ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή Εργασία

Τίτλος Πτυχιακής Εργασίας	Ανάπτυξη Ιστοσελίδας Πολιτιστικών Δρώμενων υλοποιημένη σε C# και Angular Development of a Cultural Events Website implemented in C# and Angular
Ονοματεπώνυμο Φοιτητή	Τσουμέα Κλεοπάτρα
Πατρώνυμο	Τσουμέας Χρήστος
Αριθμός Μητρώου	Π19177
Επιβλέπων	Αλέπης Ευθύμιος, Καθηγητής

Ημερομηνία Παράδοσης **Σεπτέμβριος 2024**

Copyright ©

Ευχαριστίες

Θα ήθελα να εκφράσω τις πιο θερμές μου ευχαριστίες στον επιβλέποντα καθηγητή μου κ. Ευθύμιο Αλέπη για την καθοδήγηση κατά τη διάρκεια υλοποίησης της πτυχιακής μου εργασίας. Καθώς και θα ήθελα να ευχαριστήσω και όλους τους κοντινούς μου ανθρώπους που όλο αυτό το διάστημα με στήριζαν και με ενθάρρυναν.

Περίληψη

Το θέμα της παρούσας πτυχιακής εργασίας αφορά την ανάπτυξη μιας ολοκληρωμένης ιστοσελίδας “Artena”, μέσω της οποίας παρουσιάζονται ανά κατηγορίες πολιτιστικά δρώμενα (όπως για παράδειγμα θεατρικές παραστάσεις, συναυλίες, εκθέσεις, ταινίες). Η ιστοσελίδα απευθύνεται σε χρήστες που εργάζονται στο συγκεκριμένο κλάδο και μη. Σκοπός της εφαρμογής είναι οι χρήστες να μπορούν να πλοηγηθούν σε μία εύχρηστη ιστοσελίδα που αφορά τις πολιτιστικές εκδηλώσεις στην Αθήνα και μη. Οι χρήστες έχουν τη δυνατότητα να δημιουργήσουν νέο λογαριασμό ώστε να μπορέσουν να αξιολογήσουν τις παραστάσεις, να τις σχολιάσουν καθώς και να δεχτούν εξατομικευμένες προτάσεις. Όσον αφορά τους επαγγελματίες του χώρου, στόχος της ιστοσελίδας είναι κατά τη σύνδεση τους να αναζητούν άλλους επαγγελματίες, όπως ένας σκηνοθέτης να αναζητήσει έναν ηθοποιό. Η ανάπτυξη της εφαρμογής για το backend βασίζεται στο Entity Framework για την αλληλεπίδραση με τη βάση και η υλοποίηση του έγινε σε C#, καθώς και η βάση δεδομένων που χρησιμοποιήθηκε είναι SQL Server. Όσον αφορά το κομμάτι του frontend, βασίστηκε στο εργαλείο Angular το οποίο είναι ένα πλαίσιο JavaScript ανοιχτού κώδικα γραμμένο σε TypeScript.

Λέξεις Κλειδιά: C#, Entity Framework, SQL, Angular, πολιτιστικά δρώμενα, εκδηλώσεις, παραστάσεις

Abstract

The topic of this bachelor thesis focuses on the development of a complete website called “Artena” through which cultural events (such as theatrical plays, music concerts, exhibitions, movies) can be displayed per category. This website is intended for both people that work in this field and the general public. The aim of this application is for the users to be able to navigate a user-friendly website that regards the cultural events happening in and out of Athens. Users have the ability to create a new account in order to be able to rate plays, comment on them and also receive personalized recommendations. For professionals of the industry, the website aims at enabling them, upon logging in, to search for other professionals such as a director searching for an actor. The backend development of the application is based on Entity Framework for the interaction with the database and is implemented in C#, meanwhile the database used is an SQL Server. The frontend part is based on the Angular tool which is a JavaScript open-source framework written in TypeScript.

Key Words: C#, Entity Framework, SQL, Angular, cultural events, events, plays

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Κατάλογος Διαγραμμάτων

Κεφάλαιο 1 - Εισαγωγή

Η Ελλάδα από τα αρχαία χρόνια έως και σήμερα χαρακτηρίζεται από την πολιτιστική της κληρονομιά. Λόγω της πλούσιας πολιτιστικής κληρονομιάς, η χώρα μας αποτελείται από τους κύριους ταξιδιωτικούς προορισμούς παγκοσμίως. Να σημειωθεί πως αποτελεί επίσης το θεμέλιο του Δυτικού Πολιτισμού, καθώς υπάρχουν πολυάριθμοι αρχαιολογικοί χώροι και μουσεία, όπως είναι το μουσείο της Ακρόπολης, το Εθνικό Αρχαιολογικό Μουσείο, καθώς και θέατρα όπως το Αρχαίο Θέατρο της Επιδαύρου. Γι' αυτό και η κληρονομιά της χώρας διατηρείται ζωντανή έως και σήμερα, λόγω των θεαματικών συναυλιακών χώρων σε υπαίθρια θέατρα.

Η παρούσα πτυχιακή εργασία εστιάζει στην ανάπτυξη μίας ιστοσελίδας με όνομα "Artena", η οποία έχει ως στόχο την παρουσίαση μίας μεγάλης ποικιλίας εκδηλώσεων όπως είναι οι θεατρικές παραστάσεις, οι ταινίες στο σινεμά, οι εκθέσεις, οι συναυλίες κ.α.. Σε όλους τους χρήστες δίνεται η δυνατότητα να αξιολογήσουν και να σχολιάσουν τις εκδηλώσεις, καθώς και να διαβάσουν βιογραφικά διαφόρων καλλιτεχνών π.χ. ηθοποιοί, σκηνοθέτες, τραγουδιστές, όπου θα εξυπηρετήσουν επαγγελματίες του χώρου ως προς την αναζήτηση άλλων, για την εύκολη και γρήγορη επικοινωνία σε περίπτωση που αναζητούν συνεργάτες. Συνεπώς, η παρούσα εργασία σχεδιάστηκε κυρίως για την παρακολούθηση και παρουσίαση καλλιτεχνικών εκδηλώσεων και των επαγγελματιών του συγκεκριμένου χώρου.

Αξίζει να σημειωθεί πως η εργασία περιγράφει την αναγκαιότητα μίας τέτοιας ιστοσελίδας στον χώρου του θεάματος. Έπειτα, αναλύει την υλοποίηση της ιστοσελίδας καθώς και ποιες τεχνολογίες/ εργαλεία χρησιμοποιήθηκαν, την ευέλικτη λειτουργία ως προς τον χρήστη και ποιες είναι οι απαιτήσεις. Τέλος, η εργασία ολοκληρώνεται με εξατομικευμένες προτάσεις για μελλοντικές επεκτάσεις, ώστε η εφαρμογή να μπορεί να εξελιχθεί με βάση τις νέες τεχνολογίες καθώς και με τα νέα πολιτιστικά δρώμενα.

Κεφάλαιο 2 - Ανασκόπηση Πεδίου

2.1 C# και πλατφόρμα .NET

Η πλατφόρμα .Net της Microsoft κατά την οποία εμπεριέχεται η γλώσσα προγραμματισμού C#, έχει συμβάλει στη σύγχρονη ανάπτυξη λογισμικού. Η ανάπτυξη εφαρμογών σε .NET, υλοποιούνται σε λειτουργικό σύστημα των Windows (8,10,11), ενώ παράλληλα μπορούν να δημιουργηθούν προγράμματα σε C# και σε εναλλακτικά λειτουργικά συστήματα όπως macOs,και Linux. Μία βασική λειτουργία που χρειάζεται ο προγραμματιστής σε ένα πρόγραμμα C# είναι να αντλεί δεδομένα. Η τεχνολογία Language Integrated Query (LINQ), παρουσιάστηκε πρώτα στην έκδοση .NET3.5. Σχεδιάστηκε ώστε ο προγραμματιστής να μπορεί να αποκτήσει και στη συνέχεια να αντλήσει δεδομένα από διάφορες μορφές αρχείων, όπως είναι οι σχεσιακές βάσεις δεδομένων, τα αρχεία XML και πίνακες arrays. Πιο συγκεκριμένα, μέσω της LINQ δίνεται η δυνατότητα δημιουργίας αριθμών εκφράσεων ερωτημάτων όμοιων με εκφράσεις SQL βάσης δεδομένων. Οι εκφράσεις ερωτημάτων συνδέονται με πολυάριθμους τελεστές ερωτημάτων και σχεδιάστηκαν με σκοπό να μοιάζουν σε εκφράσεις SQL.

2.2 Μοντέλα Διαδικτυακής Κίνησης και API

Για την εύκολη πρόσβαση δεδομένων από τους χρήστες χρησιμοποιούμε το API (Application Programming Interface), το οποίο αποτελείται από συναρτήσεις και διαδικασίες. Οι δυνατότητες των API προέρχονται μέσω του πρωτοκόλλου HTTP. Αξιόπιστα μοντέλα διαδικτυακής κίνησης απαιτούν ένα ενημερωμένο σύνολο δεδομένων HTTP αιτημάτων, τα οποία προέρχονται από μία ιστοσελίδα ηλεκτρονικού εμπορείου (e-commerce website). Τα συγκεκριμένα δεδομένα μπορούν να αξιοποιηθούν ανάλογα το περιεχόμενο κάθε ιστοσελίδας. Συνεπώς, για την αποτελεσματική διαχείριση των δεδομένων των API, χρειάζονται τα αξιόπιστα μοντέλα διαδικτυακής κίνησης. Τα RESTful APIs αξιοποιούνται ευρέως για την κατασκευή σύγχρονων εφαρμογών, καθώς βασίζονται σε επικοινωνίες διαδικτύου αλλά και σε εξωτερικές όπως είναι οι βάσεις δεδομένων. Απαραίτητη προϋπόθεση των REST API είναι να λειτουργούν όπως προσδιορίζονται, καθώς και να διατηρούν τις παραμέτρους ασφαλείας. Όπως αναφέρθηκε και παραπάνω, για την επικοινωνία βασίζονται στο πρωτόκολλο HTTP, με την χρήση των αντίστοιχων μεθόδων HTTP. Οι οποίες είναι οι: POST, GET, PUT, DELETE.

2.3 Το Αρχαίο Ελληνικό Θέατρο

Από την αρχαιότητα η Αθήνα ήταν η γενέτειρα του δυτικού πολιτισμού του θεάτρου, της τέχνης και μετράει χιλιάδες χρόνια πολιτισμικής κληρονομιάς καθώς και συνεχίζει να είναι ενεργά η πρωτεύουσα του πολιτισμού. Έως και σήμερα, τα πολιτιστικά δρώμενα που συμβαίνουν στην πόλη είναι άπειρα. Καθώς αξίζει να σημειωθεί πως η εξέλιξη της αρχιτεκτονικής του αρχαίου θεάτρου συνέβαλε ραγδαία στον πολιτισμό. Από την αρχαία Ελλάδα μέχρι και την ρωμαϊκή αρχαιότητα εξελίχθηκε ραγδαία η

διαμόρφωση των θεατρικών κτιρίων καθώς ασκήθηκε μεγάλη επιρροή και παρέμεινε ως σύγχρονη σχεδιαστική λύση.

Το αρχαίο ελληνικό θέατρο παρουσιάστηκε στην Αθήνα τον 6ο αιώνα π.Χ. Όπου και διαδραματίζονταν παραστάσεις τραγωδιών σε θρησκευτικές γιορτές. Στη συνέχεια μέσα από τις τραγωδίες δημιουργήθηκαν και οι κωμωδίες. Τα συγκεκριμένα είδη παραστάσεων εξαπλώθηκαν σε ολόκληρη την Μεσόγειο, τα οποία και επηρέασαν το ρωμαϊκό και ελληνιστικό θέατρο. Η δημιουργία της τραγωδίας προήλθε μέσω της λατρείας του Διόνυσου, όπου ήταν μία τελετουργία τραγουδιών με χρήση μάσκας. Τέλος, ο Διόνυσος χαρακτηρίζεται ως ο θεός του θεάτρου.

2.4 ΟΜΟΙΕΣ ΕΦΑΡΜΟΓΕΣ

Αθηνόραμα

Η ιστοσελίδα «Αθηνόραμα» ασχολείται με τα πολιτιστικά δρώμενα της χώρας και μη και λειτουργεί ως οδηγός διασκέδασης. Περιέχει κυρίως άρθρα, κριτικές καθώς και ειδήσεις σχετικά με θεατρικές παραστάσεις, ταινίες, μουσική, τηλεόραση. Επιπλέον, πέραν από τις πολιτιστικές εκδηλώσεις περιλαμβάνει και σχετικά άρθρα για οδηγούς ταξιδιού, προτείνει στους αναγνώστες bar, clubs καθώς και εστιατόρια στην Αθήνα.

Μπορείτε να επισκεφτείτε την ιστοσελίδα στο: https://www.athinorama.gr/

More.com

Η ιστοσελίδα «more.com» εξυπηρετεί τον χρήστη για την ηλεκτρονική αγορά ηλεκτρονικών εισιτηρίων σε εκδηλώσεις. Μέσω αυτής παρουσιάζεται ποικιλία εκδηλώσεων, όπου κάθε χρήστης μπορεί να αναζητήσει προσεχείς εκδηλώσεις ή και τρέχουσες, καθώς και του προτείνονται αντίστοιχες. Αποτελείται από θεατρικές παραστάσεις, συναυλίες, μουσικές σκηνές, σινεμά. Κάθε εκδήλωση περιέχει μία πλήρη παρουσίαση αυτής, όπως είναι η περιγραφή της εκδήλωσης, διάφορες εικόνες καθώς και οδηγίες για τον χώρο διεξαγωγής. Τέλος, ο χρήστης μπορεί να ελέγξει την διαθεσιμότητα της εκδήλωσης και να προβεί σε αγορά εισιτηρίου ανάλογα με την ημέρα και την ώρα που επιθυμεί. Αξίζει να σημειωθεί πως πέραν από την αγορά εισιτηρίων εκδηλώσεων, περιέχει και αγορά ακτοπλοϊκών.

Μπορείτε να επισκεφτείτε την ιστοσελίδα στο: https://www.more.com/gr/

Cultureisathens

Η ιστοσελίδα «cultureisathens» παρουσιάζει τις πολιτιστικές εκδηλώσεις της πόλης και προτείνει την συμμετοχή των αναγνωστών σε διάφορες δραστηριότητες που διεξάγονται στην πόλη. Κατά την παρουσίαση πολιτιστικών εκδηλώσεων ο αναγνώστης μπορεί να διαβάσει σχετικές πληροφορίες, όπως την περιγραφή της εκδήλωσης και τον τόπο διεξαγωγής. Επιπλέον, προτείνονται διάφορα εικαστικά εργαστήρια για μικρούς και μεγάλους καθώς και πολιτιστικά μνημεία και χώρους που μπορούν να επισκεφθούν.

Μπορείτε να επισκεφτείτε την ιστοσελίδα στο: https://cultureisathens.gr/

Artandlife

Τέλος, παρόμοια ιστοσελίδα είναι και η «Artandlife», κατά την οποία παρουσιάζονται όλες οι εκδηλώσεις που διεξάγονται ή πρόκειται να διεξαχθούν στην Αθήνα. Πιο συγκεκριμένα περιλαμβάνει άρθρα ενημέρωσης των αναγνωστών για την διεξαγωγή της εκδήλωσης και διάφορες σχετικές πληροφορίες, καθώς και η προβολή βίντεο και εικόνων. Επίσης, δίνει τη δυνατότητα στους χρήστες να αξιολογήσουν και να υποβάλλουν σχόλιο για κάθε εκδήλωση. Τέλος, η ιστοσελίδα συνδέεται με την αντίστοιχη more.com (η οποία παρουσιάστηκε νωρίτερα), όπου ο χρήστης μπορεί να μεταβεί σε αυτή και να αγοράσει εισιτήριο για την εκδήλωση που τον ενδιαφέρει.

Μπορείτε να επισκεφτείτε την ιστοσελίδα στο: https://www.artandlife.gr/athens

Κεφάλαιο 3 - Παρουσίαση και Χρήση της Εφαρμογής

3.1 Στόχος της Εφαρμογής

Λόγω του μεγάλου όγκου διεξαγωγής εκδηλώσεων ετησίως στη χώρα μας υλοποιήθηκε η παρούσα εφαρμογή. Κύριος στόχος είναι ο χρήστης να μπορεί να περιηγηθεί σε μία εύχρηστη εφαρμογή, ώστε να αναζητεί πολιτιστικές εκδηλώσεις που τον ενδιαφέρουν. Πιο συγκεκριμένα, να διαβάσει διάφορες πληροφορίες των εκδηλώσεων όπως είναι η περιγραφή, οι ημέρες και ώρες που παίζεται, καθώς και να αξιολογήσει και να τις σχολιάσει. Στην περίπτωση όπου ο χρήστης εργάζεται στον χώρο του θεάματος έχει την δυνατότητα να αναζητήσει αντίστοιχα συναδέλφους του και να επικοινωνήσει μαζί τους, μέσω του email.

3.2 Σε ποιους απευθύνεται

Η παρούσα ιστοσελίδα απευθύνεται σε οποιονδήποτε επιθυμεί να γνωρίσει και να μάθει για τα πολιτιστικά δρώμενα της χώρας μας. Απευθύνεται σε άτομα ανεξαρτήτου ηλικίας, φύλλων και γνωστικών σπουδών που ενδιαφέρονται να αναζητήσουν εκδηλώσεις και επαγγελματίες του χώρου. Να σημειωθεί πως απευθύνεται σε χρήστες που έχουν σχέση ή και όχι με τον συγκεκριμένο χώρο.

3.3 Εγχειρίδιο Χρήσης

3.3.1 Μενού Χρήστη

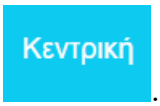
Μενού Χρήστη – Header

Κατά τη διάρκεια της πλοήγησης στην ιστοσελίδα και για την εύκολη χρήση έχουμε το παρακάτω μενού, όπου μπορούμε να επιλέξουμε όποια επιλογή επιθυμούμε και να μεταβούμε στην αντίστοιχη σελίδα.

Εικόνα 1: Header Ιστοσελίδας - Μενού

Επιλέγοντας το 

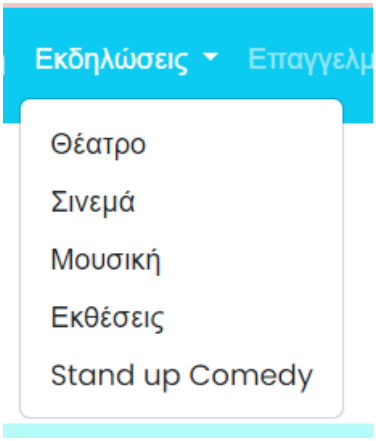
, μεταβαίνουμε στην κεντρική σελίδα της ιστοσελίδας, όπου θα αναλυθεί παρακάτω (Ενότητα 3.3.2).

Την ίδια ακριβώς λειτουργεία έχει και η επιλογή 

.

Έπειτα, πατώντας την επιλογή 

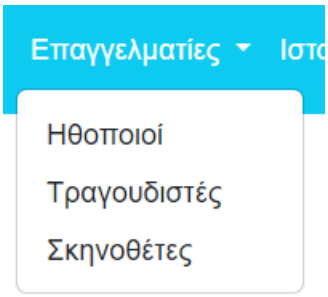
εμφανίζεται το παρακάτω dropdown με τις εκδηλώσεις τις οποίες επιθυμούμε να παρακολουθήσουμε.



Εικόνα 2: Dropdown Menu Εκδηλώσεων

Όμοια λειτουργία έχει και η επιλογή 

, πατώντας πάνω σε αυτή εμφανίζονται οι επαγγελματίες με βάση την ειδικότητα τους.



Εικόνα 3: Dropdown Menu Επαγγελματιών

Στη συνέχεια, πατώντας την επιλογή Ιστορία, μεταβαίνουμε σε μία σελίδα η οποία περιέχει διάφορες ιστορικές πληροφορίες του πολιτισμού, όπου θα αναλυθεί παρακάτω (Ενότητα 3.3.7).

Μία παρόμοια λειτουργεία έχει και η επιλογή Σχετικά, η οποία αναφέρει διάφορες πληροφορίες σχετικά με την ιστοσελίδα.

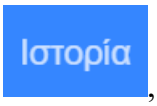
Κάνοντας κλικ στη Σύνδεση, μεταβαίνουμε στη σελίδα σύνδεσης, όπου συμπληρώνοντας τα στοιχεία μας συνδεόμαστε στη σελίδα (Ενότητα 3.3.9). Στην περίπτωση όπου είμαστε συνδεδεμένοι, η παρούσα επιλογή δεν εμφανίζεται.

Ομοίως, όταν κάνουμε κλικ στην Εγγραφή, μεταβαίνουμε στη σελίδας εγγραφής, όπου συμπληρώνοντας τα απαραίτητα στοιχεία, δημιουργούμε νέο λογαριασμό στην εφαρμογή (Ενότητα 3.3.10). Σε περίπτωση όπου είμαστε συνδεδεμένοι, η παρούσα επιλογή δεν εμφανίζεται.

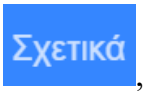
Τέλος, τελευταία επιλογή του μενού είναι η Αποσύνδεση. Η παρούσα επιλογή εμφανίζεται στο μενού στην περίπτωση που έχουμε συνδεθεί επιτυχώς στην ιστοσελίδα. Αν δεν έχουμε συνδεθεί στην εφαρμογή τότε στη θέση της επιλογής αυτής εμφανίζεται η επιλογή σύνδεση και η εγγραφή.

Μενού Χρήστη – Footer

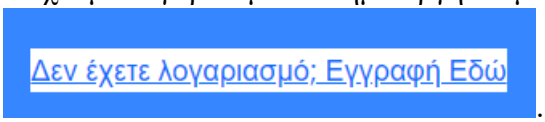
Στο τέλος κάθε σελίδας, υπάρχει η παρακάτω περιοχή (βλ. Εικόνα 5: Footer Ιστοσελίδας), όπου λειτουργεί σαν ένα σύντομο μενού χρήστη. Περιέχει την επιλογή Κεντρική, όπου μας μεταφέρει στην κεντρική σελίδα της εφαρμογής, όπως αναλύθηκε και παραπάνω.

Ομοίως, κάνοντας κλικ πάνω στην 

, μεταβαίνουμε σε μία σελίδα η οποία περιέχει διάφορες ιστορικές πληροφορίες του πολιτισμού.

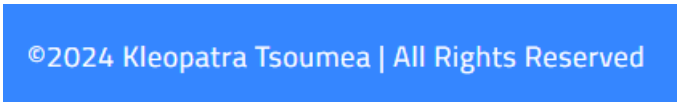
Έπειτα, όταν πατάμε την επιλογή 

, μας μεταφέρει σε μία σελίδα όπου αναφέρει διάφορες πληροφορίες σχετικά με την ιστοσελίδα.

Επιπλέον, μας δίνεται η επιλογή αν δεν έχουμε λογαριασμό να δημιουργήσουμε πατώντας το συγκεκριμένο hyperlink 

.

Τέλος, υπάρχει ένα πληροφοριακό πεδίο, όπου αναφέρει σε ποιον ανήκουν τα δικαιώματα της παρούσας ιστοσελίδας.



Εικόνα 4: Copyrights

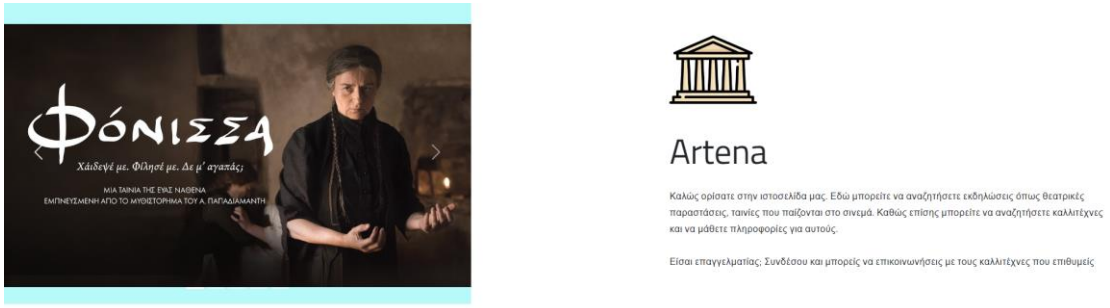
Εικόνα 5: Footer Ιστοσελίδας

3.3.2 Κεντρική Σελίδα

Ανοίγοντας την ιστοσελίδα, η πρώτη σελίδα που μας εμφανίζεται είναι η **Κεντρική Σελίδα**.

Η κεντρική σελίδα αποτελείται από ένα μενού στο πάνω μέρος της, όπου η χρήση του είναι να μας μεταφέρει σε άλλες αντίστοιχες σελίδες της ιστοσελίδας. Η παρούσα σελίδα περιέχει πληροφορίες των περισσότερων σελίδων της ιστοσελίδας (Εικόνα 12).

Πιο αναλυτικά, στην αρχή της σελίδας στο αριστερό μέρος εμφανίζονται τυχαία διάφορες εικόνες από όλες τις εκδηλώσεις που υπάρχουν μέσα στην ιστοσελίδα. Στο δεξί μέρος εμφανίζεται ο τίτλος της ιστοσελίδας καθώς και ένα κείμενο καλωσορίσματος. (Εικόνα 6: Εικόνες και Κείμενο).



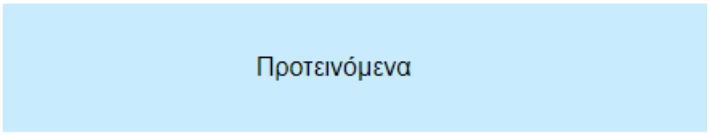
Εικόνα 6: Εικόνες και Κείμενο

Έπειτα, εμφανίζεται ένα μενού, με διάφορες επιλογές. Όπως είναι οι προτεινόμενες εκδηλώσεις, οι κατηγορίες των εκδηλώσεων καθώς και οι επαγγελματίες του χώρου του θεάματος.

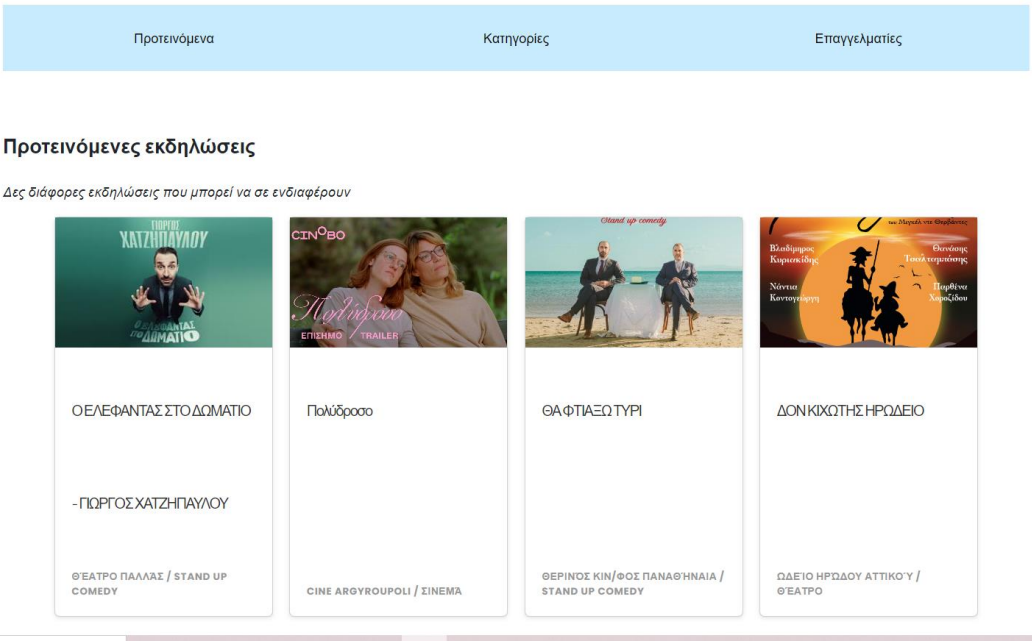
Εικόνα 7: Scroll Menu

Αφού πατήσουμε κάθε μία επιλογή από το παραπάνω μενού, μεταφερόμαστε στο αντίστοιχο σημείο της σελίδας που εμπεριέχονται οι αντίστοιχες πληροφορίες.

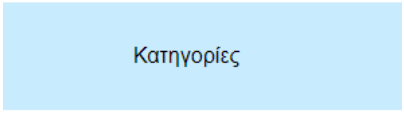
Δηλαδή, πατώντας την επιλογή



μεταβαίνουμε στο παρακάτω σημείο της κεντρικής σελίδας. (Εικόνα 8). Όπου στο συγκεκριμένο σημείο εμφανίζονται τέσσερις προτεινόμενες εκδηλώσεις οι οποίες μπορεί να μας ενδιαφέρουν. Πατώντας πάνω σε μία από αυτές τις εκδηλώσεις, μεταφερόμαστε στην αντίστοιχη σελίδα τους, όπου παρουσιάζονται σχετικές πληροφορίες για την εκδήλωση που επιλέξαμε και διάφορες άλλες λειτουργίες που θα αναλυθούν στην ενότητα της Παρουσίασης των εκδηλώσεων (Ενότητα 3.3.4).



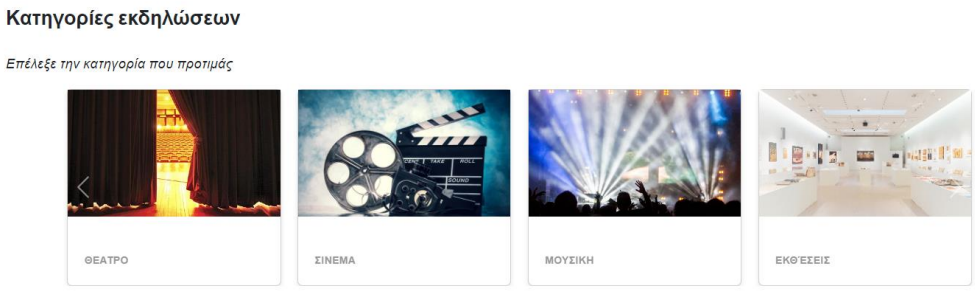
Εικόνα 8: Προτεινόμενες Εκδηλώσεις



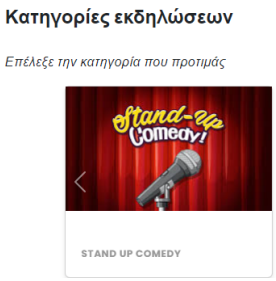
Έπειτα, πατώντας την επιλογή Κατηγορίες,
μεταφερόμαστε στο σημείο της σελίδας όπου αναγράφονται οι κατηγορίες των εκδηλώσεων που μπορούμε να αναζητήσουμε. (Εικόνα 9).

Οι κατηγορίες των εκδηλώσεων είναι περισσότερες από τέσσερις, γι' αυτό μας δίνεται η δυνατότητα πατώντας τα κουμπιά δεξιά και αριστερά μεταξύ των κατηγοριών, να τις δούμε όλες (Εικόνα 9 και Εικόνα 10).

Πατώντας πάνω σε μία από αυτές τις κατηγορίες, μεταφερόμαστε στην αντίστοιχη σελίδα τους, όπου εμφανίζονται όλες οι εκδηλώσεις της αντίστοιχης κατηγορίας που επιλέξαμε (Ενότητα 3.3.3).

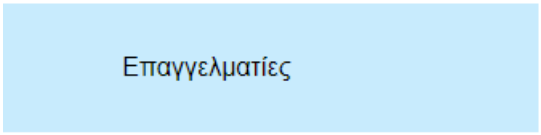


Εικόνα 9: Κατηγορίες Εκδηλώσεων I



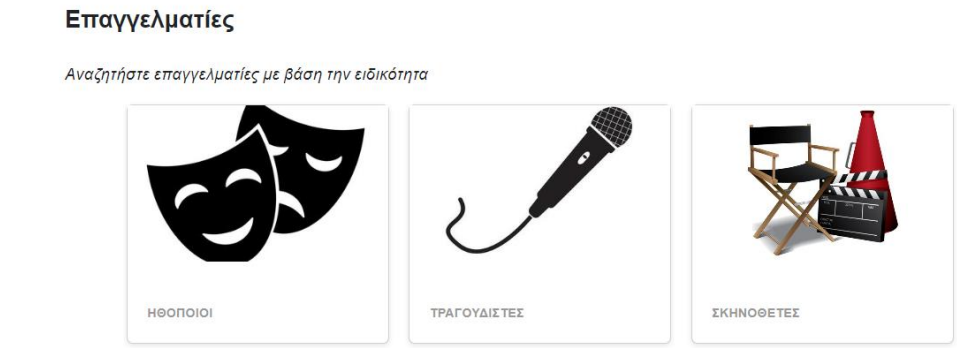
Εικόνα 10: Κατηγορίες Εκδηλώσεων II

Τελευταία επιλογή του μενού της κεντρικής σελίδας, είναι οι

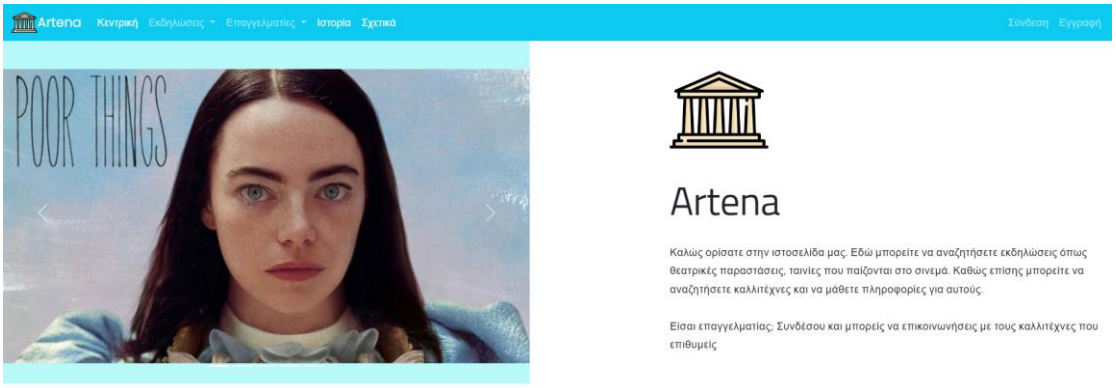


, κάνοντας κλικ πάνω σε αυτή, όπως και στις παραπάνω επιλογές, μεταφερόμαστε στο σημείο της σελίδας όπου αναγράφονται οι κατηγορίες των επαγγελματιών (Εικόνα 11).

Πατώντας πάνω σε ένα από τις κατηγορίες των επαγγελμάτων, μεταφερόμαστε στην αντίστοιχη σελίδα τους, όπου εμφανίζονται όλοι οι επαγγελματίες που εργάζονται στην αντίστοιχη κατηγορία που επιλέξαμε (Ενότητα 3.3.5).



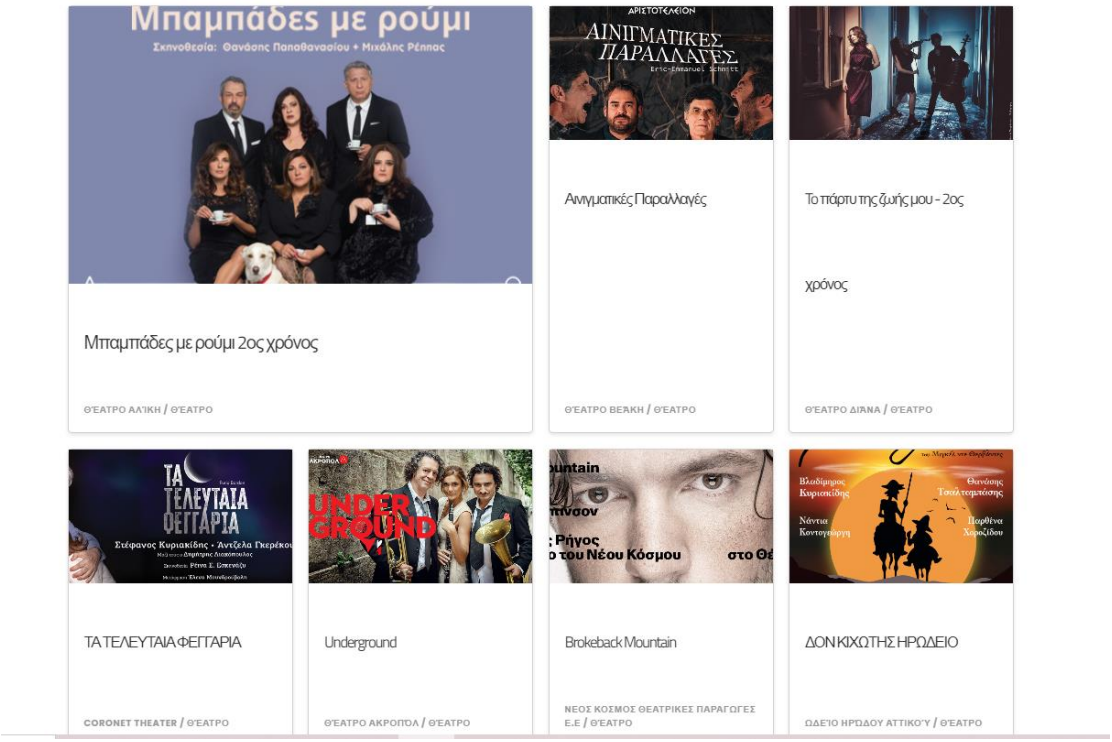
Εικόνα 11: Κατηγορίες Επαγγελματιών



Εικόνα 12: Κεντρική Σελίδα

3.3.3 Εκδηλώσεις

Η παρούσα σελίδα, εμπεριέχει όλες τις εκδηλώσεις ανάλογα την κατηγορία που έχουμε επιλέξει από το μενού της ιστοσελίδας (Εικόνα 3.1.3).



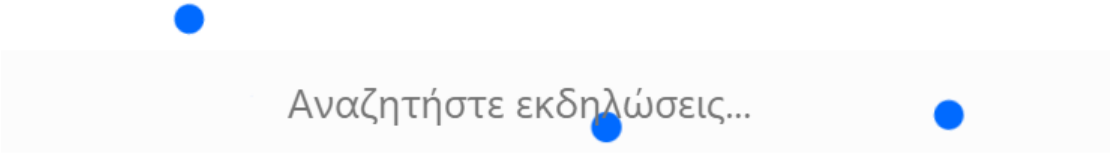
Εικόνα 13: Εκδηλώσεις

Κάθε εκδήλωση εμφανίζεται σε μία κάρτα, όπου αναφέρονται διάφορες σύντομες πληροφορίες γι' αυτή (Εικόνα 14). Πιο συγκεκριμένα, μία εικόνα της εκδήλωσης, το όνομά της, σε ποιον χώρο πραγματοποιείται, καθώς και το είδος από το οποίο αποτελείται.

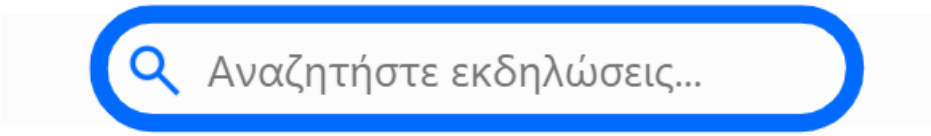
Εικόνα 14: Κάρτα Εκδήλωσης

Πατώντας πάνω στην κάρτα μίας εκδήλωσης, η σελίδα μας μεταφέρει σε μία άλλη όπου αναφέρονται όλες οι πληροφορίες της εκδήλωσης που επιλέξαμε (Ενότητα 3.3.4).

Αξίζει να σημειωθεί πως στην αρχή της παρούσας σελίδας, εμφανίζεται μία μπάρα αναζήτησης, όπου μπορούμε να αναζητήσουμε εκδηλώσεις με βάση το όνομα τους. Καλούμαστε να συμπληρώσουμε τις πληροφορίες με βάση τις οποίες θα αναζητήσουμε την εκδήλωση ή τις εκδηλώσεις που επιθυμούμε και στην συνέχεια πατώντας enter θα μας εμφανιστούν τα αποτελέσματα της αναζήτησης.

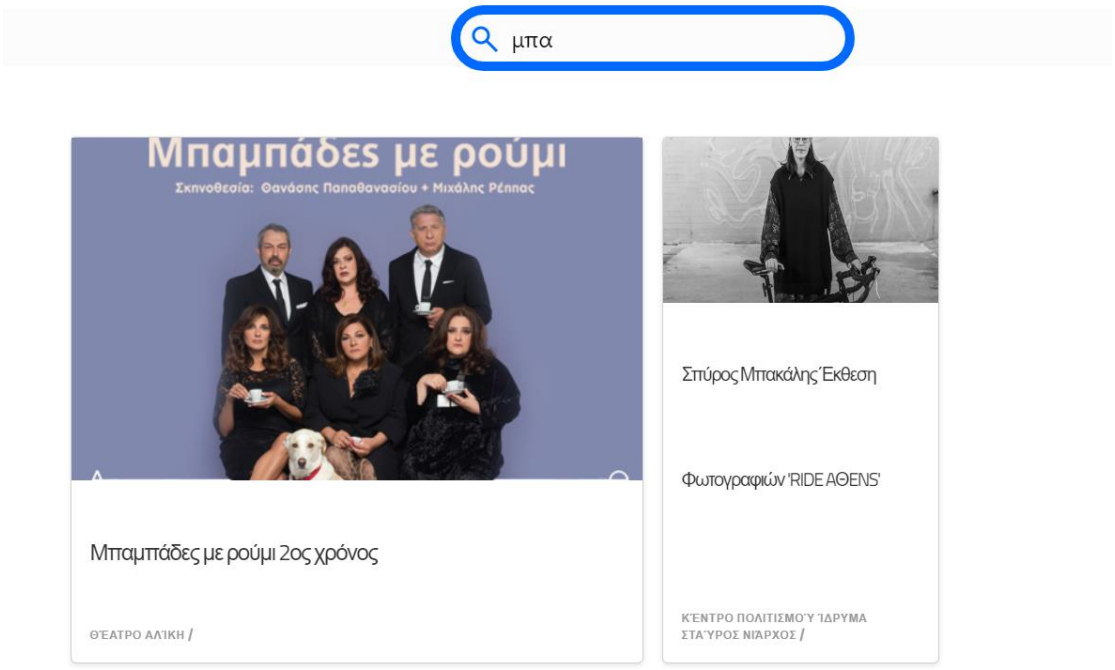


Εικόνα 15: Μπάρα Αναζήτησης I

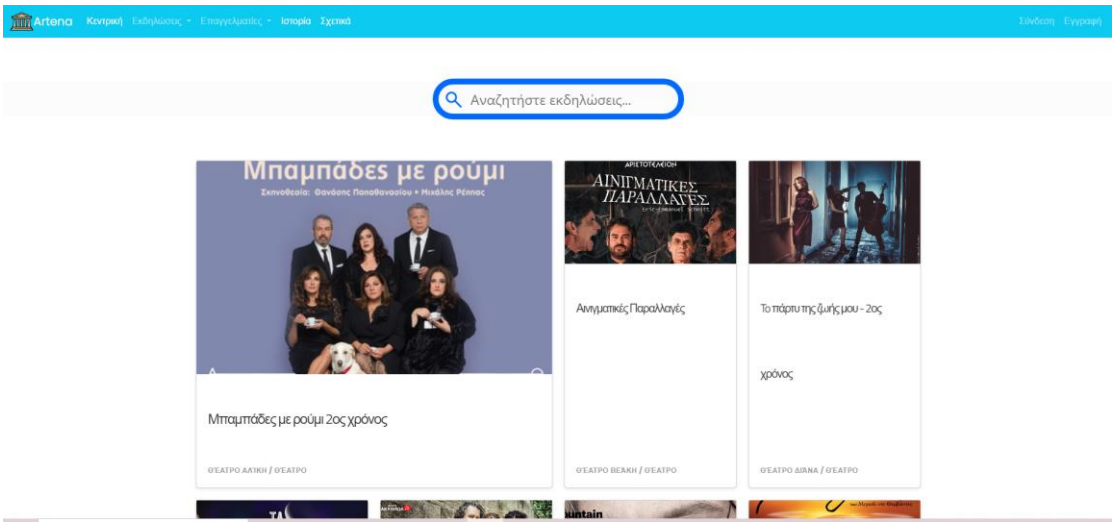


Εικόνα 16: Μπάρα Αναζήτησης II

Παραδείγματος χάρη, συμπληρώνοντας στην αναζήτηση μία λέξη, μας εμφανίζονται οι παρακάτω εκδηλώσεις ανεξαρτήτως κατηγορίας (Εικόνα 17).



Εικόνα 17: Παράδειγμα Αναζήτησης Εκδήλωσης

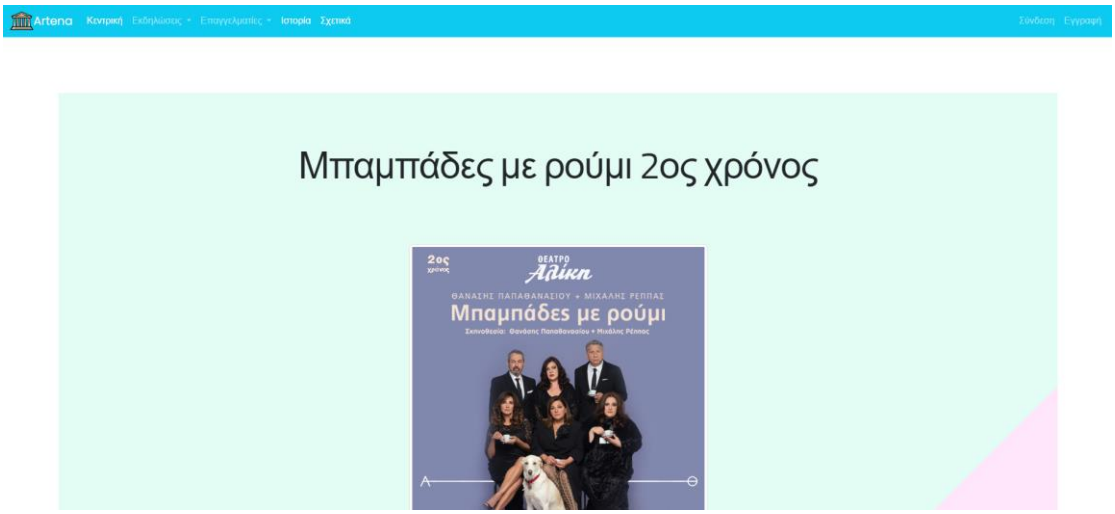


Εικόνα 18: Σελίδα Εκδηλώσεων

3.3.4 Σελίδα Παρουσίασης Εκδήλωσης

Με βάση την κατηγορία της εκδήλωσης και στη συνέχεια την εκδήλωση που θα επιλέξουμε, είτε από την αρχική είτε από την σελίδα των εκδηλώσεων, θα μεταβούμε

στην παρούσα σελίδα όπου πραγματοποιείται παρουσίαση της εκδήλωσης και περιέχει διάφορες επιπλέον λειτουργίες που θα αναλυθούν στη συνέχεια.

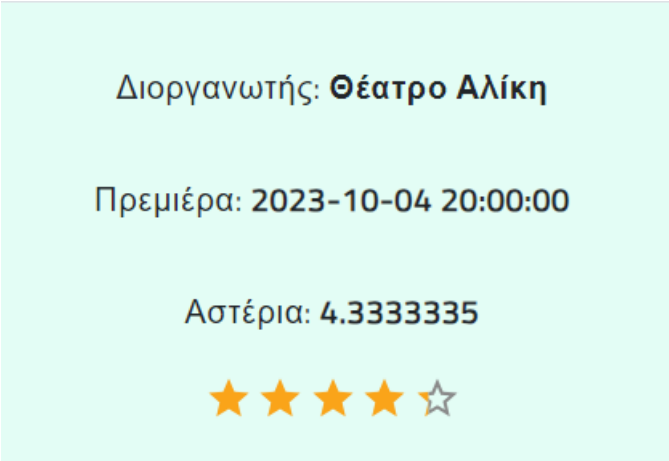


Εικόνα 19: Σελίδα Παρουσίασης Εκδήλωσης

Πιο συγκεκριμένα, η έναρξη της παρούσας σελίδας ξεκινάει με τον τίτλο της εκδήλωσης και μία εικόνα της (Εικόνα 20).

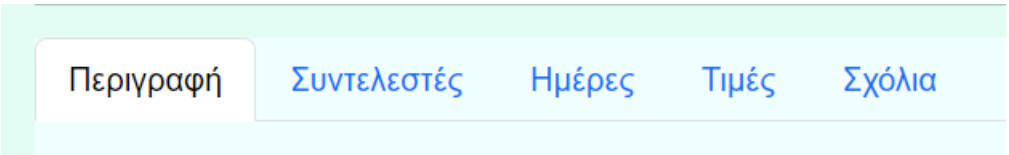
Εικόνα 20: Παρουσίαση Εκδήλωσης – Τίτλος και Εικόνα

Συνεχίζοντας, μπορούμε να δούμε τον διοργανωτή της εκδήλωσης και πότε κάνει πρεμιέρα. Έπειτα, με βάση το πλήθος αξιολογήσεων που έχουν πραγματοποιηθεί από τους χρήστες της ιστοσελίδας, αναγράφεται ο μέσος όρος αξιολόγησης (αστεριών) της εκδήλωσης (Εικόνα 21).



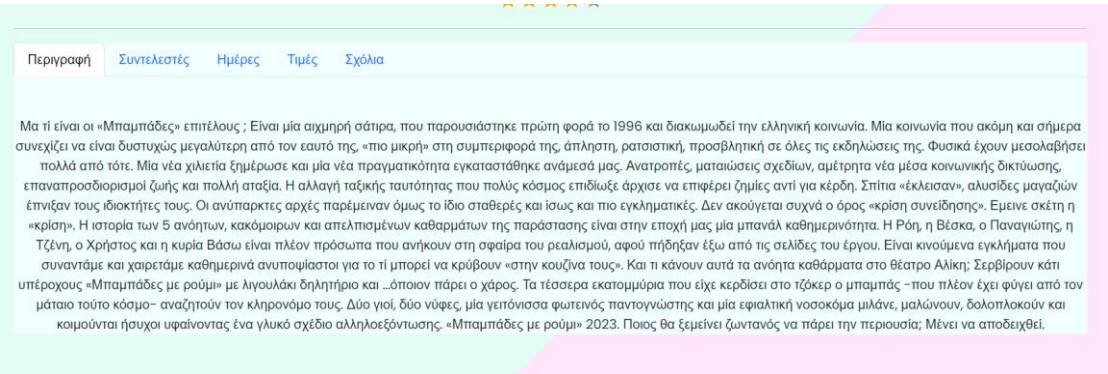
Εικόνα 21: Διοργανωτής, Πρεμιέρα, ΜΟ Αστεριών

Συνεχίζοντας την πλοήγηση στη σελίδα, μας εμφανίζεται το παρακάτω μενού (Εικόνα 22). Όπου κάνοντας κλικ πάνω σε κάθε επιλογή (tab), αλλάζουν οι πληροφορίες στο συγκεκριμένο πλαίσιο με βάση την κατηγορία.



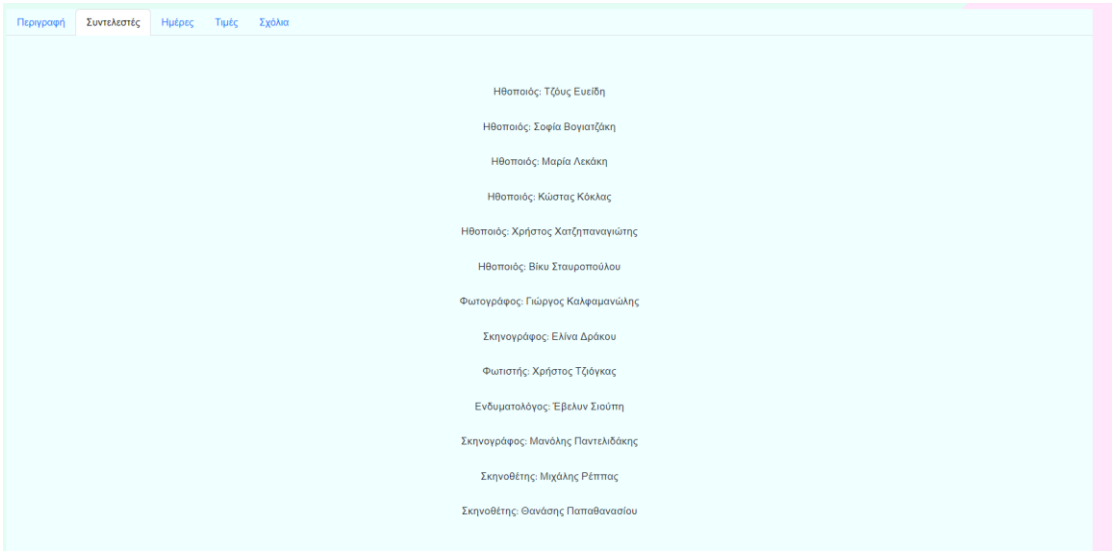
Εικόνα 22: Tab

Κάνοντας κλικ πάνω στην περιγραφή, μας εμφανίζεται η περίληψη/ περιγραφή της εκδήλωσης (Εικόνα 23).



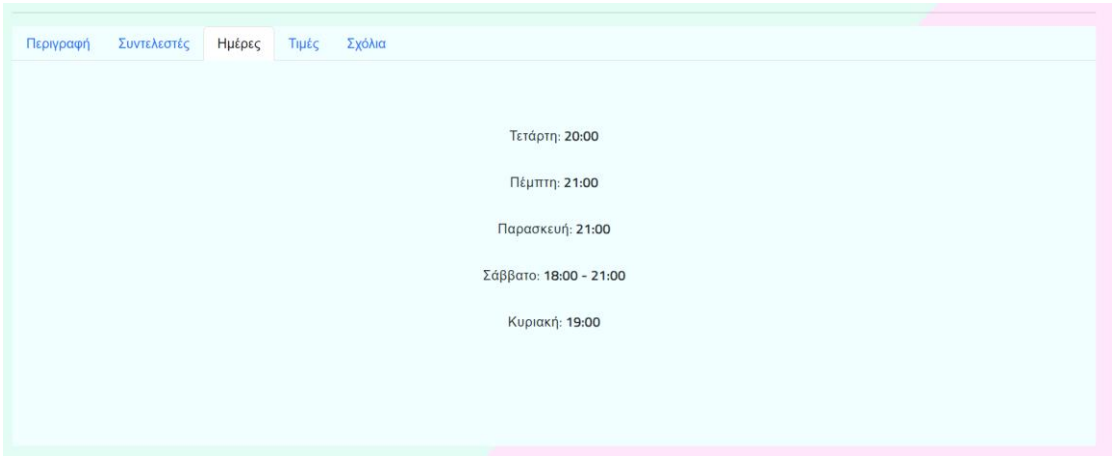
Εικόνα 23: Περιγραφή Εκδήλωσης

Κάνοντας κλικ πάνω στην επιλογή Συντελεστές, μας εμφανίζονται οι συντελεστές της εκδήλωσης (Εικόνα 24).



Εικόνα 24: Συντελεστές Εκδήλωσης

Ομοίως, κάνοντας κλικ πάνω στην επιλογή Ημέρες, μας εμφανίζονται οι ημέρες και οι ώρες που μπορούμε να παρακολουθήσουμε την εκδήλωση (Εικόνα 25).

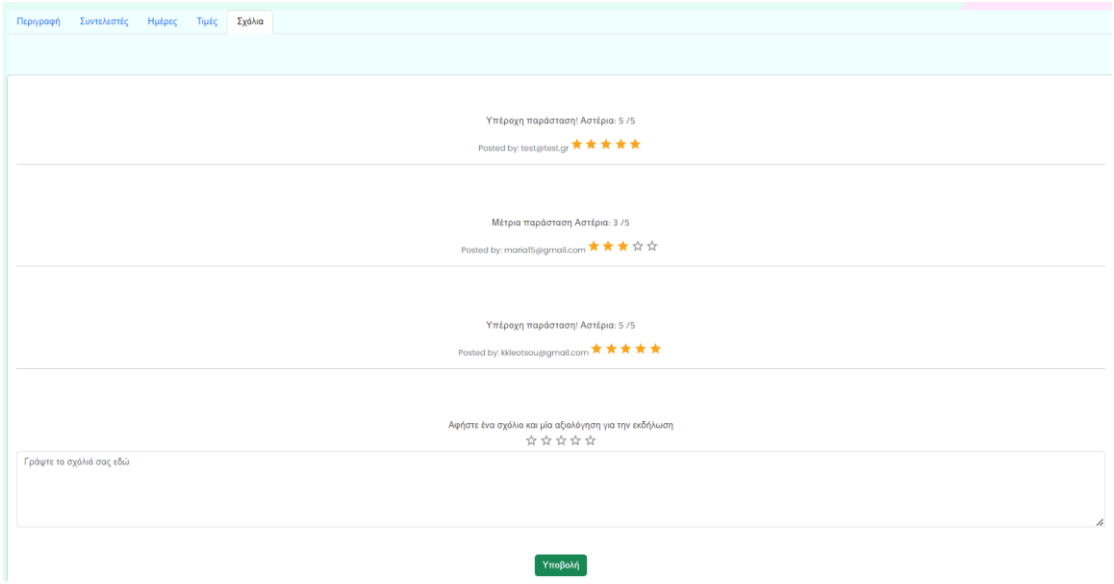


Εικόνα 25: Ημέρες και Ώρες Εκδήλωσης

Επιπροσθέτως, επιλέγοντας το tab Τιμές, μας εμφανίζονται οι τιμές με βάση την κατηγορία τους (Εικόνα 26).

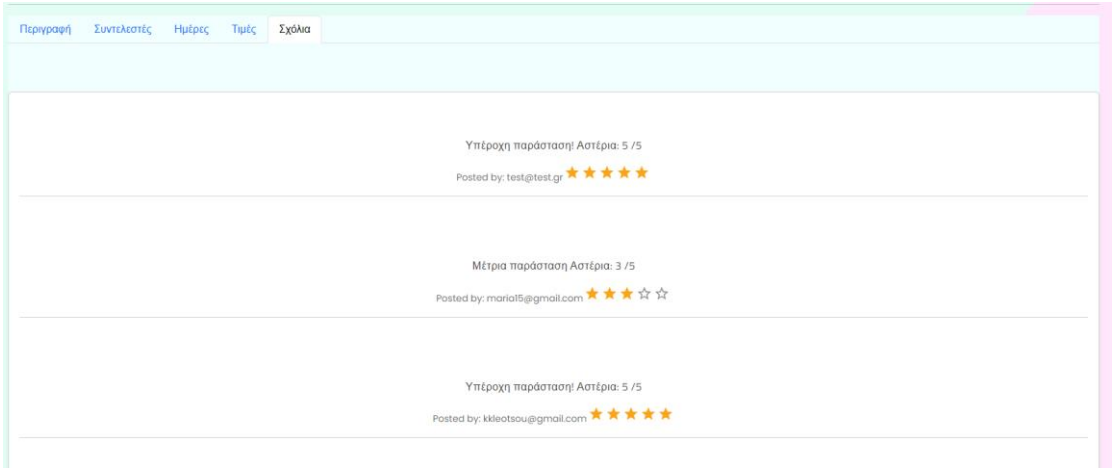
Εικόνα 26: Τιμές Εκδήλωσης

Τέλος, κάνοντας κλικ πάνω στην επιλογή Σχόλια, μας εμφανίζεται η παρακάτω φόρμα αξιολόγησης (Εικόνα 27).



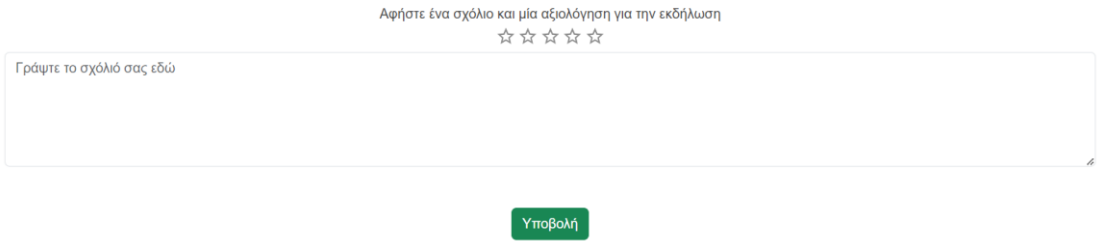
Εικόνα 27: Φόρμα Αξιολόγησης και Σχόλια Χρηστών

Πιο αναλυτικά, σχετικά με την φόρμα αξιολόγησης, μπορούμε να δούμε τα σχόλια όλων των χρηστών που έχουν υποβληθεί για την εκδήλωση που έχουμε επιλέξει. Για κάθε ένα σχόλιο μπορούμε να δούμε το κείμενο που έχει γράψει ο χρήστης, τα αστέρια καθώς και το όνομα του (Εικόνα 28).



Εικόνα 28: Σχόλια Χρηστών

Έπειτα, έχουμε την δυνατότητα να υποβάλλουμε νέο σχόλιο (Εικόνα 29).



Εικόνα 29: Φόρμα Υποβολής Αξιολόγησης

Αρχικά, καλούμαστε να συμπληρώσουμε τον αριθμό των αστεριών για να πραγματοποιηθεί η αξιολόγηση (Εικόνα 30).

Εικόνα 30: Μη Υποβληθέντα Αστέρια

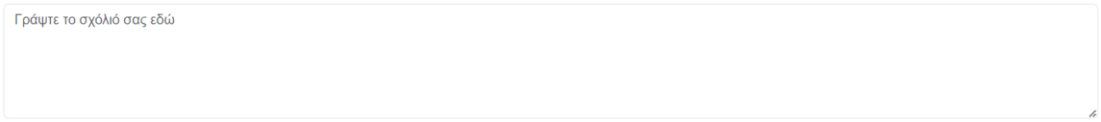
Κάνοντας κλικ πάνω στο αστέρι που επιθυμούμε συμπληρώνονται αυτόματα τα αστέρια που θα υποβάλλουμε. Παραδείγματος χάρη, αν θέλουμε να υποβάλλουμε τέσσερα αστέρια για την εκδήλωση, πατάμε πάνω στο τέταρτο αστέρι και αυτομάτως συμπληρώνονται και τα υπόλοιπα τρία (Εικόνα 31).

Εικόνα 31: Υποβληθέντα Αστέρια

Αφού λοιπόν επιλέξουμε τα αστέρια, κάτω από αυτά εμφανίζεται ένα μήνυμα με το πόσα αστέρια έχουμε δηλωθεί. Αν αλλάξουμε τον αριθμό των αστεριών, αλλάζει και το συγκεκριμένο μήνυμα (Εικόνα 32).

Εικόνα 32: Μήνυμα Υποβολής Αστεριών

Συνεχίζοντας, κάτω από την υποβολή των αστεριών (αξιολόγηση), εμφανίζεται ένα πλαίσιο κειμένου, όπου μπορούμε να συμπληρώσουμε διάφορα σχόλια που έχουμε να κάνουμε για την παρούσα εκδήλωση.

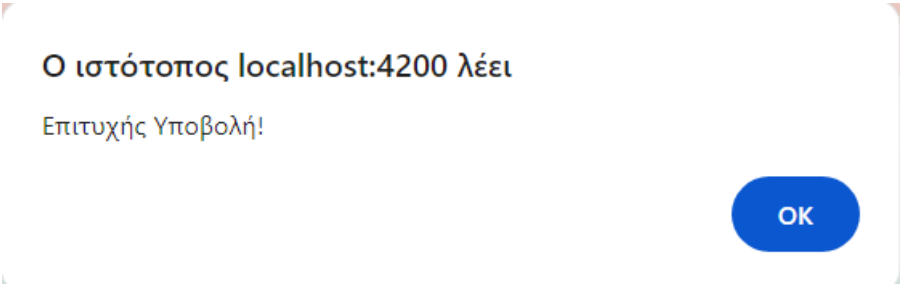


Εικόνα 33: Πλαίσιο Κειμένου

Έπειτα, αφού έχουμε συμπληρώσει το πλαίσιο κειμένου και τον αριθμό των αστεριών, πατώντας το κουμπί 

, μπορούμε πλέον να καταχωρήσουμε την αξιολόγηση μας για την εκδήλωση.

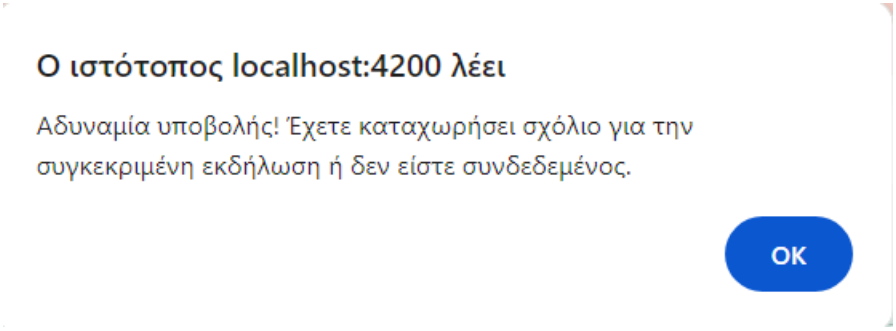
Δεδομένου ότι πατήσαμε το κουμπί της Υποβολής, εμφανίζεται το παρακάτω μήνυμα επιτυχής υποβολής αξιολόγησης/ σχόλιου (Εικόνα 34).



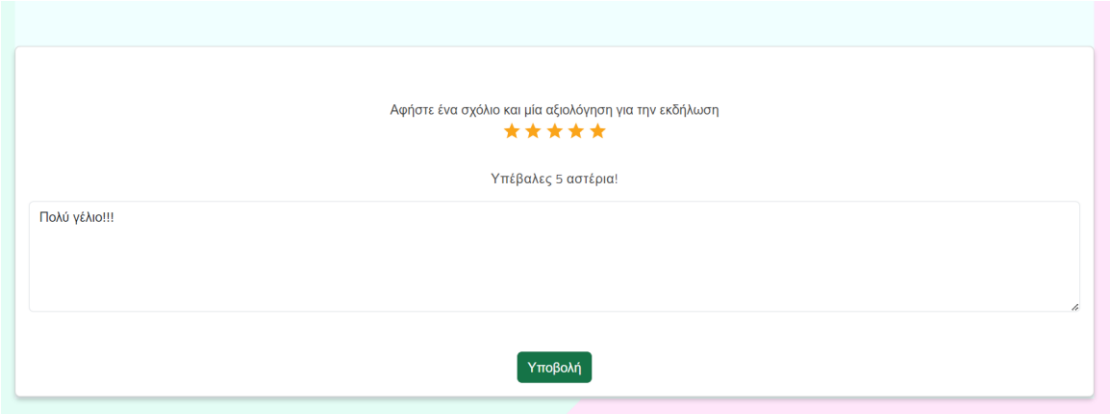
Εικόνα 34: Μήνυμα Επιτυχής Καταχώρησης Αξιολόγησης

Αντιθέτως, αν δεν έχουμε συμπληρώσει το πλαίσιο κειμένου ή δεν έχουμε επιλέξει τα αστέρια για να πραγματοποιηθεί η αξιολόγηση, εμφανίζεται το παρακάτω μήνυμα λάθους (Εικόνα 35).

Αξίζει να σημειωθεί πως αν δεν είμαστε συνδεδεμένοι στην ιστοσελίδα δεν μπορούμε να καταχωρήσουμε αξιολόγηση.

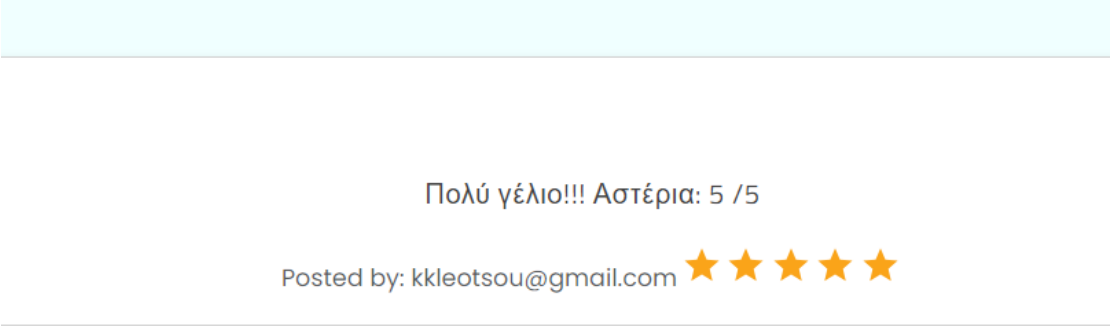


Εικόνα 35: Μήνυμα Λάθους για την Υποβολή Αξιολόγησης



Εικόνα 36: Συμπληρωμένη Φόρμα για Υποβολή Αξιολόγησης

Εφόσον ολοκληρώσουμε με επιτυχία την διαδικασία της υποβολής αξιολόγησης, πλέον το σχόλιο και τα αστέρια εμφανίζονται στην λίστα με όλα τα σχόλια, καθώς και αναγράφεται το όνομα μας (Εικόνα 37).

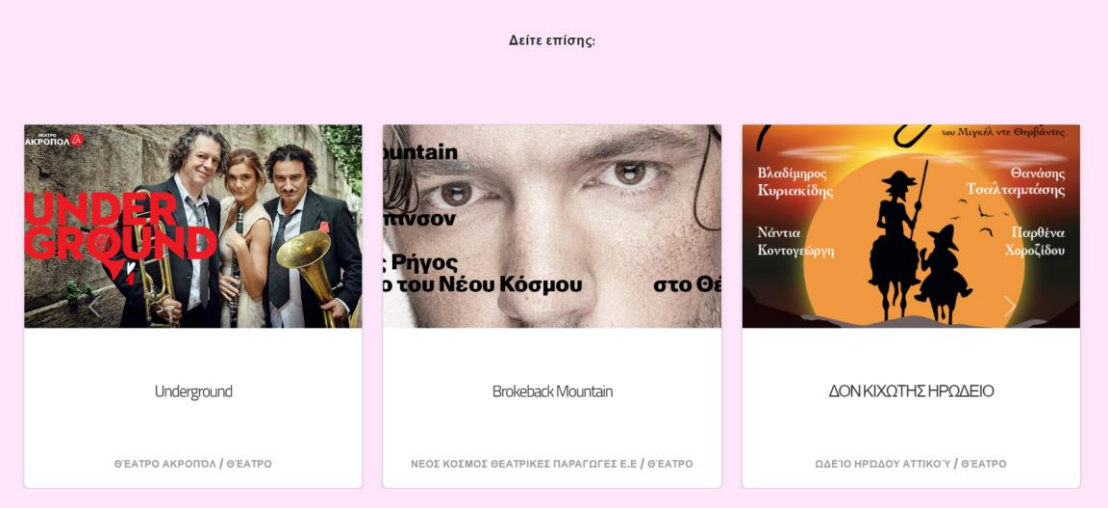


Εικόνα 37: Καταχώρηση Αξιολόγησης

Ύστερα, εμφανίζονται διάφορες εικόνες της εκδήλωσης, όπου πατώντας πάνω στα βελάκια μπορούμε να δούμε όλες τις εικόνες της παρούσας εκδήλωσης (Εικόνα 38).

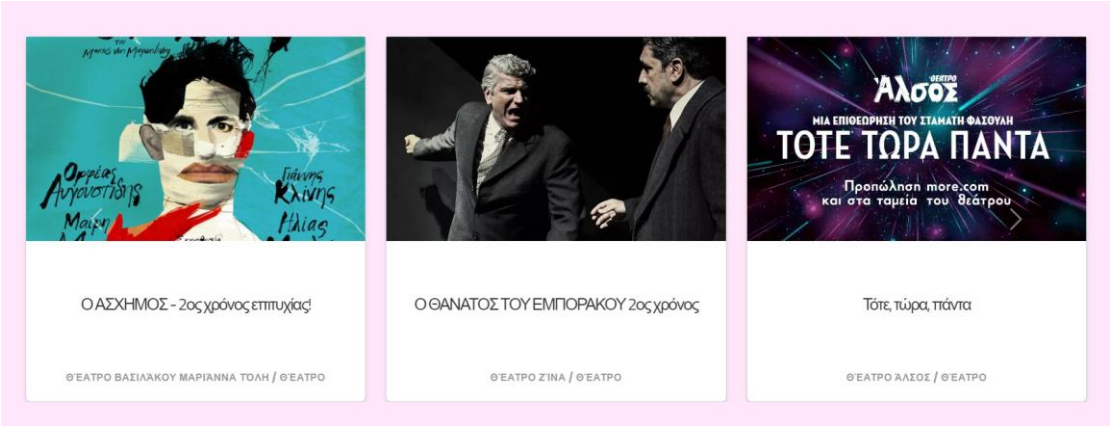
Εικόνα 38: Carousel Εικόνων

Ολοκληρώνοντας, προβάλλονται εξατομικευμένες προτάσεις σχετικά με την εκδήλωση που διαλέξαμε. Εξαιτίας του ότι οι σχετικές εκδηλώσεις είναι περισσότερες από τρεις, μας δίνεται η δυνατότητα πατώντας τα κουμπιά δεξιά και αριστερά μεταξύ των εκδηλώσεων, να δούμε όλες τις προτάσεις που μας συστήνονται (Εικόνα 4.39).



Εικόνα 39: Εξατομικευμένες Προτάσεις I

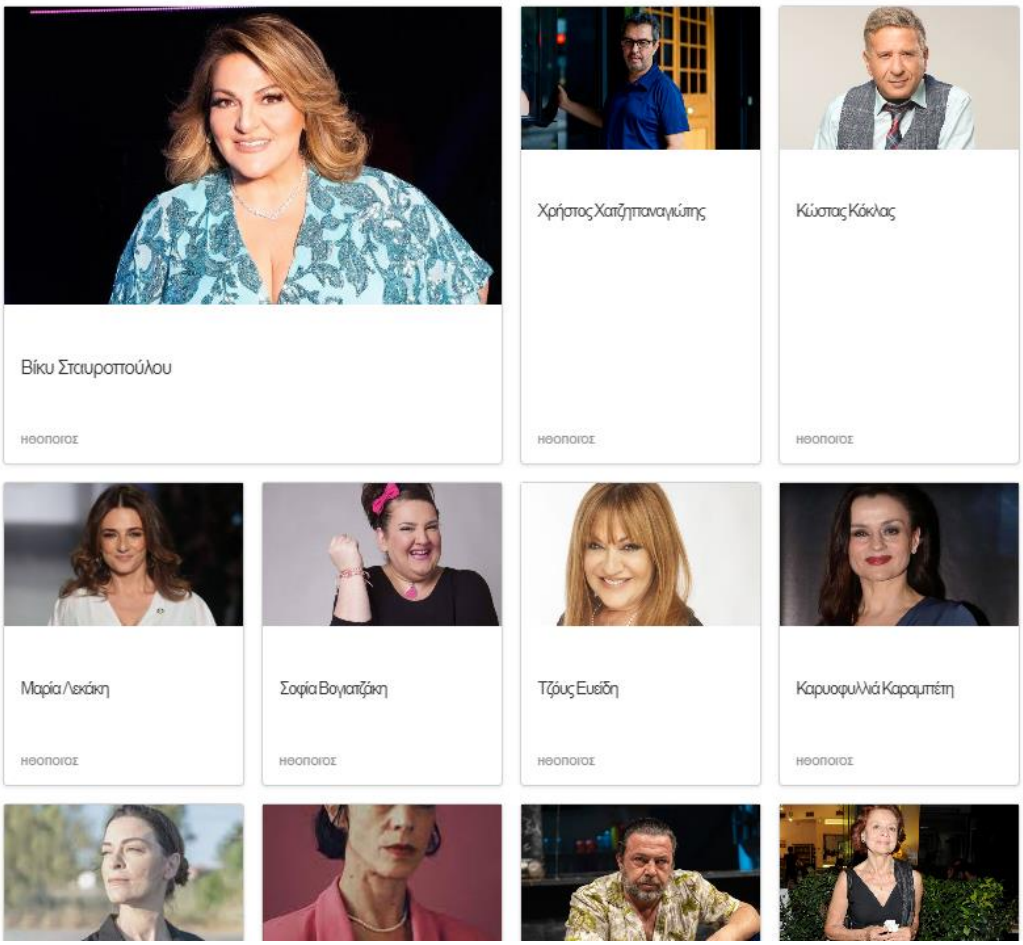
Πατώντας πάνω σε μία από αυτές τις προτάσεις, μεταφερόμαστε στην αντίστοιχη σελίδα τους, όπου εμφανίζεται η εκδήλωση που επιλέξαμε.



Εικόνα 40: Εξατομικευμένες Προτάσεις II

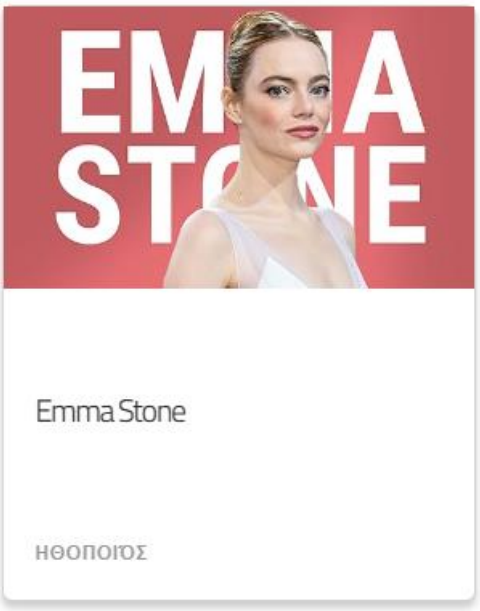
3.3.5 Επαγγελματίες

Η παρούσα σελίδα είναι παρόμοια με την σελίδα των εκδηλώσεων (βλ. Εκδηλώσεις 3.3.3). Πιο συγκεκριμένα, εμπεριέχει όλους τους επαγγελματίες ανάλογα την κατηγορία του επαγγέλματός τους που έχουμε επιλέξει από το μενού της ιστοσελίδας (Εικόνα 41).



Εικόνα 41: Επαγγελματίες

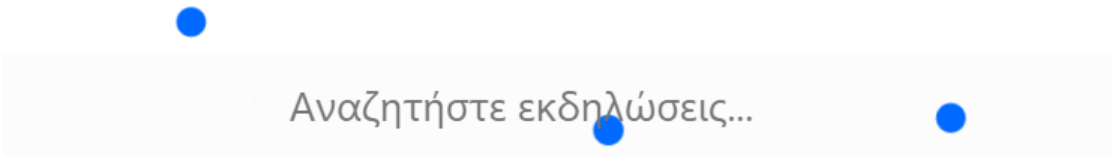
Κάθε επαγγελματίας εμφανίζεται σε μία κάρτα, όπου αναφέρονται διάφορες σύντομες πληροφορίες γι' αυτόν (Εικόνα 42). Πιο συγκεκριμένα, μία εικόνα του επαγγελματία, το όνομα και το επώνυμο του καθώς και σε ποιον κλάδο εργάζεται.



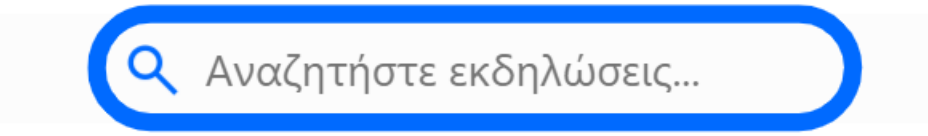
Εικόνα 42: Κάρτα Επαγγελματία

Πατώντας πάνω στην κάρτα ενός επαγγελματία, η σελίδα μας μεταφέρει σε μία άλλη όπου αναφέρονται όλες οι πληροφορίες σχετικά με αυτόν, π.χ. το βιογραφικό του. (Ενότητα 3.3.6).

Αξίζει να σημειωθεί πως στην αρχή της παρούσας σελίδας, εμφανίζεται μία μπάρα αναζήτησης όπου μπορούμε να αναζητήσουμε επαγγελματίες με βάση το όνομα τους. Καλούμαστε να συμπληρώσουμε τις πληροφορίες με βάση τις οποίες και θα αναζητήσουμε τον επαγγελματία, πατώντας στην συνέχεια το enter, όπου και μας εμφανίζει τα αποτελέσματα της αναζήτησης.

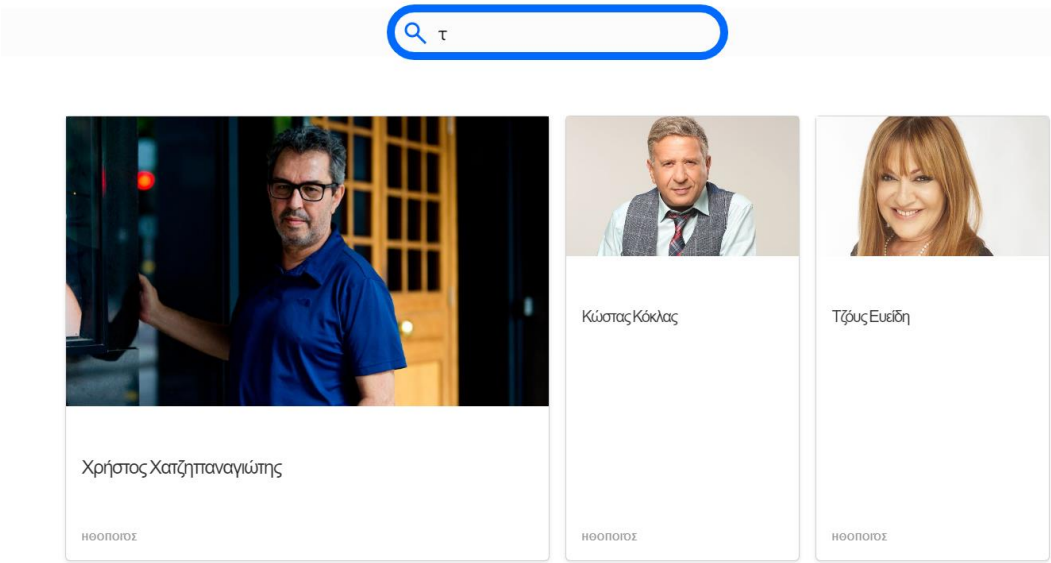


Εικόνα 43: Μπάρα Αναζήτησης Ι

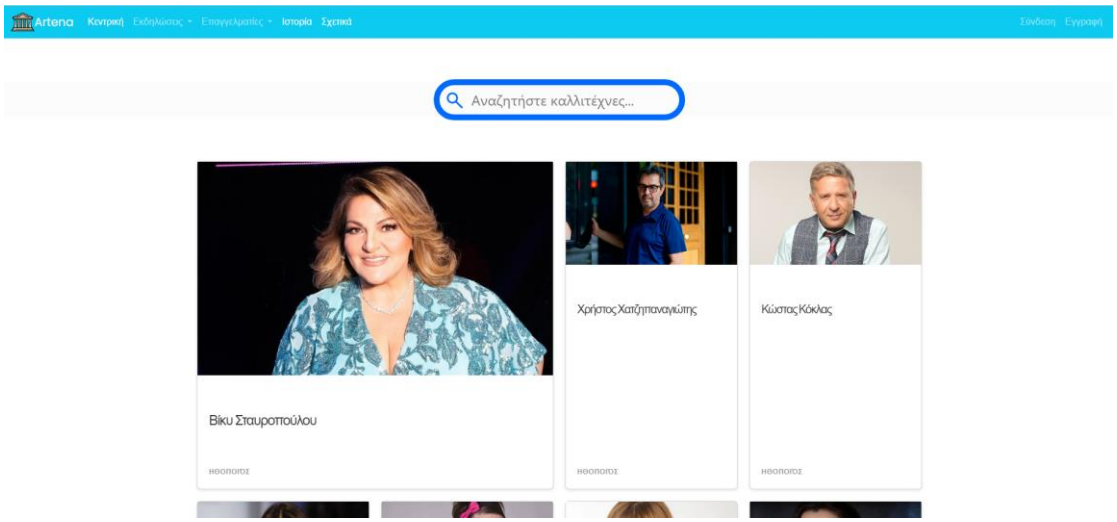


Εικόνα 44: Μπάρα Αναζήτησης II

Παραδείγματος χάρη, συμπληρώνοντας στην μπάρα της αναζήτησης μία λέξη, μας εμφανίζονται τα παρακάτω αποτελέσματα (Εικόνα 45).



Εικόνα 45: Παράδειγμα Αναζήτησης Επαγγελματία

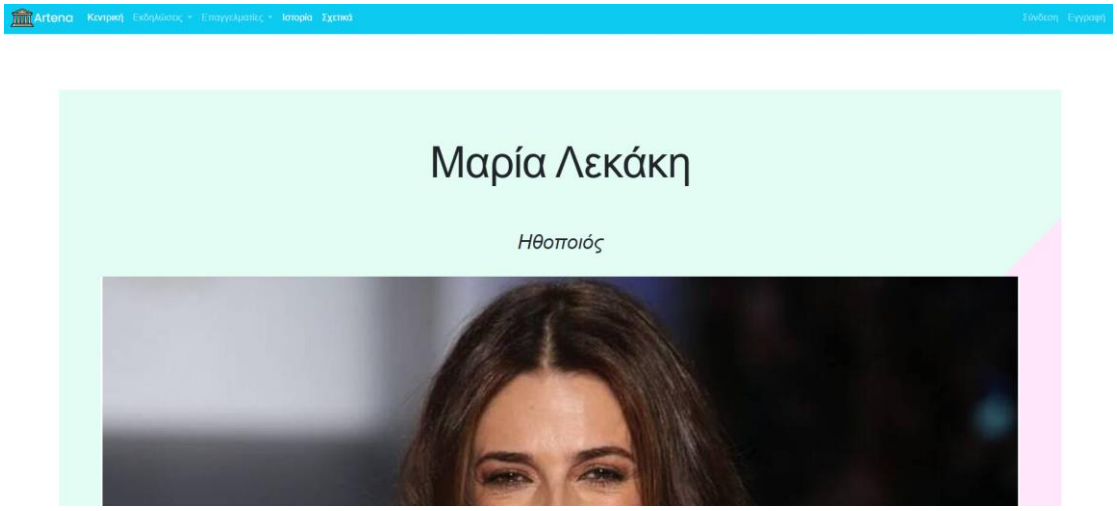


Εικόνα 46: Σελίδα Επαγγελματιών

3.3.6 Σελίδα Παρουσίασης Επαγγελματία

Με βάση το επάγγελμα και στη συνέχεια τον επαγγελματία που θα επιλέξουμε, θα μεταβούμε στην παρούσα σελίδα όπου γίνεται παρουσίαση του επαγγελματία.

Πιο συγκεκριμένα, αναγράφεται το όνομα και το επώνυμο του επαγγελματία, καθώς και το επάγγελμα με το οποίο ασχολείται και μία φωτογραφία του (Εικόνα 47).



Εικόνα 47: Σελίδα Παρουσίασης Επαγγελματία

Έπειτα, αναγράφεται το βιογραφικό του σημείωμα, διάφορες πληροφορίες για την καριέρα του κτλ. (Εικόνα 48).

Εικόνα 48: Βιογραφικό Επαγγελματία

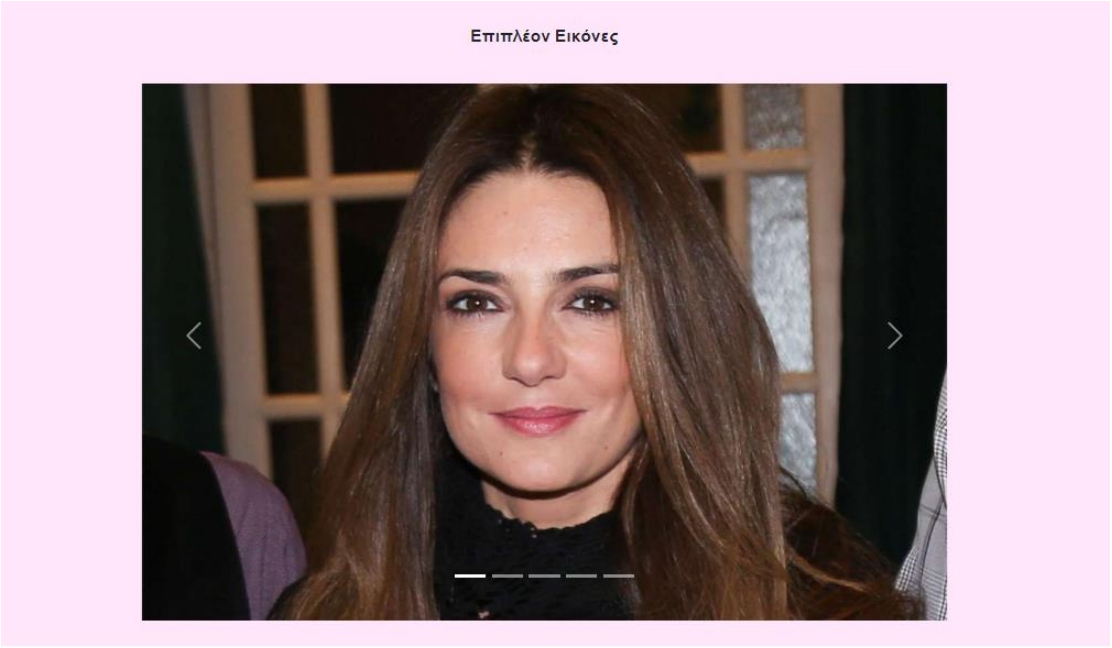
Αξίζει να σημειωθεί πως αν δεν έχουμε συνδεθεί στην ιστοσελίδα ή αν έχουμε συνδεθεί και το επάγγελμα μας δεν συνάδει με τα επαγγέλματα του χώρου του θεάματος, τότε δεν θα μας εμφανιστεί το email του επαγγελματία ώστε να επικοινωνήσουμε μαζί του (Εικόνα 49).

Εικόνα 49: Απόκρυψη Email

Αντιθέτως, αν είμαστε συνδεδεμένοι στην ιστοσελίδα και το επάγγελμά μας αφορά τον χώρο του θεάματος π.χ. ηθοποιός, σκηνοθέτης, χορευτής κλπ., εμφανίζεται το email του επαγγελματία που έχουμε επιλέξει (Εικόνα 50).

Εικόνα 50: Εμφάνιση Email Επαγγελματία αν ο χρήστης είναι επίσης επαγγελματίας

Στο τέλος της σελίδας, εμφανίζονται διάφορες εικόνες του επαγγελματία που έχουμε επιλέξει (Εικόνα 51).



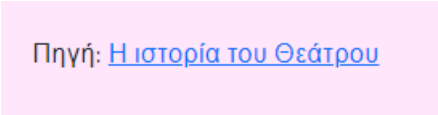
Εικόνα 51: Εικόνες Επαγγελματία με την χρήση Carousel

3.3.7 Ιστορία

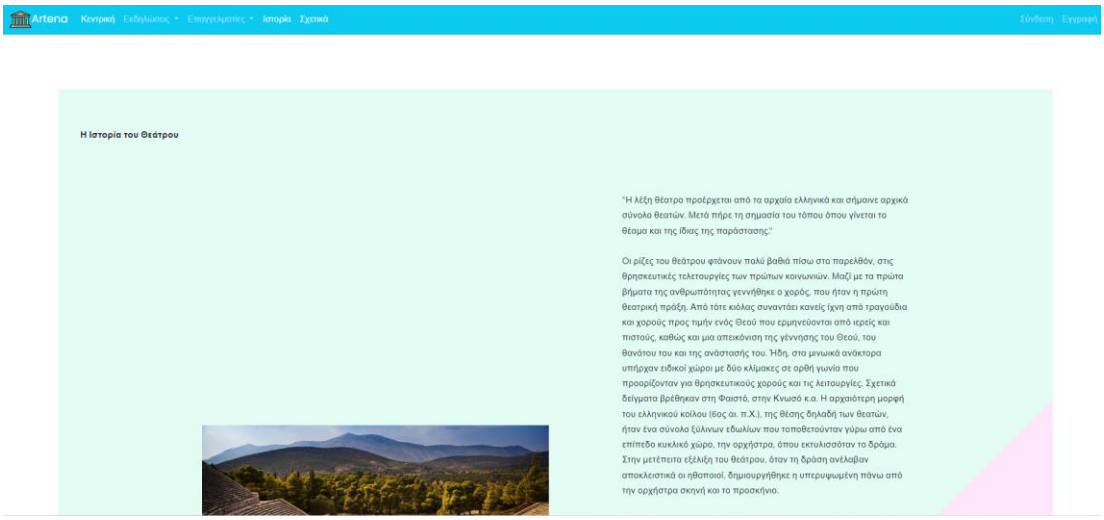
Πατώντας από το μενού την επιλογή **Σχετικά**, μεταβαίνουμε στην παρακάτω σελίδα (Εικόνα 53). Η παρούσα σελίδα είναι μία πληροφοριακή σελίδα.

Πιο συγκεκριμένα, αναφέρει διάφορες ιστορικές πληροφορίες σχετικά με την Ιστορία του Θεάτρου.

Να σημειωθεί, πως κάτω από τις πληροφορίες υπάρχει σχετικό link, όπου μπορούμε να μεταβούμε και να διαβάσουμε περισσότερες πληροφορίες σχετικά με την Ιστορία του Θεάτρου, καθώς και αποτελεί πηγή του κειμένου που αναγράφεται στην σελίδα.



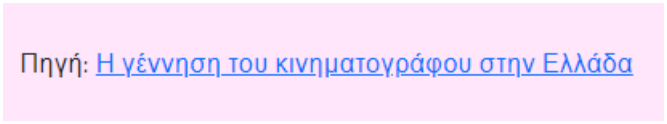
Εικόνα 52: Hyperlink I



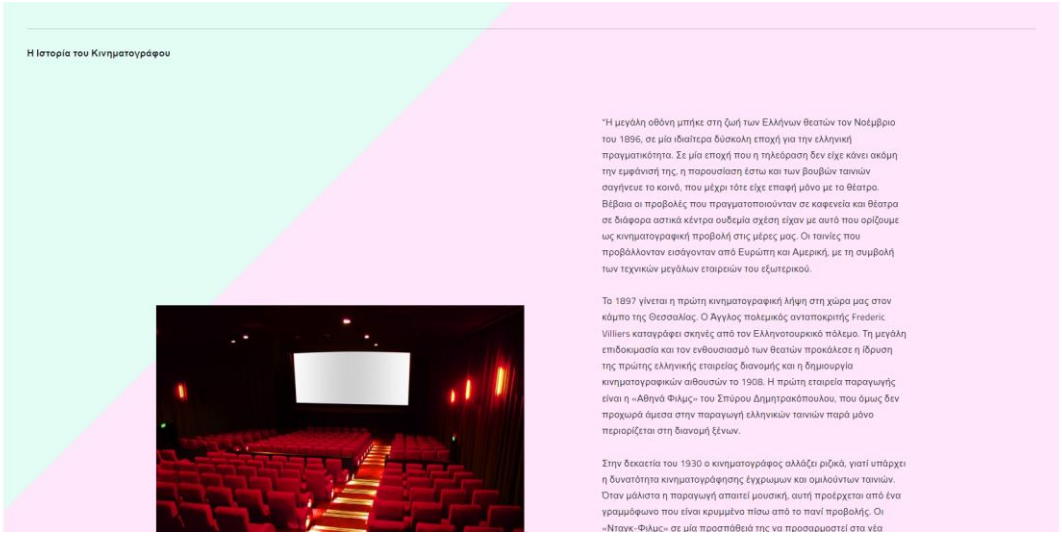
Εικόνα 53: Η Ιστορία του Θεάτρου

Καθώς και αναφέρει διάφορες ιστορικές πληροφορίες σχετικά με την Ιστορία του Κινηματογράφου.

Αξίζει να σημειωθεί, πως κάτω από τις πληροφορίες υπάρχει σχετικό link, όπου μπορούμε να μεταβούμε και να διαβάσουμε ολόκληρο το άρθρο σχετικά με την Ιστορία του Κινηματογράφου, καθώς και έχει αντληθεί από την σελίδα που αναγράφεται.



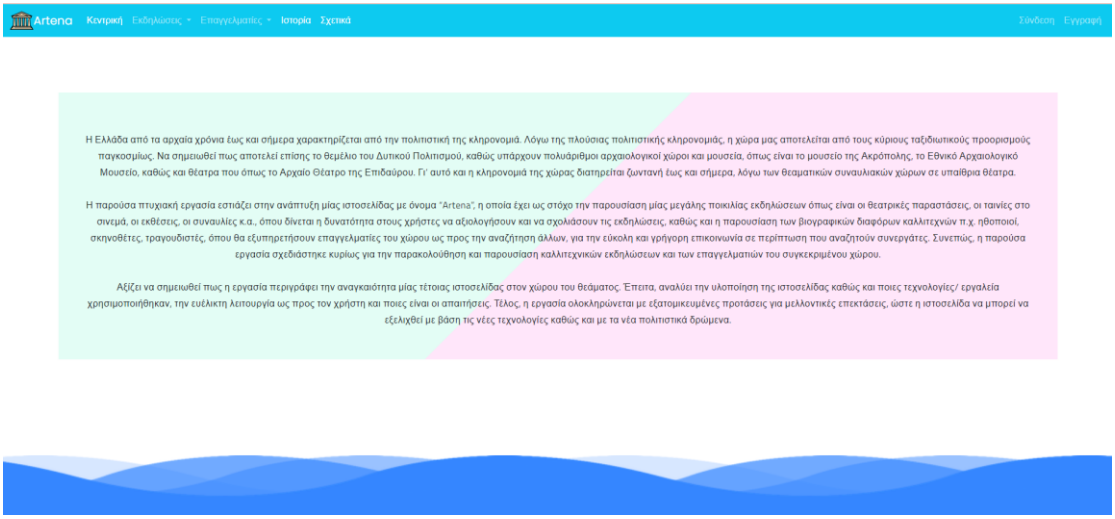
Εικόνα 54: Hyperlink II



Εικόνα 55: Η Ιστορία του Κινηματογράφου

3.3.8 Σχετικά

Αφού επιλέξουμε από το μενού την επιλογή **Σχετικά**, μεταβαίνουμε στην παρακάτω σελίδα. Στη συγκεκριμένη σελίδα, γίνεται μία παρουσίαση της εργασίας σχετικά με το αντικείμενο της.



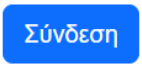
Εικόνα 56: Σελίδα Σχετικά

3.3.9 Σύνδεση

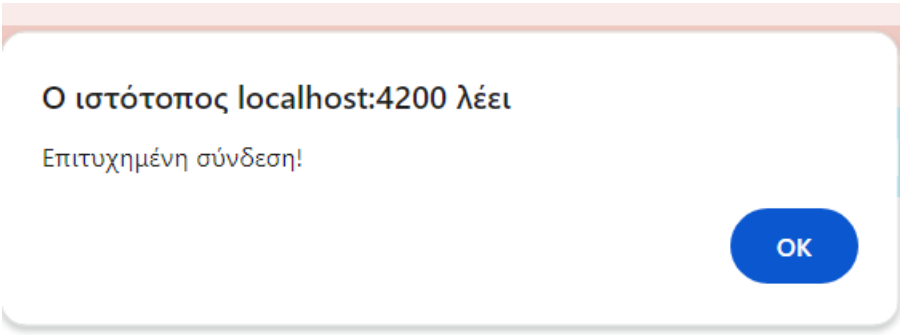
Η συγκεκριμένη σελίδα αφορά τη σύνδεση ενός χρήστη στην ιστοσελίδα. Θα αποτελείται από απαραίτητα πεδία για να μπορέσει να γίνει η σύνδεση όπως είναι το email και ο κωδικός.

Κάνοντας κλικ στην επιλογή **Σύνδεση** μεταβαίνουμε στην παρακάτω σελίδα (Εικόνα 59). Για την είσοδο στην εφαρμογή, απαραίτητη προϋπόθεση είναι να συμπληρώσουμε τα υποχρεωτικά πεδία. Πιο συγκεκριμένα το όνομα χρήστη, που είναι το email, σε αυτό το πεδίο

καθώς και τον κωδικό πρόσβασης που έχουμε δηλώσει στο αντίστοιχο πεδίο που φαίνεται στην ιστοσελίδα.

Έπειτα, αφού έχουμε συμπληρώσει τα παραπάνω στοιχεία με επιτυχία, πατώντας το κουμπί 

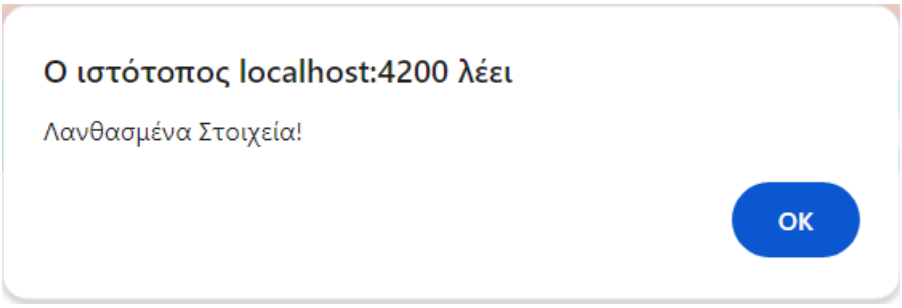
πραγματοποιούμε σύνδεση στην ιστοσελίδα, εμφανίζοντας το παρακάτω μήνυμα επιτυχημένης σύνδεσης (Εικόνα 57).



Εικόνα 57: Μήνυμα Επιτυχημένης Σύνδεσης

Αφού πατήσουμε το κουμπί OK, μεταβαίνουμε στην κεντρική σελίδα της εφαρμογής (Ενότητα 3.3.2).

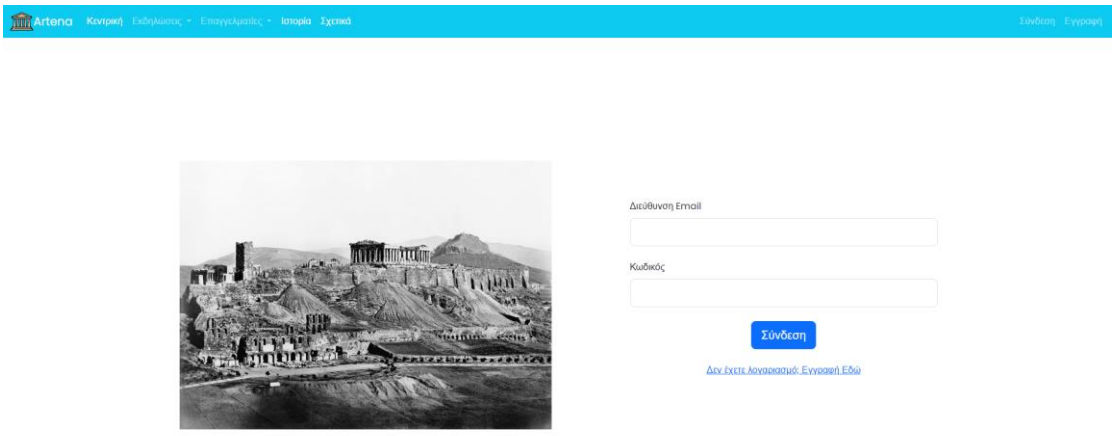
Αντιθέτως, σε περίπτωση όπου τα στοιχεία μας είναι λανθασμένα για να πραγματοποιήσουμε είσοδο στην εφαρμογή εμφανίζεται το παρακάτω μήνυμα λάθους (Εικόνα 58).



Εικόνα 58: Μήνυμα Λάθους Σύνδεσης

Τέλος, μας δίνεται η δυνατότητα μίας επιπλέον επιλογής. Σε περίπτωση όπου δεν έχουμε λογαριασμό πατώντας το hyperlink Δεν έχετε λογαριασμό; Εγγραφή Εδώ, μεταφερόμαστε στη σελίδα δημιουργίας νέου χρήστη (Ενότητα 3.3.10).

Εφόσον η σύνδεση στην ιστοσελίδα είναι επιτυχής, πλέον δεν εμφανίζονται δεξιά στο μενού οι επιλογές Σύνδεση, Εγγραφή αλλά η επιλογή Αποσύνδεση, όπου πατώντας την, αποσυνδεόμαστε από την ιστοσελίδα και πλέον μπορούμε να περιηγηθούμε ανώνυμα.



Εικόνα 59: Σελίδα Σύνδεσης

3.3.10 Εγγραφή

Η παρούσα σελίδα αφορά την εγγραφή ενός νέου χρήστη. Στη συγκεκριμένη περίπτωση, περιέχει διάφορα πεδία απαιτούμενα για την δημιουργία νέου χρήστη, όπου και θα πρέπει να συμπληρώσει. Έπειτα, για κάθε πεδίο θα γίνεται ο κατάλληλος έλεγχος για το αν έχουν συμπληρωθεί τα πεδία σωστά ή και αν δεν έχουν συμπληρωθεί.

Πιο αναλυτικά, κάνοντας κλικ από το μενού την επιλογή Εγγραφή, μεταβαίνουμε στην παρακάτω σελίδα εγγραφής νέου χρήστη (Εικόνα 69).

Για να ολοκληρωθεί επιτυχώς η εγγραφή, καλούμαστε να συμπληρώσουμε τα παρακάτω υποχρεωτικά πεδία όπως είναι το όνομα χρήστη (email), το email, ο κωδικός, το όνομα, το επώνυμο καθώς και η επιλογή του επαγγέλματος του όποιου κάνουμε.

Εικόνα 60: Όνομα Χρήστη

Εικόνα 61: Email

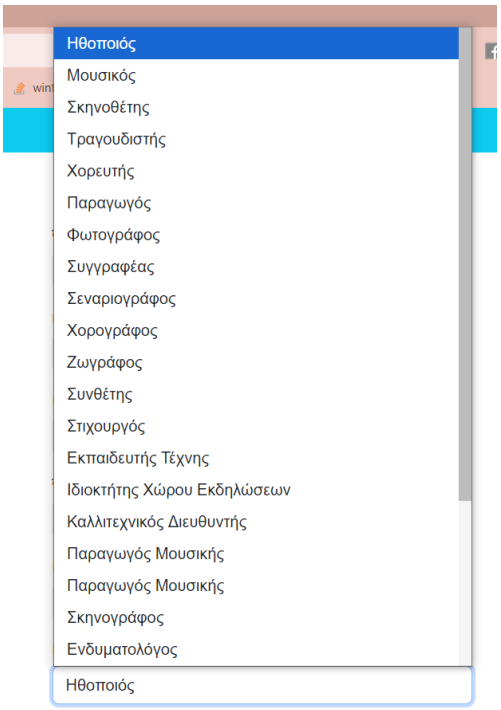
Εικόνα 62: Κωδικός

Εικόνα 63: Όνομα

Εικόνα 64: Επώνυμο

Για την επιλογή του επαγγέλματος, υπάρχουν προεπιλεγμένες εγγραφές τις οποίες μπορούμε να επιλέξουμε (έως 1), οι οποίες εμφανίζονται στο παρακάτω dropdown. (Εικόνα 3.3.6).

Εικόνα 65: Επάγγελμα

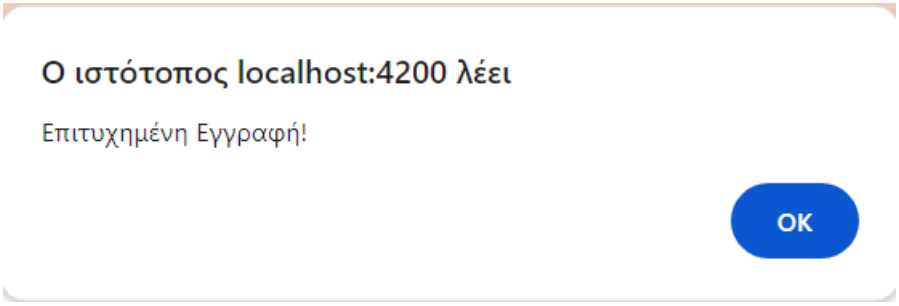


Εικόνα 66: Επιλογή Επαγγέλματος Χρήστη – Dropdown List

Αφού λοιπόν συμπληρώσουμε όλα τα προ απαιτούμενα στοιχεία της φόρμας, συνεχίζουμε πατώντας το κουμπί 

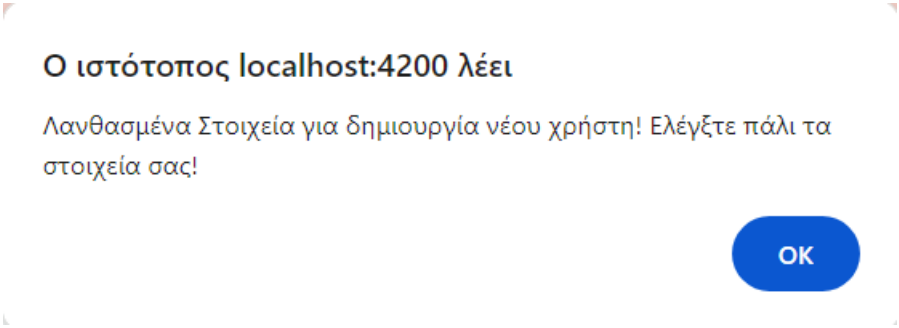
, για να δημιουργήσουμε τον χρήστη μας.

Εφόσον ο χρήστης δημιουργηθεί με επιτυχία εμφανίζεται το παρακάτω μήνυμα ενημέρωσης.

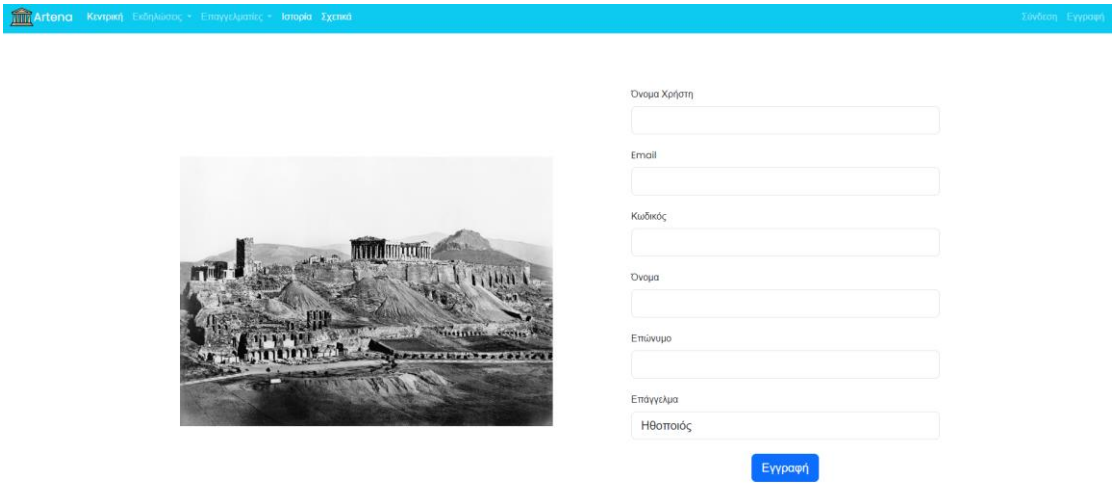


Εικόνα 67: Μήνυμα Επιτυχημένης Εγγραφής

Αντιθέτως, αν δεν έχουμε συμπληρώσει όλα τα πεδία ή δεν τα έχουμε συμπληρώσει σωστά, όπως το email να μην είναι της σωστής μορφής ή ο κωδικός να μην περιέχει παραπάνω από 8 ψηφία κ.α., εμφανίζεται το παρακάτω μήνυμα λάθους, όπου μας ενημερώνει πως τα στοιχεία εγγραφής είναι λάθος.



Εικόνα 68: Μήνυμα Λάθους Εγγραφής



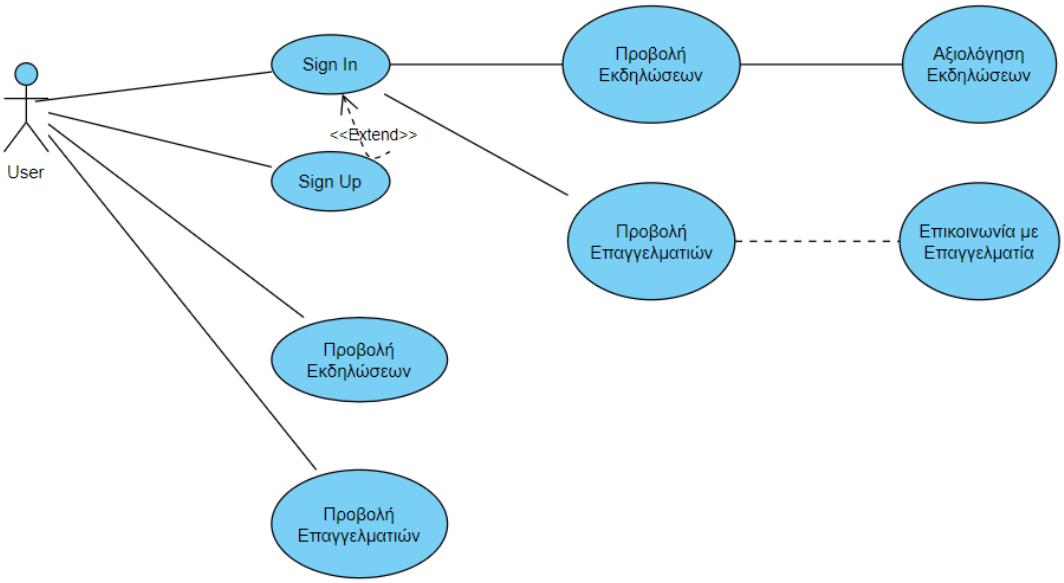
Εικόνα 69: Σελίδα Εγγραφής

Κεφάλαιο 4 - Αρχιτεκτονική Συστήματος

4.1 Εισαγωγή Αρχιτεκτονικής

Για την πλήρη υλοποίηση της εφαρμογής χρειάστηκαν δύο σημαντικά μέρη. Το πρώτο είναι το backend, όπου αποτελείται από κλήσεις οι οποίες παρέχουν διάφορες πληροφορίες χρήσιμες ως προς την εφαρμογή. Παραδείγματος χάρη, οι πληροφορίες των εκδηλώσεων και των επαγγελματιών. Το δεύτερο είναι το frontend, όπου πραγματοποιείται η αλληλεπίδραση χρήστη και εφαρμογής. Όσον αφορά το backend, αξίζει να σημειωθεί πως αντλεί τα δεδομένα από την Βάση Δεδομένων.

Για την ανάπτυξη της εφαρμογής, αρχικά χρειάστηκε να αναλύσουμε όλες τις απαιτήσεις της εφαρμογής, ώστε ο χρήστης να μπορεί να περιηγηθεί ανάλογα τις περιπτώσεις.

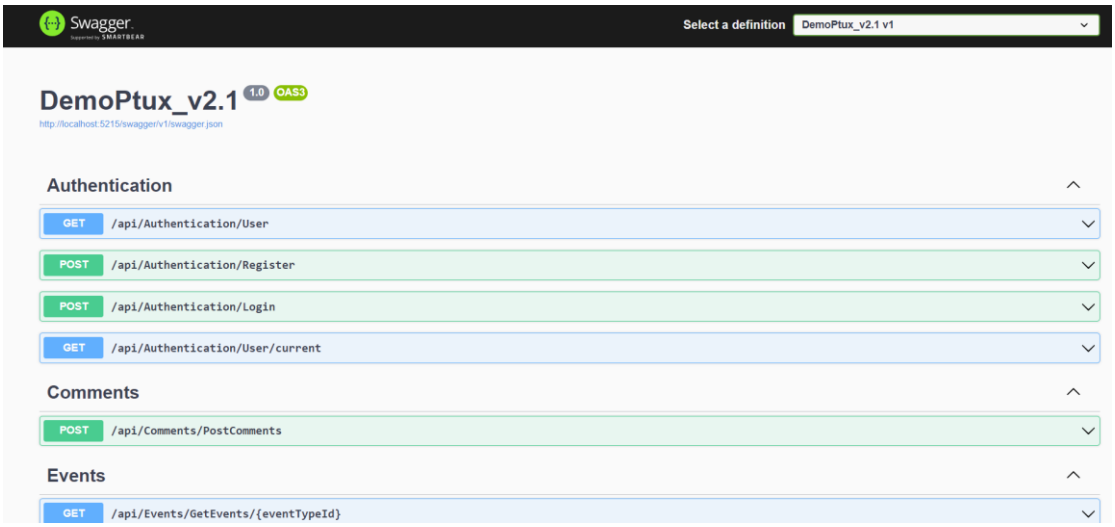


Διάγραμμα 1: Use Case Diagram

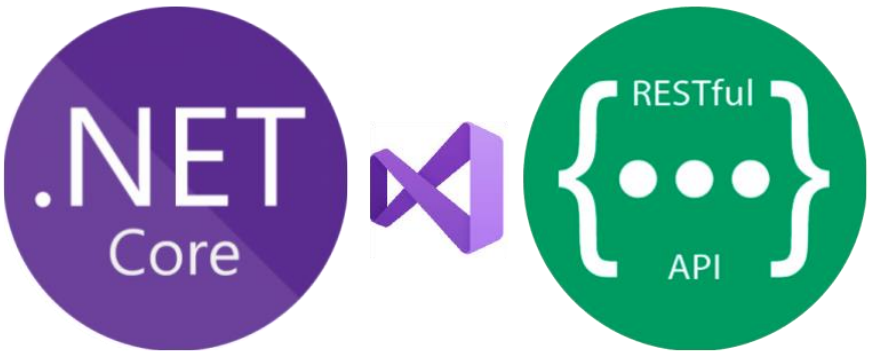
4.2 Ανάλυση backend

4.2.1 Τεχνολογία backend

Όσον αφορά το backend, η υλοποίηση του προγράμματος πραγματοποιήθηκε μέσω του Visual Studio. Το Visual Studio είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης IDE σχεδιασμένο για προγραμματιστές .NET και C++ σχετικά με τα λειτουργικά συστήματα των Windows. Το REST API που χρησιμοποιήθηκε από το Visual Studio είναι το ASP.NET Core Web API με τη χρήση Swagger. Το Swagger προσφέρει μία διεπαφή και δίνει την δυνατότητα κατανόησης ενός REST API, χωρίς να υπάρχει πρόσβαση στον κώδικα. Η διεπαφή είναι της μορφής (Εικόνα 70).



Εικόνα 70: Διεπαφή Swagger

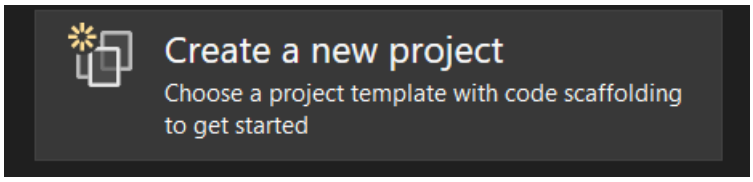


Εικόνα 71 : Visual Studio

4.2.2 Δημιουργία Project ASP.NET Core WEB API

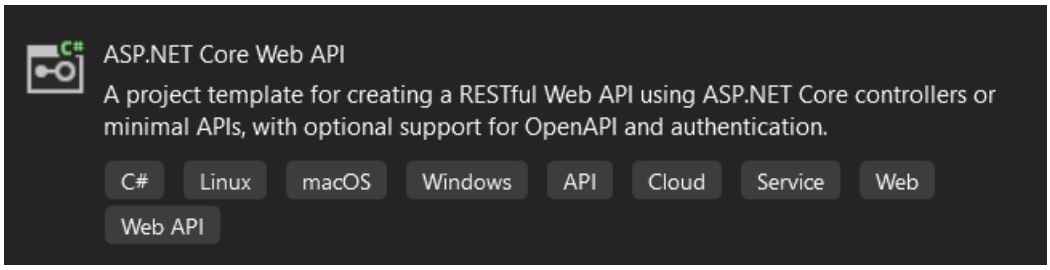
Η κατασκευή του προγράμματος πραγματοποιήθηκε ως εξής:

Εφόσον έχουμε ανοίξει το Visual Studio, επιλέγουμε την δημιουργία νέου project.



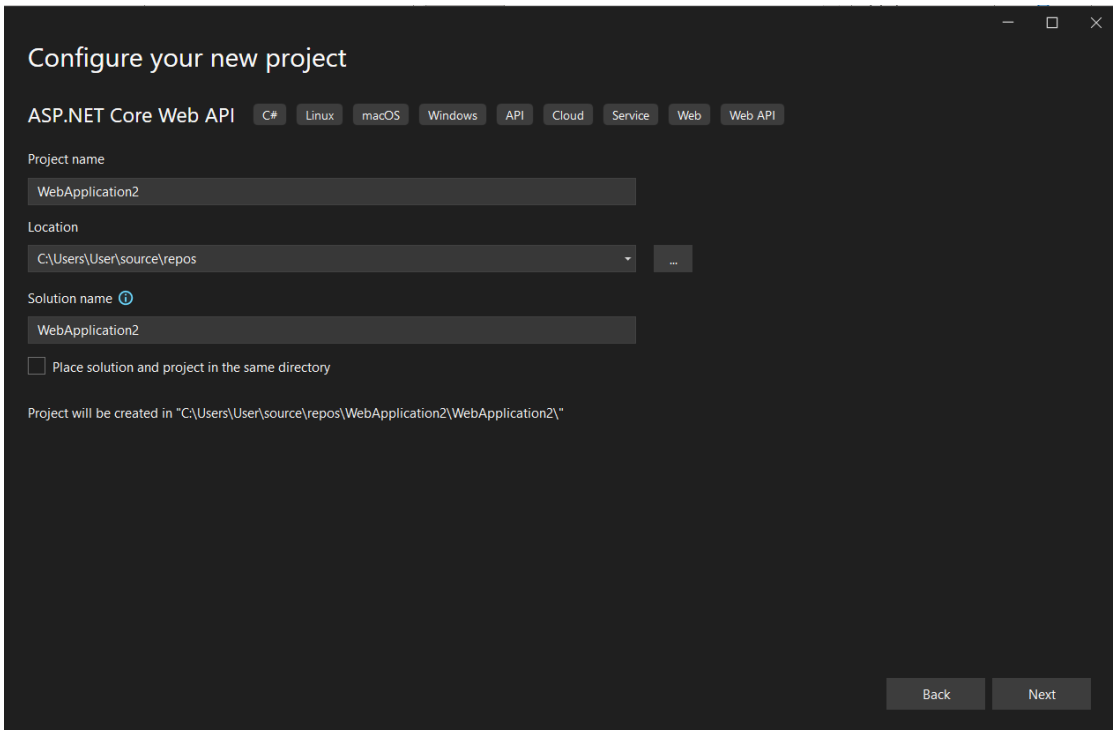
Εικόνα 72: Βήμα 1 Δημιουργία project

Έπειτα, επιλέγουμε το template ASP.NET Core Web API.



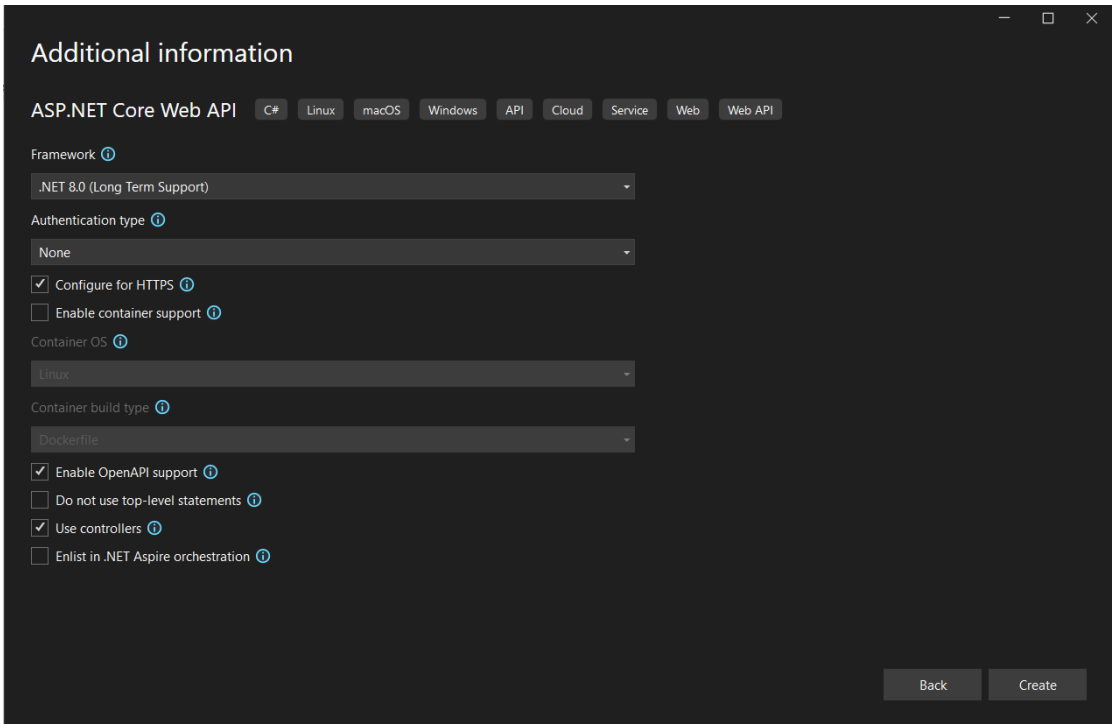
Εικόνα 73: Βήμα 2 Επιλογή template

Στη συνέχεια, συμπληρώνουμε το όνομα του project και την θέση που θα αποθηκευτεί.



Εικόνα 74: Βήμα 3 Ορισμός ονόματος και θέσης αρχείου

Ολοκληρώνουμε την διαδικασία με την επιλογή του framework και διάφορων επιπλέον επιλογών.



Εικόνα 75: Βήμα 4 Επιλογή ρυθμίσεων

4.2.3 Υλοποίηση

Το πρόγραμμα υλοποιήθηκε μέσω της χρήσης του Visual Studio, όμως χρειάστηκαν και επιπλέον εργαλεία. Πιο συγκεκριμένα, χρησιμοποιήθηκε το ASP.NET Core 7.0, το οποίο συμβάλλει στην υλοποίηση του Web Api.

Έπειτα, εγκαταστάθηκαν χρήσιμα packages για την εφαρμογή, όπως το EF Core. Το EF Core, δίνει την δυνατότητα να εργαστούμε σε μία βάση δεδομένων χρησιμοποιώντας αντικείμενα .NET. Συνεπώς, αξιοποιήθηκε για την επικοινωνία με την βάση δεδομένων (Ενότητα 4.4). Η σύνδεση για την επικοινωνία ορίστηκε στο πρόγραμμα εκκίνησης της εφαρμογής Program.cs, όπου και προστέθηκε το ConnectionString της βάσης, με την βοήθεια του json αρχείου appsetings.json.

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=DESKTOP-NMA4KVG;Database=DbPtuxiakis;Trusted_Connection=True;Integrated Security=true;MultipleA
  },
```

Εικόνα 76: Connection String

```
// For Entity Framework
builder.Services.AddDbContext<ApplicationDbContext>(options =>
options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));
```

Εικόνα 77: Σύνδεση Βάσης με Visual Studio

Ένα σημαντικό μέρος της εφαρμογής είναι η αυθεντικοποίηση και η εξουσιοδότηση του χρήστη σε αυτή, η οποία και πραγματοποιήθηκε μέσω ενός συστήματος διαχείρισης ταυτότητας και αυθεντικοποίησης, το ASP.NET Core Identity. Το ASP.NET Core Identity είναι ένα API κατά το οποίο εκτελείται η σύνδεση διεπαφής με τον χρήστη (UI). Πιο συγκεκριμένα, για να μπορέσουν οι χρήστες να συνδεθούν και να δημιουργήσουν λογαριασμό, η διαχείριση των κωδικών πρόσβασης, των δεδομένων κλπ. πραγματοποιείται μέσω αυτού. Τέλος για να εκτελεστούν οι παραπάνω λειτουργίες χρησιμοποιεί διάφορες μεθόδους όπως η CheckPasswordAsync και η FindByNameAsync.

Για την προσθήκη λοιπόν του συγκεκριμένου συστήματος, κατεβάσαμε διάφορα NuGet packages όπως είναι το Microsoft.AspNetCore.Identity.EntityFrameworkCore, Microsoft.EntityFrameworkCore.SqlServer. Έπειτα, προσθέσαμε την κλάση ApplicationDbContext, η οποία κληρονομεί από το IdentityDbContext. Στην παρούσα κλάση προστέθηκαν και οι επιπλέον κλάσης που χρησιμοποιούμε στην βάση δεδομένων μας, όπως οι Events, Comments κλπ. Για να πραγματοποιηθεί η ενεργοποίηση του Identity API, καλέσαμε την κλάση ApplicationDbContext στο πρόγραμμα εκκίνησης της εφαρμογής (Program.cs). Για την προσθήκη και αλλαγή δεδομένων στην βάση μέσω της εφαρμογής χρησιμοποιήθηκαν οι εντολές add-migration nameOfMigration και η Update-Database.

Ένα επίσης πολύ σημαντικό package είναι το Microsoft.IdentityModel.Tokens.Jwt. Το JwtSecurityToken είναι ένα security token το οποίο έχει σχεδιαστεί για να αναπαριστά ένα Json Web Token.

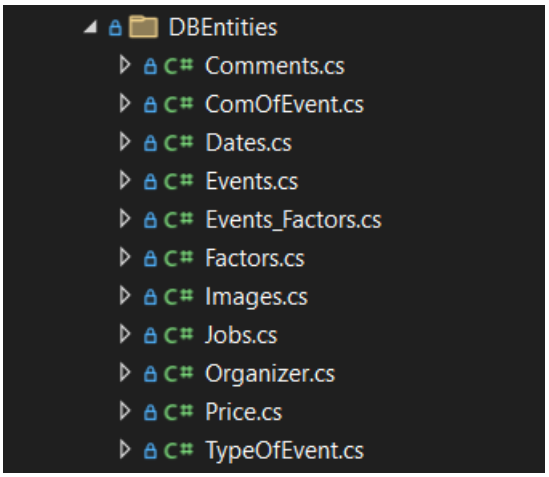
Για την εξαγωγή του Token, χρειάστηκε να ακολουθήσουμε τις παρακάτω διαδικασίες. Αρχικά, ορίσαμε στο αρχείο json της εφαρμογής (appsettings.json) τις παραμέτρους ValidAudience (ορίζεται ο αποδέκτης), ValidIssuer (ορίζεται ο εκδότης), SecurityKey (ορίζεται το κλειδί ασφαλείας για την διεξαγωγή του token). Και τέλος, προστέθηκε η διαδικασία επικύρωσης των tokens στο πρόγραμμα εκκίνησης της εφαρμογής.

Η εφαρμογή για το backend αποτελείται από διάφορα αρχεία, controllers οι οποίοι περιέχουν συναρτήσεις για την διεξαγωγή των κλήσεων και βοηθητικές κλάσεις. Το καθένα ξεχωριστά συμβάλλει στην ομαλή λειτουργία της backend εφαρμογής μας.

Models

Φάκελος DBEntities

Εφόσον, συνδέθηκε η βάση δεδομένων, δημιουργήθηκαν τα αντίστοιχα Data Models. Η εφαρμογή μέσα στο φάκελο των Models, περιέχει ένα επιπλέον, τον DBEntities, όπου περιλαμβάνει τις κλάσεις που συσχετίζονται με τους πίνακες της βάσης δεδομένων.



Εικόνα 78: Κλάσεις που αντικατοπτρίζουν τα αντικείμενα της βάσης

Comments: Η κλάση Comments περιλαμβάνει τα σχόλια που έχουν υποβληθεί από τον κάθε χρήστη.

ComOfEvent: Η παρούσα κλάση λειτουργεί ως διασύνδεση της κλάσης Comments και της κλάσης Events, καθώς περιέχει τα id και των 2 ώστε να διαχωρίζονται τα σχόλια κάθε εκδήλωσης.

Dates: Περιλαμβάνει τις ημέρες και ώρες που διεξάγεται μία εκδήλωση.

Events: Περιλαμβάνει σημαντικές πληροφορίες των εκδηλώσεων, όπως είναι ο τίτλος, η περιγραφή κ.α. καθώς και κρατάει ως πληροφορία τα id των υπόλοιπων κλάσεων, δηλαδή των Organizer, Images, Prices, Dates, TypeOfEvent.

EventsFactors: Η παρούσα κλάση λειτουργεί ως διασύνδεση της κλάσης Factors και Events, καθώς περιέχει τα id και των 2 ώστε να διαχωρίζονται οι συντελεστές μίας εκδήλωσης διότι πολύ πιθανόν είναι ένας καλλιτέχνης να συμπεριλαμβάνεται σε περισσότερες από μία εκδηλώσεις.

Factors: Η κλάση αυτή περιέχει τους επαγγελματίες που εργάζονται στον χώρο του θεάματος και διάφορες σημαντικές πληροφορίες γι’ αυτούς.

Images: Εμπεριέχονται τα μονοπάτια των εικόνων κάθε εκδήλωσης και κάθε επαγγελματία.

Jobs: Η κλάση Jobs αντιπροσωπεύει τα δεδομένα των επαγγελμάτων που είναι αποθηκευμένα στην βάση.

Organizer: Αντιπροσωπεύει τα δεδομένα των διοργανωτών κάθε εκδήλωσης.

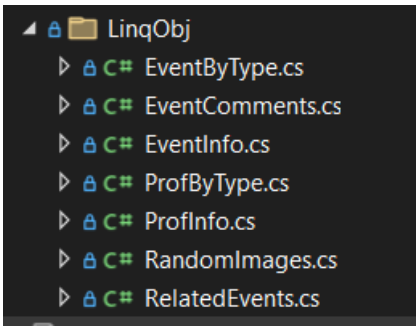
Price: Μέσω της συγκεκριμένης κλάσης αντιπροσωπεύονται οι τιμές κάθε εκδήλωσης.

TypeOfEvent: Κλάση η οποία αντιπροσωπεύει τα δεδομένα του τύπου κάθε event, αν δηλαδή είναι θέατρο, κινηματογράφος κ.α.

Διάγραμμα 2: Διάγραμμα Κλάσεων UML (Φάκελος DBEntities)

Φάκελος LinqObj

Ομοίως, στον φάκελο του Models, υπάρχει ένας ακόμη ο LinqObj. Ο συγκεκριμένος φάκελος περιλαμβάνει κλάσεις οι οποίες χρησιμοποιούνται για την διεξαγωγή αντικειμένων που αφορούν τις κλήσεις της εφαρμογής.



Εικόνα 79: Αντικείμενα διεξαγωγής API Calls

EventByType: Η παρούσα κλάση συγχωνεύει συγκεκριμένες πληροφορίες μεταξύ των κλάσεων Events, TypeOfEvent, Image, Organizer, ώστε να επιστρέφει συνοπτικές πληροφορίες κάθε εκδήλωσης, για την προβολή μαζικών εκδηλώσεων.

EventComments: Η κλάση EventComments χρησιμοποιείται για τα σχόλια κάθε εκδήλωσης.

EventInfo: Περιέχει όλες τις πληροφορίες αναλυτικά μιας συγκεκριμένης εκδήλωσης. Δηλαδή, όλα τα δεδομένα από τις κλάσεις Images, Dates, Price, Factors, Comments.

ProfByType: Είναι παρόμοιου τύπου με την EventByType, μόνο που εδώ συγχωνεύονται συγκεκριμένες πληροφορίες σχετικά με τους επαγγελματίες.

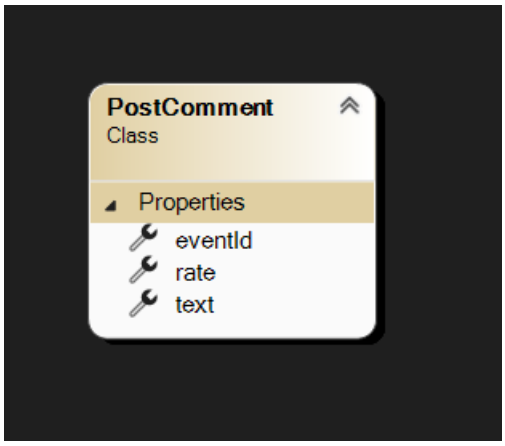
ProfInfo: Επίσης παρόμοιου τύπου με την κλάση EventInfo, μόνο που στη συγκεκριμένη περίπτωση συγχωνεύονται αναλυτικά όλες οι πληροφορίες που αφορούν τον επαγγελματία.

RandomImages: Η παρούσα κλάση χρησιμοποιήθηκε με σκοπό να αποθηκεύει σε αντικείμενο το id της εκδήλωσης, το id της αντίστοιχης εικόνας και το μονοπάτι της εικόνας.

RelatedEvents: Περιέχει ένα αντικείμενο EventInfo για την παρουσίαση μίας εκδήλωσης, μία λίστα της κλάσης EventByType η οποία χρησιμοποιείται για τις σχετικές παραστάσεις και τέλος μία λίστα της κλάσης EventComments όπου περιέχει όλα τα σχόλια την παράστασης που παρουσιάζεται.

Φάκελος Comments

Έπειτα, στον φάκελο του Models, υπάρχει ένας ακόμη ο Comments. Ο συγκεκριμένος φάκελος περιλαμβάνει μία μόνο κλάση την **PostComments**, η οποία χρησιμοποιείται για την υποβολή ενός νέου σχόλιου σε εκδήλωση από έναν χρήστη.



Διάγραμμα 3: Διάγραμμα Κλάσεων UML (Φάκελος Comments)

Φάκελος Authentication

Ο τελευταίος φάκελος των Models είναι ο Authentication, όπου στον συγκεκριμένο υπάρχουν κλάσεις οι οποίες χρησιμοποιούνται για την αυθεντικοποίηση του χρήστη στην εφαρμογή. Όπως είναι η εγγραφή νέου χρήστη, η σύνδεση του καθώς και η ενεργή παραμονή του στην εφαρμογή.

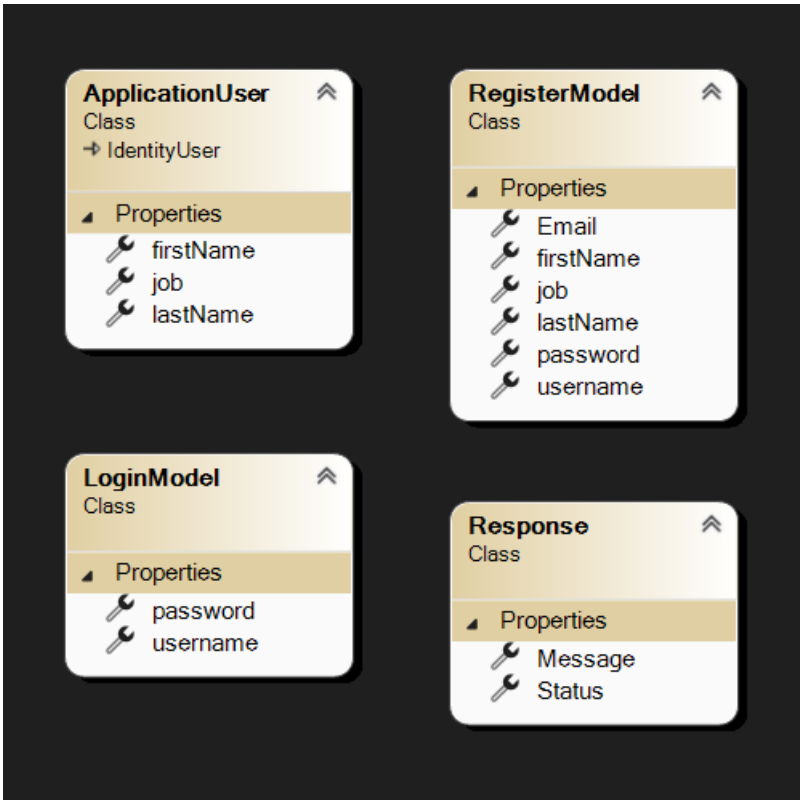
ApplicationUser: Η συγκεκριμένη κλάση χρησιμοποιήθηκε για την προσθήκη επιπλέον στοιχείων του χρήστη που δεν υπάρχουν στην βάση, καθώς η μέθοδος

σύνδεσης που χρησιμοποιήθηκε αποτελείται από ένα σύστημα διαχείρισης ταυτότητας (Identity Management System).

LoginModel: Η κλάση LoginModel περιλαμβάνει τα στοιχεία σύνδεσης του χρήστη, username και password.

RegisterModel: Σε αυτή τη κλάση περιέχονται τα απαραίτητα στοιχεία εγγραφής νέου χρήστη.

Response: Η παρούσα κλάση περιλαμβάνει το σώμα των απαντήσεων που χρησιμοποιήθηκε, όπως είναι ένα status και ένα μήνυμα.



Διάγραμμα 4: Διάγραμμα Κλάσεων UML (Φάκελος Authentication)

Controllers

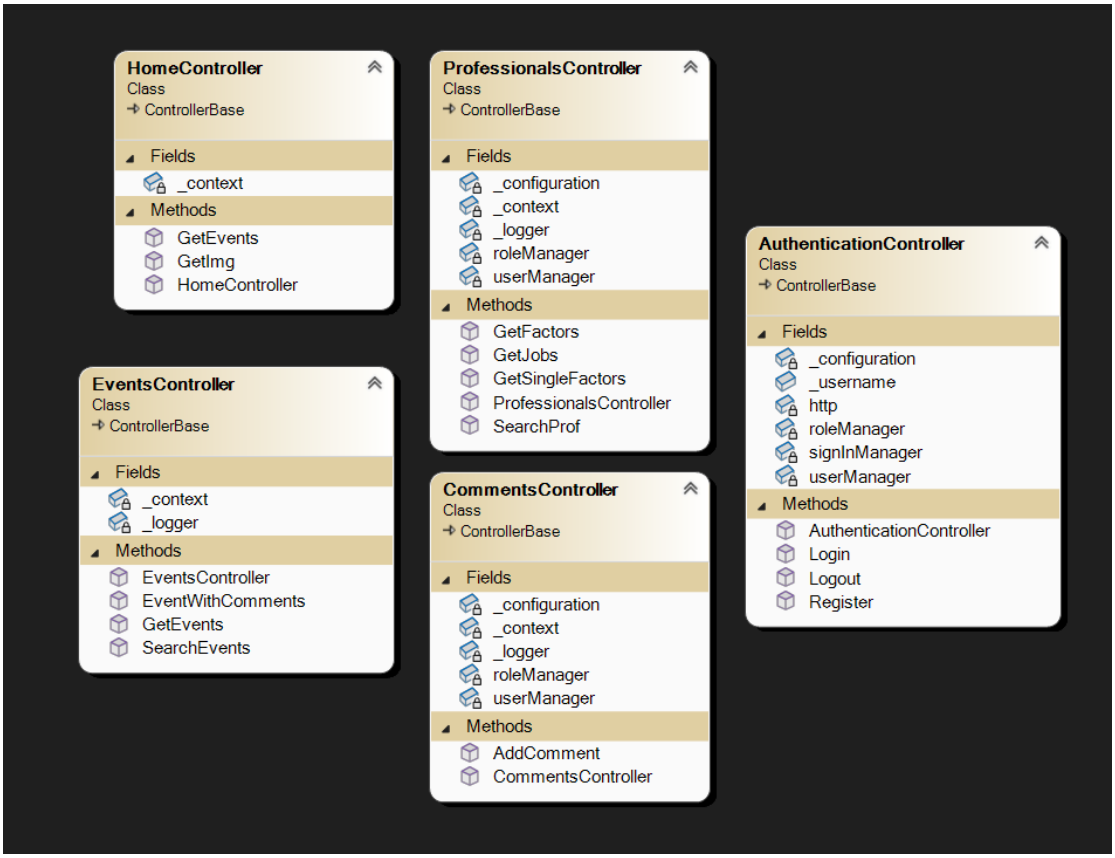
Για να πραγματοποιηθεί η διεπαφή πελάτη με διακομιστή (client - server), υλοποιήθηκα οι Controllers, οι οποίοι περιλαμβάνουν μεθόδους του πρωτοκόλλου HTTP, μέσω των οποίων πραγματοποιείται η διεξαγωγή των δεδομένων. Οι βασικές μέθοδοι πρωτοκόλλου HTTP είναι οι ακόλουθες:

1. GET: Η μέθοδος GET, επιστρέφει τα δεδομένα της βάσης που ζητήθηκαν
2. POST: Με τη μέθοδο POST, πραγματοποιείται μία νέα καταχώρηση στη βάση δεδομένων
3. PUT: Με τη μέθοδο PUT, πραγματοποιείται μία ενημέρωση στη βάση δεδομένων
4. DELETE: Ενώ η μέθοδος DELETE, πραγματοποιεί διαγραφή δεδομένων.

Ωστόσο, στη συγκεκριμένη εφαρμογή χρειάστηκαν οι δύο από τις τέσσερις, δηλαδή η μέθοδος GET και η μέθοδος POST.

Έπειτα, για να πραγματοποιηθεί η επικοινωνία και να επιστραφούν τα δεδομένα στον client χρειαζόμαστε URL. Το URL πρέπει να αποτελείται από το πρωτόκολλο HTTP, το domain name καθώς και το endpoint.

Π.χ. *http://localhost:5215/api/Events/GetEvents/{id}*



Διάγραμμα 5: Διάγραμμα Κλάσεων UML (Controllers)

EventsController

Ο παρόν controller δημιουργήθηκε με σκοπό την διεξαγωγή δεδομένων που αφορούν τις εκδηλώσεις.

1. Endpoint: /api/Events/GetEvents/{eventTypeId}

Η συγκεκριμένη κλήση πραγματοποιείται με τη μέθοδο της μορφής GET, για να επιστρέψει όλες τις εκδηλώσεις με βάση τον τύπο τους, παραδείγματος χάρη αν το id είναι 1 επιστρέφονται όλες οι θεατρικές παραστάσεις. Η κλάση που χρησιμοποιείται ως απάντηση στο response body είναι η EventByType.

URL	Περιγραφή	Μέθοδος HTTP	Request Body	Response Body

http://localhost:5215/api/Events/GetEvents/{eventTypeId}	Επιστρέφει τις εκδηλώσεις με βάση το eventTypeId	GET		[
				{
				"idEvent": 0,
				"name": "string",
				"type": "string",
				"eventTypeId": 0,
				"organizerName": "string",
				"image": "string"
				}
				]

Πίνακας 1: Κλήση GetEvents

2. Endpoint: /api/Events/GetEventWithCom/{idEvent}

Η παρούσα κλήση πραγματοποιείται με τη μέθοδο GET, μόνο που εδώ επιστρέφει μεμονωμένη εκδήλωση και τις πληροφορίες που διαθέτει με βάση το id της, η κλάση που χρησιμοποιείται στο response body ως απάντηση είναι η RelatedEvents. Πιο συγκεκριμένα μέσω της χρήσης LINQ, η κλήση αναζητάει τα δεδομένα που αφορούν την εκδήλωση δημιουργώντας τα αντικείμενα EventInfo, EventByType και EventComments. Όπου αντίστοιχα τα συγκεκριμένα αντικείμενα αποθηκεύονται στο αντικείμενο RelatedEvents.

Παραδείγματος χάρη, αν το id είναι 1, τότε επιστρέφονται οι αντίστοιχες πληροφορίες της θεατρικής παράστασης Μπαμπάδες με ρούμι. Επιπλέον, επιστρέφονται τα αντίστοιχα σχόλια που έχουν υποβληθεί για την παρούσα εκδήλωση καθώς και αλληλοσχετιζόμενες εκδηλώσεις.

Ως απάντηση χρησιμοποιείται η κλάση RelatedEvents.

URL	**Περιγραφή**	**Μέθοδος HTTP**	**Request Body**	**Response Body**

				[
				{
				"infoOfEvents": {
				"eventId": 0,
				"eventName": "string",
				"eventInfo": "string",
				"premiere": "string",
				"monday": "string",
				"tuesday": "string",
				"wednesday": "string",
				"thursday": "string",
				"friday": "string",
				"saturday": "string",
				"sunday": "string",
				"eventType": "string",
				"nameOrg": "string",

http://localhost:5215/api/Events/GetEventWithCom/{idEvent}	Επιστρέφει μεμονωμένη εκδήλωση με βάση το idEvent	GET		"_img1": "string", "_img2": "string", "_img3": "string", "_img4": "string", "_img5": "string", "vip1": 0, "vip2": 0, "zoneA": 0, "zoneB": 0, "discount": 0, "factors_jobs": ["string"], "zoneC": 0, "zoneD": 0, "normal": 0, "eventTypeId": 0, "stars": 0 }, "related_events": [[{ "idEvent": 0, "name": "string", "type": "string", "eventTypeId": 0, "organizerName": "string", "image": "string" }]], "comments": [{ "id": 0, "text": "string", "user": "string", "time": "2024-09-25T21:00:24.581Z", "rating": 0 }] }]

Πίνακας 2: Κλήση GetEventWithCom

3. Endpoint: /api/Events/SearchEvent

Τέλος και η συγκεκριμένη κλήση πραγματοποιείται με τη μέθοδο της μορφής GET, η οποία επιστρέφει ως απάντηση τις εκδηλώσεις όπου έχει αναζητήσει ο χρήστης. Αξίζει να σημειωθεί πως για το response body της κλήσης χρησιμοποιείται η κλάση EventByType. Δηλαδή, υποβάλλοντας ένα κείμενο, γίνεται αναζήτηση με βάση την περιγραφή της εκδήλωσης στη βάση δεδομένων και επιστρέφονται όλες οι εκδηλώσεις που περιέχουν στον τίτλο τους το συγκεκριμένο λεκτικό.

URL	Περιγραφή	Μέθοδος HTTP	Request Body	Response Body

http://localhost:5215/api/Events/SearchEvent?text=%CE%BC%CF%80%CE%B1	Επιστρέφει τις εκδηλώσεις που αναζητήθηκαν με βάση το κείμενο που ορίστηκε	GET		[
				{
				"idEvent": 0,
				"name": "string",
				"type": "string",
				"eventTypeId": 0,
				"organizerName": "string",
				"image": "string"
				}
				]

Πίνακας 3: Κλήση SearchEvent

ProfessionalsController

1. Endpoint: /api/Professionals/GetFactors/{jobId1}

Η παρούσα κλήση πραγματοποιείται με τη μέθοδο της μορφής GET, για να επιστρέφει όλους τους επαγγελματίες με βάση το επάγγελμα τους, παραδείγματος χάρη αν το id είναι 1 επιστρέφονται όλοι οι ηθοποιοί. Η κλάση που επιστρέφεται ως απάντηση στο response body είναι η ProfByType.

URL	Περιγραφή	Μέθοδος HTTP	Request Body	Response Body

http://localhost:5215/api/Professionals/GetFactors/{jobId1}	Επιστρέφει τους επαγγελματίες με βάση το επάγγελμα τους, το jobId1	GET		[
				{
				"idFactor": 0,
				"firstName": "string",
				"lastName": "string",
				"img": "string",
				"_job": "string",
				"jobId1": 0
				}
				]

Πίνακας 4: Κλήση GetFactors

2. Endpoint: /api/Professionals/GetProfInfo/{idFactors}

Ομοίως πραγματοποιείται με τη μέθοδο της μορφής GET, μόνο που επιστρέφει έναν επαγγελματία με όλες τις πληροφορίες που διαθέτει με βάση το id του. Η κλάση που χρησιμοποιείται στο response body ως απάντηση είναι η ProfInfo. Η παρούσα κλάση έχει

μία ιδιαιτερότητα. Αν ο χρήστης δεν έχει συνδεθεί στην εφαρμογή τότε δεν επιστρέφεται η πληροφορία που αφορά το email του επαγγελματία. Αντίστοιχα δεν επιστρέφεται και όταν ο χρήστης δεν περιέχει επάγγελμα που αφορά τον χώρο του θεάματος.

Παραδείγματος χάρη, αν το id είναι 8, τότε επιστρέφονται οι πληροφορίες της ηθοποιού Βίκυς Σταυροπούλου.

URL	Περιγραφή	Μέθοδος HTTP	Request Body	Response Body

http://localhost:5215/api/Professionals/GetProfInfo/{idFactors}	Επιστρέφει μεμονωμένο επαγγελματία με βάση το idFactors	GET		[
				{
				"idFactor": 0,
				"firstName": "string",
				"lastName": "string",
				"email": "string",
				"bio": "string",
				"_img1": "string",
				"_img2": "string",
				"_img3": "string",
				"_img4": "string",
				"_img5": "string",
				"job": "string"
				}
				]

Πίνακας 5: Κλήση GetProfInfo

3. Endpoint: /api/Professionals/SearchProf

Έπειτα, και η συγκεκριμένη κλήση αντίστοιχα πραγματοποιείται με τη μέθοδο της μορφής GET, η οποία επιστρέφει ως απάντηση τους επαγγελματίες που έχει αναζητήσει ο χρήστης. Αξίζει να σημειωθεί πως για το response body της κλήσης χρησιμοποιήθηκε η κλάση ProfByType. Δηλαδή, υποβάλλοντας ένα κείμενο, γίνεται αναζήτηση με βάση το όνομα του επαγγελματία στη βάση δεδομένων και επιστρέφονται όλοι οι επαγγελματίες που περιέχουν στο όνομα τους το συγκεκριμένο λεκτικό.

URL	Περιγραφή	Μέθοδος HTTP	Request Body	Response Body

http://localhost:5215/api/Professionals/SearchProf?text=%CE%BC%CE%B1	Επιστρέφει τους επαγγελματίες που αναζητήθηκαν με βάση το κείμενο που ορίστηκε	GET		[
				{
				"idFactor": 0,
				"firstName": "string",
				"lastName": "string",
				"img": "string",
				"_job": "string",
				"jobId1": 0
				}
				]

Πίνακας 6: Κλήση SearchProf

4. Endpoint: /api/Professionals/AllJobs

Τέλος, η κλήση AllJobs ομοίως πραγματοποιείται με τη μέθοδο της μορφής GET, η οποία επιστρέφει ως απάντηση όλα τα επαγγέλματα που είναι καταχωρημένα στη βάση δεδομένων. Η παρούσα κλήση χρησιμοποιείται κατά τη διαδικασία της εγγραφής νέου χρήστη, ώστε μέσα από μία λίστα να επιλέξει ένα από τα επαγγέλματα που είναι καταχωρημένα στη βάση. Η κλάση που αντικατοπτρίζει τα δεδομένα ως απάντηση στο response body είναι η Jobs.

URL	Περιγραφή	Μέθοδος HTTP	Request Body	Response Body

http://localhost:5215/api/Professionals/AllJobs	Επιστρέφει όλα τα επαγγέλματα	GET		[
				{
				"id": 0,
				"job": "string"
				}
				]

Πίνακας 7: Κλήση AllJobs

HomeController

1. Endpoint: /api/Home/RandomImg

Η παρούσα κλήση όπως και οι προηγούμενες, πραγματοποιείται με τη μέθοδο της μορφής GET, η οποία υλοποιήθηκε για να επιστρέφει τυχαίες φωτογραφίες εκδηλώσεων. Η κλάση που επιστρέφεται ως απάντηση στο response body είναι η RandomImages.

URL	Περιγραφή	Μέθοδος HTTP	Request Body	Response Body

http://localhost:5215/api/Home/RandomImg	Επιστρέφει τυχαίες εικόνες εκδηλώσεων	GET		[
				{
				"eventId": 0,
				"imgId": 0,
				"img1": "string"
				}
				]

Πίνακας 8: Κλήση RandomImg

2. Endpoint: /api/Home/RandomEvents

Παρόμοια κλήση είναι και η παρούσα. Πραγματοποιείται με τη μέθοδο της μορφής GET και υλοποιήθηκε για να επιστρέφει τυχαίες εκδηλώσεις. Η κλάση που επιστρέφεται ως απάντηση στο response body είναι η EventByType, καθώς επιστρέφει συνοπτικές πληροφορίες εκδηλώσεων.

URL	Περιγραφή	Μέθοδος HTTP	Request Body	Response Body

http://localhost:5215/api/Home/RandomEvents	Επιστρέφει τυχαίες εκδηλώσεις	GET		[
				{
				"idEvent": 0,
				"name": "string",
				"type": "string",
				"eventTypeId": 0,
				"organizerName": "string",
				"image": "string"
				}
				]

Πίνακας 9: Κλήση RandomEvents

CommentsController

1. Endpoint: /api/Comments/PostComments

Η παρούσα κλήση πραγματοποιείται με τη μέθοδο της μορφής POST και υλοποιήθηκε με σκοπό την καταχώρηση νέου σχόλιου από έναν χρήστη σε μία εκδήλωση. Πιο συγκεκριμένα, ακολουθεί μία σειρά από ελέγχους για την επιτυχή καταχώρηση ενός σχόλιου. Αρχικά, πραγματοποιείται έλεγχος αν ο χρήστης έχει ολοκληρώσει την διαδικασία του Authentication στην εφαρμογή. Εφόσον λοιπόν ο χρήστης είναι συνδεδεμένος, γίνεται έλεγχος στην βάση δεδομένων αν έχει ξανά υποβάλλει αξιολόγηση για την συγκεκριμένη εκδήλωση. Τέλος, δημιουργείται ένα νέο αντικείμενο της μορφής ComOfEvent, όπου περιέχει τα απαραίτητα στοιχεία υποβολής (αστέρια εκδήλωσης, σχόλιο, ημερομηνία κλπ) και καταχωρείται στην βάση δεδομένων.

URL	Περιγραφή	Μέθοδος HTTP	Request Body	Response Body

http://localhost:5215/api/Comments/PostComments	Καταχωρεί νέα αξιολόγηση/ σχόλιο για μία εκδήλωση	POST	{
			"eventId": 0,
			"rate": 0,
			"text": "string"
			}
				POSTED

Πίνακας 10: Κλήση PostComments

AuthenticationController

1. Endpoint: /api/AuthenticationController/Register

Η κλήση πραγματοποιείται με τη μέθοδο της μορφής POST και υλοποιήθηκε με σκοπό τη δημιουργία νέου χρήστη στην εφαρμογή χρησιμοποιώντας ένα σύστημα διαχείρισης ταυτότητας (Identity Management System), το οποίο διαθέτει λειτουργίες για αυθεντικοποίηση και εξουσιοδότηση στην εφαρμογή. Πιο συγκεκριμένα, ως όρισμα δέχεται το σώμα που στέλνει στην κλήση, δηλαδή την κλάση RegisterModel. Πραγματοποιείται έλεγχος αν ο χρήστης υπάρχει ήδη στην βάση δεδομένων και στην συνέχεια καταχωρεί τα νέα στοιχεία μέσω της χρήσης της μεθόδου CreateAsync. Αν οι παραπάνω διαδικασίες δεν έχουν ολοκληρωθεί με επιτυχία τότε η απάντηση που επιστρέφεται στο response body είναι το σφάλμα της μη καταχώρησης του χρήστη, διαφορετικά επιστρέφεται επιτυχής απάντηση.

URL	Περιγραφή	Μέθοδος HTTP	Request Body	Response Body
http://localhost:5215/api/Authentication/Register	Πραγματοποιεί εγγραφή νέου χρήστη	POST	{ "username": "string", "email": "string", "password": "string", "firstName": "string", "lastName": "string", "job": "string" }	

Πίνακας 11: Κλήση Register

2. Endpoint: /api/AuthenticationController/Login

Η κλήση πραγματοποιείται με τη μέθοδο της μορφής POST και υλοποιήθηκε με σκοπό τη σύνδεση χρήστη στην εφαρμογή χρησιμοποιώντας ένα σύστημα διαχείρισης ταυτότητας (Identity Management System), το οποίο αναλύθηκε προηγουμένως. Πιο συγκεκριμένα, ως όρισμα δέχεται το σώμα που στείλει στην κλήση, δηλαδή την κλάση LoginModel. Αρχικά, πραγματοποιείται έλεγχος αν ο χρήστης είναι εγγεγραμμένος στην εφαρμογή μέσω της μεθόδου FindByNameAsync και στη συνέχεια αν έχει οριστεί σωστός κωδικός πρόσβασης μέσω της CheckPasswordAsync. Έπειτα, για να ολοκληρωθεί η αυθεντικοποίηση του χρήστη, δημιουργείται ένα token μέσω του JwtSecurityKey, με σκοπό την πρόσβαση του στην εφαρμογή, για λειτουργίες που απαιτούν αυθεντικοποίηση. Η απάντηση της κλήσης στο response body σε περίπτωση επιτυχής σύνδεσης είναι το token του χρήστη, διαφορετικά επιστρέφει μήνυμα σφάλματος.

URL	Περιγραφή	Μέθοδος HTTP	Request Body	Response Body
http://localhost:5215/api/Authentication/Login	Πραγματοποιεί σύνδεση του χρήστη	POST	{ "username": "string", "password": "string" }	

Πίνακας 12: Κλήση Login

3. Endpoint: /api/AuthenticationController/Logout

Η κλήση πραγματοποιείται με τη μέθοδο της μορφής GET και υλοποιήθηκε με σκοπό την αποσύνδεση του χρήστη από την εφαρμογή. Η αποσύνδεση ολοκληρώνεται με τη χρήση της ασύγχρονης μεθόδου SignOutAsync, κατά την οποία ακυρώνεται το token του χρήστη. Κατά την επιτυχής αποσύνδεση του χρήστη, η απάντηση της κλήσης στο response body είναι το status.

URL	Περιγραφή	Μέθοδος HTTP	Request Body	Response Body

http://localhost:5215/api/Authentication/Logout	Πραγματοποιεί αποσύνδεση του χρήστη	GET		{
				"status": true
				}

Πίνακας 13: Κλήση Logout

4.3 Ανάλυση frontend

4.3.1 Τεχνολογία frontend

Αντίστοιχα, η τεχνολογία που χρησιμοποιήθηκε για την ολοκληρωμένη υλοποίηση του frontend είναι το λογισμικό της Angular. Η Angular βασίζεται στην Typescript και περιλαμβάνει μεγάλη γκάμα βιβλιοθηκών και λειτουργιών όπως είναι η διαχείριση των φορμών, η οποία βοηθάει στην ολοκληρωμένη οργάνωση και υλοποίηση της αλληλεπίδρασης του χρήστη με την εφαρμογή. Για την ολοκληρωμένη ανάπτυξη της εφαρμογής χρησιμοποιούνται αρχεία components όπου κάθε ένα περιέχει αντίστοιχα ένα template HTML και CSS.

Εικόνα 80: Angular

4.3.2 Δημιουργία Project Angular

Σχετικά με την δημιουργία του προγράμματος σε Angular, αρχικά χρειάζεται η δημιουργία νέου φακέλου στον υπολογιστή. Στη συνέχεια, ανοίγοντας την Γραμμή Εντολών του υπολογιστή πρέπει να «τρέξουμε» την εντολή:

npm install -g @angular/cli

ώστε να εγκαταστήσουμε το Angular Command Line Interface (CLI). Έπειτα, για την δημιουργία νέου έργου χρειάστηκε η παρακάτω εντολή κατά την οποία δηλώνουμε και το όνομα του έργου.

Ng new DemoFront_v1

Τέλος, εφόσον η δημιουργία νέου έργου έχει πραγματοποιηθεί με επιτυχία συμπληρώνοντας την παρακάτω εντολή, ανοίγεται το νέο έργο μέσω του Visual Studio Code.

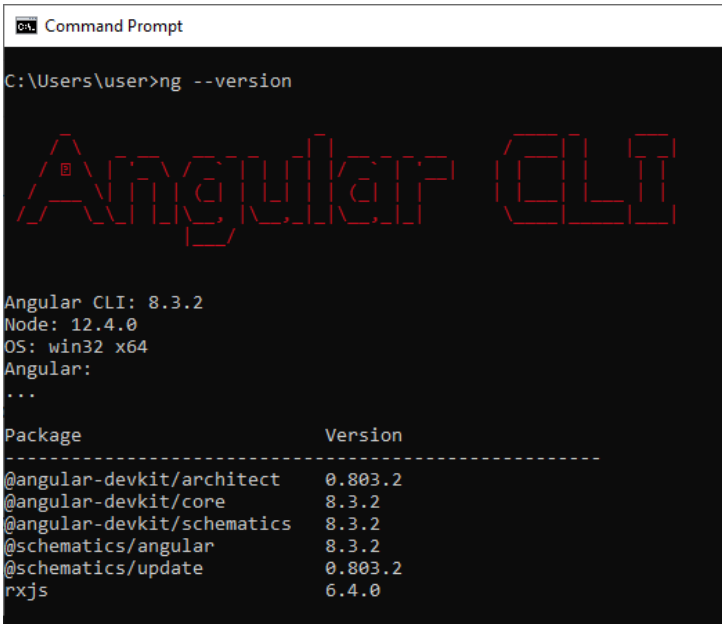
ng serve -o

Η δημιουργία των services πραγματοποιείται με τη συγκεκριμένη εντολή:

ng generate services data

Ενώ για την δημιουργία αρχείων (components) πραγματοποιείται με τη συγκεκριμένη εντολή:

ng g c user –standalone



Εικόνα 81: Εγκατάσταση Angular CLI

Εικόνα 82: Visual Studio Code

4.3.3 Υλοποίηση

Για να πραγματοποιηθεί η υλοποίηση του frontend της εφαρμογής και να εμφανίσουμε τα δεδομένα που απαιτούνται στον χρήστη, χρειάστηκε να δημιουργήσουμε components ανά κατηγορία δεδομένων και services ώστε να χρησιμοποιήσουμε λειτουργίες που χρειάζονται σε περισσότερα από ένα components.

Πιο αναλυτικά, παρακάτω αναλύονται σημαντικά κομμάτια της εφαρμογής.

AppComponent

Το AppComponent δημιουργείται αυτόματα κατά τη δημιουργία της εφαρμογής και είναι το βασικό component. Κάθε component περιέχει από έναν selector ώστε να ενσωματώνεται ο κώδικας του σε κάποιο άλλο component. Στο συγκεκριμένο component χρησιμοποιούμε τον selector του NavbarComponent, app-navbar, ώστε να εμφανίζεται σε όλες τις σελίδες της εφαρμογής το μενού της. Αντίστοιχα, έχει προστεθεί το footer της εφαρμογής για να εμφανίζεται σε όλες τις σελίδες.

App.routes.ts

Το συγκεκριμένο αρχείο περιλαμβάνει τα Routes της εφαρμογής, ώστε ο χρήστης να μπορεί να μεταβαίνει από μία σελίδα σε μία άλλη. Πιο αναλυτικά, για κάθε component που έχει δημιουργηθεί για τις απαιτήσεις της εφαρμογής αποτελείται από μία διεύθυνση Url.

Main.ts

Το συγκεκριμένο αρχείο λειτουργεί ως το αρχείο εκκίνησης της εφαρμογής. Περιέχει το bootstrapApplication το οποίο βοηθάει στην ομαλή υλοποίηση της εφαρμογής χρησιμοποιώντας τα components που περιλαμβάνει, ορίζοντας τα μονοπάτια που διαθέτουν στην εφαρμογή.

AboutComponent

Ο συγκεκριμένος φάκελος περιέχει ένα component, το AboutComponent χρησιμοποιήθηκε κυρίως για την διεξαγωγή πληροφοριών σχετικά με την ιστοσελίδα, πιο συγκεκριμένα ένα πληροφοριακό κείμενο, το οποίο έχει ενταχθεί στον HTML κώδικα του component και περιέχει κώδικα CSS για την εμφάνιση και μορφοποίηση του κειμένου.

EventsComponent

Το EventsComponent υλοποιήθηκε με σκοπό την αναπαράσταση όλων των εκδηλώσεων ανάλογα το είδος της εκδήλωσης π.χ. αν είναι θεατρικές παραστάσεις εμφανίζεται το σύνολο αυτών. Οι εκδηλώσεις απεικονίζονται μέσω μίας κάρτας όπου περιέχει συνοπτικές πληροφορίες κάθε εκδήλωσης, εικόνα, όνομα, είδος εκδήλωσης, διοργανωτής. Για την φόρτωση των δεδομένων αυτών, χρησιμοποιήθηκε μία κλήση από το backend της εφαρμογής ,όπου επιστρέφει μία λίστα εκδηλώσεων ανάλογα το Id (το είδος των εκδηλώσεων). Τέλος, το component περιλαμβάνει ένα κουμπί, το searchBtn, κατά το οποίο πραγματοποιείται η αναζήτηση εκδηλώσεων από τον χρήστη.

HistoryComponent

Ο συγκεκριμένος φάκελος περιέχει ένα component, το οποίο χρησιμοποιήθηκε κυρίως για την διεξαγωγή ιστορικών πληροφοριών σχετικά με το θέατρο και τον κινηματογράφο, πιο συγκεκριμένα περιέχει πληροφοριακά κείμενα και εικόνες, το οποίο έχει ενταχθεί στον HTML κώδικα του component και περιέχει κώδικα CSS για την εμφάνιση και μορφοποίηση του κειμένου.

HomepageComponent

Το παρόν component αποτελείται από περισσότερες απαιτήσεις σε σχέση με τα υπόλοιπα, καθώς εμφανίζει μία γκάμα δεδομένων της εφαρμογής. Πιο συγκεκριμένα, εμφανίζει ένα σύνολο από τυχαίες εκδηλώσεις και ένα σύνολο από τυχαίες εικόνες των εκδηλώσεων (η διεξαγωγή αυτών πραγματοποιήθηκε μέσω κλήσεων του backend). Έπειτα, διαθέτει κατηγορίες εκδηλώσεων και επαγγελματιών, όπου μέσω της χρήσης του route, ο χρήστης μεταφέρεται στην αντίστοιχη κατηγορία των εκδηλώσεων και επαγγελματιών. Οι κατηγορίες αυτές απεικονίζονται με την χρήση καρτών. Τέλος, διαθέτει μία επιπλέον λειτουργία, την ScrollIntoView, όπου ανάλογα την κατηγορία του εσωτερικού μενού της εφαρμογής, ο χρήστης μεταφέρεται στο αντίστοιχο σημείο της σελίδας που απεικονίζονται οι πληροφορίες που έχει επιλέξει.

InfoEventComponent

Το InfoEventComponent υλοποιήθηκε με σκοπό την αναπαράσταση πληροφοριών συγκεκριμένης εκδήλωσης. Για κάθε εκδήλωση εμφανίζονται: όνομα, περιγραφή, τιμές, μέρες και ώρες παρακολούθησης και διάφορες άλλες χρήσιμες πληροφορίες καθώς και ένα σύνολο από σχόλια και αξιολογήσεις χρηστών. Έπειτα, διαθέτει ένα κουμπί, το onSubmit, κατά το οποίο πραγματοποιείται υποβολή νέας αξιολόγησης μέσω μεθόδου HTTP που είναι καταχωρημένη στο backend της εφαρμογής. Αφού ο χρήστης έχει συμπληρώσει τα απαραίτητα στοιχεία και είναι συνδεδεμένος στην εφαρμογή πραγματοποιούνται έλεγχοι. Ανάλογα τον έλεγχο που πραγματοποιείται, εμφανίζεται και το κατάλληλο μήνυμα μέσω της χρήσης του alert().

InfoFactorComponent

Το InfoFactorComponent υλοποιήθηκε για να αντικατοπτρίσει τις πληροφορίες κάθε επαγγελματία, όνομα, εικόνα επαγγελματία, βιογραφικό και το email σε περίπτωση όπου το επάγγελμα του χρήστη αφορά τον χώρο του θεάματος. Δηλαδή λειτουργεί ως η καρτέλα του κάθε επαγγελματία.

NavbarComponent

Το NavbarComponent χρησιμοποιεί ως το header της ιστοσελίδας. Αναπαριστά το μενού της εφαρμογής, ώστε ο χρήστης να μπορεί να πλοηγείτε ανάλογα την σελίδα που επιθυμεί, π.χ. εκδηλώσεις, επαγγελματίες, σύνδεση κλπ. Για την μεταφορά από μία σελίδα σε μία άλλη χρησιμοποιούνται routes, όπου ορίζουν τα components με τα αντίστοιχα Url τους.

ProfessionalsComponent

Ομοίως με το EventsComponent λειτουργεί και το παρόν component, μόνο που αναπαριστά όλους τους επαγγελματίες ανάλογα το επάγγελμα τους, π.χ. αν είναι ηθοποιοί. Οι επαγγελματίες απεικονίζονται μέσω μίας κάρτας όπου περιέχει συνοπτικές πληροφορίες αυτών, δηλαδή το όνομα τους το επάγγελμα τους και μία εικόνα τους. Για την φόρτωση των δεδομένων αυτών, χρησιμοποιήθηκε μία κλήση από το backend της εφαρμογής ,όπου επιστρέφει μία λίστα από επαγγελματίες ανάλογα το Id (το είδος εργασίας τους). Τέλος, το component περιλαμβάνει ένα κουμπί, το searchBtn, κατά το οποίο πραγματοποιείται η αναζήτηση επαγγελματιών από τον χρήστη.

SigninComponent

Το SigninComponent υλοποιήθηκε για την σύνδεση των χρηστών στην εφαρμογή. Διαθέτει μία φόρμα από πεδία για την συμπλήρωση των στοιχείων, καθώς και το κουμπί onSubmit για την είσοδο στην εφαρμογή. Η διαχείριση της φόρμας χρησιμοποιήθηκε μία υπηρεσία που διαθέτει η Angular, το FormBuilder. Για να ελέγχουν τα στοιχεία εισόδου ώστε να πραγματοποιηθεί η σύνδεση, χρησιμοποιήθηκε

το authentication service. Αφού ολοκληρωθεί η σύνδεση με επιτυχία μέσω της χρήσης του route, ο χρήστης μεταφέρεται στο HomepageComponent. Τέλος, ανάλογα την περίπτωση επιστρέφεται αντίστοιχο μήνυμα στον χρήστη.

SignupComponent

Το SignupComponent υλοποιήθηκε για την εγγραφή νέου χρήστη στην εφαρμογή. Διαθέτει μία φόρμα από πεδία για την συμπλήρωση των απαιτούμενων στοιχείων, καθώς και το κουμπί onSignUp για την ολοκλήρωση της εγγραφής του νέου χρήστη. Η διαχείριση της φόρμας χρησιμοποιήθηκε μία υπηρεσία που διαθέτει η Angular, το FormBuilder. Για να ελεγχτούν τα στοιχεία εγγραφής χρησιμοποιήθηκε το authentication service. Τέλος, ανάλογα την περίπτωση επιστρέφεται αντίστοιχο μήνυμα στον χρήστη.

Κατά την φόρτωση της σελίδας, με την χρήση της κλήσης AllJobs, εμφανίζονται όλα τα επαγγέλματα που είναι καταχωρημένα στη βάση, ώστε ο χρήστης να επιλέξει το επάγγελμα με το οποίο ασχολείται.

Helpers

Κλάση ValidateForm: Η συγκεκριμένη κλάση δημιουργήθηκε για να πραγματοποιείται έλεγχος αν όλα τα πεδία μία φόρμας της εφαρμογής είναι συμπληρωμένα. Ο έλεγχος γίνεται μέσω βρόγχου για κάθε πεδίο της φόρμας.

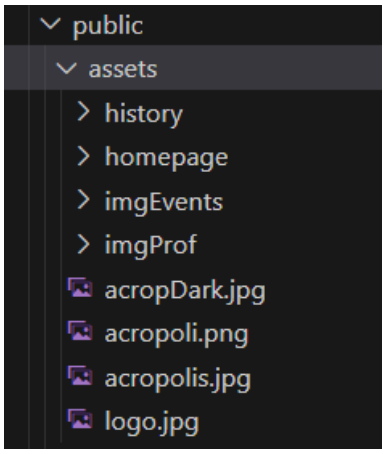
Authentication - Service

Ο συγκεκριμένος φάκελος Services → Authentication περιέχει υπηρεσίες που θα χρησιμοποιηθούν για να πραγματοποιεί ο χρήστης σύνδεση και εγγραφή στην εφαρμογή, καθώς και αποσύνδεση. Οι υπηρεσίες αυτές υλοποιήθηκαν καθώς η χρήση τους απαιτείται σε περισσότερα από ένα component.

Αποτελούνται από τρεις κλήσεις, η μία είναι για την σύνδεση του χρήστη στην εφαρμογή, η άλλη για την εγγραφή και η τελευταία για την αποσύνδεση. Η κλήση της σύνδεσης και της εγγραφής χρησιμοποιούν ένα αντικείμενο ως παράμετρο, καθώς είναι τύπου POST. Έπειτα, τέθηκε σε χρήση η συνάρτηση όπου ανακτά το token του χρήστη καθώς και μία συνάρτηση όπου πραγματοποιείται έλεγχος αν ο χρήστης είναι συνδεδεμένος στην εφαρμογή, δηλαδή αν υπάρχει token.

Εικόνες

Η εικόνες της ιστοσελίδας αποθηκεύτηκαν τοπικά στο φάκελο public → assets, και ανάλογα την κατηγορία έχουν δημιουργηθεί οι αντίστοιχοι φάκελοι.



Εικόνα 83: Φάκελος Εικόνων

Βιβλιοθήκες – Λοιπά

Για την διαδικασία της αξιολόγησης συγκεκριμένης εκδήλωσης μέσω αστεριών από τους χρήστες, χρησιμοποιήθηκε η βιβλιοθήκη Ignite UI for Angular. Για την προσθήκη της βιβλιοθήκης της εφαρμογής τρέξαμε την αντίστοιχη εντολή.

npm install igniteui-webcomponents --save

Για την μορφοποίηση των ιστοσελίδων χρησιμοποιήθηκε η βιβλιοθήκη Bootstrap.

4.4 Ανάλυση Βάσης Δεδομένων

Η βάση δεδομένων δημιουργήθηκε μέσω μίας εφαρμογής λογισμικού της Microsoft, το Microsoft SQL Server Management Studio (SSMS), η οποία διαχειρίζεται δεδομένα του SQL Server.

Εικόνα 84: SSMS

Καταρχάς, η βάση δεδομένων αποτελείται από τους πίνακες που δημιουργήθηκαν αυτόματα από το ASP.NET Core Identity, οι οποίοι χρησιμοποιούνται για την διαχείριση αυθεντικοποίησης και εξουσιοδότησης της εφαρμογής, καθώς και ένας πίνακας που χρησιμοποιείται για την διαχείριση του ιστορικού των migrations που δημιουργούνται μέσω του Entity Framework.

Κατόπιν, για να αντλήσουμε αποθηκεύσουμε και να επεξεργαστούμε τα δεδομένα της εφαρμογής που απαιτούνται, χρειάστηκε να δημιουργηθούν έντεκα επιπλέον πίνακες.

4.4.1 Πίνακας Events

Ο πίνακας Events αποτελείται από τα δεδομένα μίας εκδήλωσης. Επιπλέον, περιέχει id άλλων πινάκων για την άντληση δεδομένων που σχετίζονται με αυτή. Οι πίνακες αυτοί είναι οι: Organizer, Images, Price, Dates, TypeOfEvent.

Όνομα	Περιγραφή	Τύπος	Περιορισμοί
Id	Αναγνωριστικό κλειδί	Integer	Primary Key, NOT NULL
name	Όνομα	Nvarchar(max)	NULL
information	Περιγραφή	Nvarchar(max)	NULL
premiere	Ημ/νία πρεμιέρας	Datetimeoffset(7)	NULL
orgId	Αναγνωριστικό κλειδί διοργανωτή	Integer	Foreign Key, NOT NULL
imgId	Αναγνωριστικό κλειδί εικόνων	Integer	Foreign Key, NOT NULL
priceId	Αναγνωριστικό κλειδί τιμών	Integer	Foreign Key, NOT NULL
dateId	Αναγνωριστικό κλειδί ημ/νίας και ώρας	Integer	Foreign Key, NOT NULL
eventTypeId	Αναγνωριστικό κλειδί τύπου εκδήλωσης	integer	Foreign Key, NOT NULL

Πίνακας 14: Δομή Πίνακα Events

4.4.2 Πίνακας Factors

Ο πίνακας Factors αποτελείται από τα δεδομένα των επαγγελματιών, καθώς και αλληλοσχετίζεται με άλλους πίνακες και είναι οι: Images, Jobs.

Όνομα	Περιγραφή	Τύπος	Περιορισμοί
Id	Αναγνωριστικό κλειδί	Integer	Primary Key, NOT NULL
firstName	Όνομα	Nvarchar(50)	NULL
lastName	Επώνυμο	Nvarchar(50)	NULL
jobId1	Αναγνωριστικό κλειδί επαγγέλματος	Integer	Foreign Key, NOT NULL
email	Email	Nvarchar(50)	NULL
jobId2	Αναγνωριστικό κλειδί επαγγέλματος 2	Integer	Foreign Key, NOT NULL

imgId	Αναγνωριστικό κλειδί εικόνων	Integer	Foreign Key, NOT NULL
bio	Βιογραφικό	Nvarchar(max)	NULL

Πίνακας 15: Δομή Πίνακα Factors

4.4.3 Πίνακας Comments

Ο πίνακας Comments αποτελείται από τα σχόλια και την αξιολόγηση που έχει υποβάλλει κάθε χρήστης σε κάθε εκδήλωση.

Όνομα	Περιγραφή	Τύπος	Περιορισμοί
Id	Αναγνωριστικό κλειδί	Integer	Primary Key, NOT NULL
text	Κείμενο σχόλιου	Nvarchar(max)	NULL
userId	Αναγνωριστικό κλειδί χρήστη	Nvarchar(max)	NULL
time	Ώρα υποβολής	Datetimeoffset(7)	NULL
rating	Αριθμός αστεριών	Integer	NULL

Πίνακας 16: Δομή Πίνακα Comments

4.4.4 Πίνακας ComOfEvent

Ο πίνακας ComOfEvent λειτουργεί ως διασύνδεση μεταξύ των πινάκων Events και Comments, όπου συνδέει για κάθε παράσταση όποιο σχόλιο αναλογεί.

Όνομα	Περιγραφή	Τύπος	Περιορισμοί
Id	Αναγνωριστικό κλειδί	Integer	Primary Key, NOT NULL
eventId	Αναγνωριστικό κλειδί εκδήλωσης	Integer	Foreign Key, NOT NULL
comId	Αναγνωριστικό κλειδί σχόλιων	Integer	Foreign Key, NOT NULL

Πίνακας 17: Δομή Πίνακα ComOfEvent

4.4.5 Πίνακας Dates

Ο πίνακας Dates αποτελείται από τις ημέρες και ώρες που πραγματοποιείται κάθε εκδήλωση. Να σημειωθεί πως οι ώρες δεν είναι τύπου datetimeoffset(7), διότι υπάρχει περίπτωση μία εκδήλωση να πραγματοποιείται σε δύο συγκεκριμένες ώρες μέσα σε μία ημέρα.

Όνομα	Περιγραφή	Τύπος	Περιορισμοί
Id	Αναγνωριστικό κλειδί	Integer	Primary Key, NOT NULL
Monday	Ώρες Δευτέρας	Nvarchar(50)	NULL
Tuesday	Ώρες Τρίτης	Nvarchar(50)	NULL
Wednesday	Ώρες Τετάρτης	Nvarchar(50)	NULL
Thursday	Ώρες Πέμπτης	Nvarchar(50)	NULL
Friday	Ώρες Παρασκευής	Nvarchar(50)	NULL
Saturday	Ώρες Σαββάτου	Nvarchar(50)	NULL
Sunday	Ώρες Κυριακής	Nvarchar(50)	NULL

Πίνακας 18: Δομή Πίνακα Dates

4.4.6 Πίνακας Events_Factors

Ο πίνακας Events_Factors λειτουργεί ως διασύνδεση μεταξύ των πινάκων Events και Factors. Στην ουσία για κάθε εκδήλωση συνδέεται και ο αντίστοιχος επαγγελματίας που εργάζεται σε αυτή. Η δημιουργία του συγκεκριμένου πίνακα πραγματοποιήθηκε καθώς υπάρχει περίπτωση ένας επαγγελματίας να εργάζεται σε παραπάνω από μία εκδήλωση.

Όνομα	Περιγραφή	Τύπος	Περιορισμοί
Id	Αναγνωριστικό κλειδί	Integer	Primary Key, NOT NULL
eventId	Αναγνωριστικό κλειδί εκδήλωσης	Integer	Foreign Key, NOT NULL
factorId	Αναγνωριστικό κλειδί επαγγελματία	Integer	Foreign Key, NOT NULL

Πίνακας 19: Δομή Πίνακα Events_Factors

4.4.7 Πίνακας Images

Ο πίνακας Images αποτελείται από τα μονοπάτια των εικόνων κάθε εκδήλωσης και κάθε επαγγελματία.

Όνομα	Περιγραφή	Τύπος	Περιορισμοί
Id	Αναγνωριστικό κλειδί	Integer	Primary Key, NOT NULL
img1	Μονοπάτι εικόνας 1	Nvarchar(50)	NULL
img2	Μονοπάτι εικόνας 2	Nvarchar(50)	NULL
img3	Μονοπάτι εικόνας 3	Nvarchar(50)	NULL
img4	Μονοπάτι εικόνας 4	Nvarchar(50)	NULL
img5	Μονοπάτι εικόνας 5	Nvarchar(50)	NULL

Πίνακας 20: Δομή Πίνακα Images

4.4.8 Πίνακας Jobs

Ο πίνακας Jobs αποτελείται από τα επαγγέλματα των επαγγελματιών καθώς και τα επαγγέλματα που μπορούν να χρησιμοποιηθούν για την δήλωση επαγγέλματος του νέου χρήστη.

Όνομα	Περιγραφή	Τύπος	Περιορισμοί
Id	Αναγνωριστικό κλειδί	Integer	Primary Key, NOT NULL
job	Επάγγελμα	Navarchar(50)	NULL

Πίνακας 21: Δομή Πίνακα Jobs

4.4.9 Πίνακας Organizer

Ο πίνακας Organizer αποτελείται από τους διοργανωτές των εκδηλώσεων.

Όνομα	Περιγραφή	Τύπος	Περιορισμοί
Id	Αναγνωριστικό κλειδί	Integer	Primary Key, NOT NULL
name	Όνομα	Nvarchar(max)	NULL
address	Διεύθυνση	Nvarchar(50)	NULL
city	Πόλη	Nvarchar(50)	NULL

tk	T.K.	Integer	NULL
phone	Τηλέφωνο	Nvarchar(50)	NULL
email	Email	Nvarchar(50)	NULL

Πίνακας 22: Δομή Πίνακα Organizer

4.4.10 Πίνακας Price

Ο πίνακας Price αποτελείται από τις τιμές που έχει κάθε εκδήλωση για να παρακολουθηθεί.

Όνομα	Περιγραφή	Τύπος	Περιορισμοί
Id	Αναγνωριστικό κλειδί	Integer	Primary Key, NOT NULL
VIP1	Τιμή Vip1	Float	NULL
VIP2	Τιμή Vip2	Float	NULL
ZoneA	Τιμή Ζώνης Α	Float	NULL
ZoneB	Τιμή Ζώνης Β	Float	NULL
discount	Τιμή με έκπτωση	Float	NULL
ZoneC	Τιμή Ζώνης Γ	Float	NULL
ZoneD	Τιμή Ζώνης Δ	Float	NULL
normal	Κανονική τιμή	Float	NULL

Πίνακας 23: Δομή Πίνακα Price

4.4.11 Πίνακας TypeOfEvent

Ο πίνακας TypeOfEvent αποτελείται από τα είδη εκδηλώσεων.

Όνομα	Περιγραφή	Τύπος	Περιορισμοί
Id	Αναγνωριστικό κλειδί	Integer	Primary Key, NOT NULL
name	Όνομα	Navarchat(50)	NULL

Πίνακας 24: Δομή Πίνακα TypeOfEvent

Κεφάλαιο 5 - Συμπεράσματα

Συμπεραίνοντας λοιπόν, η παρούσα Πτυχιακή εργασία περιλαμβάνει την υλοποίηση μίας διαδικτυακής εφαρμογής παρουσιάζοντας τα πολιτιστικά δρώμενα της πόλης. Για τη διεκπεραίωση της έγινε πλήρη ανάλυση των ζητούμενων. Μέσω των τεχνολογιών ASP.NET Core Web API και Angular, αναπτύχθηκε η εφαρμογή με σκοπό την παρουσίαση διάφορων εκδηλώσεων και την αναζήτηση επαγγελματιών του χώρου του θεάματος. Κατά την περιήγηση στην εφαρμογή δίνεται η δυνατότητα προβολής εκδηλώσεων, οι οποίες περιλαμβάνουν αναλυτικές πληροφορίες, όπως το πότε και που διεξάγονται. Για κάθε χρήστη επιτρέπεται η ανώνυμη περιήγηση στην εφαρμογή, όμως δεν θα του παρέχονται κάποιες ιδιαίτερες λειτουργίες. Ο χρήστης κατόπιν έχει ολοκληρώσει την εγγραφή του στην εφαρμογή, δηλώνοντας το επάγγελμα κατά το οποίο ασχολείται, σε περίπτωση που το επάγγελμα είναι σχετικό με το χώρο του θεάματος, μπορεί να αναζητήσει συναδέλφους για την πραγματοποίηση είτε συνεργασίας είτε επικοινωνίας. Παράλληλα ανεξαρτήτως ιδιότητας του χρήστη, είναι

επιτρεπτό η υποβολή αξιολόγησης και σχόλιου για κάθε εκδήλωση. Καθώς κατά την υποβολή της αξιολόγησης εμφανίζονται δημόσια τα στοιχεία του.

5.1 Μελλοντικές Επεκτάσεις της Εφαρμογής

Η παρούσα πτυχιακή εστίασε στην προβολή των πολιτιστικών δρώμενων. Λόγω λοιπόν του μεγάλου όγκου πολιτιστικών δρώμενων που πραγματοποιούνται στην χώρα μας, μία σημαντική μελλοντική επέκταση θα ήταν ένα σύστημα διαχείρισης εκδηλώσεων, κατόπιν δημιουργίας ενός χρήστη με ιδιότητα Admin. Για τον συγκεκριμένο χρήστη θα δίνεται η δυνατότητα καταχώρησης νέων εκδηλώσεων με τις απαραίτητες πληροφορίες που χρειάζονται. Η επέκταση αυτή, θα βοηθούσε στην αποφυγή της υποβολής των δεδομένων άμεσα από την βάση δεδομένων, καθώς στην παρούσα φάση, για την καταχώρηση νέων εκδηλώσεων πρέπει ο διαχειριστής της εφαρμογής να καταχωρεί τα δεδομένα απευθείας μέσα από τη βάση δεδομένων.

Μία σημαντική υλοποίηση για τους επαγγελματίες θα μπορούσε να είναι η υποβολή βιογραφικού ενός επαγγελματία καθώς και η αποστολή email από έναν επαγγελματία σε έναν άλλο. Δηλαδή, πρώτον εφόσον ένας χρήστης είναι συνδεδεμένος στην εφαρμογή και έχει αναρτηθεί ή και όχι η παρουσίαση του βιογραφικού του, θα του επιτρέπεται η καταχώρηση ή η αλλαγή στοιχείων του, ώστε ένας επαγγελματίας του χώρου να μπορεί να τον αναζητήσει. Δεύτερον, για την υλοποίηση αποστολής email θα μπορούσε να χρησιμοποιηθεί ένας SMTP Server, όπου κάθε χρήστης που σχετίζεται με το χώρο του θεάματος και είναι συνδεδεμένος, να μπαίνει στη σελίδα παρουσίασης του επαγγελματία που τον ενδιαφέρει και μέσω μίας φόρμας υποβολής να συμπληρώνει τα απαραίτητα στοιχεία και να στέλνει το email που επιθυμεί.

Τέλος, μία σημαντική μελλοντική υλοποίηση θα ήταν κάθε χρήστης να συνδέεται αυτόματα μέσω της χρήση των social media ή του Google account του. Αυτό θα διευκόλυνε τους χρήστες που δεν είναι επαγγελματίες, καθώς δεν θα χρειαζόταν η συμπλήρωση των στοιχείων του και η ταυτοποίηση θα γινόταν απευθείας μέσω ενός account του στα social media π.χ. Facebook, Instagram.

Κεφάλαιο 6 - Βιβλιογραφία

[1] ajcvickers. “Overview of Entity Framework Core - EF Core,” May 25, 2021. https://learn.microsoft.com/en-us/ef/core/.

[2] “Angular - Τι Είναι η Angular.” Accessed September 29, 2024. https://angular-gr.web.app/guide/what-is-angular.

[3] “Angular Star Rating Component – Ignite UI for Angular.” Accessed September 29, 2024. https://www.infragistics.com/products/ignite-ui-angular/angular/components/rating.

[4] athinorama.gr. “Athinorama.gr.” Accessed September 29, 2024. https://www.athinorama.gr.

[5] Cartwright, Mark. “Αρχαιοελληνικό Θέατρο.” Εγκυκλοπαίδεια Παγκόσμιας Ιστορίας. Accessed September 29, 2024. https://www.worldhistory.org/trans/el/1-14956/.

[6] Chodak, Grzegorz, Grażyna Suchacka, and Yash Chawla. “HTTP-Level e-Commerce Data Based on Server Access Logs for an Online Store.” Computer Networks 183 (December 24, 2020): 107589. https://doi.org/10.1016/j.comnet.2020.107589.

[7] Cretalive. “Η ιστορία του θεάτρου.” Accessed September 29, 2024. https://www.cretalive.gr/istoria/h-istoria-toy-theatroy.

[8] dotnet-bot. “JwtSecurityToken Class (System.IdentityModel.Tokens.Jwt) - Microsoft Authentication Library for .NET.” Accessed September 29, 2024. https://learn.microsoft.com/en-us/dotnet/api/system.identitymodel.tokens.jwt.jwtsecuritytoken?view=msal-web-dotnet-latest.

[9] Emeka, B., S. Hidaka, and S. Liu. “A Formal Approach to Secure Design of RESTful Web APIs Using SOFL.” Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 12723 LNCS (2021): 105–25. https://doi.org/10.1007/978-3-030-77474-5_8.

[10] Golmohammadi, A., M. Zhang, and A. Arcuri. “Testing RESTful APIs: A Survey.” ACM Transactions on Software Engineering and Methodology 33, no. 1 (2023). https://doi.org/10.1145/3617175.

[11] panagiota. “Η γέννηση του κινηματογράφου στην Ελλάδα.” Text. Ellinismos Online, February 21, 2022. https://www.ellinismosonline.gr/el/tradition/i-gennisi-toy-kinimatografoy-stin-ellada.

[12] Portnova, T.V. “Ancient Theater Architecture as an Element of the World Cultural Landscape.” Periodicals of Engineering and Natural Sciences 9, no. 3 (2021): 236–50. https://doi.org/10.21533/pen.v9i3.2113.

[13] Rick-Anderson. “Call a Web API From a .NET Client (C#) - ASP.NET 4.x,” August 26, 2022. https://learn.microsoft.com/en-us/aspnet/web-api/overview/advanced/calling-a-web-api-from-a-net-client.

[14] ———. “Introduction to Identity on ASP.NET Core,” April 26, 2024. https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-8.0.

[15] RicoSuter. “ASP.NET Core Web API Documentation with Swagger / OpenAPI,” August 26, 2024. https://learn.microsoft.com/en-us/aspnet/core/tutorials/web-api-help-pages-using-swagger?view=aspnetcore-8.0.

[16] Troelsen, A., and P. Japikse. Pro C# 7: With .NET and .NET Core, Eighth Edition. Pro C# 7: With .NET and .NET Core, Eighth Edition, 2017. https://doi.org/10.1007/978-1-4842-3018-3.

[17] Visual Studio. “Visual Studio: IDE and Code Editor for Software Developers and Teams.” Accessed September 29, 2024. https://visualstudio.microsoft.com.

[18] “Γνώρισε Την Ελλάδα - Enterprise Greece.” Accessed September 29, 2024. https://www.enterprisegreece.gov.gr/h-ellada-shmera/zoi-stin-ellada/xrhsimes-plhrofories.

[19] “Εκδηλώσεις και εκθέσεις στην Αθήνα - Culture is Athens,” December 3, 2018. https://cultureisathens.gr/event-listing-1/.

[20] “Όλες οι εκδηλώσεις στην Αθήνα.” Accessed September 29, 2024. https://www.artandlife.gr/athens/events.

[21] Υπηρεσίες, More com Ηλεκτρονικές. “Εκδηλώσεις & more | Εισιτήριο στον κόσμο του θεάματος.” More.com. Accessed September 29, 2024. https://www.more.com/gr/.

[22] Alepis, Efthimios, and Maria Virvou. Object-Oriented User Interfaces for Personalized Mobile Learning. Vol. 64. Intelligent Systems Reference Library. Berlin, Heidelberg: Springer, 2014. https://doi.org/10.1007/978-3-642-53851-3.

[23] Alepis, Efthymios, and Maria Virvou. “Object oriented architecture for affective multimodal e-learning interfaces.” Intelligent Decision Technologies 4, no. 3 (January 1, 2010): 171–80. https://doi.org/10.3233/IDT-2010-0078.

[24] Virvou, Maria, George Tsihrintzis, Efthymios Alepis, and Ioanna-Ourania Stathopoulou. “Designing a Multi-Modal Affective Knowledge-Based User Interface: Combining Empirical Studies.” In Knowledge-Based Software Engineering, 250–59. IOS Press, 2008. https://doi.org/10.3233/978-1-58603-900-4-250.

Κεφάλαιο 7 Πίνακες

HTTP	Hypertext Transfer Protocol
API	Application Programming Interface
EF	Entity Framework
JWT	JSON Web Token
LINQ	Language-Integrated Query
URL	Uniform Resource Locator
CSS	Cascading Style Sheets
C#	C Sharp
SQL	Structured Query Language
IDE	Integrated Development Environment
SMTP	Simple Mail Transfer Protocol
UI	User Interface

Πίνακας 25: Συντομογραφίες

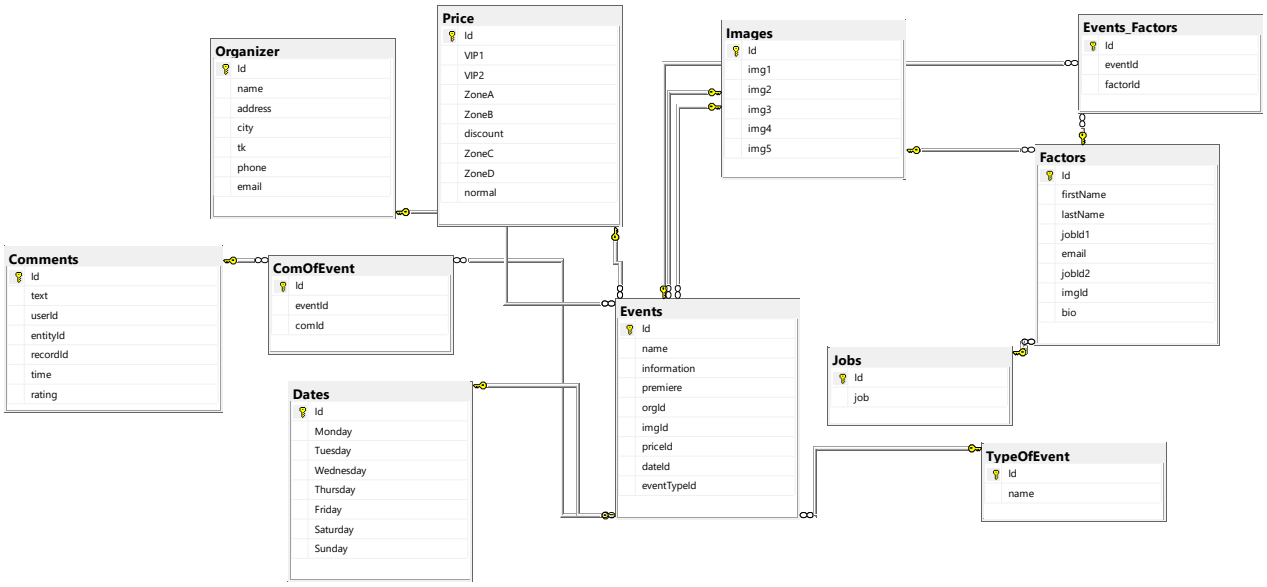
Ξενόγλωσσος Όρος	**Ελληνικός Όρος**
Framework	Σκελετός
Authentication	Αυθεντικοποίηση
Models	Μοντέλα
Controllers	Ελεγκτές
Client – Server	Πελάτης – Διακομιστής
Endpoint	Σημεία Τέλους

Routes	Διαδρομές
Components	Εξαρτήματα
Services	Υπηρεσίες
Social media	Μέσα Κοινωνικής Δικτύωσης
Google account	Λογαριασμός Google

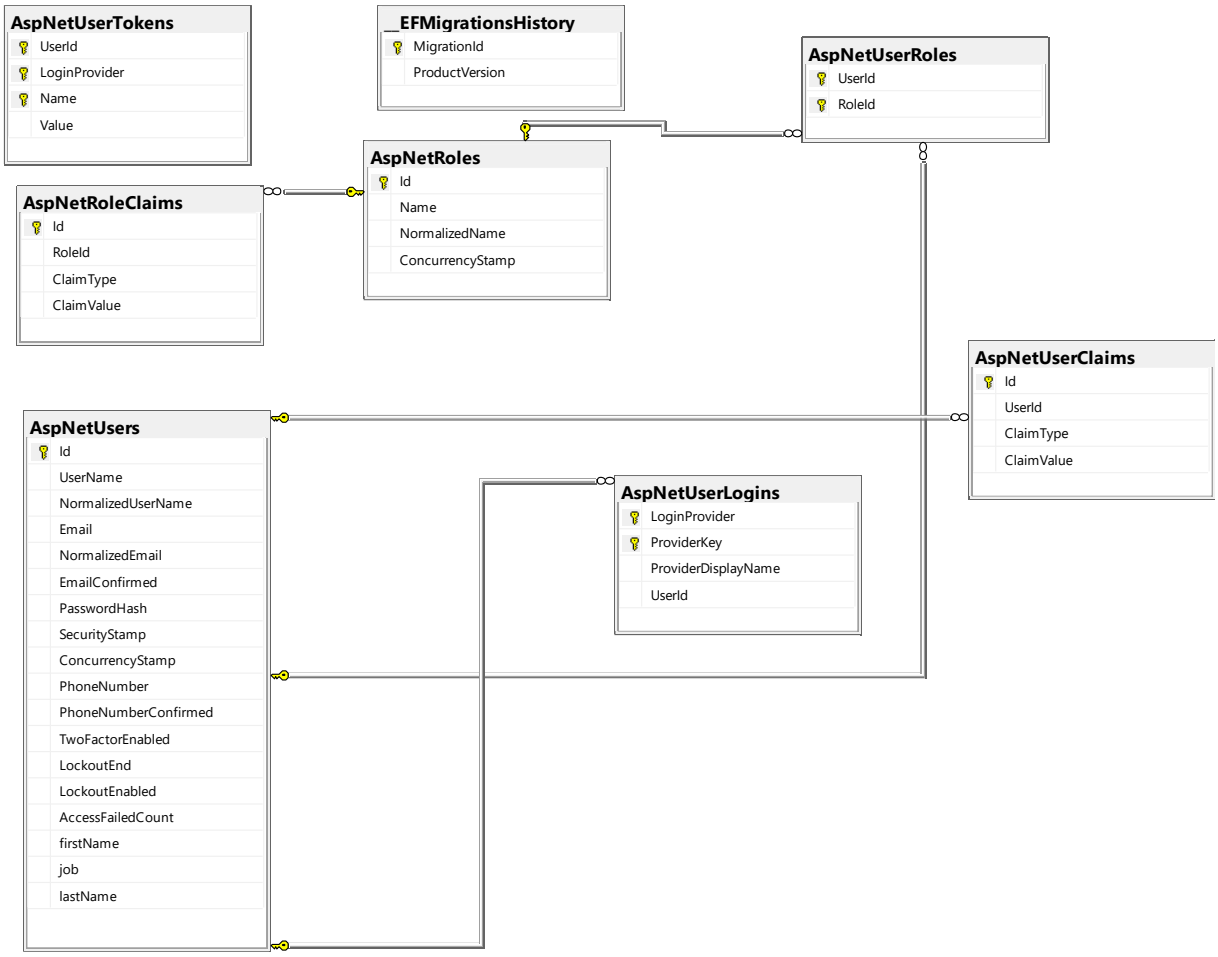
Πίνακας 26: Ορολογίες

Κεφάλαιο 8 - Παραρτήματα

8.1 Παράρτημα Α – Πίνακας SQL

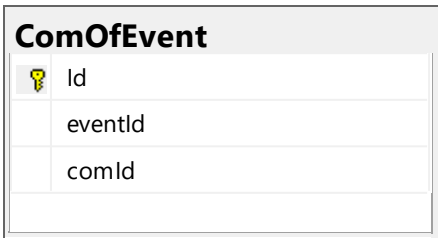


Διάγραμμα 6: Διάγραμμα Οντοτήτων Συσχετίσεων της ΒΔ

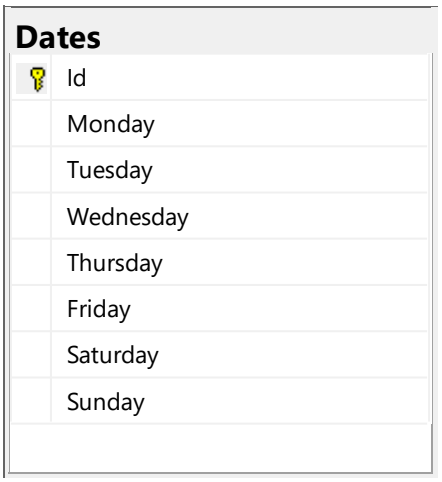


Διάγραμμα 7: ΔΟΣ Βάσης Δεδομένων (αυτόματη δημιουργία από ASP.NET Core Identity)

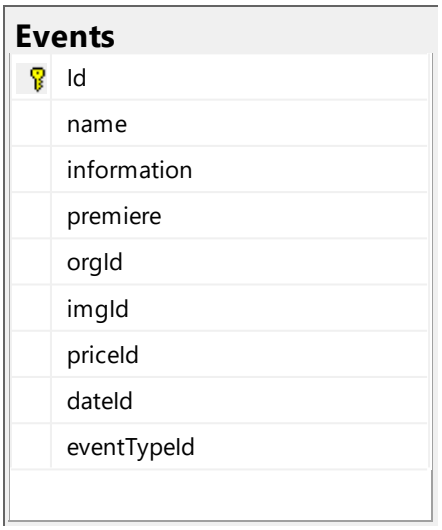
Εικόνα 85: Πίνακας Comments



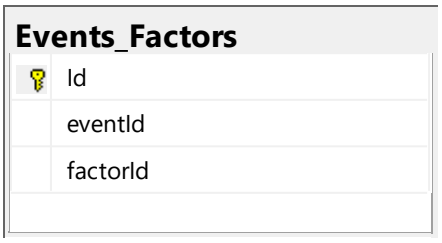
Εικόνα 86: Πίνακας ComOfEvent



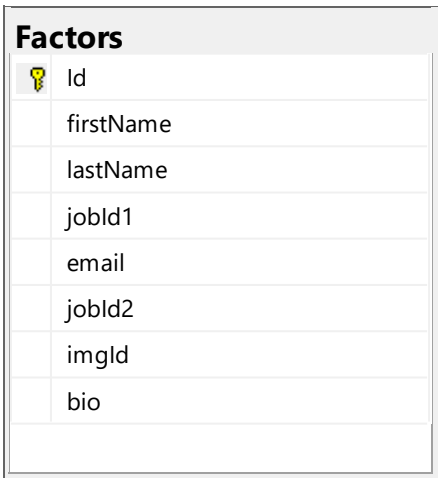
Εικόνα 87: Πίνακας Dates



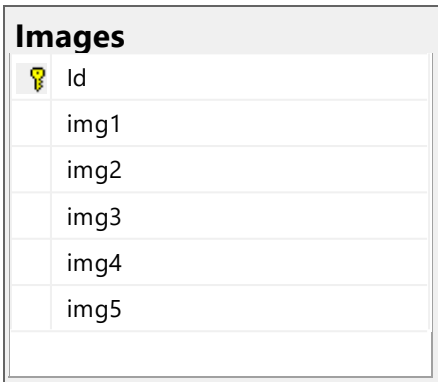
Εικόνα 88: Πίνακας Events



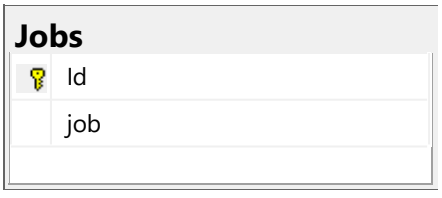
Εικόνα 89: Πίνακας Events_Factors



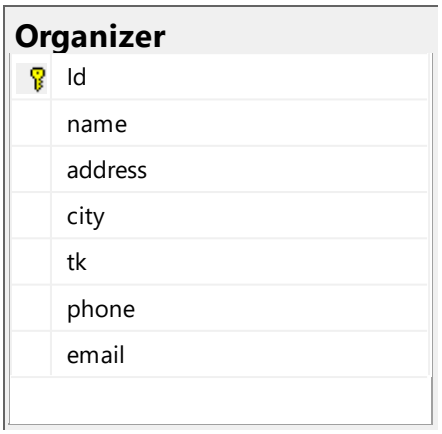
Εικόνα 90: Πίνακας Factors



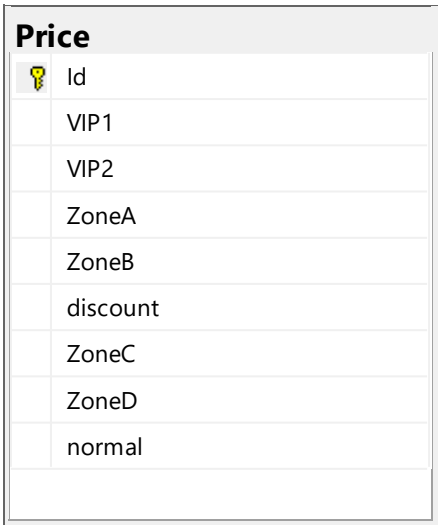
Εικόνα 91: Πίνακας Images



Εικόνα 92: Πίνακας Jobs



Εικόνα 93: Πίνακας Organizer



Εικόνα 94: Πίνακας Price

Εικόνα 95: Πίνακας TypeOfEvent

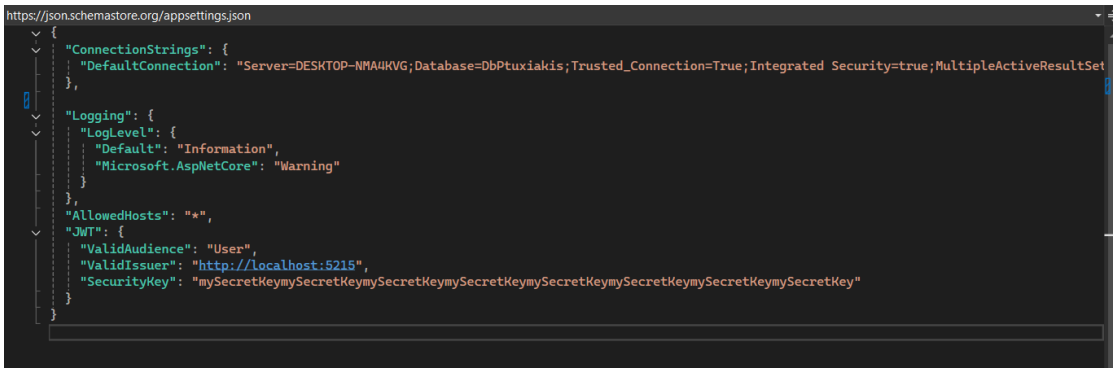
8.2 Παράρτημα Β – Κώδικας Backend

```
namespace DemoPtux_v2._1.Data
{
    16 references
    public class ApplicationDbContext : IdentityDbContext<ApplicationUser>
    {
        0 references
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) : base(options)
        {
        }
        0 references
        protected override void OnModelCreating(ModelBuilder builder)
        {
            base.OnModelCreating(builder);
        }
        9 references
        public DbSet<Events> Events { get; set; }
        5 references
        public DbSet<Jobs> Jobs { get; set; }
        1 reference
        public DbSet<Dates> Dates { get; set; }
        1 reference
        public DbSet<Events_Factors> Events_Factors { get; set; }
        6 references
        public DbSet<Factors> Factors { get; set; }
        9 references
        public DbSet<Images> Images { get; set; }
        5 references
        public DbSet<Organizer> Organizer { get; set; }
        1 reference
        public DbSet<Price> Price { get; set; }
        3 references
        public DbSet<TypeOfEvent> TypeOfEvent { get; set; }
        5 references
        public DbSet<Comments> Comments { get; set; }
        5 references
        public DbSet<ComOfEvent> ComOfEvent { get; set; }
    }
}
```

Εικόνα 96: ApplicationDbContext

```
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;

namespace DemoPtux_v2._1.Models.DBEntities
{
    7 references
    public class Events
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        11 references
        public int Id { get; set; }
        [Column(TypeName = "nvarchar(MAX)")]
        6 references
        public string name { get; set; }
        [Column(TypeName = "nvarchar(MAX)")]
        1 reference
        public string information { get; set; }
        [Column(TypeName = "datetimeoffset(7)")]
        1 reference
        public DateTimeOffset premiere { get; set; }

        [Column(TypeName = "int")]
        5 references
        public int orgId { get; set; }
        [ForeignKey("orgId")]
        0 references
        public Organizer Organizer { get; set; }

        [Column(TypeName = "int")]
        6 references
        public int imgId { get; set; }
        [ForeignKey("imgId")]
        0 references
        public Images Images { get; set; }

        [Column(TypeName = "int")]
        1 reference
        public int priceId { get; set; }
```

Εικόνα 97: Κλάση Events από DBEntities



Εικόνα 98: appsetings.json

```
var configuration = builder.Configuration;
// Add services to the container.
builder.Services.AddControllers();

// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
// For Entity Framework
builder.Services.AddDbContext<ApplicationDbContext>(options =>
options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

// For Identity
builder.Services.AddIdentity<ApplicationUser, IdentityRole>()
    .AddEntityFrameworkStores<ApplicationDbContext>()
    .AddDefaultTokenProviders();

// Adding Authentication
builder.Services.AddAuthentication(options =>
{
    options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
    options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
    options.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
}).AddJwtBearer(options =>      // Adding Jwt Bearer
{
    options.SaveToken = true;
    options.RequireHttpsMetadata = true;
    options.TokenValidationParameters = new TokenValidationParameters()
    {
        ValidateIssuerSigningKey = true,
        ValidateIssuer = true,
        ValidateLifetime = true,
        ValidateAudience = true,
        ValidAudience = configuration["JWT:ValidAudience"],
        ValidIssuer = configuration["JWT:ValidIssuer"],
        IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(configuration["JWT:SecurityKey"])),
        ClockSkew = TimeSpan.FromHours(1)
    };
});
```

Εικόνα 99: Program.cs

```
[HttpPost]
[Route("Register")]
0 references
public async Task<IActionResult> Register([FromBody] RegisterModel register)
{
    var userExist = await userManager.FindByNameAsync(register.username);
    if (userExist != null)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, new Response { Status = "Error", Message = "User not exist" });
    }
    ApplicationUser identityUser = new ApplicationUser();
    identityUser.Email = register.Email;
    identityUser.SecurityStamp = Guid.NewGuid().ToString();
    identityUser.UserName = register.username;
    identityUser.firstName = register.firstName;
    identityUser.lastName = register.lastName;
    identityUser.job = register.job;
    var result = await userManager.CreateAsync(identityUser, register.password);
    if (!result.Succeeded)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, new Response { Status = "Error", Message = "User creation faild" });
    }
    if(!await roleManager.RoleExistsAsync(UserRole.Admin))
    {
        await roleManager.CreateAsync(new IdentityRole(UserRole.Admin));
    }
    if(!await roleManager.RoleExistsAsync(UserRole.User))
    {
        await roleManager.CreateAsync(new IdentityRole(UserRole.User));
    }
    if (await roleManager.RoleExistsAsync(UserRole.Admin))
    {
        await userManager.AddToRoleAsync(identityUser, UserRole.Admin);
    }
    _username = identityUser.Email;
    return Ok(new Response { Status = "Success", Message = "User Created Successfully" });
}
```

Εικόνα 100: Authentication Controller I

```
[HttpPost]
[Route("Login")]
0 references
public async Task<IActionResult> Login([FromBody] LoginModel login)
{
    var user = await userManager.FindByNameAsync(login.username);

    if (user != null && await userManager.CheckPasswordAsync(user, login.password))
    {
        //nameOfUser = user.ToString();
        var userRoles = await userManager.GetRolesAsync(user);
        var authClaims = new List<Claim>
        {
            new Claim(ClaimTypes.Name, user.UserName),
            new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString())
        };

        foreach(var userrole in userRoles)
        {
            authClaims.Add(new Claim(ClaimTypes.Role, userrole));
        }

        var authSigninKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(_configuration["JWT:SecurityKey"]));
        var token = new JwtSecurityToken(
            issuer: _configuration["JWT:ValidIssuer"],
            audience: _configuration["JWT:ValidAudience"],
            expires: DateTime.Now.AddHours(5),
            claims: authClaims,
            signingCredentials: new SigningCredentials(authSigninKey, SecurityAlgorithms.HmacSha256)
        );
        if (User.Identity.IsAuthenticated)
```

Εικόνα 101: Authentication Controller II

```
[HttpGet("GetEvents/{eventTypeId}")]
0 references
public async Task<ActionResult<IEnumerable<EventByType>>> GetEvents(int eventTypeId)
{
    var events = from e in _context.Events
                 join d in _context.TypeOfEvent on eventTypeId equals d.Id
                 join o in _context.Organizer on e.orgId equals o.Id
                 join i in _context.Images on e.imgId equals i.Id
                 where e.eventTypeId == eventTypeId
                 select new EventByType
                 {
                     idEvent = e.Id,
                     name = e.name,
                     type = d.name,
                     organizerName = o.name,
                     image = i.img1
                 };
    if (events.Count() > 0)
    {
        return Ok(events);
    }
    else
    {
        return BadRequest("Δεν υπάρχουν εκδηλώσεις.");
    }
}
```

Εικόνα 102: Events Controller I

```
[HttpGet("SearchEvent")]
0 references
public async Task<ActionResult<IEnumerable<EventByType>>> SearchEvents(string text)
{
    List<Events> events = new List<Events>();
    List<EventByType> _events = new List<EventByType>();

    if (text == null || text == "")
    {
        EventByType events1 = new EventByType();
        events1.name = "Δεν υπάρχουν εγγραφές";
        _events.Add(events1);
        return BadRequest("Δεν υπάρχουν εγγραφές!");
    }
    else
    {
        events = _context.Events
            .Where(e => e.name.Contains(text)).ToList();

        foreach (var i in events)
        {
            var _query = (from e in _context.Events
                          join o in _context.Organizer on e.orgId equals o.Id
                          join _img in _context.Images on e.imgId equals _img.Id
                          where e.Id == i.Id
                          select new EventByType
                          {
                              idEvent = e.Id,
                              name = e.name,
                              organizerName = o.name,
                              image = _img.img1
                          }).ToList();
            EventByType _eventByType = new EventByType();
            if (_query.Count() > 0)
            {
                _eventByType.idEvent = _query[0].idEvent;
                _eventByType.name = _query[0].name;
                _eventByType.organizerName = _query[0].organizerName;
                _eventByType.image = _query[0].image;
                _events.Add(_eventByType);
            }

        }
    }
```

Εικόνα 103: Events Controller II

8.3 Παράρτημα Γ – Κώδικας Frontend

```
export class AuthService {
  constructor(private http: HttpClient, private router: Router) {}
  private baseUrl: string = "http://localhost:5215/api/Authentication/";


  signIn(loginObj: any) {
    return this.http.post<any>('http://localhost:5215/api/Authentication/Login', loginObj);
  }
  signUp(signUpObj: any) {
    return this.http.post<any>('http://localhost:5215/api/Authentication/Register', signUpObj);
  }

  signOut() {

    this.http.get<any>('http://localhost:5215/api/Authentication/Logout').subscribe({
      next: (res) => {
        if(res && res.status === true){
          localStorage.clear();
          alert("logout!");
          this.router.navigate(['signin']);
        }
      }, error: (err) => {
        alert("not logout!");
      }
    })
  }
  storeToken(tokenValue: string) {
    localStorage.setItem('token', tokenValue)
  }

  getToken(){
    return localStorage.getItem('token')
  }
```

Εικόνα 104: Authentication Service

```
export class SigninComponent implements OnInit{
  loginForm!: FormGroup;
  constructor(private fb: FormBuilder, private auth: AuthService, private router: Router, public http: HttpClient) {}

  ngOnInit(): void {
    this.loginForm = this.fb.group({
      username: ['', Validators.required],
      password: ['', Validators.required]
    })
  }
  onSubmit(){
    if(this.loginForm.valid) {
      console.log(this.loginForm.value)
      //db obj
      this.auth.signIn(this.loginForm.value)
      .subscribe({
        next:(res)=>{
          alert("Επιτυχημένη σύνδεση!");
          this.loginForm.reset();
          this.auth.storeToken(res.token);
          this.router.navigate(['homepage']);
        },
        error:(err)=>{
          alert("Λανθασμένα Στοιχεία!")
        }
      })
    } else {
      console.log("Λανθασμένα Στοιχεία!")
      //error
      ValidateForm.validateAllFormFields(this.loginForm)
      alert("Λανθασμένα Στοιχεία!")
    }
  }
```

Εικόνα 105: SignInComponent

```
export class SignupComponent implements OnInit{
  signUpForm!: FormGroup;
  public jobs: any = {};
  id: number | null = null;
  constructor(private http: HttpClient, public auth: AuthService, private fb: FormBuilder, private route:ActivatedRoute) {}

  ngOnInit(): void {
    this.signUpForm = this.fb.group({
      email: ['', Validators.required],
      username: ['', Validators.required],
      password: ['', Validators.required],
      firstName: ['', Validators.required],
      lastName: ['', Validators.required],
      job: ['', Validators.required]
    });
    this.route.paramMap.subscribe(i => {
      this.id = +i.get('id')!;
      this.fetchDataJobs();
    });
  }

  fetchDataJobs(){
    const apiUrl = 'http://localhost:5215/api/Professionals/AllJobs';
    this.http.get(apiUrl).subscribe(
      (resp:any) => {
        console.log(resp);
        this.jobs = resp;
      }
    )
  }

  onSignUp() {
    if(this.signUpForm.valid) {

      this.auth.signUp(this.signUpForm.value)
      .subscribe({
        next:(res=>{
```

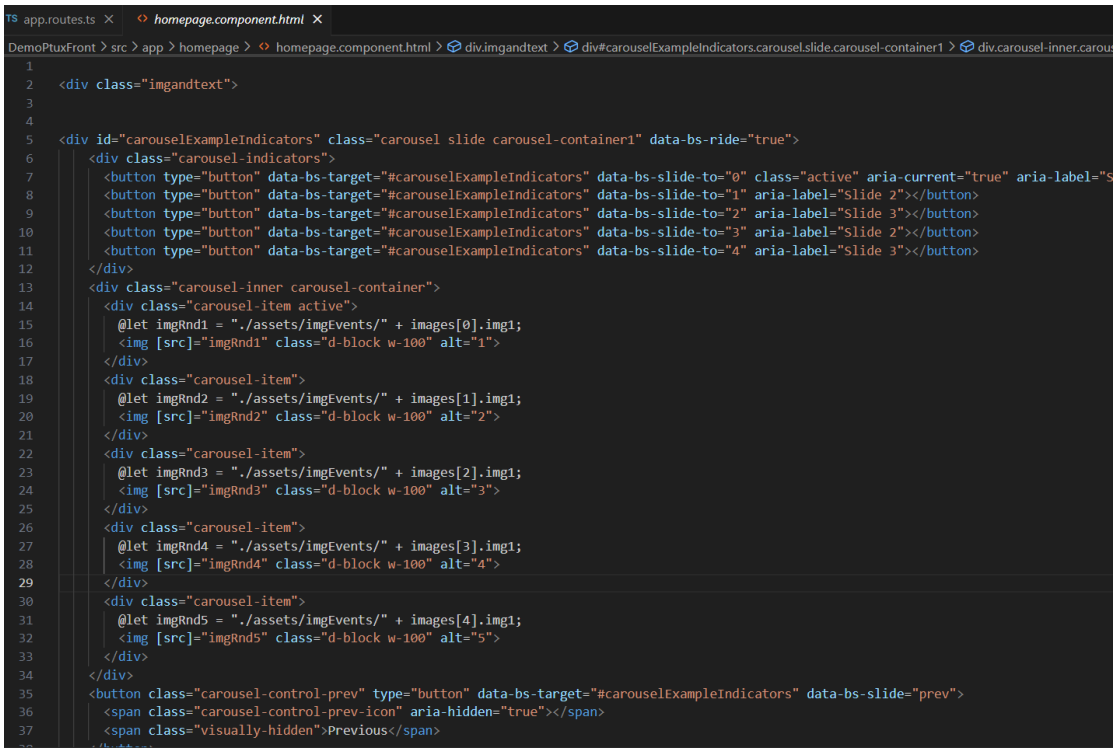
Εικόνα 106: SignUpComponent

app.routes.ts

DemoPtuxFront > src > app > app.routes.ts > ...

```
1   import { Routes } from '@angular/router';
2   import { SigninComponent } from './signin/signin.component';
3   import { SignupComponent } from './signup/signup.component';
4   import { HomepageComponent } from './homepage/homepage.component';
5   import { EventsComponent } from './events/events.component';
6   import { TestComponent } from './test/test.component';
7   import { ProfessionalsComponent } from './professionals/professionals.component';
8   import { InfoEventComponent } from './info-event/info-event.component';
9   import { InfoFactorComponent } from './info-factor/info-factor.component';
10  import { authGuard } from './guards/auth.guard';
11  import { HistoryComponent } from './history/history.component';
12  import { AboutComponent } from './about/about.component';
13
14  export const routes: Routes = [
15      {path: '', redirectTo:'/homepage', component:HomepageComponent},
16      {path: 'homepage', component:HomepageComponent},
17      {path: 'signin', component:SigninComponent},
18      {path: 'signup', component:SignupComponent},
19      {path: 'events/:id', component:EventsComponent},
20      {path: 'professionals/:id', component:ProfessionalsComponent},
21      {path: 'infoEvent/:id', component:InfoEventComponent},
22      {path: 'infoFactor/:id', component:InfoFactorComponent},
23      {path: 'history', component:HistoryComponent},
24      { path: 'about', component:AboutComponent},
```

Εικόνα 107: app.routes.ts



Εικόνα 108: Homepage.component.ts