage.php με χρήση της επικοινωνίας Ajax με σκοπό να επιστρέψει τα δεδομένα σχετικά με τους ορόφους ενός μουσείου. Δέχεται ως μεταβλητές μόνο το userid από το SESSION καθώς για να υπάρξει όροφος θα πρέπει να έχει δημιουργηθεί ένα αντίστοιχο μουσείο.

- Το αρχείο homepagetopainter.php χρησιμοποιείται για να ανακατευθύνει το χρήστη στη σελίδα painter.php λαμβάνοντας ως μεταβλητή το id του ορόφου που θέλει να επεξεργαστεί και θέτοντας το στο session ώστε να χρησιμοποιηθεί για την περιήγηση του μεταξύ των σελίδων painter.php και painterpoints.

- Το αρχείο painterpointstohomepage.php χρησιμοποιείται για να ανακατευθύνει το χρήστη στη σελίδα homepage.php όταν έχει τελειώσει με την επεξεργασία του ορόφου και κάνει unset τη μεταβλητή id του ορόφου από το SESSION.

- Το αρχείο logoutbtn.php χρησιμοποιείται όταν ο χρήστης θέλει να αποσυνδεθεί απο τη πλατφόρμα σβήνοντας έτσι ότι δεδομένα περιέχει το SESSION με χρήση της συνάρτησης session_destroy()

```
<?php
      session_start();
      session_destroy();
      header("Location: ../index.php");
      exit;
?>
```

3.7.3 Περιγραφή αρχείων υποκαταλόγου js

Μέσα στον συγκεκριμένο υποκατάλογο βρίσκονται 3 αρχεία της γλώσσας JavaScript που αντιστοιχούν στις 3 σελίδες της πλατφόρμας μας.

- Το αρχείο homepage.js περιέχει συναρτήσεις που σχετίζονται με τις ενέργειες που μπορούν να γίνουν στη σελίδα homepage.php όπως για παράδειγμα όταν ο χρήστης επιλέγει το όνομα ενός μουσείου εμφανίζεται από κάτω η λίστα με τους ορόφους που έχει δημιουργήσει για το συγκεκριμένο μουσείο ή όταν ο χρήστης θέλει να δημιουργήσει ένα μουσείο ή εναν όροφο, εμφανίζεται το modal για να συμπληρώσει το όνομα του ορόφου. Επίσης περιέχει τις συναρτήσεις που χρησιμοποιούνται για την επικοινωνία Ajax με τη βάση δεδομένων ανάλογα με τις ενέργειες που θέλει να εκτελέσει ο χρήστης πάνω στις οντότητες μουσείο ή όροφος.

- Το αρχείο painter.js περιέχει 3 συναρτήσεις σχετικές με τη σελίδα painter.php:

  1. initCanvas() που χρησιμοποιείται για να λάβει τις κινήσεις του χρήστη από τη στιγμή που πατάει το δεξί κουμπί του ποντικιού του για να σχεδιάσει ένα δωμάτιο μέχρι τη στιγμή που θα το αφήσει ώστε να αποθηκεύσει στη βάση δεδομένων τις συντεταγμένες του δωματίου

  2. addrooms() που λαμβάνει ως μεταβλητές τις συντεταγμένες του δωματίου που σχεδίασε ο χρήστης ώστε να κάνει την εισαγωγή των δεδομένων μέσω της επικοινωνίας Ajax στη βάση.

  3. databasetopainter() που χρησιμοποιείται για να σχεδιάσει πάνω στο στοιχείο canvas της HTML τα ήδη υπάρχον δωμάτια που έχει σχεδιάσει ο χρήστης.

- Το αρχείο points.js περιέχει συναρτήσεις σχετικές με τη σελίδα painterpoints.php. Πιο συγκεκριμένα περιέχει συναρτήσεις που χρησιμοποιούνται για τις ενέργειες δημιουργία, επεξεργασία και διαγραφή των δωματίων ή των εκθεμάτων αλλά και συναρτήσεις που χρησιμοποιούνται για να δημιουργήσουν τον χάρτη των δωματίων στο στοιχείο canvas της HTML που βρίσκεται στο αριστερό μέρος της σελίδας αλλά και στη λίστα των δωματίων που βρίσκεται στο δεξί μέρος της σελίδας καθώς και τα εκθέματα από κάθε στοιχείο δωματίου μέσα στη λίστα.

3.7.4 Περιγραφή του υποκαταλόγου painterscripts

Ο συγκεκριμένος υποκατάλογος περιέχει αρχεία της γλώσσας Javascript κάνοντας χρήση της βιβλιοθήκης jQuery και χρησιμοποιούνται για τη χρήση της βιβλιοθήκης jcanvas στη σελίδα painter.php

3.7.5 Περιγραφή του υποκατάλογου androidPHPFiles

Ο υποκατάλογος androidPHPFiles περιέχει αρχεία παρόμοια με τον υποκατάλογο PHPFiles τα οποία χρησιμοποιούνται για την επικοινωνία μεταξύ της εφαρμογής για κινητά τηλέφωνα και της βάσης δεδομένων.

Η κύρια διαφορά των αρχείων για την ιστοσελίδα με τα αρχεία για την εφαρμογή για τα κινητά τηλέφωνα είναι ο τρόπος που ψάχνει η κάθε εφαρμογή τα δεδομένα. Η εφαρμογή για κινητά τηλέφωνα λαμβάνει όλα τα δεδομένα για ένα μουσείο μαζικά και τα αποθηκεύει σε πίνακες τοπικά στη συσκευή, ενώ η διαδικτυακή εφαρμογή ζητάει τα δεδομένα από τη βάση πιο συγκεκριμένα, πχ για ένα συγκεκριμένο όροφο ή για ένα συγκεκριμένο δωμάτιο. Παρ' όλα αυτά για λόγους ευχρηστίας, τα αρχεία για την εφαρμογή κινητού τηλεφώνου έχουν χωριστεί με βάση τα δεδομένα που επιστρέφουν για να είναι πιο εύκολη η συντήρηση του κώδικα τους.

Μέσα στον συγκεκριμένο υποκατάλογο βρίσκονται και δύο αρχεία που δεν αντιστοιχούν σε κάποιο αρχείο του υποκατάλογου PHPFiles :

- appadminlogin.php , αρχείο που χρησιμοποιείται για να μπορεί να συνδεθεί ο διαχειριστής ενός μουσείου από οποιαδήποτε κινητό τηλέφωνο. Λαμβάνει ως μεταβλητές το username και το password που έχει εισάγει ο χρήστης καθώς και το museumid, το αναγνωριστικό του μουσείου στον αντίστοιχο πίνακα της βάσης δεδομένων

```
<?php
  session_start();
  require_once('../dbconfig/config.php');
  $success=false;
  $username=$_POST['username'];
  $password=$_POST['password'];
  $museumid=$_POST['museumid'];

  $query = "select * from userinfo JOIN museum ON
museum.userid=userinfo.userid where userinfo.username='$username' and
userinfo.password='$password' AND museum.museumid='$museumid'";

  $query_run = mysqli_query($con,$query);
  $result=mysqli_query($con,$query);

  if (mysqli_num_rows($result) > 0) {
    $success=true;
  }
  else{
    $success=false;
  }
  $ar=array('success'=>$success);
  $c= json_encode($ar);
  echo $c;
?>
```

- androidexhibitphototosever.php, αρχείο που χρησιμοποιείται για να αποθηκεύει στον κατάλληλο υποκατάλογο τη φωτογραφία που ανεβάσει ο χρήστης όταν επεξεργάζεται ένα έκθεμα Δεν υπάρχει μεγάλη διαφορά από τη μέθοδο που χρησιμοποιείται η διαδικασία αποθήκευσης φωτογραφιών στη διαδικτυακή εφαρμογή από μερίας PHP, όσο από μεριάς Java που είναι και η γλώσσα που χρησιμοποιήθηκε στην εφαρμογή για κινητά

```
<?php
session_start();
require_once('../dbconfig/config.php');
if (is_uploaded_file($_FILES['photo']['tmp_name'])) {
  $uploads_dir = '../uploads/';
  $tmp_name = $_FILES['photo']['tmp_name'];
  $pic_name = $_FILES['photo']['name'];
  move_uploaded_file($tmp_name, $uploads_dir.$pic_name);
}
else{
 echo "File not uploaded successfully.";
}
?>
```

3.8 Περιγραφή αρχείων εφαρμογής για κινητά τηλέφωνα

Τα αρχεία που αποτελούν την εφαρμογή για κινητά τηλέφωνα δημιουργήθηκαν μέσω του Android Studio χρησιμοποιώντας τη γλώσσα προγραμματισμού Java για το back-end κομμάτι και αρχεία τύπου XML για το front-end κομμάτι. Το Android Studio χωρίζει τον κώδικα της εφαρμογής σε τρεις βασικούς υποκατάλογους:

- manifests
- java
- res

3.8.1 Υποκατάλογος manifests

Μέσα στο συγκεκριμένο υποκατάλογο βρίσκεται ένα αρχείο που δημιουργείται αυτόματα απο το Android Studio για κάθε νέα εφαρμογή που θέλει να φτιάξει ένας χρήστης και ονομάζεται AndroidManifest.xml. Παρόλο που τα αρχεία τύπου XML χρησιμοποιούνται για το εμφανισιακό κομμάτι μιας εφαρμογής, το Android Studio τα χρησιμοποιεί για να ορίσει και τις ρυθμίσεις της εφαρμογής από την πλευρά του προγραμματιστή. Πιο συγκεκριμένα στο συγκεκριμένο αρχείο δηλώνονται

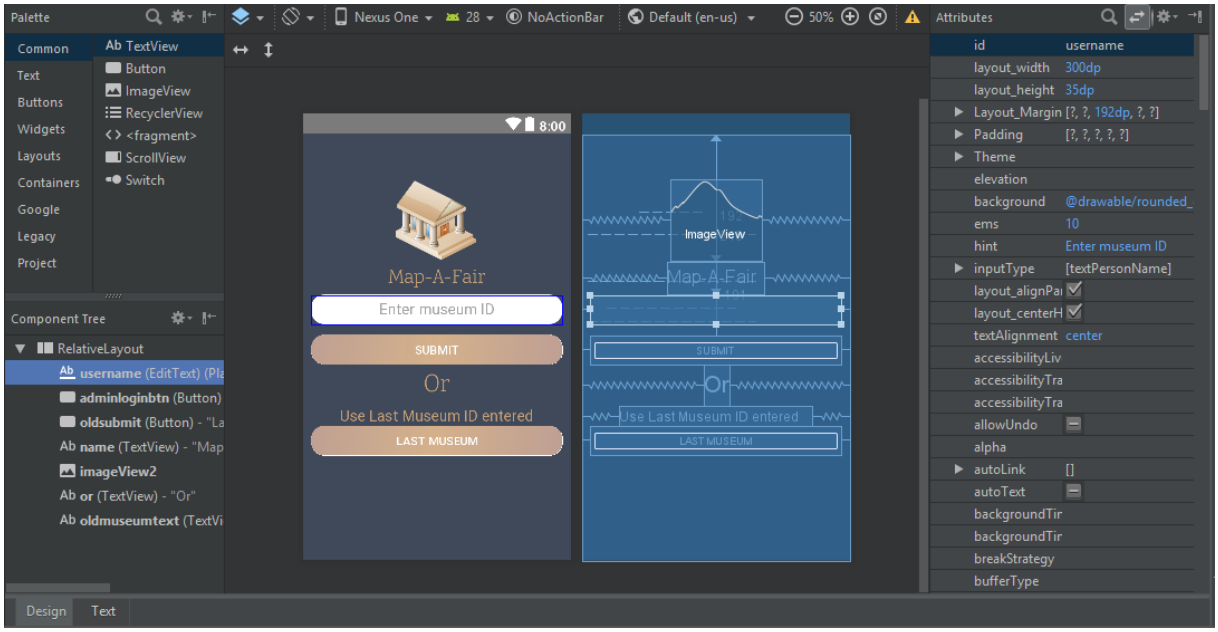
- οι άδειες που χρειάζεται η εφαρμογή για να μπορεί να έχει πρόσβαση σε αυτή ο χρήστης, όπως για παράδειγμα να χρησιμοποιεί την κάμερα ή σύνδεση στο διαδίκτυο
- το αρχείο το οποίο θα χρησιμοποιηθεί κατά την είσοδο του χρήστη στο περιβάλλον της εφαρμογής
- το θέμα που θα χρησιμοποιήσει η εφαρμογή ως προς την εμφάνιση
- τη γραμματοσειρά που θα χρησιμοποιηθεί μέσα στην εφαρμογή
- ορίζεται το εικονίδιο που θα έχει η εφαρμογή όταν ο χρήστης βρίσκεται στο μενού του κινητού του τηλεφώνου

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="mapafair.mapafair">
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.CAMERA"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <application
        android:allowBackup="true"
        android:icon="@drawable/androidavatar"
        android:label="@string/app_name"
        android:largeHeap="true"
        android:hardwareAccelerated="false"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.AppCompat.Light.NoActionBar">
        <activity android:name=".LoginActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".MuseumActivity"></activity>
        <meta-data
            android:name="preloaded_fonts"
            android:resource="@array/preloaded_fonts" />
    </application>
</manifest>
```

3.8.2 Υποκατάλογος res

Μέσα στον συγκεκριμένο υποκατάλογο τοποθετούνται τα αρχεία XML που θα χρησιμοποιήθουν για το front-end κομμάτι της εφαρμογής, αρχεία τύπου .jpeg , .png και .ico που χρησιμοποιούνται σε συνδυασμό με τα αρχεία XML και αρχεία τύπου .ttf που περιγράφουν τη γραμματοσειρά και χρησιμοποιούνται και αυτά σε συνδυασμό με τα αρχεία xml.

Για το front-end κομμάτι μιας εφαρμογής, το Android Studio επιτρέπει τη δημιουργία του και με κώδικα XML αλλά και με ένα σύστημα (Palette) στο οποίο ο χρήστης μπορεί να επιλέξει και να σύρει ένα στοιχείο πάνω στην οθόνη και να το επεξεργαστεί με το διαχειριστικό που δίνεται, παράγοντας κώδικα XML



Εικόνα 3: Πρόγραμμα Android Studio στην παραμετροποίηση του XML αρχείου μέσω του στοιχείου Palette

3.8.3 Υποκατάλογος java

Μέσα σε αυτόν τον υποκατάλογο βρίσκονται όλα τα αρχεία που χρησιμοποιούν τη γλώσσα προγραμματισμού Java. Τα αρχεία νοητά χωρίζονται σε 3 κατηγορίες:

- αρχεία που χρησιμοποιούνται για την επικοινωνία με τη βάση δεδομένων. Όταν καλούνται, αποστέλλουν δεδομένα σε ένα συγκεκριμένο αρχείο του διακομιστή μέσα στον υποκατάλογο androidPHPFiles. Τα αρχεία αυτής της χρήσης είναι τα Adminrequest.java, Exhibittodatabase.java, Maprequest.java.

Παρακάτω είναι ο κώδικας που καλεί το αρχείο Adminrequest.java, το οποίο χρησιμοποιείται για να συνδεθεί ένας διαχειριστής με το λογαριασμό του και να του δοθεί η δυνατότητα να επεξεργαστεί το χάρτη του μουσείου του.

```
AdminRequest adminRequest = new AdminRequest(ausername, apassword, museumid,
responseListener);
RequestQueue queue = Volley.newRequestQueue(MuseumActivity.this);
queue.add(adminRequest);
```

Παρακάτω φαίνεται ο κώδικας που χρησιμοποιείται για να καλέσει το αρχείο της PHP στη μεριά του διακομιστή.

```
package mapafair.mapafair;

import com.android.volley.Response;
import com.android.volley.toolbox.StringRequest;
import java.util.HashMap;
import java.util.Map;

public class AdminRequest extends StringRequest {
    private static final String LOGIN_REQUEST_URL =
"http://mapafair.hostingerapp.com/androidPHPFiles/appadminlogin.php";
    private Map<String, String> params;
    public AdminRequest(String username, String password, int museumid,
Response.Listener<String> listener) {
        super(Method.POST, LOGIN_REQUEST_URL, listener, null);
        params = new HashMap<>();
        params.put("username", username);
        params.put("password", password);
        params.put("museumid", String.valueOf(museumid));
    }
    @Override
    public Map<String, String> getParams() {
        return params;
```

```
    }
}
```

- Αρχεία που χρησιμοποιούνται για να εκτελέσουν κάποια συγκεκριμένη εργασία λειτουργώντας καθαρά ως κλάσεις. Τα αρχεία αυτής της χρήσης είναι τα CustomView.java που χρησιμοποιείται για να δημιουργήσει τα δωμάτια μέσα στο αντικείμενο τύπου canvas στην εφαρμογή και το uploadFile.java που χρησιμοποιείται για να σταλθεί η εικόνα που εισάγει ο χρήστης στη βάση δεδομένων. Το αρχείο uploadFile.java υπάγεται και στην προηγούμενη κατηγορία καθώς χρησιμοποιείται και για επικοινωνία με τον διακομιστή αφού θα πρέπει να τοποθετήσει το αρχείο στον υποκατάλογο 'uploads' κάνοντας χρήση του αρχείου androidexhibitphototoserver.php.

Παρακάτω είναι ο κώδικας που χρησιμοποιείται από τη κλάση CustomView.java για να ‘ζωγραφίσει’ τα δωμάτια. Δέχεται ως παραμέτρους τις συντεταγμένες ενός δωματίου, το χρώμα και τον τύπο του και το σχεδιάζει πάνω στο στοιχείο καμβά του Android Studio.

```
@Override
protected void onDraw(Canvas canvas) {
    SharedPreferences pref = getContext().getSharedPreferences("floorid",
MODE_PRIVATE);
    floorid = pref.getString("floorid", null);
    canvas.save();
    canvas.scale(scaleFactor, scaleFactor);
    canvas.drawColor(Color.rgb(205, 205, 205));
    canvas.scale(1f, 1f);
    for (i = 0; i < length; i++) {
        if (coords[i][7].equals(floorid)) {
            startX = Integer.parseInt(coords[i][1]) + extraX;
            startY = Integer.parseInt(coords[i][2]) + extraY;
            endX = Integer.parseInt(coords[i][3]) + extraX;
            endY = Integer.parseInt(coords[i][4]) + extraY;
            roomtype = coords[i][5];
            String rgb = coords[i][6];
            String newrgb = rgb.replaceAll("[rgb(),]", "");
            String[] color = newrgb.split(" ");
            paint.setColor(Color.rgb(Integer.parseInt(color[0]),
Integer.parseInt(color[1]), Integer.parseInt(color[2])));
```

```
            if (roomtype.equals("Rect")) {
                rectangle = new Rect(startX, startY, endX, endY);
                canvas.drawRect(rectangle, paint);
                canvas.drawRect(rectangle, paint2);
            } else {
                oval = new RectF(startX, startY, endX, endY);
                canvas.drawOval(oval, paint);
                canvas.drawOval(oval, paint2);
            }
        }
    }
    canvas.restore();
}
```

- Αρχεία που χρησιμοποιούνται για να δημιουργηθεί η όψη της εφαρμογής. Πιο συγκεκριμένα είναι αυτά που καλούν τα XML αρχεία και τα ενώνουν με τις συναρτήσεις της java που θέλουμε ώστε να δείξουμε στον χρήστη τα δεδομένα με τη μορφή που θέλουμε. Τα αρχεία αυτά είναι της κατηγορίας Activity και στον κώδικά μας υπάρχουν δυο, το LoginActivity που είναι και η πρώτη όψη που εμφανίζεται σε ένα χρήστη και χρησιμοποιείται για να δώσει την επιλογή στον χρήστη να ανοίξει τον χάρτη ενός μουσείου, εισάγοντας το αναγνωριστικό του μουσείου και το Museum Activity που χρησιμοποιείται για να εμφανίσει σε ένα χρήστη το χάρτη του μουσείου χωρίζοντας το σε ορόφους. Η μεταφορά από ένα Activity σε ένα άλλο γίνεται με χρήση της κλάσης Intent της java, που χρησιμοποιείται και για τη μεταφορά δεδομένων.

Παρακάτω είναι ο κώδικας που χρησιμοποιείται για να περάσει δεδομένα από το αρχείο LoginActivity.java, που χρησιμοποιείται για να εισάγει ο χρήστης το αναγνωριστικό ενός μουσείου στο αρχείο MuseumActivity.java. Αυτή η διαδικασία ονομάζεται Intent.

```
Intent intent = new Intent(LoginActivity.this, MuseumActivity.class);
intent.putExtra("museumid", museumid);
intent.putExtra("museumname", museumname);
LoginActivity.this.startActivity(intent);
```

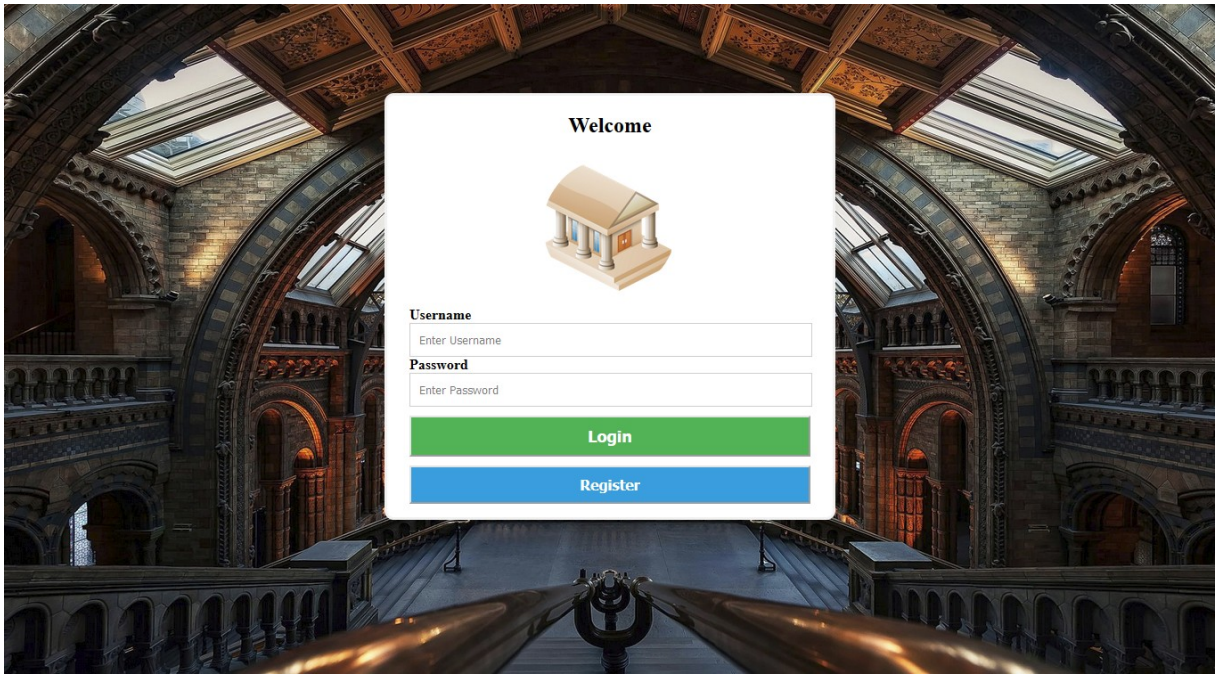
Παρακάτω είναι ο κώδικας που χρησιμοποιείται από το αρχείο MuseumActivity.java, για να λάβει τα δεδομένα που έστειλε το LoginActivity.java.

```
Intent intent = getIntent();
String museumname = intent.getStringExtra("museumname");
museumid = intent.getIntExtra("museumid", 0);
```

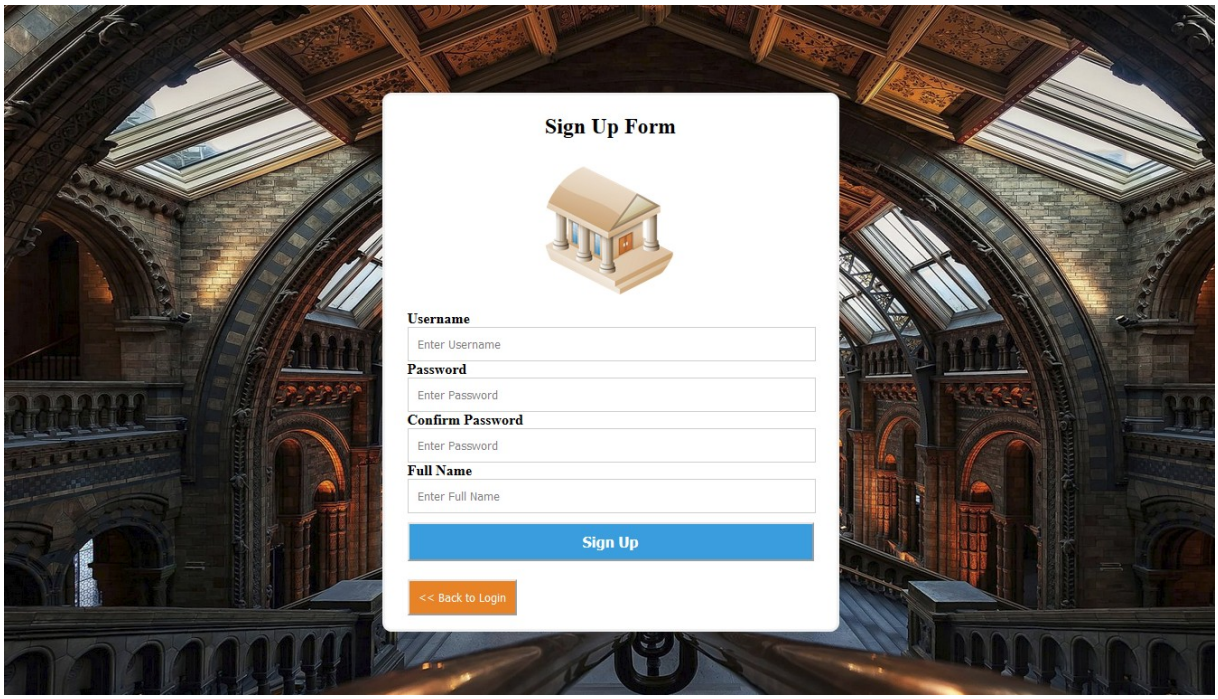
ΚΕΦΑΛΑΙΟ 4: ΠΑΡΟΥΣΙΑΣΗ ΕΦΑΡΜΟΓΩΝ

4.1 Παρουσίαση διαδικτυακής εφαρμογής

Η πρόσβαση στην εφαρμογή γίνεται μέσω του συνδέσμου mapafair.hostingerapp.com. Κατά την είσοδο του χρήστη στην ιστοσελίδα, του ζητείται να συμπληρώσει το όνομα χρήστη και τον κωδικό πρόσβασης για να έχει πρόσβαση στις λειτουργίες της εφαρμογής. Σε περίπτωση που δεν έχει λογαριασμό ο χρήστης, τότε μπορεί να δημιουργήσει έναν νέο, εισάγοντας στη φόρμα το ονοματεπώνυμο του, το όνομα χρήστη και τον κωδικό του. Η διαδικτυακή εφαρμογή είναι μόνο για διαχειριστές ενός μουσείου και δεν είναι για πρόσβαση σε απλούς χρήστες-επισκέπτες.

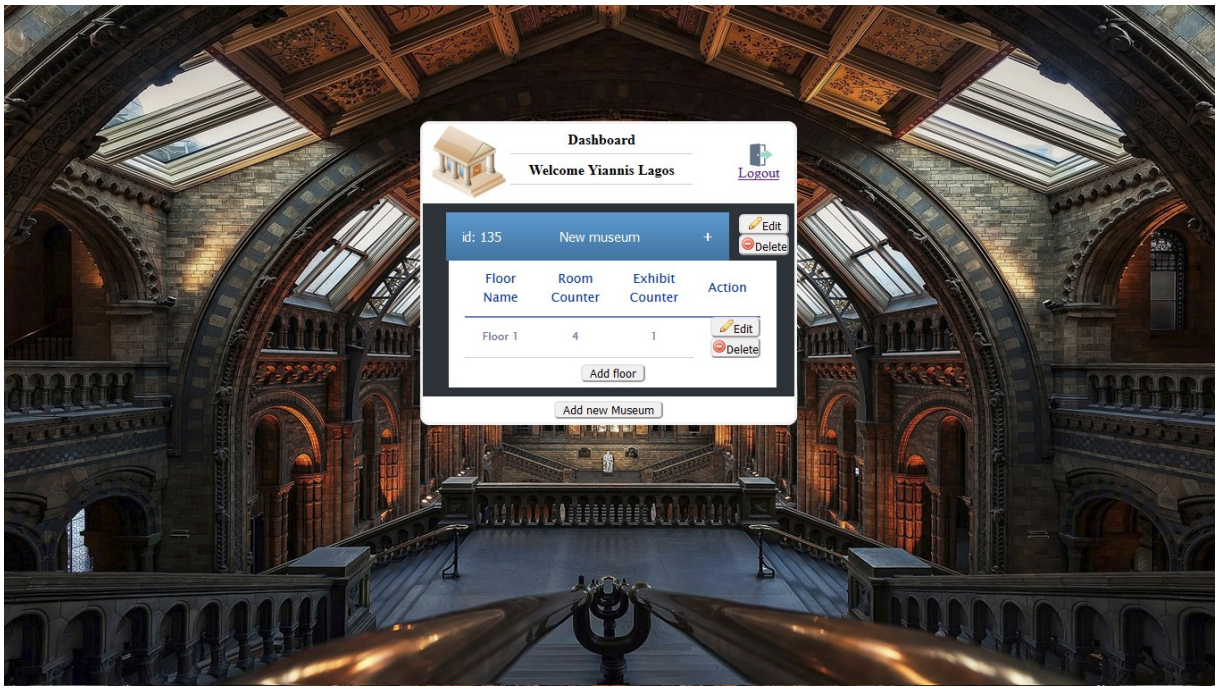


Εικόνα 4: Παράθυρο σύνδεσης χρήστη-διαχειριστή στη διαδικτυακή εφαρμογή



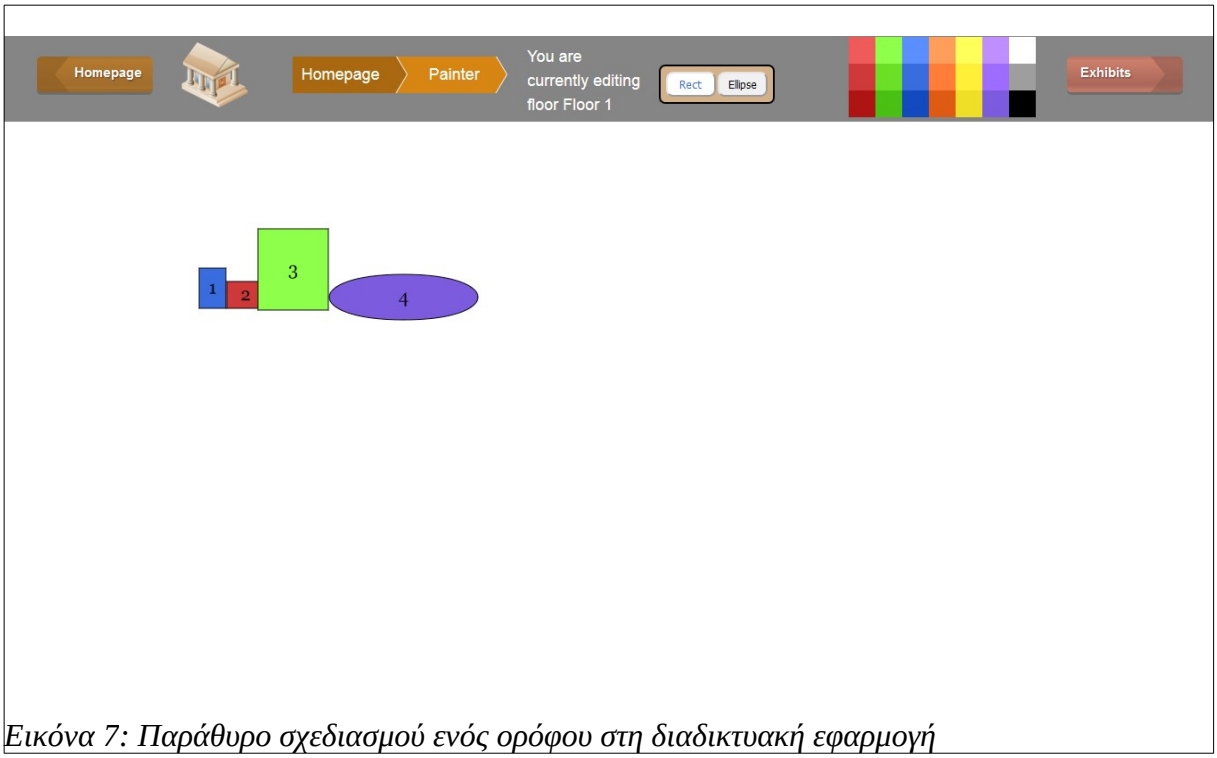
Εικόνα 5: Παράθυρο εγγραφής χρήστη-διαχειριστή στη διαδικτυακή εφαρμογή

Όταν ο χρήστης κάνει επιτυχή είσοδο με τον λογαριασμό του, μπορεί να δει τον πίνακα διαχείρισης, μέσα στον οποίο αρχικά φαίνονται τα μουσεία που διαχειρίζεται μαζί με το αναγνωριστικό τους, που χρησιμοποιούν οι χρήστες για να εμφανίσουν τον χάρτη στο κινητό τους τηλέφωνο, και το κουμπί εξόδου . Μπορεί να προσθέσει ένα μουσείο, να επεξεργαστεί το όνομα του ή να το διαγράψει. Πατώντας πάνω σε ένα μουσείο, ανοίγει ο πίνακας με τους ορόφους που διαθέτει, δίνοντας και κάποιες πληροφορίες σχετικά με το πόσα δωμάτια έχει ο όροφος και πόσα εκθέματα. Μπορεί να προσθέσει ή να αφαιρέσει έναν όροφο σε αυτό το σημείο, και πατώντας επεξεργασία, να ανακατευθυνθεί στην σελίδα στην οποία σχεδιάζεται ο όροφος.



Εικόνα 6: Παράθυρο διαχείρισης μουσείων στη διαδικτυακή εφαρμογή

Βρισκόμενος στην σελίδα painter.php, ο χρήστης βλέπει με λευκό χρώμα τον καμβά (canvas αντικείμενο της HTML) πάνω στο οποίο μπορεί να σχεδιάσει τον όροφο του μουσείου. Αν υπάρχουν ήδη δωμάτια σχεδιασμένα, εμφανίζονται και αυτά μέσα στον καμβά. Στο πάνω μέρος, ξεκινώντας από τα αριστερά προς τα δεξιά, υπάρχει το κουμπί homepage, που γυρνάει τον χρήστη στη προηγούμενη σελίδα, το εικονίδιο που χρησιμοποιείται ως logo της εφαρμογής, στη συνέχεια τα κουμπιά πλοήγησης για να καταλαβαίνει ο χρήστης σε πιο βήμα της επεξεργασίας ενός ορόφου βρίσκεται, ένα λεκτικό που τον ενημερώνει σχετικά με το ποιον όροφο επεξεργάζεται, 2 κουμπιά για εναλλαγή σχημάτων μεταξύ έλλειψης και ορθογωνίου, ένα πίνακα με διάφορα χρώματα και τέλος το κουμπί για να προχωρήσει στο επόμενο βήμα επεξεργασίας του ορόφου. Σε αυτό το σημείο, ο χρήστης μπορεί να σχεδιάσει μόνο και όχι να αφαιρέσει η να επεξεργαστεί τα δωμάτια. Αφού τελειώσει με την σχεδίαση, ο χρήστης πρέπει να πατήσει το κουμπί 'Exhibits' για να προχωρήσει στο επόμενο βήμα.



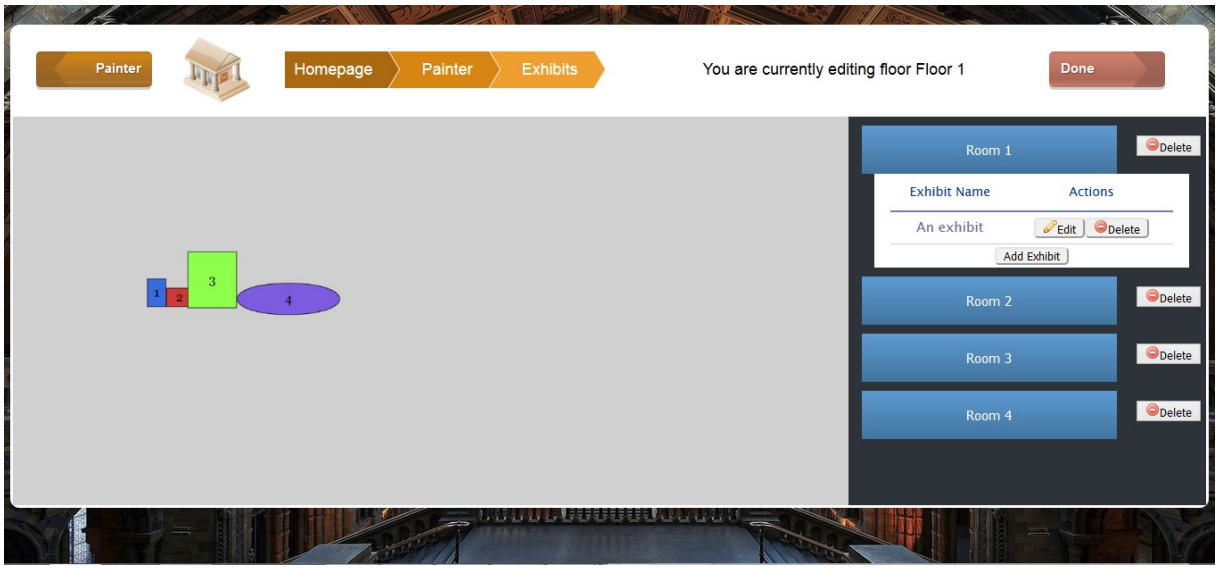
Εικόνα 7: Παράθυρο σχεδιασμού ενός ορόφου στη διαδικτυακή εφαρμογή

Στην τελευταία σελίδα της διαδικτυακής εφαρμογής, την painterpoints.php , ο χρήστης μπορεί να διαγράψει αν θέλει τα δωμάτια ή να προσθέσει,επεξεργαστεί ή να αφαιρέσει τα εκθέματα κάθε δωματίου. Η σελίδα χωρίζεται σε 3 μέρη:

- το πάνω μέρος που βρίσκονται το κουμπί για να γυρίσει ο χρήστης στο προηγούμενο βήμα,το logo της εφαρμογής, τα κουμπιά πλοήγησης, το λεκτικό που τον ενημερώσει για το όνομα του ορόφου και το κουμπί 'Done' που επαναφέρει το χρήστη στην σελίδα homepage.php

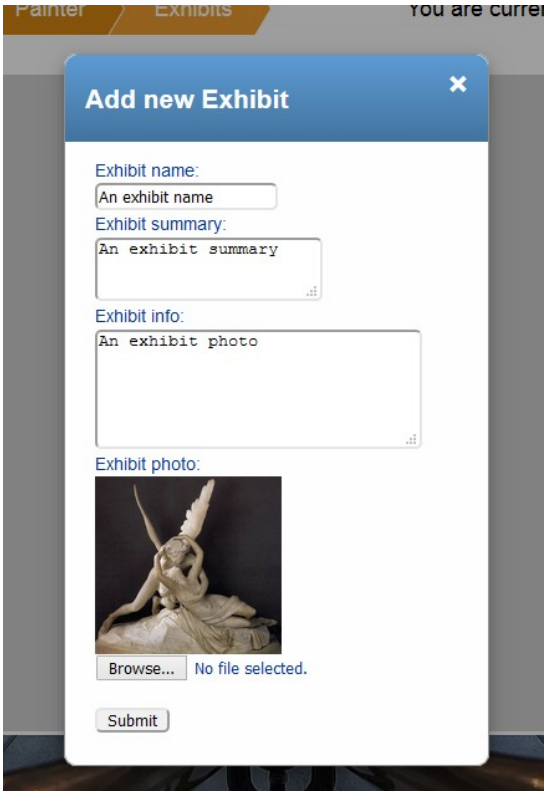
- Στο αριστερό κομμάτι που βρίσκεται το αντικείμενο καμβάς με τα σχεδιασμένα δωμάτια από το προηγούμενο βήμα αριθμημένα για να ταιριάζουν με τη λίστα στα δεξιά

- Στη λίστα που περιέχει τα δωμάτια, τα οποία πατώντας τα, εμφανίζεται ένας πίνακας με τα διαθέσιμα εκθέματα, με τα κουμπί επεξεργασίας και διαγραφής, και το κουμπί προσθήκης ενός νέου εκθέματος.



Εικόνα 8: Παράθυρο διαχείρισης ορόφου-δωματίων-εκθεμάτων στη διαδικτυακή εφαρμογή

Πατώντας το κουμπί προσθήκης ή επεξεργασίας ενός εκθέματος, εμφανίζεται μια φόρμα, στην οποία ο χρήστης πρέπει να εισάγει το όνομα του εκθέματος, μια μικρή περιγραφή, τις πληροφορίες του εκθέματος και μια εικόνα.



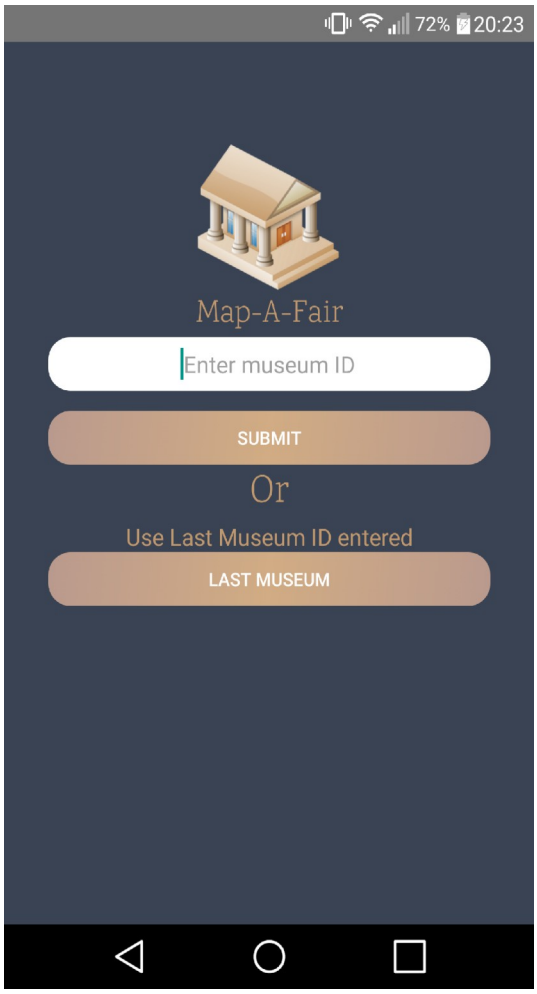
Εικόνα 9: Παράθυρο επεξεργασίας εκθέματος στη διαδικτυακή εφαρμογή

Όταν ο χρήστης τελειώσει με την επεξεργασία των εκθεμάτων, και πατήσει το κουμπί 'Done', ανακατευθύνεται στην σελίδα homepage.php όπου και είναι και η αρχική σελίδα για έναν συνδεμένο χρήστη.

4.2 Παρουσίαση εφαρμογής για κινητά τηλέφωνα

Αφού γίνει η εγκατάσταση της εφαρμογής σε μια κινητή συσκευή, πατώντας πάνω της, ο χρήστης μεταφέρεται στην πρώτη σελίδα(Activity) της εφαρμογής. Εκεί έχει 2 επιλογές:

- Να εισάγει το αναγνωριστικό που του έχει δοθεί για ένα συγκεκριμένο μουσείο
- Να πατήσει το κουμπί για να του εμφανιστεί ο χάρτης του τελευταίου μουσείου που είχε εισάγει.



Εικόνα 10: Παράθυρο εισαγωγής αναγνωριστικού μουσείου στην εφαρμογή για κινητά τηλέφωνα

Κάνοντας οποιαδήποτε από τις 2 ενέργειες, μεταφέρει το χρήστη στην επόμενη σελίδα της εφαρμογής. Η συγκεκριμένη σελίδα έχει ως κύριο σκοπό να μπορεί ένας χρήστης-επισκέπτης να δεί τον χάρτη του μουσείου, να πατήσει πάνω σε δωμάτια και να δει τις πληροφορίες ενός εκθέματος αλλά δίνεται και η δυνατότητα σε ένα χρήστη-διαχειριστή να επεξεργαστεί η να προσθέσει εκθέματα από μια κινητή συσκευή.

Η σελίδα αυτή χωρίζεται σε 3 μέρη:

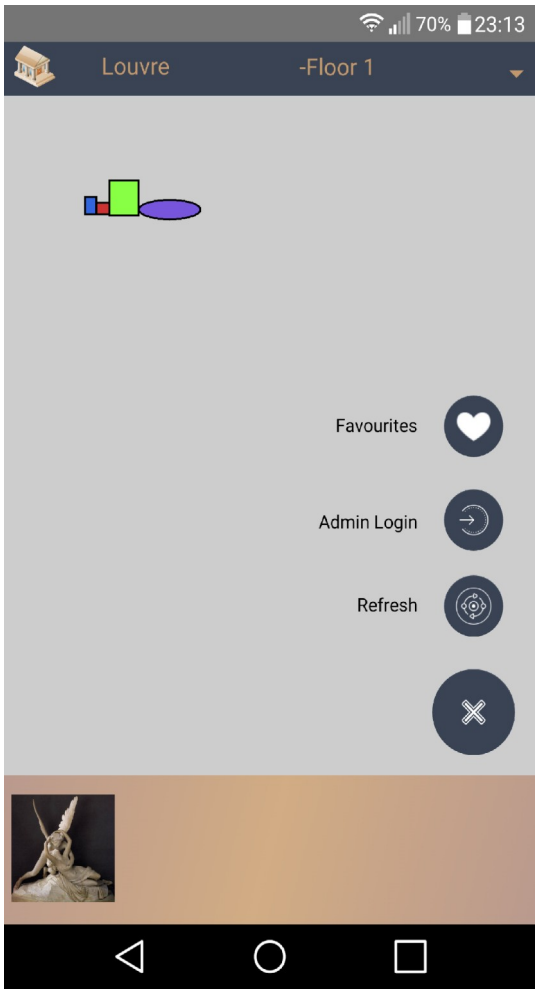
- Στο πάνω μέρος βρίσκεται το logo της εφαρμογής, το όνομα του μουσείου που κοιτά και ένα πτυσσόμενο μενού(drop down menu) με τους ορόφους του μουσείου.

- Στο μεσαίο μέρος, που καταλαμβάνει και το περισσότερο μέρος της σελίδας, βρίσκεται το αντικείμενο canvas για κινητά τηλέφωνα, που έχει σχεδιασμένο τον χάρτη
- Στο κάτω μέρος βρίσκεται το μενού που εμφανίζονται τα εκθέματα όταν ο χρήστης πατήσει πάνω σε κάποιο δωμάτιο.

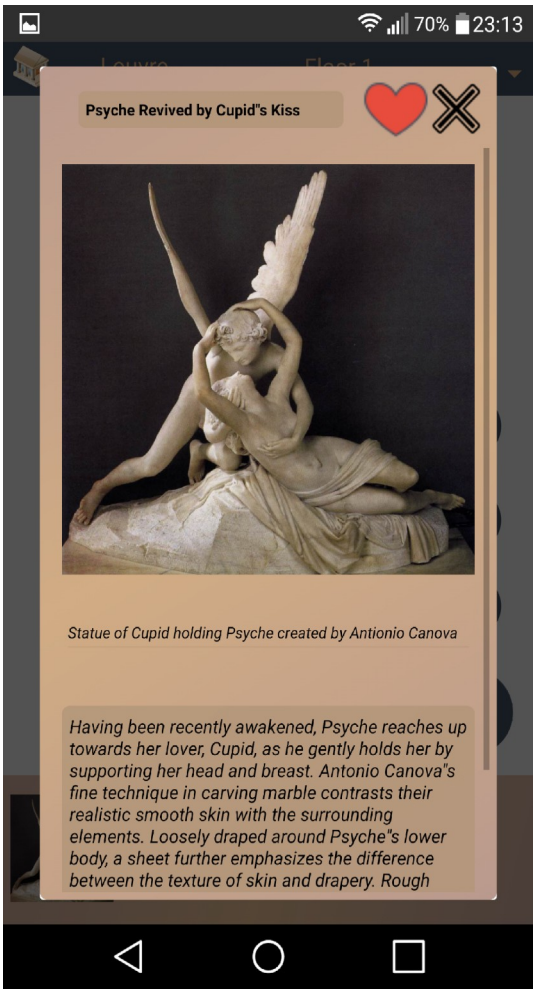
Μέσα στη σελίδα υπάρχει επίσης και ένα κουμπί με έναν σταυρό ως εικονίδιο, το οποίο όταν ο χρήστης το πατήσει, ανοίγει ένα μενού με τις εξής επιλογές:

- Refresh, που ανανεώνει τη σελίδα
- Admin login, Που επιτρέπει στο διαχειριστή του μουσείου να συνδεθεί με το λογαριασμό του
- Favourites, που περιέχει μια λίστα με εκθέματα που έχει θέσει ο χρήστης ως αγαπημένα.

Ο απλός χρήστης-επισκέπτης μπορεί να πατήσει πάνω σε ένα δωμάτιο, να του εμφανίσει τα διαθέσιμα εκθέματα, και πατώντας πάνω σε αυτά να δει τις πληροφορίες που διατίθενται και αν θέλουν να το προσθέσουν στα αγαπημένα τους.

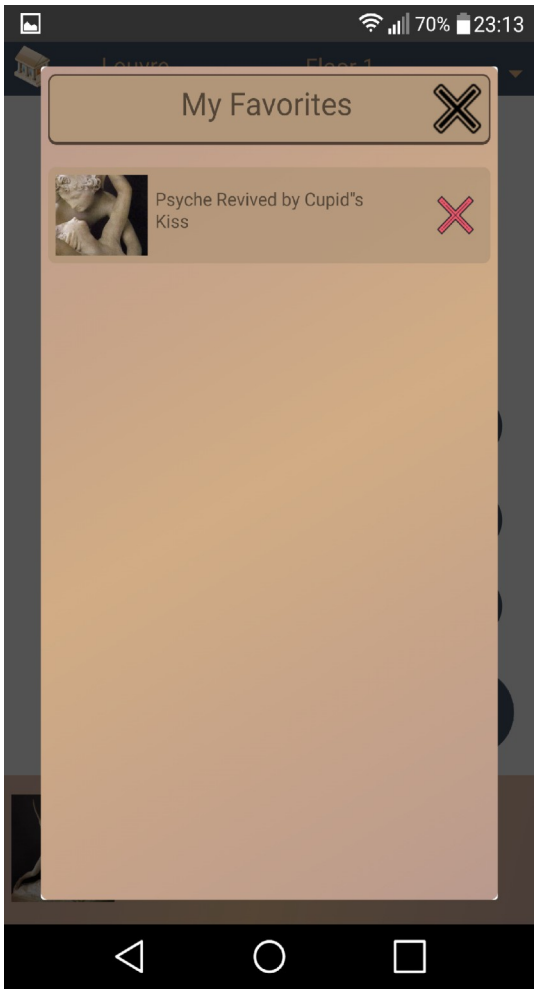


Εικόνα 11: Παράθυρο που εμφανίζεται ο όροφος ενός μουσείου μαζί με το μενού των ορόφων, μενού ενεργειών και λίστα εκθεμάτων.



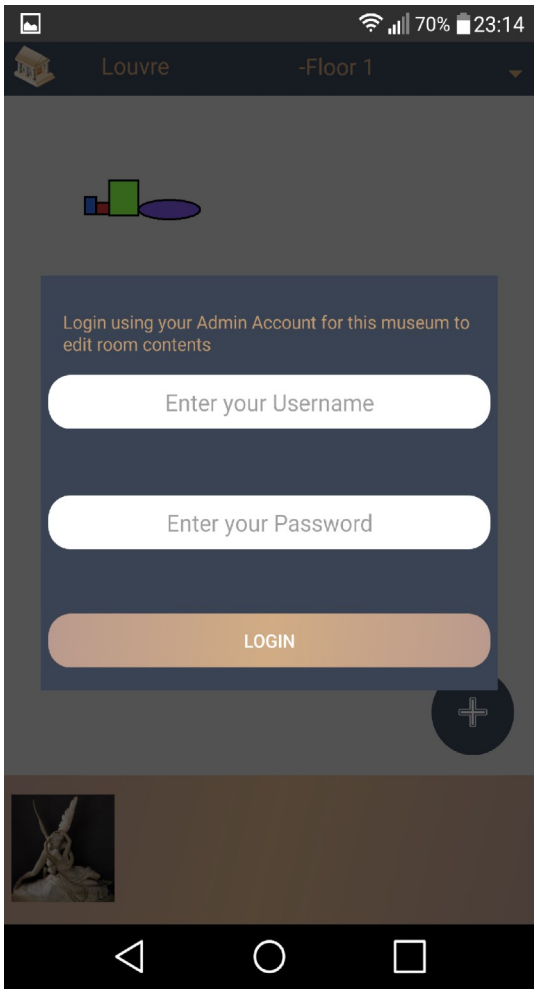
Εικόνα 12: Παράθυρο με τις πληροφορίες ενός εκθέματος.

Πατώντας στην λίστα με τα Αγαπημένα, εμφανίζονται όλα τα εκθέματα που έχει επιλέξει ο χρήστης, και πατώντας πάνω σε ένα από αυτά, ανοίγουν οι πληροφορίες του εκθέματος.



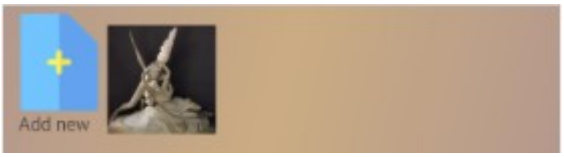
Εικόνα 13: Λίστα αγαπημένων εκθεμάτων

Όταν πατηθεί το κουμπί 'Admin login' εμφανίζεται μια σελίδα, στην οποία ο χρήστης-διαχειριστής πρέπει να εισάγει το username και τον κωδικό του για να μπορέσει να έχει επιπλέον δυνατότητες μέσα στην εφαρμογή. Η σύνδεση του διαχειριστή μπορεί να γίνει από οποιαδήποτε συσκευή, και όταν αυτή είναι επιτυχής, το κουμπί 'Admin login' μετατρέπεται σε 'Logout'



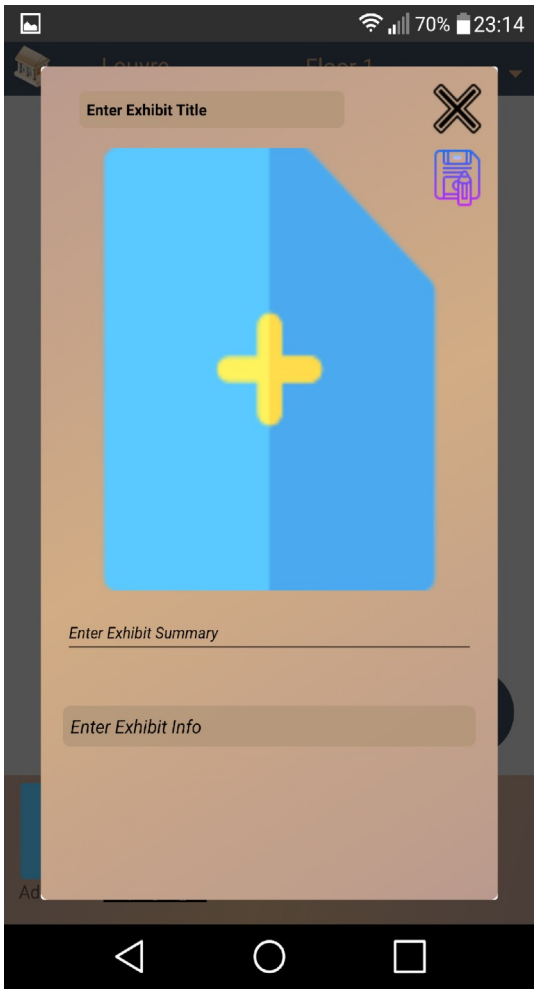
Εικόνα 14: Παράθυρο εισαγωγής στοιχείων σύνδεσης του διαχειριστή του μουσειακού χώρου

Έχοντας κάνει είσοδο ως διαχειριστής και πατώντας πάνω σε ένα δωμάτιο, πλέον εμφανίζεται, μαζί με την λίστα των εκθεμάτων, και ένα κουμπί για να γίνει προσθήκη ενός νέου εκθέματος.



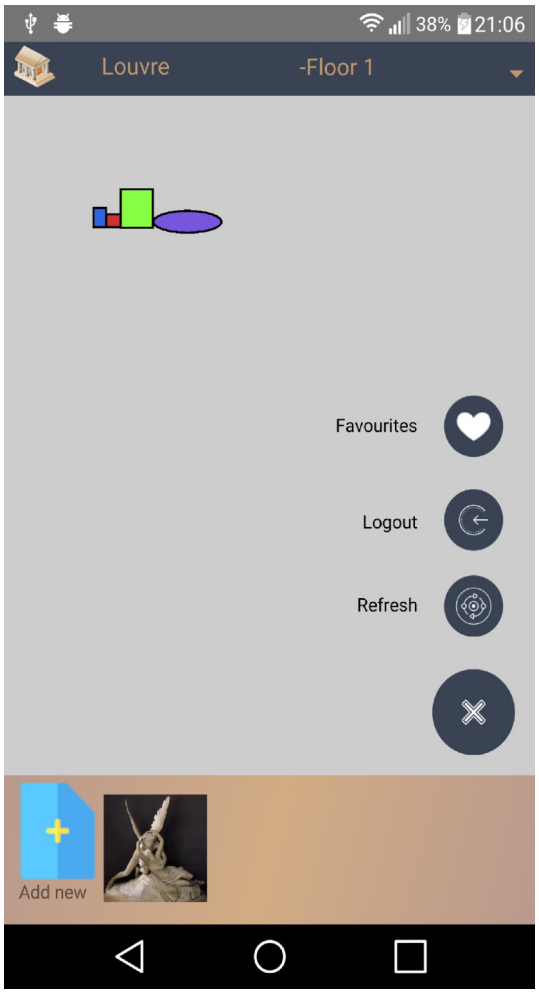
Εικόνα 15: Λίστα εκθεμάτων με το επιπρόσθετο κουμπί προσθήκης εκθέματος για το χρήστη διαχειριστή

Πατώντας το, ο χρήστης πρέπει να εισάγει τον τίτλο, την περιγραφή και τις πληροφορίες του εκθέματος καθώς και την εικόνα.

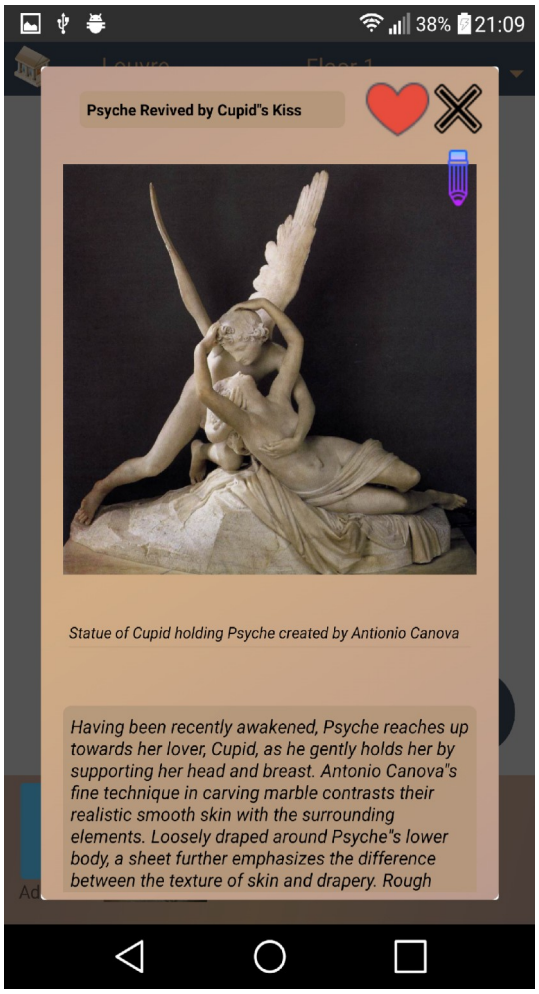


Εικόνα 16: Παράθυρο προσθήκης εκθέματος

Η εικόνα που θα εισάγουμε μπορεί να είναι από τη συλλογή μας, αλλά δίνεται και η δυνατότητα να τραβηχτεί εκ νέου μια φωτογραφία και να επιλεγεί. Όταν έχει τελειώσει η επεξεργασία του εκθέματος, ο χρήστης-διαχειριστής πρέπει να πατήσει το εικονίδιο της αποθήκευσης για να αποθηκευτούν τα δεδομένα που εισήγαγε. Ο χρήστης μπορεί επίσης να επεξεργαστεί και τα ήδη υπάρχοντα εκθέματα και να τους αλλάξει τις πληροφορίες και την εικόνα, πατώντας πάνω σε ένα έκθεμα και πατώντας στο εικονίδιο επεξεργασίας(μολύβι)



Εικόνα 17: Παράθυρο που εμφανίζεται ο όροφος ενός μουσείου μαζί με το μενού των ορόφων, μενού ενεργειών και λίστα εκθεμάτων όταν ο διαχειριστής είναι συνδεμένος.



Εικόνα 18: Παράθυρο με τις πληροφορίες του εκθέματος όταν ο διαχειριστής είναι συνδεμένος.

Όταν έχει τελειώσει με την επεξεργασία, μπορεί να πατήσει το κουμπί 'Logout' για να επανέλθει η εφαρμογή στη βασική λειτουργία της, δηλαδή να μην μπορεί να γίνει κανένα είδος επεξεργασίας.

ΚΕΦΑΛΑΙΟ 5: ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΠΡΟΤΑΣΕΙΣ ΑΝΑΠΤΥΞΗΣ

5.1 Συμπεράσματα

Το φιλικό περιβάλλον ως προς το χρήστη που αναπτύχθηκε εξασφαλίζει πως η δυσφορία του χρήστη, είτε ως διαχειριστής είτε ως απλός επισκέπτης ενός χώρου, θα είναι μειωμένη καθώς η διαχείριση της εφαρμογής είναι πολύ απλοποιημένη μέσω του design της. Πρόκειται μια εφαρμογή με διακριτά βήματα ως προς τη χρήση της και ως προς τον σκοπό που έχει δημιουργηθεί και λειτουργεί, που σε συνδυασμό με την ταχύτητα απόκρισης που έχουν και οι 2 εκδόσεις της εφαρμογής, συμβάλλουν και αυτά στην συνολική εμπειρία του χρήστη.

Τέλος να αναφερθεί πως η εφαρμογή είναι open source, δηλαδή ο κώδικας είναι ανοιχτός προς τροποποίηση, χρησιμοποιώντας απλές τεχνολογίες ώστε να μπορεί να τροποποιηθεί η εφαρμογή στις ανάγκες του προγραμματιστή.

5.2 Μελλοντικές επεκτάσεις

Στο μέλλον θα μπορούσαν να προστεθούν οι παρακάτω επεκτάσεις:

- Να μπορούν οι χρήστες-διαχειριστές να προσθέτουν επιπλέον πληροφορίες για το μουσειακό τους χώρο και τους ορόφους τους ώστε να μπορούν να τα βλέπουν οι χρήστες-επισκέπτες στα κινητά τους τηλέφωνα
- Να έχουν οι χρήστες-διαχειριστές τη δυνατότητα να προσθέτουν και να επεξεργάζονται ορόφους και δωμάτια από ένα κινητό τηλέφωνο
- Να υποστηρίζει η εφαρμογή κάθε είδος εσωτερικού χώρου πέραν χώρων εκθεσιακής φύσεως, όπως για παράδειγμα χάρτες ενός κτηρίου εταιρείας ή χάρτες αεροδρομίων κτλπ.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Αναφορα σε ιστοσελίδες:

- Front and back ends : https://en.wikipedia.org/wiki/Front_and_back_ends
- Τι Είναι Το Frontend, Τι Το Backend και Οι Διαφορές Τους: http://www.wdf.gr/articles/general/τι-είναι-το-frontend-τι-το-backend-και-οι-διαφορές-του.html
- XAMPP : https://en.wikipedia.org/wiki/XAMPP
- Sublime Text: https://en.wikipedia.org/wiki/Sublime_Text
- Apache HTTP εξυπηρετητής : https://el.wikipedia.org/wiki/Apache_HTTP_εξυπηρετητής
- MySQL: https://en.wikipedia.org/wiki/MySQL
- History of MySQL: https://www.datasciencecentral.com/profiles/blogs/history-of-mysql
- Android Studio: https://en.wikipedia.org/wiki/Android_Studio
- HTML: https://en.wikipedia.org/wiki/HTML
- Cascading Style Sheets : https://en.wikipedia.org/wiki/Cascading_Style_Sheets
- JavaScript: https://en.wikipedia.org/wiki/JavaScript
- jQuery: https://en.wikipedia.org/wiki/JQuery
- AJAX: https://en.wikipedia.org/wiki/Ajax_(programming)
- jCanvas: https://projects.calebevans.me/jcanvas/
- Java(programming language): https://en.wikipedia.org/wiki/Java_(programming_language)
- PHP: https://en.wikipedia.org/wiki/PHP

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

XAMPP	Cross-Platform (X), Apache (A), MariaDB (M), PHP (P) and Perl (P)
LAMP	Linux, Apache, MySQL, PHP/Python/Perl
W3C	World Wide Web Consortium
HTTP	HyperText Transfer Protocol
HTTPd	HyperText Transfer Protocol daemon
AJAX	Asynchronous JavaScript and XML
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
PHP	Personal Home Page
PHP/FI	Personal Home Page /File Intepreter
JDK	Java Development Kit
API	Application Program Interface
XML	Extensible Markup Language
JSON	JavaScript Object Notation

ΠΑΡΑΡΤΗΜΑ Α’

Κώδικας του αρχείου MuseumActivity.java που αναφέρθηκε στην ενότητα 3.7

```
package mapafair.mapafair;

//import the following that are needed for the file
import android.annotation.SuppressLint;
import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Matrix;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.support.design.widget.FloatingActionButton;
import android.support.v7.app.AppCompatActivity;
import android.view.LayoutInflater;
import android.view.MotionEvent;
import android.view.ScaleGestureDetector;
import android.view.View;
import android.view.animation.OvershootInterpolator;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.RelativeLayout;
import android.widget.ScrollView;
import android.widget.Spinner;
```

```
import android.widget.TextView;
import android.widget.Toast;
import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.JsonArrayRequest;
import com.android.volley.toolbox.Volley;
import com.esafirm.imagepicker.features.ImagePicker;
import com.esafirm.imagepicker.features.IpCons;
import com.esafirm.imagepicker.features.ReturnMode;
import com.esafirm.imagepicker.model.Image;
import com.squareup.picasso.Picasso;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import java.io.File;
import java.util.ArrayList;
import static android.text.TextUtils.isEmpty;


public class MuseumActivity extends AppCompatActivity {

//Initialize variables that will be used in this file
    private RequestQueue mQueue;
    int[][] coords = new int[150][150];
    String[][] exhibits = new String[150][150];
    String[][] fexhibits = new String[150][150];
    String[][] floors = new String[150][2];
    int length, favslength = 0;
    float scalenumber = 1;
    int museumid, floorid;
    FloatingActionButton fabmenu, reloadbtn, adminbtn, favbtn;
    TextView refreshtv, adminlogin, favtv;
    Float translationY = 100f;
    OvershootInterpolator interpolator = new OvershootInterpolator();
    final String TAG = "MainActivity";
    String path;
    Boolean isMenuOpen = false;
```

```
    int exhibitphotoid = 0;
    View vi;
    int imagebutton = 0;
    Boolean isadmin = false;
    ImageView exhibitphoto;
    String name;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_museum);
//get Intent data from LoginActivity.java
        Intent intent = getIntent();
        String museumname = intent.getStringExtra("museumname");
        TextView tvmuseumname = findViewById(R.id.etmuseumname);
//set the name of the museum
        tvmuseumname.setText(museumname);
        mQueue = Volley.newRequestQueue(this);

        jsonGetfloors();

        initFabMenu();


    }
//function used to add the image in the activity when editing an exhibit
    @Override
    protected void onActivityResult(int requestCode, final int
resultCode, Intent data) {

        super.onActivityResult(requestCode, resultCode, data);
        if (ImagePicker.shouldHandle(requestCode, resultCode, data)) {
            if (resultCode != RESULT_CANCELED && resultCode ==
RESULT_OK) {
                // Get a list of picked images
                //List<Image> images = ImagePicker.getImages(data);
                // or get a single image only

                Image image = ImagePicker.getFirstImageOrNull(data);
```

```
                Uri selectedImageURI = data.getData();
                LayoutInflater inflater = (LayoutInflater)
getSystemService(Context.LAYOUT_INFLATER_SERVICE);

                exhibitphoto = vi.findViewById(exhibitphotoid);

                path = "\"" + String.valueOf(selectedImageURI) + "\"";
                Bitmap myImg =
BitmapFactory.decodeFile(image.getPath());
                Matrix matrix = new Matrix();
                Bitmap rotated = Bitmap.createBitmap(myImg, 0, 0,
myImg.getWidth(), myImg.getHeight(),
                        matrix, true);
                exhibitphoto.setImageBitmap(rotated);



                path = image.getPath();
                name = image.getName();

                return;
            }
        }


    }
//function that initialized the floors, then grabs the room and exhibits
data from the database
    private void jsonGetfloors() {
        Intent intent = getIntent();
        museumid = intent.getIntExtra("museumid", 0);
        final ArrayList<String> arrayList1 = new ArrayList<String>();
        final Spinner floorspinner = findViewById(R.id.floorspinner);

        final String url =
"http://mapafair.hostingerapp.com/androidPHPFiles/androidgetfloorcounter
.php?museumid=" + museumid;
        JsonArrayRequest request = new
JsonArrayRequest(Request.Method.GET, url, null,
```

```
                new Response.Listener<JSONArray>() {
                    @Override
                    public void onResponse(JSONArray response) {
                        try {

                            for (int i = 0; i < response.length(); i++)
{
                                JSONObject room =
response.getJSONObject(i);
                                String floorid =
room.getString("floorid");
                                String floorname =
room.getString("floorname");
                                floors[i][0] = floorid;
                                floors[i][1] = floorname;


                                arrayList1.add("-" + floors[i][1]);

                                ArrayAdapter<String> adp = new
ArrayAdapter<String>(MuseumActivity.this, R.layout.spinner, arrayList1);

adp.setDropDownViewResource(R.layout.spinner_item);
                                floorspinner.setAdapter(adp);

                            }

                        } catch (JSONException e) {
                            e.printStackTrace();

                        }
                    }
                }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                error.printStackTrace();
            }
        });
```

```
        mQueue.add(request);
        floorspinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            //on floor dropdown click, get the rooms in that floor
            @Override
            public void onItemSelected(AdapterView<?> parentView, View
selectedItemView, int position, long id) {
                final RelativeLayout hsv = findViewById(R.id.hsvlinear);
                hsv.removeAllViews();
                try {
                    SharedPreferences preferences =
getSharedPreferences("Mypref", 0);
                    preferences.edit().clear().commit();

                    setFloor(position);
                    jsonParseroom(position);
                    jsonGetroom();
                    jsonParseexhibit();

                } catch (JSONException e) {
                    e.printStackTrace();

                }
            }

            @Override
            public void onNothingSelected(AdapterView<?> parentView) {

            }

        });
    }

//function to exit anything opened, including this activity
    @Override
    public void onBackPressed() {
        finish();
```

```
    }
//function used to populate and handle events of floating action buttons

    public void initFabMenu() {
        fabmenu = findViewById(R.id.fabmenu);
        reloadbtn = findViewById(R.id.reloadbtn);
        refreshtv = findViewById(R.id.refreshtv);
        adminbtn = findViewById(R.id.adminbtn);
        adminlogin = findViewById(R.id.adminlogin);

        favbtn = findViewById(R.id.favbtn);
        favtv = findViewById(R.id.favtv);

        reloadbtn.setAlpha(0f);
        reloadbtn.setVisibility(View.GONE);
        refreshtv.setAlpha(0f);
        refreshtv.setVisibility(View.GONE);
        adminbtn.setAlpha(0f);
        adminbtn.setVisibility(View.GONE);
        adminlogin.setAlpha(0f);
        adminlogin.setVisibility(View.GONE);
        favbtn.setAlpha(0f);
        favbtn.setVisibility(View.GONE);
        favtv.setAlpha(0f);
        favtv.setVisibility(View.GONE);

        reloadbtn.setTranslationY(translationY);
        refreshtv.setTranslationY(translationY);
        adminbtn.setTranslationY(translationY);
        adminlogin.setTranslationY(translationY);
        favbtn.setTranslationY(translationY);
        favtv.setTranslationY(translationY);

        fabmenu.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if (isMenuOpen) {
                    closeMenu();
                } else {
```

```
                    openMenu();
                }
            }
        });

        reloadbtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                recreate();

            }
        });

        adminbtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if (!isadmin) {
                    final AlertDialog alertDialog = new
AlertDialog.Builder(MuseumActivity.this).create(); //Read Update
                    LayoutInflater inflater = (LayoutInflater)
getSystemService(Context.LAYOUT_INFLATER_SERVICE);
                    vi = inflater.inflate(R.layout.activity_admin_login,
null, true);
                    alertDialog.setView(vi);
                    alertDialog.setInverseBackgroundForced(false);
                    alertDialog.show();
                    final EditText username =
vi.findViewById(R.id.username);
                    final EditText password =
vi.findViewById(R.id.password);
                    Button adminloginbtn =
vi.findViewById(R.id.adminloginbtn);
                    // Response received from the server
                    adminloginbtn.setOnClickListener(new
View.OnClickListener() {
                        @Override
                        public void onClick(View view) {
                            final String ausername =
username.getText().toString();
```

```
                            final String apassword =
password.getText().toString();
                            Response.Listener<String> responseListener =
new Response.Listener<String>() {
                                @Override
                                public void onResponse(String response)
{
                                    try {

                                        JSONObject jsonResponse = new
JSONObject(response);
                                        boolean success =
jsonResponse.optBoolean("success");
                                        if (success) {
                                            isadmin = true;
                                            alertDialog.cancel();

                                            //set logout text and icon
                                            closeMenu();
                                            final RelativeLayout hsv =
findViewById(R.id.hsvlinear);
                                            hsv.removeAllViews();
                                        } else {

                                            isadmin = false;
                                            closeMenu();

                                        }
                                    } catch (JSONException e) {
                                        e.printStackTrace();
                                    }
                                }
                            };
                            AdminRequest adminRequest = new
AdminRequest(ausername, apassword, museumid, responseListener);
                            RequestQueue queue =
Volley.newRequestQueue(MuseumActivity.this);
                            queue.add(adminRequest);
```

```
                        }


                    });
                } else {
                    isadmin = false;
                    final RelativeLayout hsv =
findViewById(R.id.hsvlinear);
                    hsv.removeAllViews();
                }
                closeMenu();


            }
        });
        favbtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                favoritefunction();



            }
        });
    }
//sets the favourite list of a user, showing all the exhibits that were
pressed as favourites
    private void favoritefunction() {

        final AlertDialog alertDialog = new
AlertDialog.Builder(MuseumActivity.this).create(); //Read Update
        LayoutInflater inflater = (LayoutInflater)
getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        vi = inflater.inflate(R.layout.favoritealertbox, null, true);
        final ScrollView fsv = vi.findViewById(R.id.fsv);
        final RelativeLayout frl = vi.findViewById(R.id.frl);

        Button close = vi.findViewById(R.id.close);
        final SharedPreferences pref =
getApplicationContext().getSharedPreferences("myfavs", MODE_PRIVATE);
        favslength = pref.getInt("favslength", 0);
        String counter;
```

```
        int counter2 = 0;
        if (favslength != 0) {
            for (int i = 0; i <= favslength; i++) {
                counter = String.valueOf(i);

                if ((pref.getString("exhibitid" + counter, null)) ==
null) {
                    continue;
                }


                RelativeLayout newflayout = new
RelativeLayout(MuseumActivity.this);
                ImageButton in = new ImageButton(MuseumActivity.this);
                TextView tvexhibittitle = new
TextView(MuseumActivity.this);
                Button dlt = new Button(MuseumActivity.this);
                newflayout.setClickable(true);

                RelativeLayout.LayoutParams layout_params = new
RelativeLayout.LayoutParams(RelativeLayout.LayoutParams.MATCH_PARENT,
250);
                layout_params.topMargin = counter2 * 300 + 50;
                newflayout.setLayoutParams(layout_params);
                newflayout.setElevation(16);
                int exhibitid =
Integer.parseInt(pref.getString("exhibitid" + counter, null));
                in.setId(exhibitid);
                in.setVisibility(View.VISIBLE);

Picasso.get().load("http://mapafair.hostingerapp.com/uploads/" +
pref.getString("exhibitphoto" + counter, null)).into(in);
                RelativeLayout.LayoutParams in_params = new
RelativeLayout.LayoutParams(RelativeLayout.LayoutParams.WRAP_CONTENT,
RelativeLayout.LayoutParams.WRAP_CONTENT);
                in_params.width = 250;
                in_params.height = 250;

                RelativeLayout.LayoutParams tvmname_params = new
```

```
RelativeLayout.LayoutParams(RelativeLayout.LayoutParams.WRAP_CONTENT,
RelativeLayout.LayoutParams.WRAP_CONTENT);
                tvmname_params.setMargins(270, 0, 0, 0);
                tvmname_params.addRule(RelativeLayout.CENTER_VERTICAL);
                tvmname_params.width = 600;
                tvmname_params.height = 150;
                tvexhibittitle.setText(pref.getString("exhibitname" +
counter, null));


                RelativeLayout.LayoutParams dlt_params = new
RelativeLayout.LayoutParams(RelativeLayout.LayoutParams.WRAP_CONTENT,
RelativeLayout.LayoutParams.WRAP_CONTENT);
                dlt_params.addRule(RelativeLayout.CENTER_VERTICAL);
                dlt_params.addRule(RelativeLayout.ALIGN_PARENT_RIGHT);
                dlt_params.width = 150;
                dlt_params.height = 150;
                //dlt_params.topMargin=50;

                dlt.setClickable(true);
                dlt.setId(Integer.parseInt(pref.getString("exhibitid" +
counter, null)));

                frl.addView(newflayout);

newflayout.setBackgroundResource(R.drawable.favorexhibit);
                dlt.setBackgroundResource(R.drawable.red_delete_icon);
                dlt.setLayoutParams(dlt_params);
                in.setLayoutParams(in_params);
                tvexhibittitle.setLayoutParams(tvmname_params);
                newflayout.addView(in);
                newflayout.addView(tvexhibittitle);
                newflayout.addView(dlt);


                final String finalCounter = counter;
                dlt.setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View view) {
```

```
                        removefromfavpref(finalCounter);
                        alertDialog.cancel();
                        favoritefunction();

                    }
                });

                final int icounter =
Integer.parseInt(pref.getString("icounter" + counter, null));
                newflayout.setOnClickListener(new View.OnClickListener()
{
                    @Override
                    public void onClick(View view) {

                        exhibitalertbox(icounter);
                    }
                });

                counter2++;
            }
        } else {

            RelativeLayout newflayout = new
RelativeLayout(MuseumActivity.this);
            RelativeLayout.LayoutParams layout_params = new
RelativeLayout.LayoutParams(RelativeLayout.LayoutParams.MATCH_PARENT,
250);
            layout_params.topMargin = 300 + 50;
            newflayout.setLayoutParams(layout_params);
            newflayout.setElevation(16);

            TextView tvnodata = new TextView(MuseumActivity.this);
            RelativeLayout.LayoutParams tvmname_params = new
RelativeLayout.LayoutParams(500, 100);
            tvmname_params.setMargins(0, 0, 0, 0);

            tvmname_params.addRule(RelativeLayout.CENTER_IN_PARENT);
            tvmname_params.addRule(RelativeLayout.CENTER_HORIZONTAL);
```

```
            tvnodata.setText("No favorites added");


            frl.addView(newflayout);
            newflayout.setBackgroundResource(R.drawable.favorexhibit);
            tvnodata.setLayoutParams(tvmname_params);
            newflayout.addView(tvnodata);
        }
        close.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                alertDialog.cancel();
            }
        });


        alertDialog.setInverseBackgroundForced(false);
        alertDialog.setView(vi);
        alertDialog.show();
    }
//function to open the FAB menu
    private void openMenu() {
        isMenuOpen = !isMenuOpen;
        reloadbtn.setVisibility(View.VISIBLE);
        refreshtv.setVisibility(View.VISIBLE);
        adminbtn.setVisibility(View.VISIBLE);
        if (isadmin) {
            adminlogin.setText("Logout");
            adminbtn.setRotation(180);
            adminlogin.setVisibility(View.VISIBLE);


        } else {
            adminbtn.setRotation(0);
            adminlogin.setText("Admin Login");
            adminlogin.setVisibility(View.VISIBLE);

        }
        favbtn.setVisibility(View.VISIBLE);
```

```
        favtv.setVisibility(View.VISIBLE);


fabmenu.animate().setInterpolator(interpolator).rotation(45f).setDuratio
n(300).start();

reloadbtn.animate().translationY(0f).alpha(1f).setInterpolator(interpola
tor).setDuration(300).start();

refreshtv.animate().translationY(0f).alpha(1f).setInterpolator(interpola
tor).setDuration(300).start();

adminbtn.animate().translationY(0f).alpha(1f).setInterpolator(interpolat
or).setDuration(300).start();

adminlogin.animate().translationY(0f).alpha(1f).setInterpolator(interpol
ator).setDuration(300).start();

favbtn.animate().translationY(0f).alpha(1f).setInterpolator(interpolator
).setDuration(300).start();

favtv.animate().translationY(0f).alpha(1f).setInterpolator(interpolator)
.setDuration(300).start();

refreshtv.animate().translationY(0f).alpha(1f).setInterpolator(interpola
tor).setDuration(300).start();

    }
//function to close the FAB menu
    private void closeMenu() {
        isMenuOpen = !isMenuOpen;


fabmenu.animate().setInterpolator(interpolator).rotation(0f).setDuration
(300).start();

reloadbtn.animate().translationY(translationY).alpha(0f).setInterpolator
(interpolator).setDuration(300).start();
```

```
refreshtv.animate().translationY(translationY).alpha(0f).setInterpolator
(interpolator).setDuration(300).start();

adminbtn.animate().translationY(translationY).alpha(0f).setInterpolator(
interpolator).setDuration(300).start();

adminlogin.animate().translationY(translationY).alpha(0f).setInterpolato
r(interpolator).setDuration(300).start();

favbtn.animate().translationY(translationY).alpha(0f).setInterpolator(in
terpolator).setDuration(300).start();

favtv.animate().translationY(translationY).alpha(0f).setInterpolator(int
erpolator).setDuration(300).start();

        reloadbtn.setVisibility(View.GONE);
        refreshtv.setVisibility(View.GONE);
        adminbtn.setVisibility(View.GONE);
        adminlogin.setVisibility(View.GONE);
        favbtn.setVisibility(View.GONE);
        favtv.setVisibility(View.GONE);
    }
//function to remove favourite from favourites list
    public void removefromfavpref(String counter) {
        SharedPreferences pref =
getApplicationContext().getSharedPreferences("myfavs", MODE_PRIVATE);
        SharedPreferences.Editor editor = pref.edit();


        editor.remove("exhibitid" + counter)
                .remove("exhibitname" + counter)
                .remove("exhibitsum" + counter)
                .remove("exhibitinfo" + counter)
                .remove("exhibitphoto" + counter);


        favslength--;
        editor.putInt("favslength",favslength);
```

```
        editor.apply();


    }
//function that is called when pressing the heart(favourite) button on
an exhibit
    public void addtomyfavpref(String icounter, String exhibitid, String
exhibitname, String exhibitsum, String exhibitinfo, String exhibitphoto)
{
        SharedPreferences pref =
getApplicationContext().getSharedPreferences("myfavs", MODE_PRIVATE);
        SharedPreferences.Editor editor = pref.edit();
        fexhibits[favslength][0] = exhibitid;
        fexhibits[favslength][1] = exhibitname;
        fexhibits[favslength][2] = exhibitsum;
        fexhibits[favslength][3] = exhibitinfo;
        fexhibits[favslength][4] = exhibitphoto;

        editor.putString("icounter" + favslength, icounter);
        editor.putString("exhibitid" + favslength, fexhibits[favslength]
[0]);
        editor.putString("exhibitname" + favslength,
fexhibits[favslength][1]);
        editor.putString("exhibitsum" + favslength,
fexhibits[favslength][2]);
        editor.putString("exhibitinfo" + favslength,
fexhibits[favslength][3]);
        editor.putString("exhibitphoto" + favslength,
fexhibits[favslength][4]);

        favslength++;
        editor.putInt("favslength", favslength);


        editor.apply();
    }
```

```
//sets the current floor viewed by a user
    private void setFloor(int position) {
        SharedPreferences museumidpref =
getApplicationContext().getSharedPreferences("floorid", MODE_PRIVATE);
        museumidpref.edit().clear().commit();
        SharedPreferences.Editor editor = museumidpref.edit();
        editor.putString("floorid", floors[position][0]);
        editor.apply();
        floorid = Integer.parseInt(floors[position][0]);
    }
//creates the rooms in the user view's canvas
    private void jsonParseroom(int selectedfloor) {
        Intent intent = getIntent();
        museumid = intent.getIntExtra("museumid", 0);
        final LinearLayout layout = findViewById(R.id.linear2);
        layout.removeAllViews();


        final String url =
"http://mapafair.hostingerapp.com/androidPHPFiles/databasetopainterandro
id2.php?museumid=" + museumid;
        //missing floor changing. Will deal with that later
        JsonArrayRequest request = new
JsonArrayRequest(Request.Method.GET, url, null,
                new Response.Listener<JSONArray>() {
                    @Override
                    public void onResponse(JSONArray response) {
                        try {

                            for (int i = 0; i < response.length(); i++)
{
                                JSONObject room =
response.getJSONObject(i);
                                String id = room.getString("id");
                                String floorid =
room.getString("floorid");
                                String startX =
room.getString("startX");
```

```
                                String startY =
room.getString("startY");
                                String endX = room.getString("endX");
                                String endY = room.getString("endY");
                                String roomtype =
room.getString("roomtype");
                                String roomcolor =
room.getString("roomcolor");
                                SharedPreferences pref =
getApplicationContext().getSharedPreferences("MyPref", MODE_PRIVATE);
                                SharedPreferences.Editor editor =
pref.edit();
                                editor.putString("json" + i,
room.toString());
                                editor.putInt("length",
response.length());
                                editor.putInt("id",
Integer.parseInt(id));
                                editor.putInt("floorid",
Integer.parseInt(floorid));
                                editor.putInt("startX",
Integer.parseInt(startX));
                                editor.putInt("startY",
Integer.parseInt(startY));
                                editor.putInt("endX",
Integer.parseInt(endX));
                                editor.putInt("endY",
Integer.parseInt(endY));
                                editor.putString("roomtype", roomtype);
                                editor.putString("roomcolor",
roomcolor);
                                editor.apply();
                            }

                        } catch (JSONException e) {
                            e.printStackTrace();

                        }
                    }
```

```
                }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                error.printStackTrace();
            }
        });


        mQueue.add(request);

    }
//sets the rooms in an array used in this file
    private void jsonGetroom() throws JSONException {
        SharedPreferences pref =
getApplicationContext().getSharedPreferences("MyPref", MODE_PRIVATE);
        length = pref.getInt("length", 0);
        for (int i = 0; i < length; i++) {
            String json = pref.getString("json" + i, null);

            JSONObject myJSON = new JSONObject(json);

            coords[i][0] = myJSON.getInt("id");
            coords[i][1] = myJSON.getInt("startX");
            coords[i][2] = myJSON.getInt("startY");
            coords[i][3] = myJSON.getInt("endX");
            coords[i][4] = myJSON.getInt("endY");
            coords[i][5] = myJSON.getInt("floorid");

        }

    }

//used for the zooming in
    @Override
    public boolean onTouchEvent(MotionEvent ev) {
        // Let the ScaleGestureDetector inspect all events.
        CustomView.mScaleDetector.onTouchEvent(ev);

        return true;
```

```
    }
//used to create the canvas
    private void createCustomView() throws JSONException {
        final LinearLayout layout = findViewById(R.id.linear2);
        layout.addView(new CustomView(this));
    }
//used to create the exhibits when clicking on a room
    @SuppressLint("ClickableViewAccessibility")
    private void jsonParseexhibit() throws JSONException {

        final String url =
"http://mapafair.hostingerapp.com/androidPHPFiles/databasetoaccordion1.p
hp?museumid=" + museumid;

        JsonArrayRequest request = new
JsonArrayRequest(Request.Method.GET, url, null,
                new Response.Listener<JSONArray>() {

                    @Override
                    public void onResponse(JSONArray response) {

                        try {
                            for (int i = 0; i < response.length(); i++)
{
                                JSONObject exhibit =
response.getJSONObject(i);

                                String eid =
exhibit.getString("exhibitid");
                                String ename =
exhibit.getString("exhibitname");
                                String esum =
exhibit.getString("exhibitsum");
                                String einfo =
exhibit.getString("exhibitinfo");
                                String ephoto =
exhibit.getString("exhibitphoto");
                                String eroomid =
exhibit.getString("roomid");
```

```
                                exhibits[i][0] = eid;
                                exhibits[i][1] = ename;
                                exhibits[i][2] = esum;
                                exhibits[i][3] = einfo;
                                exhibits[i][4] = ephoto;
                                exhibits[i][5] =
String.valueOf(Integer.parseInt(eroomid));

                            }

                        } catch (JSONException e) {
                            e.printStackTrace();
                        }

                    }

                }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                error.printStackTrace();
            }
        });

        mQueue.add(request);

        final LinearLayout layout = findViewById(R.id.linear2);
        createCustomView();
        final RelativeLayout hsv = findViewById(R.id.hsvlinear);


        layout.setOnTouchListener(new View.OnTouchListener() {
            //used to zoom in-out
            @Override
            public boolean onTouch(View view, MotionEvent motionEvent) {
                switch (motionEvent.getAction()) {
                    case MotionEvent.ACTION_DOWN:
                        hsv.removeAllViews();
                        boolean addnewbtn = true;
                        for (int i = 0; i < length; i++) {
```

```
                            if (motionEvent.getX() > coords[i][1] *
CustomView.scaleFactor && motionEvent.getX() < coords[i][3] *
CustomView.scaleFactor) {
                                if (motionEvent.getY() > coords[i][2] *
CustomView.scaleFactor && motionEvent.getY() < coords[i][4] *
CustomView.scaleFactor) {
                                    if (coords[i][5] == floorid) {

                                        imagebutton = 0;
                                        int roomid = 0;
                                        if (isadmin && addnewbtn) {

addnewexhibitbutton(coords[i][0]);
                                            addnewbtn = false;
                                        }
                                        for (int j = 0; j < 40; j++) {

                                            if (exhibits[j][5] != null
&& !exhibits[j][5].isEmpty()) {
                                                roomid =
Integer.parseInt(exhibits[j][5]);

                                                if (coords[i][0] ==
roomid && coords[i][5] == floorid) {


createimagebutton(j);
                                                }
                                            }
                                        }
                                    }
                                }
                            }
                            break;
                        case MotionEvent.ACTION_MOVE:
                            int count = motionEvent.getPointerCount();
                            View v = layout.getChildAt(0);
```

```
                        if (count == 2) {
                            scalenumber = 1 + distance(motionEvent, 0,
1) / (layout.getWidth() + layout.getHeight());
                        }
                        break;
                    case MotionEvent.ACTION_UP:
                        break;
                }
                return false;
            }

        });

    }
//initialized the add new exhibit button so that it will appear first in
the list when the user is admin and includes
    //the functionalities used to edit/create an exhibit
    private void addnewexhibitbutton(final int roomid) {
        RelativeLayout hsv = findViewById(R.id.hsvlinear);


        ImageButton in = new ImageButton(MuseumActivity.this);
        TextView tvmname = new TextView(MuseumActivity.this);
        in.setId(roomid);
        in.setClickable(true);

        in.setBackgroundResource(R.drawable.new_exhibit);
        in.setVisibility(View.VISIBLE);

        RelativeLayout.LayoutParams btn_params = new
RelativeLayout.LayoutParams(RelativeLayout.LayoutParams.WRAP_CONTENT,
RelativeLayout.LayoutParams.WRAP_CONTENT);
        btn_params.leftMargin = imagebutton * 220;
        btn_params.topMargin = 0;
        btn_params.width = 250;
        btn_params.height = 250;
        hsv.addView(in);
        in.setLayoutParams(btn_params);
        RelativeLayout.LayoutParams tvmname_params = new
```

```
RelativeLayout.LayoutParams(RelativeLayout.LayoutParams.WRAP_CONTENT,
RelativeLayout.LayoutParams.WRAP_CONTENT);
        tvmname_params.addRule(RelativeLayout.BELOW, in.getId());
        tvmname_params.leftMargin = 20;
        tvmname_params.width = 250;
        tvmname.setText("Add new ");
        tvmname.setLayoutParams(tvmname_params);
        hsv.addView(tvmname);
        imagebutton++;




        in.setOnClickListener(new View.OnClickListener() {
            public void onClick(View view) {
                final AlertDialog alertDialog = new
AlertDialog.Builder(MuseumActivity.this).create(); //Read Update
                LayoutInflater inflater = (LayoutInflater)
getSystemService(Context.LAYOUT_INFLATER_SERVICE);
                vi = inflater.inflate(R.layout.exhibitalertbox, null,
true);

                alertDialog.setView(vi);
                alertDialog.setInverseBackgroundForced(false);
                RelativeLayout eab = findViewById(R.id.eab);
                final ImageView exhibitphoto =
vi.findViewById(R.id.exhibitphoto);
                exhibitphotoid = 742019;
                exhibitphoto.setId(exhibitphotoid);

                final EditText etitle =
vi.findViewById(R.id.exhibittitle);

                final EditText exhibitsum =
vi.findViewById(R.id.exhibitsum);

                final EditText exhibitinfo =
vi.findViewById(R.id.exhibitinfo);
```

```
                Button close = vi.findViewById(R.id.close);
                final Button savebtn = vi.findViewById(R.id.savebtn);
                final Button editbtn = vi.findViewById(R.id.editbtn);
                final Button favorbtn = vi.findViewById(R.id.favorbtn);

exhibitphoto.setBackgroundResource(R.drawable.new_exhibit_big);
                favorbtn.setVisibility(View.GONE);
                editbtn.setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        setterexhibitphotoid(exhibitphotoid);
                        editbtn.setVisibility(View.GONE);
                        savebtn.setVisibility(View.VISIBLE);

                        exhibitphoto.isClickable();
                        etitle.setEnabled(true);
                        exhibitinfo.setEnabled(true);

                        exhibitsum.setEnabled(true);
                    }
                });
                editbtn.setVisibility(View.GONE);
                savebtn.setVisibility(View.VISIBLE);
                exhibitphoto.isClickable();
                exhibitphoto.setOnClickListener(new
View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        File filee = new
File("/storage/emulated/0/IMG_20190519_2145382076248583.jpg");

startActivityForResult(getImagePicker().getIntent(MuseumActivity.this),
IpCons.RC_IMAGE_PICKER);


                    }
                });
                etitle.setEnabled(true);
                exhibitinfo.setEnabled(true);
```

```
                exhibitsum.setEnabled(true);
                savebtn.setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        if (isEmpty(name)) {

                            final AlertDialog.Builder emptyphoto = new
AlertDialog.Builder(MuseumActivity.this);
                            emptyphoto.setTitle("Upload Not completed");
                            emptyphoto.setMessage("Please Upload a
photo");
                            emptyphoto.setPositiveButton("Close", new
DialogInterface.OnClickListener() {
                                        public void
onClick(DialogInterface dialog, int which) {
                                            //
innerdialog.setCancelable(true);
                                        }
                                    });
                                    // A null listener allows the button
to dismiss the dialog and take no further action.


                                    emptyphoto.show();


                        } else {
                        editbtn.setVisibility(View.VISIBLE);
                        savebtn.setVisibility(View.GONE);
                        etitle.setEnabled(false);
                        exhibitinfo.setEnabled(false);
                        exhibitsum.setEnabled(false);
                        exhibitphoto.setEnabled(false);
                        final String title =
etitle.getText().toString();
                        final String einfo =
exhibitinfo.getText().toString();
                        final String esum =
exhibitsum.getText().toString();
```

```
                        final String eroomid = String.valueOf(roomid);
                        final String exhibitid = String.valueOf('0');
                        final String photo = name;
                        final AlertDialog.Builder innerdialog = new
AlertDialog.Builder(MuseumActivity.this);
                        Response.Listener<String> responseListener = new
Response.Listener<String>() {
                            @Override
                            public void onResponse(String response) {

                                innerdialog.setTitle("Upload
Completed");
                                innerdialog.setMessage("A new exhibit
has been successfully created");
                                // Specifying a listener allows you to
take an action before dismissing the dialog.
                                // The dialog is automatically dismissed
when a dialog button is clicked.
                                innerdialog.setPositiveButton("Close",
new DialogInterface.OnClickListener() {
                                    public void onClick(DialogInterface
dialog, int which) {
                                        //
innerdialog.setCancelable(true);
                                    }
                                });
                                // A null listener allows the button to
dismiss the dialog and take no further action.


                                innerdialog.show();
                            }
                        };

                            UploadFileAsync ufa = new UploadFileAsync();

                            ufa.execute(path);
                            Exhibittodatabase etd = new
Exhibittodatabase(title, einfo, esum, eroomid, exhibitid, photo,
```

```
responseListener);
                            RequestQueue queue =
Volley.newRequestQueue(MuseumActivity.this);
                            queue.add(etd);
                        }
                    }
                });


                close.setOnClickListener(new View.OnClickListener() {

                    @Override
                    public void onClick(View v) {
                        alertDialog.cancel();
                    }
                });
                alertDialog.show();
                //<-- See This!
            }
        });
    }
//used to get the image
    private ImagePicker getImagePicker() {


        ImagePicker imagePicker = ImagePicker.create(this)
                .language("in") // Set image picker language

                .returnMode(ReturnMode.ALL)
                .single()
                .toolbarFolderTitle("Folder") // folder selection title
                .toolbarImageTitle("Tap to select") // image selection
title
                .toolbarDoneButtonText("DONE"); // done button text

        return imagePicker.limit(10) // max images can be selected (99
by default)
                .showCamera(true) // show camera or not (true by
default)
```

```
                .imageDirectory("Camera")   // captured image directory
name ("Camera" folder by default)
                .imageFullDirectory(Environment.getExternalStorageDirect
ory().getPath()); // can be full path
    }
//used in the zoom in-out
    public float distance(MotionEvent event, int first, int second) {
        if (event.getPointerCount() >= 2) {
            final float x = event.getX(first) - event.getX(second);
            final float y = event.getY(first) - event.getY(second);

            return (float) Math.sqrt(x * x + y * y);
        } else {
            return 0;
        }
    }

    public void setterexhibitphotoid(int id) {

        exhibitphotoid = id;

    }

    // initializes the exhibits' view for simple users and events
    public void createimagebutton(int i) {


        RelativeLayout hsv = findViewById(R.id.hsvlinear);
        RelativeLayout inbtn = new RelativeLayout(MuseumActivity.this);
        ImageButton in = new ImageButton(MuseumActivity.this);
        TextView tvmname = new TextView(MuseumActivity.this);
        in.setId(Integer.parseInt(exhibits[i][0]));
        in.setClickable(true);


        in.setBackgroundResource(R.drawable.new_exhibit);
        in.setVisibility(View.VISIBLE);

        if(!isEmpty(exhibits[i][4])) {
```

```
Picasso.get().load("http://mapafair.hostingerapp.com/uploads/" +
exhibits[i][4]).fit().into(in);
        }else{
            in.setBackgroundResource(R.drawable.no_image);
        }

        RelativeLayout.LayoutParams btn_params = new
RelativeLayout.LayoutParams(RelativeLayout.LayoutParams.WRAP_CONTENT,
RelativeLayout.LayoutParams.WRAP_CONTENT);
        btn_params.leftMargin = imagebutton * 250;
        btn_params.topMargin = 30;
        btn_params.width = 280;
        btn_params.height = 280;

        hsv.addView(in);
        in.setLayoutParams(btn_params);
        RelativeLayout.LayoutParams tvmname_params = new
RelativeLayout.LayoutParams(RelativeLayout.LayoutParams.MATCH_PARENT,
RelativeLayout.LayoutParams.WRAP_CONTENT);
        tvmname_params.addRule(RelativeLayout.BELOW, in.getId());

        tvmname_params.leftMargin = imagebutton * 250;

        tvmname_params.width = 240;
        tvmname.setText(exhibits[i][1]);
        tvmname.setLayoutParams(tvmname_params);

        imagebutton++;

        final int ii = i;

        in.setOnClickListener(new View.OnClickListener() {
            public void onClick(View view) {
                exhibitalertbox(ii);
            }

        });
```

```
    }
//creates the alert box in which appear the exhibit's
title,photo,summary and info and functionalities used
    private void exhibitalertbox(int ii) {

        final AlertDialog alertDialog = new
AlertDialog.Builder(MuseumActivity.this).create(); //Read Update
        LayoutInflater inflater = (LayoutInflater)
getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        vi = inflater.inflate(R.layout.exhibitalertbox, null, true);

        alertDialog.setView(vi);
        alertDialog.setInverseBackgroundForced(false);
        RelativeLayout eab = findViewById(R.id.eab);

        exhibitphoto = vi.findViewById(R.id.exhibitphoto);
        if (isEmpty(exhibits[ii][4])) {
            Picasso.get().load(R.drawable.no_image).into(exhibitphoto);

        } else {

Picasso.get().load("http://mapafair.hostingerapp.com/uploads/" +
exhibits[ii][4]).into(exhibitphoto);
        }

        final EditText etitle = vi.findViewById(R.id.exhibittitle);
        etitle.setText(exhibits[ii][1]);
        final EditText exhibitsum = vi.findViewById(R.id.exhibitsum);
        exhibitsum.setText(exhibits[ii][2]);
        final EditText exhibitinfo = vi.findViewById(R.id.exhibitinfo);
        exhibitinfo.setText(exhibits[ii][3]);
        Button close = vi.findViewById(R.id.close);
        final Button favorbtn = vi.findViewById(R.id.favorbtn);
        final Button savebtn = vi.findViewById(R.id.savebtn);
        final Button editbtn = vi.findViewById(R.id.editbtn);
        if (!isadmin) {
            editbtn.setVisibility(View.INVISIBLE);
        } else {
            editbtn.setVisibility(View.VISIBLE);
```

```
        }
        final int iii = ii;
        editbtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                editbtn.setVisibility(View.GONE);
                savebtn.setVisibility(View.VISIBLE);
                favorbtn.setVisibility(View.GONE);
                etitle.setEnabled(true);
                exhibitinfo.setEnabled(true);
                exhibitsum.setEnabled(true);
                exhibitphoto.isClickable();
                exhibitphotoid = exhibitphoto.getId();
                exhibitphoto.setOnClickListener(new
View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        File filee = new
File("/storage/emulated/0/IMG_20190519_2145382076248583.jpg");

startActivityForResult(getImagePicker().getIntent(MuseumActivity.this),
IpCons.RC_IMAGE_PICKER);



                    }
                });
                savebtn.setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        editbtn.setVisibility(View.VISIBLE);
                        savebtn.setVisibility(View.GONE);
                        favorbtn.setVisibility(View.VISIBLE);
                        etitle.setEnabled(false);
                        exhibitinfo.setEnabled(false);
                        exhibitsum.setEnabled(false);
                        exhibitphoto.setEnabled(false);
                        final String title =
etitle.getText().toString();
```

```
                        final String einfo =
exhibitinfo.getText().toString();
                        final String esum =
exhibitsum.getText().toString();
                        final String eroomid = exhibits[iii][5];
                        final String exhibitid = exhibits[iii][0];
                        final String photo = name;

                        final AlertDialog.Builder innerdialog = new
AlertDialog.Builder(MuseumActivity.this);
                        Response.Listener<String> responseListener = new
Response.Listener<String>() {
                            @Override
                            public void onResponse(String response) {

                                innerdialog.setTitle("Upload
Completed");
                                innerdialog.setMessage("An exhibit has
been successfully updated");
                                // Specifying a listener allows you to
take an action before dismissing the dialog.
                                // The dialog is automatically dismissed
when a dialog button is clicked.
                                innerdialog.setPositiveButton("Close",
new DialogInterface.OnClickListener() {
                                    public void onClick(DialogInterface
dialog, int which) {
                                        //
innerdialog.setCancelable(true);
                                    }
                                });
                                // A null listener allows the button to
dismiss the dialog and take no further action.


                                innerdialog.show();
                            }
                        };
                        UploadFileAsync ufa = new UploadFileAsync();
```

```
                        ufa.execute(path);
                        Exhibittodatabase etd = new
Exhibittodatabase(title, einfo, esum, eroomid, exhibitid, photo,
responseListener);
                        RequestQueue queue =
Volley.newRequestQueue(MuseumActivity.this);
                        queue.add(etd);
                    }
                });
            }
        });

        favorbtn.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                String icounter = String.valueOf(iii);

                addtomyfavpref(icounter, exhibits[iii][0], exhibits[iii]
[1], exhibits[iii][2], exhibits[iii][3], exhibits[iii][4]);
                Context context = getApplicationContext();
                CharSequence text = "Exhbiit added to favourites!";
                int duration = Toast.LENGTH_SHORT;

                Toast toast = Toast.makeText(context, text, duration);
                toast.show();

            }
        });
        close.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                alertDialog.cancel();
            }
        });
        alertDialog.show();
    }
}
```

ΠΑΡΑΡΤΗΜΑ Β’

Οδηγίες για εγκατάσταση των εφαρμογών τοπικά

Τα αρχεία μπορείτε να τα βρείτε στο Google drive λογαριασμό μου: Εδώ
Μέσα στο αρχείο που θα κατεβάσετε, θα βρείτε δύο καταλόγους:

- Κατάλογος server που περιέχει τα αρχεία που πρέπει να μπουν στην μεριά του εξυπηρετητή.
    - Θα πρέπει να παραμετροποιήσετε το αρχείο στον υποκατάλογο dbconfig και να χρησιμοποιήσετε τα δικά σας στοιχεία βάσης δεδομένων για να μπορούν τα αρχεία να συνδέονται με τη βάση δεδομένων.
    - Θα πρέπει να τρέξετε τις εντολές SQL που βρίσκονται στον υποκατάλογο SQLQueries για να δημιουργηθούν οι πίνακες στη βάση δεδομένων

- Κατάλογος mobile app που περιέχει το project που θα εισάγετε στο Android Studio. Το project κατά ένα μεγάλο ποσοστό δεν χρειάζεται παραμετροποίηση πέραν από τις παρακάτω περιπτώσεις:
    - Το αρχείο AdminRequest.java στη γραμμή 9
    - Το αρχείο Exhibittodatabase.java στη γραμμή 14
    - Το αρχείο MapRequest.java στη γραμμή 12
    - Το αρχείο MuseumActivity.java στις γραμμές 139,367, 572,658,972,1022

  Και τα τέσσερα αρχεία περιέχουν υπερσυνδέσμους ως προς την σελίδα που χρησιμοποίησα και θα πρέπει να τροποποιηθούν για να έχουν επικοινωνία με τον εξυπηρετητή σας.