Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής – Ανάπτυξη Λογισμικού και Τεχνητής Νοημοσύνης»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Σχεδιασμός και κατασκευή μιας εφαρμογής πελάτη-εξυπηρετητή για την καταχώριση μουσείων** **Design and implementation of a client-server application for museums' registration**
Ονοματεπώνυμο Φοιτητή	**Αλέξανδρος Ζαριδάκης**
Πατρώνυμο	**Κωνσταντίνος**
Αριθμός Μητρώου	**ΜΠΣΠ/ 18013**
Επιβλέπων	**Χρήστος Δουληγέρης, Καθηγητής**

Ημερομηνία Παράδοσης: Δεκέμβριος 2022

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Χρήστος Δουληγέρης Καθηγητής	Ρόζα Μαυροπόδη Ε.ΔΙ.Π	Δημήτριος Βεργάδος Καθηγητής

Ευχαριστίες

Ξεκινώντας, θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες στον κ. Δουληγέρη Χρήστο καθηγητή του τμήματος πληροφορικής στο Πανεπιστήμιο Πειραιώς για την ανάθεση της διπλωματικής εργασίας. Επίσης, θα ήθελα να ευχαριστήσω τον κ. Σαράντη Μητρόπουλο για την καθοδήγηση και την επίβλεψή του κατά την διάρκεια εκπόνησης της μεταπτυχιακής διατριβής. Τέλος, ένα μεγάλο ευχαριστώ στην οικογένεια μου καθώς και στους ανθρώπους που βρίσκονται δίπλα μου όλο αυτό το διάστημα, για την πολύτιμη στήριξή τους.

Περιεχόμενα

Ευχαριστίες ... 3
Κατάλογος Εικονών ... 3
Περίληψη ... 5
Abstract ... 5
1. Εισαγωγή – Σύντομη Περιγραφή ... 6
1.1 Συναφή Συστήματα και Σύγκριση με την Προσέγγισή μας ... 6
2. Απαιτήσεις και Τεχνολογίες Συστήματος ... 10
2.1 Περίληψη Κεφαλαίου ... 10
2.2 Απαιτήσεις Συστήματος ... 10
2.3 Spring Boot Framework ... 10
2.4 Dependency Injection ... 11
2.5 Spring Boot Beans και Inversion of Control ... 11
2.6 IntelliJ IDEA and Visual Studio Code ... 11
2.7 PostgreSQL ... 12
2.8 Hibernate ... 12
2.9 REST API ... 12
2.10 Project Lombok ... 14
2.11 Maven ... 15
2.12 Angular Material and Bootstrap ... 15
3. Υλοποίηση και Σχεδιασμός Συστήματος ... 16
3.1 Περίληψη Κεφαλαίου ... 16
3.2 Σχεδιασμός και λειτουργίες στο API ... 16
3.2.1 Σχεδιασμός Αρχιτεκτονικής Spring Boot ... 16
3.2.2 Σχεδιασμός Domain Επιπέδου (Domain Layer) ... 20
3.2.3 Σχεδιασμός Επιπέδου Service (Service Layer) ... 27
3.2.4 Σχεδιασμός Επιπέδου Persistence (Repositories) ... 33
3.2.5 Συνολική Αρχιτεκτονική του API ... 35
3.2.6 Διαδικτυακό τεκμήριο JSON - JWT (JSON Web Token) ... 37
3.2.7 Αυθεντικοποίηση και Εξουσιοδότηση (Authentication and Authorization) ... 39
3.2.8 Ρόλοι Χρηστών, Εξαιρέσεις(Exceptions), DTOs ... 47
3.3 Σχεδιασμός και λειτουργίες στο Front End ... 53
3.3.1 Σχεδιασμός των Components ... 55
3.3.2 Σχεδιασμός των Services ... 56
3.3.3 AuthInterceptor και AuthenticationGuard ... 58

4 Εκτέλεση Συστήματος, Περιπτώσεις Χρήσης 60
4.1 Περίληψη Κεφαλαίου 60
4.2 Εκτέλεση 60
4.3 Παραδείγματα και περιγραφή χρήσης συστήματος 60
4.3.1 Μη Συνδεδεμένος Χρήστης 61
4.3.2 Εγγραφή Σύνδεση Απλού Χρήστη στο Σύστημα 65
4.3.3 Σύνδεση Διαχειριστή(Admin) Χρήστη στο σύστημα 71
4.3.4 Σύνδεση Χρήστη Διαχειριστή Μουσείου στο Σύστημα 76
5 Αξιολόγηση, Συμπεράσματα και Πιθανές Επεκτάσεις 82
5.1 Αξιολόγηση 82
5.2 Συμπεράσματα 84
5.3 Πιθανές επεκτάσεις 85
Βιβλιογραφία 86
ΠΑΡΑΡΤΗΜΑ Α - ΕΡΩΤΗΜΑΤΟΛΟΓΙΟ 88
Πρόλογος 88
Ενότητα Α | Γενικά στοιχεία 88
Ενότητα Β | Οδηγίες για την πλοήγηση στην ιστοσελίδα 89
Σενάριο 1 89
Σενάριο 2 89
Σενάριο 3 89
Σενάριο 4 90
Σενάριο 5 90
Σενάριο 6 91
Σενάριο 7 91
Σενάριο 8 91
Σενάριο 9 92
Σενάριο 10 92
Σενάριο 11 92
Ενότητα Γ | Ερωτήσεις Αξιολόγησης της Ιστοσελίδας 93
Ενότητα Δ | Σχόλια και Παρατηρήσεις 94
ΠΑΡΑΡΤΗΜΑ Β – ΔΕΛΤΙΟ ΑΞΙΟΛΟΓΗΣΗΣ 95
ΠΑΡΑΡΤΗΜΑ Γ – ΔΕΔΟΜΕΝΑ 99

Κατάλογος Εικονών

Εικόνα 1 – Αρχιτεκτονική REST API (Mamun 2019) 13
Εικόνα 2 - Επικοινωνία του front-end με το back-end 14
Εικόνα 3 - Επικοινωνία μεταξύ controller, service και repository 16
Εικόνα 4 - Επίπεδα επικοινωνίας (Stavast 2018) 17
Εικόνα 5 - Επικοινωνία με την βάση (Rao and Swamy 2020) 18
Εικόνα 6 - Επικοινωνία του presentation layer με χρήση DTO (kexugit n.d.) 19
Εικόνα 7 - Μετατροπή αντικειμένου JSON σε DTO 20
Εικόνα 8 - Entity domain classes 20
Εικόνα 9 - ImageModel class 21
Εικόνα 10 - Users class 22
Εικόνα 11 - Users class 23
Εικόνα 12 - Museum class 25
Εικόνα 13 - Museum class 26
Εικόνα 14 - Υλοποίηση του UserService Interface 28
Εικόνα 15 - MuseumService 29
Εικόνα 16 - UserService 30
Εικόνα 17 - UserService 31
Εικόνα 18 - Μετατροπή κλάσης User Entity σε DTO 32
Εικόνα 19 - Μετατροπή Museum Entity σε DTO 33
Εικόνα 20 – Έγχυση εξαρτήσεων των Repositories (‘Spring Boot Architecture and Its Workflow’ n.d.) 34
Εικόνα 21 - Users Repository 34
Εικόνα 22 - Κύκλος ροής δεδομένων στο back-end (‘Spring Boot Architecture and Its Workflow’ n.d.) 35
Εικόνα 23 - Επικοινωνία όλων των επιπέδων (‘Spring Boot Architecture and Its Workflow’ n.d.) 37
Εικόνα 24 – Αυθεντικοποίηση(Authentication) 40
Εικόνα 25 – Διαχειριστής για την Αυθεντικοποίηση (Authentication Manager) 40
Εικόνα 26 - Δημιουργία και ανταλλαγή JWT (‘[Spring Security] - Spring Boot Security JWT Authentication’ 2021) 41
Εικόνα 27 - Υλοποίηση JWT 42
Εικόνα 28 - Authentication και Authorization του χρήστη 43
Εικόνα 29 - JWT verifier 44
Εικόνα 30 – Φίλτρα των αιτημάτων για JWT (Filter των requests για JWT) 45
Εικόνα 31 - Επικοινωνία με αυθεντικοποιημένο χρήστη (‘JWT Authentication with Spring Boot Resource Server | by Imesha Sudasingha | The Startup’ n.d.) 46
Εικόνα 32 - Παράδειγμα χρήστης του annotation PreAuthorize 47
Εικόνα 33 - Οι ρόλοι του συστήματος στο Role enum 48
Εικόνα 34 - Η κλάση που περιέχει τα Authorities 49
Εικόνα 35 - Τα εξατομικευμένα Exception Classes της Εφαρμογής 50
Εικόνα 36 - Δομή πακέτων για τα Data Transfer Objects 51
Εικόνα 37 - Οι σχεσιακοί πίνακες της Βάσης Δεδομένων 52
Εικόνα 38 - Διάγραμμα επικοινωνίας των modules της Angular (TekTutorialsHub 2018) 53

Εικόνα 39 - Διάγραμμα κύκλου ζωής Angular (‘Angular Data Flow Architecture - Web Development with MongoDB and Node - Third Edition [Book]’ n.d.) 54
Εικόνα 40 - Η δομή των components στην εφαρμογή 55
Εικόνα 41 - Διάγραμμα ροής επικοινωνίας μεταξύ observer και observable (‘JavaScript Observer (Publish/Subscribe) Pattern’ n.d.) 56
Εικόνα 42 - Τα common Services του συστήματος 57
Εικόνα 43 - Ανταλλαγή HTTP αιτημάτων με JWT αναχαίτηση(introspection) (‘Angular 7 + Spring Boot JWT Authentication | JavaInUse’ n.d.) 58
Εικόνα 44 - Αρχιτεκτονική επικοινωνίας Angular με Spring Boot (‘Angular + Spring Boot + MySQL CRUD Example’ n.d.) 59
Εικόνα 45 - Η μορφή των json που αποθηκεύονται τα translation keys-values 61
Εικόνα 46 - Αρχική οθόνη στο σύστημα 62
Εικόνα 47 - Αρχική οθόνη, νέες προσθήκες μουσείων 62
Εικόνα 48 – Σύστημα σύνδεσης Διάγραμμα Καταστάσεων(Login System User Case Diagram) (Nym 2022a) 63
Εικόνα 49 - Φόρμα σύνδεσης στο σύστημα 64
Εικόνα 50 - Φόρμα εγγραφής στο σύστημα 65
Εικόνα 51 - Η λίστα με τα διαθέσιμα μουσεία 65
Εικόνα 52 - Σελίδα προφίλ χρήστη 66
Εικόνα 53 - Σελίδα Μουσείου με δυνατότητα αιτήματος για περιήγηση και προσθήκη στα αγαπημένα 67
Εικόνα 54 - Εμφάνιση των αγαπημένων μουσείων στο προφίλ του χρήστη και επιλογή αφαίρεσης από τα αγαπημένα 68
Εικόνα 55 - Μήνυμα ενημέρωσης του χρήστη πως το μουσείο που προσπάθησε να προσθέσει στην λίστα αγαπημένων υπάρχει ήδη 69
Εικόνα 56 - Απεικόνιση λίστας αγαπημένων στο προφίλ συνδεδεμένου χρήστη 70
Εικόνα 57 - Ενημέρωση στοιχείων με Email που χρησιμοποιείται από άλλο χρήστη και ανταπόκριση συστήματος 71
Εικόνα 58 - Προφίλ Admin User με δυνατότητα δημιουργίας μουσείου 72
Εικόνα 59 - Λίστα χρηστών εφαρμογής με δυνατότητα διαγραφής χρήστη 73
Εικόνα 60 - Λίστα χρηστών εφαρμογής με εφαρμοσμένο το φίλτρο αναζήτησης 74
Εικόνα 61 - Modal που εμφανίζεται στον super Admin του συστήματος και δίνει την δυνατότητα αλλαγής στοιχείων άλλου εγγεγραμμένου χρήστη 75
Εικόνα 62 - Τρόπος διαγραφής μουσείου από το σύστημα και τον super Admin 76
Εικόνα 63 - Παράδειγμα κειμένου στην φόρμα εισαγωγής νέου μουσείου 77
Εικόνα 64 - Παράδειγμα φόρμας καταχώρησης και δημιουργίας μουσείου 78
Εικόνα 65 - Παράδειγμα προεπισκόπησης εικόνων κατά την δημιουργία μουσείου. ... 79
Εικόνα 66 - Μόνο το μουσείο που ανήκει στον συνδεδεμένο museum-admin έχει το εικονίδιο επεξεργασίας 80
Εικόνα 67 - Προβολή αιτημάτων ξενάγησης στο προφίλ του διαχειριστή μουσείου 81
Εικόνα 68 - Διάγραμμα απεικόνισης χρηστών βάσει ηλικίας 82
Εικόνα 69 - Διάγραμμα απεικόνισης βάσει φύλου 83
Εικόνα 70 - Διάγραμμα απεικόνισης βάσει εκπαίδευσης 83
Εικόνα 71 - Διάγραμμα απεικόνισης βάσει χρήσης internet 84

Περίληψη

Τα τελευταία χρόνια, παρατηρείται ταχεία ανάπτυξη του ψηφιακού περιβάλλοντος. Η πανδημία σίγουρα έπαιξε το ρόλο της και έτσι έχουμε οδηγηθεί στην ψηφιακή εποχή, όπου πλέον και η πιο μικρή επιχείρηση διατηρεί ιστοσελίδα. Η ανάγκη αυτή, σε συνδυασμό με την ανάγκη του κόσμου για πολιτιστικά δρώμενα οδήγησε σε εξατομικευμένες σελίδες μουσείων με δυνατότητες ακόμα και ζωντανής περιήγησης στις εκθέσεις και τους χώρους των μουσείων. Αντικείμενο λοιπόν, της παρούσας διατριβής είναι η υλοποίηση και ο σχεδιασμός ενός συστήματος για την αναζήτηση και την προβολή μουσείων.

Πιο συγκεκριμένα, η διαδικτυακή εφαρμογή που δημιουργήθηκε δίνει την δυνατότητα δημιουργίας προφίλ διαχειριστή όλης της εφαρμογής (super-admin), διαχειριστή του μουσείου (admin) και προφίλ χρήστη. Ο διαχειριστής της εφαρμογής (super-admin) ο οποίος θα μπορεί να διαχειρίζεται τους διαχειριστές των μουσείων και τα δικαιώματά τους καθώς και να διαγράψει κάποιο μουσείο από την σελίδα. Ο διαχειριστής του μουσείου μπορεί να διαχειρίζεται το σχετικό μουσείο και τις πληροφορίες του. Τέλος, υπάρχει το προφίλ χρήστη, ο οποίος θα μπορεί να βλέπει τα μουσεία και να αιτηθεί περιήγηση.

Για την υλοποίηση της εφαρμογής χρησιμοποιήθηκε το Spring Boot Framework σε συνδυασμό με Angular, ενώ για την βάση χρησιμοποιήθηκε η PostgreSQL και το Hibernate Framework για την χαρτογράφηση και την διασύνδεση της εφαρμογής με την βάση.

Abstract

In the recent years, there has been a rapid development of the digital environment. The pandemic certainly played its role and thus we have been led to the digital age, where now even the smallest business maintains a website. This need, combined with the world's need for cultural events, led to personalized museum pages with the possibility of even a live tour of the exhibitions and museum spaces.

The focus of this thesis is the implementation and design of a system for searching and viewing museums. More specifically, the web application that was created allows the creation of an administrator profile of the entire application (super-admin), an administrator of the museum (admin) and a user profile. The administrator of the application (super-admin), who will be able to manage museum administrators and their rights, as well as delete a museum from the page. The museum administrator can manage the relevant museum and its information. Finally, there is the user profile, who will be able to see the museums and request a tour.

For the implementation of the application, the Spring Boot Framework was used in combination with Angular, while for the database, PostgreSQL and the Hibernate Framework were used to map and interface the application with the database.

1. Εισαγωγή – Σύντομη Περιγραφή

Τα μουσεία για πολλά χρόνια αποτελούν έναν από τους σημαντικότερους πολιτιστικούς πόρους κάθε χώρας. Περιέχουν τα δημιουργήματα ενός λαού, που αποτελούν δομικά στοιχεία του πολιτισμού και της ανάπτυξής του μέσα στο χρόνο. Ιδιαίτερα, η Ελλάδα, έχοντας πλούσιο πολιτιστικό απόθεμα του απώτερου και πρόσφατου παρελθόντος της, μέσα από τα μουσεία συγκεντρώνει πολιτιστικά επιτεύγματα, τα οποία στη συνέχεια είναι διαθέσιμα για έκθεση. Τα πολιτιστικά επιτεύγματα αυτά παρουσιάζουν την ιστορική πορεία της περιοχής που αναφέρονται, και γίνονται σημεία θαυμασμού για το κοινό.

Επιπλέον, με την εμφάνιση της πανδημίας του Covid-19, λόγω των συνεχόμενων μέτρων ώστε να εξασφαλιστεί η δημόσια υγεία, τα μουσεία ανέπτυξαν ψηφιακά περιβάλλοντα ώστε να διατηρηθεί η επικοινωνία με το αποκλεισμένο κοινό. Ο ψηφιακός μετασχηματισμός των πολιτιστικών οργανισμών περιλαμβάνει μια εξατομικευμένη σελίδα του κάθε μουσείου, που παρέχει όλες τις απαραίτητες πληροφορίες για τα εκθέματα του μουσείου. Πιο συγκεκριμένα, στις εποχές όπου αναφέρονται, το είδος των εκθεμάτων όπως αρχαιολογικά, νομισματικά, λαογραφικά αλλά και πληροφορίες για πρόσβαση, επικοινωνία ωράριο λειτουργείας του μουσείου.

Αντικείμενο της παρούσας μεταπτυχιακής διατριβής είναι η μελέτη, σχεδίαση και υλοποίηση ενός συστήματος αναζήτησης και προβολής μουσείων ανά περιφέρεια. Μέσω του συστήματος θα δίνεται η δυνατότητα στο χρήστη, να αναζητήσει μουσεία, να αναζητήσει πληροφορίες για τα εκθέματα, το κόστος την διάρκεια ξενάγησης, τον τρόπο πρόσβασης καθώς και αίτημα για ξενάγηση στο μουσείο της επιλογής του. Στα πλαίσια της διατριβής η περιφέρεια που επιλέχθηκε για παρουσίαση των μουσείων της είναι η περιφέρεια Ιονίων Νήσων.

Η παρούσα εργασία χωρίζεται σε 5 κεφάλαια με το πρώτο να αποτελεί την εισαγωγή. Το δεύτερο κεφάλαιο αναλύει τις απαιτήσεις συστήματος, και εξηγεί την αρχιτεκτονική καθώς και τις τεχνολογίες και εργαλεία που χρησιμοποιήθηκαν για την αποπεράτωσή του. Στο τρίτο κεφάλαιο παρουσιάζονται στοιχεία της υλοποίησης με παραδείγματα κώδικα αλλά και με διαγράμματα των κλάσεων. Στο τέταρτο κεφάλαιο περιέχονται περιπτώσεις χρήσης του συστήματος με παραδείγματα εκτέλεσης του προγράμματος. Ακόμα, αναλύεται και περιγράφεται πως διαφοροποιούνται οι ρόλοι των χρηστών που υπάρχουν στο σύστημα. Τέλος, στο πέμπτο και τελευταίο κεφάλαιο της μεταπτυχιακής διατριβής παρουσιάζονται τα τελικά συμπεράσματα, η αξιολόγηση και τα αποτελέσματά της αλλά και πιθανές μελλοντικές επεκτάσεις που θα μπορούσαν να ενσωματωθούν στο σύστημα με στόχο την βελτίωσή του.

1.1 Συναφή Συστήματα και Σύγκριση με την Προσέγγισή μας

Στο κεφάλαιο αυτό, θα αναφερθούν και θα αναλυθούν συναφή συστήματα που προσεγγίζουν το πρόβλημα και το ζητούμενο που είναι το θέμα της παρούσας διατριβής. Επίσης, θα αναλυθούν τα πλεονεκτήματα αλλά και πως υπερτερεί το σύστημα που θα παρουσιαστεί στα πλαίσια της συγκεκριμένης διατριβής. Αρχικά, όπως έχει γίνει κατανοητό το θέμα και το αντικείμενο μελέτης είναι η προβολή και η αύξηση της διαθεσιμότητας των μουσείων προς το κοινό. Επίσης, η διευκόλυνση της επικοινωνίας και αλληλεπίδρασης μεταξύ των μουσείων και των πολιτών, αλλά και η μείωση των οικονομικών πόρων που χρειάζονται για την διαχείριση των μουσείων. Η μείωση των οικονομικών δαπανών στα μουσεία διαδραματίζει πολύ σημαντικό ρόλο καθώς όπως έδειξε η έρευνα που πραγματοποιήθηκε από το Network of European Museum Organisations (‘NEMO_Corona_Survey_Results_6_4_20.Pdf’ n.d.) το 92% των μουσείων κατά την διάρκεια της πανδημίας έκλεισαν και αυτό είχε ως αποτέλεσμα να χάσουν όλα τους τα έσοδα. Επιπλέον, από το ποσοστό των μουσείων τα οποία έμειναν ανοιχτά, και λειτουργούσαν μερικώς και με περιορισμένο αριθμό επισκεπτών τα στοιχεία που παρουσίασαν ήταν επίσης αποθαρρυντικά. Συγκεκριμένα το 30% αυτών των μουσείων δήλωσε πως χάνουν 1000€ ανά εβδομάδα και το 25% χάνουν 5.000€ επίσης ανά εβδομάδα.

Επιπλέον, δεδομένης της κατάστασης πραγματοποιήθηκε επίσης μία έρευνα για το τι δράσεις, πρωτοβουλίες αλλά και ποιες ενέργειες θα μπορούσαν να λάβουν χώρα από τα μουσεία έτσι ώστε να βρεθούν εναλλακτικοί τρόποι ώστε να έχουν ροή οικονομικών πόρων μέσα στην κρίση (‘Initiatives_of_museums_in_times_of_corona_4_20.Pdf’ n.d.). Τα αποτελέσματα έδειξαν πως ο ψηφιακός μετασχηματισμός αλλά και η έκθεση των μουσείων στο διαδίκτυο με πολύπλευρους τρόπους μπορούν να βοηθήσουν την κατάσταση.

Με βάση τα παραπάνω, γίνεται κατανοητό πως έχει δημιουργηθεί η ανάγκη για την προβολή και την έκθεση των μουσείων στο διαδίκτυο ώστε οι χρήστες να μπορούν να βρουν πιο εύκολα πληροφορίες για αυτά. Πιο συγκεκριμένα, είναι αρκετά χρήσιμο και διαδεδομένο την σημερινή εποχή να υπάρχει για κάθε προϊόν μία πλατφόρμα που να τα κατηγοριοποιεί και να τα ομαδοποιεί όλα μαζί. Είναι πιο χρηστικό, πιο εύκολο και γλυτώνει χρόνο στον χρήστη η αναζήτηση σε μία πλατφόρμα που έχει όλα τα προϊόντα μαζεμένα σε σύγκριση με το να ανοίγει πολλές διαφορετικές ιστοσελίδες για κάθε ένα προϊόν ξεχωριστά. Όπως γίνεται κατανοητό, στα πλαίσια αυτής της διατριβής το μουσείο είναι το προϊόν που αναφέρθηκε. Υπάρχουν λοιπόν, δύο πλατφόρμες που εφαρμόζουν εν μέρει το επιχειρησιακό μοντέλο που θα παρουσιαστεί στα πλαίσια αυτής της διατριβής.

Αρχικά, υπάρχει το «Δίκτυο Μουσείων και Πολιτιστικών Φορέων Αθηνών»(http://www.athensmuseums.net/index.php) και το «Δίκτυο Μουσείων Ιονίων Νήσων»(https://dimin.museum.ionio.gr/). Και οι δύο πλατφόρμες αυτές, δεν αναφέρονται σε ένα συγκεκριμένο μουσείο αλλά έχουν ως περιεχόμενο πολλά διαφορετικά μουσεία. Πιο συγκεκριμένα, η πρώτη πλατφόρμα αναφέρεται σε μουσεία που βρίσκονται εντός Αττικής και αντίστοιχα, η δεύτερη απευθύνεται και περιέχει κάποια μουσεία από τα Ιόνια Νησιά. Η πρώτη πλατφόρμα που αναφέρεται και περιλαμβάνει τα μουσεία της Αθήνας, αν και η λογική της είναι κοντά στην προσέγγιση που θα υλοποιηθεί στα πλαίσια αυτής της διατριβής, η υλοποίηση στερείται από πολλά πράγματα. Ξεκινώντας, η πλατφόρμα αποτελείται από παλιό και μη ενημερωμένο UI. Επίσης, δεν περιλαμβάνει τρόπο αναζήτησης μουσείων και αυξάνει την δυσκολία εύρεσης μουσείων και των πληροφοριών τους. Ακόμα, οι φωτογραφίες των μουσείων αλλά και οι πληροφορίες για αυτά είναι αρκετά περιορισμένες. Πιο συγκεκριμένα, το Δίκτυο Μουσείων Ιονίων Νήσων παρέχει μία πλατφόρμα που έχει ως στόχο συγκεκριμένα την ομαδοποίηση μουσείων στο σύμπλεγμα των Ιονίων Νήσων. Αποτελείται από μία αρχική οθόνη, που παρουσιάζει τα μουσεία που συμμετέχουν. Σε κάθε ένα από τα μουσεία μπορείς να ανοίξεις μία σελίδα που περιέχει τις απαραίτητες πληροφορίες του μουσείου. Δεν δίνεται η δυνατότητα για αναζήτηση συγκεκριμένου μουσείου, όπως επίσης δεν προβλέπεται δημιουργία προφίλ χρηστών στο σύστημα. Και τα δύο συστήματα τα οποία αναλύουμε είναι υλοποιημένα σε δύο γλώσσες, Αγγλικά και Ελληνικά.

Στη συνέχεια, θα γίνει περιγραφή της επιχειρησιακής λογικής που υλοποιείται στα πλαίσια της παρούσας διατριβής και θα παρατεθούν τα πλεονεκτήματα κατά τα οποία το σύστημα πλεονεκτεί και επεκτείνει τις λειτουργικότητες των δύο αναφερόμενων συστημάτων. Ο στόχος είναι η δημιουργία ενός συστήματος το οποίο θα έχει επεκτασιμότητα και θα μπορεί να λειτουργεί μακροπρόθεσμα ανεξάρτητα από τον όγκο των μουσείων και των χρηστών που το χρησιμοποιούν, καθώς θα είναι εφικτό να γίνει στο σύστημα καταχώρηση μουσείων από όλη την χώρα οπότε και αναμένεται ο αριθμός να είναι μεγάλος. Δηλαδή, να αποκτήσει να βασικά χαρακτηριστικά ενός Πληροφοριακού Συστήματος και συγκεκριμένα για τις ανάγκες της παρούσας υλοποίησης ένα Πληροφοριακό Σύστημα Βασισμένο στο Διαδίκτυο (‘Web Based Information Systems’ n.d.). Θα γίνει αναφορά σε κάποια από τα χαρακτηριστικά που πρέπει να υπάρχουν σε αυτό. Ένα από τα χαρακτηριστικά είναι η πλατφόρμα να ανοίγει και να ανταποκρίνεται σωστά σε όλους τους περιηγητές διαδικτύου ανεξάρτητα από το λειτουργικό σύστημα του υπολογιστή(cross platform compatibility) και φυσικά να είναι responsive και να προσαρμόζεται σε διαφορετικά μεγέθη οθονών όπως για παράδειγμα σε tablet ή κινητές συσκευές. Επίσης, ένα ακόμα σημαντικό χαρακτηριστικό είναι οι χρήστες να μπορούν να γίνουν μέλη σε μία πλατφόρμα και να διαχειρίζονται και να αποθηκεύουν σε αυτή τα στοιχεία που επιθυμούν. Ακόμα, για τους σκοπούς διαχείρισης, είναι σημαντικό επίσης να γίνεται δυναμική προβολή δεδομένων όπως πόσοι ενεργοί χρήστες υπάρχουν στο σύστημα ή πόσα καταχωρημένα μουσεία. Τα στατιστικά αυτά, θα διευκολύνουν τις αποφάσεις που πρέπει να

παρθούν από τους διαχειριστές της πλατφόρμας όσον αφορά το προσωπικό που πρέπει να απασχολούν. Ακόμα, τα στατιστικά αυτά θα μπορούν να χρησιμοποιηθούν και από τους τεχνικούς του συστήματος για να αποφασίζουν αν το υλικό που έχει χρησιμοποιηθεί για την πλατφόρμα μπορεί να ανταποκριθεί στις ανάγκες του συστήματος ή χρειάζεται κάποια αναβάθμιση υλικού.

Όπως αναφέρθηκε είναι σημαντική η ύπαρξη της διαδικτυακής κοινότητας καθώς η δυνατότητα ο χρήστης να μπορεί να δημιουργήσει λογαριασμό και να τον χρησιμοποιεί για να κάνει σύνδεση ανεβάζει για τον ίδιο την αξιοπιστία και την λειτουργικότητα του συστήματος. Επίσης, από την πλευρά της πλατφόρμας είναι σημαντικό να έχει εγγεγραμμένους χρήστες καθώς μπορεί να χρησιμοποιήσει τα δεδομένα αυτά για εξαγωγή επιχειρηματικών και τεχνικών συμπερασμάτων. Επιπλέον, η ύπαρξη λίστας αγαπημένων είναι αρκετά σημαντική (‘Why Are ECommerce Wish Lists & Favorites Important?’ 2020) καθώς ο χρήστης μπορεί να βρίσκει και να έχει εύκολα πρόσβαση στα μουσεία που τον ενδιαφέρουν μέσα στην πλατφόρμα. Επίσης, βοηθάει και κάνει πιο ακριβής την εξαγωγή στατιστικών συμπερασμάτων για τον διαχειριστή ενός συγκεκριμένου μουσείου. Για να εξηγηθεί πως θα μπορούσε να γίνει αυτό, είναι χρήσιμο να αναφερθεί πως στο σύστημα οι συνδεδεμένοι χρήστες μπορούν να κάνουν αίτημα ξενάγησης στον διαχειριστή του μουσείου. Οπότε, επιστρέφοντας στην εξαγωγή επιχειρηματικών συμπερασμάτων, αν ένα μουσείο είναι στην λίστα αγαπημένων πολλών χρηστών και ταυτόχρονα στο μουσείο δεν έχει αποσταλθεί κάποιο αίτημα για ξενάγηση, ο διαχειριστής του μουσείου θα μπορούσε να συμπεράνει για παράδειγμα πως το κόστος ξενάγησης είναι πολύ υψηλό.

Σαν επιπλέον λειτουργικότητα ώστε να υποστηρίζεται η επεκτασιμότητα του συστήματος είναι ο τρόπος αναζήτησης μουσείων μέσα στο σύστημα. Όπως αναφέρθηκε, αν έχουν καταχωρηθεί μουσεία από όλη την χώρα ο αριθμός θα είναι αρκετά μεγάλος. Για τον λόγο αυτό, κρίθηκε σκόπιμο ο χρήστης να μπορεί να αναζητήσει μουσεία και με βάση την πόλη στην οποία βρίσκονται. Φυσικά, θα μπορεί να αναζητήσει και με τον τίτλο ή το όνομα του μουσείου, το email του μουσείου και στη περίπτωση που το γνωρίζει ακόμα και με το τηλέφωνο του. Είναι σαφές πως προσφέρονται ικανοποιητικοί τρόποι αναζήτησης και ο στόχος είναι να μπορεί ο χρήστης να βρει όσο πιο εύκολα και γρήγορα το μουσείο που επιθυμεί. Ακόμα, στα πλαίσια του Web Based Information System κρίθηκε σκόπιμο ο φόρτος εργασίας για την διαχείριση των μουσείων να γίνει κατανεμημένος. Πιο συγκεκριμένα, τα μουσεία θα δημιουργούνται και θα διαχειρίζονται από έναν χρήστη ο οποίος μέσα στην πλατφόρμα θα είναι υπεύθυνος για ενημέρωση των στοιχείων των μουσείων, να ελέγχει και να ανταποκρίνεται στα αιτήματα για ξενάγηση στο μουσείο που θα πραγματοποιούνται από εγγεγραμμένους χρήστες της πλατφόρμας. Η επιλογή αυτή πάρθηκε καθώς ο στόχος είναι η πλατφόρμα να μην χρειάζεται να απασχολεί μεγάλο αριθμό προσωπικού και να είναι αποκεντροποιημένος ο τρόπος λειτουργίας της. Αυτό σημαίνει πως για παράδειγμα, αν θέλει κάποιο μουσείο από μία απομακρυσμένη περιοχή να καταχωρηθεί και να γίνει διαθέσιμο στο σύστημα, ο υπεύθυνος του μουσείου πρέπει απλά να κάνει εγγραφή στο σύστημα, και κατόπιν επικοινωνίας με τον διαχειριστή της πλατφόρμας να του δοθεί άδεια δημιουργίας μουσείου. Αφού ολοκληρωθεί η ενέργεια αυτή, ύστερα η δημιουργία και η καταχώρηση των στοιχείων του μουσείου στη πλατφόρμα είναι δικιά του αρμοδιότητα. Όπως είναι σαφές, ο χρήστης που διαχειρίζεται την πλατφόρμα, γλυτώνει αρκετό χρόνο καθώς απλώς δίνει άδεια δημιουργίας μουσείου στον εγγεγραμμένο χρήστη αντί να κάνει συλλογή και καταχώρηση στοιχείων για το μουσείο.

Φυσικά, υπάρχει δυνατότητα διαγραφής μουσείου ή χρήστη από την πλατφόρμα σε περίπτωση που τα δεδομένα που εισάγει δεν συμβαδίζουν με την πολιτική χρήσης που θα έχει οριστεί. Το σύστημα είναι υλοποιημένο σε δύο γλώσσες, Ελληνικά και Αγγλικά. Καθώς πρόκειται για πολιτισμικό περιεχόμενο, κρίθηκε χρήσιμο η εισαγωγή μιας καινούργιας γλώσσας στο σύστημα να μπορεί να γίνει σχετικά εύκολα. Για τον λόγο αυτό, ο τρόπος υλοποίησης έγινε με την χρήση εξωτερικών αρχείων τύπου key-value. Το key αποτελεί ένα μοναδικό κλειδί στην πλατφόρμα και το value είναι η μετάφραση στη γλώσσα που αναφέρεται το αρχείο. Συγκεκριμένα στο σύστημα όπως αναφέρθηκε έχουμε δύο αρχεία gr.json kai en.json για Ελληνικά και Αγγλικά αντίστοιχα. Το περιεχόμενό τους για παράδειγμα είναι “museum.portal.welcome”:”Καλώς ήρθατε” “museum.portal.welcome”:”Welcome”. Όπως γίνεται αντιληπτό, για την προσθήκη μίας επιπλέον γλώσσας αρκεί να δημιουργηθεί ένα ακόμα αρχείο μεταφρασμένο στην γλώσσα που επιθυμούμε και να εισαχτεί στο σύστημα.

Τέλος, θα αναφερθούν συνοπτικά τα επιπλέον χαρακτηριστικά-πλεονεκτήματα που θα περιέχει η πλατφόρμα στα πλαίσια της παρούσας διατριβής σε σχέση με τα συναφή συστήματα που έγιναν περιγραφή.

- Δυνατότητα δημιουργίας χρηστών και δημιουργία λίστες αγαπημένων μουσείων
- Οι εγγεγραμμένοι χρήστες μπορούν να πραγματοποιούν αιτήματα για ξενάγηση σε μουσεία
- Διαχειριστές των μουσείων θα είναι διαφορετικοί χρήστες με σκοπό η κατανομή φόρτου εργασίας για καταχώρηση και συντήρηση των μουσείων να είναι αποκεντροποιημένη
- Η αναζήτηση μουσείων στοχεύει σε πολλαπλές παραμέτρους όπως όνομα ή τίτλο μουσείου, πόλη στην οποία βρίσκεται, email, τηλέφωνο
- Εύκολος τρόπος εισαγωγής επιπλέον γλωσσών στο σύστημα

Με βάση την ανάλυση που προηγήθηκε, η παρούσα διατριβή παρέχει μία πληθώρα από πλεονεκτήματα και επεκτάσεις σε σχέση με τα συναφή συστήματα που υπάρχουν για την ίδια λειτουργικότητα για τα οποία ακολουθεί αναλυτική περιγραφή στα επόμενα κεφάλαια.

2. Απαιτήσεις και Τεχνολογίες Συστήματος

2.1 Περίληψη Κεφαλαίου

Στο κεφάλαιο αυτό αναλύονται οι απαιτήσεις του συστήματος, οι τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν για την υλοποίησή του. Αρχικά, θα παρουσιαστούν σύντομα οι απαιτήσεις του συστήματος. Έπειτα, θα παρουσιαστούν τα εργαλεία ανάπτυξης λογισμικού IntelliJ και Visual Studio Code. Θα παρουσιαστούν επίσης τα Spring Boot Framework και η Angular καθώς είναι τα 2 κύρια εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη του συστήματος. Θα γίνει αναφορά και περιγραφή των βιβλιοθηκών που χρησιμοποιήθηκαν στο σύστημα στο front-end κομμάτι καθώς και στο back-end. Οι βιβλιοθήκες αυτές, είναι λογισμικά ανοιχτού κώδικα και μπορούν να χρησιμοποιούνται στις εφαρμογές. Τέλος, η PostgreSQL η οποία χρησιμοποιήθηκε επίσης ως βάση δεδομένων για τις ανάγκες τις εφαρμογής και το FrameWork Hibernate για την χαρτογράφηση και διασύνδεση της εφαρμογής με την βάση δεδομένων.

2.2 Απαιτήσεις Συστήματος

Οι απαιτήσεις συστήματος καθορίστηκαν από τις λειτουργίες που πρέπει να παρέχει το σύστημα στους χρήστες του. Πιο συγκεκριμένα, στη συγκεκριμένη διπλωματική εργασία παρουσιάζεται μία web εφαρμογή με σκοπό να περιλαμβάνει συγκεντρωμένα τα μουσεία έτσι ώστε ο χρήστης να μπορεί να αναζητήσει πληροφορίες για αυτά όπως το περιεχόμενο τους, τα εκθέματά τους, ώρες λειτουργίας και η τοποθεσία τους, εύκολα και γρήγορα. Η εφαρμογή παρέχει την δυνατότητα στο χρήστη να κάνει εγγραφή και να δημιουργεί προφίλ στο σύστημα. Αυτό γίνεται έτσι ώστε, να μπορεί να κάνει αίτηση σε κάποιο μουσείο ώστε να κλείσει κάποιο ραντεβού και να γίνει περιήγηση από το προσωπικό του μουσείου. Φυσικά, μελλοντικά για το σύστημα μπορεί να γίνει επέκταση των λειτουργιών που έχει ένας απλός χρήστης μέσα σε αυτό αλλά και να προστεθούν διαφορετικά είδη χρηστών.

Επίσης, υπάρχουν οι χρήστες που διαχειρίζονται ή ακόμα προσθέτουν μουσεία. Για να γίνει εφικτή μια τέτοια λειτουργικότητα, όπως θα αναλυθεί περισσότερο παρακάτω έχουν δημιουργηθεί διαφορετικά ήδη χρηστών στο σύστημα με διαφορετικά δικαιώματα και σελίδες πρόσβασης ανάλογα με το ρόλο που έχει ο χρήστης. Τέλος, υπάρχει και ο χρήστης με όλα τα δικαιώματα χρήσης super admin, ο οποίος έχει δυνατότητα να διαγράψει κάποιον χρήστη, να δώσει δικαίωμα σε κάποιον χρήστη να μπορεί να προσθέτει κάποιο μουσείο και να επεξεργάζεται τα στοιχεία του.

Με βάση τα παραπάνω, το σύστημα χρειάζεται να αποθηκεύει τα στοιχεία των χρηστών, δεδομένα για τα μουσεία όπως φωτογραφίες, τοποθεσία, στοιχεία επικοινωνίας αλλά και σύνθετες συσχετίσεις μεταξύ των χρηστών, αλλά και μεταξύ χρηστών και μουσείων όπως ποιος χρήστης μπορεί να δημιουργήσει και να προσθέσει κάποιο νέο μουσείο στην εφαρμογή. Για την υλοποίηση των παραπάνω, υλοποιήθηκε ένα σύστημα πιστοποίησης και εξουσιοδότησης χρηστών στο back end μέρος, αλλά και στο front end το UI διαφοροποιείται ανάλογα με αν κάποιος χρήστης είναι συνδεδεμένος ή μη συνδεδεμένος αλλά και τι δικαιώματα έχει ως συνδεδεμένος χρήστης. Θα ακολουθήσει ανάλυση του κάθε τμήματος περιλαμβάνοντας και την δομή της βάσης δεδομένων αναλυτικά και ξεχωριστά.

2.3 Spring Boot Framework

Το Spring είναι ένα Framework προγραμματισμού που δημιουργήθηκε αρχικά από τον Rod Johnson και βγήκε για πρώτη φορά στη παραγωγή το 2004 με την άδεια λογισμικού Apache 2.0. Η γλώσσα προγραμματισμού που μπορεί να χρησιμοποιήσει τις δυνατότητες του Spring είναι η Java. Κάθε web εφαρμογή γραμμένη σε Java μπορεί να χρησιμοποιήσει τις δυνατότητες του Spring όπως dependency injection, inversion of control (IoC). Πιο συγκεκριμένα, για τους σκοπούς της διατριβής έγινε χρήση του του Spring Boot το οποίο είναι μία επέκταση του Spring, το οποίο παρέχει ένα γρηγορότερο και πιο αποδοτικό περιβάλλον προγραμματισμού

διατηρώντας παράλληλα όλες τις δυνατότητες και λειτουργίες. Τα κύρια χαρακτηριστικά του Spring boot είναι, αυτόματη διαμόρφωση, δημιουργία αυτόνομων εφαρμογών με ενσωματωμένο server, επιλογή εξαρτήσεων κατά την δημιουργία της εφαρμογής, πρόσβαση σε αναφορές χρήσης του συστήματος και στατιστικά απόδοσης.

2.4 Dependency Injection

Το dependency injection είναι μία τεχνική σχεδιασμού λογισμικού κατά την οποία επιτυγχάνεται η χαλαρή σύζευξη αντικειμένων. Τα αντικείμενα τα οποία έχουν στενή σύζευξη μεταξύ τους, μπορούν να οδηγήσουν σε προβλήματα όταν για παράδειγμα μία αλλαγή στο ένα αντικείμενο δημιουργεί προβλήματα σε ένα άλλο στενά συνδεδεμένο. Τα αντικείμενα λοιπόν που είναι στενά συνδεδεμένα μεταξύ τους δεν μπορούν να αλλάξουν εύκολα χωρίς συνέπειες και δεν είναι εύκολα επαναχρησιμοποιήσιμα. Αντιθέτως, η τεχνική του dependency injection στοχεύει στον διαχωρισμό των αντικειμένων. Πιο συγκεκριμένα, διασφαλίζει ότι ένα αντικείμενο ή συνάρτηση που θέλει να χρησιμοποιήσει ένα συγκεκριμένο service δεν πρέπει να γνωρίζει πως να κατασκευάσει αυτά τα services. Αντιθέτως, ο ληφθείς πελάτης (client) (αντικείμενο ή συνάρτηση) παρέχεται με τις εξαρτήσεις του από εξωτερικό κώδικα (injector). Κάποια από τα προβλήματα που λύνει η τεχνική του dependency injection είναι τα παρακάτω: (‘The GoF Design Patterns Memory - Learning Object-Oriented Design & Programming’ n.d.)

- Πώς μία κλάση μπορεί να είναι ανεξάρτητη από την δημιουργία των αντικειμένων από τα οποία εξαρτάται;
- Πώς μπορεί μια εφαρμογή και τα αντικείμενα που χρησιμοποιεί να υποστηρίζουν διαφορετικές παραμετροποιήσεις;
- Πώς μπορεί να αλλάξει η συμπεριφορά ενός κομματιού κώδικα χωρίς απευθείας επεξεργασία;

Το dependency injection περιλαμβάνει τέσσερις ρόλους, services, clients, interfaces and injectors. Ένα service είναι μία οποιαδήποτε κλάση η οποία περιέχει χρήσιμη λειτουργικότητα. Από την άλλη, client είναι μία οποιαδήποτε κλάση που χρησιμοποιεί ένα service. Οι clients δεν θα πρέπει να γνωρίζουν πως οι εξαρτήσεις τους είναι υλοποιημένες, παρά μόνο το όνομα τους και του API, αυτό ουσιαστικά είναι το interface. Τέλος, οι injectors πρακτικά εισάγουν services στους clients. Ο ρόλος τους είναι να κατασκευάζουν και να συνδέουν πολύπλοκα γραφήματα αντικειμένων, όπου τα αντικείμενα μπορεί να είναι και services και clients.

2.5 Spring Boot Beans και Inversion of Control

Στο Spring, τα αντικείμενα που αποτελούν την ραχοκοκαλιά της εφαρμογής και τα οποία διαχειρίζεται το Spring IoC (Inversion of Control) container, ονομάζονται beans. Το Inversion of Control είναι μία διαδικασία κατά την οποία ένα αντικείμενο ορίζει τις εξαρτήσεις του χωρίς να τις δημιουργεί. Το αντικείμενο αυτό αναθέτει την κατασκευή τέτοιων εξαρτήσεων σε ένα IoC container. Στην παραδοσιακή προσέγγιση, χωρίς την χρήση των beans, η δημιουργία ενός αντικειμένου γίνεται με την χρήση του constructor. Αντίθετα, όταν γίνεται χρήση της τεχνικής του IoC, αντί να δημιουργεί εξαρτήσεις από μόνο του, ένα αντικείμενο μπορεί να ανακτήσει τις εξαρτήσεις του από ένα IoC container, αρκεί να παρέχουμε στο container τα κατάλληλα μετα-δεδομένα διαμόρφωσης (Thai 2018).

2.6 IntelliJ IDEA and Visual Studio Code

Το IntelliJ IDEA είναι ένα IDE (Integrated Development Environment) ειδικά σχεδιασμένο για προγραμματισμό με τις γλώσσες προγραμματισμού Java kai Kotlin. Δημιουργήθηκε και αναπτύσσεται από την JetBrains και σήμερα αποτελεί το πιο διαδεδομένο IDE για ανάπτυξη εφαρμογών. Μερικά από τα χαρακτηριστικά που προσφέρει είναι dataflow analysts, visual debugger,profiler.

Το Visual Studio Code είναι ένα κειμενογράφος που αναπτύχθηκε από την Microsoft και κυκλοφόρησε το 2015. Προσφέρει υποστήριξη για μεγάλο αριθμό γλωσσών προγραμματισμού, και λειτουργίες όπως debugger, git integration code suggestions. Στα πλαίσια της διατριβής, χρησιμοποιήθηκε για να υλοποιηθεί το front end μέρος του συστήματος με Angular Framework.

2.7 PostgreSQL

Η PostgreSQL, είναι ένα δωρεάν και ανοιχτού κώδικα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων που δίνει έμφαση στην επεκτασιμότητα και τη συμμόρφωση με την SQL. Το αρχικό της όνομα ήταν POSTGRES, ως διάδοχος της βάσης δεδομένων Ingres που αναπτύχθηκε στο Πανεπιστήμιο Μπέρκλεϋ της Καλιφόρνια και μετονομάστηκε σε PostgreSQL για να αντικατοπτρίζει ότι κάνει χρήση και επέκταση της SQL. Έχει σχεδιαστεί για να χειρίζεται μία σειρά από φόρτους εργασίας από μεμονωμένα μηχανήματα έως και αποθήκες δεδομένων ή υπηρεσίες Web με πολλούς ταυτόχρονους χρήστες. Είναι η προεπιλεγμένη βάση δεδομένων για macOs, ενώ είναι διαθέσιμη για Windows, Linux, FreeBSD και OpenBSD. Υποστηρίζει το μοντέλο ACID (atomicity, consistency, isolation, durability) για την διασφάλιση των συναλλαγών της (Stonebraker and Rowe, n.d.). Πιο αναλυτικά, το μοντέλο ACID περιγράφει τις παρακάτω δυνατότητες:

Ατομικότητα(atomicity): τα αποτελέσματα μιας συναλλαγής εμφανίζονται πλήρως ή καθόλου σε άλλες συναλλαγές

Συνοχή(consistency): περιορισμοί συνέπειας που καθορίζονται από το σύστημα επιβάλλονται στα αποτελέσματα των συναλλαγών

Απομόνωση(isolation): οι συναλλαγές δεν επηρεάζονται από τη συμπεριφορά των ταυτόχρονων συναλλαγών

Αντοχή(durability): μόλις μία συναλλαγή δεσμευτεί, τα αποτελέσματά της δεν θα χαθούν ανεξάρτητα από επακόλουθες αποτυχίες (Lane 2000).

2.8 Hibernate

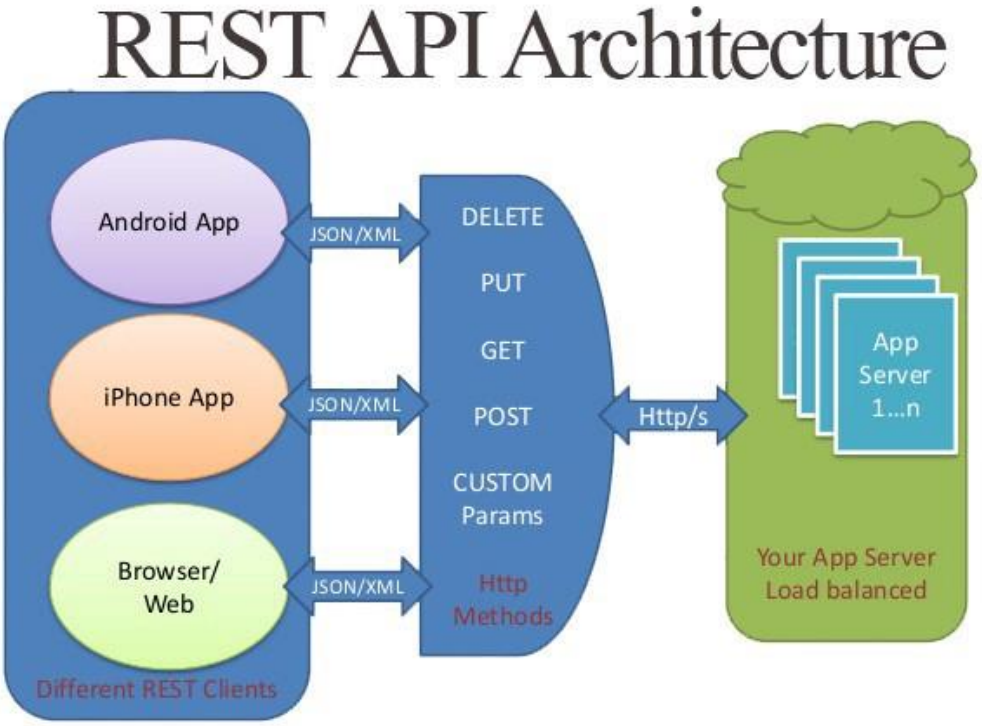
Είναι ένα δωρεάν, ανοιχτού λογισμικού σύστημα σχεσιακής χαρτογράφησης αντικειμένων (object relation mapping ORM). Παρέχει διασύνδεση και χαρτογράφηση μεταξύ της σχεσιακής βάσης δεδομένων και αντικείμενα σε επίπεδο εφαρμογής (java entity classes). Δημιουργήθηκε από την Red Hat Software το 2021 με άδεια λογισμικού GNU Lesser General Public License.

2.9 REST API

Rest (representational state transfer) API ή RESTful API (Application Programming Interface) είναι μία διεπαφή προγραμματισμού που συμμορφώνεται με του περιορισμούς της αρχιτεκτονικής Rest ως αντιπροσωπευτική μεταβίβαση κατάστασης και επιτρέπει αλληλεπίδραση με υπηρεσίες διαδικτύου. Επιπλέον, ένα σύστημα φτιαγμένο με Restful API χρησιμοποιεί HTTP requests για να έχει πρόσβαση στα δεδομένα της υπηρεσίας διαδικτύου. Τα requests αυτά είναι τύπου Get, Put, Post, Delete και χρησιμοποιούνται για ανάκτηση, ενημέρωση, δημιουργία και διαγραφή δεδομένων αντιστοίχως. Κάποια από τα πλεονεκτήματα της Rest αρχιτεκτονικής είναι ότι προσφέρει εύκολο τρόπο για οργάνωση περίπλοκων συστημάτων, χρησιμοποιεί χαμηλό bandwidth και σε περίπτωση που υπάρχει μεγάλος φόρτος δεδομένων στο δίκτυο, παρέχει δυνατότητα για χρήση HTTTP proxy server και cache για να κατανεμηθεί ο όγκος δεδομένων. Ακόμα, ένα ιδιαίτερο χαρακτηριστικό στο REST API είναι πως είναι stateless. Δηλαδή, κάθε request που δέχεται ο server, είναι και εξυπηρετείται ανεξάρτητα από άλλα requests που δέχεται. **(‘What Is RESTful API? - RESTful API Beginner’s Guide - AWS’ n.d.)**

Συνοπτικά, ο τρόπος που λειτουργεί ένα σύστημα υλοποιημένο σε RESTful API όπως και το σύστημα της διατριβής είναι:

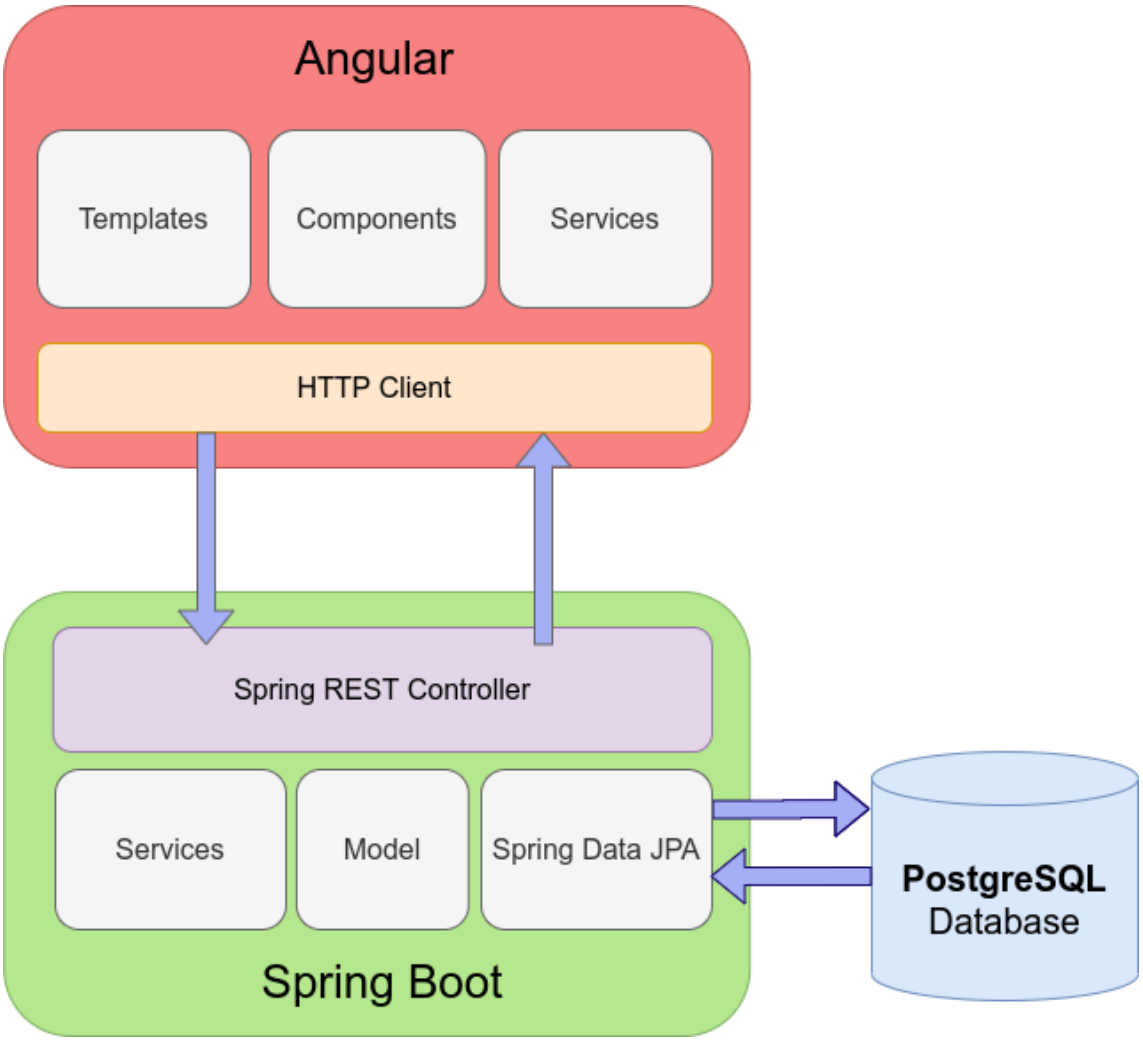
- Ο client στέλνει ένα Http request στον server. Είναι ευθύνη του client να ακολουθήσει τις οδηγίες που βρίσκονται στην τεκμηρίωση του API ώστε το request να έχει το κατάλληλο format που περιμένει ο server.
- Ο server λαμβάνει το request, αυθεντικοποιεί τον χρήστη που έκανε το request και επιβεβαιώνει ότι του επιτρέπεται να προχωρήσει.
- Έπειτα από την επιτυχημένη αυθεντικοποίηση, το request του client προχωράει και εκτελείται σύμφωνα με το business logic.
- Τέλος, ο server επιστρέφει την απάντηση στον client του front end. Η απάντηση περιέχει πληροφορίες όπως αν το request είναι επιτυχές ή όχι. Στη περίπτωση που δεν είναι, περιέχεται και ένα μήνυμα που υποδηλώνει τι προκάλεσε την απόρριψη. Επίσης, περιέχει και τα δεδομένα που ενδεχομένως ο χρήστης να ζήτησε με το request.



Εικόνα 1 – Αρχιτεκτονική REST API (Mamun 2019)

Στην εικόνα 1 απεικονίζεται η αρχιτεκτονική του Rest API και πιο συγκεκριμένα ο τρόπος επικοινωνίας των client με τον server.

Στα πλαίσια της διατριβής, όπως αναφέρθηκε έγινε χρήση του Spring Boot Framework με Java για να γίνει η έκθεση του API. Στο front-end μέρος χρησιμοποιήθηκε η Angular για να γίνεται η κατανάλωση-χρήση του API και των endpoints που έχουν δημιουργηθεί στον server της εφαρμογής.



Εικόνα 2 - Επικοινωνία του front-end με το back-end

Στην εικόνα 2 γίνεται απεικόνιση σε high level η επικοινωνία του front end με το back end κομμάτι και κατά επέκταση η επικοινωνία του με την βάση δεδομένων. Θα ακολουθήσει εκτεταμένη παρουσίαση αυτής λειτουργικότητας στο σύστημα της διατριβής στο επόμενα κεφάλαια.

2.10 Project Lombok

Το Project Lombok είναι μία βιβλιοθήκη της Java που δίνει την δυνατότητα να μειωθεί ποσοτικά ο κώδικας που γράφεται σε συγκεκριμένες κλάσεις και διαδικασίες μέσω συγκεκριμένων annotation που υποστηρίζει η βιβλιοθήκη αυτή. Για παράδειγμα, υπάρχουν annotations για αυτόματη δημιουργία contractors κλάσεων, setters και getters και χρήσιμες συναρτήσεις κλάσεων όπως toString kai equals. Για την χρησιμοποίησή του στο σύστημα χρειάζεται απλά η έκδοση της Java να είναι >= 8. Για τους σκοπούς της εργασίας έγινε χρήση και εισαγωγή με maven dependency. Τέλος, είναι ανοιχτού κώδικα και κυκλοφορεί υπό την άδεια λογισμικού MIT License. (‘Project Lombok’ n.d.)

2.11 Maven

Για την διαχείριση και εκτέλεση του back end μέρους της εφαρμογής αυτής της διατριβής, χρησιμοποιήθηκε το maven. Είναι ένα εργαλείο αυτοματοποίησης, διαχείρισης και εκτέλεσης προγραμμάτων, που είναι γραμμένα κυρίως σε Java, αλλά μπορεί να υποστηρίξει επίσης προγράμματα γραμμένα σε άλλες γλώσσες όπως C# και Ruby. Δημιουργήθηκε από την Apache Software Foundation, είναι open source και η άδεια λογισμικού του είναι Apache License 2.0.

Δημιουργήθηκε για να καλύψει τις ανάγκες απλοποίησης της κατασκευής έργων, δίνοντας ένα σαφή ορισμό από τι αποτελείται το έργο, ένα τρόπο κοινής χρήσης ενός JAR σε πολλά έργα καθώς και έναν εύκολο τρόπο δημοσίευσης πληροφοριών του έργου. Πρωταρχικός σκοπός του Maven είναι να επιτρέψει στον προγραμματιστή να κατανοήσει την πλήρη κατάσταση μιας προσπάθειας ανάπτυξης στο συντομότερο χρονικό διάστημα. Ακόμα, το Maven προσφέρει ευκολία στην συντήρηση των projects και στον κύκλο ζωής των εξαρτήσεων που χρησιμοποιεί καθώς όλα ελέγχονται μέσα από αυτό (‘Maven – Introduction’ n.d.).

Για την επίτευξη αυτού, διαχειρίζεται τα παρακάτω πεδία ανησυχίας:

- Κάνει εύκολη τη διαδικασία κατασκευής (build process)
- Παρέχει ένα ομοιόμορφο σύστημα κατασκευής
- Παρέχει ποιοτικές πληροφορίες για το έργο
- Προωθεί καλύτερες πρακτικές ανάπτυξης

2.12 Angular Material and Bootstrap

Στο front end μέρος της διατριβής, για τους σκοπούς του User Interface (UI) αλλά και για το User Experience (UX) έγιναν χρήση των βιβλιοθηκών Angular Material και του Bootstrap. Πιο συγκεκριμένα, το Angular Material είναι μία open-source βιβλιοθήκη για την Angular και παρέχει έτοιμα UI δομικά μέρη (components) έτσι ώστε να επιταχυνθεί η διαδικασία ανάπτυξης του λογισμικού και ταυτόχρονα να διατηρήσει τις σελίδες λειτουργικές, σταθερές αλλά και ελκυστικές προς τον χρήστη. Δημιουργήθηκε και κυκλοφόρησε από την Google. Το Bootstrap είναι επίσης ένα open-source framework για CSS. Περιέχει HTML,CSS και JavaScript και ο σκοπός του είναι οι σελίδες που δημιουργούνται με την χρήση του να είναι responsive και φιλικές προς τον χρήστη. Δημιουργήθηκε και κυκλοφόρησε για πρώτη φορά το 2011, και βρίσκεται κάτω από την άδεια λογισμικού MIT License.

3. Υλοποίηση και Σχεδιασμός Συστήματος

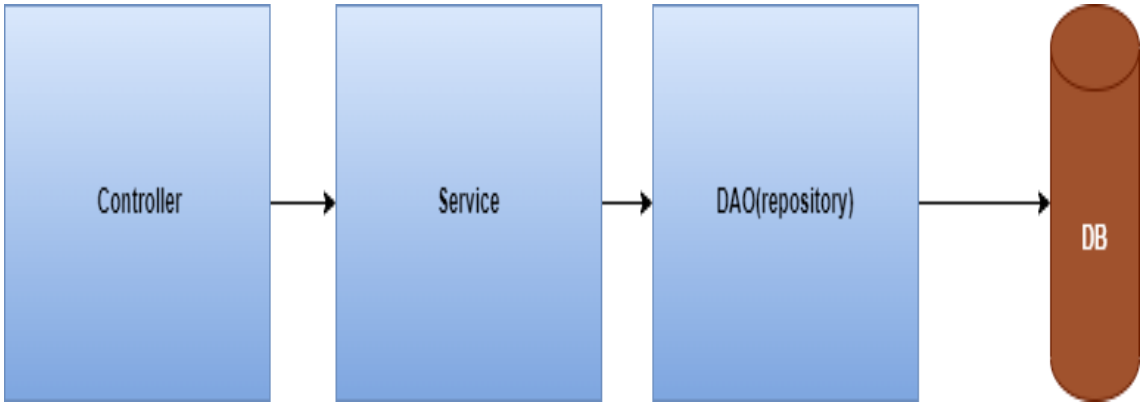
3.1 Περίληψη Κεφαλαίου

Στο κεφάλαιο αυτό θα αναλυθεί ο τρόπος υλοποίησης του back-end συστήματος της διατριβής, το front-end, καθώς και η δομή της βάσης δεδομένων. Ακόμα, πως δομήθηκε συνολικά το σύστημα, πως πάρθηκαν κάποιες σχεδιαστικές και τεχνικές αποφάσεις, αλλά και αναλυτική παρουσίαση των δυνατοτήτων του συστήματος και των τεχνολογιών που το απαρτίζουν.

3.2 Σχεδιασμός και λειτουργίες στο API

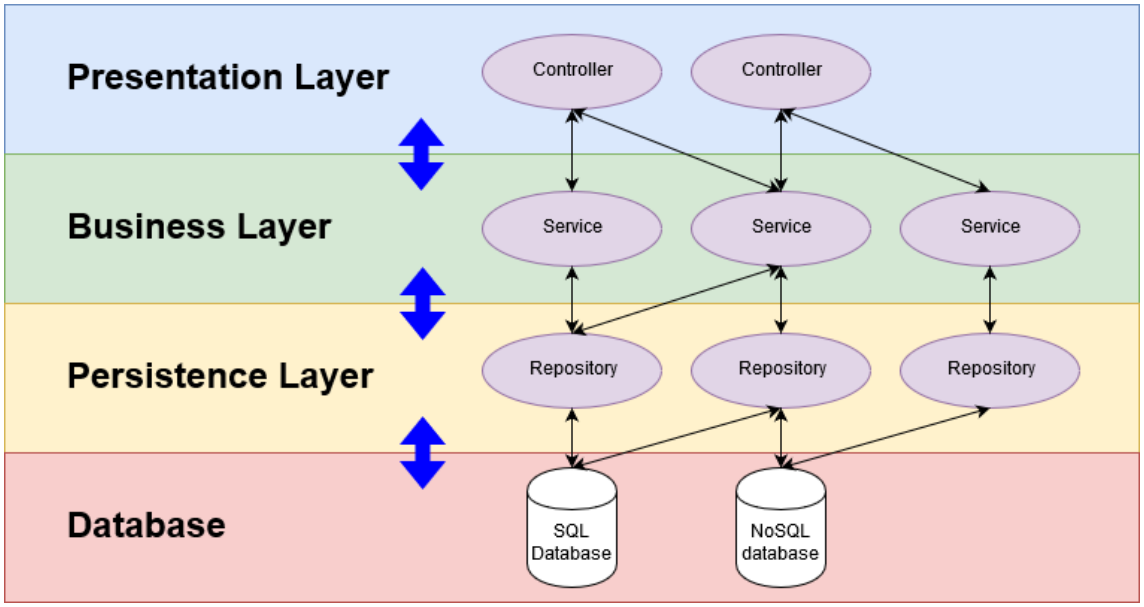
3.2.1 Σχεδιασμός Αρχιτεκτονικής Spring Boot

Στο σημείο αυτό, θα αναλυθεί και θα παρουσιαστεί η δομή και η αρχιτεκτονική που έχει επιλεχθεί για τον πυρήνα του συστήματος. Το αρχιτεκτονικό design που έχει επιλεχθεί στα πλαίσια της μεταπτυχιακής διατριβής είναι το Controller Service Repository Pattern (Paul 2017). Είναι ένα ευρέως χρησιμοποιούμενο αρχιτεκτονικό πρότυπο για τις εφαρμογές Spring Boot. Πιο συγκεκριμένα, θα αναλυθούν τα χαρακτηριστικά του και τα πλεονεκτήματά του καθώς και γιατί επιλέχθηκε η συγκεκριμένη αρχιτεκτονική.



Εικόνα 3 - Επικοινωνία μεταξύ controller, service και repository

Στην εικόνα 3 απεικονίζεται η σειρά με την οποία είναι δομημένα και επικοινωνούν μεταξύ τους στο σύστημα οι controller, τα services και τα repositories. Επίσης, στην εικόνα 3 φαίνεται πως επικοινωνία με την βάση δεδομένων της εφαρμογής, δεν έχουν απευθείας όλα τα επίπεδα αλλά μόνο το επίπεδο DAO μέσω των repositories.



Εικόνα 4 - Επίπεδα επικοινωνίας (Stavast 2018)

Στην εικόνα 4, απεικονίζεται με μεγαλύτερη σαφήνεια πως το κάθε επίπεδο (layer) απαρτίζεται από αρκετές κλάσεις. Στη συνέχεια το κάθε επίπεδο θα αναλυθεί ξεχωριστά, ώστε να γίνει κατανοητός ο λόγος που χρησιμοποιήθηκε και πως εξυπηρετεί στο συνολικό σχεδιασμό του συστήματος.

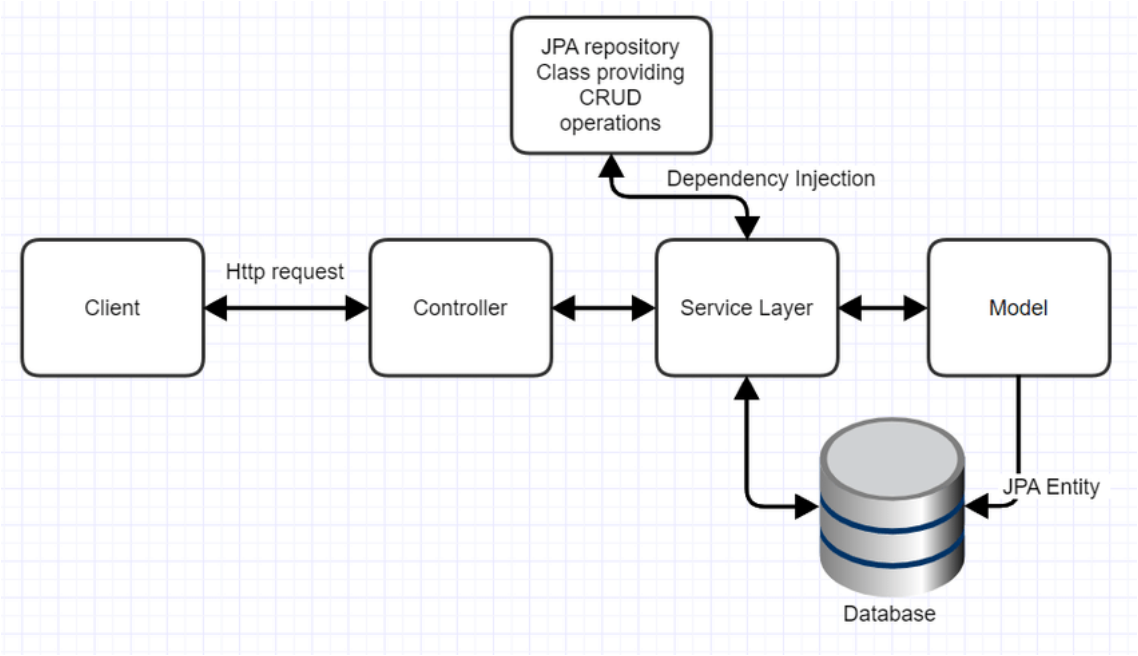
Ξεκινώντας με το Presentation Layer στο οποίο υπάρχουν οι κλάσεις Controllers. Αυτές είναι ουσιαστικά εξυπηρετητές διαδικτυακών αιτημάτων (web request handlers). Δηλώνονται στο σύστημα με το annotation “Controller” και συνδυάζονται με την διεύθυνση που κάνουν έκθεση(expose) στο API του εκάστοτε συστήματος. Μέσα στους Controllers, δεν υπάρχει επιχειρησιακή λογική και είναι απομονωμένοι από το επίπεδο που την υλοποιεί, καθώς επίσης δεν έχουν άμεση επικοινωνία και με το επίπεδο που χειρίζεται την βάση δεδομένων. Είναι ανεξάρτητες κλάσεις και αυτό προσφέρει ευελιξία στο σύστημα τόσο για την ταχύτερη ανάπτυξή του όσο και για την ευκολότερη συντήρησή του.

Κάθε διαδικτυακός χειριστής(web handler), μόλις καλεστεί θα προωθήσει το αίτημα που δέχτηκε στο επόμενο επίπεδο. Το επόμενο επίπεδο που είναι το Business Layer θα εκτελέσει τις ενέργειες που αφορούν το συγκεκριμένο αίτημα και θα επιστρέψει τα αποτελέσματα στην συνάρτηση του controller από τον οποίο καλέστηκε. Ο Controller τελικά θα επιστρέψει την τελική απάντηση στο σύστημα που το κάλεσε και τότε, έχει λάβει θέση η ολοκλήρωση ενός διαδικτυακού αιτήματος(web request). Στους Controllers, υπάρχει επίσης και έλεγχος προσβασιμότητας και ασφάλειας, αλλά το συγκεκριμένο θέμα θα αναλυθεί σε επόμενο στάδιο.

Στη συνέχεια, στο Business Layer βρίσκονται οι κλάσεις Services. Σε αυτές έχει υλοποιηθεί η επιχειρησιακή λογική(business logic) που διαμορφώνεται ανάλογα με τις απαιτήσεις που θέτει η κάθε εφαρμογή. Για να ολοκληρωθεί η διαδικασία σε ένα αίτημα, τα services που ενδεχομένως χρειάζονται δεδομένα από την βάση δεδομένων, είτε για εγγραφή είτε για ανάκτηση θα χρησιμοποιήσουν το αμέσως επόμενο layer το οποίο είναι το Persistence Layer. Επίσης, στο Service Layer, θα εκτελεστούν όλοι οι κανόνες σύμφωνα με την επιχειρησιακή λογική της εφαρμογής. Οι κανόνες αυτοί μπορούν να αφορούν θέματα ασφάλειας, όπως αν για παράδειγμα ο συγκεκριμένος χρήστης έχει πρόσβαση στα δεδομένα που ζητάει ή αν κάποιος χρήστης έχει κλειδωμένο λογαριασμό και δεν μπορεί να προχωρήσει σε σύνδεση. Τέλος, όταν ολοκληρώσει το Service Layer τις ενέργειές του, επιστρέφει το αποτέλεσμα στο Presentation Layer και συγκεκριμένα στον Controller που το κάλεσε.

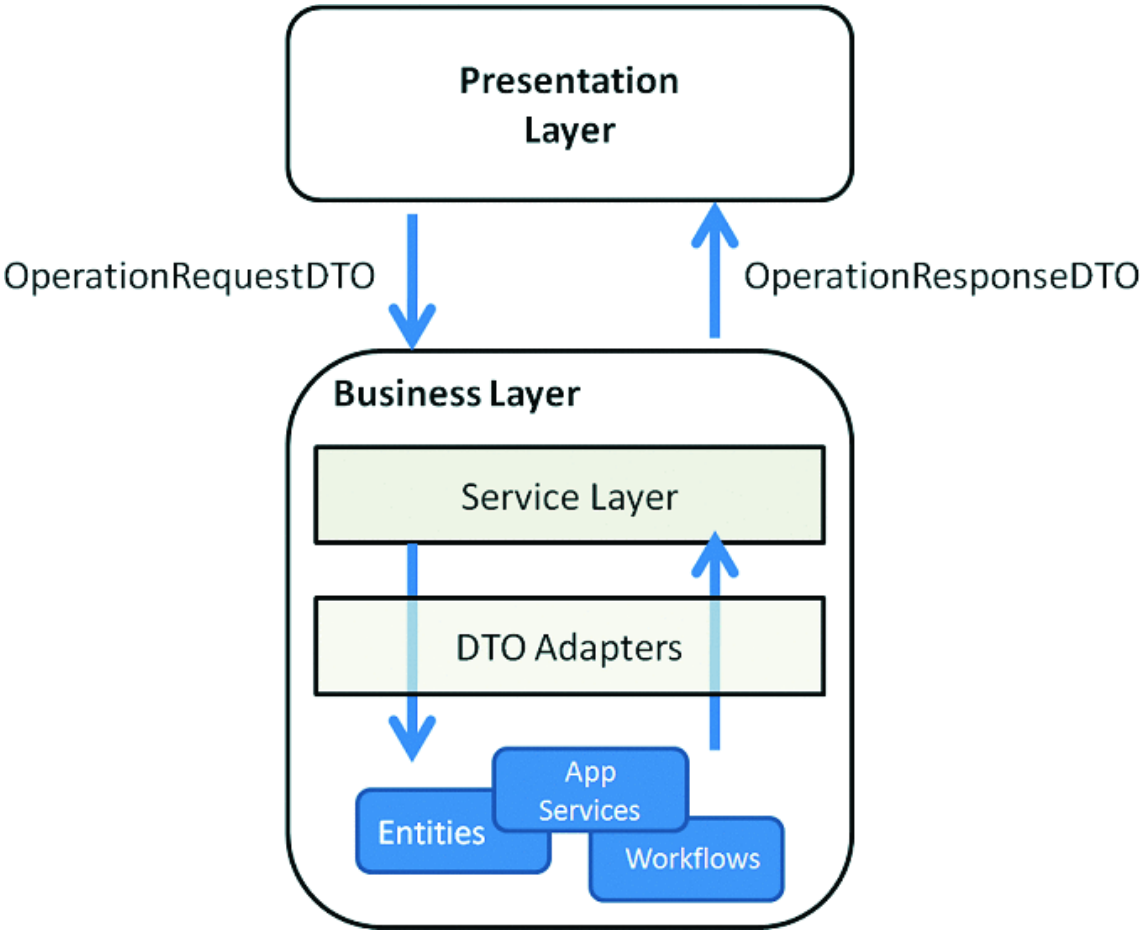
Τέλος, θα αναλυθούν το Persistence Layer και η βάση δεδομένων. Το πρώτο είναι υπεύθυνο για την αποθήκευση, ανάκτηση και προστασίας των δεδομένων στην εφαρμογή. Είναι το μοναδικό Layer που έχει επικοινωνία με την βάση. Στο επίπεδο αυτό, βρίσκεται το Hibernate που

έχει ήδη αναφερθεί και είναι υπεύθυνο για την συσχέτιση των Java Objects με τα αντίστοιχα αντικείμενα στη βάση δεδομένων. Μέσω του Hibernate και όλων των Repositories που έχουν αναφερθεί, γίνονται όλες οι συναλλαγές με την βάση. Αυτό προσφέρει ασφάλεια, ευελιξία αλλά και διατήρηση της συνοχής των δεδομένων(data consistency) σε περίπτωση που προκύψει κάποιο σφάλμα. Αυτό επιτυγχάνεται με την διαχείριση σφαλμάτων(error handling) που προσφέρει το Hibernate και την δυνατότητα αντιστροφής των συναλλαγών με την βάση δεδομένων μέχρι να ολοκληρωθεί ολόκληρη η διαδικασία όπως ορίζεται από την επιχειρησιακή λογική. Όπως απεικονίζεται και στην εικόνα 5 τα Repositories εκτελούν όλες τις διαδικασίες προσπέλασης στην βάση δεδομένων πιο συγκεκριμένα, Δημιουργία, Ανάγνωση, Ενημέρωση και Διαγραφή(CRUD Create, Read, Update, Delete).



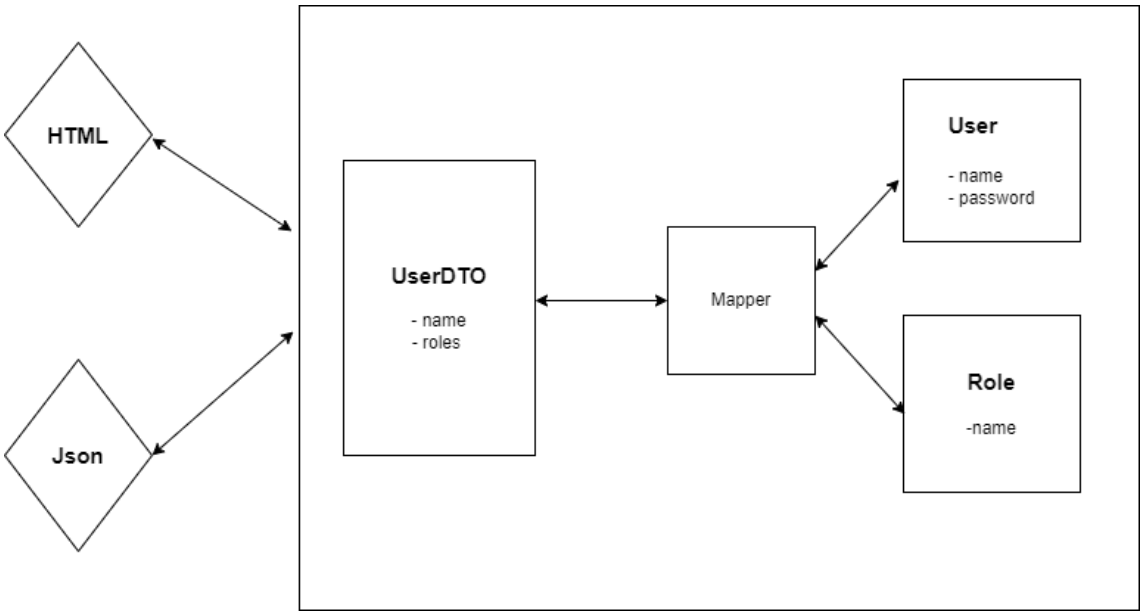
Εικόνα 5 - Επικοινωνία με την βάση (Rao and Swamy 2020)

Τέλος, υπάρχουν και οι κλάσεις που είναι για την μεταφορά δεδομένων από το front-end στο back-end μέρος της εφαρμογής. Ονομάζομαι αντικείμενα για την μεταφορά δεδομένων(Data Transfer Objects DTOs) και χρησιμοποιούνται για να επιστραφεί η απάντηση από το back-end στο front-end αλλά και το αντίστροφο. Επίσης, είναι απλές κλάσεις POJOs (Plain Old Java Objects), και ο λόγος ύπαρξής τους όπως αναφέρθηκε είναι η μεταφορά δεδομένων μεταξύ συστημάτων με τρόπο ώστε να αποκρύψουν ευαίσθητα δεδομένα που δεν πρέπει να μεταφερθούν και να μεταφέρουν τα απολύτως απαραίτητα. Με αυτό το τρόπο, έχουν και μικρότερο μέγεθος και είναι σημαντικό μέρος για την απόδοση της εφαρμογής αλλά και στην κατανάλωση πόρων δικτύου. Η δημιουργία τους γίνεται από το Service Layer. Συγκεκριμένα, μόλις ολοκληρώσει τις ενέργειες που πρέπει, θα δημιουργήσει ένα DTO το οποίο θα προωθήσει στον Controller, και ο τελευταίος στο front-end κομμάτι. Αντίστοιχα, ο Controller επίσης θα καταναλώσει-διαβάσει και ένα DTO που θα περιέχει τα δεδομένα του request που δέχθηκε από το front-end. Η προώθηση του DTO θα γίνει στο Service layer, από εκεί θα διαβαστεί και αφού ολοκληρωθούν οι απαραίτητες ενέργειες, θα επιστρέψει το καινούργιο response DTO.



Εικόνα 6 - Επικοινωνία του presentation layer με χρήση DTO (kexugit n.d.)

Η επικοινωνία του presentation layer με Data Transfer Objects απεικονίζεται στην εικόνα 6.



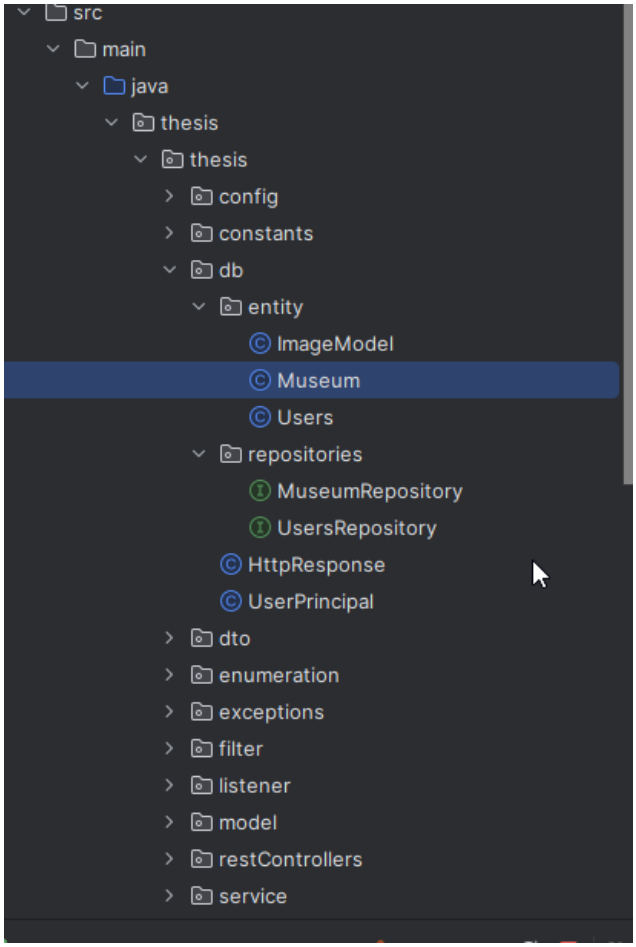
Εικόνα 7 - Μετατροπή αντικειμένου JSON σε DTO

Επιπλέον, στην εικόνα 7 γίνεται μια πιο λεπτομερείς αναπαράσταση για την μετατροπή των δεδομένων στα DTOs καθώς και αναφέρεται ότι η ανταλλαγή δεδομένων ανάμεσα σε front-end και back-end γίνεται με το format JSON(JavaScript Object Notation).

3.2.2 Σχεδιασμός Domain Επιπέδου (Domain Layer)

Στο κεφάλαιο αυτό, θα γίνει παρουσίαση για το Domain επίπεδο της εφαρμογής, δηλαδή για τα μουσεία και για τους χρήστες και πως αυτά αποθηκεύονται μέσα στο σύστημα. Επίσης, για τα Java Repositories που χρησιμοποιούνται ως Αντικείμενα για πρόσβαση στα δεδομένα(Data Access Objects DAO) και είναι υπεύθυνα για την επικοινωνία-διασύνδεση της Java με την βάση δεδομένων.

Στο σημείο αυτό, θα γίνει μία επεξήγηση λοιπόν στο Domain επίπεδο για να εξηγηθούν πιο αναλυτικά τα Domain Entities και τα Repositories της εφαρμογής.



Εικόνα 8 - Entity domain classes

Όπως φαίνεται στην εικόνα 8, βλέπουμε ότι υπάρχουν τρία Entity Domain classes. Το museum, το Users και το ImageModel. Αντίστοιχα, έχουμε δύο Repositories από τα οποία έχουμε και την επικοινωνία με την βάση δεδομένων την εφαρμογής. Με βάση αυτές τις τρεις οντότητες, θα

γίνονται όλες οι αλλαγές, ενημερώσεις και διαγραφές της εφαρμογής στην βάση δεδομένων του συστήματος.

Ξεκινώντας με την οντότητα ImageModel, η οποία αποτελεί την πιο απλή οντότητα του συστήματος και χρησιμοποιείται για την αποθήκευση των φωτογραφιών μέσα στο σύστημα.

```
11 usages   azaridak *
@Entity
@Table(name = "image_model")
public class ImageModel {

    3 usages
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    4 usages
    @Column(name="name")
    private String name;

    4 usages
    @Column(name="type")
    private String type;

    4 usages
    @Column(name="byte")
    private byte[] picByte;
```

Εικόνα 9 - ImageModel class

Όπως φαίνεται και στην εικόνα 9, βλέπουμε ότι έχει μία απλή δομή. Αποτελείται από ένα id που παράγεται αυτόματα(auto generated), το όνομα χρήστη(username) που αφορά τον χρήστη που είναι να αποθηκεύσει το όνομα της φωτογραφίας, και ένα πίνακα από bytes(byte array) που σώζει τα πραγματικά δεδομένα της φωτογραφίας. Θα αναλυθεί και θα εξηγηθεί περισσότερο πως λειτουργεί στο σύστημα η αποθήκευση φωτογραφιών, στο κεφάλαιο των μουσείων.

Στη συνέχεια, υπάρχει η κλάση Users που είναι το Entity-Domain για τους χρήστες του συστήματος. Πιο συγκεκριμένα, αποτελεί την χαρτογράφηση των χρηστών που έχουν κάνει εγγραφή στο σύστημα και στο πως είναι αποθηκευμένοι μέσα στη βάση δεδομένων της εφαρμογής.

```
3 usages
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
@Column(name="user_id",nullable = false,insertable = false,updatable = false)
private int user_id;

3 usages
@Column(name = "username",nullable = false)
private String username;

3 usages
@Column(name = "email", nullable = false,unique = true)
private String email;

3 usages
@Column(name = "profileImgUrl", unique = true)
private String profileImgUrl;

3 usages
@Column(name = "password",nullable = false)
@JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
private String password;

3 usages
@Column(name = "created_on",nullable = false,updatable = false,insertable = false)
private Timestamp createdOn;

3 usages
@Column(name = "first_name",nullable = false)
private String firstName;
```

Εικόνα 10 - Users class

```
3 usages
@Column(name = "last_name",nullable = false)
private String lastName;

3 usages
@Column(name = "gender",nullable = false)
private String gender;

3 usages
@Column(name = "mobile",nullable = true)
private String mobile;

3 usages
@Column(name = "cancreatemuseum", nullable = true)
private String canCreateMuseum;

3 usages
@JoinColumn(name = "museum_id_fk")
@OneToOne(cascade = CascadeType.REFRESH, fetch = FetchType.LAZY)
private Museum museum;


// variables for spring principle
3 usages
private String role;

4 usages
private String authorities;

4 usages
@Column(name = "accountislocked", nullable = true)
private Boolean accountIsLocked;

3 usages
@Column(name = "lastlogindate")
private Long lastLoginDate;
```

Εικόνα 11 - Users class

Όπως φαίνεται από τις εικόνες 10 και 11, η κλάση Users περιέχει όλα τα στοιχεία του χρήστη τα οποία και αποθηκεύονται μέσα στο σύστημα. Εκτός από τα στοιχεία που εισάγει ο ίδιος ο χρήστης στην εφαρμογή, πάρθηκε η σχεδιαστική απόφαση να αποθηκευτούν κάποια επιπλέον πεδία, τα οποία χρησιμεύουν στην εσωτερική λειτουργία της εφαρμογής. Για παράδειγμα, οι μεταβλητές “Boolean accountIsLocked” χρησιμοποιείται κατά την σύνδεση(log-in) του χρήστη. Στη περίπτωση που η τιμή της είναι true, τότε ο χρήστης δεν θα μπορεί να πραγματοποιήσει σύνδεση και θα πρέπει να επικοινωνήσει με τον διαχειριστή του συστήματος. Επίσης, οι μεταβλητές “role” και “authorities” χρησιμοποιούνται εσωτερικά στο σύστημα έτσι ώστε να ελέγχει αν ένας χρήστης έχει πρόσβαση σε κάποιο αίτημα το οποίο αιτήθηκε. Για παράδειγμα, για να μπορεί να διαγράψει έναν χρήστη από το σύστημα, θα πρέπει ο χρήστης που έκανε αυτό το αίτημα να έχει δικαιώματα διαγραφής χρηστών και ταυτόχρονα ρόλο διαχειριστή(admin). Τα οποία καθορίζονται από την οντότητα του χρήστη και συγκεκριμένα, τα πεδία role και authorities τα οποία μπορούν να πάρουν συγκεκριμένες τιμές που καθορίζονται μέσα στο σύστημα στα security enumeration. Θα γίνει αναλυτική παρουσίαση των χαρακτηριστικών αυτών και στη συνέχεια.

Επιπλέον, κάποια από τα γνωρίσματα του χρήστη έχουν το Entity attribute “unique = true”. Το γνώρισμα αυτό, λειτουργεί ως έξτρα δικλείδα ασφαλείας πως δε θα υπάρχουν διπλότυπα στο σύστημα σε τιμές που δεν θα έπρεπε να επιτρέπεται. Για παράδειγμα, δεν είναι εφικτό στο σύστημα να εγγραφούν δύο διαφορετικοί χρήστες και να έχουν την ίδια διεύθυνση ηλεκτρονικού ταχυδρομείου(email) ή το ίδιο όνομα-χρήστη(username) καθώς αυτά αποτελούν μοναδικά χαρακτηριστικά των χρηστών μέσα στο σύστημα και χρησιμοποιούνται για την αναγνώρισή τους μέσα σε αυτό. Τέλος, υπάρχει το γνώρισμα museum το οποίο χαρτογραφείται με την “museum_id_fk” της βάσης δεδομένων και είναι Join Column που συσχετίζει τον εγγεγραμμένο χρήστη με ένα μουσείο που συσχετίζεται και για το οποίο είναι διαχειριστής και έχει την δυνατότητα να κάνει κάποια παραμετροποίηση στα στοιχεία του. Όπως επίσης φαίνεται στην λίστα, η σχέση που υπάρχει μεταξύ των δύο αυτών οντοτήτων είναι μία προς μία (One To One) που σημαίνει ότι ένας χρήστης μπορεί να είναι διαχειριστής σε ένα μουσείο. Αντίστοιχα, ένα μουσείο μπορεί να διαχειρίζεται από ένα χρήστη-διαχειριστή. Φυσικά, όπως έχει ήδη αναφερθεί, για να είναι ένας χρήστης διαχειριστής σε οποιοδήποτε μουσείο θα πρέπει να έχει τα κατάλληλα δικαιώματα χρήσης μέσα στην εφαρμογή. Για τον τρόπο με τον οποίο παρέχονται τα δικαιώματα σε κάθε χρήστη, θα ακολουθήσει αναλυτική παρουσίαση στο επόμενο κεφάλαιο με περιπτώσεις χρήσεως.

Συνεχίζοντας, θα αναλύσουμε την οντότητα του μουσείου μέσα στο σύστημα καθώς και την Entity class Museum.

```
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name="museum_id",nullable = false,insertable = false,updatable = false)
    private int museum_id;

    3 usages
    @Column(name = "title", nullable = false,length = 128)
    private String title;

    3 usages
    @Column(name = "email", nullable = false,length = 32)
    private String email;

    3 usages
    @Column(name = "address", nullable = false,length = 64)
    private String address;

    3 usages
    @Column(name = "genre")
    private String genre;

    3 usages
    @Column(name = "description")
    private String description;

    3 usages
    @Column(name = "shortdescription")
    private String shortDescription;

    3 usages
    @Column(name = "city", nullable = false)
    private String city;

    3 usages
    @Column(name = "createdonyear", nullable = false)
    private String createdOnYear;

    3 usages
    @Column(name = "telephone", nullable = false,unique = true)
    private String telephone;
```

Εικόνα 12 - Museum class

```
    @Column(name = "telephone2")
    private String telephone2;

    3 usages
    @Column(name = "url",unique = true)
    private String url;

    3 usages
    @Column(name = "youtubeUrl",unique = true)
    private String youtubeUrl;

    3 usages
    @Column(name = "facebookUrl",unique = true)
    private String facebookUrl;

    3 usages
    @Column(name = "twitterUrl",unique = true)
    private String twitterUrl;

    3 usages
    @JoinColumn(name = "user_id_fk", nullable = true, updatable = true)
    @OneToOne(cascade = CascadeType.REFRESH, fetch = FetchType.LAZY)
    private Users users;

    3 usages
    @ManyToMany(fetch = FetchType.EAGER, cascade = CascadeType.ALL)
    @JoinTable(
            name = "museum_images",
            joinColumns = {
                @JoinColumn(name="museum_id")
            },
            inverseJoinColumns = {
                @JoinColumn(name = "image_id")
            }
    )
    private Set<ImageModel> museumImages;
```

Εικόνα 13 - Museum class

Όπως φαίνεται στις εικόνες 12 και 13, παρουσιάζονται τα χαρακτηριστικά, τα γνωρίσματα και το πως έχει δομηθεί η οντότητα μουσείου μέσα στο σύστημα για την μεταπτυχιακή διατριβή. Στην εικόνα 12 παρουσιάζονται τα βασικά χαρακτηριστικά του μουσείου που συμπληρώνονται από τον χρήστη και αποθηκεύονται στην βάση δεδομένων. Κάποια γνωρίσματα του μουσείου, όπως έγινε

και στο Entity των Users έχουν το γνώρισμα “unique = true” καθώς τα γνωρίσματα αυτά πρέπει να είναι μοναδικά για το κάθε μουσείο και χρησιμοποιούνται ως αναγνωριστικά χαρακτηριστικά μέσα στην εφαρμογή. Για παράδειγμα, δεν μπορούν δύο μουσεία να έχουν την ίδια διεύθυνση ηλεκτρονικού ταχυδρομείου ή τον ίδιο προσωπικό ιστότοπο.

Στη συνέχεια, συγκεκριμένα στην εικόνα 13 απεικονίζονται επίσης κάποια μοναδικά γνωρίσματα, τα οποία είναι τα μέσα κοινωνικής δικτύωσης(social media) του μουσείου. Οι τιμές αυτές δεν είναι υποχρεωτικές να συμπληρωθούν, διότι κάποιο μουσείο δύναται να μην έχει λογαριασμό σε κάποιο ή όλα από τα μέσα κοινωνικής δικτύωσης. Ακόμα, φαίνονται και τα γνωρίσματα Users και «museumImages» τα οποία θα αναλυθούν εκτεταμένα. Ξεκινώντας, με το γνώρισμα Users, γίνεται η χαρτογράφηση της κολόνας user_id_fk η οποία είναι η αναφορά στο τραπέζι των χρηστών της βάσης δεδομένων. Συγκεκριμένα, είναι το JoinColumn για την διασύνδεση του μουσείου με τον χρήστη του συστήματος που φυσικά είναι και ο διαχειριστής του. Επιπλέον, η σχέση μεταξύ των δύο οντοτήτων είναι μία προς μία (one to one). Πρακτικά, αυτό σημαίνει πως ένας χρήστης μπορεί να διαχειρίζεται το συγκεκριμένο μουσείο, και ως επακόλουθο ένα μουσείο μπορεί να έχει ένα διαχειριστή.

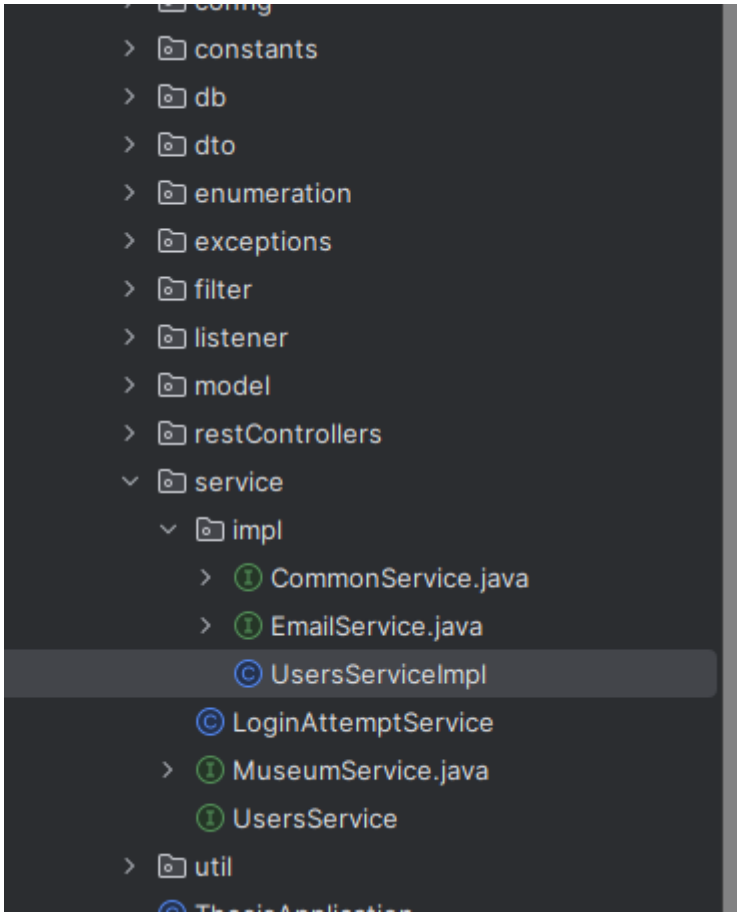
Το επόμενο γνώρισμα που φαίνεται στην εικόνα 13 είναι το «museumImages». Για το συγκεκριμένο έχει επιλεχθεί η δομή δεδομένων Set(HashSet) από την οντότητα «ImageModel». Για την δομή δεδομένων HashSet, κάποια από τα χαρακτηριστικά της είναι πως δεν επιτρέπει διπλότυπες τιμές, έχει χρόνο πολυπλοκότητας για ανάκτηση στοιχείων O(1) και επίσης έχει O(1) χρονική πολυπλοκότητα στην προσθήκη νέων δεδομένων. (‘HashSet (Java Platform SE 7)’ n.d.) Τα τρία αυτά χαρακτηριστικά που αναφέρθηκαν το καθιστούν ιδανική επιλογή για να αποθηκεύσουμε τις φωτογραφίες των μουσείων, καθώς ο πολύ αποδοτικός ο χρόνος για την προσθήκη και την ανάκτηση δεδομένων διαδραματίζει σημαντικό ρόλο στην συνολική απόδοση της εφαρμογής.

Στο σημείο αυτό θα εξηγηθεί το annotation @JoinTable. Αρχικά, η σχέση που υπάρχει ανάμεσα στο μουσείο και στις φωτογραφίες είναι «πολλά σε πολλά»(many to many). Για να χαρτογραφηθεί μία σχέση οντοτήτων «πολλά σε πολλά», χρησιμοποιείται ένα βοηθητικό τραπέζι που συχνά αναφέρεται ως συνδετικό τραπέζι (join table). Το Join Table αποτελείται από κολόνες που περιέχουν τα κύρια κλειδιά–primary keys των οντοτήτων που συσχετίζει. Συγκεκριμένα, στην περίπτωση του συστήματος της μεταπτυχιακής διατριβής, αποτελείται από μία κολώνα για το “museum_id” που είναι το μοναδικό αναγνωριστικό των μουσείων, και μία κολόνα από το “image_id” που επίσης είναι το μοναδικό αναγνωριστικό για τις εικόνες. Στο σύνολό του λοιπόν, κάθε γραμμή θα έχει δύο μοναδικά αναγνωριστικά, ένα για το μουσείο και ένα για την εικόνα.

Τέλος, επειδή για να μπορέσει να λειτουργήσει η χαρτογράφηση σε επίπεδο της εφαρμογής μέσω του Hibernate ORM που όπως αναφέρθηκε γίνεται χρήση στα πλαίσια της μεταπτυχιακής διατριβής, χρειάζεται επιπλέον πιο σύνθετα annotations τα οποία και θα εξηγηθούν. Το στοιχείο «names» αναφέρεται στο όνομα του συνδετικού τραπεζιού που βρίσκεται μέσα στη βάση δεδομένων. Ύστερα, ακολουθεί το “joincolumns = museum_id”, με αυτόν τον τρόπο δηλώνεται στο σύστημα πως η μία τιμή του συνδετικού τραπεζιού θα είναι αυτή. Αντίστοιχα, το «inverseJoincolumns» αναφέρεται στο μοναδικό γνώρισμα «image_id», ο συνδυασμός αυτών των δύο, συνθέτουν το βοηθητικό τραπέζι και ο συνδυασμός «museum_id-image_id» λειτουργεί ως κλειδί για την ανάκτηση και εγγραφή δεδομένων στο τραπέζι αυτό. (‘Relational Data Modeling - Association Table (Bridge, Cross)’ 2020)

3.2.3 Σχεδιασμός Επιπέδου Service (Service Layer)

Στο κεφάλαιο αυτό, θα γίνει παρουσίαση του Service Layer και το πως έχει δομηθεί στα πλαίσια της μεταπτυχιακής διατριβής. Στο συγκεκριμένο Layer, βρίσκεται ουσιαστικά ο πυρήνας του συστήματος καθώς σε αυτό έχει υλοποιηθεί έχει εφαρμοστεί η επιχειρησιακή λογική. Βάσει αυτού, έχει δομηθεί και όλη η λειτουργικότητα της εφαρμογής. Θα ακολουθήσει επεξήγηση των κλάσεων που απαρτίζουν το Service Layer, αλλά και πως αλληλοεπιδρούν συνολικά με το σύστημα για την αποπεράτωση των λειτουργειών του.



Εικόνα 14 - Υλοποίηση του UserService Interface

Όπως απεικονίζεται στην εικόνα 14, μέσα στο πακέτο «service» υπάρχουν τα Services classes και συγκεκριμένα Interfaces. Ο λόγος που υλοποιήθηκαν με Interfaces είναι ότι με αυτόν τον σχεδιασμό προσφέρουν στο σύστημα τεχνικά πλεονεκτήματα. Πιο συγκεκριμένα, μερικά από τα πλεονεκτήματα αυτά είναι η αφαιρετικότητα (abstraction), η πολλαπλή κληρονομικότητα και η χαλαρή σύζευξη (loose coupling) μεταξύ των κλάσεων. Τα αναφερόμενα ως πλεονεκτήματα, αποτελούν θεμελιώδη λίθο στον αντικειμενοστραφή προγραμματισμό (Object Oriented Programming) και στην καλή τεχνοτροπία δημιουργίας λογισμικού. Η αφαιρετικότητα, επιτρέπει οι υπογραφές των συναρτήσεων να βρίσκονται σε διαφορετικό αρχείο από την υλοποίηση τους. Αυτό έχεις ως αποτέλεσμα, το λεγόμενο συμβόλαιο υπογραφών των συναρτήσεων, να μπορεί να είναι και δημόσιο καθώς, αποκρύβονται όλες οι λεπτομέρειες υλοποίησης. Αφού λοιπόν, οι λεπτομέρειες υλοποίησης αποκρύβονται είναι ασφαλές να χρησιμοποιείται ως δημόσιο συμβόλαιο υπογραφών συναρτήσεων. Όμως, η υλοποίηση των συναρτήσεων αυτών που βρίσκονται σε ένα άλλο αρχείο φυσικά δεν μπορεί να είναι δημόσιο καθώς περιέχουν τον τρόπο που έχει εφαρμοστεί η επιχειρησιακή λογική και κακόβουλοι χρήστες θα μπορούσαν να το χρησιμοποιήσουν για να βρουν πιθανά κενά ασφαλείας. Επιπλέον, η πολλαπλή κληρονομικότητα με την χρήση των Interfaces δηλαδή, η δυνατότητα ύπαρξης πολλαπλών διαφορετικών υλοποιήσεων για κάθε υπογραφή-συνάρτηση από διαφορετικές κλάσεις, προσφέρει ευελιξία στην υλοποίηση του συστήματος. Τέλος, χαλαρή σύζευξη(loose coupling) επιτυγχάνεται και αναφέρεται στο ότι οι υπογραφές με την υλοποίηση των συναρτήσεων βρίσκονται σε διαφορετικά αρχεία τα οποία είναι μεταξύ τους και απολύτως ανεξάρτητα.

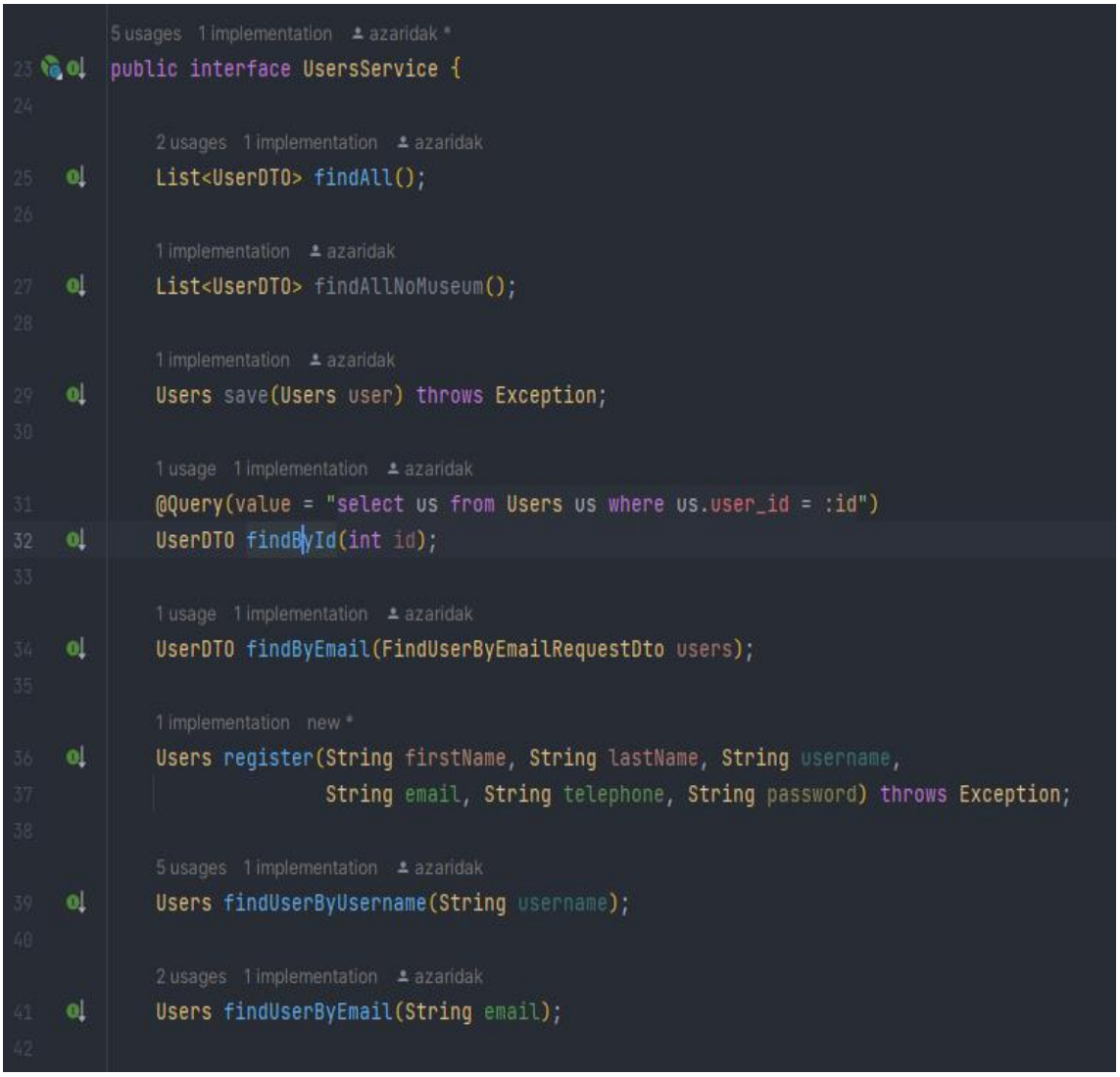
Στη συνέχεια, απεικονίζονται συνολικά πέντε Services. Συγκεκριμένα, το αρχείο LoginAttemptService θα αναλυθεί σε επόμενο στάδιο που θα αναφέρεται στον τρόπο σύνδεσης των χρηστών στο σύστημα. Αρχικά, ας ξεκινήσουμε με το EmailServiceImpl, το οποίο όπως προσδιορίζει και το όνομά του είναι υπεύθυνο για την αποστολή ηλεκτρονικών μηνυμάτων(email) από το σύστημα προς τους χρήστες. Επιπλέον, το CommonServiceImpl είναι ένα βοηθητικό Service που χρησιμοποιείται από τα άλλα Services του συστήματος, και περιέχει την υλοποίηση για τις διαδικασίες που χρειάζονται σε αρκετές περιπτώσεις. Για αυτό θεωρήθηκε σκόπιμο να υλοποιηθούν οι διαδικασίες σε ένα κοινό Service και να γίνεται επαναχρησιμοποίησή τους από το Service αυτό όποτε χρειάζεται. Όπως αναφέρθηκε, αυτό έχει το πλεονέκτημα ότι σε μία ενδεχόμενη αλλαγή των απαιτήσεων του συστήματος, θα χρειαστεί αλλαγή μόνο σε ένα σημείο και συγκεκριμένα στο CommonServiceImpl και από εκεί αυτόματα θα ενημερωθούν όλα τα σημεία που κάνουν χρήση την διαδικασία που ενημερώθηκε μέσα στο κοινό Service.

Τώρα θα αναλυθούν τα δύο κύρια Services, συγκεκριμένα το UserService και το MuseumService τα οποία χειρίζονται και εκτελούν την λειτουργεία των Χρηστών και των Μουσείων αντίστοιχα.

```
import ...

4 usages  1 implementation  azaridak *
public interface MuseumService {

    1 usage  1 implementation  azaridak
    List<MuseumDTO> findAll();

    1 usage  1 implementation  azaridak
    MuseumDTO findById(int id);

    1 usage  1 implementation  azaridak
    MuseumDTO findByEmail(Museum museum);

    1 implementation  new *
    MuseumDTO findByTitle(Museum museum);

    1 implementation  azaridak
    MuseumDTO updateMuseum(Museum museum) throws Exception;

    1 implementation  azaridak
    Museum save(Museum museum) throws Exception;
```

Εικόνα 15 - MuseumService

Στην εικόνα 15, απεικονίζονται οι υπογραφές από τις λειτουργίες που έχουν δηλωθεί και υλοποιηθεί από το MuseumService. Ενδεικτικά, είναι λειτουργίες όπως η ανάκτηση όλων των μουσείων, η αναζήτηση με κάποιο αναγνωριστικό για παράδειγμα τον τίτλο του μουσείου ή το email του μουσείου. Ακόμα, λειτουργία για ενημέρωση των στοιχείων του μουσείου και τέλος λειτουργία για την προσθήκη ενός νέου μουσείου στο σύστημα.



Εικόνα 16 - UserService

```
1 usage  1 implementation  new *
AddUserResponseDto addNewUser(AddUserRequestDto requestDto) throws UserNotFoundException,
        UsernameExistsException, EmailExistsException, IOException, UserBadInputException;

1 usage  1 implementation  azaridak
DeleteUserResponseDto deleteUser(DeleteUserRequestDto requestDto) throws Exception;

1 implementation  new *
void ResetPassword(ResetPasswordRequestDto requestDto) throws EmailNotFoundException,
        MessagingException;

1 usage  1 implementation  new *
UpdateUserResponseDto updateUser(UpdateUserRequestDto requestDto) throws UserNotFoundException,
        UsernameExistsException, EmailExistsException, IOException, UserBadInputException;

1 usage  1 implementation  new *
UserDTO UpdateProfileImage(String username, MultipartFile image) throws UserNotFoundException,
        UsernameExistsException, EmailExistsException, IOException, UserBadInputException;

1 usage  1 implementation  azaridak
CustomUserDetails getCustomUserDetails(UserDTO userDTO);
```

Εικόνα 17 - UserService

Στις εικόνες 16 και 17, απεικονίζονται οι υπογραφές των συναρτήσεων που έχουν υλοποιηθεί για τους σκοπούς της μεταπτυχιακής διατριβής. Οι λειτουργικότητες, αφορούν όλα τα είδη των χρηστών, όπως τον απλό επισκέπτη και μη συνδεδεμένο στο σύστημα, τον συνδεδεμένο απλό χρήστη αλλά και τον συνδεδεμένο χρήστη που είναι και ταυτόχρονα διαχειριστής ενός μουσείου. Επιπλέον, περιέχει και τις διαδικασίες που επιτρέπονται μόνο σε χρήστη που είναι super-admin όπως για παράδειγμα, η λειτουργία της διαγραφής κάποιου χρήστη από το σύστημα.

Το Service Layer είναι σε επικοινωνία με το κομμάτι των Controllers και το κομμάτι Persistence. Για το λόγο αυτό, βλέπουμε πως τα ορίσματα των συναρτήσεων είναι Data Transfer Objects (DTO) όπως έχει γίνει ήδη αναφορά. Οι διαδικασίες αυτές, είναι υπεύθυνες για την εφαρμογή και εκτέλεση της επιχειρησιακής λογικής της εφαρμογής και πάντοτε ανάλογα με τις εκάστοτε απαιτήσεις αυτές ολοκληρώνουν τα αιτήματα ακολουθώντας τους συγκεκριμένους κανόνες που ορίζονται από την επιχειρησιακή λογική. Στην περίπτωση που χρειαστούν να έχουν πρόσβαση σε αποθηκευμένα δεδομένα της εφαρμογής, ή να αποθηκεύσουν νέα δεδομένα ή να ενημερώσουν ήδη αποθηκευμένα δεδομένα, θα εφαρμόσουν το design pattern που αναφέρθηκε και θα χρησιμοποιήσουν τα Repositories από το Persistence layer. Σε ατυχές σενάριο που κάποια διαδικασία εγγραφής, διαγραφής ή ανάκτησης δεδομένων από την βάση δεδομένων της εφαρμογής αποτύχει να ολοκληρωθεί από το Persistence layer, τα Services διαθέτουν τους κατάλληλους μηχανισμούς ώστε να ενημερώσουν με φιλικό μήνυμα κατάλληλα τον χρήστη και να φροντίσουν να μην αποθηκευτούν ή διαγραφούν δεδομένα ενώ δεν έπρεπε λόγο τους σφάλματος. Επιπλέον, στην ίδια περίπτωση που μπορεί να προκύψει κάποιο σφάλμα, οι μηχανισμοί αυτοί σε συνεργασία με τα Repositories του Persistence layer, εξασφαλίζουν ταυτόχρονα ότι δεν θα υπάρχει λανθασμένη καταχώρηση δεδομένων στην εφαρμογή. Πιο συγκεκριμένα, αν προκύψει κάποιο exception μέσα στο σύστημα όλες οι εγγραφές που είχαν γίνει στη βάση δεδομένων από την διαδικασία που προκάλεσε το exception θα αντιστραφούν (rollback). Με τον τρόπο αυτό, εξασφαλίζεται η σωστή διαχείριση των δεδομένων σε περίπτωση σφάλματος του λογισμικού ή προσωρινού προβλήματος στη λειτουργία του δικτύου ή της βάσης δεδομένων.

Ακόμα, τα Services είναι υπεύθυνα για την εκτέλεση επικυρώσεων (validations) πάντα σύμφωνα με τις απαιτήσεις του συστήματος. Ένα παράδειγμα τέτοιας επικύρωσης είναι στην προσθήκη κάποιου νέου Μουσείου στο σύστημα όπου η διαδικασία θα πρέπει αρχικά να πιστοποιήσει πως ο χρήστης έχει δικαίωμα να κάνει προσθήκη νέου μουσείου. Σε αντίθετη περίπτωση, αν ο χρήστης δεν έχει δικαίωμα δημιουργίας νέου μουσείου θα πρέπει να μην επιτρέψει την διαδικασία και να επιστρέψει το κατάλληλο μήνυμα στον χρήστη. Όπως γίνεται λοιπόν σαφές, αυτό το layer εκτός από ότι είναι υπεύθυνο για την τήρηση και εκτέλεση των διαδικασιών με βάση όλων των απαιτήσεων της εφαρμογής, είναι και συνδετικός κρίκος ανάμεσα στη βάση δεδομένων και στους Controllers (Presentation Layer). Τέλος, για την επικοινωνία μεταξύ των layers Service και Persistence χρησιμοποιούνται απλές κλάσεις (Pojo models). Σε αυτές δεν περιλαμβάνεται επιχειρησιακή λογική, και υπάρχουν απλά για να γίνεται αντιστοίχιση των τιμών από την βάση δεδομένων σε προσωρινά αντικείμενα και να χρησιμοποιηθούν για πιθανούς υπολογισμούς που μπορεί να χρειαστούν. Έπειτα, αν χρειαστεί να γίνει εγγραφή δεδομένων στη βάση, αυτές οι κλάσεις μέσω μετατροπέων(adapters) μετατρέπονται σε Entity Classes. Αντίστοιχα, αν χρειαστούν να επιστραφούν στους Controllers και κατά επέκταση στο front-end κομμάτι της εφαρμογής, μέσω των κατάλληλων μετατροπέων μετατρέπονται σε αντικείμενα DTO.

```
4 usages  azaridak
public static UserDTO getUserDTOWithOutMuseum(Users db){
    UserDTO ret = new UserDTO();
    if(db != null){
        ret.setUser_id(db.getUser_id());
        ret.setUsername(db.getUsername());
        ret.setEmail(db.getEmail());
        ret.setPassword(db.getPassword());
        ret.setCreatedOn(db.getCreatedOn());
        ret.setFirstName(db.getFirstName());
        ret.setLastName(db.getLastName());
        ret.setGender(db.getGender());
        ret.setRole(db.getRole());
        ret.setMobile(db.getMobile());
        ret.setAuthorities(db.getAuthorities());
    }
    return ret;
}
```

Εικόνα 18 - Μετατροπή κλάσης User Entity σε DTO

Στην εικόνα 18, βλέπουμε ένα παράδειγμα από τους μετατροπείς που αναφέρθηκαν για μετατροπή της κλάσης Entity σε DTO και κατά επέκταση την επιστροφή της στο front-end κομμάτι.

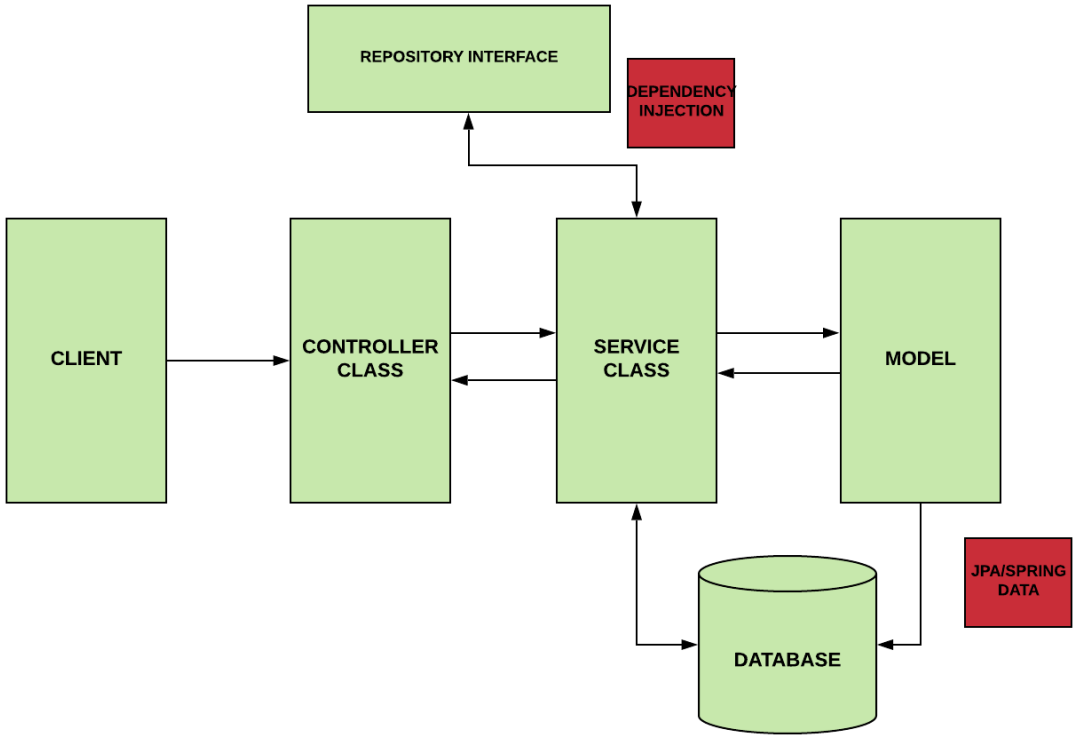
```
4 usages  azaridak
public static MuseumDTO getMuseumDTOWithOutUser(Museum db){
    MuseumDTO ret = new MuseumDTO();
    if(db != null){
        ret.setCity(db.getCity());
        ret.setMuseum_id(db.getMuseum_id());
        ret.setTitle(db.getTitle());
        ret.setEmail(db.getEmail());
        ret.setAddress(db.getAddress());
        ret.setGenre(db.getGenre());
        ret.setDescription(db.getDescription());
        ret.setShortDescription(db.getShortDescription());
        ret.setCity(db.getCity());
        ret.setTelephone(db.getTelephone());
        ret.setTelephone2(db.getTelephone2());
        ret.setUrl(db.getUrl());
        ret.setYoutubeUrl(db.getYoutubeUrl());
        ret.setFacebookUrl(db.getFacebookUrl());
        ret.setTwitterUrl(db.getTwitterUrl());
        ret.setUsers(getUserDTOWithOutMuseum(db.getUsers()));
    }
    return ret;
}
```

Εικόνα 19 - Μετατροπή Museum Entity σε DTO

Αντίστοιχα, στην εικόνα 19, βλέπουμε ένα παράδειγμα για την μετατροπή από κλάση Entity Museum σε DTO μουσείο. Σε αυτό το σημείο είναι σκόπιμο να σημειωθεί, πως είναι πολύ σημαντικό να μην επιστρέφει το επίπεδο Presentation (Controllers) αντικείμενα από Entity classes στο front-end. Τα DTO που πρέπει να επιστρέφονται, είναι πιο μικρά αντικείμενα σε μέγεθος που έχει ως αποτέλεσμα να μεταφέρονται λιγότερα δεδομένα στο διαδίκτυο και άρα να είναι πιο μικρές οι απαιτήσεις σε αυτό. Επιπλέον, με την χρήση των μετατροπέων που γίνεται, εξασφαλίζεται ότι δε θα επιστρέψουν με το Entity αντικείμενο όλα τα δεδομένα, αλλά μόνο αυτά που χρίζονται απαραίτητα και σε αυτά δίνεται τιμή. Με αυτό το τρόπο επίσης, γίνεται σίγουρο πως τα δεδομένα που θα εξέρχονται του συστήματος θα είναι ελεγχόμενα, και δε θα επιστρέψουν τιμές από στοιχεία που δεν θα έπρεπε.

3.2.4 Σχεδιασμός Επιπέδου Persistence (Repositories)

Στο κεφάλαιο αυτό θα αναλυθεί το επίπεδο Persistence(Repositories) και ο τρόπος που λειτουργεί. Όπως έχει γίνει κατανοητό, η επικοινωνία των Repositories γίνεται αποκλειστικά μεταξύ των Services και της βάσης δεδομένων. Πιο συγκεκριμένα, όπως απεικονίζεται και στην εικόνα 20, στις κλάσεις των Services, γίνεται έγχυση εξαρτήσεων(dependency injection) των Repositories. Με αυτό το τρόπο, τα Services αποκτούν επικοινωνία με την βάση.



Εικόνα 20 – Έγχυση εξαρτήσεων των Repositories (‘Spring Boot Architecture and Its Workflow’ n.d.)

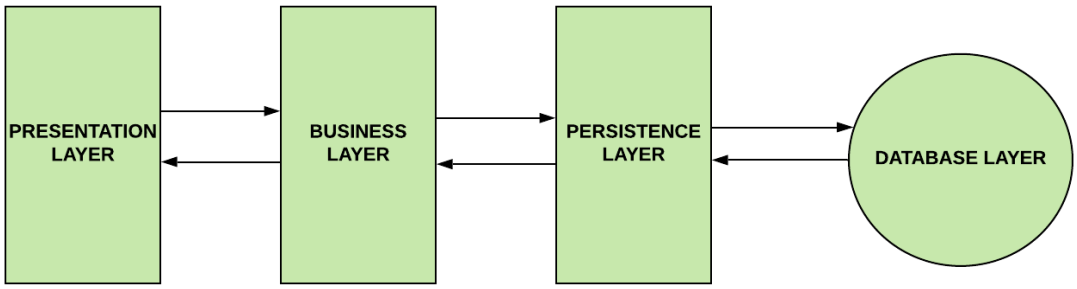
```
3 usages  azaridak *
@Repository
public interface UsersRepository extends JpaRepository<Users,Integer> {

    1 usage  azaridak
    Users findById(int id);

    1 usage  azaridak
    Users getFirstByEmail(String email);

    3 usages  azaridak
    Users findUsersByUsername(String username);

    2 usages  azaridak
    Users findUsersByEmail(String email);

    1 usage  azaridak
    void deleteByUsername(String username);
```

Εικόνα 21 - Users Repository

Στην εικόνα 21 απεικονίζεται ένα από τα Repositories του συστήματος και συγκεκριμένα του User. Περιέχει συναρτήσεις αλληλεπίδρασης με την βάση δεδομένων οι οποίες χρησιμοποιούνται είτε από το UserService είτε από το MuseumService με βάση πάντα, ποιο Service θέλει να χρησιμοποιήσει τα δεδομένα των χρηστών. Το Interface κάνει επέκταση(extend) το JpaRepository, το οποίο παρέχει την δυνατότητα για απευθείας χρήση συναρτήσεων προσπέλασης της βάσης αλλά ταυτόχρονα εξασφαλίζει και την σωστή αντιστοίχιση δεδομένων. Αυτό επιτυγχάνεται καθώς το JPA Repository είναι ένα ήδη υλοποιημένο API για την ανάκτηση, την εγγραφή και την διαγραφή δεδομένων από την βάση και η χρήση του εξασφαλίζει επίσης, μείωση του χρόνου εγγραφής και υλοποίησης του λογισμικού αλλά και συνολική σταθερότητα στο σύστημα. Είναι υλοποιημένο από το Spring και για την χρήση του αρκεί να δηλωθεί ως εξάρτηση στο Maven της εφαρμογής (‘JpaRepository (Spring Data JPA Parent 3.0.0 API)’ n.d.).

3.2.5 Συνολική Αρχιτεκτονική του API

Στο σημείο αυτό με βάση όσων έχουν αναφερθεί στο κεφάλαιο αυτό, θεωρείται σκόπιμο να γραφτούν δεδομένα σχετικά με την αρχιτεκτονική, τον διαχωρισμό των επιπέδων στο σύστημα, τα πλεονεκτήματα αλλά και πως έλαβαν όλα τα χαρακτηριστικά αυτά ρόλο στην σωστή δομή και την σωστή ανάπτυξη του λογισμικού στα πλαίσια της μεταπτυχιακής διατριβής.



Εικόνα 22 - Κύκλος ροής δεδομένων στο back-end (‘Spring Boot Architecture and Its Workflow’ n.d.)

Στην εικόνα 22, απεικονίζεται ένας πλήρης κύκλος ροής των δεδομένων μέσα στο back-end με την χρήση της αρχιτεκτονικής του Spring boot Framework. (‘Spring Boot Architecture and Its Workflow’ n.d.). Όπως έχει ήδη αναφερθεί στα προηγούμενα κεφάλαια, ο διαχωρισμός αυτός έχει πολλά πλεονεκτήματα και στα οποία θα γίνει ανάλυση τώρα.

Αρχικά, το Presentation Layer είναι το πρώτο επίπεδο στο σύστημα, και είναι υπεύθυνο για τα εξής:

- Να εκτελεί την αυθεντικοποίηση και την εξουσιοδότηση του χρήστη.
- Να μετατρέπει τα δεδομένα JSON σε αντικείμενα και το αντίστροφο.
- Να κάνει διαχείριση των HTTP requests που έρχονται στο API.
- Να μεταφέρει την αυθεντικοποίηση των χρηστών στο business layer επίπεδο.

Το Presentation Layer από τεχνική άποψη, πρακτικά είναι ισοδύναμο και με όλες τις κλάσεις Controllers μέσα από τις οποίες το σύστημα χειρίζεται και ανταποκρίνεται σε όλα τα εισερχόμενα

HTTP αιτήματα(requests)(GET, POST, PUT, DELETE) τα οποία λαμβάνει από τον εκάστοτε χρήστη που χρησιμοποιεί το API.

Εν συνεχεία, θα ακολουθήσουν τα βασικά χαρακτηριστικά του επιπέδου Business (Service)

- Εκτέλεση επικυρώσεων-νομιμοποίηση των λειτουργιών.
- Υπολογισμός και παροχή της εξουσιοδότησης για τις ζητούμενες ενέργειες του κάθε αιτήματος, βάσει των δικαιωμάτων του εκάστοτε χρήστη.
- Χειρισμός και εκτέλεση του business logic μέσω του κώδικα και των κανόνων.

Οι κλάσεις που είναι Services από τεχνικής άποψης, είναι ισοδύναμες με το Business Layer και επίσης είναι το σημείο όπου χειριζόμαστε την επιχειρησιακή λογική. Εδώ αξίζει να γίνει αναφορά στο τι είναι πιο συγκεκριμένα η επιχειρηματική λογική. Είναι το μέρος του συστήματος κατά το οποίο κωδικοποιούνται και ενσωματώνονται οι αληθινές απαιτήσεις που καθορίζονται με επιχειρησιακά κριτήρια. Επιπλέον, ορίζει πως τα αντικείμενα αλληλοεπιδρούν μεταξύ τους και επιβάλει τις κατευθύνσεις και την σειρά που θα ακολουθήσουν οι μέθοδοι του συστήματος ώστε τελικά να γίνει προσπέλαση των αντικειμένων/δεδομένων και να αποκτήσουν τις κατάλληλες τιμές.

Ακόμα, αξίζει να αναφερθεί και ο όρος επιχειρησιακοί κανόνες (business rules), ο οποίος πολλές φορές συγχέεται με το λογική(logic), όμως ο αποκλειστικός του σκοπός είναι να περιγράφει διαδικασίες, ορισμούς αλλά και ποιοι περιορισμοί λαμβάνουν χώρα στην εφαρμογή. Οι ενέργειες αυτές, σχηματίζουν συλλογικά μια διαδικασία που κάθε ενέργεια στο σύστημα ακολουθεί. Συνοπτικά λοιπόν, στον όρο επιχειρησιακή λογική στην τεχνολογία λογισμικού περιλαμβάνονται όλες οι αποφάσεις για το τι πρέπει να γίνει και αυτό γίνεται στα Services classes. Τέλος, το επίπεδο αυτό επικοινωνεί και με το επίπεδο των Controllers αλλά και με το Persistence Layer.

Persistence Layer είναι το επίπεδο που είναι υπεύθυνο για να:

- Περιέχει την επιχειρησιακή λογική στην αποθήκευση.
- Μετατρέπει και να μεταφράσει τα Java αντικείμενα σε γραμμές πινάκων δεδομένων να τα αποθηκεύει στη βάση και φυσικά το αντίστροφο δηλαδή, μετατροπή από την βάση σε αντικείμενα Java. Από γραμμές πινάκων της βάσης, τα μετατρέπει σε αντικείμενα Java για να τα επεξεργαστεί το σύστημα.

Όπως είναι κατανοητό, το επίπεδο αυτό είναι το μόνο που επικοινωνεί ταυτόχρονα με δύο επίπεδα, με το Business Layer και με τη βάση δεδομένων. Αυτό το επίπεδο επιλέχθηκε για τα πλεονεκτήματά που προσφέρει στην δομή του λογισμικού.

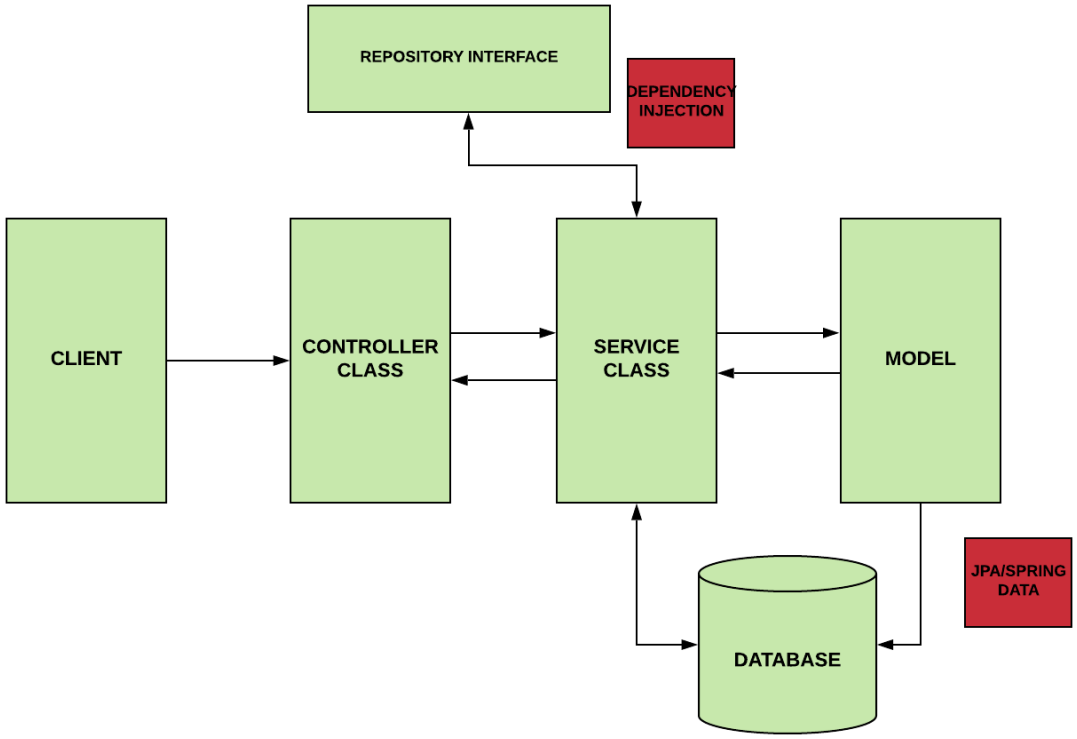
Τέλος, το Database Layer είναι υπεύθυνο για την εκτέλεση όλων των διαδικασιών προσπέλασης στη βάση του συστήματος CRUD(Create, Read, Update, Delete).

- Οι λειτουργίες πρόσβασης στη βάση έχουν διαχωριστεί από την επιχειρησιακή λογική. Έτσι, μπορεί να γίνει έλεγχος σωστής λειτουργίας και των δύο επιπέδων ανεξάρτητα. Αυτή η ανεξαρτησία προσφέρει ταχύτητα και ευελιξία στο κύκλο ανάπτυξης λογισμικού.
- Καθαρότητα, ευκολία συντήρησης και επαναχρησιμοποίηση κώδικα. Πιο αναλυτικά, το επίπεδο που χειρίζεται τις διαδικασίες CRUD είναι γενικευμένο και έτσι μπορεί να ξαναχρησιμοποιηθεί σε κάποιο άλλο έργο, που είναι σημαντικό πλεονέκτημα καθώς κερδίζει αρκετό χρόνο στον κύκλο της ανάπτυξης.
- Με αυτόν τον διαχωρισμό των επιπέδων, δύναται το καθένα να αναπτυχθεί και να υλοποιηθεί από δύο διαφορετικές ομάδες. Όμως, με αυτή την αρχιτεκτονική κρύβονται ευαίσθητες πληροφορίες από το επίπεδο των Services για το πως τα δεδομένα τελικά αποθηκεύονται στην βάση δεδομένων, και αντίστοιχα κρύβεται από το Persistence layer η επιχειρησιακή λογική καθώς δεν έχει πρόσβαση σε αυτή.

Κάποια από τα μειονεκτήματα που μπορούν να αναφερθούν είναι ότι με αυτό τον τρόπο γίνεται προσθήκη ενός έξτρα επιπέδου στην εφαρμογή. Η προσθήκη ενός επιπέδου στα πρώτα στάδια της ανάπτυξης ενδέχεται να προσθέσει μερική πολυπλοκότητα η οποία συνήθως θεωρείται

αχρείαστη ή υπερβολική σε μικρού μεγέθους εφαρμογές. Επιπλέον, σε κάθε Entity κλάση που θα θέλουμε να χαρτογραφήσουμε και να έχουμε πρόσβαση στην βάση, θα πρέπει να κάνουμε έγχυση ένα αντικείμενο των Repositories.

Με βάση όλα τα παραπάνω αξίζει να αναφερθούν συνοπτικά όλες οι λειτουργίες που γίνονται μέσα στο κύκλο λειτουργίας του Spring.



Εικόνα 23 - Επικοινωνία όλων των επιπέδων ('Spring Boot Architecture and Its Workflow' n.d.)

Στην εικόνα 23 βλέπουμε:

- Ο πελάτης πραγματοποιεί ένα HTTP αίτημα σε JSON μορφή.
- Κάποια από τις Controller class, λαμβάνει το αίτημα.
- Γίνεται αναγνώριση του request από τον controller, και μετά το διαχειρίζεται.
- Αν χρειαστεί προωθείται στο Service Layer/class.
- Η Service class, χειρίζεται το αίτημα με βάση πάντα την επιχειρησιακή λογική. Επίσης, όποτε χρειάζεται κάνει προσπέλαση δεδομένων από την βάση του συστήματος.
- Αν όλες οι διαδικασίες ολοκληρωθούν επιτυχώς, επιστρέφεται στο πελάτη ένα HTTP response με την μορφή JSON.

3.2.6 Διαδικτυακό τεκμήριο JSON - JWT (JSON Web Token)

Στο κεφάλαιο αυτό θα γίνει αναφορά στο JWT(JSON Web Token) καθώς χρησιμοποιείται μέσα στο σύστημά στα πλαίσια της μεταπτυχιακής διατριβής για την σύνδεση του χρήστη, την αυθεντικοποίηση αλλά και την πιστοποίησή του μέσα σε αυτό (auth0.com n.d.). Είναι ένα ανοιχτό πρότυπο που ορίζει έναν ευέλικτο καθώς και αυτόνομο-αυτοτελή τρόπο για την ασφαλή μετάδοση

πληροφοριών ανάμεσα σε συστήματα με την μορφή αντικειμένων τύπου JSON. Η πληροφορία που μεταφέρεται μπορεί να πιστοποιηθεί και να εξακριβωθεί επειδή είναι ηλεκτρονικά υπογεγραμμένη. Η ηλεκτρονική υπογραφή γίνεται χρησιμοποιώντας ένα κρυφό κλειδί με τον Αλγόριθμο HMAC ή με την χρήση της τεχνικής ζευγαριού δημόσιο–ιδιωτικό κλειδί. Η υλοποίηση στην μεταπτυχιακή διατριβή που αναλύουμε έγινε με τον πρώτο τρόπο δηλαδή το κρυφό κλειδί με αλγόριθμο HMAC. (‘JWT Token Generation for HMAC Using SHA Algorithm’ n.d.)

Παρόλο που το JWT μπορεί να παρέχει μυστικότητα ανάμεσα στις οντότητες που ανταλλάζουν δεδομένα, η εστίαση θα γίνει στα υπογεγραμμένα tokens (signed tokens). Με την χρήση τους, γίνεται η πιστοποίηση πως όλα τα αιτήματα και οι απαντήσεις κατά την ανταλλαγή μηνυμάτων είναι ακέραια ως προς την ταυτότητα του αποστολέα, αλλά επιπλέον πως και τα στοιχεία που έδωσε για αυθεντικοποίηση είναι τα πραγματικά. Ένας από τους λόγους που το χρησιμοποιούμε είναι όπως αναφέρθηκε η αυθεντικοποίηση. Μόλις ο χρήστης πραγματοποιήσει την επιτυχές σύνδεση στο σύστημα, από εκείνο το στάδιο και μετά, κάθε αίτημα από και προς τον server (API) θα έχει ενσωματωμένο το JTW. Με αυτό τον τρόπο, η χρήση του JWT επιτρέπει και στο front-end και στο back-end σύστημα να γνωρίζει τα απαραίτητα δεδομένα για τον χρήστη.

Η δομή του JWT αποτελείται από τρία μέρη. Το Header, το Payload και τέλος την Signature χωρισμένα μεταξύ τους απλά με μία τελεία «.». Το πρώτο μέρος το Header συνήθως αποτελείται από δύο μέρη. Τον τύπο του token (που είναι “JWT”) και τον τύπου του αλγόριθμου που χρησιμοποιήθηκε για την κρυπτογράφηση για παράδειγμα ο HMAC. Έπειτα, το κομμάτι του JSON αυτό γίνεται κωδικοποίηση σε Base64URL για να σχηματίσει τελικά το πρώτο κομμάτι του JWT. Στη συνέχεια, στο δεύτερο μέρος βρίσκεται όπως αναφέραμε το Payload. Σε αυτό περιέχονται όλα τα δικαιώματα ή οι αξιώσεις που θέλει ο χρήστης να μεταφέρει με το αίτημά του. Αυτά λοιπόν, είναι δηλώσεις για ένα αντικείμενο στις περισσότερες περιπτώσεις, όπως συμβαίνει και στα πλαίσια αυτής της μεταπτυχιακής διατριβής είναι ένα αντικείμενο χρήστη. Υπάρχουν τρία είδη αξιώσεων. Οι Εγγεγραμμένες αξιώσεις (Registered claims) που είναι μη υποχρεωτικές, αλλά υπάρχει ισχυρή σύσταση να γίνεται η χρήση τους γιατί παρέχουν μία ισχυρή συλλογή από δεδομένα όπως iss (issues), exp (expiration time), sub (subject). Τα αναφερόμενα claims χρησιμοποιούνται και στα πλαίσια της μεταπτυχιακής διατριβής. Τα δημόσια αξιώματα (Public Claims), τα οποία μπορούν να οριστούν με ό,τι τιμές ορίζει ο χρήστης κατά την δημιουργία του JWT. Τέλος, είναι τα ιδιωτικά αξιώματα (Private Claims) τα οποία είναι ιδιωτικά και πρακτικά κατασκευασμένα έπειτα από συμφωνία μεταξύ των συστημάτων που θα ανταλλάζουν μηνύματα. Το Payload επίσης, γίνεται σε κωδικοποίηση Base64Url και ολοκληρώνει το δεύτερο κομμάτι του JWT.

Είναι σημαντικό να αναφερθεί, ότι επειδή τα πρώτα δύο μέρη του JTW είναι ορατά και μπορούν να διαβαστούν από τρίτες εφαρμογές, δεν πρέπει να τοποθετούνται σε αυτά ευαίσθητες πληροφορίες εκτός αν είναι κρυπτογραφημένες. Τέλος, το τρίτο κομμάτι είναι η υπογραφή (signature). Για να δημιουργηθεί η υπογραφή, χρησιμοποιείται το κωδικοποιημένο πρώτο μέρος Header, το επίσης κωδικοποιημένο δεύτερο μέρος Payload, ένα κρυφό μυστικό που ορίζεται από τον χρήστη το οποίο πρέπει να παραμένει πάντα κρυφό και ο αλγόριθμος που δηλώθηκε πως θα χρησιμοποιηθεί στο Header. Με όλα αυτά δημιουργείται η υπογραφή, στο παράδειγμά μας λοιπόν είναι:

```
HMACSHA512 (
  Base64UrlEncode(header) + “.” +
  Base64UrlEncode(payload) + “.” +
  Secret).
```

Ο λόγος ύπαρξης της υπογραφής είναι για να πιστοποιηθεί και να εξασφαλιστεί πως κατά την διάρκεια που το JWT μεταφερόταν από τον έναν κόμβο στον άλλον δεν έχει αλλαχθεί κάποια τιμή του είτε από συστημικό λάθος είτε από κάποια κακόβουλη επίθεση.

Ποιος είναι ο τρόπος όμως που γίνεται χρήσιμο το JWT μέσα στο σύστημα; Για την αυθεντικοποίηση του χρήστη, αφού κάνει επιτυχώς σύνδεση δημιουργείται από το back-end κομμάτι της εφαρμογής ένα JWT και επιστρέφεται στο front-end. Θα πρέπει να γίνει με πολύ προσοχή ο σχεδιασμός με τον οποίο θα μπουν τα δεδομένα του χρήστη στο JWT καθώς είναι

σημαντικό να περιοριστούν όσο περισσότερο γίνεται τα κενά ασφαλείας. Κατά γενική περίπτωση, τα tokens θα πρέπει να τα κρατάει το σύστημα για όσο χρονικό διάστημα χρειάζεται και μετά να καταστρέφονται από το σύστημα.

Σε κάθε περίπτωση που ο χρήστης του front-end μέρος της εφαρμογής θέλει να προχωρήσει σε μία προστατευμένη σελίδα και σε προστατευμένα δεδομένα που πρέπει να είναι συνδεδεμένος, το σύστημα θα πρέπει να στείλει ένα JWT με το πιστοποιητικό Header χρησιμοποιώντας την δομή Bearer. Αυτός, είναι ένας τρόπος για να πραγματοποιηθεί πιστοποίηση καθώς και αυθεντικοποίηση χωρίς κατάσταση. Τα προστατευμένα δεδομένα του Server θα ελέγχουν ότι το JWT που τους έστειλε το front-end μέρος της εφαρμογής είναι έγκυρα και επίσης πως έχει τα κατάλληλα δικαιώματα έτσι ώστε να έχει πρόσβαση στα δεδομένα που αιτήθηκε. Σε περίπτωση που δεν έχει άδεια πρόσβασης στα δεδομένα που αιτήθηκε θα επιστρέφεται μήνυμα λάθους.

Τώρα θα αναφέρουμε γιατί είναι καλή πρακτική να χρησιμοποιούμε τα JWT και αργότερα θα γίνει επίδειξη στο πως έχουν εφαρμοστεί στα πλαίσια της παρούσας μεταπτυχιακής διατριβής. Αρχικά, είναι αρκετά απλό στη δημιουργία, στην συντήρηση, στην ενημέρωση των στοιχείων που θα αποθηκεύονται σε αυτό και ταυτόχρονα προσφέρει πολύ μεγάλη ασφάλεια. Ακόμα, είναι πάρα πολύ αποδοτικό και δεν προσθέτει πρόσθετο χρόνο εκτέλεσης στην εφαρμογή για δύο λόγους. Ο πρώτος είναι το πολύ μικρό του μέγεθος, με αποτέλεσμα η μεταφορά του στο διαδίκτυο να μην επιβαρύνει συνολικά τον χρόνο μεταφοράς των υπόλοιπων δεδομένων της εφαρμογής. Ο δεύτερος λόγος είναι, οι JSON parsers που διαβάζουν το token πλέον είναι πολύ εξελιγμένοι, έτσι είναι πάρα πολύ γρήγορο και αποδοτικό να διαβαστεί και να αποκωδικοποιηθεί τόσο από το front-end κομμάτι αλλά και φυσικά και από το back-end. Τέλος, σε σύγκριση με αντίστοιχες τεχνοτροπίες όπως Simple Web Tokens(SWT) και Security Assertion Markup Language Tokens(SAML) το JWT είναι αρκετά πιο ελαφρύ στην δημιουργία και αποστολή από το front-end στο back-end αλλά και ποιο ασφαλές.

3.2.7 Αυθεντικοποίηση και Εξουσιοδότηση (Authentication and Authorization)

Σε αυτό το σημείο, θα γίνει ανάλυση των δύο εννοιών αλλά και πως υλοποιήθηκαν στο σύστημα για την μεταπτυχιακή διατριβή. Αρχικά, να γίνει ένας ορισμός για την διαφορά που έχουν αυτές οι δύο έννοιες μεταξύ τους. Με την αυθεντικοποίηση το σύστημα καταλαβαίνει και αναγνωρίζει ποιος είναι ο χρήστης που πραγματοποίησε την είσοδο στο σύστημα με τα στοιχεία σύνδεσής του. Αυτό συχνά γίνεται με την χρήση του ονόματος χρήστη και τους κωδικού πρόσβασης που δηλώνει κατά την εγγραφή στην πλατφόρμα, και έτσι έχει υλοποιηθεί και στο σύστημά μας. Έχει λοιπόν πλέον δεδομένα όπως για παράδειγμα το email του, το όνομά του. Η υλοποίηση αυτού έγινε με την χρήση OAAuth 2.0 που ενσωματώνει τις πληροφορίες πρόσβασης του χρήστη σε ένα token.

Το σύστημα τώρα, πρέπει να μπορεί να πάρει αποφάσεις για το αν οι ενέργειες ή τα δεδομένα που αιτείται ο χρήστης πρέπει να του επιτραπούν ή όχι. Για να πραγματοποιηθεί αυτό, χρησιμοποιείται η Εξουσιοδότηση. Αναλυτικά, μόλις ο χρήστης πραγματοποιήσει σύνδεση, θα δημιουργηθεί και αποσταλεί ένα JWT από το back-end στο front-end. Το JWT θα περιέχει μέσα ένα access token το οποίο αφού το λάβει για πρώτη φορά το front-end θα το στέλνει έπειτα μαζί με κάθε επόμενο αίτημα που θα κάνει όσο ο χρήστης είναι συνδεδεμένος και χρησιμοποιεί το σύστημα. Με την ανάγνωση αυτού του token μέσα στο αίτημα, το back-end θα αναγνωρίζει τον χρήστη και θα μπορεί να κάνει εξουσιοδότηση αναγνωρίζοντας τι δικαιώματα πρόσβασης έχει καθώς είναι αποθηκευμένα στη βάση δεδομένων. Να σημειωθεί, υπάρχει δυνατότητα τα δικαιώματα χρήσης να τα παίρνει από κάποιο άλλο εξωτερικό σύστημα. Όμως, στα πλαίσια της παρούσας μεταπτυχιακής διατριβής αυτά βρίσκονται αποθηκευμένα στην βάση δεδομένων της εφαμοργής.

Ας αρχίσουμε λοιπόν, να βλέπουμε πως έγινε υλοποίηση μέσα στο σύστημα της διατριβής.

```java
azaridak
@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true,jsr250Enabled = true)
public class SecurityConfiguration extends WebSecurityConfigurerAdapter {

    2 usages
    private BCryptPasswordEncoder bCryptPasswordEncoder;
    2 usages
    private JwtAuthorizationFilter jwtAuthorizationFilter;
    2 usages
    private JwtAccessDeniedHandler jwtAccessDeniedHandler;
    2 usages
    private JwtAuthenticationEntryPoint jwtAuthenticationEntryPoint;
    2 usages
    private UsersServiceImpl usersService;


    azaridak
    @Autowired
    public SecurityConfiguration(
                                 JwtAuthorizationFilter jwtAuthorizationFilter,
                                 JwtAccessDeniedHandler jwtAccessDeniedHandler,
                                 BCryptPasswordEncoder bCryptPasswordEncoder,
                                 JwtAuthenticationEntryPoint jwtAuthenticationEntryPoint,
                                 @Qualifier("userDetailsService") UsersServiceImpl usersService){
        this.bCryptPasswordEncoder = bCryptPasswordEncoder;
        this.jwtAccessDeniedHandler = jwtAccessDeniedHandler;
        this.jwtAuthorizationFilter = jwtAuthorizationFilter;
        this.jwtAuthenticationEntryPoint = jwtAuthenticationEntryPoint;
        this.usersService = usersService;

    }

    azaridak
    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception{
        auth.userDetailsService(usersService).passwordEncoder(bCryptPasswordEncoder);

    }
```

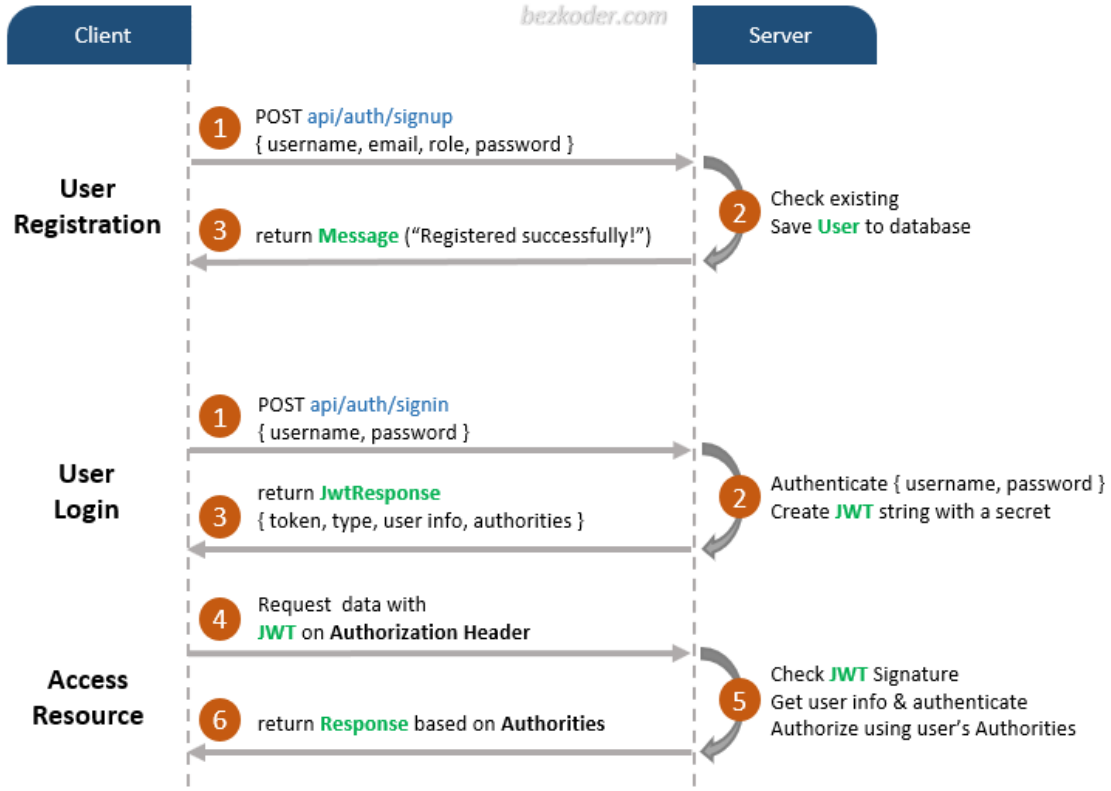
Εικόνα 24 – Αυθεντικοποίηση(Authentication)

```java
@Override
protected void configure(HttpSecurity http) throws Exception {
        http.csrf().disable() HttpSecurity
            .cors().and()//.disable()
            .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS) SessionMana
            .and() HttpSecurity
            .authorizeRequests().antMatchers(SecurityConstant.PUBLIC_URLS).permitAll() ExpressionUrlA
            .anyRequest().authenticated()
            .and() HttpSecurity
            .exceptionHandling() ExceptionHandlingConfigurer<HttpSecurity>
            .accessDeniedHandler(jwtAccessDeniedHandler)
            .authenticationEntryPoint(jwtAuthenticationEntryPoint)
            .and() HttpSecurity
            .addFilterBefore(jwtAuthorizationFilter, UsernamePasswordAuthenticationFilter.class);
}

1 usage  azaridak
@Bean
@Override
public AuthenticationManager authenticationManagerBean() throws Exception{
    return super.authenticationManagerBean();
}
```

Εικόνα 25 – Διαχειριστής για την Αυθεντικοποίηση (Authentication Manager)

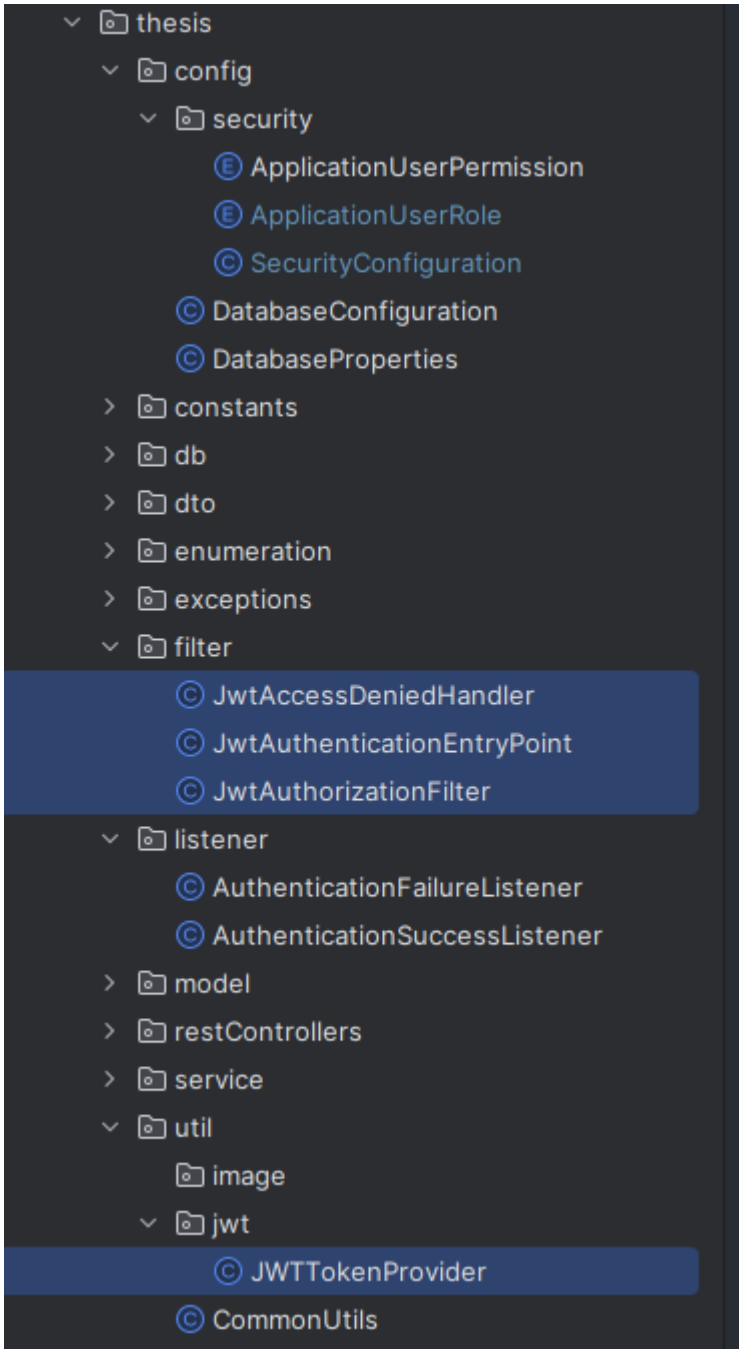
Στις εικόνες 24 και 25 απεικονίζεται ο τρόπος που γίνονται αυτά που αναφέρθηκαν παραπάνω. Με το annotation @EnableWebSecurity, παρέχονται οι απαραίτητες οδηγίες στο Spring Framework ώστε να χρησιμοποιήσει τις μεθόδους και τις ιδιότητες για ασφάλεια(security) που δηλώνονται σε αυτή την κλάση. Επίσης, δηλώνουμε τα GET requests να επιτρέπονται πριν την

αυθεντικοποίηση(authentication) του χρήστη καθώς αυτά χρησιμοποιούνται και για την ανταλλαγή των JWT και πρέπει να είναι όλα διαθέσιμα και ελεύθερα. Επιπλέον, δηλώνουμε ποια Urls ή endpoints στην εφαρμογή είναι δημόσια(public) δηλαδή, έχουν ελεύθερη πρόσβαση για όλους τους χρήστες και χωρίς να έχει πραγματοποιηθεί σύνδεση. Δηλώνεται επίσης πως η αυθεντικοποίηση θα γίνεται με όνομα χρήστη και κωδικό χρήστη, και πως οι σελίδες πρέπει να φιλτράρονται για JWT να γίνεται με τα custom JWT που έχουν υλοποιηθεί.



Εικόνα 26 - Δημιουργία και ανταλλαγή JWT ('[Spring Security] - Spring Boot Security JWT Authentication' 2021)

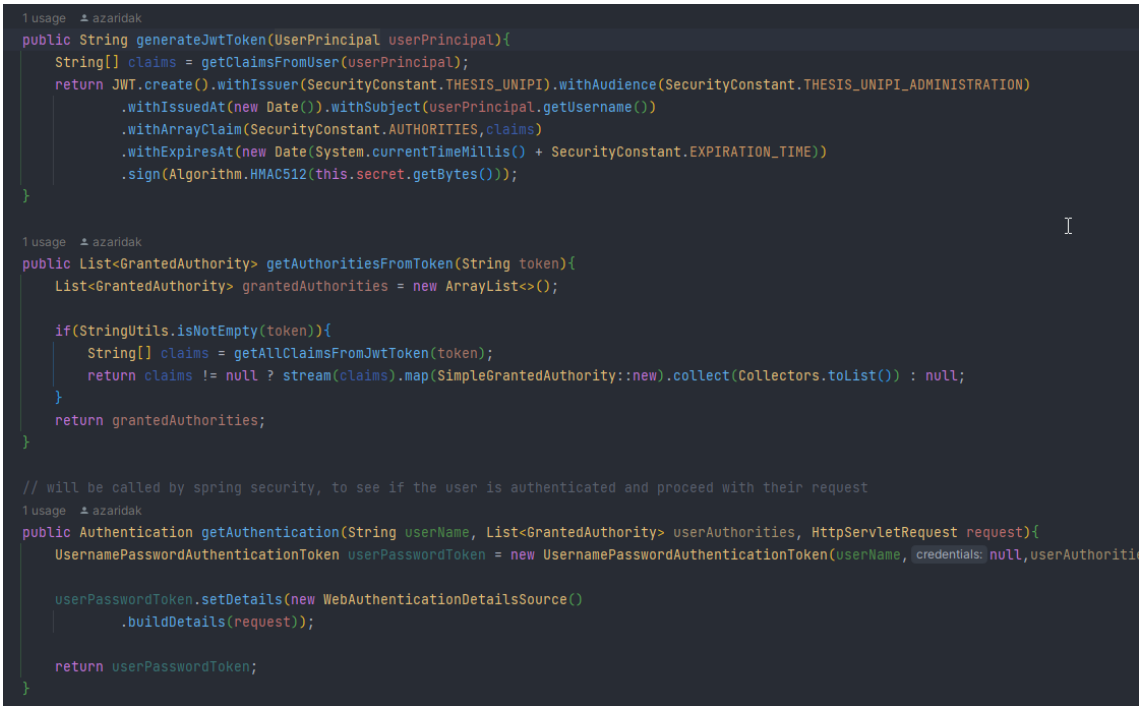
Για να γίνει πιο κατανοητό κάθε πότε γίνεται ανταλλαγή του JWT, όπως φαίνεται στην εικόνα 26, τα βήματα 4, 5 και 6 όπως αναφέρθηκε ήδη, από την στιγμή που ο χρήστης πραγματοποιήσει είσοδο στο σύστημα, από εκεί και πέρα για κάθε request που πραγματοποιείται, μέχρι ο χρήστης να πραγματοποιήσει αποσύνδεση από το σύστημα θα επαναλαμβάνονται. Ακόμα να σημειωθεί, πως αν ο χρόνος ζωής(expiration time) του JWT παρέλθει τότε το back-end σύστημα δε θα τον αναγνωρίσει ως συνδεδεμένο, και θα πρέπει να συνδεθεί στο σύστημα εκ νέου. Ας δούμε πως έχουν υλοποιηθεί αυτές οι λειτουργίες σε τεχνικό επίπεδο μέσω του κώδικα.



Εικόνα 27 - Υλοποίηση JWT

Στην εικόνα 27, πρέπει να δώσουμε προσοχή στο πακέτο filter και τις τρεις κλάσεις που έχει μέσα. Στο πακέτο listener καθώς και στο πακέτο jwt που έχει μέσα την κλάση JWTTokenProvider. Αρχικά, θα αναλύσουμε την κλάση JWTTokenProvider η οποία έχει υλοποιημένες τις λειτουργίες που αφορούν το JWT μέσα στο σύστημα. Ξεκινώντας με την πιο βασική από όλες, κάνει την δημιουργία (generation) με τον τρόπο και την δομή που έχει ήδη αναφερθεί δηλαδή header.payload.secret. Πιο συγκεκριμένα, τα γνωρίσματα που δηλώθηκαν όπως αυτά για την υπογραφή δηλαδή, ο αλγόριθμος HMAC512 και τέλος ο χρόνος ζωής που ορίστηκε να διαρκεί το token μέχρι να λήξει είναι πέντε ημέρες δηλωμένο σε χιλιοστά του δευτερολέπτου. Επίσης, μέσα

εισάγονται κρυπτογραφημένα τα δικαιώματα πρόσβασης του χρήστη στο σύστημα, και το όνομα χρήστη (username) όπως φαίνεται και στην εικόνα 28.



```
1 usage  azaridak
public String generateJwtToken(UserPrincipal userPrincipal){
    String[] claims = getClaimsFromUser(userPrincipal);
    return JWT.create().withIssuer(SecurityConstant.THESIS_UNIPI).withAudience(SecurityConstant.THESIS_UNIPI_ADMINISTRATION)
            .withIssuedAt(new Date()).withSubject(userPrincipal.getUsername())
            .withArrayClaim(SecurityConstant.AUTHORITIES,claims)
            .withExpiresAt(new Date(System.currentTimeMillis() + SecurityConstant.EXPIRATION_TIME))
            .sign(Algorithm.HMAC512(this.secret.getBytes()));
}

1 usage  azaridak
public List<GrantedAuthority> getAuthoritiesFromToken(String token){
    List<GrantedAuthority> grantedAuthorities = new ArrayList<>();

    if(StringUtils.isNotEmpty(token)){
        String[] claims = getAllClaimsFromJwtToken(token);
        return claims != null ? stream(claims).map(SimpleGrantedAuthority::new).collect(Collectors.toList()) : null;
    }
    return grantedAuthorities;
}

// will be called by spring security, to see if the user is authenticated and proceed with their request
1 usage  azaridak
public Authentication getAuthentication(String userName, List<GrantedAuthority> userAuthorities, HttpServletRequest request){
    UsernamePasswordAuthenticationToken userPasswordToken = new UsernamePasswordAuthenticationToken(userName, credentials: null,userAuthoriti

    userPasswordToken.setDetails(new WebAuthenticationDetailsSource()
            .buildDetails(request));

    return userPasswordToken;
}
```

Εικόνα 28 - Authentication και Authorization του χρήστη

Στην εικόνα 28, απεικονίζονται οι δύο συναρτήσεις με τις οποίες μετά την δημιουργία του token, το διαβάζουν και έπειτα αναγνωρίζουν από αυτό την αυθεντικοποίηση του χρήστη και στη συνέχεια την πιστοποίηση. Η αυθεντικοποίηση εκτελείται από την συνάρτηση «getAuthentication» και η πιστοποίηση αντίστοιχα από την «getAuthoritiesFromToken». Θα αναλυθεί περισσότερο στο σημείο που θα αναλυθεί η κλάση JwtAuthorizationFilter. Συνεχίζοντας στην ίδια κλάση, θα δούμε πως υπάρχουν επίσης κάποιες συναρτήσεις βοηθητικές (utilities) οι οποίες χρησιμοποιούνται μέσα στο σύστημα για τον έλεγχο του JWT. Όπως φαίνονται και στην εικόνα 29, υπάρχει συνάρτηση «isTokenExpired» η οποία δέχεται για όρισμα ένα token και αναγνωρίζει αν έχει παρέλθει ο χρόνος ζωής του ή όχι. Επιπλέον, «getAllClaimsFromJwtToken» και η «getClaimsFromUser». Η πρώτη, δέχεται ένα token και επιστρέφει έναν πίνακα με τα claims που είχαν αποθηκευτεί στο token και η δεύτερη εκτελεί την ίδια διαδικασία όμως τραβάει δεδομένα από το UserPrincipal αντί για token (χρησιμοποιείται δηλαδή με δεδομένα από την βάση και όχι από το token).

```java
public boolean isTokenValid(String token,String userName){
    if(StringUtils.isNotEmpty(userName) && StringUtils.isNotEmpty(token)){
        return !isTokenExpired(this.getJWTVerifier(), token);
    }
    return false;
}

1 usage  azaridak
private boolean isTokenExpired(JWTVerifier verifier, String token){
    Date expiration = verifier.verify(token).getExpiresAt();
    return expiration.before(new Date());
}

1 usage  azaridak
public String getSubject(String token) { return this.getJWTVerifier().verify(token).getSubject(); }

1 usage  azaridak
private String[] getAllClaimsFromJwtToken(String token){
    if(StringUtils.isNotEmpty(token)){
        JWTVerifier verifier = this.getJWTVerifier();
        return verifier.verify(token).getClaim(SecurityConstant.AUTHORITIES).asArray(String.class);
    }
    return null;
}

3 usages  azaridak
public JWTVerifier getJWTVerifier(){
    JWTVerifier verifier;
    try {
        Algorithm algorithm = Algorithm.HMAC512(this.secret);
        verifier = JWT.require(algorithm).withIssuer(SecurityConstant.THESIS_UNIPI).build();
    }catch (JWTVerificationException ex){
        throw new JWTVerificationException(SecurityConstant.TOKEN_CANNOT_BE_VERIFIED);
    }
    return verifier;
}
```

Εικόνα 29 - JWT verifier

Στην συνέχεια, όπως αναφέρθηκε παραπάνω τώρα θα γίνει αναφορά και ανάλυση στις δύο κλάσεις «JwtAuthenticationEntryPoint» «JwtAuthorizationFilter» που είναι επίσης για το JWT και βρίσκονται στο πακέτο filter μαζί με την «JwtAccessDeniedHandler». Με την άφιξη κάποιου αιτήματος από το front-end στο back-end ενεργοποιείται η πρώτη κλάση και εφόσον το request είναι σωστό τότε το προωθεί στην δεύτερη κλάση. Εκεί γίνεται με ακρίβεια η ανάλυση του αιτήματος όσο αναφορά τις λειτουργίες και τα δεδομένα του token. Λόγο της κομβικής σημασίας αυτής της κλάσης, θα αναλυθεί πιο εκτεταμένα. Η ροή εκτέλεσης ξεκινάει με την αναγνώριση του αιτήματος του HTTP request. Αν είναι τύπου «Options» τότε επιτρέπει να συνεχιστεί σε κάθε περίπτωση. Σε διαφορετική περίπτωση, ελέγχει πρώτα στο request, αν υπάρχει token. Αν υπάρχει token, χρησιμοποιεί το header του και λαμβάνει τα δεδομένα της αυθεντικοποίηση. Αν μέσα τους περιέχουν το keyword «Bearer » σημαίνει ότι είναι token από πιστοποιημένο και ανήκει σε συνδεδεμένο χρήστη άρα θα πρέπει να γίνει ανάγνωση και πιστοποίηση ολόκληρου του token ώστε το σύστημα να είναι σίγουρο ότι δεν έχει κενά ασφαλείας και ο χρήστης που στέλνει το αίτημα είναι στην πραγματικότητα αυτός που δηλώνει.

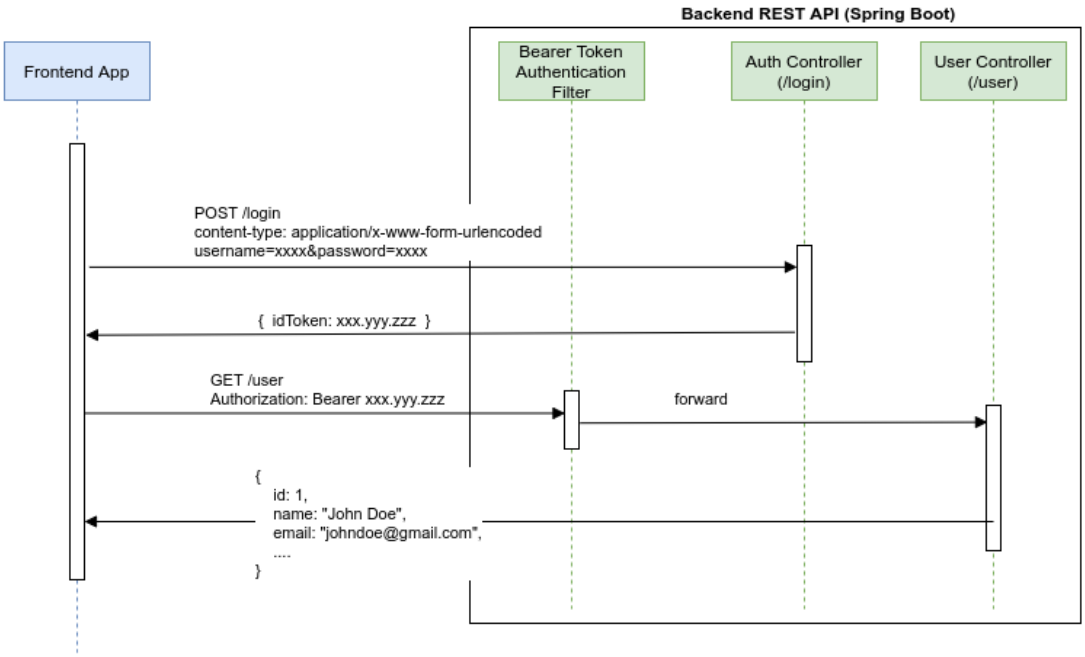
Συνεχίζοντας, αν τελικά το Bearer περιέχεται, συνεχίζει ο έλεγχος και γίνεται ανάκτηση του ονόματος χρήστη(username) από το token. Έπειτα, στο στάδιο αυτό με την χρήση του «JwtTokenProvider» γίνεται verify πως είναι σωστά με βάση τον αλγόριθμο κρυπτογράφησης και

το κρυφό κλειδί. Αν είναι επιτυχής η πιστοποίηση, αναγνωρίζονται και επιστρέφονται λοιπόν τα δικαιώματα χρήσης του χρήστη στην εφαρμογή που είναι αποθηκευμένα στη βάση δεδομένων. Σε περίπτωση που δεν είναι επιτυχής η ταυτοποίηση, το αίτημα θεωρείται ως «unauthorized» και δεν του επιτρέπεται να συνεχίσει μέσα στο σύστημα. Η κλάση «JwtAccessDeniedHandler» αναλαμβάνει σε αυτή τη περίπτωση και επιστρέφει το κατάλληλο μήνυμα που έχουμε ορίσει όταν γίνονται αιτήματα στο back-end με λάθος στοιχεία. Ακολουθεί εικόνα για την κλάση που υλοποιεί όσα αναφέρθηκαν.

```
1 azaridak
@Override
protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain) throu

    // if http option allow it, option request is just asking information about the server
    // ,and it sent before the actual request
    if(request.getMethod().equalsIgnoreCase(SecurityConstant.OPTIONS_HTTP_METHOD)){
        response.setStatus(HttpStatus.OK.value());
    }
    else {
        String authorizationHeader = request.getHeader(HttpHeaders.AUTHORIZATION); // string of authorization header
        if(authorizationHeader == null || !authorizationHeader.startsWith(SecurityConstant.TOKEN_PREFIX)){
            filterChain.doFilter(request,response);
            return;
        }

        String token = authorizationHeader.substring(SecurityConstant.TOKEN_PREFIX.length()); // the token without the
        String username = jwtTokenProvider.getSubject(token);

        if(jwtTokenProvider.isTokenValid(token,username) ){
            List<GrantedAuthority> authorities = jwtTokenProvider.getAuthoritiesFromToken(token);
            Authentication  authentication  = jwtTokenProvider.getAuthentication(username,authorities,request);
            SecurityContextHolder.getContext().setAuthentication(authentication);
        }
        else { // clear the context
            SecurityContextHolder.clearContext();
        }
    }
    filterChain.doFilter(request,response); // request continues
}
}
```

Εικόνα 30 – Φίλτρα των αιτημάτων για JWT (Filter των requests για JWT)

Στην συνέχεια, αφού ο χρήστης πιστοποιηθεί και συνδεθεί στο σύστημα το αίτημα φτάνει στο επίπεδο των Controllers που απεικονίζεται πιο αναλυτικά στο σχήμα της εικόνας 31.



Εικόνα 31 - Επικοινωνία με αυθεντικοποιημένο χρήστη ('JWT Authentication with Spring Boot Resource Server | by Imesha Sudasingha | The Startup' n.d.)

Σε αυτό το σημείο, το αίτημα βρίσκεται πλέον στο Presentation Layer. Εδώ υπάρχουν επίσης δύο περιπτώσεις. Η πρώτη είναι το αίτημα αν αναφέρεται σε έναν Controller ο οποίος είναι δημόσιος. Δηλαδή, μπορεί να έχει πρόσβαση ο κάθε χρήστης χωρίς να χρειάζεται να είναι πιστοποιημένος. Τότε τα πράγματα είναι σχετικά απλά, ο Controller λαμβάνει το αίτημα και στη συνέχεια το προωθεί στο Service Layer. Τότε η διαδικασία συνεχίζει σύμφωνα με όσα έχουν ήδη αναφερθεί στα προηγούμενα κεφάλαια. Στην δεύτερη περίπτωση όμως, το αίτημα από το front-end μπορεί να απευθύνεται σε έναν Controller ο οποίος για να επιτρέψει την είσοδο θα πρέπει ο χρήστης αρχικά να έχει κάνει αυθεντικοποίηση και έπειτα να μπορεί να πιστοποιηθεί πως έχει δικαίωμα για την λειτουργία που ζήτησε.

Για παράδειγμα, θα μελετήσουμε το σενάριο που ο χρήστης του front-end επιχειρήσει να κάνει διαγραφή κάποιου άλλου χρήστη από το σύστημα. Η επιλογή αυτής της ενέργειας έγινε με βάση πως είναι μια σημαντική τροποποίηση στα στοιχεία του συστήματος άρα αξίζει να αναλυθεί. Ξεκινώντας λοιπόν, φυσικά ο χρήστης για να αιτηθεί διαγραφή κάποιου άλλου χρήστη θα πρέπει να είναι συνδεδεμένος. Όμως, να είναι απλά συνδεδεμένος για την ενέργεια της διαγραφής δεν αρκεί γιατί με αυτό τον τρόπο, ένας απλός χρήστης που είχε κάνει εγγραφή στο σύστημα θα μπορούσε να διαγράψει άλλους χρήστες και γενικά να επηρεάζει κακόβουλα την εφαρμογή.

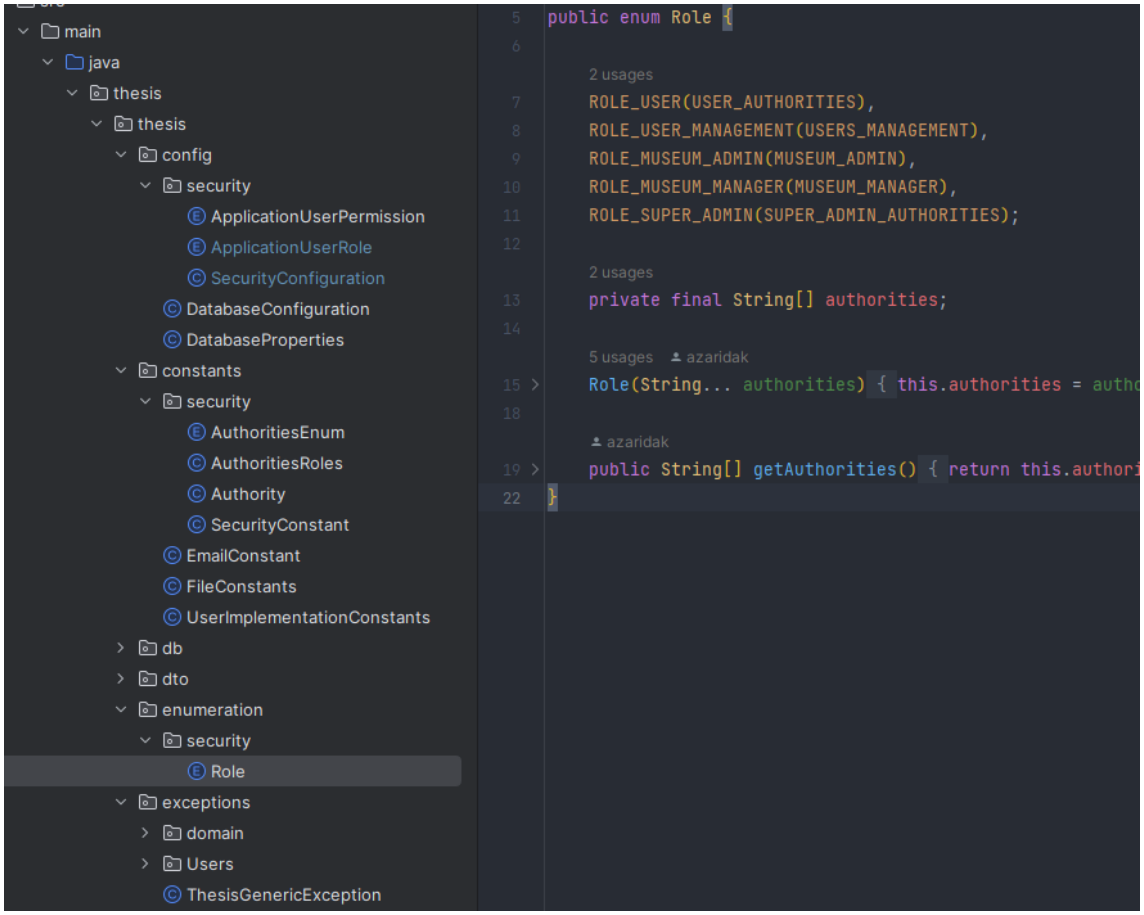
Για να γίνει διαχείριση στο Spring τέτοιων καταστάσεων γίνεται χρήση του annotation «preauthorize» από το Spring Security ('Spring Method Security with PreAuthorize' n.d.). Αυτό το annotation μας δίνει την δυνατότητα να προστατέψουμε την κάθε μέθοδο (τον κάθε controller δηλαδή) ξεχωριστά. Αναλυτικά, αυτό χρειάζεται γιατί όπως ειπώθηκε, ένας χρήστης μπορεί να είναι συνδεδεμένος αλλά να μην έχει πρόσβαση βάση δικαιωμάτων σε όλες τις λειτουργίες–συναρτήσεις. Εδώ λοιπόν, με το «preauthorize» σε κάθε μέθοδο εξασφαλίζουμε πως δεν θα καλεστεί κάποια μέθοδος από χρήστη που δεν είχε πρόσβαση καθώς αυτό είναι πολύ σημαντικό για την διασφάλιση της εφαρμογής από κάποιον κακόβουλο χρήστη. Θα ακολουθήσει η εικόνα 32 που δείχνει ένα παράδειγμα του annotation αυτού μέσα από τον κώδικα της διατριβής.

Εικόνα 32 - Παράδειγμα χρήστης του annotation PreAuthorize

Μέσα στον controller για τους users, υπάρχει η μέθοδος για διαγραφή και κάποιου χρήστη. Όπως φαίνεται στην εικόνα 32, υπάρχει το annotation «PreAuthorize» με όρισμα «hasAnyAuthority» και σε αυτό παρέχεται η τιμή «AuthoritiesRoles.USER_DELETE». Αυτό τεχνικά σημαίνει ότι σε αυτή την ενέργεια επιτρέπεται μόνο χρήστης που έχει πιστοποιηθεί με δικαιώματα «USER_DELETE». Είναι πλέον προφανές πως το σύστημα αυτά τα στοιχεία τα έχει αναγνωρίσει κατά την σύνδεση του χρήστη στο σύστημα από την ανάγνωση του JWT. Αν ο χρήστης έχει δικαιώματα λοιπόν για να πραγματοποιήσει την διαγραφή χρήστη, το αίτημα θα προωθηθεί στο επόμενο επίπεδο (Service) και σε αυτό με βάση τους κανόνες από την επιχειρησιακή λογική θα αποπερατωθεί. Αν είναι επιτυχές, θα επιστραφεί το αντίστοιχο μήνυμα στο front-end ώστε να προβεί στις απαραίτητες ενέργειες για την ενημέρωση του χρήστη. Αντίστοιχα, αν αποτύχει το αίτημα το front-end μέρος της υλοποίησης οφείλει να έχει υλοποιήσει την διαχείριση μιας τέτοιας περίπτωσης και να εμφανίσει για παράδειγμα ένα μήνυμα στον χρήστη πως το αίτημά του απέτυχε.

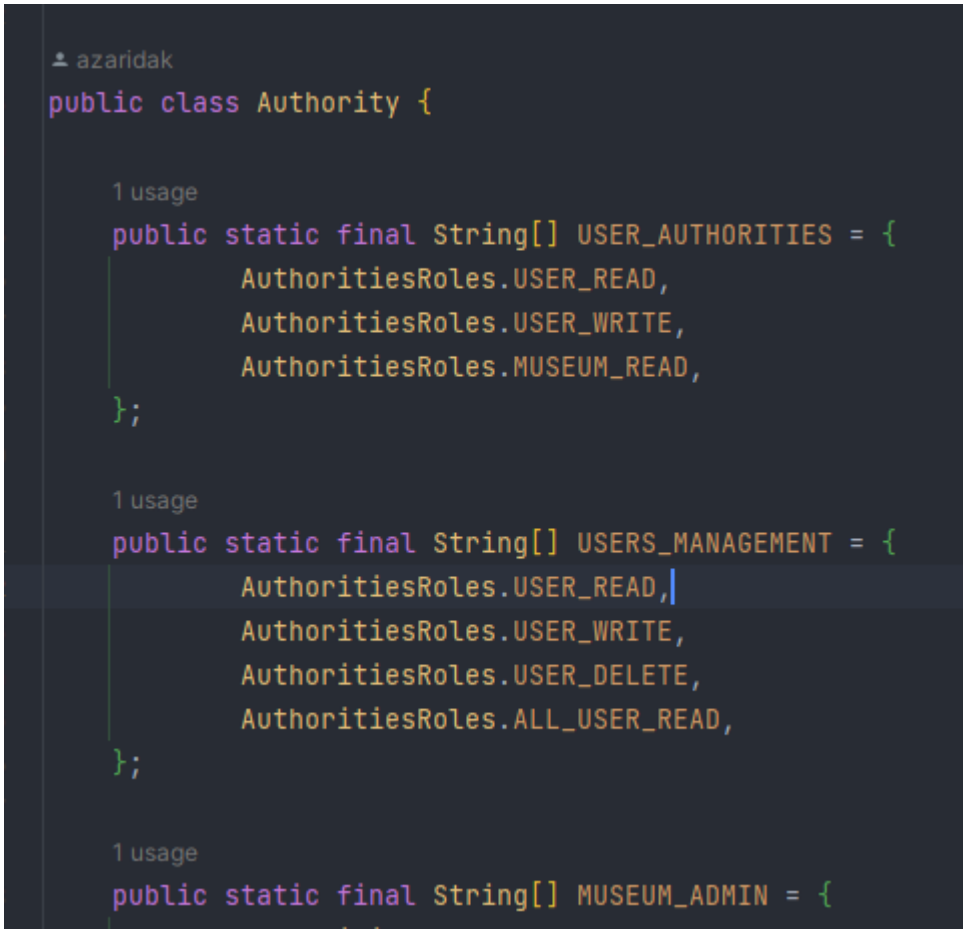
3.2.8 Ρόλοι Χρηστών, Εξαιρέσεις(Exceptions), DTOs

Σε αυτό το σημείο, θα παρουσιαστούν κλάσεις που χρησιμεύουν στο σύστημα για την διαχείριση σφαλμάτων (exceptions), για τους ρόλους και τα δικαιώματα που υπάρχουν στο σύστημα και τέλος οι κλάσεις Data Transfer Object για τις οποίες έχει αναλυθεί σε προηγούμενο κεφάλαιο ο λόγος που τις χρησιμοποιούμε και τα πλεονεκτήματα που μας παρέχουν. Θα αναφερθούν πρώτα οι ρόλοι, που είναι οι τιμές που αποθηκεύονται στην βάση δεδομένων για τον κάθε χρήστη και καθορίζουν όπως ειπώθηκε για τον χρήστη τα δικαιώματα πρόσβασης και διαχείρισης στην εφαρμογή.



Εικόνα 33 - Οι ρόλοι του συστήματος στο Role enum

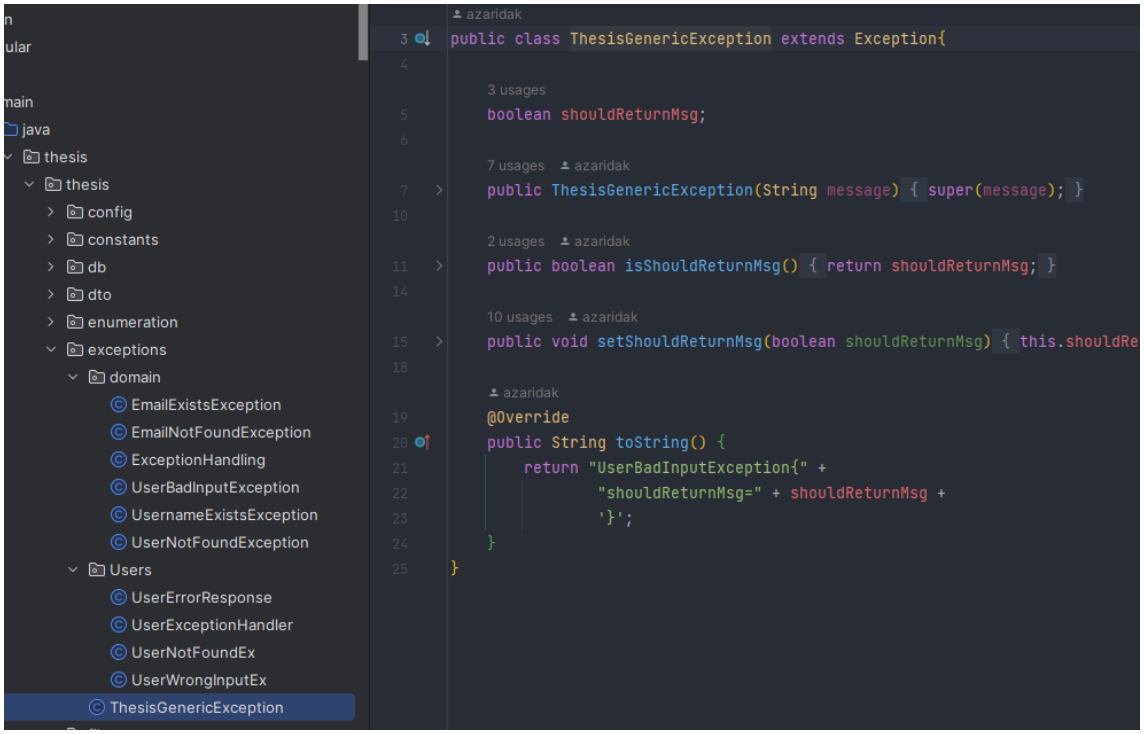
Στην εικόνα 33 φαίνεται το πακέτο security, που μέσα περιέχει τα Java Enums και Classes που χρησιμοποιούνται για την οργάνωση αυτών των δεδομένων. Στην εικόνα αυτή φαίνεται το Enum Role που είναι οι ρόλοι των χρηστών στο σύστημα και κάθε ρόλος έχει συγκεκριμένα authorities δηλαδή δικαιώματα για ενέργειες. Για παράδειγμα, ο «ROLE_USER» έχει τα δικαιώματα «USER_AUTHORITIES» που βρίσκονται στην κλάση «Authority». Με την σειρά τους αυτά με τις τιμές που τους έχουν δηλωθεί από το Enum «AuthoritiesRoles» παρέχουν στους χρήστες τα αντίστοιχα δικαιώματα. Στην εικόνα 34 που ακολουθεί φαίνεται πως είναι διαμορφωμένα και οργανωμένα σε επίπεδο κώδικα για να γίνει κατανοητό.



Εικόνα 34 - Η κλάση που περιέχει τα Authorities

Συνεχίζοντας, φαίνεται πως για έναν χρήστη με ρόλο συστήματος «ROLE_USER» στον οποίο παραχωρούνται δικαιώματα «USER_AUTHORITIES» από την εικόνα 34 είναι φανερό πως έχει δικαιώματα για ανάγνωση, εγγραφή των προσωπικών του δεδομένων και για τα μουσεία έχει μόνο δικαίωμα ανάγνωσης. Άρα, είναι φανερό πως ο απλός χρήστης δεν έχει δικαίωμα να διαγράψει είτε κάποιον χρήστη είτε κάποιο μουσείο. Με τον ίδιο τρόπο έχουν διαμορφωθεί και τα υπόλοιπα ζεύγη ρόλων-δικαιωμάτων ώστε να προκύπτει ένα συμπαγές σύστημα χρηστών.

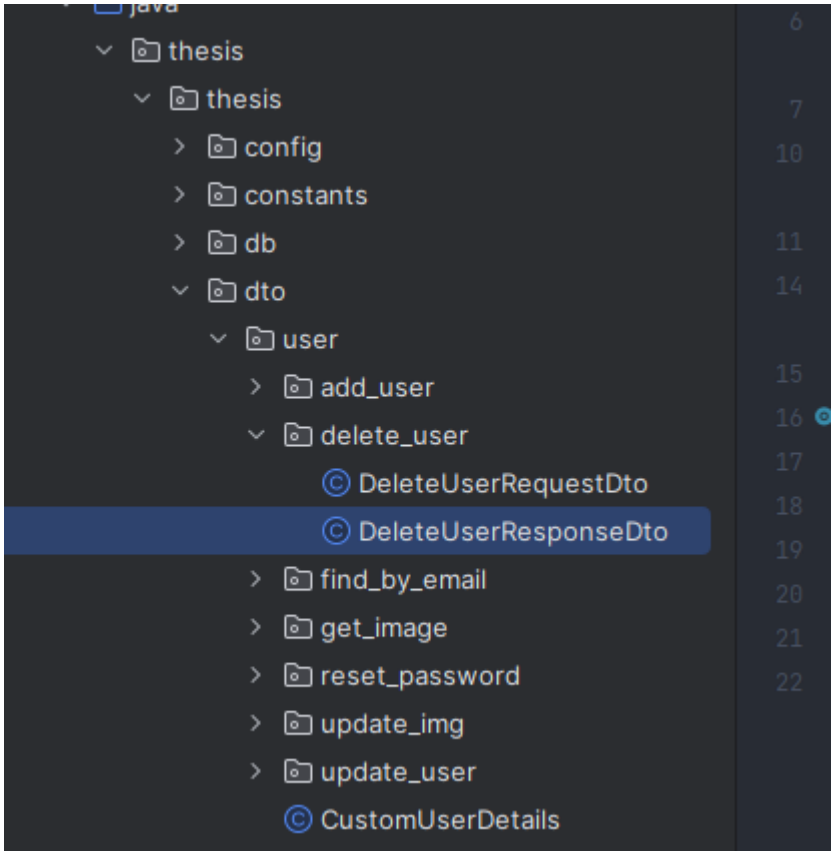
Στο σύστημα έχει επίσης σχεδιαστεί και υλοποιηθεί ο χειρισμός των σφαλμάτων με συγκεκριμένο τρόπο και με εξαιρέσεις(exceptions) φτιαγμένα για τους σκοπούς της εφαρμογής. Ακολουθεί η εικόνα 35 με παραδείγματα εξαιρέσεων.



Εικόνα 35 - Τα εξατομικευμένα Exception Classes της Εφαρμογής

Όπως λοιπόν απεικονίζεται στην εικόνα 35, βλέπουμε πως υπάρχει το γενικό Exception το οποίο κάνει επέκταση(extends) την γενική κλάση Exception και το όνομα της κλάσης είναι «ThesisGenericException». Έπειτα, έχουν υλοποιηθεί αρκετά συγκεκριμένα Exceptions που χρησιμοποιούνται σε διάφορες λειτουργικότητες της εφαρμογής όπως «EmailExistsException» και «EmailNotFoundException». Το πρώτο προκαλείται όταν ένας χρήστης επιχειρεί να κάνει εγγραφή με διεύθυνση ηλεκτρονικού ταχυδρομείου που υπάρχει ήδη στο σύστημα και χρησιμοποιείται από άλλον χρήστη ή όταν ένας χρήστης πάει να ενημερώσει τα στοιχεία του και εισάγει όνομα χρήστη που τυχαίνει να είναι δεσμευμένο από άλλον χρήστη και άρα είναι ήδη σε χρήση οπότε η ενημέρωση των στοιχείων δε μπορεί να πραγματοποιηθεί. Συνεχίζοντας, το δεύτερο exception χρησιμοποιείται στη περίπτωση που ένας χρήστης επιχειρεί να συνδεθεί στο σύστημα όμως η διεύθυνση ηλεκτρονικού ταχυδρομείου που παρέχει δεν είναι καταγεγραμμένη στη βάση δεδομένων. Αντίστοιχα, όλα τα άλλα Exceptions που έχουν φτιαχτεί είναι για τις διάφορες περιπτώσεις χρήσεων που γίνονται μέσα στο σύστημα.

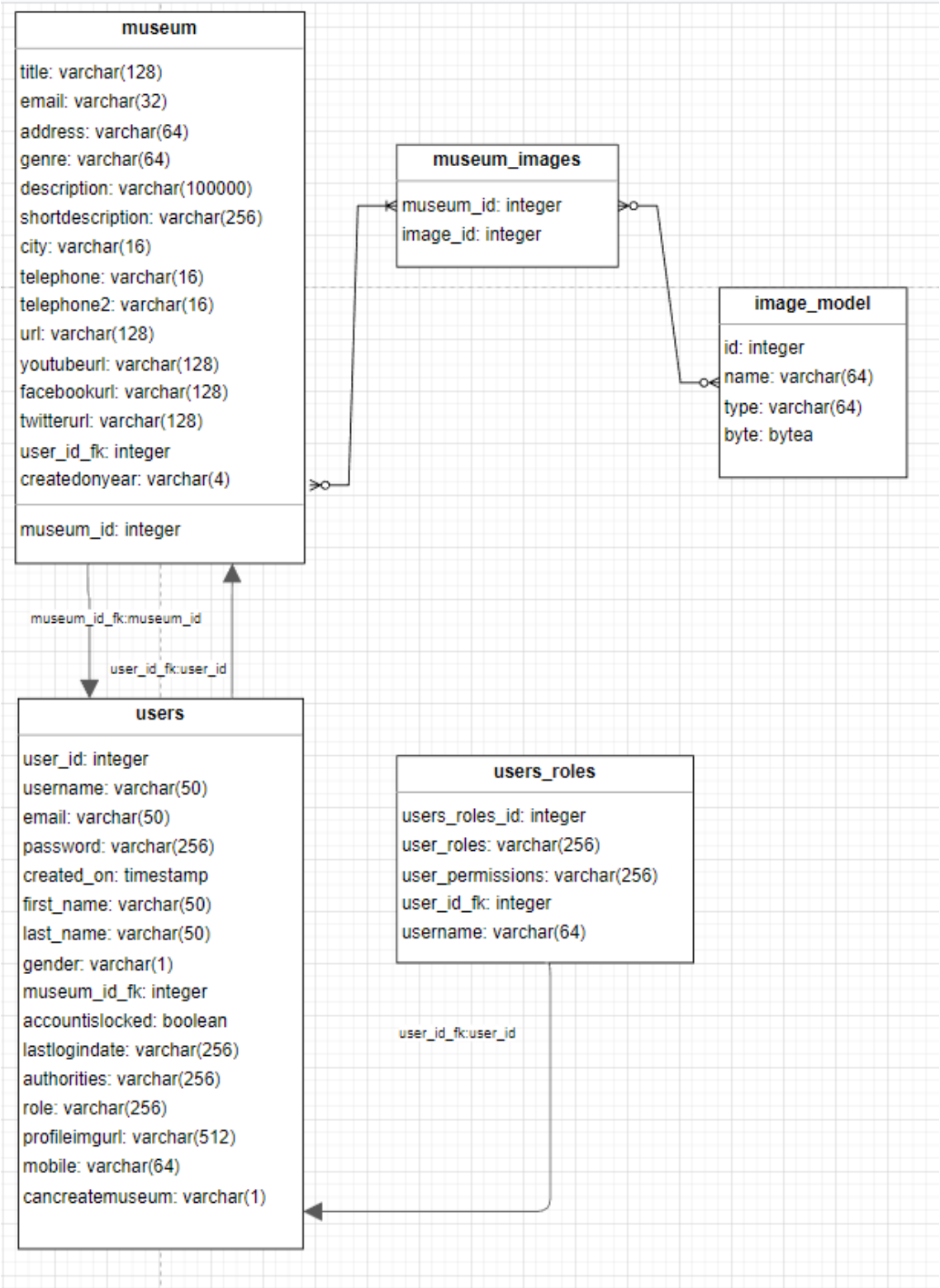
Ο λόγος που επιλέχθηκε να φτιαχτούν τα ειδικά(custom) Exceptions στα πλαίσια της μεταπτυχιακής διατριβής είναι πως παρέχουν αρκετά πλεονεκτήματα για την δομή αλλά και για την οργάνωση του λογισμικού. Αρχικά, προσφέρουν καλύτερο έλεγχο στην διαχείριση των Exceptions καθώς βοηθάνε τον προγραμματιστή να κάνει καλύτερο διαχωρισμό των λειτουργιών κατά την ανάπτυξη του λογισμικού αλλά και κατά την αποσφαλμάτωση του κώδικα καθώς κάθε Exception που θα εγείρετε θα έχει συγκεκριμένο όνομα και τύπο μηνύματος και θα είναι αρκετά εύκολο να γίνει αναζήτηση του προβλήματος που το προκάλεσε το συγκεκριμένο exception. Επίσης, παρέχουν ευκολία στην ανάγνωση του κώδικα και τον συναρτήσεων. Το όνομα και το μήνυμα των Exceptions ακολουθούν την ροή του προγράμματος, και πολλές φορές το όνομά τους βοηθάει αμέσως να κατανοήσει ο προγραμματιστής πότε θα εμφανιστεί το συγκεκριμένο Exception. Τέλος, είναι πιο εύκολο και οργανωμένο το να υπάρχει σωστή ενημέρωση στο front-end στην περίπτωση κάποιου σφάλματος με συγκεκριμένα και ελεγχόμενα μηνύματα που θα φτάνουν στο χρήστη, χωρίς να αποκαλύπτουν τον πραγματικό λόγο του προβλήματος αλλά το τι κρίνεται σκόπιμο βάση της επιχειρησιακής λογικής να εμφανιστεί στον χρήστη.



Εικόνα 36 - Δομή πακέτων για τα Data Transfer Objects

Επιπλέον, στην εικόνα 36 φαίνεται το πακέτο που περιέχει τα DTOs για την διαχείριση των χρηστών, όπως όταν γίνεται κάποιο αίτημα για ενημέρωση στοιχείων του χρήστη, προσθήκη νέου χρήστη ή διαγραφή.

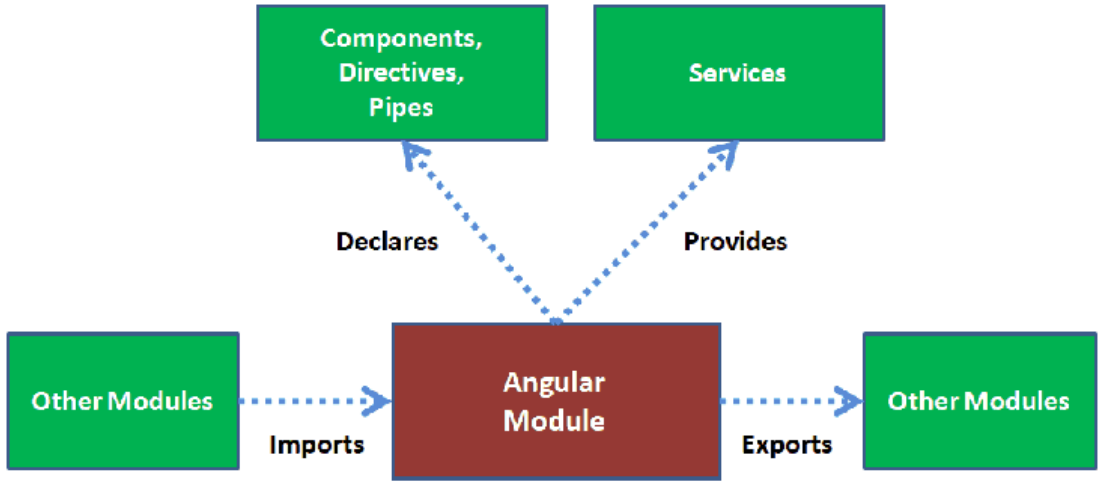
Τέλος, στην εικόνα 37 απεικονίζεται το διάγραμμα οντοτήτων συσχετίσεων της βάσης δεδομένων την εφαρμογής με βάση όσα όλα έχουν αναφερθεί ήδη για την λειτουργία και τον τρόπο αποθήκευσης δεδομένων στην εφαρμογή.



Εικόνα 37 - Οι σχεσιακοί πίνακες της Βάσης Δεδομένων

3.3 Σχεδιασμός και λειτουργίες στο Front End

Στο σημείο αυτό θα παρουσιαστεί και θα αναλυθεί η δομή και ο τρόπος λειτουργίας του front-end κομμάτι της εφαρμογής για την μεταπτυχιακή διατριβή. Όπως ήδη αναφέρθηκε, η υλοποίηση έχει γίνει με το framework της Angular και θα αρχίσουμε με την ανάλυση της τεχνικής και αρχιτεκτονικής που χρησιμοποιήθηκε για την υλοποίηση. Οι βασικές έννοιες που απαρτίζουν την Angular είναι τα Modules, τα Components, τα Templates, τα Directives, τα Services, τα Pipes και το Data Biding. Πολύ σημαντικό κομμάτι της αρχιτεκτονικής είναι επίσης το modularity που προσφέρει. Επίσης, αυτό οργανώνεται από τα Modules που αναφέρθηκαν. Πιο συγκεκριμένα, ονομάζονται Ng Modules, και είναι ο τρόπος που με τον οποίο ενώνει μεταξύ τους τα components, τα directives και τα services που είναι μεταξύ τους σχετικά και απαρτίζουν ένα συγκεκριμένο κομμάτι του συστήματος. Στο τελικό αποτέλεσμα λοιπόν, αφού η εφαρμογή θα έχει ολοκληρωθεί το κάθε module θα είναι σαν ένα κομμάτι του συνολικού puzzle. Με αυτό τον τρόπο όπως γίνεται κατανοητό αποκτά μεγάλη ευελιξία σαν σύστημα όσο αναφορά την περαιτέρω ανάπτυξη, την βελτίωση αλλά και την αποσφαλμάτωση. Διότι, με την αλλαγή ενός κομματιού από το puzzle, μπορούμε να ενσωματώσουμε μία μεγάλη αλλαγή χωρίς να χρειαστούν αλλαγές σε όλη την εφαρμογή. Τέλος, κάθε εφαρμογή σε Angular είναι σίγουρο πως θα έχει τουλάχιστον ένα module, αλλά στην συντριπτική πλειοψηφία οι εφαρμογές απαρτίζονται από αρκετά μεγαλύτερο αριθμό modules.



Εικόνα 38 - Διάγραμμα επικοινωνίας των modules της Angular (TekTutorialsHub 2018)

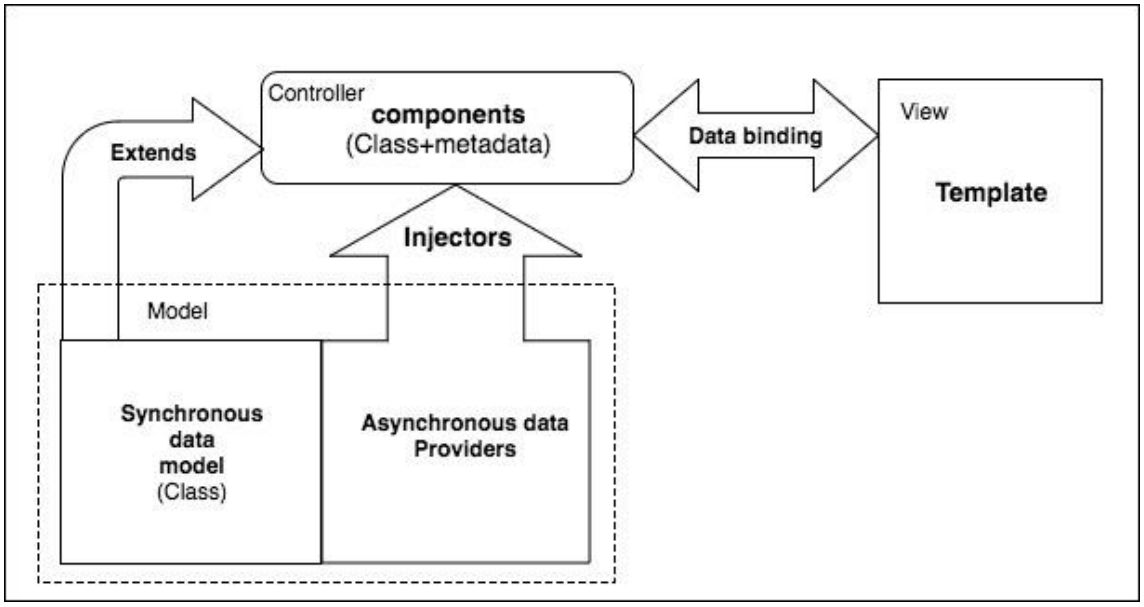
Στην εικόνα 38 απεικονίζονται τρία διαφορετικά modules, το κάθε ένα όπως φαίνεται είναι συνδεδεμένο με το άλλο και το κάθε module επίσης, έχει τα δικά του components, directives κ.λπ. Επίσης, είναι πιο ξεκάθαρο στην εικόνα 38 πως απαρτίζει το κάθε module ένα κομμάτι του puzzle και πως θα μπορούσε να αφαιρεθεί ή να προστεθεί ένα νέο.

Στην συνέχεια, θα γίνει ανάλυση των components τα οποία είναι τα κύρια δομικά συστατικά σε κάθε εφαρμογή. Κάθε ένα από αυτά, αποτελείται από ένα HTML template με το οποίο γίνεται αναπαράσταση της σελίδας στον χρήστη, μία κλάση σε γλώσσα προγραμματισμού Typescript που σε αυτή συμπεριλαμβάνεται τυχόν επιχειρησιακή λογική και ένα αρχείο CSS για το συγκεκριμένο component ώστε να δοθούν συγκεκριμένες στυλιστικές λεπτομέρειες. Το τελευταίο ωστόσο δεν είναι υποχρεωτικό να υπάρχει. Ακόμα, το αρχείο που περιέχει την κλάση Typescript, περιλαμβάνει και κάποια metadata τα «selector», «templateUrl», «styleUrl» τα οποία τα χρησιμοποιεί για να γνωρίζει με ποιο html αρχείο να κάνει binding (templateUrl), πότε πρέπει να ενεργοποιηθεί (selector) και τέλος, εφόσον και αν υπάρχει συγκεκριμένο αρχείο με css πως

ονομάζεται έτσι ώστε να το χρησιμοποιήσει (styleUrl). Το data binding που αναφέρθηκε σημαίνει πως υπάρχει μια αλληλεπίδραση ανάμεσα στο κλάση typescript και στο αρχείο html. Ένα παράδειγμα πραγματικής χρήσης αν υποθέσουμε για παράδειγμα σε μία ηλεκτρονική διαδικτυακή πλατφόρμα αγορών αν ο χρήστης εισάγει αγαθά στο καλάθι το σύνολο τιμής του καλαθιού πρέπει να υπολογίζεται δυναμικά. Έτσι, η ενημέρωση που γίνεται στο UI από τον χρήστη φτάνει στην κλάση, κάνει τους υπολογισμούς στην περίπτωσή μας για το σύνολο του καλαθιού και μόλις έχει ολοκληρώσει ενημερώνει το HTML για να αλλάξει και την τιμή που βλέπει ο χρήστης. Φυσικά, η κλάση μπορεί να κάνει και πολύ πιο σύνθετους υπολογισμούς καθώς όπως αναφέρθηκε περιέχει και επιχειρησιακή λογική αρκετές φορές. Ακόμα, με την χρήση έγχυσης εξαρτήσεων(dependency injection) γίνεται εφικτό να χρησιμοποιηθούν πολλά services μέσα στο component και να υπάρξει και σύνδεση μέσω αυτών με το back-end κομμάτι. Για αυτό τον τρόπο λειτουργίας θα γίνει περιγραφή στην ανάλυση των services.

Τα services λοιπόν, είναι αντικείμενα(objects) που χρησιμοποιούνται από άλλα components και σε αυτά τοποθετούμε κώδικα και συναρτήσεις οι οποίες μπορούν να χρησιμοποιηθούν από πολλαπλά αρχεία. Αυτό σημαίνει πως αφού θα γραφτούν μία φορά, έπειτα δίνεται η δυνατότητα να μπορούν να χρησιμοποιούνται από πολλά διαφορετικά components έτσι ώστε να αποφεύγεται ο επαναλαμβανόμενος κώδικας. Οι συναρτήσεις αυτές, μπορεί να είναι για να εξυπηρετούν επιχειρησιακή λογική, να είναι εσωτερικές βοηθητικές συναρτήσεις όπως να πιστοποιούν πως ένα κινητό τηλέφωνο αποτελείται από δέκα αριθμητικούς χαρακτήρες, αλλά και συναρτήσεις οι οποίες κάνουν κάποιο HTTP αίτημα στο back-end API και άρα είναι ο συνδετικός κρίκος της εφαρμογής που αλληλοεπιδρούν με αυτό το front-end με το back-end. Επίσης, ένα άλλο χαρακτηριστικό τους είναι πως δημιουργούνται μία φορά, και είναι διαθέσιμα για χρήση σε όλο τον κύκλο ζωής της εφαρμογής και μπορούν να χρησιμοποιούνται από περισσότερα από ένα components.

Επιπλέον, αξίζει να αναφερθεί και ο router. Με αυτόν, ουσιαστικά δίνεται η δυνατότητα στον χρήστη να αλλάζει σελίδες δηλαδή τα HTML components που έχουν αναφερθεί, με το χαρακτηριστικό ότι δεν χάνεται η κατάσταση του συστήματος. Δηλαδή, για παράδειγμα στην περίπτωση περιήγησης σε ένα ηλεκτρονικό διαδικτυακό κατάστημα αγορών, όσο αλλάζει ο χρήστης σελίδες στο σύστημα, δεν χάνονται δεδομένα όπως τι υπάρχει ήδη στο καλάθι του.

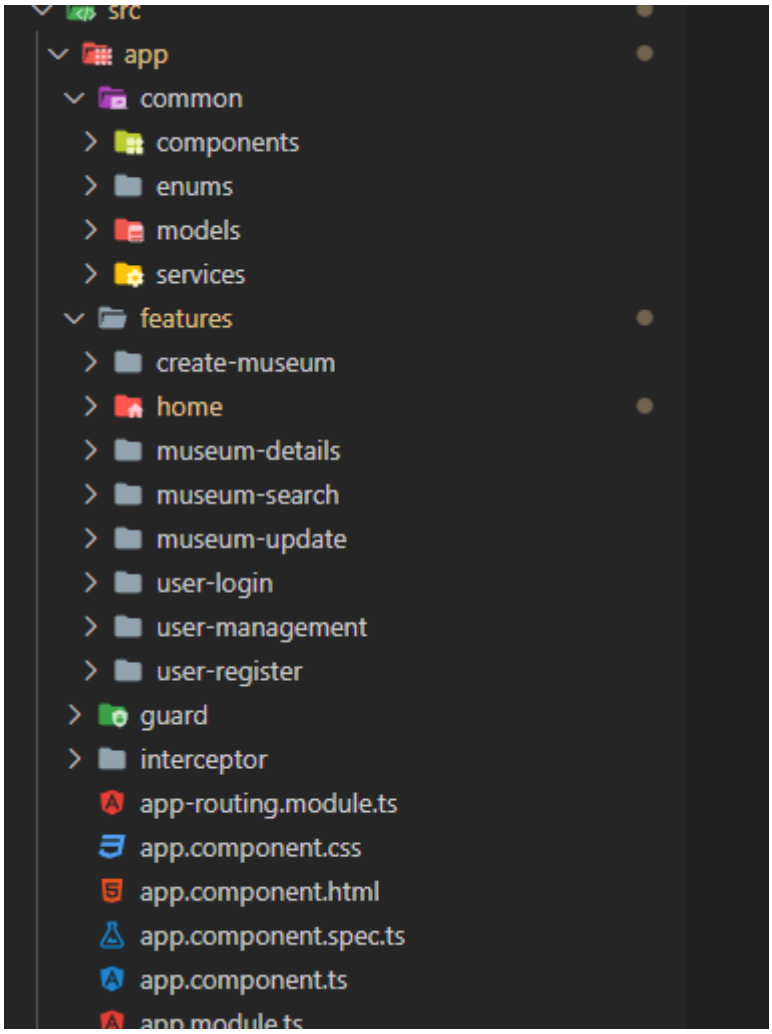


Εικόνα 39 - Διάγραμμα κύκλου ζωής Angular ('Angular Data Flow Architecture - Web Development with MongoDB and Node - Third Edition [Book]' n.d.)

Στην Εικόνα 39, βλέπουμε ένα διάγραμμα με το κύκλο ζωής της εφαρμογής στο front-end μέρος.

3.3.1 Σχεδιασμός των Components

Εδώ θα αναλυθούν τα components της εφαρμογής και πως έχουν χωριστεί μέσα στο σύστημα.



Εικόνα 40 - Η δομή των components στην εφαρμογή

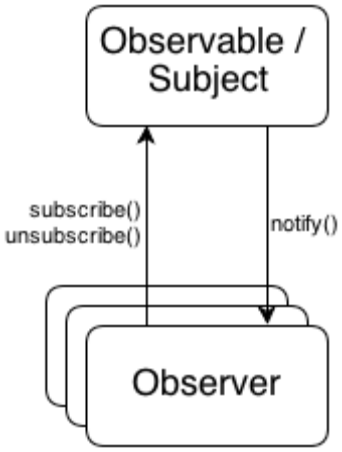
Στην εικόνα 40 απεικονίζονται τα component της εφαρμογής. Αρχικά, βλέπουμε μέσα στο φάκελο common όπου εκεί υπάρχουν κοινόχρηστα components όπως το footer που καθώς είναι κοινό σε όλες τις σελίδες, έχει υλοποιηθεί μία φορά και γίνεται εισαγωγή και χρησιμοποιείται από όλες τις σελίδες. Αυτό, μας προσφέρει το πλεονέκτημα ότι δεν χρειάζεται να επαναλαμβάνεται ο ίδιος κώδικας αλλά και πως αν χρειαστεί να γίνει μία αλλαγή στο footer, θα γίνει σε ένα σημείο και αυτόματα θα ενημερωθούν όλες οι σελίδες. Στη συνέχεια, στον φάκελο features, έχουν μπει όλα τα lazy-loaded components. Lazy-loaded σημαίνει πως δεν γίνεται φόρτωση στην εφαρμογή συνέχεια αλλά μόνο όταν επιλέξει ο χρήστης να μεταβεί σε μία σελίδα η οποία απεικονίζεται από ένα συγκεκριμένο lazy loaded component. Αυτό έχει το πλεονέκτημα πως η σελίδα είναι ελαφριά και αποδοτική καθώς δεσμεύει πόρους δυναμικά και όποτε χρειάζεται. Για κάθε σελίδα υπάρχει και ένα component, για παράδειγμα για την εγγραφή χρήστη, για την είσοδο χρήστη για την δημιουργία μουσείων κ.λπ. Κάποια από αυτά, χρησιμοποιούν κοινά services όπως για

παράδειγμα το service που κρατάει την πληροφορία για τον συνδεδεμένο χρήστη. Περισσότερα για αυτή την λειτουργία θα αναλυθούν μαζί με τα υπόλοιπα services.

Το γραφικό περιβάλλον της εφαρμογής έχει βασιστεί στο Angular Material (Team n.d.) αλλά και στο Bootstrap (contributors n.d.). Πρακτικά αυτό μεταφράζεται πως στα HTML αρχεία και στα CSS έχουν χρησιμοποιηθεί οι κλάσεις και τα components από το Bootstrap και το Angular material αντίστοιχα με σκοπό το design της εφαρμογής, να είναι φιλικό προς τον χρήστη και responsive δηλαδή, να προσαρμόζεται σε διάφορα μεγέθη οθονών όπως κινητές συσκευές και tablet.

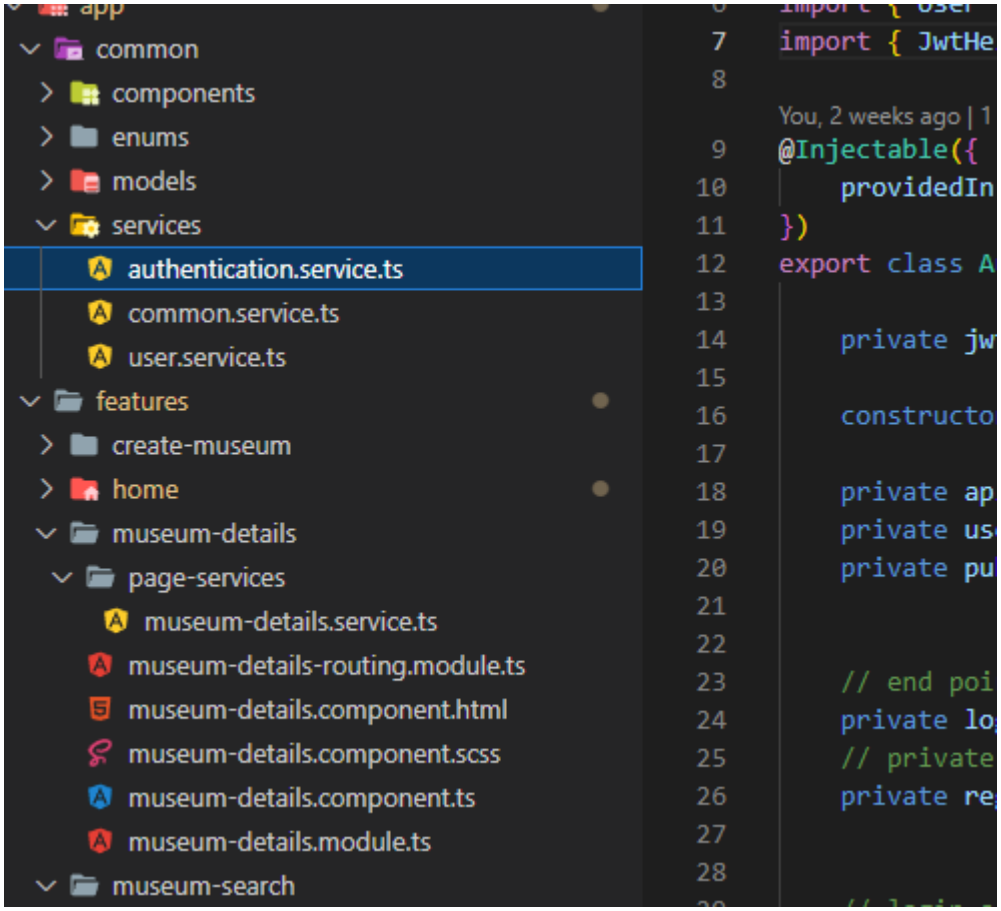
3.3.2 Σχεδιασμός των Services

Ακολουθεί η ανάλυση των services, πως έχουν χρησιμοποιηθεί αλλά και πως έγινε ο διαχωρισμός τους στο σύστημα. Τα services, μπορούν να περιέχουν μέσα συναρτήσεις για υπολογισμό δεδομένων, HTTP αιτήματα για επικοινωνία με το back-end ή Subscriptions και Observables. Σε αυτό το σημείο, θεωρείται χρήσιμο να εξηγηθούν αυτοί οι δύο όροι καθώς είναι κομβικοί για το πως λειτουργούν ακριβώς μέσα στο σύστημα. Ξεκινώντας με τα Observables, μια γενική περιγραφή είναι πως χρησιμοποιούνται μέσα στην Angular για να χειρίζεται διάφορα ασύγχρονα αιτήματα και διαδικασίες όπως μεταφορές δεδομένων ανάμεσα σε δύο διαφορετικά components. Επιπλέον, χρησιμοποιούνται και για να εκπέμπουν συμβάντα (event emit) μέσα στην εφαρμογή όπως για παράδειγμα το event για να ανοίξει ένα νέο παράθυρο με μία φόρμα μέσα ώστε να συμπληρώσει ο χρήστης κάποια στοιχεία όπως τα στοιχεία σύνδεσής του. Το Subscription για να αναλυθεί θα γίνει συνδυαστικά με το Observable. Πιο συγκεκριμένα, αναφέρεται σε έναν πόρο που έχει δεσμευτεί εκείνη την στιγμή, συνήθως με την εκτέλεση ενός Observable. Στο Subscription υπάρχει μία πολύ σημαντική μέθοδος η «unsubscribe» που ουσιαστικά αποδεσμεύει τους πόρους που είχε δεσμεύσει και δηλώνει πως δεν περιμένει άλλες αλλαγές. Στην εικόνα 41, φαίνεται πως λειτουργεί το Subscription καθώς βλέπουμε πως μόλις ο παρατηρητής (observer) κάνει εγγραφή στο Observable, ενημερώνεται συνεχώς σε περίπτωση που έρθει κάποια καινούργια αλλαγή. Όπως ειπώθηκε, αυτό ολοκληρώνεται μόλις γίνει χρήση της συνάρτησης unsubscribe όπου και τότε γίνεται αποδέσμευση των πόρων που είχαν δεσμευτεί. Τέλος, να σημειωθεί πως είναι πολύ σημαντικό να καλείται πάντα το unsubscribe όταν πλέον δεν χρειάζεται αλλιώς, μένουν ανοιχτά και δεσμευμένα τα subscriptions μέσα στην εφαρμογή χωρίς να χρειάζονται. Αυτό μπορεί να οδηγήσει σε περιττή και μεγάλη κατανάλωση υπολογιστικών πόρων στο client-side της εφαρμογής και αυτό να έχει ως αποτέλεσμα να γίνει η εφαρμογή αργή και δύσχρηστη για τους χρήστες.



Εικόνα 41 - Διάγραμμα ροής επικοινωνίας μεταξύ observer και observable ('JavaScript Observer (Publish/Subscribe) Pattern' n.d.)

Θα ακολουθήσουν στη συνέχεια κάποια παραδείγματα των services που έχουν υλοποιηθεί μέσα στην εφαρμογή για τα πλαίσια της μεταπτυχιακής διατριβής.



Εικόνα 42 - Τα common Services του συστήματος

Στο φάκελο common υπάρχουν τα services τα οποία χρησιμοποιούνται από διαφορετικά components και για αυτό το λόγο έχουν τοποθετηθεί στον φάκελο common. Υπάρχει το authentication service, το common service και το user service. Το πρώτο, χρησιμοποιείται για να χειρίζεται την σύνδεση και την αποσύνδεση του χρήστη στο σύστημα. Επιπλέον, χειρίζεται και το JWT που όπως έχει ήδη αναφερθεί στέλνεται από το back-end μέρος της εφαρμογής μόλις ο χρήστης πραγματοποιήσει σύνδεση στο σύστημα. Πιο συγκεκριμένα, λαμβάνει το JWT διαβάζει και αναγνωρίζει τα headers του, αν είναι αυτά που πρέπει (JWT token) τότε το αποθηκεύει στην μνήμη του browser μαζί με τα δεδομένα του χρήστη που περιέχονται μέσα σε αυτό. Φυσικά, στο back-end μέρος έχουμε φροντίσει να μην αποστέλλονται ευαίσθητα δεδομένα του χρήστη οπότε είναι ασφαλές να αποθηκευτεί στην μνήμη του browser. Ακόμα, εκτός από την αποθήκευση υπάρχει και μέθοδος για ανάκτηση του token από την μνήμη ώστε να μπορεί κάθε φορά να αναγνωρίζεται.

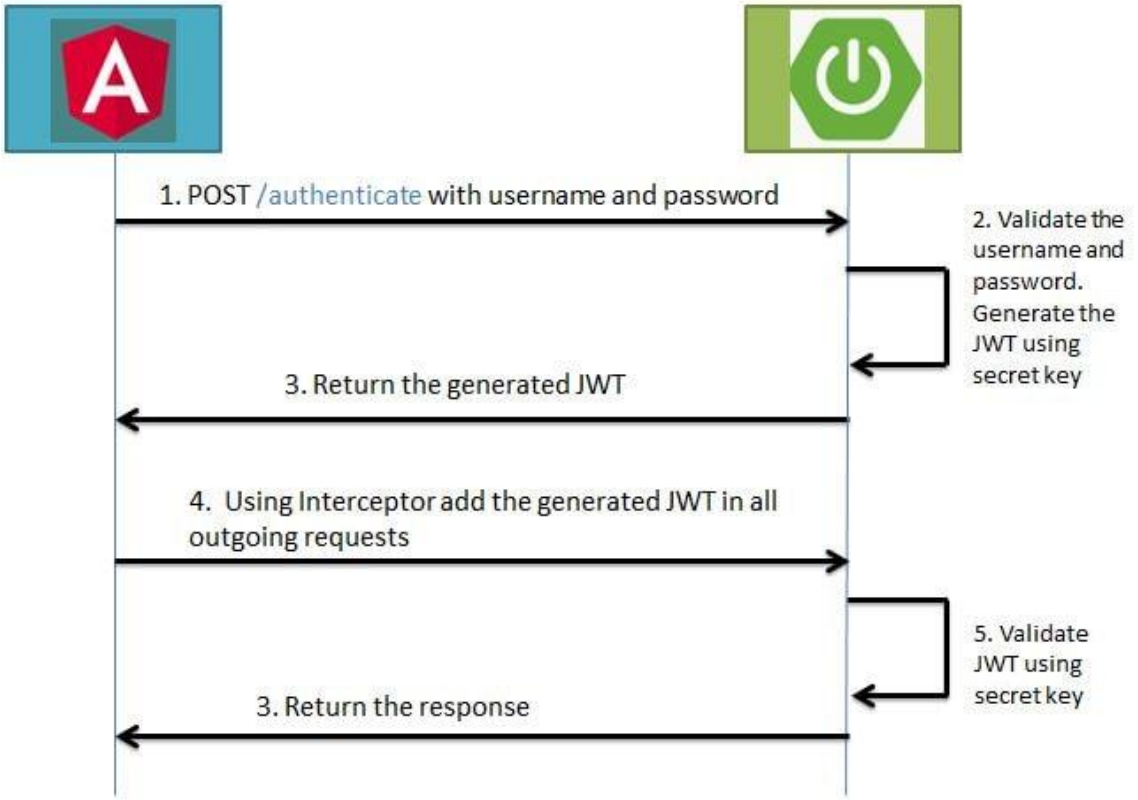
Στη συνέχεια, στο user service, υπάρχουν συναρτήσεις που διαχειρίζονται δεδομένα για τον χρήστη. Μία συνάρτηση που γίνεται χρήση της εκτεταμένα είναι η «getUserDetails» η οποία στην ουσία όταν ένας χρήστης συνδεθεί στέλνει ένα request στο back-end API για να πάρει στοιχεία του χρήστη που έχουν να κάνουν με τα δικαιώματα χρήσης του και περιήγησης στην εφαρμογή. Για παράδειγμα, αν μπορεί να διαγράψει ένα μουσείο, αν μπορεί να διαγράψει κάποιον χρήστη και άλλα τέτοια δικαιώματα τα οποία χρησιμοποιούνται από το σύστημα. Ακόμα, βάση αυτών των δικαιωμάτων αλλάζουν δυναμικά κάποιες σελίδες για τον χρήστη ή άλλες σελίδες δεν φαίνονται καθόλου καθώς δεν του επιτρέπεται η πρόσβαση σε αυτές. Τέλος, από τα κοινόχρηστα services είναι το common, μέσα σε αυτό υπάρχουν βοηθητικές συναρτήσεις όπως για τον έλεγχο ότι δεδομένα που έχουν εισαχθεί σε κάποια φόρμα είναι σωστά. Για παράδειγμα στην εισαγωγή για

ένα τηλέφωνο πρέπει να είναι μόνο αριθμητικούς χαρακτήρες ή ένα email πρέπει να περιέχει τον χαρακτήρα παπάκι (@).

3.3.3 AuthInterceptor και AuthenticationGuard

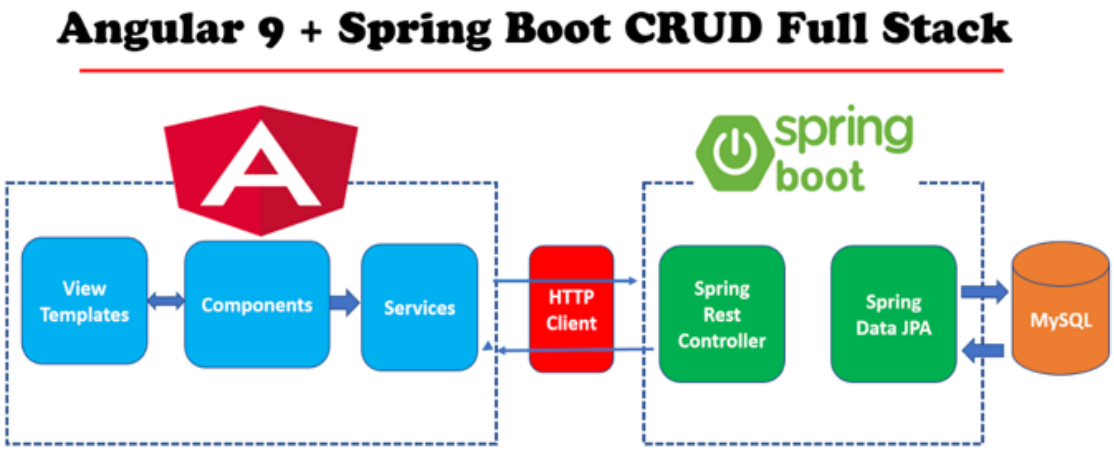
Σε αυτό το κεφάλαιο θα γίνει αναφορά ακόμα δύο σημαντικών κλάσεων στην λειτουργία του front-end μέρους της εφαρμογής. Η κλάση «AuthInterceptor» και η «AuthenticationGuard». Και των δύο αυτών κλάσεων, οι λειτουργίες σχετίζονται με την ασφάλεια της εφαρμογής και την διασφάλιση λειτουργίας των συνδεδεμένων χρηστών. Πιο συγκεκριμένα, η κλάση AuthenticationGuard έχει ως ρόλο χρήσης να προστατεύει τα Routes της Angular τα οποία πρέπει να ανοίγουν σε συνδεδεμένους και μόνο χρήστες. Για παράδειγμα, αν δοκιμάσει να ανοίξει κάποιος χρήστης την σελίδα του «το προφίλ μου» ενώ δεν είναι συνδεδεμένος στο σύστημα, το αποτέλεσμα με την χρήση της κλάσης AuthenticationGuard θα είναι να κάνει ανακατεύθυνση στην σελίδα σύνδεσης στο σύστημα. Φυσικά, αυτό γίνεται σε κάθε απόπειρα να ανοίξει μία σελίδα ενώ δεν έχει δικαίωμα ο χρήστης να την προσπελάσει επειδή δεν είναι συνδεδεμένος.

Η κλάση AuthInterceptor έχει τον ρόλο να αναχαιτίζει όλα τα HTTP αιτήματα τα οποία παράγονται και κατευθύνονται από το front-end κομμάτι στο back-end και να τους τοποθετεί το JWT token στο HTTP Authorization token. Φυσικά, υπάρχουν και αιτήματα που δεν πρέπει να πραγματοποιηθεί η προσθήκη του JWT token. Για παράδειγμα, το request για σύνδεση στο σύστημα δεν πρέπει να έχει ένα τέτοιο token. Ακόμα, όπως έχει ήδη αναφερθεί, υπάρχουν και τα δημόσια public endpoints ή σελίδες, τις οποίες έχουν δικαίωμα να τις βλέπουν όλοι οι χρήστες χωρίς να χρειάζεται να είναι συνδεδεμένοι και άρα δεν χρειάζεται να πηγαίνει το JWT στο back-end. Μέσα στην υλοποίηση λοιπόν, όπως γίνεται κατανοητό πρέπει να οριστούν τα δημόσια end points έτσι ώστε να μην γίνεται αυτή η αναχαίτηση σε αυτά, όπως δεν πρέπει να γίνεται και στο αίτημα για την σύνδεση του χρήστη στο σύστημα.



Εικόνα 43 - Ανταλλαγή HTTP αιτημάτων με JWT αναχαίτηση(introspection) ('Angular 7 + Spring Boot JWT Authentication | JavaInUse' n.d.)

Στην εικόνα 43 φαίνεται καθαρά η δράση της αναχαίτησης που γίνεται στα HTTP αιτήματα συγκεκριμένα στο βήμα 4. Το JWT που γίνεται αναχαίτιση και ελέγχεται από τα filters του back-end και όπως εξηγήθηκε στο προηγούμενο κεφάλαιο και συνεχίζεται η ροή του συστήματος. Στο σημείο αυτό, καθώς έχει επεξηγηθεί και έχει γίνει κατανοητή η συνολική επικοινωνία του front-end μέρος της εφαρμογής(client side) με το back-end API στην εικόνα 44 θα φανεί σε high level όλη η επικοινωνία που γίνεται σε μορφή διαγράμματος.



Εικόνα 44 - Αρχιτεκτονική επικοινωνίας Angular με Spring Boot ('Angular + Spring Boot + MySQL CRUD Example' n.d.)

4 Εκτέλεση Συστήματος, Περιπτώσεις Χρήσης

4.1 Περίληψη Κεφαλαίου

Σε αυτό το κεφάλαιο θα παρουσιαστούν περιπτώσεις χρήσεως (use cases) μέσα στο σύστημα και ποιες ενέργειες μπορεί να κάνεις ένας χρήστης. Επίσης, τι λογισμικό χρειάζεται για να τρέξει σε έναν υπολογιστή.

4.2 Εκτέλεση

Για να γίνει εκτέλεση του back-end API πρέπει στο σύστημα να είναι εγκατεστημένη Java version “14.0.1”, για βάση δεδομένων η PostgreSQL “9.3.10” και το maven version “3.6.3”. Για να τρέξει το application το build και deploy γίνεται με την εντολή maven:

clean spring-boot:run -Dspring-boot.run.fork=false -Dspring.profiles.active=local -Dspring.config.location=file:C:/{{path to application.properties}}/application.properties

Επίσης, για να γίνει καθαρή εγκατάσταση πριν την εκκίνηση είναι η εντολή

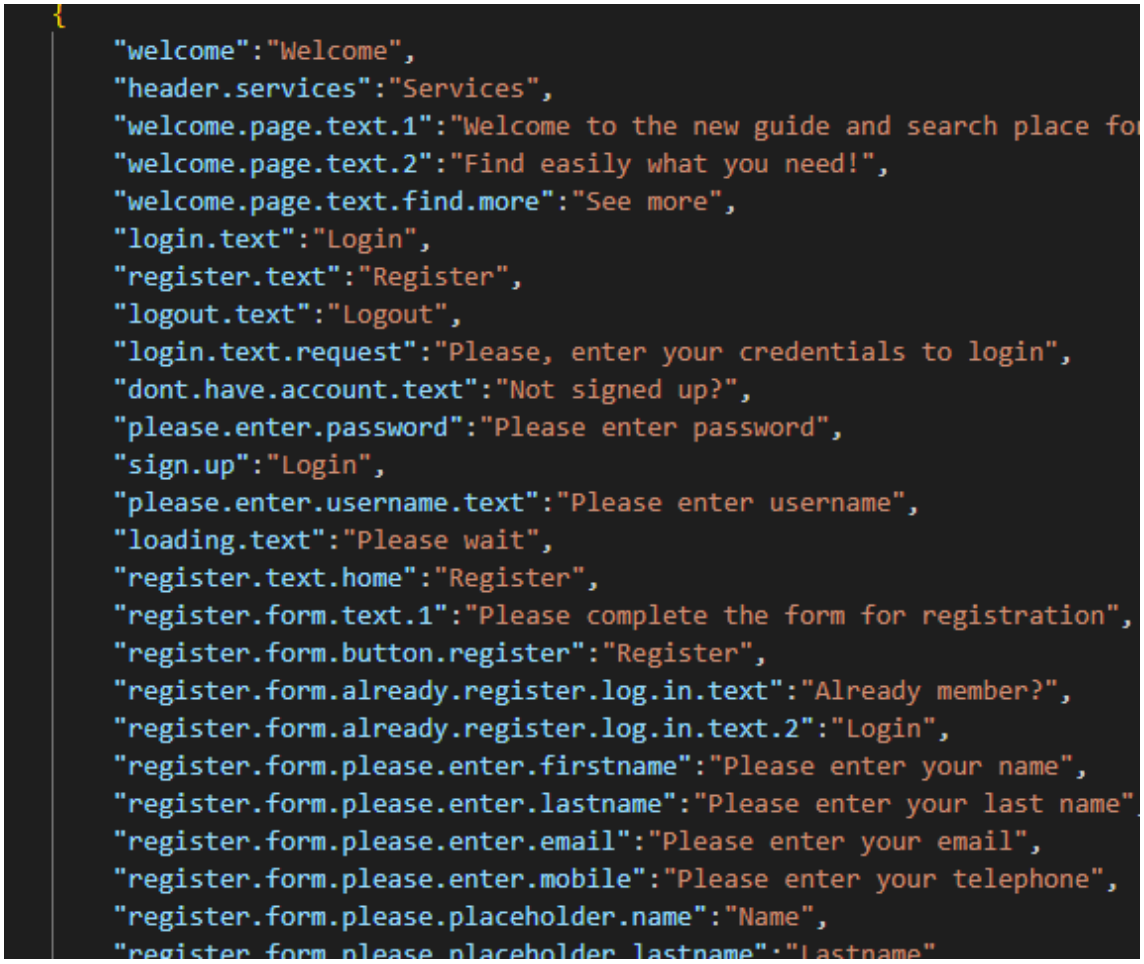
clean install -Dmaven.test.skip=true

Το clean install, σβήνει τυχών compiled αρχεία που είχαν μείνει από προηγούμενο build. Και οι δύο εντολές πρέπει να τρέχουν στο home folder path του project.

Στο front-end κομμάτι χρειάζεται το σύστημα να έχει εγκατεστημένο node version 16.15.0, Angular CLI version 9.1.11. Αφού πρώτα έχει σηκωθεί το back-end, η εντολή για να τρέξει το front-end κομμάτι του λειτουργικού είναι η “ng serve”. Αφού το build ολοκληρωθεί, η σελίδα είναι διαθέσιμη στο URL “http://localhost:4200/”.

4.3 Παραδείγματα και περιγραφή χρήσης συστήματος

Σε αυτό το σημείο πρέπει να σημειωθεί πως το σύστημα είναι υλοποιημένο σε δύο γλώσσες. Για προκαθορισμένη (default) γλώσσα είναι τα Ελληνικά, και επίσης υπάρχει και η επιλογή για αλλαγή σε Αγγλικά κείμενα. Η αλλαγή αυτή γίνεται αυτοματοποιημένα σε όλο το σύστημα μέσω του translator Service της Angular. Επίσης, το σύστημα είναι υλοποιημένο έτσι ώστε, να είναι σχετικά απλή η διαδικασία να προστεθούν μελλοντικά και άλλες γλώσσες. Πιο συγκεκριμένα, όλα τα κείμενα που εμφανίζονται στον χρήστη έχουν ένα μοναδικό κλειδί. Το κλειδί αυτό υπάρχει σε json αρχεία, «en.json» kai «gr.json» για τα Αγγλικά και Ελληνικά αντίστοιχα. Έτσι, αν θέλουμε να προσθέσουμε ακόμα μία γλώσσα για παράδειγμα Γαλλικά, αρκεί να δημιουργηθεί ένα ακόμα json και συγκεκριμένα το «fr.json» και το σύστημα θα μπορεί να δείχνει όλα τα κείμενα στα Γαλλικά.

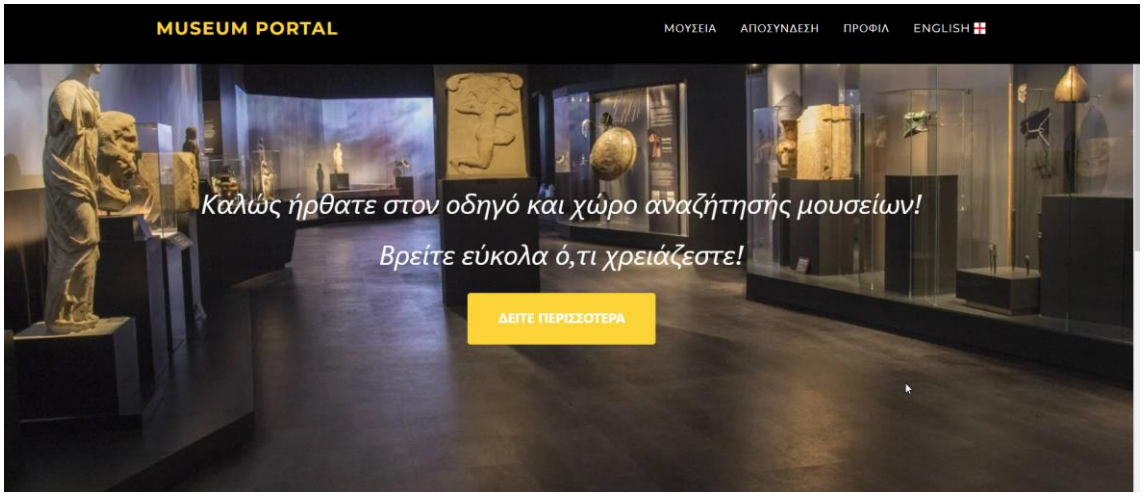


Εικόνα 45 - Η μορφή των json που αποθηκεύονται τα translation keys-values

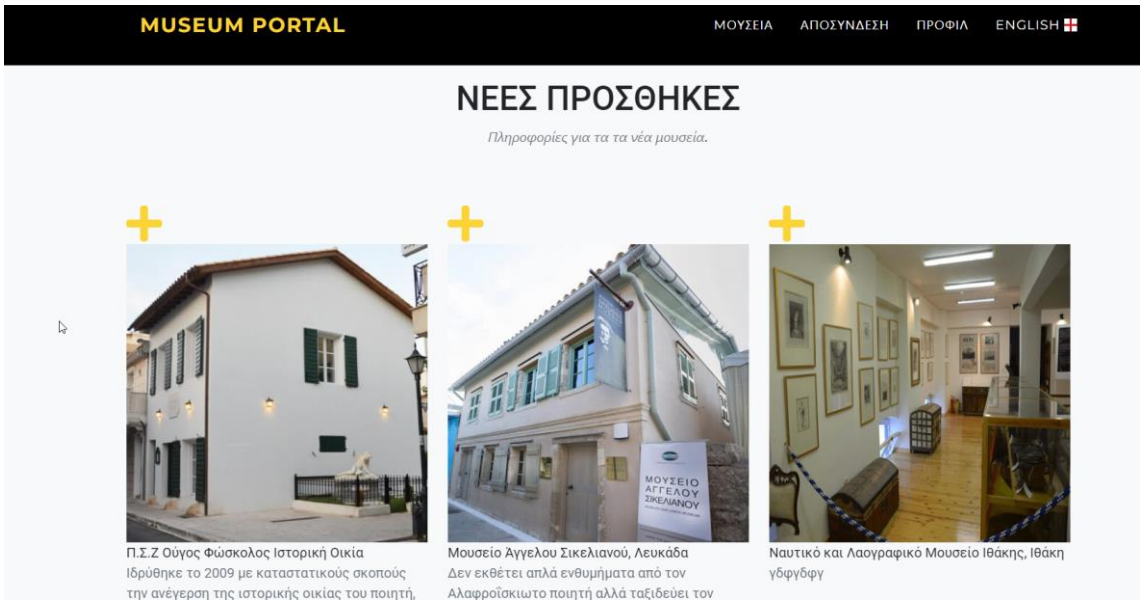
Στην εικόνα 45 βλέπουμε τα αριστερά τα μοναδικά κλειδιά για κάθε κείμενο μέσα στην εφαρμογή, και δεξιά την μετάφρασή τους. Το αρχείο που απεικονίζεται είναι το «en.json», φυσικά αντίστοιχα είναι δομημένο και το json με τις ελληνικές μεταφράσεις, και θα πρέπει να είναι και τα επόμενα αρχεία για τις γλώσσες που ενδεχομένως να προστεθούν στο σύστημα.

4.3.1 Μη Συνδεδεμένος Χρήστης

Τώρα θα γίνει αναφορά στις ενέργειες που μπορεί να πραγματοποιήσει ένας μη συνδεδεμένος χρήστης. Αρχικά, εισέρχεται στην πλατφόρμα και μπορεί να περιηγηθεί σε αυτή και στις σελίδες που δεν χρειάζεται να είναι συνδεδεμένος.

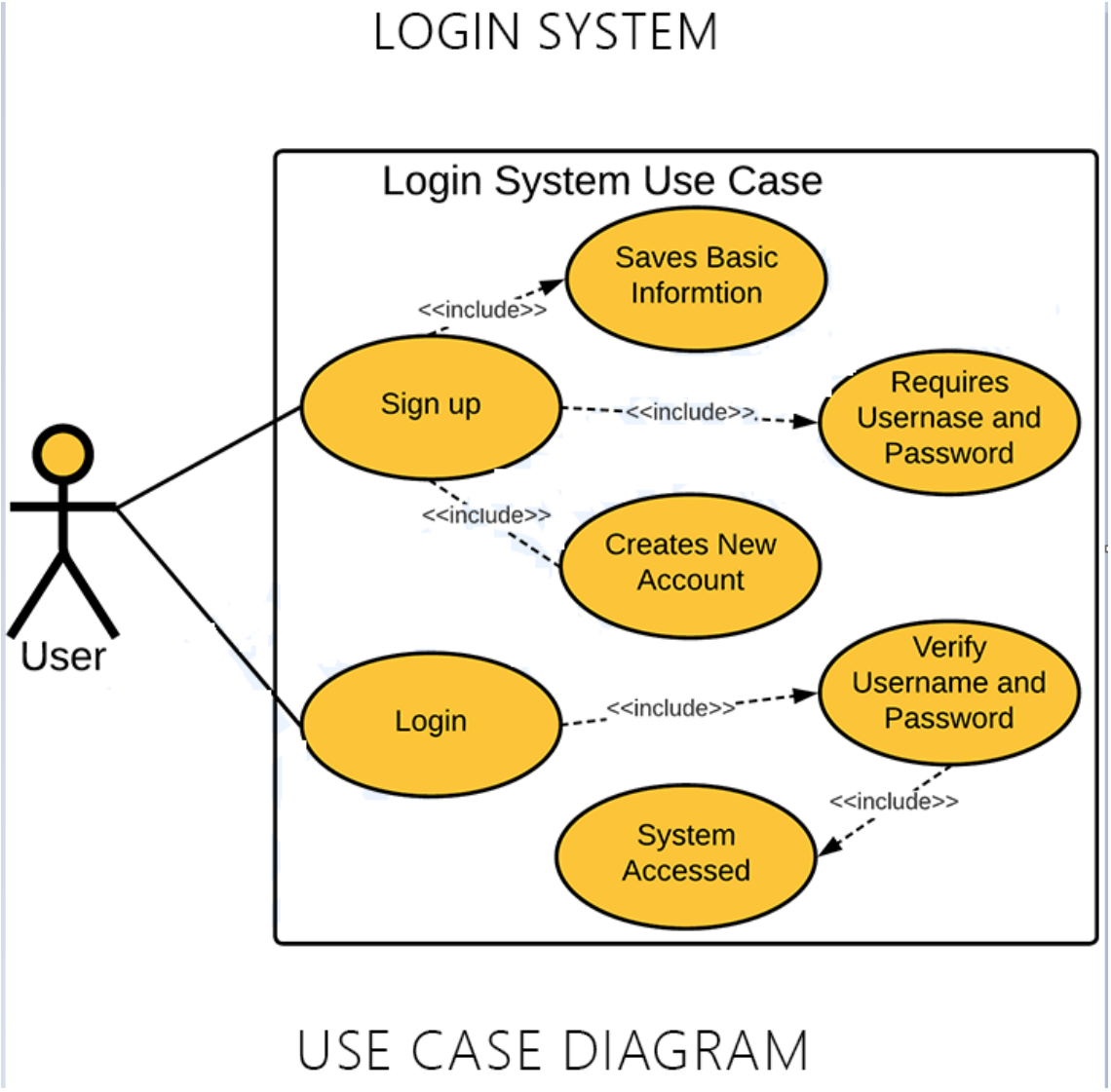


Εικόνα 46 - Αρχική οθόνη στο σύστημα

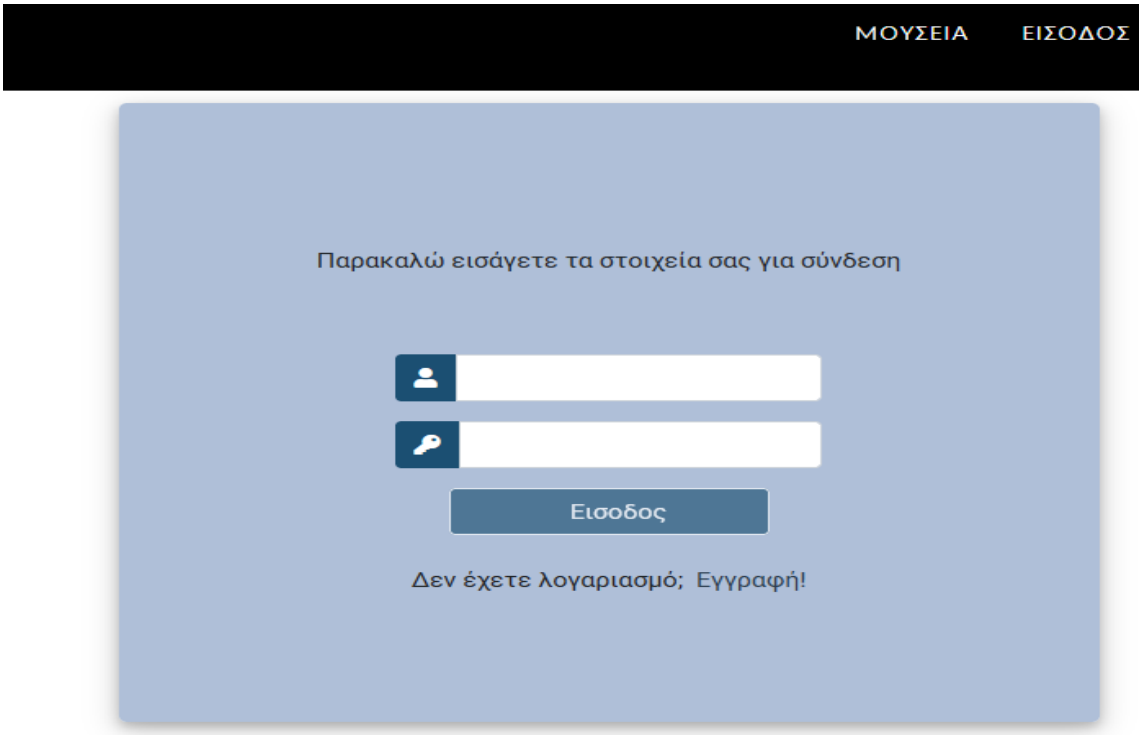


Εικόνα 47 - Αρχική οθόνη, νέες προσθήκες μουσείων

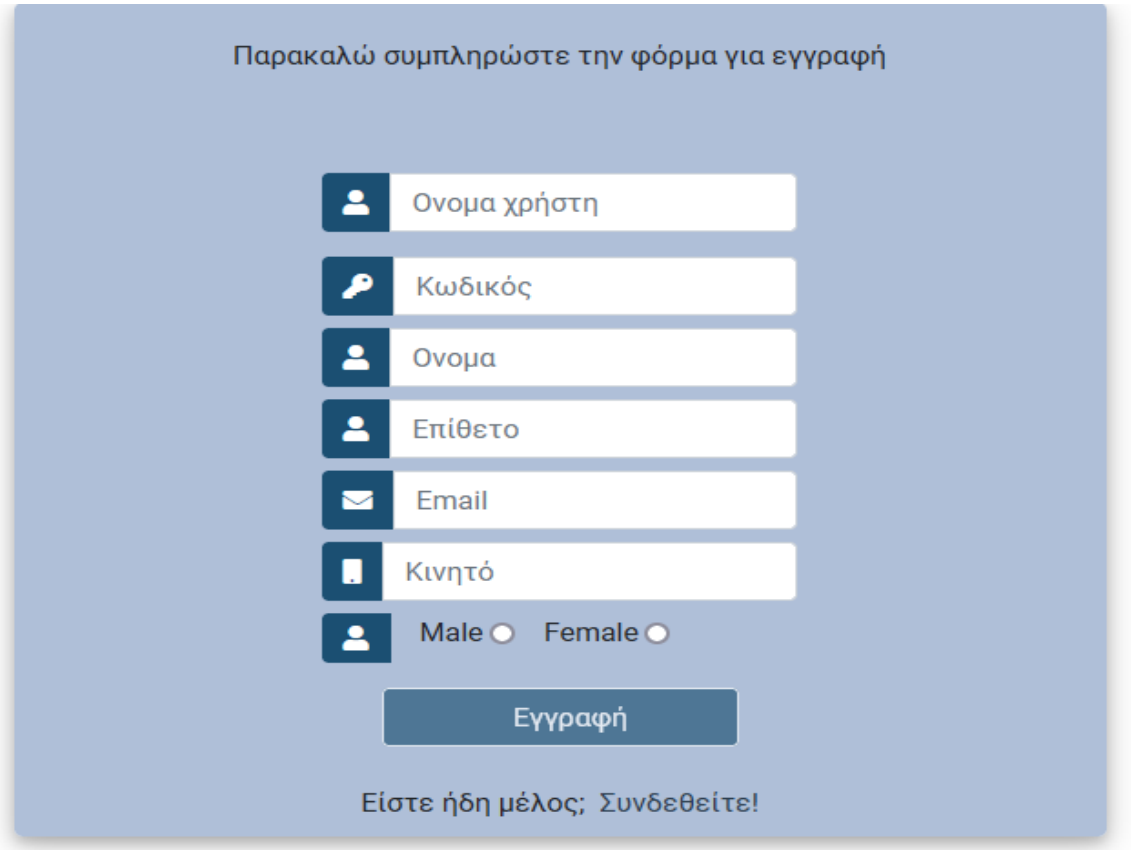
Στις εικόνες 46 και 47 απεικονίζεται η εικόνα που βλέπει ο χρήστης μόλις εισέρθει στο σύστημα. Υπάρχουν οι επιλογές μουσεία, είσοδος, εγγραφή και αλλαγή γλώσσας. Αν επιλέξει την πρώτη επιλογή, θα προβληθούν τα μουσεία του συστήματος στη σελίδα αναζήτησης που έχει φτιαχτεί. Στην είσοδο θα του ζητηθούν τα στοιχεία με τα οποία έχει κάνει εγγραφή όπως φαίνεται στην εικόνα 50. Στην περίπτωση που δεν έχει κάνει εγγραφή στο σύστημα και δεν έχει στοιχεία σύνδεσης, θα πρέπει να ανοίξει την σελίδα της εγγραφής να συμπληρώσει τα στοιχεία του και να ολοκληρώσει την διαδικασία. Η σελίδα με την φόρμα εγγραφής απεικονίζεται στην εικόνα 48. Επίσης, ακολουθεί και διάγραμμα χρήσης για το σύστημα εγγραφής και σύνδεσης στο σύστημα. (Nym 2022b). Στη συνέχεια, το σύστημα εμφανίζει στην αρχική σελίδα πρόσφατες προσθήκες μουσείων που έχουν πραγματοποιηθεί στο σύστημα πάνω στις οποίες ο χρήστης μπορεί να πατήσει και να δει αναλυτικά τις πληροφορίες για το κάθε μουσείο.



Εικόνα 48 – Σύστημα σύνδεσης Διάγραμμα Καταστάσεων(Login System User Case Diagram) (Nym 2022a)

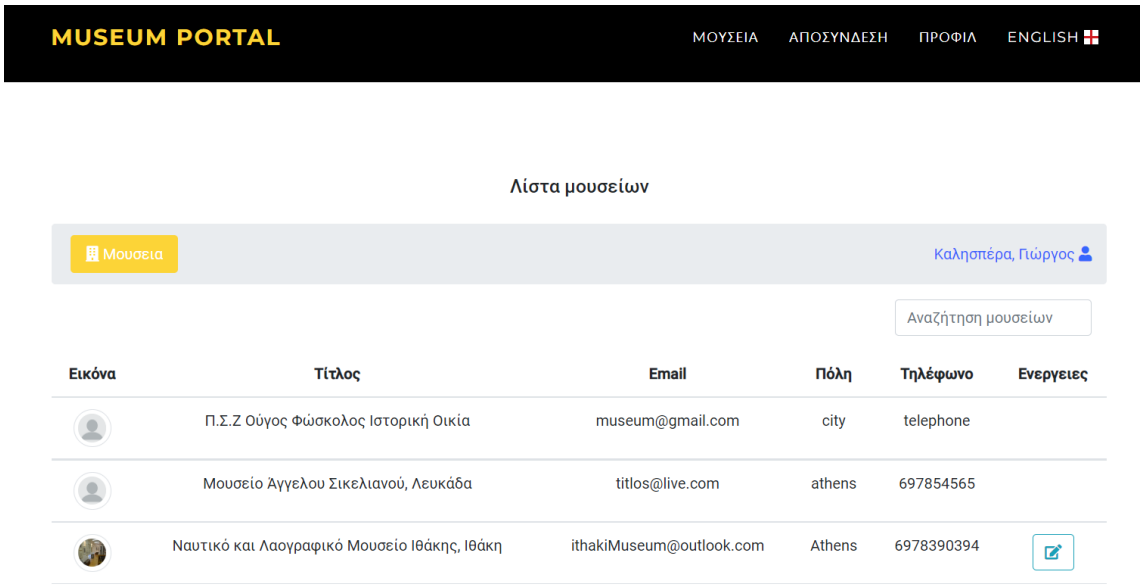


Εικόνα 49 - Φόρμα σύνδεσης στο σύστημα



Εικόνα 50 - Φόρμα εγγραφής στο σύστημα

Στη συνέχεια, ως μη συνδεδεμένος χρήστης όταν ανοίξει την σελίδα για την προβολή μουσείων, θα γίνει προβολή η λίστα με τα μουσεία όπως φαίνεται στην εικόνα 51 που είναι καταχωρημένα στην εφαρμογή. Επίσης, θα μπορεί να αναζητήσει κάποιο μουσείο από αυτά με βάση λέξεις κλειδιά (keywords) που στοχεύουν στο όνομα του μουσείου, το email, την πόλη καθώς και στην πόλη την οποία βρίσκεται το μουσείο. Στη συνέχεια, πατώντας σε ένα από τα μουσεία, θα ανοίξει η σελίδα που τις παρουσιάζει εικόνες και αναλυτικά τις πληροφορίες που απαρτίζουν το μουσείο. Για παράδειγμα την περιγραφή του, τα τηλέφωνα επικοινωνίας, τον ιστότοπο, την διεύθυνση και άλλα. Θα αναλυθούν πιο αναλυτικά στο κεφάλαιο για την καταχώρηση νέου μουσείο μέσα στο σύστημα.



Εικόνα 51 - Η λίστα με τα διαθέσιμα μουσεία

4.3.2 Εγγραφή Σύνδεση Απλού Χρήστη στο Σύστημα

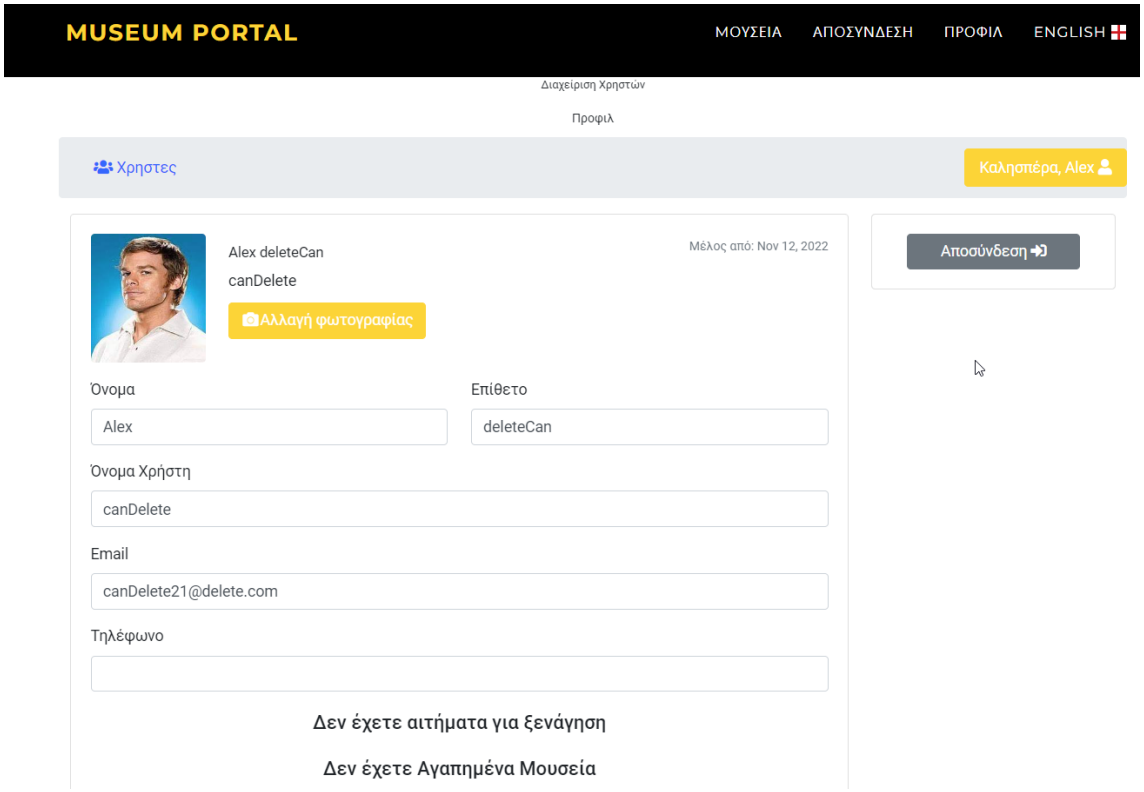
Για να εκκινήσει η διαδικασία εγγραφής στο σύστημα, όπως ήδη ειπώθηκε θα πρέπει ο χρήστης να πραγματοποιήσει πλοήγηση στη σελίδα εγγραφής χρήστη. Το σύστημα θα ζητήσει από τον χρήστη να συμπληρώσει τα εξής πεδία:

- Όνομα χρήστη-Username (υποχρεωτικό)
- Κωδικό-Password (υποχρεωτικό)
- Όνομα (υποχρεωτικό)
- Επίθετο (υποχρεωτικό)
- Email (υποχρεωτικό)
- Κινητό Τηλέφωνο (προαιρετικό)
- Φύλο (προαιρετικό)

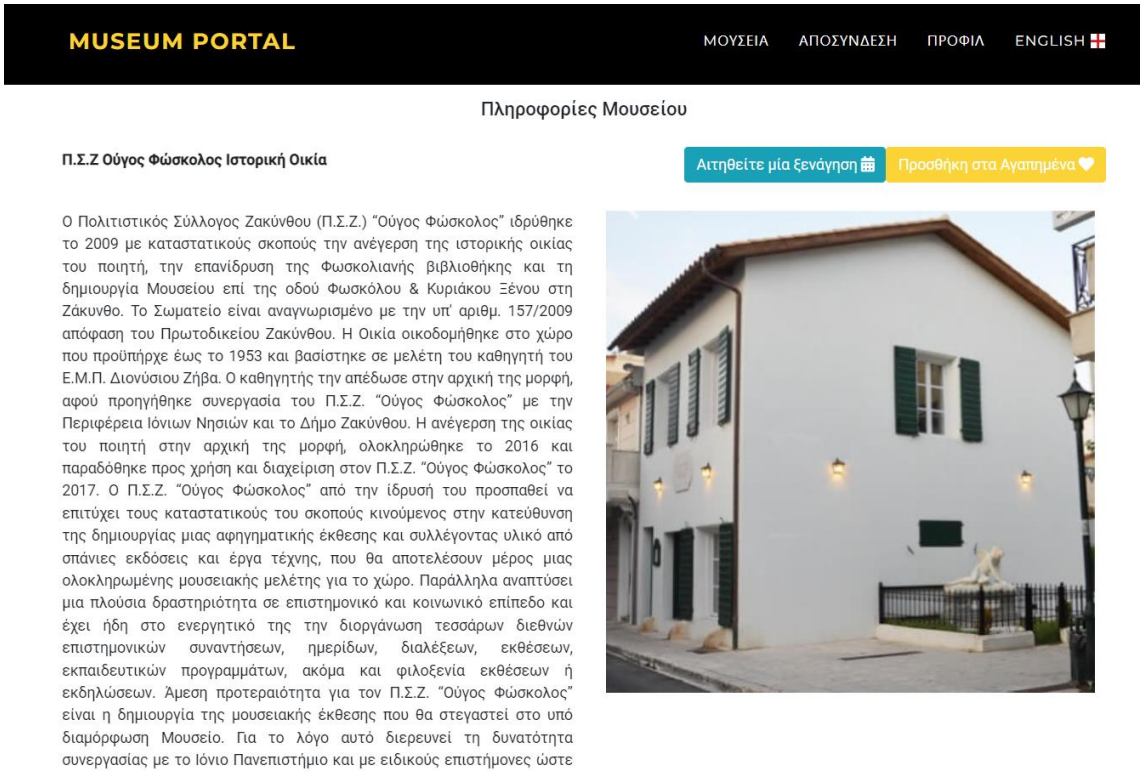
Αφού συμπληρώσει τουλάχιστον όλα τα υποχρεωτικά στοιχεία, τότε μπορεί να ολοκληρώσει την εγγραφή στο σύστημα. Έπειτα από επιτυχής εγγραφή, γίνεται αυτόματη ανακατεύθυνση στην σελίδα σύνδεσης και εκεί ο χρήστης πλέον μπορεί να πραγματοποιήσει είσοδο στο σύστημα. Αφού ο χρήστης πιστοποιηθεί πως έχει ενεργό λογαριασμό και συνδεθεί στο σύστημα, κατευθύνεται στην σελίδα του προσωπικού του προφίλ όπως φαίνεται στην εικόνα 52. Στη σελίδα αυτή, μπορεί να ενημερώσει ή να αλλάξει κάποια από τα στοιχεία με τα οποία πραγματοποίησε

την εγγραφή στο σύστημα όπως για παράδειγμα την διεύθυνση ηλεκτρονικού ταχυδρομείου (email) του ή το όνομα χρήστη. Επίσης, μπορεί να προσθέσει εφόσον το επιθυμεί μία φωτογραφία προφίλ χωρίς ωστόσο η ενέργεια αυτή να είναι υποχρεωτική. Επίσης, ο χρήστης μόλις συνδεθεί, ενημερώνονται-αλλάζουν οι διαθέσιμες επιλογές που εμφανίζονται στην σελίδα και συγκεκριμένα στο οριζόντιο μενού πλοήγησης(navigation menu). Συγκεκριμένα, εμφανίζεται η επιλογή αποσύνδεση και γίνεται απόκρυψη στις επιλογές σύνδεση και εγγραφή. Φυσικά, αν ο χρήστης πραγματοποιήσει αποσύνδεση θα εμφανιστούν εκ νέου. Η επιλογή της αλλαγής γλώσσας, είναι εμφανές και μόνιμα εμφανίσιμη στο πάνω δεξιά μέρος της σελίδας καθώς ο χρήστης πρέπει να μπορεί να κάνει αλλαγή της γλώσσας οποιαδήποτε στιγμή επιθυμεί.

Έπειτα, ως συνδεδεμένος χρήστης μόλις κατευθυνθεί στη σελίδα κάποιου συγκεκριμένου μουσείου, θα εμφανίζεται επιπλέον η επιλογή για να αιτηθεί μία ξενάγηση στο συγκεκριμένο μουσείο. Αυτή η επιλογή προσφέρεται μόνο στον συνδεδεμένο χρήστη, καθώς το σύστημα και κατά επέκταση στη συγκεκριμένη περίπτωση ο διαχειριστής του μουσείου στο οποίο έκανε αίτηση για ξενάγηση, έχει πρόσβαση στα στοιχεία του για να επικοινωνήσει μαζί του και να προγραμματίσουν μία ξενάγηση. Το αίτημα που πραγματοποιήθηκε για την ξενάγηση φαίνεται στο προφίλ του διαχειριστή μουσείου. Από εκεί μόλις ο διαχειριστής του μουσείου επικοινωνήσει και ολοκληρώσει την διαδικασία σχετικά με το αίτημα μπορεί να το διαγράψει. Επίσης, όπως φαίνεται στην εικόνα 52 του προφίλ χρήστη, φαίνονται τα δύο μηνύματα. Συγκεκριμένα τα «Δεν έχετε αιτήματα για ξενάγηση» και «Δεν έχετε Αγαπημένα Μουσεία». Στο πρώτο, αν ο χρήστης είχε υπό την διαχείρισή του κάποιο μουσείο, θα υπήρχε από κάτω ένας πίνακας με όλα τα αιτήματα για ξενάγηση. Ακόμα, για το δεύτερο μήνυμα, από κάτω εμφανίζεται ένας πίνακας στον οποίο εμφανίζονται μέσα τα αγαπημένα μουσεία που έχει προσθέσει ο χρήστης στο προφίλ του. Φυσικά, υπάρχει και η δυνατότητα να αφαιρέσει κάποιο από τα μουσεία από την λίστα των αγαπημένων του.



Εικόνα 52 - Σελίδα προφίλ χρήστη



Εικόνα 53 - Σελίδα Μουσείου με δυνατότητα αιτήματος για περιήγηση και προσθήκη στα αγαπημένα

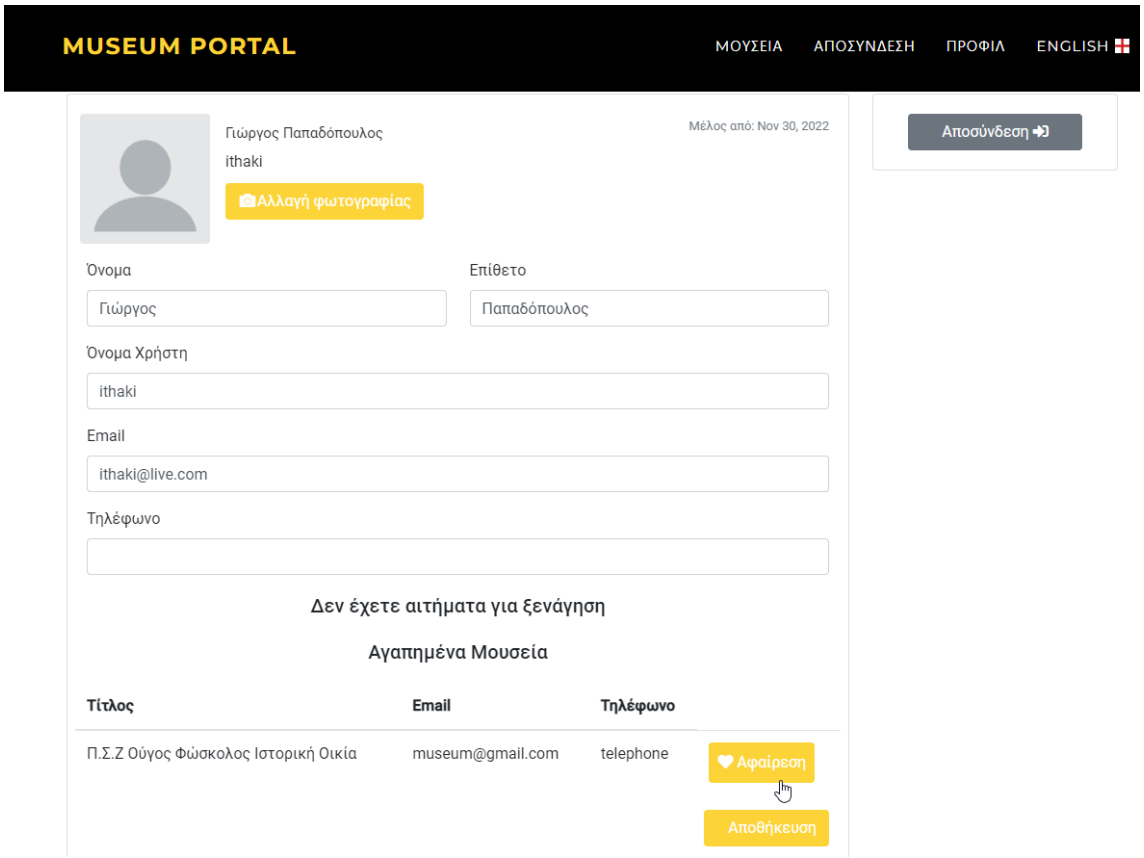
Όπως φαίνεται στην εικόνα 53, ο χρήστης έχει πραγματοποιήσει σύνδεση στο σύστημα και έχει προηγηθεί στην σελίδα προβολής συγκεκριμένου μουσείου. Μέσα στην σελίδα, μπορεί να δει τις πληροφορίες για το μουσείο. Πιο συγκεκριμένα:

- Όνομα του μουσείου
- Περιγραφή του μουσείο
- Διεύθυνση του μουσείου
- Τηλέφωνα επικοινωνίας για το μουσείο
- Που είναι η διάρκεια αλλά και το κόστος ξενάγησης στο μουσείο
- Διεύθυνση Email του μουσείου για επικοινωνία
- Διεύθυνση επίσημης ιστοσελίδας του μουσείου(αν υπάρχει)
- Διεύθυνση επίσημου προφίλ του μουσείου στο Facebook(αν υπάρχει)
- Διεύθυνση επίσημου προφίλ του μουσείου στο Twitter(αν υπάρχει)
- Διεύθυνση επίσημου προφίλ του μουσείου στο YouTube(αν υπάρχει)
- Κουμπί για προσθήκη μουσείου στα αγαπημένα του χρήστη
- Κουμπί για αίτημα ξενάγησης στο μουσείο

Όλα τα παραπάνω απαρτίζουν μία πλήρης περιγραφή του μουσείου και των σημαντικών στοιχείων που μπορεί να χρειαστεί ένας χρήστης στην αναζήτησή του. Σε περίπτωση που το επιθυμεί ο χρήστης μπορεί να επικοινωνήσει μέσω τηλεφώνου, να δει την διεύθυνση ώστε να μεταβεί στο μουσείο, να ανατρέξει στη σελίδα του Facebook για τυχόν ενημερώσεις και τέλος να μεταβεί στην επίσημη ιστοσελίδα του μουσείου εφόσον το επιθυμεί έτσι ώστε να δει κάποια

συγκεκριμένη πληροφορία όπως για παράδειγμα κάποιο έκθεμα, πληροφορίες για το προσωπικό του μουσείου. Η λειτουργικότητα να το προσθέσει κάποιο μουσείο στα αγαπημένα του δίνει την δυνατότητα, να μπορεί να περιηγηθεί εύκολα στο μουσείο μέσα από το προφίλ του έτσι ώστε να βλέπει κάποια ενημέρωση στα στοιχεία όπως το κόστος και η διάρκεια ξενάγησης. Θα ακολουθήσει τώρα συγκεκριμένο παράδειγμα προσθήκης μουσείου στα αγαπημένα, και πως φαίνεται μέσα στο προφίλ του χρήστη στην πλατφόρμα που υλοποιήθηκε στα πλαίσια της διατριβής.

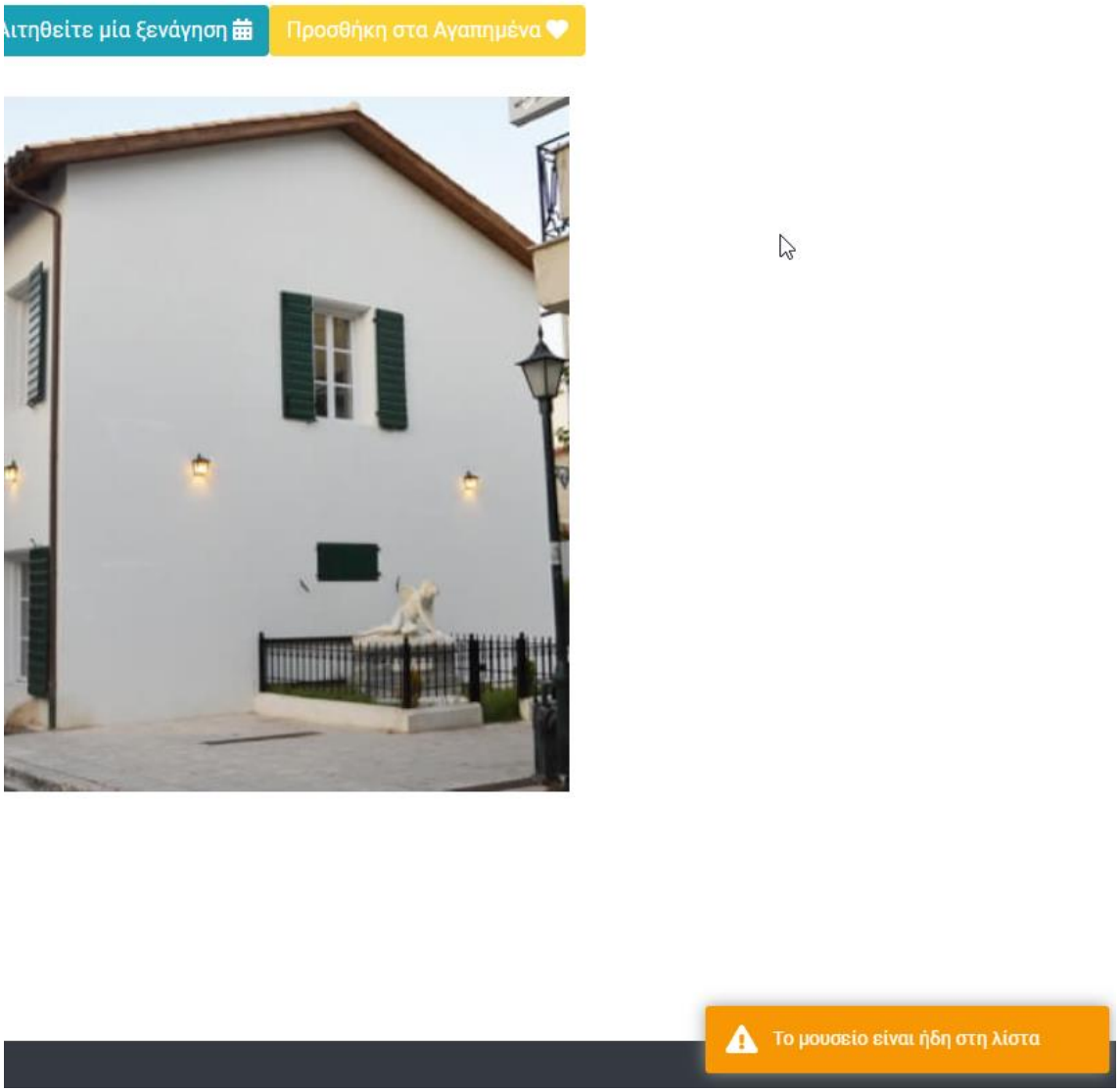
Στο συγκεκριμένο λοιπόν παράδειγμα, ο χρήστης έχει πραγματοποιήσει σύνδεση στο σύστημα και κατευθύνεται στην σελίδα του μουσείου «Ούγος Φώσκολος Ιστορική Οικία». Μόλις πατηθεί το κουμπί για την προσθήκη του μουσείου στα αγαπημένα και ενημερώνεται πως η προσθήκη ολοκληρώθηκε επιτυχώς μέσω pop-up μήνυμα που εμφανίζεται στο κάτω δεξιά μέρος της οθόνης. Στη συνέχεια κατευθύνεται στην προσωπική σελίδα προφίλ του μέσα στην πλατφόρμα πατώντας στο κουμπί προφίλ. Εκεί πλέον θα εμφανίζεται στον χρήστη το μουσείο που πρόσθεσε στα αγαπημένα όπως απεικονίζεται στην Εικόνα 54.



Εικόνα 54 - Εμφάνιση των αγαπημένων μουσείων στο προφίλ του χρήστη και επιλογή αφαίρεσης από τα αγαπημένα

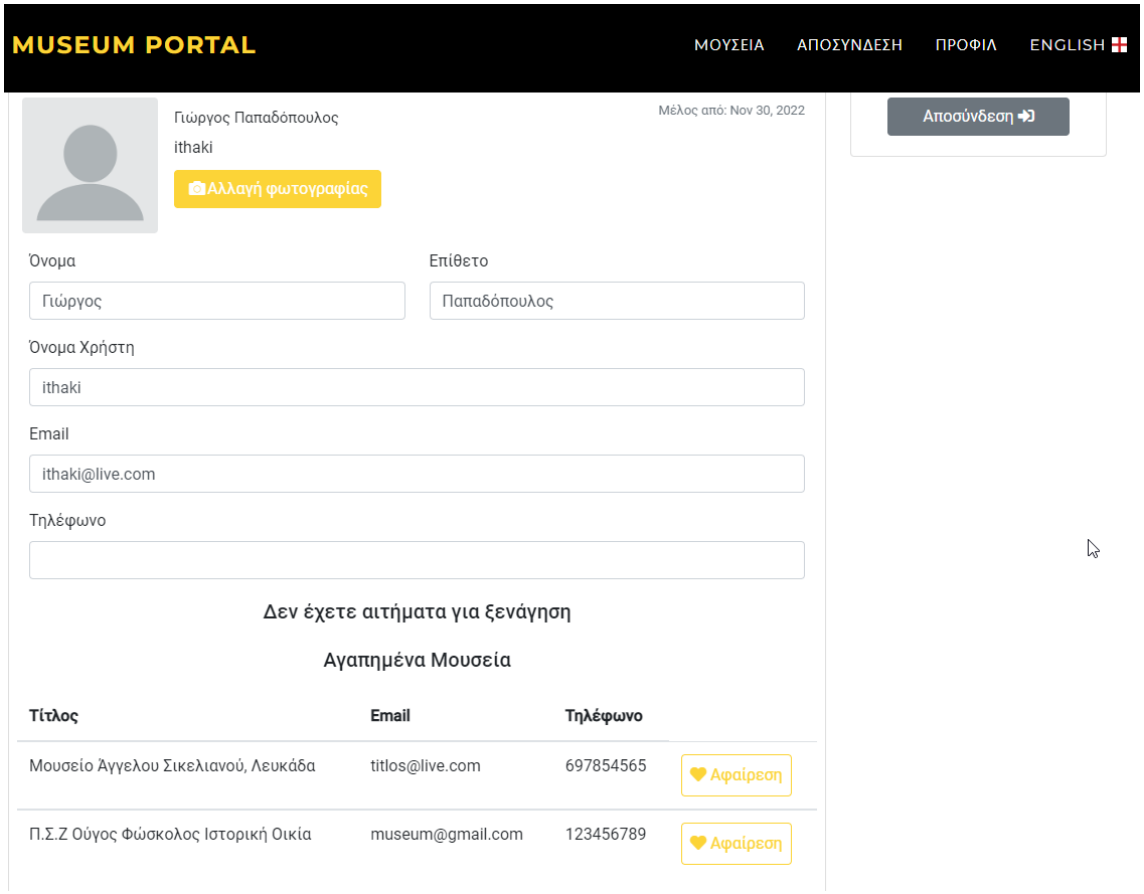
Στο προφίλ του ο χρήστης λοιπόν, βλέπει τα αγαπημένα μουσεία και έχει την δυνατότητα να πατήσει στην γραμμή για το μουσείο έτσι ώστε να του ανοίξει η προσωπική σελίδα του μουσείου και να δει παραπάνω λεπτομέρειες για το μουσείο. Επίσης, υπάρχει ακόμα η δυνατότητα να πατήσει πάνω στο κουμπί «αφαίρεση» για να αφαιρεθεί από την λίστα των αγαπημένων μουσείων σε περίπτωση που το επιθυμεί. Σε συνέχεια, ο χρήστης μπορεί να προσθέσει φυσικά και άλλα μουσεία στην λίστα των αγαπημένων του. Αν προσπαθήσει να προσθέσει ένα μουσείο που υπάρχει ήδη στην λίστα, το σύστημα είναι υλοποιημένο έτσι ώστε φυσικά να μην βάζει το ίδιο

μουσείο δύο φορές, αλλά και να ενημερώσει τον χρήστη πως το συγκεκριμένο μουσείο βρίσκεται ήδη στα αγαπημένα του το μήνυμα φαίνεται στην εικόνα 55.



Εικόνα 55 - Μήνυμα ενημέρωσης του χρήστη πως το μουσείο που προσπάθησε να προσθέσει στην λίστα αγαπημένων υπάρχει ήδη

Στην περίπτωση που το μουσείο δεν υπήρχε μέσα στις υπάρχον επιλογές αγαπημένων μουσείων το αίτημα φυσικά θα ολοκληρωνόταν και ο χρήστης θα έβλεπε την καινούργια προσθήκη στην λίστα των αγαπημένων του όπως φαίνεται και στην εικόνα 56.



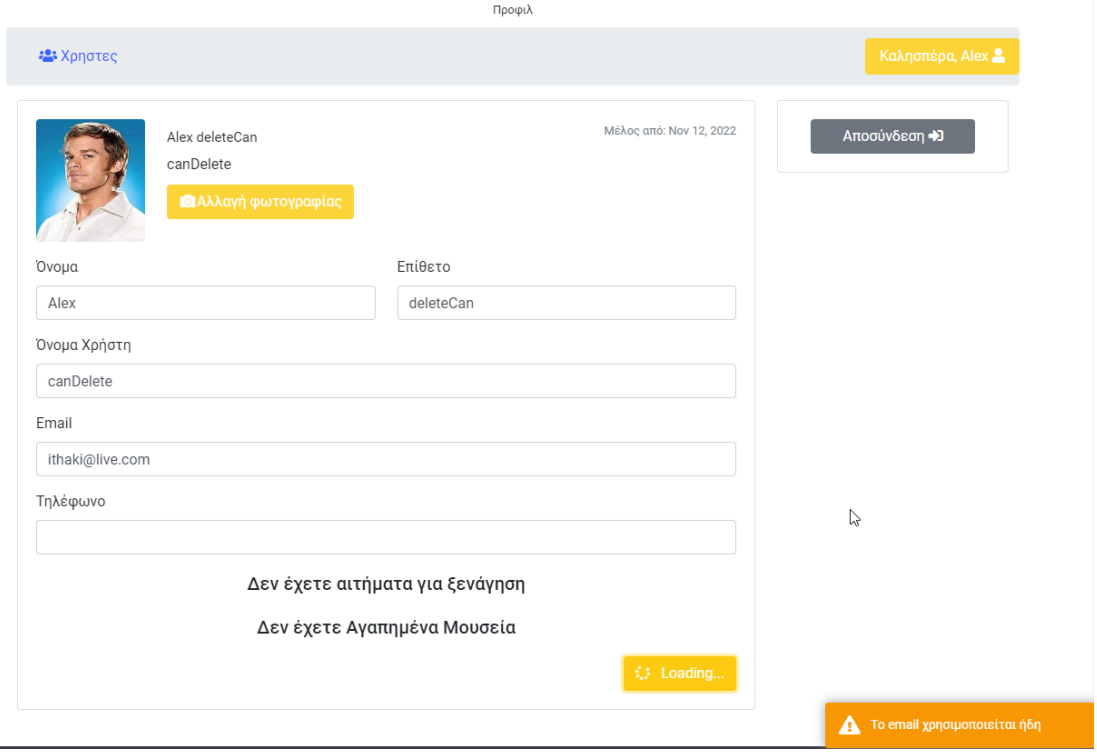
Εικόνα 56 - Απεικόνιση λίστας αγαπημένων στο προφίλ συνδεδεμένου χρήστη

Απεικονίζεται λοιπόν η λειτουργικότητα της αποθήκευσης των μουσείων στη λίστα αγαπημένων για τον κάθε έναν χρήστη. Τέλος, δίνεται η δυνατότητα στο χρήστη να κάνει αλλαγή των δεδομένων που έχει καταχωρήσει μέσα στην εφαρμογή. Για να πραγματοποιήσει μία τέτοια ενέργεια, πρέπει να προηγηθεί πλοήγηση στην σελίδα του προσωπικού του προφίλ. Από εκεί, έχει πρόσβαση να δει όλα τα στοιχεία του και συγκεκριμένα:

- Όνομα
- Επίθετο
- Όνομα Χρήστη
- Διεύθυνση Email
- Τηλέφωνο επικοινωνίας

Η διαδικασία της ενημέρωσης στοιχείων έχει υλοποιηθεί με τέτοιο τρόπο ώστε να είναι γρήγορη και να μην δημιουργεί δυσκολίες στο χρήστη. Σε αυτό το σημείο, αξίζει να σημειωθεί πως όπως έχει ήδη αναφερθεί κάποια στοιχεία των χρηστών είναι μοναδικά και τα αναγνωριστικά τους μέσα στο σύστημα. Για παράδειγμα, δεν μπορούν δύο διαφορετικοί χρήστες να έχουν κοινή διεύθυνση ηλεκτρονικού ταχυδρομείου(email) ή όνομα χρήστη(username). Αυτό δεν είναι εφικτό και από άποψη επιχειρησιακής λογικής καθώς το προφίλ κάθε χρήστη πρέπει να είναι μοναδικό, αλλά είναι και επίσης τεχνικός περιορισμός καθώς αυτά τα στοιχεία χρησιμοποιούνται για την αναγνώριση των χρηστών στο σύστημα. Ως αποτέλεσμα, όπως γίνεται κατανοητό σε περίπτωση που ο χρήστης συμπληρώσει κάποια διεύθυνση ηλεκτρονικού ταχυδρομείου ή όνομα χρήστη ή τηλέφωνο που χρησιμοποιείται ήδη από κάποιον άλλο εγγεγραμμένο χρήστη στο σύστημα η ανανέωση των στοιχείων δε θα μπορέσει να ολοκληρωθεί. Το σύστημα θα ανταποκριθεί με

μήνυμα που θα ενημερώνει τον χρήστη πως τα στοιχεία που έχει συμπληρώσει χρησιμοποιούνται ήδη από άλλο χρήστη του συστήματος και να δοκιμάσει ξανά με διαφορετικά στοιχεία.



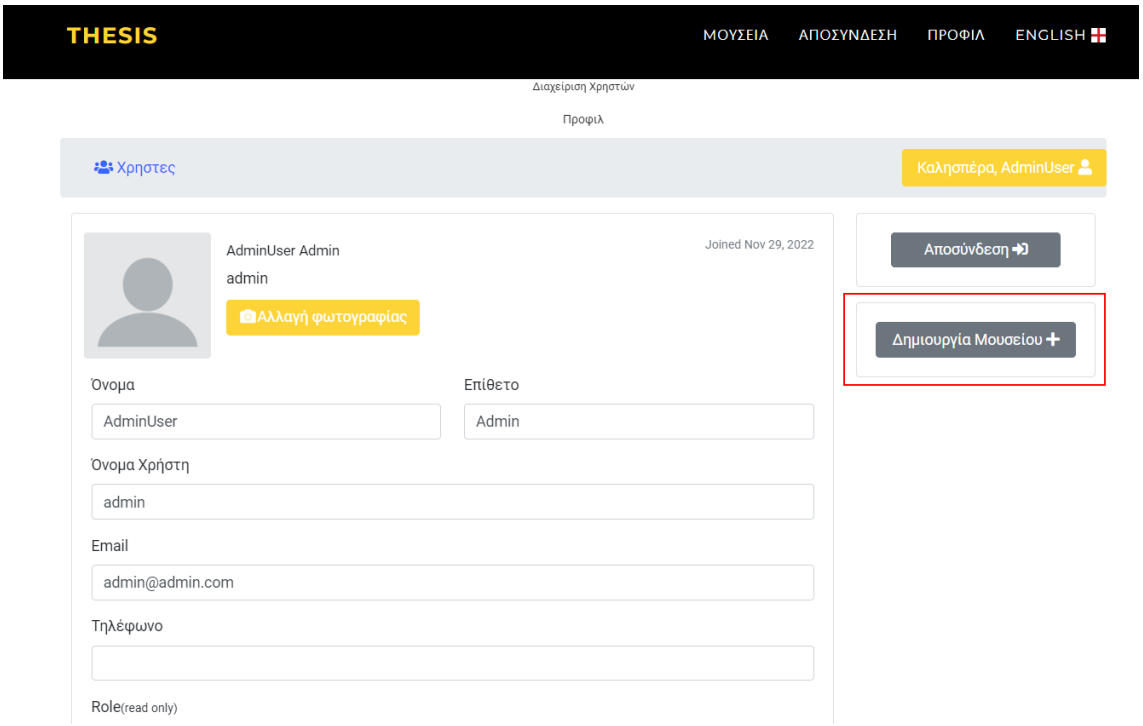
Εικόνα 57 - Ενημέρωση στοιχείων με Email που χρησιμοποιείται από άλλο χρήστη και ανταπόκριση συστήματος

Σε συνέχεια του μηνύματος που εμφανίζεται επειδή ο χρήστης έβαλε ίδια διεύθυνση ηλεκτρονικού ταχυδρομείου με άλλο χρήστη, μόλις ο χρήστης ολοκληρώσει τις απαραίτητες αλλαγές στα στοιχεία του όπως να αλλάξει την καινούργια διεύθυνση ηλεκτρονικού ταχυδρομείου που θέλει να χρησιμοποιεί και πατήσει αποθήκευση, τα νέα στοιχεία θα καταχωρηθούν στην βάση δεδομένων της εφαρμογής και η διαδικασία θα έχει πλέον ολοκληρωθεί με επιτυχία. Αντίστοιχα, ο χρήστης θα ενημερωθεί με μήνυμα που θα εμφανιστεί στο κάτω δεξιά μέρος της οθόνης του πως ολοκληρώθηκε με επιτυχία η διαδικασία ενημέρωσης των στοιχείων του. Φυσικά, πλέον στα παλιά αλλά και στα καινούργια αιτήματα για ξενάγηση που έχει ή θα πραγματοποιήσει αντίστοιχα, καθώς και στον διαχειριστή του συστήματος θα φαίνονται τα καινούργια στοιχεία που συμπλήρωσε και αποθήκευσε στο σύστημα στην διαδικασία που μόλις αναφέρθηκε.

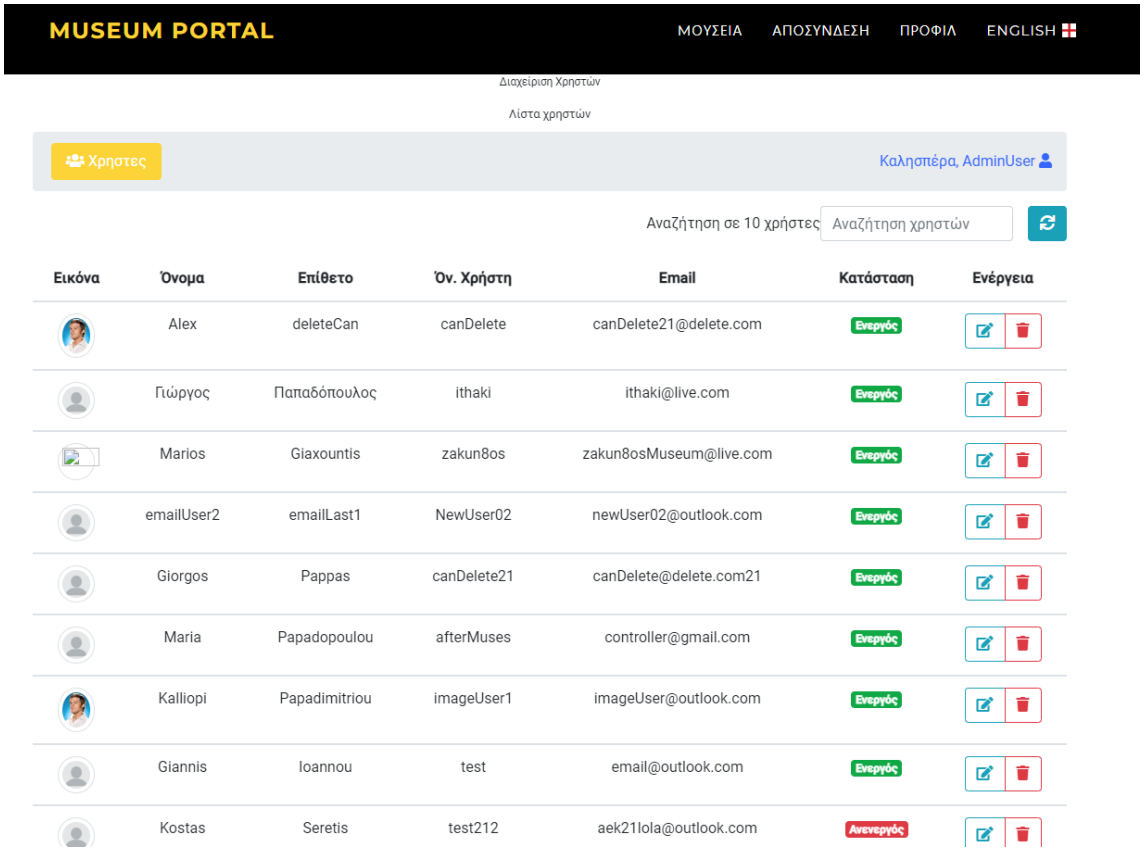
4.3.3 Σύνδεση Διαχειριστή(Admin) Χρήστη στο σύστημα

Στο σύστημα όπως αναφέρθηκε, υπάρχει ο χρήστης admin δηλαδή ένας super user που έχει δικαιώματα για ενέργειες όπως την διαγραφή χρηστών, την διαγραφή μουσείων ή και το μπλοκάρισμα χρηστών ώστε να μην τους επιτρέπεται η σύνδεση στο σύστημα. Ο χρήστης αυτός, όπως γίνεται κατανοητό δεν μπορεί να κάνει εγγραφή στο σύστημα από την σελίδα για λόγους ασφαλείας. Η διαφοροποίηση που υπάρχει για αυτόν τον χρήστη στο σύστημα είναι πως όταν πηγαίνει στο προσωπικό του προφίλ στην σελίδα, υπάρχει και μία νέα καρτέλα την οποία αν επιλέξει θα δει όλους τους ενεργούς και μη ενεργούς χρήστες που βρίσκονται στο σύστημα. Θα μπορεί να πατήσει σε κάθε έναν από αυτούς για να δει το προφίλ τους, θα μπορεί να τους αλλάξει τα στοιχεία και να τους απενεργοποιήσει τον λογαριασμό. Εδώ, είναι σημαντικό να αναφερθεί πως υπάρχει και η επιλογή στην παραμετροποίηση χρήστη από τον admin που του δίνει την

δυνατότητα να μπορεί να δημιουργήσει κάποιο μουσείο. Αυτή η ενέργεια γίνεται μέσα από τον admin για λόγους ασφαλείας ώστε να μην μπορεί ο κάθε ένας χρήστης να δημιουργεί προφίλ από μουσεία, αλλά μόνο πιστοποιημένοι χρήστες. Αν λοιπόν, δώσει σε κάποιον χρήστη δυνατότητα να δημιουργήσει μουσείο, ο χρήστης μόλις συνδεθεί στο σύστημα στο προσωπικό του προφίλ, θα έχει εμφανιστεί μία νέα επιλογή «Δημιουργία μουσείου» που προηγουμένως δεν υπήρχε. Επίσης, ο admin χρήστης έχει την δυνατότητα να διαγράψει κάποιον χρήστη οριστικά από το σύστημα, όπως φαίνεται στην εικόνα 59. Όλες οι ενέργειες αυτές καθώς επηρεάζουν σημαντικά τα δεδομένα του συστήματος, αξίζει να αναφερθεί ξανά πως φυσικά προστατεύονται τόσο και στο front-end κομμάτι όσο και στο back-end με την χρήση μεθόδων ασφαλείας.



Εικόνα 58 - Προφίλ Admin User με δυνατότητα δημιουργίας μουσείου

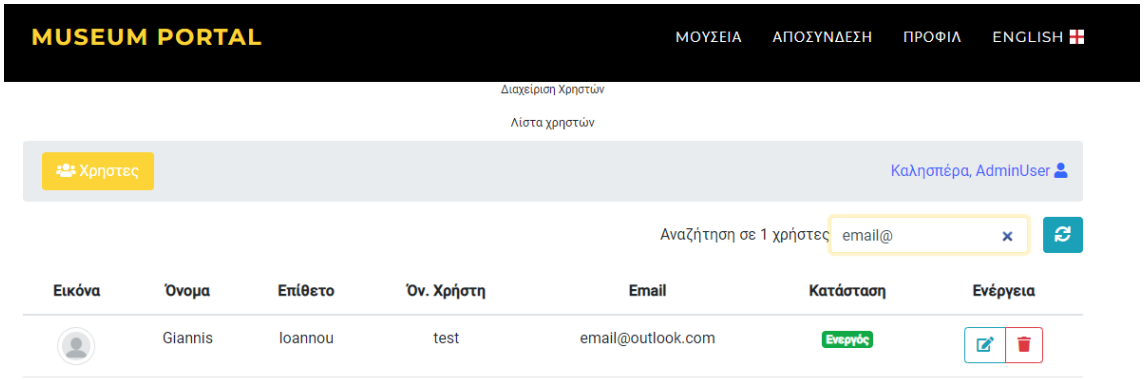


Εικόνα 59 - Λίστα χρηστών εφαρμογής με δυνατότητα διαγραφής χρήστη

Ακόμα, στην σελίδα αναζήτησης και παρουσίασης χρηστών στον super Admin της πλατφόρμας έχει υλοποιηθεί και είναι διαθέσιμη για χρήση η αναζήτηση χρηστών μέσω πληκτρολόγησης δεδομένων. Συγκεκριμένα, όταν ο super Admin πληκτρολογεί δεδομένα στην μπάρα αναζήτησης το σύστημα αναζητεί χρήστες με βάσει αυτά που πληκτρολογεί ο χρήστης. Πιο συγκεκριμένα, η αναζήτηση μουσείου στοχεύει στις στήλες με τα δεδομένα:

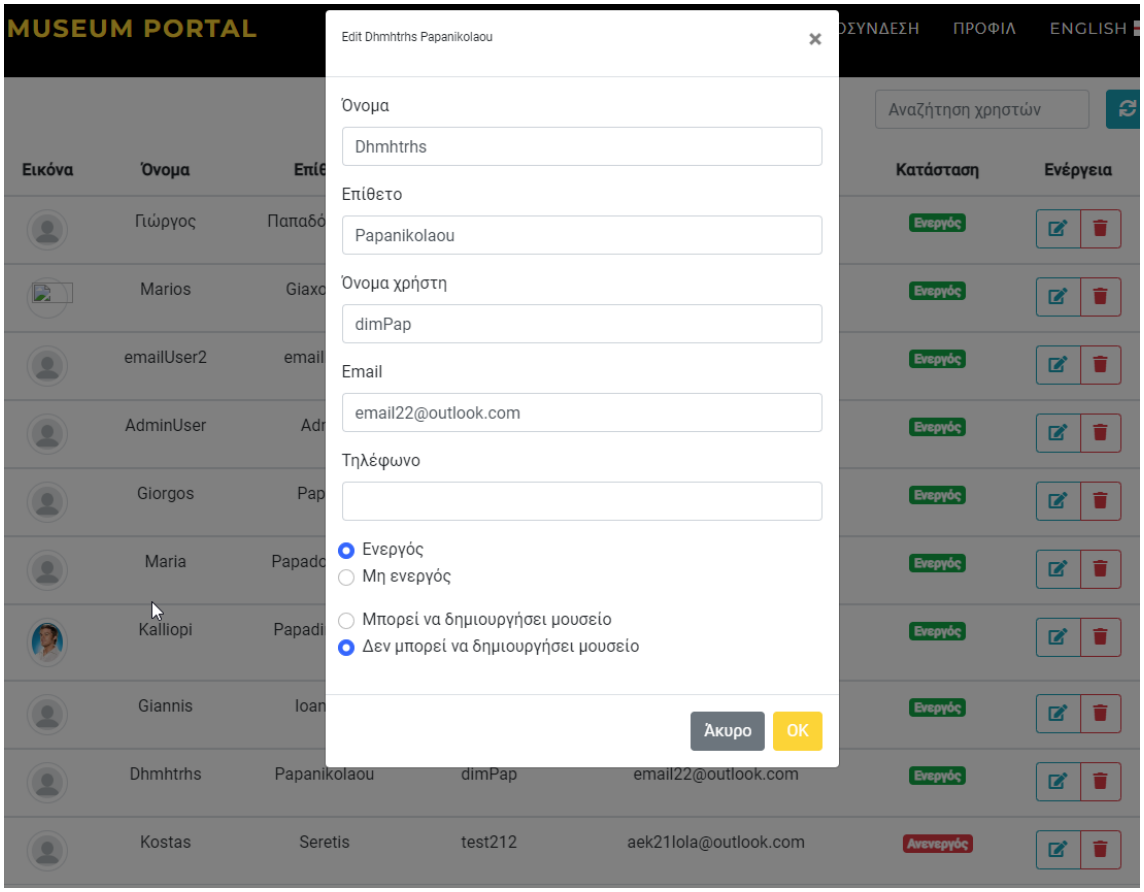
- Όνομα
- Επίθετο
- Email

Αυτό έχει υλοποιηθεί με σκοπό να διευκολύνει τον χρήστη στην αναζήτηση των διαθέσιμων χρηστών. Αυτό επιτυγχάνεται επειδή το σύστημα αναζητά ταυτόχρονα και στις τρεις στήλες δεδομένων που αναφέρθηκαν και ανάλογα σε ποια βρει θα επιστρέψει το αποτέλεσμα. Ακολουθεί η εικόνα 60 η οποία δείχνει πως είναι το αποτέλεσμα αναζήτησης μόλις ο χρήστης έχει πληκτρολογήσει χαρακτήρες στην μπάρα αναζήτησης.



Εικόνα 60 - Λίστα χρηστών εφαρμογής με εφαρμοσμένο το φίλτρο αναζήτησης

Επιπλέον, ο super Admin της πλατφόρμας πέρα από την δυνατότητα που έχει να διαγράφει χρήστες με το κουμπί που φαίνεται και στις παραπάνω εικόνες, έχει επίσης την δυνατότητα να ενημερώσει τα στοιχεία των χρηστών. Αυτό πραγματοποιείται επίσης από την σελίδα των χρηστών. Πιο συγκεκριμένα, ο super Admin πατάει στο κουμπί edit που βρίσκεται σε κάθε γραμμή του πίνακα των χρηστών, και τότε εμφανίζεται στο κέντρο της οθόνης ένα παράθυρο(modal) στο οποίο εμφανίζονται τα στοιχεία του χρήστη αλλά και δύο έξτρα λειτουργικότητες που είναι αποκλειστικό προνόμιο του super Admin. Τέλος, μπορεί να δει οποιαδήποτε στιγμή τον αριθμό των εγγεγραμμένων χρηστών που βρίσκονται μέσα στην πλατφόρμα.



Εικόνα 61 - Modal που εμφανίζεται στον super Admin του συστήματος και δίνει την δυνατότητα αλλαγής στοιχείων άλλου εγγεγραμμένου χρήστη

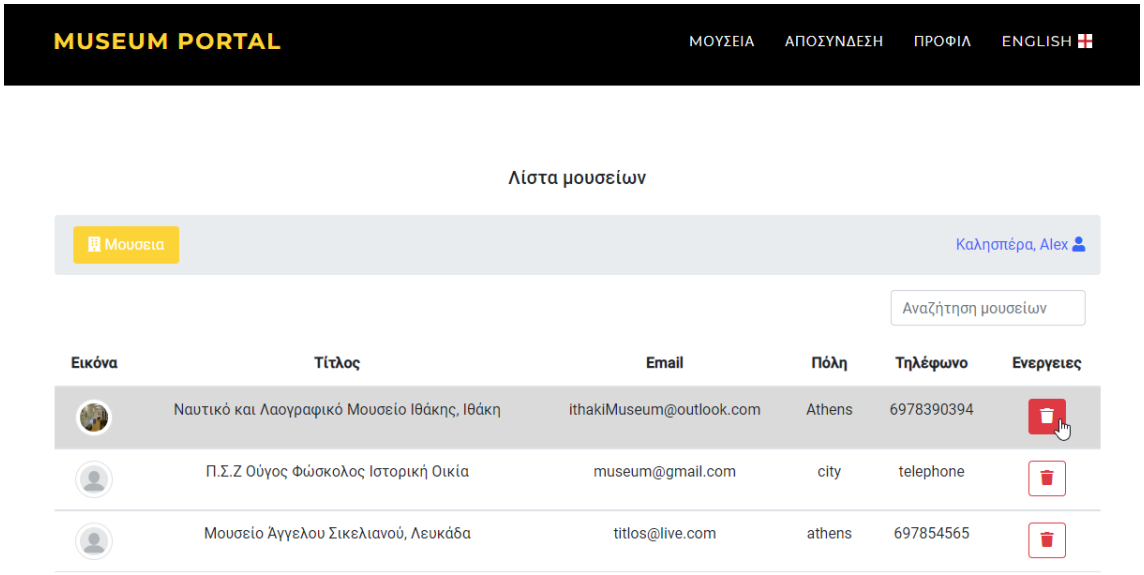
Απεικονίζονται τα στοιχεία που μπορεί να αλλάξει σε άλλους εγγεγραμμένους χρήστες και συγκεκριμένα:

- Όνομα
- Επίθετο
- Όνομα Χρήστη
- Email
- Τηλέφωνο
- Ενεργός – Μη Ενεργός
- Μπορεί να δημιουργήσει μουσείο – Δεν μπορεί να δημιουργήσει μουσείο

Τα πέντε πρώτα στοιχεία, είναι δεδομένα τα οποία μπορεί επίσης να ενημερώσει και ο χρήστης στον οποίο ανήκουν. Η μεγάλη διαφορά όπως φαίνεται στην εικόνα 61 είναι τα δύο τελευταία στοιχεία που ενημερώνονται από τα radio buttons. Αναλυτικά, το «Ενεργός-μη», «ενεργός» έχει την λειτουργικότητα πως σε περίπτωση που επιλεχθεί «μη-ενεργός» και έπειτα αποθήκευση, ο χρήστης στον οποίο τέθηκε αυτή η επιλογή δε θα μπορεί να πραγματοποιήσει είσοδο στο σύστημα. Φυσικά, αν ο super admin επαναφέρει πάλι την επιλογή στο «Ενεργός» τότε ο χρήστης θα μπορεί και πάλι να συνδέεται φυσιολογικά στο σύστημα.

Στη συνέχεια, υπάρχει και η επιλογή «Δεν μπορεί να δημιουργήσει μουσείο» και «Μπορεί να δημιουργήσει μουσείο». Αυτές οι δύο επιλογές διαφοροποιούν και δίνουν την δυνατότητα σε έναν χρήστη να δημιουργεί και να διαχειρίζεται κάποιο μουσείο. Δηλαδή, αν επιλεχθεί η πρώτη επιλογή, όταν ο χρήστης του οποίου το προφίλ ενημερώθηκε πραγματοποιήσει σύνδεση στο σύστημα, στη σελίδα του προσωπικού του προφίλ πλέον θα εμφανίζεται το κουμπί και η δυνατότητα να δημιουργήσει και να διαχειρίζεται ένα νέο μουσείο.

Τέλος, ο super Admin χρήστης του συστήματος έχει φυσικά και την δυνατότητα να διαγράφει ένα μουσείο απευθείας από το σύστημα. Η ενέργεια αυτή έχει υλοποιηθεί διότι αν κάποιος χρήστης ο οποίος ήταν και διαχειριστής μουσείου αποφασίσει να αποχωρήσει από την πλατφόρμα, υπάρχει περίπτωση να μην διαγράψει το μουσείο του. Σε αυτή την περίπτωση θα πρέπει να μπορεί να διαγραφτεί το μουσείο που έμεινε χωρίς διαχειριστή από το σύστημα. Επίσης, είναι επίσης πιθανό σενάριο τα στοιχεία που έχει παραχωρήσει για κάποιο μουσείο να είναι λανθασμένα και να προβαίνει στις απαραίτητες ενέργειες έτσι ώστε να τα ενημερώσει παρόλο που έχει μεσολαβήσει επικοινωνία για τα καταχωρημένα λάθη. Αυτό το κενό λοιπόν, θα καλύψει η λειτουργικότητα διαγραφής μουσείου από το σύστημα. Για να πραγματοποιηθεί η διαγραφή του μουσείου, πρέπει να πλοηγηθεί στη σελίδα προβολής των μουσείων, να βρει το μουσείο το οποίο θέλει να διαγράψει και να πατήσει του κουμπί διαγραφής. Έπειτα, το σύστημα θα διαγράψει το μουσείο από την λίστα και φυσικά από την βάση δεδομένων και όλο το σύστημα. Θα ακολουθήσει η εικόνα 62 που απεικονίζει τον τρόπο διαγραφής κάποιου μουσείου.



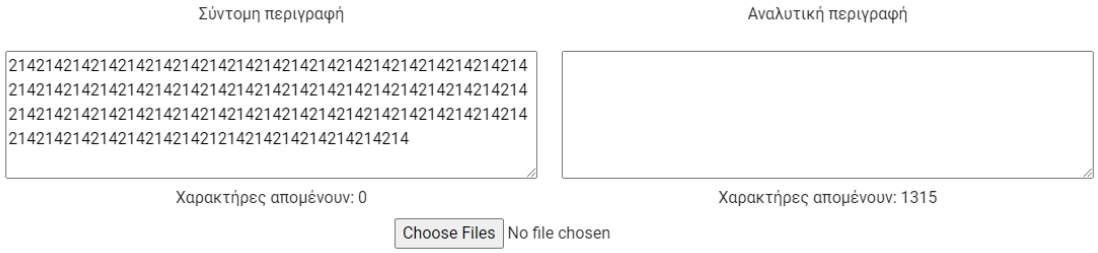
Εικόνα 62 - Τρόπος διαγραφής μουσείου από το σύστημα και τον super Admin

4.3.4 Σύνδεση Χρήστη Διαχειριστή Μουσείου στο Σύστημα

Όπως έχει γίνει ήδη αναφορά, υπάρχουν οι χρήστες οι οποίοι μπορούν να δημιουργήσουν κάποιο μουσείο. Την άδεια και την δυνατότητα αυτή τους την παρέχει ο super Admin του συστήματος με τον τρόπου που αναλύθηκε στο προηγούμενο κεφάλαιο. Αφού λοιπόν πραγματοποιήσουν σύνδεση στο σύστημα, από το προσωπικό τους προφίλ εμφανίζεται πλέον το κουμπί «Δημιουργία Μουσείου». Μόλις το κουμπί αυτό πατηθεί, το σύστημα κάνει ανακατεύθυνση στην σελίδα για την δημιουργία-προσθήκης νέου μουσείου στο σύστημα. Σε αυτή λοιπόν, θα πρέπει να γίνει συμπλήρωση των στοιχείων που απαιτούνται για την καταχώρηση του μουσείου:

- Όνομα ή Τίτλος μουσείου(Υποχρεωτικό)
- Έτος Ίδρυσης(Υποχρεωτικό)
- Διεύθυνση(Υποχρεωτικό)
- Πόλη(Υποχρεωτικό)
- Επίσημος Ιστοσελίδα
- Email (Υποχρεωτικό)
- Τηλέφωνο Επικοινωνίας
- Δεύτερο Τηλέφωνο Επικοινωνίας
- Διεύθυνση Καναλιού στο YouTube
- Διεύθυνση Facebook Λογαριασμού
- Διεύθυνση Twitter Λογαριασμού
- Διάρκεια και κόστος ξενάγησης
- Σύντομη περιγραφή
- Αναλυτική Περιγραφή
- Προσθήκη Φωτογραφιών

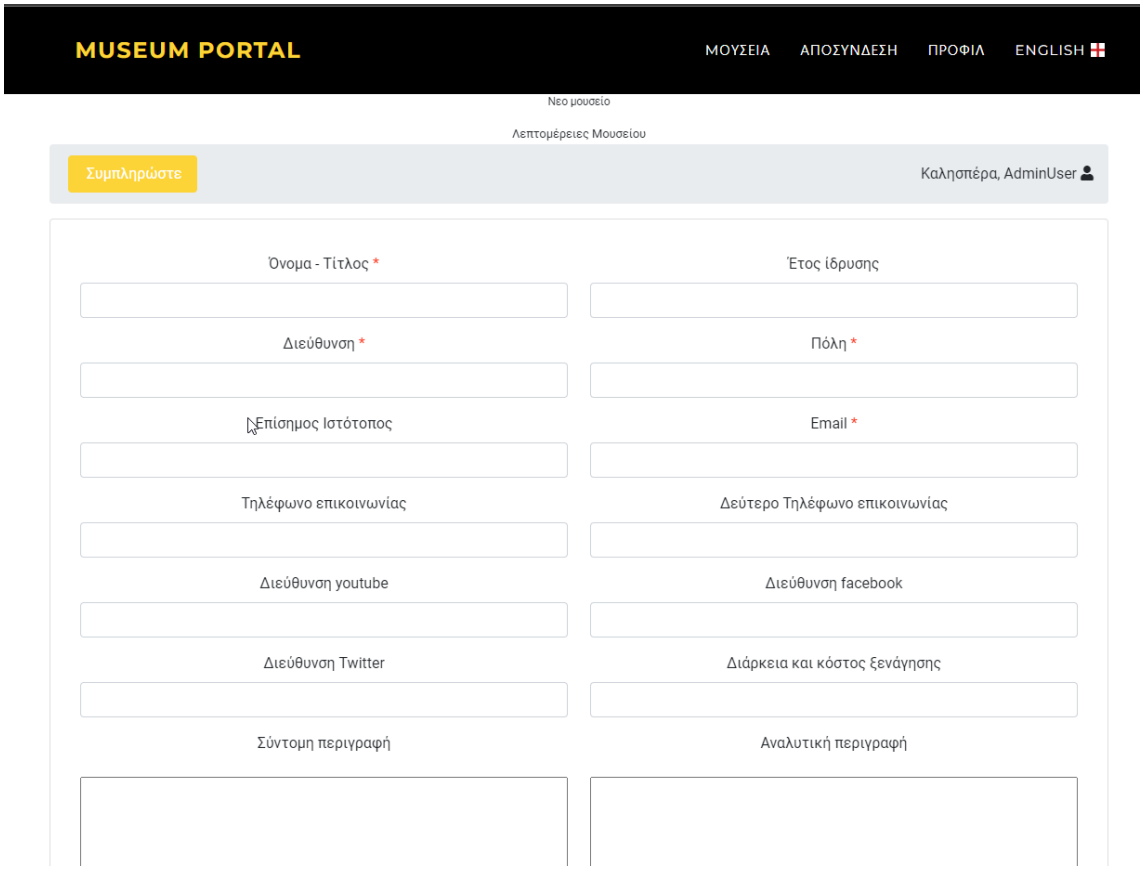
Αξίζει να σημειωθεί στο σημείο αυτό, τα πεδία σύντομη περιγραφή και αναλυτική περιγραφή έχουν όριο για το μέγιστο αριθμό χαρακτήρων που ο χρήστης μπορεί να πληκτρολογήσει. Αυτό γνωστοποιείται στον χρήστη με τρόπο φιλικό και εύκολα κατανοητό. Συγκεκριμένα, φαίνεται και στα δύο πεδία ο μέγιστος αριθμών χαρακτήρων που μπορείς να πληκτρολογήσεις. Επιπλέον, έχει υλοποιηθεί δυναμικά και καθώς ο χρήστης πληκτρολογεί οι χαρακτήρες που έχει πληκτρολογήσει αφαιρούνται από το αρχικό σύνολο ώστε να έχει συνεχώς ενημέρωση για το πόσους χαρακτήρες μπορεί να γράψει ακόμα. Τέλος, όταν ο χρήστης φτάσει τον μέγιστο επιτρεπόμενο αριθμό που μπορεί να εισάγει, δεν μπορεί πλέον να πληκτρολογήσει κάποιον άλλο χαρακτήρα. Ένα παράδειγμα της παραπάνω υλοποίησης φαίνεται και στην εικόνα 63.



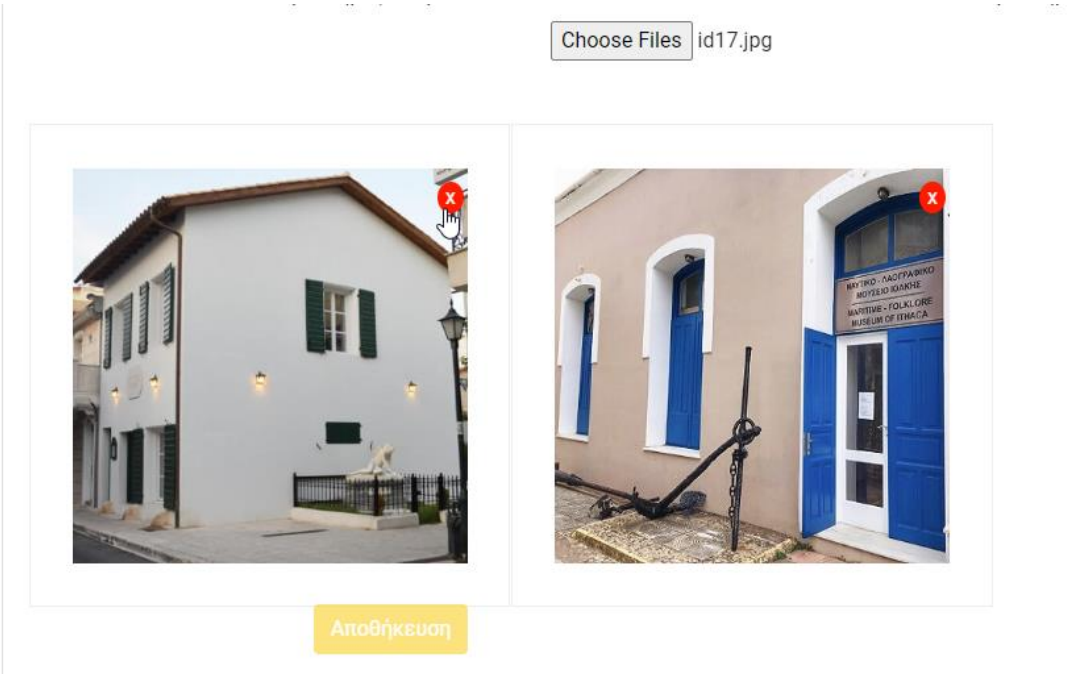
Εικόνα 63 - Παράδειγμα κειμένου στην φόρμα εισαγωγής νέου μουσείου

Συγκεκριμένα στην εικόνα 63, στο αριστερό μέρος βλέπουμε πως το κείμενο έχει φτάσει στον μέγιστο αριθμό επιτρεπόμενων χαρακτήρων και άρα ο χρήστης δεν μπορεί να πληκτρολογήσει κάτι άλλο. Αντίστοιχα, στο δεξιό μέρος, η φόρμα είναι ακόμα άδεια και αναγράφεται η μέγιστη χωρητικότητα χαρακτήρων. Φυσικά, όταν αρχίσει η εισαγωγή χαρακτήρων από τον χρήστη, οι αριθμός με τους χαρακτήρες που απομένουν θα μειώνονται μέχρι τελικά να φτάσουν μηδέν.

Τέλος, στην εισαγωγή φωτογραφιών, οι φωτογραφίες που επιλέγει ο χρήστης εμφανίζονται στο παράθυρο, και υπάρχει και η επιλογή να αφαιρέσει κάποια από αυτές σε περίπτωση που έκανε λάθος επιλογή. Η αφαίρεση γίνεται με την χρήση Χ που βρίσκεται στη δεξιά πάνω γωνία. Αναλυτικά φαίνεται και στην εικόνα 65.

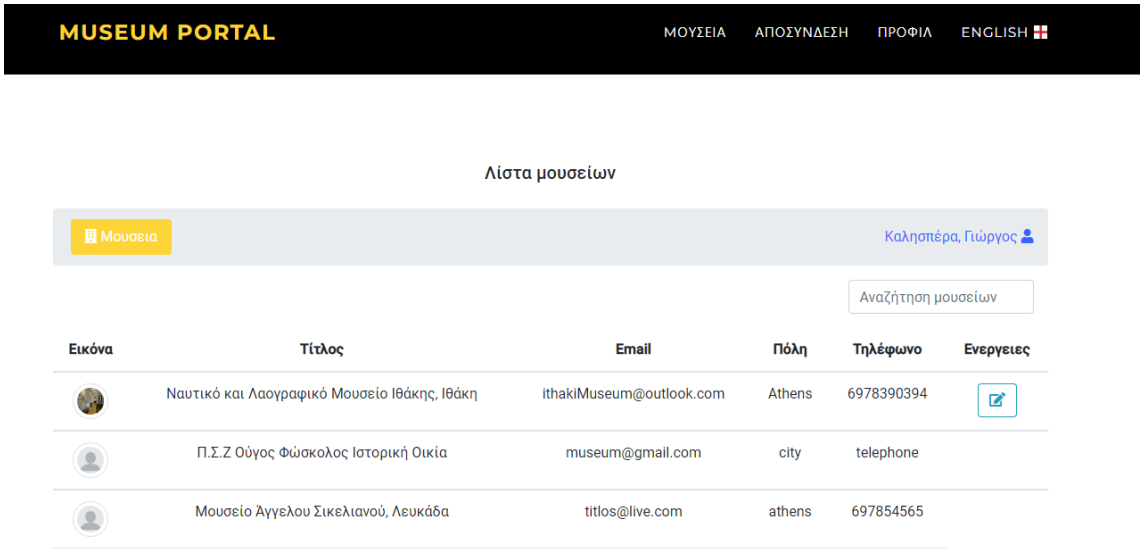


Εικόνα 64 - Παράδειγμα φόρμας καταχώρησης και δημιουργίας μουσείου.



Εικόνα 65 - Παράδειγμα προεπισκόπησης εικόνων κατά την δημιουργία μουσείου.

Στη συνέχει, καθώς έχουν εισαχθεί όλα τα στοιχεία για το μουσείο από τον χρήστη στο σύστημα και πατήσει να ολοκληρωθεί η εγγραφή, πλέον το μουσείο βρίσκεται ενεργό μέσα στο σύστημα και μπορεί να αναζητηθεί από τους χρήστες της πλατφόρμας. Στο σημείο αυτό όπως έχει ήδη αναφερθεί, ο διαχειριστής του μουσείου θα μπορεί να δει και ταυτόχρονα να διαχειριστεί τα αιτήματα για ξενάγηση που έχουν πραγματοποιηθεί από άλλους χρήστες και φυσικά να ενημερώσει αν χρειαστεί κάποιο από τα στοιχεία του μουσείου σε περίπτωση που αυτό χρειαστεί. Θα γίνει τώρα παρουσίαση πως μπορούν να πραγματοποιηθούν αυτές οι ενέργειες ξεκινώντας από την ενημέρωση στοιχείων του μουσείου. Ο διαχειριστής του μουσείου, θα πρέπει να πραγματοποιήσει πλοήγηση στην σελίδα αναζήτησης μουσείων. Εκεί, θα πρέπει να βρει το μουσείο στο οποίο είναι διαχειριστής. Μπορεί να χρησιμοποιήσει φυσικά την μπάρα αναζήτησης για να βρει γρήγορα το μουσείο που του ανήκει. Επίσης, θα μπορεί να το διακρίνει εύκολα καθώς θα είναι το μοναδικό μέσα στο πίνακα που θα έχει το σύμβολο για επεξεργασία όπως φαίνεται και στην εικόνα 66.

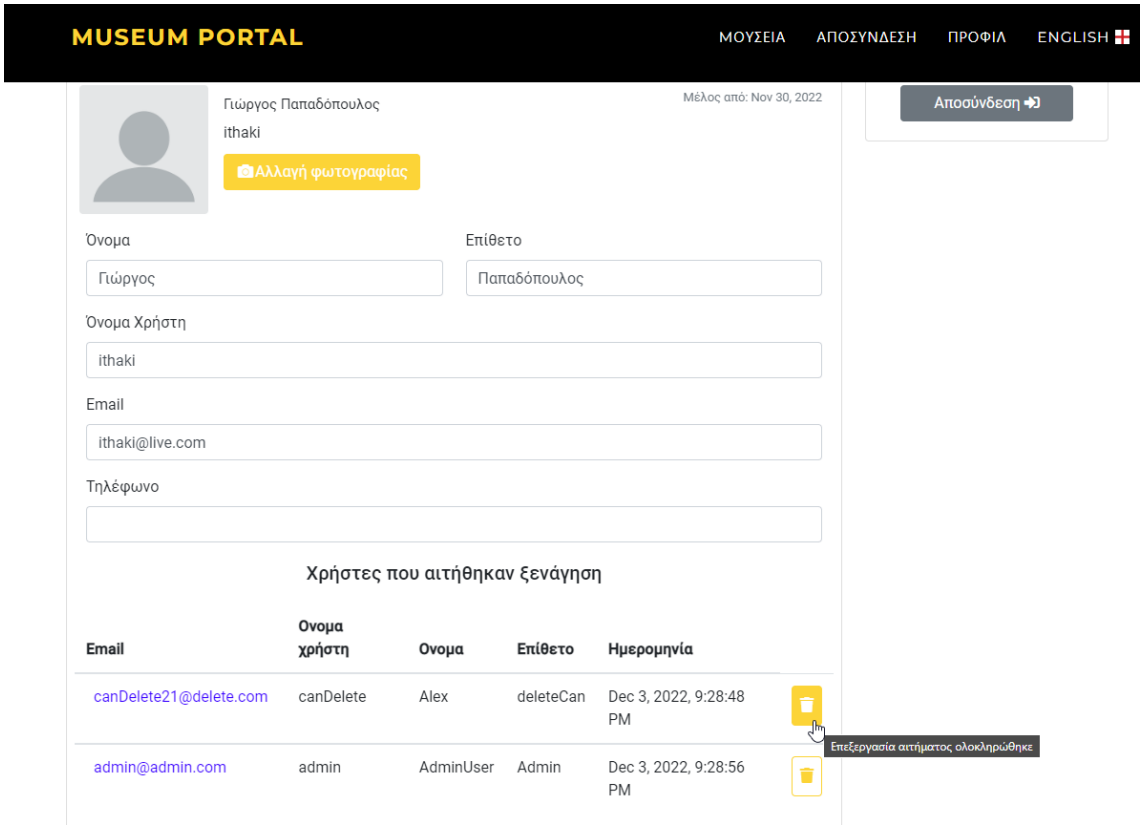


Εικόνα 66 - Μόνο το μουσείο που ανήκει στον συνδεδεμένο museum-admin έχει το εικονίδιο επεξεργασίας

Όπως απεικονίζεται, μόνο στην γραμμή που βρίσκεται το μουσείο το οποίο ανήκει στον συνδεδεμένο museum Admin έχει το σύμβολο της επεξεργασίας. Στη συνέχεια, όταν ο χρήστης επιλέξει και το πατήσει, θα ανοίξει ένα παράθυρο-modal με τα υπάρχοντα στοιχεία του μουσείου και είναι διαθέσιμα προς επεξεργασία. Πιο συγκεκριμένα θα προβληθούν και θα είναι διαθέσιμα για επεξεργασία τα:

- Όνομα Μουσείου
- Έτος ίδρυσης
- Διεύθυνση
- Πόλη
- Διάρκεια και κόστος ξενάγησης
- Διεύθυνση επίσημης ιστοσελίδας
- Email
- Τηλέφωνο
- Ιστοσελίδα για το προφίλ στο YouTube
- Ιστοσελίδα για το προφίλ Twitter
- Ιστοσελίδα για το προφίλ Facebook

Μόλις λοιπόν, ολοκληρωθούν από την χρήστη οι αλλαγές στις οποίες έχει αποφασίσει πως θέλει να πραγματοποιήσει και πατήσει αποθήκευση, τα νέα στοιχεία θα έχουν καταχωρηθεί και αποθηκευτεί στη βάση δεδομένων και θα είναι διαθέσιμα σε όλους τους χρήστες της πλατφόρμας. Στη συνέχεια, θα παρουσιαστεί ο τρόπος παρουσίασης και διαχείρισης των αιτημάτων που έχουν γίνει για την ξενάγηση στο μουσείο. Ο χρήστης θα πρέπει να επισκεφτεί την σελίδα που προβάλλεται το προσωπικό του προφίλ. Στο σημείο αυτό, θα υπάρχει ο πίνακας με τυχόν αιτήματα που έχουν πραγματοποιηθεί από άλλους χρήστες για επίσκεψη και ξενάγηση στο μουσείο.



Εικόνα 67 - Προβολή αιτημάτων ξενάγησης στο προφίλ του διαχειριστή μουσείου

Όπως απεικονίζεται στην εικόνα 67, ο χρήστης βλέπει σε λίστα όλα τα αιτήματα ξενάγησης που έχουν πραγματοποιηθεί. Πιο συγκεκριμένα στον διαχειριστή απεικονίζονται τα συγκεκριμένα δεδομένα του χρήστη που έκανε το αίτημα ξενάγησης:

- Email
- Όνομα Χρήστη
- Όνομα
- Επίθετο
- Ημερομηνία αιτήματος

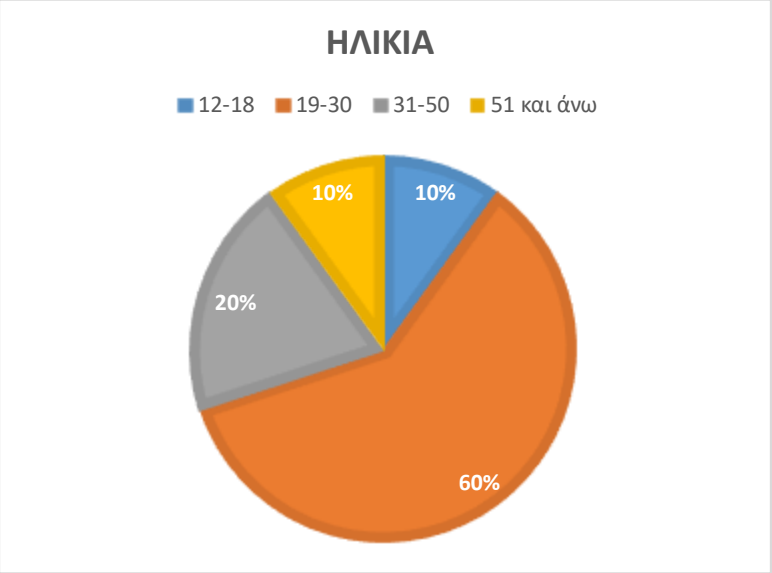
Όπως είναι πλέον κατανοητό, ο διαχειριστής έχει όλα τα στοιχεία του χρήστη που έκανε το αίτημα. Αυτό του δίνει την δυνατότητα να επικοινωνήσει μαζί του με όποιον τρόπο επιθυμεί όπως για παράδειγμα με αποστολή μηνύματος ηλεκτρονικού ταχυδρομείου ή να τον καλέσει στο τηλέφωνο. Επίσης, μέσα στον πίνακα υπάρχει και το κουμπί για την διαγραφή του αιτήματος. Αν για παράδειγμα προηγηθεί επικοινωνία με τον χρήστη που έκανε το αίτημα, και η έκβαση του αιτήματος αποσαφηνιστεί ο διαχειριστής πλέον μπορεί να διαγράψει το αίτημα για την ξενάγηση. Αυτή η δυνατότητα βέβαια, έχει το πλεονέκτημα πως πλέον είναι διαχειρίσιμος ο όγκος αιτημάτων, καθώς δεν θα συγκεντρώνονται με το βάθος χρόνου παλιά και ήδη εξυπηρετημένα αιτήματα. Κάτι τέτοιο γίνεται εύκολα κατανοητό πως δεν θα ήταν εξυπηρετικό και βολικό για τον χρήστη και θα δυσκόλευε πολύ το έργο του στην διαχείριση. Όπως επίσης φαίνεται στην εικόνα 67, ο χρήστης εφόσον έχει ήδη μουσείο δεν υπάρχει η δυνατότητα και το κουμπί πλέον για προσθήκη νέου μουσείου. Τέλος, όπως σε κάθε συνδεδεμένο χρήστη έτσι και στον διαχειριστή μουσείου, υπάρχει λίστα με αγαπημένα μουσεία που θα προβάλλονται τυχόν μουσεία τα οποία έχει προσθέσει στην λίστα με την γνωστή διαδικασία προσθήκης μουσείο στα αγαπημένα.

5 Αξιολόγηση, Συμπεράσματα και Πιθανές Επεκτάσεις

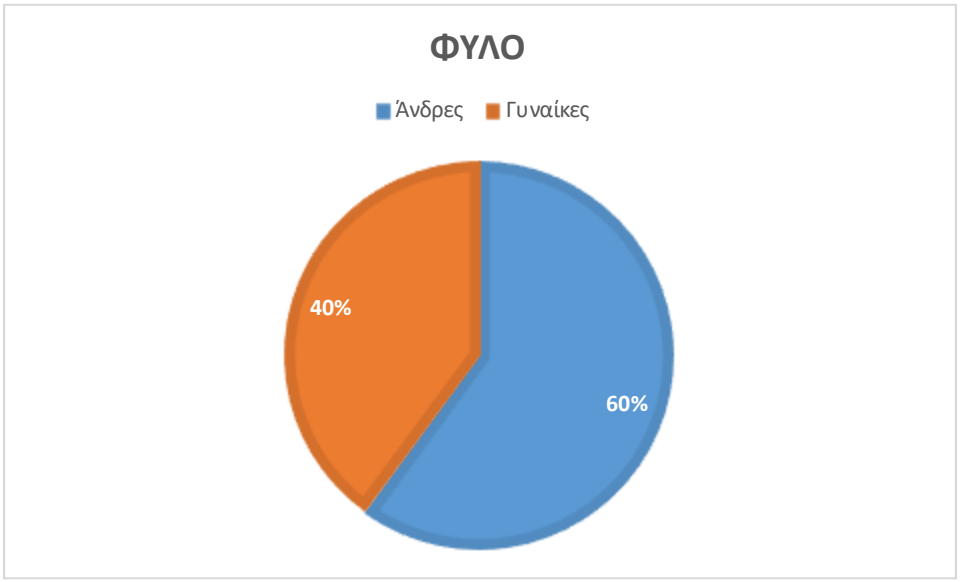
5.1 Αξιολόγηση

Προκειμένου να πραγματοποιηθεί η αξιολόγηση της πλατφόρμας που υλοποιήθηκε στα πλαίσια της παρούσας διατριβής, δόθηκε σε 10 χρήστες το ερωτηματολόγιο, σενάρια για χρήση της πλατφόρμας και ερωτήσεις αξιολόγησης τα οποία είναι διαθέσιμα στο παράρτημα Α. Για τον σκοπό αυτό επιλέχθηκαν χρήστες από όλες τις ηλιακές ομάδες με έμφαση στις ηλικίες 19-50 οι οποίες είτε επιλέγουν να επισκεφτούν την πλατφόρμα και να δουν πληροφορίες για τα μουσεία που ενδιαφέρουν εκείνους είτε τα παιδιά τους.

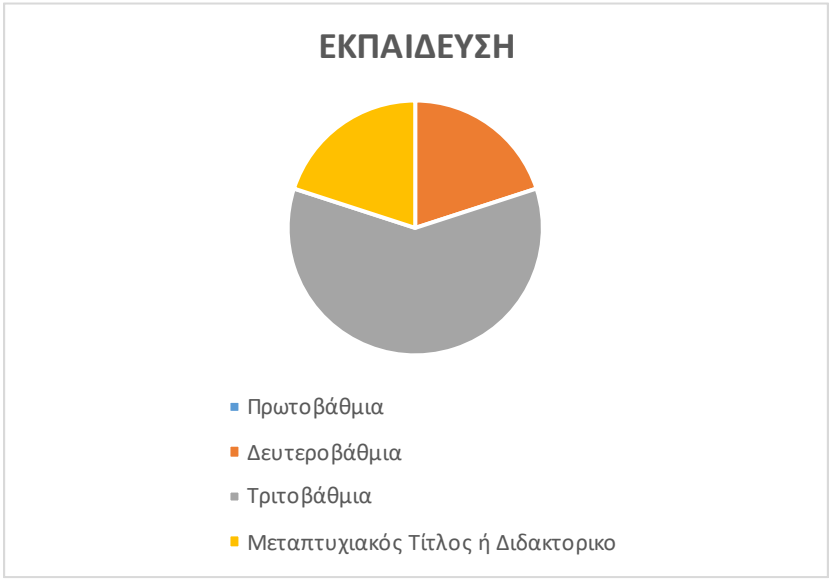
Επίσης, επιλέχθηκαν χρήστες με διαφορετικό μορφωτικό επίπεδο αλλά και διαφορετική συχνότητα χρήσης του διαδικτύου ώστε να υπάρχει μια όσο το δυνατόν πιο όμορφη κατανομή του υπόβαθρου του συνόλου των χρηστών, η οποία να ανταποκρίνεται στο πραγματικό προφίλ του μέσου επισκέπτη της πλατφόρμας. Πιο συγκεκριμένα, στα παρακάτω διαγράμματα αποτυπώνονται τα στατιστικά για την ηλικία, για το φύλο, το επίπεδο εκπαίδευσης και πόσο εξοικειωμένοι είναι με την χρήση διαδικτύου οι χρήστες οι οποίοι έτρεξαν τα σενάρια που τους δόθηκαν στα πλαίσια της αξιολόγησης.



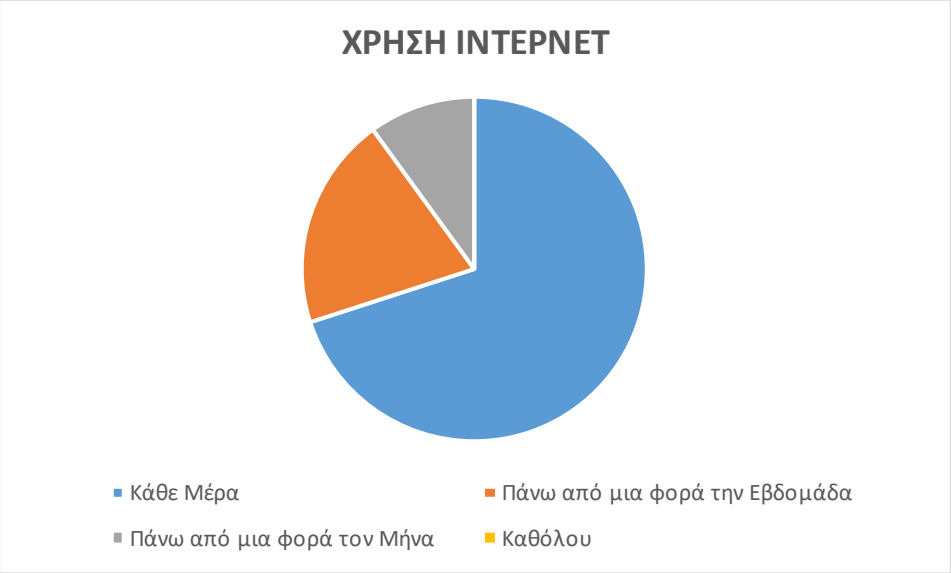
Εικόνα 68 - Διάγραμμα απεικόνισης χρηστών βάσει ηλικίας



Εικόνα 69 - Διάγραμμα απεικόνισης βάσει φύλου



Εικόνα 70 - Διάγραμμα απεικόνισης βάσει εκπαίδευσης



Εικόνα 71 - Διάγραμμα απεικόνισης βάσει χρήσης internet

Από την αξιολόγηση και την επίδοση των χρηστών στα σενάρια που τους δόθηκαν να εκτελέσουν στην πλατφόρμα, παρατηρήθηκαν κάποιες αστοχίες στην εμφάνιση πληροφοριών μέσα στο σύστημα οι οποίες προκάλεσαν δυσκολία και καθυστέρηση στους χρήστες κατά την εκτέλεση των σεναρίων. Αρχικά, στην σελίδα πληροφοριών ενός μουσείου, παρατηρήθηκε δυσκολία στην εύρεση του email επικοινωνίας. Αυτό συνέβη διότι, υπήρχε το εικονίδιο του email και μόλις ο χρήστης πάταγε πάνω του άνοιγε το προ εγκατεστημένο πρόγραμμα email του υπολογιστή με συμπληρωμένη την διεύθυνση αποστολή του μουσείου. Όμως, φάνηκε πως οι χρήστες περίμεναν συγκεκριμένα να δουν την εμφάνιση της διεύθυνσης email, και οι περισσότεροι πίστευαν πως αφού δεν την δείχνει, το μουσείο δεν είχε. Από την συγκεκριμένη παρατήρηση, έγινε ενημέρωση του συστήματος και πλέον φαίνεται και η διεύθυνση email. Επίσης, κάποιοι χρήστες κατά την διάρκεια εκτέλεσης σεναρίου που απαιτούσε σύνδεση και εγγραφή στο σύστημα, αναζήτησαν δυνατότητα σύνδεσης μέσω κάποιου άλλου τρόπου πιστοποίησης όπως λογαριασμό Google. Η παρατήρηση – αίτημα τους σημειώθηκε και προστέθηκε στις μελλοντικές επεκτάσεις, καθώς η πιστοποίηση και σύνδεση με SSO είναι εκτός του πεδίου της παρούσας διατριβής.

Οι χρήστες σχολίασαν θετικά την δυνατότητα που παρέχεται από την πλατφόρμα να υπάρχουν συγκεντρωμένα σε μία πλατφόρμα πολλά μουσεία, τον εύκολο τρόπο αναζήτησής τους από την μπάρα αναζήτησης, την δυνατότητα εγγραφής σύνδεσης και προσθήκης μουσείων σε λίστα αγαπημένων καθώς και πως μπορούσαν να πραγματοποιήσουν αίτημα για ξενάγηση σε μουσείο της επιλογής τους.

5.2 Συμπεράσματα

Η παρούσα μεταπτυχιακή διατριβή είχε ως στόχο την ανάλυση και την υλοποίηση ενός συστήματος για την αποθήκευση, προβολή και διαχείριση μουσείων και χρηστών έτσι ώστε να προστεθεί μια επιπλέον λειτουργικότητα και αλληλεπίδραση ανάμεσα στους εγγεγραμμένους χρήστες τους συστήματος με τα μουσεία. Ανάμεσα στους χρήστες υπήρχε διαχωρισμός ως προς τα δικαιώματα που είναι ο απλός χρήστης, ο διαχειριστής μουσείου και ο διαχειριστής όλης της πλατφόρμας.

Η συγκεκριμένη αρχιτεκτονική στον διαχωρισμό των χρηστών στην παρούσα μεταπτυχιακή διατριβή επιλέχθηκε με την προοπτική ο φόρτος εργασίας για την δημιουργία, συντήρηση και επέκταση πληροφοριών για τα μουσεία να έρθει σε ισορροπία με τον αριθμό των χρηστών και να κατανεμηθεί σωστά. Πιο συγκεκριμένα, όπως αναφέρθηκε ήδη για το κάθε ένα μουσείο υπάρχει

ένας διαχειριστής και ταυτόχρονα δημιουργός του. Αυτό έγινε με σκοπό, να απλοποιηθεί για τους χρήστες η εργασία πάνω στο σύστημα καθώς θα έχουν να διαχειρίζονται μόνο ένα μουσείο. Στην αντίθετη περίπτωση, αν υπήρχε ένας admin όπως γίνεται σε άλλα συστήματα πρέπει ένας χρήστης να διαχειρίζεται πολλά μουσεία με ενδεχόμενο αποτέλεσμα, καθυστερημένη ανταπόκριση. Είναι λοιπόν, μία επιχειρηματική λογική για την κατανομή του φόρτου εργασίας, χωρίς υπερβολές και φυσικά με ασφάλεια καθώς κάθε χρήστης που έχει υπό την ευθύνη του κάποιο μουσείο έχει πρώτα εγκριθεί από τον super admin του συστήματος.

Επίσης, οι τεχνολογίες και οι τεχνικές που επιλέχθηκαν για την υλοποίηση όπως αναφέρθηκε στα προηγούμενα κεφάλαια έγιναν με γνώμονα την επεκτασιμότητα. Έτσι ώστε, ακόμα και αν αυξηθεί αρκετά το πλήθος μουσείων και χρηστών να μην επηρεάζεται η αποδοτικότητά του συστήματος και να συνεχίσει να λειτουργεί άρτια.

Σκοπός λοιπόν, της παρούσας εργασίας ήταν να παρουσιάσει ένα διαφορετικό και νέο σύστημα για την εύρεση και την αναζήτηση μουσείων και να δοκιμαστεί η κατανομή της διαχείρισης μουσείων από πολλαπλούς χρήστες. Καθώς τα μουσεία είναι μέρος του πολιτισμού και της ιστορίας της κάθε χώρας, η πλατφόρμα αυτή στοχεύει τόσο στην διευκόλυνση του απλού χρήστη στην αναζήτηση και εύρεση μουσείων, όσο και στην ευκολία διαχείρισης την πλατφόρμας από τους διαχειριστές των μουσείων προβάλλοντας τον νέο αυτό επιχειρησιακό μοντέλο.

5.3 Πιθανές επεκτάσεις

Η υλοποίηση της παρούσας μεταπτυχιακής διπλωματικής εργασίας αποτελεί μία εφαρμογή για την παρουσίαση μουσείων. Όπως έχει ήδη αναφερθεί, είναι υλοποιημένη σε δύο γλώσσες. Καθώς το θέμα των μουσείων είναι πολιτισμικού ενδιαφέροντος, και αφορά πολίτες όλων των κρατών που επισκέπτονται την Ελλάδα, θα είναι μία πολύ καλή επέκταση στο σύστημα να προστεθούν και άλλες γλώσσες ώστε να εξυπηρετεί μεγαλύτερο αριθμό επισκεπτών.

Επιπλέον, για μεγαλύτερη ευκολία των χρηστών του συστήματος και καλύτερη εμπειρία χρήσης, μία ακόμα πιθανή επέκταση είναι η προσθήκη φίλτρων τόσο για την αναζήτηση μουσείου ανά περιοχή όσο και ανά κατηγορία ή τύπο εκθεμάτων. Ακόμα, καθώς ο απώτερος στόχος για την πλατφόρμα είναι να μπορεί να χρησιμοποιηθεί από όλους, είναι απαραίτητη η επέκτασή της ως προς τις λειτουργίες για άτομα με προβλήματα όρασης. Τέλος, στην περίπτωση χρήσης στην οποία ο χρήστης κάνει αίτηση στο μουσείο για να κλείσει μία ξενάγηση, ο διαχειριστής του μουσείου στην παρούσα υλοποίηση έχει την δυνατότητα να επικοινωνήσει μαζί του μέσω μηνύματος ηλεκτρονικού ταχυδρομείου. Σαν μελλοντική επέκταση και βελτιστοποίηση θα μπορούσε να προστεθεί η δυνατότητα επικοινωνίας των χρηστών μέσω της ίδιας της πλατφόρμας.

Ακόμα, πιθανή επέκταση των λειτουργειών της πλατφόρμας θα ήταν να γίνεται εγγραφή και σύνδεση στην πλατφόρμα μέσω κάποιου λογαριασμού που οι χρήστες έχουν ήδη σε άλλες πλατφόρμες όπως Google, Facebook. Τέλος, για τους εγγεγραμμένους χρήστες θα μπορούσαν να δημιουργηθούν και να αποστέλνονται νέα-newsletters μηνύματα ηλεκτρονικού ταχυδρομείου για τα μουσεία τα οποία έχουν προσθέσει στην λίστα αγαπημένων τους.

Βιβλιογραφία

'Angular + Spring Boot + MySQL CRUD Example'. n.d. Accessed 4 December 2022. https://www.javaguides.net/2020/07/spring-boot-angular-10-crud-example-tutorial.html.

'Angular 7 + Spring Boot JWT Authentication | JavaInUse'. n.d. Accessed 4 December 2022. https://www.javainuse.com/spring/ang7-jwt.

'Angular Data Flow Architecture - Web Development with MongoDB and Node - Third Edition [Book]'. n.d. Accessed 4 December 2022. https://www.oreilly.com/library/view/web-development-with/9781788395083/eccdb4b9-4d4b-4bc2-bf3a-56cfe45d6360.xhtml.

auth0.com. n.d. 'JWT.IO - JSON Web Tokens Introduction'. Accessed 23 November 2022. http://jwt.io/.

contributors, Mark Otto, Jacob Thornton, and Bootstrap. n.d. 'Bootstrap'. Accessed 26 November 2022. https://getbootstrap.com/.

'HashSet (Java Platform SE 7)'. n.d. Accessed 21 November 2022. https://docs.oracle.com/javase/7/docs/api/java/util/HashSet.html.

'Initiatives_of_museums_in_times_of_corona_4_20.Pdf'. n.d. Accessed 3 December 2022. https://www.ne-mo.org/fileadmin/Dateien/public/NEMO_documents/Initiatives_of_museums_in_times_of_corona_4_20.pdf.

'JavaScript Observer (Publish/Subscribe) Pattern'. n.d. Accessed 4 December 2022. https://bumbu.me/javascript-observer-publish-subscribe-pattern/.

'JpaRepository (Spring Data JPA Parent 3.0.0 API)'. n.d. Accessed 23 November 2022. https://docs.spring.io/spring-data/jpa/docs/current/api/org/springframework/data/jpa/repository/JpaRepository.html.

'JWT Authentication with Spring Boot Resource Server | by Imesha Sudasingha | The Startup'. n.d. Accessed 4 December 2022. https://medium.com/swlh/stateless-jwt-authentication-with-spring-boot-a-better-approach-1f5dbae6c30f.

'JWT Token Generation for HMAC Using SHA Algorithm'. n.d. Accessed 23 November 2022. https://docs.mashery.com/connectorsguide/GUID-A516D8BB-B5CA-45D0-8C3F-391D8C24F1AE.html.

kexugit. n.d. 'Pros and Cons of Data Transfer Objects'. Accessed 4 December 2022. https://learn.microsoft.com/en-us/archive/msdn-magazine/2009/august/pros-and-cons-of-data-transfer-objects.

Lane, Tom. 2000. 'Transaction Processing in PostgreSQL', 22.

Mamun, Iftekher. 2019. 'A Simple API Summary'. *Medium* (blog). 19 May 2019. https://medium.com/@imamun/a-simple-api-summary-e84af524aa11.

'Maven – Introduction'. n.d. Accessed 28 November 2022. https://maven.apache.org/what-is-maven.html.

'NEMO_Corona_Survey_Results_6_4_20.Pdf'. n.d. Accessed 3 December 2022. https://www.ne-mo.org/fileadmin/Dateien/public/NEMO_documents/NEMO_Corona_Survey_Results_6_4_20.pdf.

Nym. 2022a. 'Login System Use Case Diagram | UML'. Itsourcecode.Com. 6 June 2022. https://itsourcecode.com/uml/login-use-case-diagram-uml/.

———. 2022b. 'UML Diagram for Login Page'. Itsourcecode.Com. 9 June 2022. https://itsourcecode.com/uml/uml-diagram-for-login-page/.

Paul, Javin. 2017. 'Difference between @Component, @Service, @Controller, and @Repository in Spring - Java Code Geeks - 2022'. *Java Code Geeks* (blog). 30 November 2017. https://www.javacodegeeks.com/2017/11/difference-component-service-controller-repository-spring.html.

'Project Lombok'. n.d. Accessed 20 November 2022. https://projectlombok.org/.

Rao, Rakshith, and Sr Swamy. 2020. 'Review on Spring Boot and Spring Webflux for Reactive Web Development' 7 (May).

'Relational Data Modeling - Association Table (Bridge, Cross)'. 2020. Datacadamia - Data and Co. 21 March 2020. https://datacadamia.com/data/type/relation/modeling/association.

‘Spring Boot Architecture and Its Workflow’. n.d. Tech with Maddy. Accessed 23 November 2022. https://techwithmaddy.com/spring-boot-architecture.
‘Spring Method Security with PreAuthorize’. n.d. Okta Developer. Accessed 24 November 2022. https://developer.okta.com/blog/2019/06/20/spring-preauthorize.
‘[Spring Security] - Spring Boot Security JWT Authentication’. 2021. *AI Design - Thiết Kế Web Theo Yêu Cầu Tại Hồ Chí Minh* (blog). 30 November 2021. https://aithietke.com/spring-security-spring-boot-security-jwt-authentication/.
Stavast, Pieter. 2018. ‘Spring Boot Tutorial’. *Anchormen | Data Activators* (blog). 2 July 2018. https://anchormen.nl/blog/big-data-services/spring-boot-tutorial/.
Stonebraker, Michael, and Lawrence A Rowe. n.d. ‘THE DESIGN OF POSTGRES’, 28.
Team, Angular Components. n.d. ‘Angular Material’. Angular Material. Accessed 20 November 2022. https://material.angular.io/.
TekTutorialsHub. 2018. ‘Introduction to Angular Modules or NgModule’. *TekTutorialsHub* (blog). 23 November 2018. https://www.tektutorialshub.com/angular/angular-modules/.
Thai, Nguyen Nam. 2018. ‘What Is a Spring Bean? | Baeldung’. 28 September 2018. https://www.baeldung.com/spring-bean.
‘The GoF Design Patterns Memory - Learning Object-Oriented Design & Programming’. n.d. W3sDesign. Accessed 28 November 2022. http://www.w3sdesign.com.
‘Web Based Information Systems’. n.d. Accessed 4 December 2022. https://www.ukessays.com/essays/information-systems/web-based-information-systems.php.
‘What Is RESTful API? - RESTful API Beginner’s Guide - AWS’. n.d. Amazon Web Services, Inc. Accessed 20 November 2022. https://aws.amazon.com/what-is/restful-api/.
‘Why Are ECommerce Wish Lists & Favorites Important?’ 2020. Logicblock. 1 December 2020. https://www.logicblock.com/the-importance-of-wish-lists-and-favorites/.

ΠΑΡΑΡΤΗΜΑ Α - ΕΡΩΤΗΜΑΤΟΛΟΓΙΟ

Πρόλογος

Σας προσκαλούμε να συμπληρώσετε το παρόν Ερωτηματολόγιο Αξιολόγησης που αφορά την αξιολόγηση ιστοσελίδας Museum Portal, η οποία κατασκευάστηκε στα πλαίσια διπλωματικής εργασίας για το μεταπτυχιακό «Προηγμένα Συστήματα Ανάπτυξης Λογισμικού και Τεχνητής Νοημοσύνης» του τμήματος Πληροφορικής του Πανεπιστημίου Πειραιώς.

Η συμβολή σας είναι πολύ σημαντική ώστε να μπορέσουμε να κατανοήσουμε τις ανάγκες των χρηστών σε βάθος για την περαιτέρω βελτιστοποίηση της σελίδας και για το λόγο αυτό σας καλούμε να συμμετάσχετε.

Το ερωτηματολόγιο αυτό είναι ανώνυμο. Κατά τη συμμετοχή σας στην έρευνα δεν καταγράφεται κανένα στοιχείο που να σας ταυτοποιεί:

Θέλετε να συμπληρώσετε το ερωτηματολόγιο;

☐ Ναι ☐ Όχι

Δεν συμπληρώνω το ερωτηματολόγιο γιατί:

(Μπορείτε να επιλέξετε παραπάνω από μια απαντήσεις)

☐ Δεν με ενδιαφέρει η αξιολόγηση

☐ Δεν έχω εμπιστοσύνη ότι εξασφαλίζεται η ανωνυμία μου

☐ Άλλο: ……...

Ενότητα Α | Γενικά στοιχεία

Α1. Φύλο:

☐ Άνδρας ☐ Γυναίκα

Α2. Ηλικία:

……

Α3. Επίπεδο εκπαίδευσης:

☐ Πρωτοβάθμια Εκπαίδευση

☐ Δευτεροβάθμια Εκπαίδευση

☐ Ανώτατη Εκπαίδευση

☐ Μεταπτυχιακός τίτλος ή Διδακτορικό

Α4. Πόσο συχνά χρησιμοποιείτε το διαδίκτυο:

☐ Κάθε Μέρα

☐ Πάνω από μια φορά την Εβδομάδα

☐ Πάνω από μια φορά τον Μήνα

☐ Καθόλου

Α5. Υπάρχει διαθέσιμη σύνδεση Ιντερνέτ στο σπίτι:

☐ Ναι ☐ Όχι

Ενότητα Β | Οδηγίες για την πλοήγηση στην ιστοσελίδα

Σενάριο 1

Υποθέστε πως πρόκειται να επισκεφθείτε την Ιθάκη και είστε σε αναζήτηση μουσείων και αξιοθέατων για την επίσκεψή σας στο νησί. Αποφασίζετε να χρησιμοποιήσετε την σελίδα MuseumLog για την έρευνα σας και βρίσκεστε στην αρχική σελίδα της εφαρμογής.

Βήμα 1

Πλοηγηθείτε στην σελίδα με τα μουσεία.

Βήμα 2

Αναζητήστε το μουσείο Ναυτικό και Λαογραφικό Μουσείο Ιθάκης στη λίστα των μουσείων.

Βήμα 3

Ανοίξτε την σελίδα του παραπάνω μουσείου με τις γενικές πληροφορίες για το μουσείο, όπως για παράδειγμα το έτος που ιδρύθηκε, περιγραφή του μουσείου και το email του μουσείο.

Μόλις αναγνωρίσετε τα παραπάνω συνεχίστε στο επόμενο βήμα στο σενάριο.

Βήμα 4

Βρείτε τη διεύθυνση του μουσείου, την τιμή και την διάρκεια της ξενάγησης στο μουσείου από προσωπικό του μουσείου και τέλος το τηλέφωνο του μουσείου.

Μόλις αναγνωρίσετε τα παραπάνω συνεχίστε στο επόμενο βήμα στο σενάριο.

Βήμα 5

Βρείτε την λειτουργία να κάνετε αίτημα ξενάγησης και επιλέξτε το.

Σενάριο 2

Πλέον έχετε επιλέξει το αίτημα ξενάγησης. Όμως, για να έχετε την δυνατότητα αυτή χρειάζεται να είστε εγγεγραμμένος χρήστης. Με την επιλογή αιτήματος ξενάγησης έχετε μεταφερθεί στη σελίδα σύνδεσης χρήστη.

Βήμα 1

Επιλέξτε να κάνετε εγγραφή στο σύστημα.

Βήμα 2

Συμπληρώσετε τα στοιχεία χρήστη που ζητούνται και ολοκληρώστε την εγγραφή.

Βήμα 3

Πραγματοποιείστε σύνδεση λογαριασμού με τα στοιχεία που δώσατε στο προηγούμενο βήμα.

Βήμα 4

Ολοκληρώστε το αίτημα ξενάγησης.

Σενάριο 3

Είστε πλέον συνδεδεμένος χρήστης της πλατφόρμας, έχετε ολοκληρώσει το αίτημα ξενάγησης. Επιθυμείτε τώρα να δείτε να στοιχεία του λογαριασμού σας για αλλαγή του τηλεφώνου σας καθώς και προσθήκη φωτογραφίας για την ολοκλήρωση του προφίλ.

Βήμα 1

Μεταβείτε στη σελίδα του προφίλ σας.

Βήμα 2

Ενημερώστε το τηλέφωνο επικοινωνίας.

Βήμα 3

Επιλέξτε προσθήκη φωτογραφίας και διαλέξτε την φωτογραφία που επιθυμείτε.

Βήμα 4

Παρατηρείστε τις πληροφορίες και τις δυνατότητες που έχετε στην σελίδα του προφίλ σας ως εγγεγραμμένος χρήστης.

Μόλις αναγνωρίσετε τα παραπάνω συνεχίστε στο επόμενο σενάριο.

Σενάριο 4

Στη σελίδα του προφίλ σας, παρατηρήσατε ότι μπορείτε να δείτε την λίστα με τα αγαπημένα μουσεία, εάν έχετε προσθέσει κάποια. Επιθυμείς να προσθέσεις μουσεία της Ιθάκης στη λίστα.

Βήμα 1

Βρείτε και επιλέξτε από το μενού να μεταφερθείτε στην λίστα με τα διαθέσιμα μουσεία.

Βήμα 2

Ψάξτε στην αναζήτηση το μουσείο Ναυτικό και Λαογραφικό Μουσείο Ιθάκης που βλέπατε και σε προηγούμενο σενάριο και επιλέξτε να δείτε τις πληροφορίες του.

Βήμα 3

Βρείτε την δυνατότητα προσθήκης του μουσείου στη λίστα αγαπημένων και προσθέστε το.

Βήμα 4

Επιστρέψτε στην λίστα των διαθέσιμων μουσείων και αναζητήστε για περισσότερα μουσεία της Ιθάκης με το όνομα του νησιού.

Βήμα 5

Επιλέξτε να επισκεφτείτε ένα από τα μουσεία της Ιθάκης.

Βήμα 6

Πραγματοποιήστε αποσύνδεση από την εφαρμογή.

Σενάριο 5

Είστε ο διαχειριστής του μουσείου όνομα μουσείου. Διαθέτετε ήδη λογαριασμό στην πλατφόρμα με δικαιώματα museum-admin και μπορείτε να πραγματοποιήσετε καταχώρηση ενός νέου μουσείου στη πλατφόρμα.

Βήμα 1

Συνδεθείτε στην πλατφόρμα με τον λογαριασμό museum-admin.

username: canCreateMuseum

password: canCreateMuseum

Βήμα 2

Βρείτε την δυνατότητα προσθήκης μουσείου και επιλέξτε τη.

Βήμα 3

Συμπληρώστε τα απαραίτητα στοιχεία που ζητούνται για το μουσείο και συνεχίστε με αποθήκευση του νέου μουσείου.

Βήμα 4

Αφού ολοκληρώσετε την αποθήκευση-καταχώρηση του νέου μουσείου παρατηρείστε την λίστα με όλα τα διαθέσιμα μουσεία και ότι έχετε την δυνατότητα τροποποίησης στοιχείων μόνο για το μουσείο στο οποίο είστε διαχειριστής.

Μόλις αναγνωρίσετε τα παραπάνω συνεχίστε στο επόμενο βήμα στο σενάριο.

Βήμα 5
Πραγματοποιείστε μία αλλαγή σε ένα από τα στοιχεία του μουσείου όπως το τηλέφωνο.

Σενάριο 6

Ενώ είστε συνδεδεμένος στην πλατφόρμα με το προφίλ του διαχειριστή μουσείου, επιθυμείτε να δείτε να αιτήματα για ξενάγηση που έχουν γίνει από εγγεγραμμένους χρήστες της πλατφόρμας για το μουσείο στο οποίο είστε διαχειριστής.

Βήμα 1
Μεταβείτε στην σελίδα του προφίλ σας.
Βήμα 2
Αναζητείστε τα διαθέσιμα αιτήματα ξενάγησης.
Βήμα 3
Επιλέξτε ένα από τα αιτήματα προκειμένου να επικοινωνήσετε με τον χρήστη που έκανε το αίτημα για να το διεκπεραιώσετε.
Βήμα 4
Επιλέξτε ένα από τους τρόπους επικοινωνίας με τον χρήστη.
Βήμα 5
Ολοκληρώστε την ενέργεια και πραγματοποιήστε αποσύνδεση από την πλατφόρμα.

Σενάριο 7

Είστε ο super-admin της εφαρμογής και μετά από αίτημα του χρήστη Μαρία Παπαδοπούλου επιθυμείτε να απενεργοποιήσετε το προφίλ του χρήστη.

Βήμα 1
Κάνε σύνδεση στην πλατφόρμα με τα στοιχεία σύνδεσης:
Username: admin
Password: admin
Βήμα 2
Βρίσκεστε στη σελίδα με την λίστα των εγγεγραμμένων χρηστών. Αναζητήστε τον χρήστη Μαρία Παπαδοπούλου και επιλέξτε να ανοίξετε το προφίλ της. Ελέγχετε τα στοιχεία της και επιστρέφετε στη σελίδα με όλους τους χρήστες.
Βήμα 3
Αφού επιβεβαιώσατε ότι ο χρήστης αντιστοιχεί στο χρήστη που επικοινώνησε για απενεργοποίηση του λογαριασμού επιλέγετε την απενεργοποίηση του λογαριασμού.

Σενάριο 8

Ενώ είστε συνδεδεμένος στην εφαρμογή με τον λογαριασμό του super-admin, αποφασίζετε να ελέγξετε τους χρήστες της εφαρμογής και να διαγράψετε όσους από αυτούς είναι ανενεργοί.

Βήμα 1
Μεταβείτε στο προφίλ σας για να δείτε όλους τους χρήστες του συστήματος.
Βήμα 2
Ελέγξτε την λίστα με τους εγγεγραμμένους χρήστες και παρατηρείστε ποιοι είναι ανενεργοί.
Μόλις αναγνωρίσετε τα παραπάνω συνεχίστε στο επόμενο βήμα στο σενάριο.
Βήμα 3
Πραγματοποιήστε διαγραφή των ανενεργών χρηστών.

Σενάριο 9

Παραμένετε συνδεδεμένος ως super-admin της εφαρμογής και θυμηθήκατε ότι έχει επικοινωνήσει μαζί σας ο χρήστης με username BenakiMuseum και ζήτησε να του δώσετε δικαιώματα διαχειριστή μουσείου για να μπορέσει να καταχωρήσει στην εφαρμογή το Μουσείο Μπενάκη.

Βήμα 1

Μεταβείτε στο προφίλ σας για να δείτε την λίστα των εγγεγραμμένων χρηστών.

Βήμα 2

Αναζητήστε στη λίστα τον χρήστη BenakiMuseum και παρατηρήστε τις δυνατότητες που έχετε ως super-admin.

Μόλις αναγνωρίσετε τα παραπάνω συνεχίστε στο επόμενο βήμα στο σενάριο.

Βήμα 3

Πατήστε το κουμπί για επεξεργασία προφίλ του χρήστη BenakiMuseum.

Βήμα 4

Βρείτε την δυνατότητα καταχώρησης του απαραίτητου δικαιώματος, «Μπορεί να δημιουργήσει μουσείο», και επιλέξτε την προσθήκη του.

Βήμα 5

Αποθηκεύστε τις αλλαγές.

Σενάριο 10

Παραμένετε συνδεδεμένος ως super-admin της εφαρμογής, και επιθυμείτε να διαγράψετε ένα μουσείο από την σεελίδα.

Βήμα 1

Μεταβείτε στην σελίδα με την λίστα όλων των διαθέσιμων μουσείων.

Βήμα 2

Βρείτε και επιλέξτε το μουσείο «Ναυτικό και Λαογραφικό Μουσείο Ιθάκης, Ιθάκη» προς διαγραφή.

Βήμα 3

Επιλέξτε τη διαγραφή του μουσείου.

Βήμα 4

Αποσυνδεθείτε από την εφαρμογή.

Σενάριο 11

Είστε ο διαχειριστής του μουσείου Ναυτικό και Λαογραφικό Μουσείο Ιθάκης και επιλέγετε να συνδεθείτε ως museum-admin και να πραγματοποιήσετε την ενημέρωση των στοιχείων του μουσείου.

Βήμα 1

Συνδεθείτε στην πλατφόρμα με τον λογαριασμό museum-admin
username: canCreateMuseum
password: canCreateMuseum

Βήμα 2

Μεταβείτε στη λίστα των μουσείων.

Βήμα 3

Αφού βρείτε το μουσείο στο οποίο είστε διαχειριστής επιλέξτε να τροποποιήσετε τα στοιχεία του μουσείου.

Βήμα 4

Ολοκληρώστε τις αλλαγές που επιθυμείτε και αποθηκεύστε τις αλλαγές.

Βήμα 5

Μεταβείτε στη σελίδα του μουσείου για να δείτε τις ενημερωμένες πληροφορίες.

Βήμα 6

Πραγματοποιήστε αποσύνδεση από το σύστημα.

Ενότητα Γ | Ερωτήσεις Αξιολόγησης της Ιστοσελίδας

Βαθμολογική κλίμακα

Καθόλου	Λίγο	Μέτρια	Πολύ	Πάρα πολύ
1	2	3	4	5
Απαράδεκτη	Μη ικανοποιητική	Μέτρια	Ικανοποιητική	Πολύ Καλή

Εάν δεν είστε σε θέση να απαντήσετε μια ερώτηση παρακαλούμε να επιλέξετε την μέση επιλογή της κλίμακας (3 – Μέτρια).

Αξιολογήστε τις ακόλουθες προτάσεις επιλέγοντας την στήλη της επιλογής σας:

Γ1. Νομίζω ότι θα ήθελα να χρησιμοποιώ αυτό το δικτυακό τόπο συχνά.

1	2	3	4	5
Καθόλου	Λίγο	Μέτρια	Πολύ	Πάρα πολύ

Γ2. Βρήκα αυτό το δικτυακό τόπο αδικαιολόγητα περίπλοκο.

1	2	3	4	5
Καθόλου	Λίγο	Μέτρια	Πολύ	Πάρα πολύ

Γ3. Σκέφτηκα ότι αυτός ο δικτυακός τόπος ήταν εύκολος στη χρήση.

1	2	3	4	5
Καθόλου	Λίγο	Μέτρια	Πολύ	Πάρα πολύ

Γ4. Νομίζω ότι θα χρειαστώ βοήθεια από κάποιον τεχνικό για να είμαι σε θέση να χρησιμοποιήσω αυτό το δικτυακό τόπο.

1	2	3	4	5
Καθόλου	Λίγο	Μέτρια	Πολύ	Πάρα πολύ

Γ5. Βρήκα τις διάφορες λειτουργίες σε αυτό το δικτυακό τόπο καλά ολοκληρωμένες.

1	2	3	4	5
Καθόλου	Λίγο	Μέτρια	Πολύ	Πάρα πολύ

Γ6.Σκέφτηκα ότι υπήρχε μεγάλη ασυνέπεια σε αυτό το δικτυακό τόπο.

1	2	3	4	5
Καθόλου	Λίγο	Μέτρια	Πολύ	Πάρα πολύ

Γ7. Φαντάζομαι ότι οι περισσότεροι άνθρωποι θα μάθουν να χρησιμοποιούν αυτό το δικτυακό τόπο πολύ γρήγορα.

1	2	3	4	5
Καθόλου	Λίγο	Μέτρια	Πολύ	Πάρα πολύ

Γ8. Βρήκα αυτό το δικτυακό τόπο πολύ περίπλοκο/δύσκολο στη χρήση.

1	2	3	4	5
Καθόλου	Λίγο	Μέτρια	Πολύ	Πάρα πολύ

Γ9. Ένιωσα πολύ σίγουρος/η χρησιμοποιώντας αυτό το δικτυακό τόπο.

1	2	3	4	5
Καθόλου	Λίγο	Μέτρια	Πολύ	Πάρα πολύ

Γ10. Χρειάστηκε να μάθω πολλά πράγματα πριν να μπορέσω να ξεκινήσω με αυτό το δικτυακό τόπο.

1	2	3	4	5
Καθόλου	Λίγο	Μέτρια	Πολύ	Πάρα πολύ

Ιδιαίτερη αξία έχουν οι παρατηρήσεις, τα σχόλιά ή οι προτάσεις σας. Παρακαλούμε να τα συμπεριλάβετε παρακάτω. Για παράδειγμα, μπορείτε να αναφερθείτε σε λόγους που σας οδήγησαν να δώσετε συγκεκριμένες βαθμολογίας, ή να κάνετε συγκεκριμένες συστάσεις για την βελτίωση της ιστοσελίδας.

Ενότητα Δ | Σχόλια και Παρατηρήσεις

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

.....

ΠΑΡΑΡΤΗΜΑ Β – ΔΕΛΤΙΟ ΑΞΙΟΛΟΓΗΣΗΣ

ΣΕΝΑΡΙΟ 1

ΒΗΜΑ	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ	ΛΑΘΗ κ' ΧΡΟΝΟΣ ΑΝΑΚΑΜΨΗΣ	ΕΚΦΡΑΣΕΙΣ	
			ΙΚΑΝΟΠΟΙΗΣΗΣ	ΑΠΟΓΟΗΤΕΥΣΗΣ
1				
2				
3				
4				
5				
ΣΥΝΟΛΟ				

ΣΕΝΑΡΙΟ 2

ΒΗΜΑ	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ	ΛΑΘΗ κ' ΧΡΟΝΟΣ ΑΝΑΚΑΜΨΗΣ	ΕΚΦΡΑΣΕΙΣ	
			ΙΚΑΝΟΠΟΙΗΣΗΣ	ΑΠΟΓΟΗΤΕΥΣΗΣ
1				
2				
3				
4				
ΣΥΝΟΛΟ				

ΣΕΝΑΡΙΟ 3

ΒΗΜΑ	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ	ΛΑΘΗ κ' ΧΡΟΝΟΣ ΑΝΑΚΑΜΨΗΣ	ΕΚΦΡΑΣΕΙΣ	
			ΙΚΑΝΟΠΟΙΗΣΗΣ	ΑΠΟΓΟΗΤΕΥΣΗΣ
1				
2				
3				
4				
ΣΥΝΟΛΟ				

ΣΕΝΑΡΙΟ 4

ΒΗΜΑ	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ	ΛΑΘΗ κ' ΧΡΟΝΟΣ ΑΝΑΚΑΜΨΗΣ	ΕΚΦΡΑΣΕΙΣ	
			ΙΚΑΝΟΠΟΙΗΣΗΣ	ΑΠΟΓΟΗΤΕΥΣΗΣ
1				
2				
3				
4				
5				
6				
ΣΥΝΟΛΟ				

ΣΕΝΑΡΙΟ 5

ΒΗΜΑ	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ	ΛΑΘΗ κ' ΧΡΟΝΟΣ ΑΝΑΚΑΜΨΗΣ	ΕΚΦΡΑΣΕΙΣ	
			ΙΚΑΝΟΠΟΙΗΣΗΣ	ΑΠΟΓΟΗΤΕΥΣΗΣ
1				
2				
3				
4				
5				
ΣΥΝΟΛΟ				

ΣΕΝΑΡΙΟ 6

ΒΗΜΑ	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ	ΛΑΘΗ κ' ΧΡΟΝΟΣ ΑΝΑΚΑΜΨΗΣ	ΕΚΦΡΑΣΕΙΣ	
			ΙΚΑΝΟΠΟΙΗΣΗΣ	ΑΠΟΓΟΗΤΕΥΣΗΣ
1				
2				
3				
4				
5				
ΣΥΝΟΛΟ				

ΣΕΝΑΡΙΟ 7

ΒΗΜΑ	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ	ΛΑΘΗ κ' ΧΡΟΝΟΣ ΑΝΑΚΑΜΨΗΣ	ΕΚΦΡΑΣΕΙΣ	
			ΙΚΑΝΟΠΟΙΗΣΗΣ	ΑΠΟΓΟΗΤΕΥΣΗΣ
1				
2				
3				
ΣΥΝΟΛΟ				

ΣΕΝΑΡΙΟ 8

ΒΗΜΑ	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ	ΛΑΘΗ κ' ΧΡΟΝΟΣ ΑΝΑΚΑΜΨΗΣ	ΕΚΦΡΑΣΕΙΣ	
			ΙΚΑΝΟΠΟΙΗΣΗΣ	ΑΠΟΓΟΗΤΕΥΣΗΣ
1				
2				
3				
ΣΥΝΟΛΟ				

ΣΕΝΑΡΙΟ 9

ΒΗΜΑ	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ	ΛΑΘΗ κ' ΧΡΟΝΟΣ ΑΝΑΚΑΜΨΗΣ	ΕΚΦΡΑΣΕΙΣ	
			ΙΚΑΝΟΠΟΙΗΣΗΣ	ΑΠΟΓΟΗΤΕΥΣΗΣ
1				
2				
3				
4				
5				
ΣΥΝΟΛΟ				

ΣΕΝΑΡΙΟ 10

ΒΗΜΑ	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ	ΛΑΘΗ κ' ΧΡΟΝΟΣ ΑΝΑΚΑΜΨΗΣ	ΕΚΦΡΑΣΕΙΣ	
			ΙΚΑΝΟΠΟΙΗΣΗΣ	ΑΠΟΓΟΗΤΕΥΣΗΣ
1				
2				
3				
4				
ΣΥΝΟΛΟ				

ΣΕΝΑΡΙΟ 11

ΒΗΜΑ	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ	ΛΑΘΗ κ' ΧΡΟΝΟΣ ΑΝΑΚΑΜΨΗΣ	ΕΚΦΡΑΣΕΙΣ	
			ΙΚΑΝΟΠΟΙΗΣΗΣ	ΑΠΟΓΟΗΤΕΥΣΗΣ
1				
2				
3				
4				
5				
6				
ΣΥΝΟΛΟ				

ΠΑΡΑΡΤΗΜΑ Γ – ΔΕΔΟΜΕΝΑ

Συγκεντρωτικός Πίνακας Ποσοτικών Δεδομένων (το σενάριο Ολοκληρώθηκε με επιτυχία / Δεν ολοκληρώθηκε **)**

Χρόνος σε sec (λάθη)	Σενάριο 1						Σενάριο 2					Σενάριο 3				
Βήμα	**1**	**2**	**3**	**4**	**5**	**Σύνολο**	**1**	**2**	**3**	**4**	**Σύνολο**	**1**	**2**	**3**	**4**	**Σύνολο**
Χρήστης1	8(0)	9(0)	16(0)	15(0)	7(0)	**55 (0)**	46(1)	61(1)	20(0)	42(0)	**326(4)**	20(0)	19(0)	17(0)		**56(0)**
Χρήστης2	11(0)	14(0)	20(1)	19(0)	12(0)	**76 (1)**	13(0)	13(0)	36(0)	34(0)	**156(0)**	21(0)	29(0)	9(0)		**59(0)**
Χρήστης3	13(0)	18(0)	24(1)	20(0)	10(0)	**85 (1)**	15(0)	14(0)	27(0)	48(0)	**242(1)**	32(0)	50(1)	16(0)		**98(1)**
Χρήστης4	9 (0)	10(0)	18(0)	14(0)	8(0)	**59 (0)**	21(0)	13(0)	31(0)	13(0)	**249(1)**	36(0)	40(0)	45(0)		**121(0)**
Χρήστης5	15(0)	15(0)	21(0)	18(1)	10(1)	**79 (2)**	34(0)	16(0)	36(0)	20(0)	**256(0)**	14(0)	97(1)	33(0)		**144(1)**
Χρήστης6	12(0)	11(0)	18(0)	15(0)	9(0)	**65 (0)**	6(0)	14(0)	13(0)	17(0)	**114(0)**	35(0)	14(0)	11(0)		**60(0)**
Χρήστης7	7(0)	9(0)	15(0)	17(0)	9(0)	**57 (0)**	33(0)	17(0)	39(1)	38(0)	**294(3)**	92(2)	55(0)	47(0)		**194(2)**
Χρήστης8	12(0)	11(0)	16(0)	23(0)	11(0)	**73 (0)**	19(0)	9(0)	21(0)	19(0)	**126(0)**	19(0)	81(1)	77(1)		**177(2)**
Χρήστης9	18(1)	20(0)	22(0)	19(0)	10(0)	**89 (1)**	39(0)	21(0)	34(0)	33(0)	**336(4)**	41(0)	66(1)	18(0)		**125(1)**
Χρήστης10	10(0)	11(0)	17 (0)	22(0)	7(0)	**67 (0)**	76(1)	10(0)	41(0)	30(0)	**231(1)**	14(0)	29(0)	23(0)		**66(0)**
Μέσος Χρόνος (Λάθη)	11,5 (0,1)	12,8 (0)	18,7 (0,2)	18,2 (0,1)	9,3 (0,1)	**70,5 (0,5)**	30,2 (0,2)	18,8 (0,1)	29,8 (0,1)	29,4 (0)	**233 (1,4)**	32,4 (0,2)	48 (0,4)	29,6 (0,1)		**110 (0,7)**
Ολοκλήρωση με Επιτυχία %	100,-	100,-	100,-	100,-	100,-		100,-	100,-	100,-	100,-		100,-	100,-	100,-		

Χρόνος σε sec (λάθη)	Σενάριο 4							Σενάριο 5					
Βήμα	**1**	**2**	**3**	**4**	**5**	**6**	**Σύνολο**	**1**	**2**	**3**	**4**	**5**	**Σύνολο**
Χρήστης1	46(1)	61(1)	20(0)	42(0)	8(0)	121 (2)	**326(4)**	20(0)	19(0)	17(0)			**56(0)**
Χρήστης2	13(0)	13(0)	36(0)	34(0)	7(0)	25(0)	**156(0)**	21(0)	29(0)	9(0)			**59(0)**
Χρήστης3	15(0)	14(0)	27(0)	48(0)	9(0)	82(1)	**242(1)**	32(0)	50(1)	16(0)			**98(1)**
Χρήστης4	21(0)	13(0)	31(0)	13(0)	10(0)	94(1)	**249(1)**	36(0)	40(0)	45(0)			**121(0)**
Χρήστης5	34(0)	16(0)	36(0)	20(0)	6(0)	53(0)	**256(0)**	14(0)	97(1)	33(0)			**144(1)**
Χρήστης6	6(0)	14(0)	13(0)	17(0)	11(0)	14(0)	**114(0)**	35(0)	14(0)	11(0)			**60(0)**
Χρήστης7	33(0)	17(0)	39(1)	38(0)	28(0)	59(2)	**294(3)**	92(2)	55(0)	47(0)			**194(2)**
Χρήστης8	19(0)	9(0)	21(0)	19(0)	13(0)	12(0)	**126(0)**	19(0)	81(1)	77(1)			**177(2)**
Χρήστης9	39(0)	21(0)	34(0)	33(0)	7(0)	125(3)	**336(4)**	41(0)	66(1)	18(0)			**125(1)**
Χρήστης10	76(1)	10(0)	41(0)	30(0)	19(0)	18(0)	**231(1)**	14(0)	29(0)	23(0)			**66(0)**
Μέσος Χρόνος (Λάθη)	30,2 (0,2)	18,8 (0,1)	29,8 (0,1)	29,4 (0)	11,8 (0)	60,3 (0,9)	**233 (1,4)**	32,4 (0,2)	48 (0,4)	29,6 (0,1)			**110 (0,7)**
Ολοκλήρωση με Επιτυχία %	100,-	100,-	100,-	100,-	100,-	90,-		100,-	100,-	100,-			

Χρόνος σε sec (λάθη)	Σενάριο 6						Σενάριο 7				Σενάριο 8			
Βήμα	**1**	**2**	**3**	**4**	**5**	**Σύνολο**	**1**	**2**	**3**	**Σύνολο**	**1**	**2**	**3**	**Σύνολο**
Χρήστης1	27(0)	25(0)	16(0)	6(0)		**74(0)**	46(1)	61(1)	20(0)	**326(4)**	20(0)	19(0)	17(0)	**56(0)**
Χρήστης2	18(0)	32(0)	10(0)	11(0)		**71(0)**	13(0)	13(0)	36(0)	**156(0)**	21(0)	29(0)	9(0)	**59(0)**
Χρήστης3	29(0)	18(0)	13(0)	31(0)		**91(0)**	15(0)	14(0)	27(0)	**242(1)**	32(0)	50(1)	16(0)	**98(1)**
Χρήστης4	23 (0)	24(0)	12(0)	13(0)		**72(0)**	21(0)	13(0)	31(0)	**249(1)**	36(0)	40(0)	45(0)	**121(0)**
Χρήστης5	56(0)	12(0)	7(0)	41(1)		**116(1)**	34(0)	16(0)	36(0)	**256(0)**	14(0)	97(1)	33(0)	**144(1)**
Χρήστης6	12(0)	11(0)	6(0)	13(0)		**42(0)**	6(0)	14(0)	13(0)	**114(0)**	35(0)	14(0)	11(0)	**60(0)**
Χρήστης7	7(0)	23(0)	9(0)	32(0)		**71(0)**	33(0)	17(0)	39(1)	**294(3)**	92(2)	55(0)	47(0)	**194(2)**
Χρήστης8	28(0)	19(0)	8(0)	18(0)		**73(0)**	19(0)	9(0)	21(0)	**126(0)**	19(0)	81(1)	77(1)	**177(2)**
Χρήστης9	25(1)	46(0)	12(0)	30(0)		**113(1)**	39(0)	21(0)	34(0)	**336(4)**	41(0)	66(1)	18(0)	**125(1)**
Χρήστης10	26(0)	15(0)	10(0)	22(0)		**73(0)**	76(1)	10(0)	41(0)	**231(1)**	14(0)	29(0)	23(0)	**66(0)**
Μέσος Χρόνος (Λάθη)	25,1 (0,1)	22,5 (0)	10,3 (0)	21,7 (0,1)		**79,6 (0,2)**	30,2 (0,2)	18,8 (0,1)	29,8 (0,1)	**233 (1,4)**	32,4 (0,2)	48 (0,4)	29,6 (0,1)	**110 (0,7)**
Ολοκλήρωση με Επιτυχία %	100,-	100,-	100,-	100,-			100,-	100,-	100,-		100,-	100,-	100,-	

Χρόνος σε sec (λάθη)	Σενάριο 9						Σενάριο 10				
Βήμα	**1**	**2**	**3**	**4**	**5**	**Σύνολο**	**1**	**2**	**3**	**4**	**Σύνολο**
Χρήστης1	27(0)	25(0)	16(0)	6(0)		**74(0)**	46(1)	61(1)	20(0)	42(0)	**326(4)**
Χρήστης2	18(0)	32(0)	10(0)	11(0)		**71(0)**	13(0)	13(0)	36(0)	34(0)	**156(0)**
Χρήστης3	29(0)	18(0)	13(0)	31(0)		**91(0)**	15(0)	14(0)	27(0)	48(0)	**242(1)**
Χρήστης4	23 (0)	24(0)	12(0)	13(0)		**72(0)**	21(0)	13(0)	31(0)	13(0)	**249(1)**
Χρήστης5	56(0)	12(0)	7(0)	41(1)		**116(1)**	34(0)	16(0)	36(0)	20(0)	**256(0)**
Χρήστης6	12(0)	11(0)	6(0)	13(0)		**42(0)**	6(0)	14(0)	13(0)	17(0)	**114(0)**
Χρήστης7	7(0)	23(0)	9(0)	32(0)		**71(0)**	33(0)	17(0)	39(1)	38(0)	**294(3)**
Χρήστης8	28(0)	19(0)	8(0)	18(0)		**73(0)**	19(0)	9(0)	21(0)	19(0)	**126(0)**
Χρήστης9	25(1)	46(0)	12(0)	30(0)		**113(1)**	39(0)	21(0)	34(0)	33(0)	**336(4)**
Χρήστης10	26(0)	15(0)	10(0)	22(0)		**73(0)**	76(1)	10(0)	41(0)	30(0)	**231(1)**
Μέσος Χρόνος (Λάθη)	25,1 (0,1)	22,5 (0)	10,3 (0)	21,7 (0,1)		**79,6 (0,2)**	30,2 (0,2)	18,8 (0,1)	29,8 (0,1)	29,4 (0)	**233 (1,4)**
Ολοκλήρωση με Επιτυχία %	100,-	100,-	100,-	100,-			100,-	100,-	100,-	100,-	

Χρόνος σε sec (λάθη)	Σενάριο 11						
Βήμα	**1**	**2**	**3**	**4**	**5**	**6**	**Σύνολο**
Χρήστης1	20(0)	19(0)	17(0)				**56(0)**
Χρήστης2	21(0)	29(0)	9(0)				**59(0)**
Χρήστης3	32(0)	50(1)	16(0)				**98(1)**
Χρήστης4	36(0)	40(0)	45(0)				**121(0)**
Χρήστης5	14(0)	97(1)	33(0)				**144(1)**
Χρήστης6	35(0)	14(0)	11(0)				**60(0)**
Χρήστης7	92(2)	55(0)	47(0)				**194(2)**
Χρήστης8	19(0)	81(1)	77(1)				**177(2)**
Χρήστης9	41(0)	66(1)	18(0)				**125(1)**
Χρήστης10	14(0)	29(0)	23(0)				**66(0)**
Μέσος Χρόνος (Λάθη)	32,4 (0,2)	48 (0,4)	29,6 (0,1)				**110 (0,7)**
Ολοκλήρωση με Επιτυχία %	100,-	100,-	100,-				