



Πανεπιστήμιο Πειραιώς
Σχολή Τεχνολογιών Πληροφορικής και Τηλεπικοινωνίων
Τμήμα Ψηφιακών Συστημάτων

Επίπεδο: Προπτυχιακό Πρόγραμμα Σπουδών

Μάθημα : Διαδικτυακός Προγραμματισμός

Επιβλέπον Καθηγητής: Μαυρογιώργου Αργυρώ

Όνοματεπώνυμο:	A.M.:
Κυριακόπουλος Ζαχαρίας	E19076
Κουρέτας Παναγιώτης	E19072

Πειραιάς
25/06/2023

Περιεχόμενα

Ενότητα 1: Απαιτήσεις Εφαρμογής	1
Επισκόπιση Ιστοτόπου	1
Στόχοι προς επίτευξη	1
Φάσεις και Ορόσημα	2
Απαιτήσεις Χρηστών	3
User Personas	6
User Stories	8
Visual Sitemap	10
Wireframes	10
Mockup	11
Καταγραφή λειτουργικοτήτων του ηλεκτρονικού καταστήματος και πιθανών υποθέσεων τους	12
Ενότητα 2: Σύγχρονες Τάσεις Διαδικτυακών Εφαρμογών	13
1 ^η Τάση: JavaScript Frameworks	13
2 ^η Τάση: Responsive Web Design	16
3 ^η Τάση: Dark Mode	17
Ενότητα 3: Τεχνολογίες Εφαρμογής	18
Καταγραφή Τεχνολογικής Στοίβας	18
Ενότητα 4 ^η : Εγχειρίδιο Χρήσης Εφαρμογής	21
Ανάλυση Βάσης Δεδομένων	21
Ανάλυση Media Queries	25
Οδηγίες Εγκατάστασης και Εκτέλεσης	26
Ανάλυση Λειτουργιών και Αλληλεπίδραση με Βάση Δεδομένων	27

Ενότητα 1: Απαιτήσεις Εφαρμογής

Επισκόπιση Ιστοτόπου

Ο ιστότοπος μας αφορά μια καμπάνια του οργανισμού Tech-Ro Foundation μέσω της οποίας θα πραγματοποιούνται σεμινάρια τεχνολογικής εκμάθησης. Μέσω της διαδικτυακής εφαρμογής, ο κάθε χρήστης θα μπορεί να διαβάσει και να ενημερωθεί για τα σεμινάρια της καμπάνιας και σε περίπτωση που επιθυμεί να συμμετάσχει θα μπορεί να κλείσει εισιτήριο του για την αντίστοιχη εκδήλωση.

Στόχοι προς επίτευξη

Οι στόχοι του ιστοτόπου είναι οι εξής:

- Απόκτηση 20% περισσοτέρων μηνιαίων επισκεπτών.
- Απόκτηση 5% περισσοτέρων μηνιαίων επισκεπτών που κλείνουν εισιτήριο για την παρακολούθηση τουλάχιστον ενός σεμιναρίου.
- Βελτίωση εμπειρίας χρήστη για να αυξηθούν οι δυνητικοί πελάτες κατά 5%
- Παροχή πληροφοριών σχετικά με το περιεχόμενο της καμπάνιας
- Παροχή πληροφοριών για την πραγματοποίηση του κάθε σεμιναρίου (θέμα, διαθεσιμότητα, τοποθεσία, ημερομηνία)
- Παροχή πληροφοριών σχετικά με την ιστορία και τα επιτεύγματα του οργανισμού Tech-Ro Foundation
- Παροχή υπηρεσιών εξυπηρέτησης πελατών

Φάσεις και Ορόσημα

Για να διασφαλιστεί ότι το έργο κατασκευής του ιστοτόπου μας είναι ρεαλιστικό και παραδοτέο, το χωρίσαμε σε διακριτές φάσεις με συγκεκριμένα ορόσημα και προθεσμίες που ορίζονται για κάθε φάση ως εξής:

Φάση 1: Έρευνα & Ανάλυση

- Ορόσημο 1: Κατανόηση των επιχειρηματικών στόχων και των αναγκών των χρηστών. (από 1/5 μέχρι 5/5)
- Ορόσημο 2: Καθορισμός της λειτουργικότητας και του περιεχομένου Κατανόηση της λειτουργικότητας που απαιτείται και του περιεχομένου που θα παρουσιάζεται στην ιστοσελίδα. (από 6/5 μέχρι 10/5)

Φάση 2: Αρχιτεκτονική & Wireframes

- Ορόσημο 3: Ανάπτυξη Αρχιτεκτονικής Ιστοσελίδας Καθορισμός δομής πλοήγησης του ιστότοπου. (από 11/5 μέχρι 13/5)
- Ορόσημο 4: Δημιουργία Sitemap, Wireframe, mockup (από 13/5 μέχρι 15/5)

Φάση 3: Ανάπτυξη

- Ορόσημο 5: Front-end Development Μετάφραση του τελικού σχεδίου σε κώδικα και διασφάλιση ότι τηρεί το σχέδιο. (από 16/5 μέχρι 22/6)
- Ορόσημο 6: Back-end Development Ρύθμιση του server, των βάσεων δεδομένων και διασφάλιση των λειτουργιών ηλεκτρονικού εμπορίου του ιστότοπου (όπως αγορές, καλάθι και σύστημα πληρωμής). (από 16/5 μέχρι 22/6)

Φάση 4: Δοκιμή & Εκκίνηση

- Ορόσημο 7: Λειτουργικός Έλεγχος Έλεγχος της λειτουργικότητας του ιστότοπου, συμπεριλαμβανομένων των φορμών, της διαδικασίας ολοκλήρωσης αγοράς και άλλων διαδραστικών στοιχείων. (από 23/6 μέχρι 24/6)
- Ορόσημο 8: Δοκιμή εμπειρίας χρήστη Έλεγχος της χρηστικότητας και της προσβασιμότητας (από 24/6 μέχρι 25/6)

Απαιτήσεις Χρηστών

Για να καταγράψουμε τις απαιτήσεις των χρηστών της ιστοσελίδας μας τόσο από πλευράς τελικού χρήστη όσο και από πλευράς διαχειριστή, δημιουργήσαμε το παρακάτω ερωτηματολόγιο το οποίο βλέπετε και επίσης θα μπορείτε να βρείτε και στον παρακάτω σύνδεσμο:

- <https://forms.gle/Qc7GMyb58jD9VX5R9>

Ερωτηματολόγιο καταγραφής απαιτήσεων χρηστών

Περιγραφή φόρμας

Πως προτιμάτε να πραγματοποιείτε τις συνελλαγές σας;

- Προτιμώ τις ηλεκτρονικές συνελλαγές
- Προτιμώ τις συνελλαγές δια ζώσης
- Προτιμώ τις συνελλαγές από το τηλέφωνο
- Προτιμώ τις συνελλαγές με κάποιον άλλο τρόπο (αναφέρτε ποιον)

⋮

Θα προτιμούσατε να έχετε λογαριασμό στην ιστοσελίδα μας ή για οτιδήποτε πληροφορία θελήσετε να επικοινωνείται μαζί μας τηλεφωνικά;

- Θα ήθελα να έχω τον δικό μου λογαριασμό
- Θα ήθελα να επικοινωνώ μαζί σας μόνο τηλεφωνικά
- Θα ήθελα να μπορώ να τα κάνω και τα δύο

Με ποιον τρόπο θα θέλατε να πραγματοποιείτε την πληρωμή των συνελαγών σας;

- Μέσω τραπεζικής κατάθεσης
- Μέσω πιστωτικής ή χρεωστικής κάρτας
- Με κάποιον άλλο τρόπο (αναφέρετε ποιον)

Με ποιον τρόπο θα θέλατε να πραγματοποιείται η αξιολόγηση των εκδηλώσεων της καμπάνιας;

- Ηλεκτρινική αξιολόγηση μέσα από τον ιστότοπο
- Ηλεκτρινική αξιολόγηση μέσα από κάποιον τρίτο ιστότοπο
- Άλλος τρόπος αξιολόγησης (αναφέρετε ποιον)

Σε περίπτωση που έχετε τον πρωσοπικό σας λογαριαμό, με ποιον τρόπο θα θέλατε να επεξεργάζεστε τα στοιχεία σας;

- Κατευθείαν από τον ισότοπο
- Μέσω τηλεφώνου και της εξυπηρέτησης πελατών
- Άλλος τρόπος επεξεργασίας (αναφέρετε ποιον)

Από ποιές συσκευές θα θέλατε να μπορείτε να έχετε πρόσβαση στον ιστότοπό μας;

- Smarthphone
- Tablet
- Desktop / Laptop
- Όλα τα παραπάνω

Σε περίπτωση που ήσασταν διαχειριστής της ιστοσελίδας με ποιον τρόπο θα προτιμούσατε να επεξεργάζεστε τις υπάρχουσες εκδηλώσεις/χρήστες/ πληροφορίες για την καμπάνια;

- Ηλεκτρονικά μέσω της ιστοσελίδας από ένα εξελιγμένο admin dashboard
- Μέσω τροποποίησης χειρόγραφων εγγράφων
- Άλλος τρόπος επεξεργασίας (αναφέρετε ποιον)

Απαιτήσεις τελικού χρήστη

- 1. Έχει την απαίτηση να μπορεί να τροποποιεί το καλάθι εκδηλώσεων του**
2. Έχει την απαίτηση να μπορεί να συνδέεται με ασφάλεια στον ιστοτόπο.
3. Έχει την απαίτηση να μπορεί να αναζητεί εκδηλώσεις με την ονομασία τους
4. Έχει την απαίτηση να μπορεί να βλέπει τα στοιχεία του και να τα τροποποιεί
5. Έχει την απαίτηση να μπορεί να βλέπει τις παραγγελίες του και να τις ακυρώνει
6. Έχει την απαίτηση να μπορεί να βλέπει πληροφορίες για την καμπάνια.
7. Έχει την απαίτηση να αξιολογεί τις εκδηλώσεις και να βλέπει τις αξιολογήσεις άλλων χρηστών.
8. Έχει την απαίτηση να βλέπει όλες τις προγραμματισμένες καμπάνιες καθώς και τις καμπάνιες που έχουν λήξει.
9. Έχει την απαίτηση να μπορεί να βλέπει το ιστορικό των εκδηλώσεων που έχει παρακολουθήσει
- 10. Έχει την απαίτηση να μπορεί να επισκέπτεται την ιστοσελίδα από διάφορες ηλεκτρονικές συσκευές. (smartphone, tablet, desktop)**

Απαιτήσεις Διαχειριστή

1. Έχει την απαίτηση να έχει πρόσβαση σε ασφαλή σελίδα σύνδεσης

2. Έχει την απαίτηση να μπορεί να βλέπει του γεγραμμένους χρήστες, να τους τροποποιεί ή και να τους διαγράψει

3. Έχει απαίτηση να μπορεί να βλέπει πληροφορίες της καμπάνιας, να τις τροποποιεί, να τις διαγράφει ή και να τις προσθέτει.

4. Έχει την απαίτηση να βλέπει τις εκδηλώσεις, να προσθέτει, να τροποποιεί και διαγράφει εκδηλώσεις.

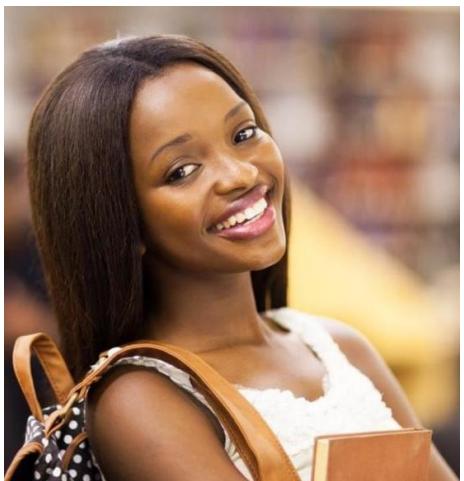
5. Έχει την απαίτηση να βλέπει πόσα εισιτήρια έχουν εκδοθεί ανά εκδήλωση.

6. Έχει την απαίτηση να μπορεί να επισκέπτεται την ιστοσελίδα από διάφορες ηλεκτρονικές συσκευές. (smartphone, tablet, desktop)

User Personas



Ο Sundar Pichai, είναι μεγαλοεπιχειρηματίας και πρόεδρος της Google. Πέρα από την επαγγελματική του θέση, είναι και πατέρας δύο παιδιών 18 και 15 χρονών. Έχοντας ο ίδιος μεγαλώσει σε δύσκολο περιβάλλον στην Ινδία, αλλά με 2 επιτυχημένους γονείς, ο Sundar θέλει να εγγράψει τα παιδιά του σε κάποια σεμινάρια αναφορικά με την τεχνητή νοημοσύνη και την επιρροή της στη καθημερινότητα.



Η Maya Angelou είναι πρωτοετής φοιτήτρια στο Πανεπιστήμιο. Έχει μεγαλώσει σε αρκετά εύπορο περιβάλλον, ωστόσο οι γονείς στης στα 18 της, της ζήτησαν να δουλέψει για να πληρώσει τα υλικά χρέη του Πανεπιστημίου. Η Maya, άριστη μαθήτρια στο τομέα του Web Development στο Πανεπιστήμιο της, καταφέρνει να δουλεύει ταυτόχρονα με την σχολή, ωστόσο ψάχνει συνεχώς σεμινάρια και εκδηλώσεις οι οποίες θα την βοηθήσουν να διευρύνει τις γνώσεις της και να αποκτήσει παραπάνω εμπειρία.



Ο Πατρίκιος Βατεμάνος είναι ένας εικοσιτετράχρονος ο οποίος μόλις άφησε τις σπουδές του στα οικονομικά με σκοπό να ασχοληθεί με τον κλάδο της πληροφορικής και της τεχνολογίας μια και θεωρεί ότι εκεί θα έχει ένα καλύτερο μέλλον. Έτσι λοιπόν ο Πατρίκιος θέλει να παρακολουθήσει διάφορα σεμινάρια τεχνολογικής εκμάθησης πάνω σε διαφορετικές θεματικές ενότητες ώστε να καταλάβει ποιος τομέας του αρέσει περισσότερο ώστε να εμβαθύνει στο μέλλον.

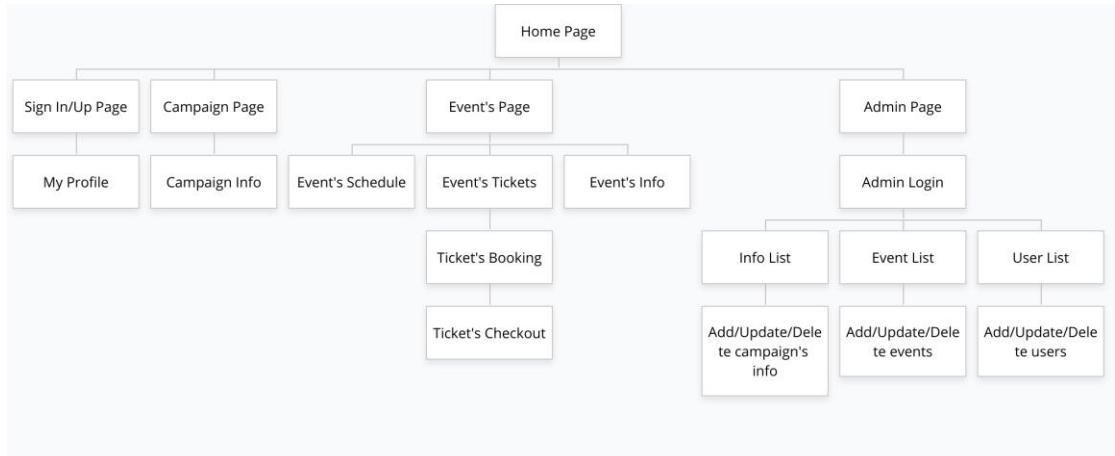
User Stories

1. Ως νέος δάσκαλος, θέλω να μπορώ να εγγράφομαι σε σεμινάρια τεχνικής κατάρτισης για την αύξηση της διαδραστικότητας του μαθήματος μου.
2. Ως δικηγόρος, θέλω να μπορώ να βλέπω οργανωμένα τις εκδηλώσεις που με αφορούν και την ημερομηνία τους, ώστε να μπορώ να οργανώσω αποδοτικά το πρόγραμμα μου.
3. Ως ηλικιωμένος, θέλω να μπορώ να συμμετέχω στις εκδηλώσεις της καμπάνιας, ώστε να εξοικειωθώ με τη χρήση των νέων τεχνολογιών.
4. Ως φοιτητής του Πανεπιστημίου Πειραιώς, θέλω να μπορώ να ενημερώνομαι για τις σφοδρές αλλαγές στον τομέα της τεχνητής νοημοσύνης καθώς και της επιρροής της σε μελλοντικά μου επαγγέλματα.
5. Ως άτομο που αναζητά εργασία σε εταιρίες πληροφορικής, θέλω να μπορώ να εξελιχτώ και να αποκτήσω νέες δεξιότητες ώστε να μπορώ να διεκδικήσω υψηλότερο μισθό.
6. Ως άνθρωπος που ανησυχεί για τους κακόβουλους χρήστες στο διαδίκτυο, θα ήθελα να μπορώ να επεξεργάζομαι τα στοιχεία μου ώστε να αλλάζω πολύ συχνά κωδικό στον προσωπικό μου λογαριασμό.
7. Ως μέλος μιας ανώνυμης οργάνωσης η οποία λαμβάνει μέρος σε διάφορα events και τα αξιολογεί, θέλω να μπορώ να αξιολογώ το κάθε σεμινάριο ώστε να βοηθήσω τον κόσμο που δεν έχει παρακολουθήσει ξανά να έχει μια γενική εικόνα.
8. Ως φοιτητής του Πανεπιστημίου Πειραιώς, θέλω να μπορώ να ενημερώνομαι για τις σφοδρές αλλαγές στον τομέα της τεχνητής νοημοσύνης καθώς και της επιρροής της σε μελλοντικά μου επαγγέλματα.
9. Ως άτομο που βγαίνει στην αγορά εργασίας, θέλω να μπορώ να παρακολουθήσω τα σεμινάρια τεχνολογικής εκμάθησης για να μπορώ να εμπλουτίσω το βιογραφικό μου.
10. Ως άτομο που ασχολείται με την τεχνολογία, θέλω η ιστοσελίδα για της καμπάνιας να είναι φιλική προς τους τελικούς χρήστες, γιατί μου αρέσουν πολύ οι όμορφες ιστοσελίδες οι οποίες σου προσελκύουν το ενδιαφέρον.

11. Ως φωτογράφος του ιδρύματος Tech-Ro, θέλω να μπορώ να βλέπω μέσω της ιστοσελίδας πληροφορίες σχετικά με τις ημερομηνίες, τις ώρες και τις τοποθεσίες των σεμιναρίων ώστε να μπορώ να οργανώσω κατάλληλα το πρόγραμμά μου.
12. Ως λάτρης των διαδικτυακών αγορών, θα ήθελα όταν κλείνω ένα εισιτήριο να μπορώ να πληρώνω με την πιστωτική μου κάρτα διαδικτυακά για να κερδίζω τραπεζικούς πόντους.
13. Ως άτομο που λατρεύει την οργάνωση, θα ήθελα να μπορώ όταν κλείνω ένα εισιτήριο να έχω την επιλογή να μου έρχονται τα στοιχεία του εισιτηρίου στο email μου ώστε να έχω πρόσβαση κάθε στιγμή.
14. Ως άτομο με μυωπία, θα ήθελα να μπορώ να βλέπω όλες τις διαθέσιμες θέσεις σε μια εκδήλωση που πρόκειται να παρακολουθήσω, για να μπορώ να επιλέξω μια θέση στις πρώτες σειρές ώστε να μπορώ να έχω την καλύτερη δυνατή εμπειρία.
15. Ως άτομο που δεν αυτοκίνητο, θέλω μέσω της ιστοσελίδας να μπορώ να βλέπω την τοποθεσία και την ώρα κάθε σεμιναρίου, για να μπορώ να σχεδιάσω την διαδρομή που θα ακολουθήσω με τα μέσα μαζικής μεταφοράς ώστε να παραβρεθώ στην εκδήλωση.
16. Ως ειδικευμένος εργαζόμενος στον τομέα της τεχνίτης νοημοσύνης, θέλω να μπορώ να αναζητώ τα σεμινάρια με λέξεις κλειδιά, γιατί θέλω να παρακολουθώ μόνο αυτά που με ενδιαφέρουν και όχι αυτά που αφορούν άλλους τομείς.
17. Ως προγραμματιστής, θέλω να μπορώ να κάνω την ιστοσελίδα να εμφανίζεται σε dark mode, για να μην κουράζονται τα μάτια μου.
18. Ως λάτρης της τεχνολογίας, θέλω να μπορώ να έχω πρόσβαση στην ιστοσελίδα της καμπάνιας από διάφορες συσκευές μιας και έχω στην κατοχή μου και tablet και smartphone και desktop.
19. Ως άτομο που δεν έχει ιδιαίτερη εξοικείωση με την τεχνολογία, θέλω να μπορώ να επικοινωνήσω με τον οργανισμό σε περίπτωση που θα έχω τυχόν απορίες ώστε να με βοηθήσουν.
20. Ως διαχειριστής, θα ήθελα να υπάρχει ένα feedback box ώστε κάθε τελικός χρήστης να μοιράζεται μαζί μας την εμπειρία του με την ιστοσελίδα, για να βελτιώνουμε συνεχώς την εμπειρίας χρήσης της ιστοσελίδας μας.

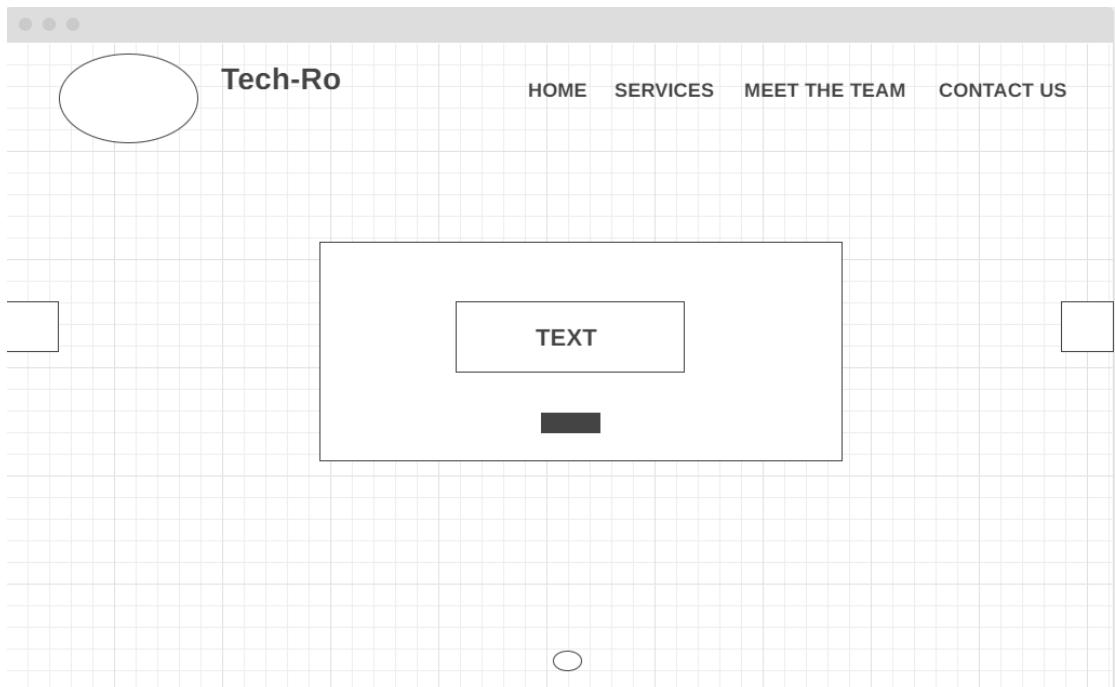
Visual Sitemap

Παρακάτω απεικονίζεται το Visual Sitemap της ιστοσελίδας μας:

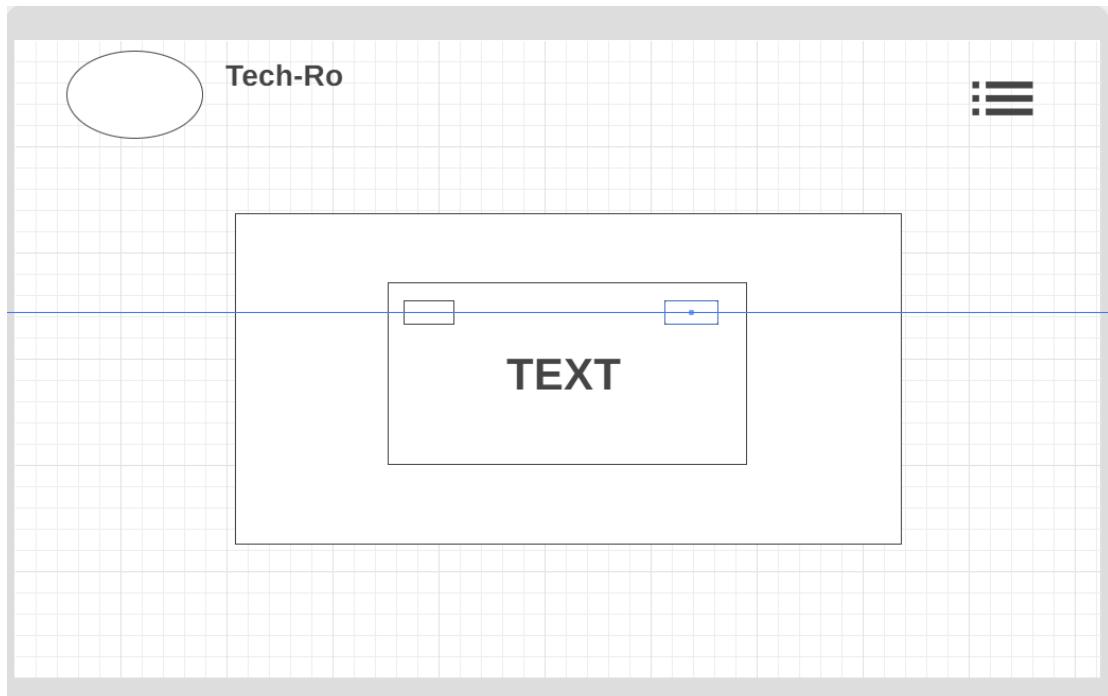


Wireframes

Παρακάτω απεικονίζονται τα wireframes της αρχικής σελλιδας του ιστοτόπου μας για desktop συσκευές:

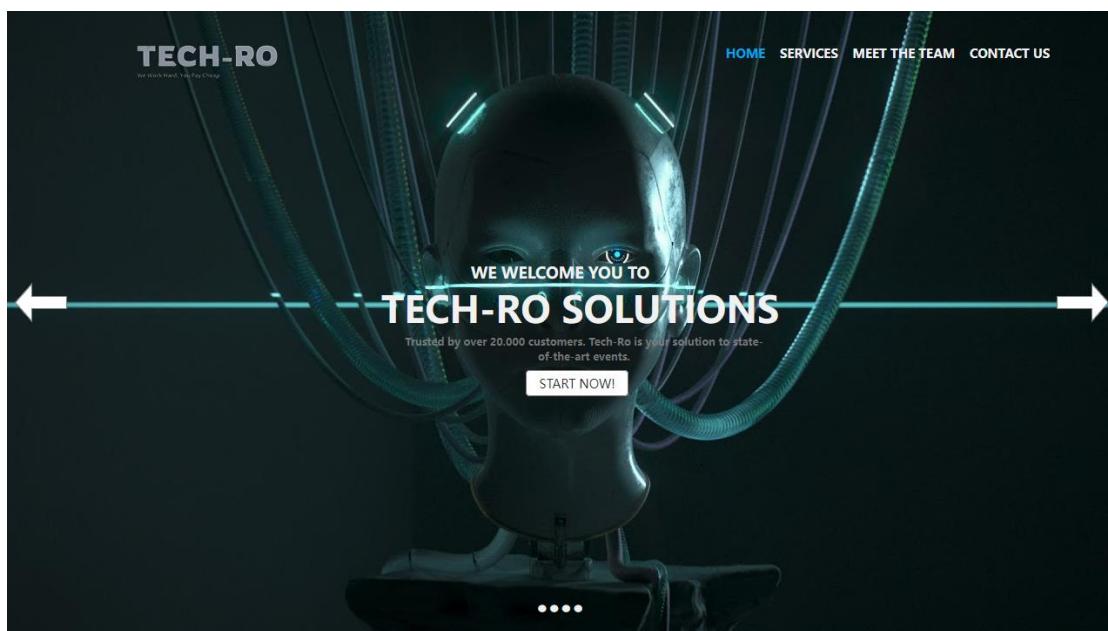


Παρακάτω απεικονίζονται τα wireframes της αρχικής σελίδας του ιστοτόπου μας για tablet και mobile συσκευές:



Mockup

Παρακάτω απεικονίζονται το mockup της αρχικής σελλιδας του ιστοτόπου μας για desktop συσκευές:



**Καταγραφή λειτουργικοτήτων του ηλεκτρονικού καταστήματος και πιθανών
υποθέσεων τους**

Οι λειτουργικότητες τις οποίες θα περιέχει η ιστοσελίδα θα είναι οι εξής:

- Δυνατότητα ηλεκτρονικής κράτησης εισιτηρίων
- Σελίδα πληροφοριών καμπάνιας
- Σελίδα σεμιναρίων καμπάνιας
- Φόρμα σύνδεσης τελικού χρήστη
- Φόρμα εγγραφής τελικού χρήστη
- Επιλογή τύπου εισιτηρίου κατά την κράτηση
- Επιλογή θέσης κατά την κράτηση
- Επιλογή αριθμού εισιτηρίων κατά την κράτηση
- Πύλες πληρωμών
- Δυνατότητα αποστολής εισιτηρίου μέσω email
- Ιστορικό σεμιναρίων
- Δυνατότητα αίτησης επαναποστολής στοιχείων κράτησης σεμιναρίου
- Λειτουργία αναζήτησης
- Δυνατότητα αναζήτησης εκδηλώσεων
- Πλήρης λειτουργικότητα σε όλες τις συσκευές
- Φόρμα σύνδεσης διαχειριστή
- Εξατομίκευση λογαριασμού
- Φόρμα σύνδεσης διαχειριστή
- Επεξεργασία εγγεγραμμένων χρηστών
- Επεξεργασία σεμιναρίων καμπάνιας
- Επεξεργασία πληροφοριών καμπάνιας
- Δυνατότητα Dark Mode
- Λειτουργία Contact Us
- Λειτουργία MailBox στο κομμάτι του διαχειριστή
- Αξιολόγηση Εμπειρίας
- Δυναμική προσθήκη νέων εκδηλώσεων

Οι πιθανές υποθέσεις για τις λειτουργικότητες του ηλεκτρονικού μας καταστήματος είναι οι εξής:

- Αν ο ισότοπος πρέπει να πληρεί συγκεκριμένη νομοθεσία ασφαλείας θα πρέπει να καταγραφεί
- Όλες οι ηλεκτρονικές συναλλαγές μέσω τραπεζικών συστημάτων πληρωμών θεωρούμε ότι είναι ασφαλείς
- Η υποστήριξη και η συντήρηση του ισοτόπου
- Πληρούνται όλες οι απαιτήσεις φιλοξενίας
- Υποστήριξη προγραμμάτων περιήγησης
- Προσβασιμότητα
- Ταχύτητα φόρτωσης σελίδας
- Ο ισότοπος πληρεί συγκεκριμένη νομοθεσία ασφαλείας

Ενότητα 2: Σύγχρονες Τάσεις Διαδικτυακών Εφαρμογών

1^η Τάση: JavaScript Frameworks

Η πρώτη από τις τάσεις των διαδικτυακών εφαρμογών που χρησιμοποιήθηκε είναι τα JavaScript Frameworks. Μερικοί από τους κύριους λόγους επιλογής ενσωμάτωσης της συγκεκριμένης τάσης είναι η αποδοτικότητα και η ταχύτητα που προσφέρουν οι έτοιμες βιβλιοθήκες της JavaScript, απλοποιώντας έτσι μερικές πολύπλοκες λειτουργίες, η επεκτασιμότητα που προσφέρει στην ιστοσελίδα καθώς και η συμβατότητα σε διαφορετικά προγράμματα περιήγησης και συσκευές. Μερικά από τα JavaScript που χρησιμοποιήθηκαν είναι:

- **AngularJS**

Το AngularJS είναι ένα framework το οποίο επεκτείνει τις δυνατότητες της HTML κάνοντας την εμπειρία του χρήστη πιο εύχρηστη και responsive. Στην περίπτωση μας μια από τις περιπτώσεις που χρησιμοποιήσαμε Angular είναι στη σελίδα εγγραφής ή σύνδεσης του χρήστη (sign-in-up-page.html). Στη συγκεκριμένη σελίδα θέλουμε να εμφανίζεται ανάλογο μήνυμα όταν ο χρήστης συνδέεται με σωστά ή λάθος στοιχεία και παρομοίως όταν προσπαθεί να εγγραφεί.

Παραδείγματα στον κώδικα:

```
</div>
<div class="col-md-6 col-lg-5" ng-controller='RegistrationController'>
  <h2 class="font-weight-bold text-5 mb-0">Register</h2>
  <form id="frmSignUp" ng-submit="registerUser()">
    <div class="row">
      <div class="form-group col">
        <label class="form-label text-color-dark text-3">Username <span class="text-color-danger">*</span></label>
        <input type="text" value="" class="form-control form-control-lg text-4 username" required>
      </div>
    </div>
    <div class="row">
      <div class="form-group col">
        <label class="form-label text-color-dark text-3">Password <span class="text-color-danger">*</span></label>
        <input type="password" value="" class="form-control form-control-lg text-4 password" required>
      </div>
    </div>
  </form>
  <div id="notification" class="alert alert-success" style="...">
    <strong><i class="far fa-thumbs-up"></i> Success!</strong> You have successfully registered. Redirecting...
  </div>
```

```
$http.post('/register', { username: username, password: password })
  .then(function (response) {
    console.log(response);
    // Display the success message
    showNotification();

    // Redirect after 3 seconds
    $timeout(function () {
      window.location.href = '/page-event-list.html';
    }, 3000);

    // This function will show the SUCCESS register notification.
    1 usage
    function showNotification() {
      const notification = document.getElementById('notification');
      notification.style.display = 'block';
    }
  })
```

Στην εφαρμογή:

Register

Username *

testuser

Password *

Your personal data will be used to support your experience throughout this website, to manage access to your account, and for other purposes described in our [privacy policy](#).

REGISTER

Success! You have successfully registered.

Redirecting...

Αντίστοιχα και για την λειτουργία login:

Login

Username *

testuser

Password *

LOGIN

⚠ Oh snap! Change a few things up and try submitting again.

- Check your username.
- Check your password.

Login

Username *

testuser

Password *

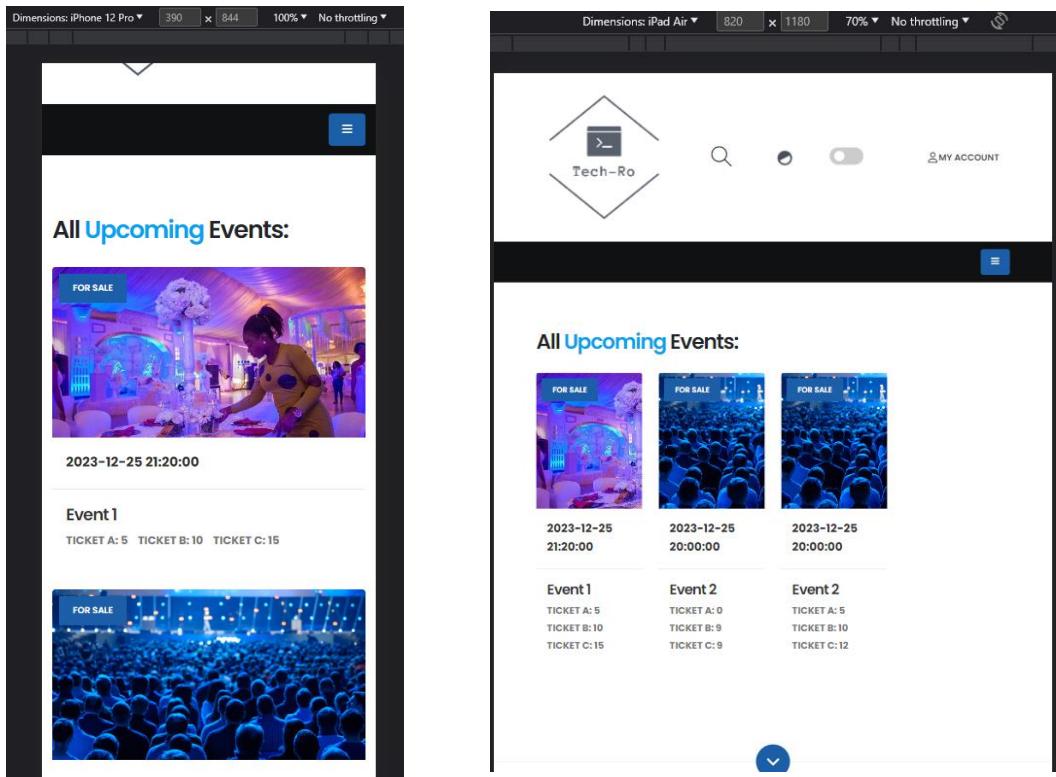
LOGIN

Success! You have successfully logged in. Redirecting...

2^η Τάση: Responsive Web Design

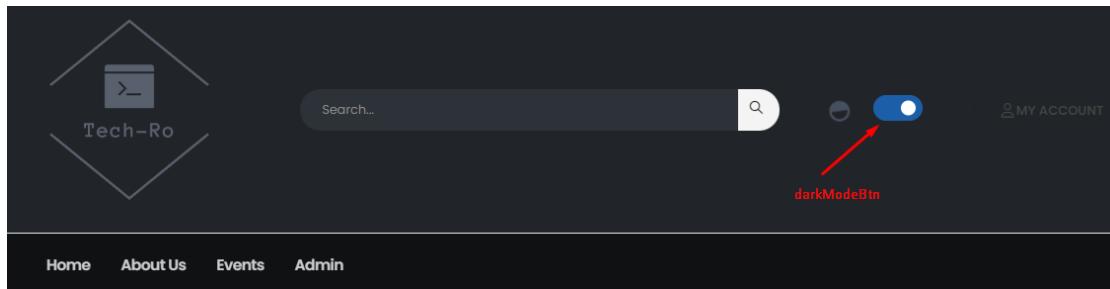
Η δεύτερη από τις τάσεις των διαδικτυακών εφαρμογών που χρησιμοποιήθηκε είναι το Responsive Web Design. Αυτή η τάση ενσωματώθηκε στην ιστοσελίδα μας γιατί θα βελτιώσει σημαντικά την εμπειρία χρήσης τόσο των τελικών χρηστών όσο και των διαχειριστών οι οποίοι θα μπορούν να έχουν πρόσβαση στην ιστοσελίδα μέσω διαφορετικών συσκευών χωρίς να αλλοιώνεται το περιεχόμενο. Έτσι δεν θα υπάρχει περιορισμός στο ποιος θα μπορεί να επισκεφτεί την σελίδα μας και κάθε χρήστης θα μπορεί να έχει μια εξίσου καλή εμπειρία περιήγησης. Το Responsive Web Design κυριαρχεί καθ' όλη την εφαρμογή. Μερικά παραδείγματα:

The screenshot displays a mobile-friendly website layout. At the top, there's a black navigation bar with white text containing links for Home, About Us, Events, and Admin. Below this is a decorative graphic consisting of two downward-pointing chevrons. The main content area features a heading 'All Upcoming Events:' followed by three event cards. Each card includes a thumbnail image, the event title, and its start date and time. The first card shows a person at a table with a 'FOR SALE' sign, titled 'Event 1' with the date '2023-12-25 21:20:00'. The second card shows a large crowd at night, titled 'Event 2' with the date '2023-12-25 20:00:00'. The third card is a map of the Acropolis area in Athens, titled 'Event 2' with the date '2023-12-25 20:00:00'. Below the event cards, there's a small section for 'Event 2' with ticket information: 'TICKET A: 5 TICKET B: 10 TICKET C: 12'. A blue circular arrow icon is located at the bottom center of the page.



3^η Τάση: Dark Mode

Η τρίτη από τις τάσεις των διαδικτυακών εφαρμογών που χρησιμοποιήθηκε είναι το Dark Mode. Οι λόγοι που προσθέσαμε το Dark Mode στην ιστοσελίδα μας είναι ότι πλέον το 2023 αποτελεί αναπόσπαστο κομμάτι κάθε διαδικτυακής εφαρμογής, βελτιώνει την εμπειρία των χρηστών, βοηθά στην μείωση καταπόνησης των ματιών και βελτιώνει την πτώση της μπαταρίας της συσκευής (στην περίπτωση χρήσης OLED panel στη συσκευή). Ο χρήστης έχει τη δυνατότητα να ενεργοποιεί ή απενεργοποιεί τη λειτουργεία dark mode από οποιαδήποτε σελίδα (user/admin). Παράδειγμα στο κώδικα και στην εφαρμογή:



```

<script>
$(document).ready(function() {
    // Get the saved dark mode state from LocalStorage, if it exists
    var darkMode = localStorage.getItem('darkMode');

    // If darkMode exists and is 'true', add the .dark class and check the switch
    if (darkMode === 'true') {
        $('html').addClass('dark');
        $('#darkModeBtn').prop('checked', true);
    }
    // event listener για το κουμπί στην κάθε σελίδα
    // Listen for switch state changes
    $('#darkModeBtn').change(function() {
        // If the switch is checked, add the .dark class and save the state
        if ($(this).is(':checked')) {
            $('html').addClass('dark');
            // localStorage.setItem('darkMode', 'true');
        }
        // If the switch is not checked, remove the .dark class and save the state
        else {
            $('html').removeClass('dark');
            // localStorage.setItem('darkMode', 'false');
        }
    });
});
</script>

```

Ενότητα 3: Τεχνολογίες Εφαρμογής

Καταγραφή Τεχνολογικής Στοίβας

Η τεχνολογική στοίβα η οποία χρησιμοποιήθηκε για την ανάπτυξη της ιστοσελίδας μας, είναι η MEAN δηλαδή MongoDB για NoSQL Database, ExpressJS Framework για Back-End, AngularJS Framework για Front-End και NodeJS για Cross-platform Server. Μερικά από τα βασικότερα χαρακτηριστικά της στοίβας τα οποία μας ώθησαν και στην επιλογή της είναι τα παρακάτω

Η στοίβα MEAN βασίζεται εξ ολοκλήρου σε JavaScript, επιτρέποντας στους προγραμματιστές να χρησιμοποιούν μια γλώσσα σε ολόκληρη την ιστοσελίδα και στο front-end αλλά και στο back-end.

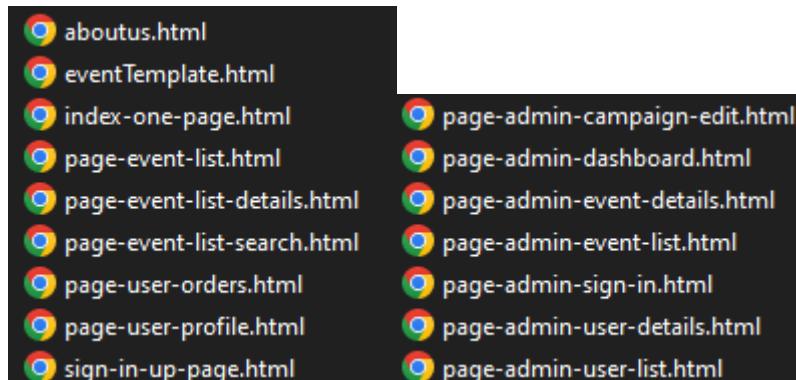
Η ανταλλαγή δεδομένων που βασίζεται σε JSON. Τα στοιχεία στοίβας MEAN (MongoDB, Express.js και Angular) χρησιμοποιούν JSON (JavaScript Object Notation) ως την τυπική μορφή για την ανταλλαγή δεδομένων. Αυτό απλοποιεί τη μεταφορά δεδομένων και την επικοινωνία μεταξύ των διαφορετικών επιπέδων της εφαρμογής.

Επίσης παρέχει επεκτασιμότητα και ευελιξία για τη δημιουργία εφαρμογών μικρής και μεγάλης κλίμακας. Το ευέλικτο μοντέλο δεδομένων της MongoDB επιτρέπει την εύκολη προσαρμογή στις μεταβαλλόμενες απαιτήσεις, ενώ το Node.js επιτρέπει τον αποτελεσματικό χειρισμό των ταυτόχρονων συνδέσεων, καθιστώντας το κατάλληλο για εφαρμογές υψηλής επισκεψιμότητας.

Οι Επιμέρους Τεχνολογίες που Χρησιμοποιήθηκαν είναι οι εξής:

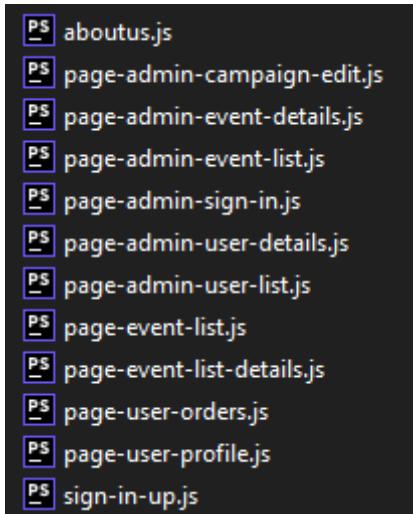
- HTML: η HTML χρησιμοποιήθηκε καθ' όλη τη διαδικτυακή εφαρμογή καθώς είναι γλώσσα που εμφανίζει περιεχόμενο στις σελίδες. Άρα κάθε σελίδα στην εφαρμογή χρησιμοποιεί HTML και το αρχείο τελειώνει με την επέκταση **.html**

Οι παρακάτω σελίδες χρησιμοποιούν HTML:



- Η CSS χρησιμοποιήθηκε για styling, στοίχιση και μορφοποίηση του content της HTML στην κάθε σελίδα καθώς και το responsiveness αυτων.
 - ⊕ theme.css: για χρήση arrangement για στοιχεία όλων των σελιδών
 - ⊕ theme-shop.css: χρήση styling για στοιχεία της εφαρμογής
 - ⊕ theme-elements.css: χρήση styling για στοιχεία των elements σε όλες τις σελίδες

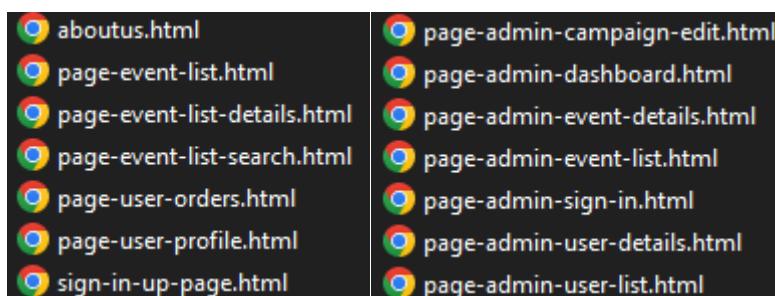
- Η JavaScript χρησιμοποιήθηκε για όλες τις λειτουργικότητες και αλληλεπιδράσεις με την βάση δεδομένων. Η χρήση JavaScript «σπάει» στα δύο. Η χρήση JavaScript στη πλευρά του διακομιστή (server-side) στην οποία αναπτύχθηκαν όλα τα routes (endpoints) για αλληλεπίδραση με την βάση δεδομένων και τα CRUD operations (Create,Read,Update,Delete). Το back-end κομμάτι της JavaScript έχει τοποθετηθεί σε ξεχωριστό φάκελο “back-end” και περιέχει τα παρακάτω αρχεία:



Τα ονόματα των αρχείων κάθε JavaScript αντιστοιχούν στην ίδια ονομασία της HTML σελίδας στην οποία χρησιμοποιούνται.

Οπότε το “aboutus.js” χρησιμοποιείται στη σελίδα “aboutus.html”, το “sign-in-up.js” χρησιμοποιείται στη σελίδα “sign-in-up.page.html” κλπ

Η χρήση JavaScript από την πλευρά του τελικού χρήστη (client) έγινε για παραμετροποίηση και «χρήση» των προαναφερθέντων endpoints στο “back-end”. Τα αρχεία HTML στα οποία χρησιμοποιήθηκε εμφωλευμένη χρήση JavaScript κώδικα είναι τα παρακάτω:



- jQuery, όπως και προηγουμένως με την JavaScript, χρησιμοποιήθηκε στο client-side κομμάτι της JavaScript για την διαχείριση HTML content. Λόγου χάρη, χρησιμοποιήθηκε στη σελίδα “page-event-list.html” για να εμφανίσει δυναμικά τα events ανάλογα την ημερομηνία τους στα Upcoming ή Expired Events. Όπως και προηγουμένως τα αρχεία στα οποία χρησιμοποιήθηκαν είναι τα παρακάτω:

 aboutus.html	 page-admin-campaign-edit.html
 page-event-list.html	 page-admin-dashboard.html
 page-event-list-details.html	 page-admin-event-details.html
 page-event-list-search.html	 page-admin-event-list.html
 page-user-orders.html	 page-admin-sign-in.html
 page-user-profile.html	 page-admin-user-details.html
 sign-in-up-page.html	 page-admin-user-list.html

- NodeJS, χρησιμοποιήθηκε για δημιουργία cross-platform περιβάλλον server, διαχείριση packages και δημιουργία endpoints.
- ExpressJS, χρησιμοποιήθηκε για δημιουργία localhost και διαχείριση των session των χρηστών και διαχειριστών.
- AngularJS χρησιμοποιήθηκε στη σελίδα “sign-in-up-page.html” για διαχείριση του HTML content ανάλογα την περίσταση κατά την διάρκεια sign-in ή registration.
- Bootstrap, χρησιμοποιήθηκε το Bootstrap framework για μορφοποίηση και responsiveness (grids & media queries) τα σχετικά αρχεία του bootstrap:

 bootpag	6/10/2023 6:49 PM	File folder
 bootstrap	6/10/2023 6:49 PM	File folder
 bootstrap-datepicker	6/10/2023 6:49 PM	File folder
 bootstrap-star-rating	6/10/2023 6:49 PM	File folder
 bootstrap-timepicker	6/10/2023 6:49 PM	File folder

Και χρησιμοποιήθηκαν σε όλα τα HTML αρχεία που έχουν αναφερθεί προηγορμένως.

Ενότητα 4^η: Εγχειρίδιο Χρήσης Εφαρμογής

Ανάλυση Βάσης Δεδομένων

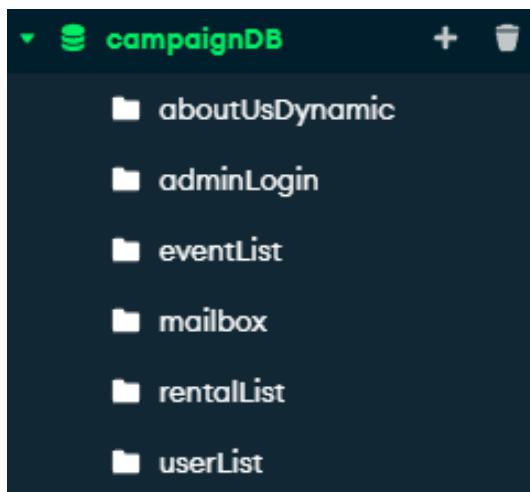
Η βάση δεδομένων που χρησιμοποιήθηκε είναι η MongoDB και το όνομα της είναι “campaignDB”:

```

test> show dbs
admin          40.00 KiB
campaignDB    368.00 KiB
config         72.00 KiB
eCommerceDB   40.00 KiB
local          72.00 KiB
test>

```

Tα collections (συλλογές) που χρησιμοποιήθηκαν είναι τα παρακάτω:



To collection `aboutUsDynamic` έχει document format:

```

1  /**
2  * Paste one or more documents here
3  */
4  {
5      "whoAreWe1": "Our company thrives at the intersection of ...
6      "whoAreWe2": "At Tech-Ro Solutions, we understand that th...
7      "aboutUsImages": {
8          "image1": "img/aboutus/image1.png",
9          "image2": "img/aboutus/image2.png",
10         "image3": "img/aboutus/image3.png",
11         "image4": "img/aboutus/image4.png"
12     }
13 }

```

To collection `adminLogin` έχει το παρακάτω format:

```

_id: ObjectId('6494c086b29db4a36648615c')
username: "admin"
password: "admin"

```

To collection `eventList` έχει το παρακάτω format:

```
To collection campaignDB.eventList
 5   "eventName": "Event 1",
 6   "eventDescription": "This is a description of Event 1",
 7   "eventImage": "img/events/event1.jpg",
 8   "eventCoordinates": {
 9     "lat": 37.9570912,
10     "long": 23.7127157
11   },
12   "eventDate": {
13     "$date": "2023-12-25T19:20:00.000Z"
14   },
15   "tickets": {
16     "A": {
17       "price": 100,
18       "totalSeats": 5,
19       "bookedSeats": []
20     },
21     "B": {
22       "price": 75,
23       "totalSeats": 10,
24       "bookedSeats": []
25     },
26     "C": {
27       "price": 50,
28       "totalSeats": 15,
29       "bookedSeats": []
30   }
31 }
32 }
```

To collection `mailbox`:

```
To collection campaignDB.mailbox
VIEW { } ⌂
```

```
1 /**
2  * Paste one or more documents here
3 */
4 {
5   "mailTitle": "Test Mail Title",
6   "mailContent": "Test Mail Content",
7   "mailDate": ""
8 }
```

To collection `rentalList` έχει το εξής format:

To collection campaignDB.rentalList

VIEW { } ⋮

fb ⋮

```
1  /**
2   * Paste one or more documents here
3   */
4  {
5    "eventId": {
6      "$oid": "6488689a6d62c5a722a20485"
7    },
8    "ticketType": "B",
9    "seats": [
10      "B-3"
11    ],
12    "totalPrice": "75",
13    "fullNames": [
14      "saassda"
15    ],
16    "email": "test@test.com",
17    "creditCard": {
18      "type": "Master Card",
19      "number": "1234 1223 1233 2132"
20    },
21    "userId": {
22      "$oid": "6489ff36c456fe0cad039b33"
23    }
24 }
```

Kai τέλος το collection `userList` έχει:

To collection campaignDB.userList

VIEW { } ⋮

fb ⋮

```
1  /**
2   * Paste one or more documents here
3   */
4  {
5    "username": "test12334214",
6    "password": "32443423",
7    "email": "",
8    "firstName": "",
9    "lastName": "",
10   "address": ""
11 }
```

Ανάλυση Media Queries

Για media queries χρησιμοποιήθηκαν κατά βάση τα media queries που παρέχει το Bootstrap Framework. Μερικά παραδείγματα αυτών που χρησιμοποιήθηκαν καθ' όλη την διαδικτυακή εφαρμογή αποτελούν τα παρακάτω:

```
1528      @media (min-width: 1200px) {  
1529          .col-xl {  
1530              flex: 1 0 0%;  
  
1315      @media (min-width: 992px) {  
1316          .col-lg {  
1317              flex: 1 0 0%;  
1318      }  
  
658      @media (min-width: 1400px) {  
659          .container-xxl, .container-xl, .container-lg, .container-md, .container-sm, .container  
660          max-width: 1320px;  
661      }  
  
648      @media (min-width: 992px) {  
649          .container-lg, .container-md, .container-sm, .container {  
650              max-width: 960px;  
651          }  
652      }  
653      @media (min-width: 1200px) {  
654          .container-xl, .container-lg, .container-md, .container-sm, .container {  
655              max-width: 1140px;  
656          }  
  
3928      @media (prefers-reduced-motion: reduce) {  
3929          .nav-link {  
3930              transition: none;  
3931          }  
  
4607      @media (min-width: 576px) {  
4608          .card-group {  
4609              display: flex;  
4610              flex-flow: row wrap;  
4611          }  
  
5478      @media (prefers-reduced-motion: reduce) {  
5479          .modal.fade .modal-dialog {  
5480              transition: none;  
5481          }
```

Οδηγίες Εγκατάστασης και Εκτέλεσης

Τοποθετούμε τα αρχεία της βάσης δεδομένων που βρίσκονται στο φάκελο `my_db` στο φάκελο bin της MongoDB (βλέπε διάλεξη 8 διαφάνεια 91).

Έπειτα κατευθυνόμαστε στο φάκελο `my_files` εκεί ανοίγουμε ένα CMD (και κάνουμε το ανάλογο cd) και εκτελούμε την εντολή `npm install` για να επιβεβαιώσουμε ότι όλα τα κατάλληλα packages είναι κατεβασμένα στο φάκελο `node_modules`.

```
Zack@DESKTOP-DGLA0FE MINGW64 ~/Desktop/Ergasia/my_files
$ npm install

up to date, audited 83 packages in 790ms

3 packages are looking for funding
  run `npm fund` for details

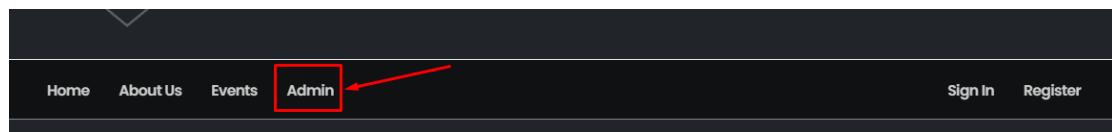
Found 0 vulnerabilities
```

Έπειτα εκτελούμε με χρήση της εντολής `node server.js` το αρχείο server.js για να έναρξη του localhost.

```
Zack@DESKTOP-DGLA0FE MINGW64 ~/Desktop/Ergasia/my_files
$ node server.js
Server is running on http://localhost:3000
```

Έπειτα από αυτό κατευθυνόμαστε χρησιμοποιώντας ένα browser της επιλογής μας, στη διεύθυνση: <http://127.0.0.1:3000/index-one-page.html>

Εκεί βρισκόμαστε στην αρχική σελίδα της εφαρμογής που αναπαριστά μια μικρή παρουσίαση της καμπάνιας. Στη πρώτη φωτογραφία του carousel πατάμε στο κουμπί “START NOW!” που μας ανακατευθύνει στη σελίδα των εκδηλώσεων (page-event-list.html) όπου μπορούμε να εκτελέσουμε τις λειτουργίες του χρήστη όπως ζητούνται από την εκφώνηση της εργασίας.



Έπειτα στη νέα σελίδα (και από κάθε σελίδα), μπορούμε να ανακατευθυνθούμε στη σελίδα της διαχειριστικής όπου ο διαχειριστής καλείται να χρησιμοποιήσει το συνδυασμό Username, Password για την πρόσβαση του στο interface του διαχειριστή.

Τα αναγνωριστικά είναι:

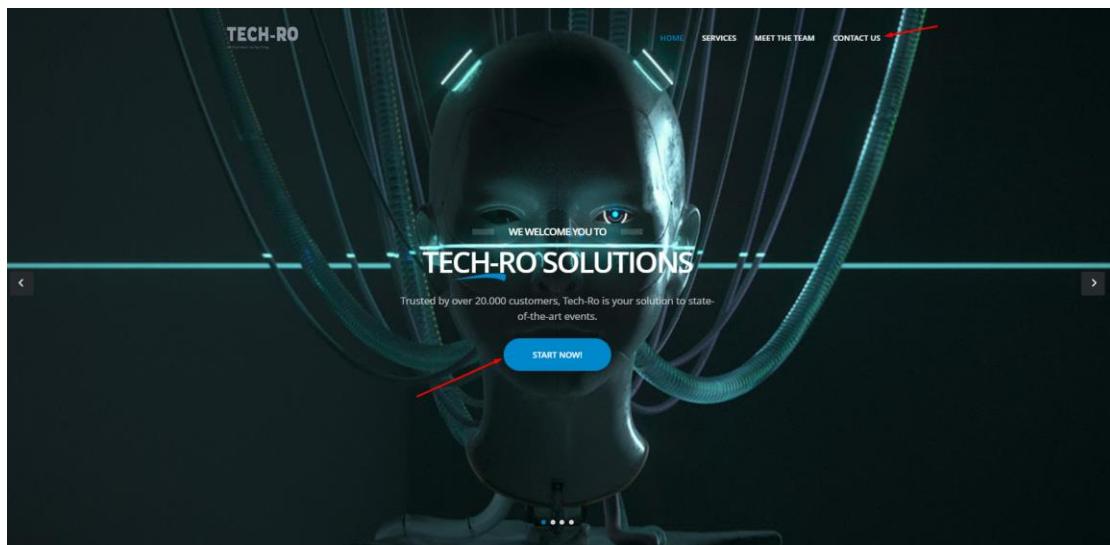
Username: admin

Password: admin

Ανάλυση Λειτουργιών και Αλληλεπίδραση με Βάση Δεδομένων
Στα screenshots που ακολουθούν τα κόκκινα παραλληλόγραμμα δείχνουν την αλληλεπίδραση με την βάση δεδομένων και το συγκεκριμένο collection κάθε φορά



1) Αρχική Σελίδα (index-one-page.html)

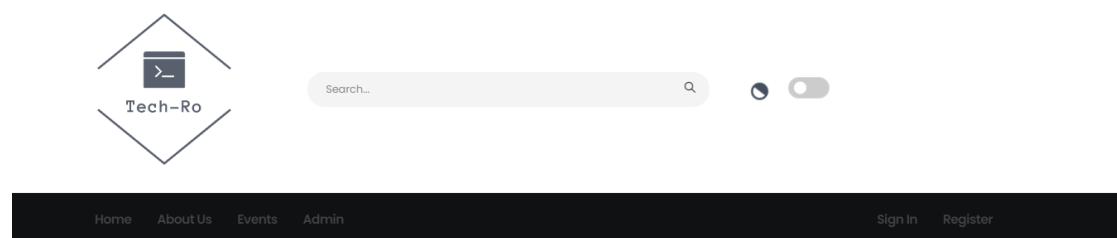


Στην αρχική σελίδα η μόνη λειτουργικότητα που παρατηρείται είναι αυτή της λειτουργίας “Contact Us” κατά την οποία ο χρήστης στέλνει μήνυμα στους διαχειριστές (μέσω της βάσης δεδομένων).

Πέραν αυτού στο carousel μπορεί να βρεθεί και το κουμπί “START NOW” που επιτρέπει στον χρήστη να προχωρήσει στην επόμενη σελίδα με τις εκδηλώσεις.

Η λειτουργικότητα του Contact Us, θα αναλυθεί αργότερα (βλέπε λειτουργικότητα 17)
καθώς δεν ζητείται στη συγκεκριμένη σελίδα.

- 2) Σύνδεση χρήστη (έχει υλοποιηθεί στην ίδια σελίδα, θα αναπτυχθεί στο 3)
- 3) Σύνδεση και Εγγραφή Χρήστη (sign-in-up-page.html)



Login

Username *

Password *

LOGIN

Register

Username *

Password *

Your personal data will be used to support your experience throughout this website, to manage access to your account, and for other purposes described in our [privacy policy](#).

REGISTER

Login

Username *

 ✓

Password *

 ✓

LOGIN

Success! You have successfully logged in. Redirecting...

Login

Username *

 ✓

Password *

 ✓

LOGIN

⚠ Oh snap! Change a few things up and try submitting again.

- Check your username.
- Check your password.

Ανάλογα μηνύματα για επιτυχία ή αποτυχία εμφανίζονται κατά την σύνδεση.

Register

Username *

Password *

Your personal data will be used to support your experience throughout this website, to manage access to your account, and for other purposes described in our [privacy policy](#).

Register

Username *

Password *

Your personal data will be used to support your experience throughout this website, to manage access to your account, and for other purposes described in our [privacy policy](#).

 **Success!** You have successfully registered.

Redirecting...

Ανάλογα μηνύματα για επιτυχία ή αποτυχία και κατά την εγγραφή.

Στο κομμάτι του κώδικα:

Sign-in-up.js (server-side κομμάτι)

/register route

```

const { MongoClient } = require('mongodb');
const url :string = 'mongodb://127.0.0.1:27017/';
const client :MongoClient = new MongoClient(url, { family: 4 });
const dbName :string = 'campaignDB';

// Define the route for user registration
router.post( path: '/register' , handlers: (req :Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res :Response<ResBody, LocalsObj> ) :void => {
    const { username, password } = req.body;
    // console.log('POST /register route handler called');
    const newUser :{ ... } = {
        username,
        password,
        email: '',
        firstName: '',
        lastName: '',
        address: ''
    };
    client.connect() Promise<MongoClient>
        .then(() :Promise<...> => {
            const db :Db = client.db(dbName);
            const collection :Collection<Document> = db.collection( name: 'UserList' );
            // Check if the username already exists
            return collection.findOne( filter: { username: username } )
                .then(existingUser :WithId<...> | null => {
                    // console.log("inside find");
                    if (existingUser) {
                        // If the user already exists, send an error response
                        res.status( code: 409 ).json( body: { error: 'Username already exists' } );
                        // Close the connection and stop further execution
                        // console.log("inside if");
                        throw new Error('Username already exists');
                    } else {
                        // console.log("inside else");
                        // If the user does not exist, insert the new user
                        return collection.insertOne(newUser);
                    }
                })
        })
})

```

/login route

```

//Define Route for User Login
router.post( path: '/login' , handlers: (req :Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res :Response<ResBody, LocalsObj> ) :void => {
    const { username, password } = req.body;
    // console.log('POST /login route handler called');

    client.connect() Promise<MongoClient>
        .then(() :Promise<...> => {
            const db :Db = client.db(dbName);
            const collection :Collection<Document> = db.collection( name: 'UserList' );
            // Try to find the user with the provided username
            return collection.findOne( filter: { username: username } );
        })
        .then(user :Document & {} => {
            // If the user doesn't exist or the password is incorrect, send an error response
            if (!user || user.password !== password) {
                res.status( code: 401 ).json( body: { error: 'Invalid username or password' } );
                // Stop further execution in this callback chain
                throw new Error('Invalid username or password');
            }

            // The user exists and the password is correct. Start a new session.
            req.session.user = { username: user.username };
            res.status( code: 200 ).json( body: { message: 'User logged in successfully' } );
        })
        .catch(err => {
            if (err.message !== 'Invalid username or password') {
                console.error('Error:', err);
                res.status( code: 500 ).json( body: { error: 'An error occurred.' } );
            }
        })
        .finally( onFinally: () :void => {
            client.close();
        });
})

```

Tα /check-session και /logout endpoints χρησιμοποιούνται σε όλες τις σελίδες του user interface (όπως ζητείται) ώστε να υπάρχει ανάλογο content ανάλογα την κατάσταση του session του user καθώς και μην μπορεί να επισκέπτεται κάποιες σελίδες όταν δεν είναι συνδεδεμένος.

Αναφέρονται εδώ και σε κάθε άλλη λειτουργικότητα (1-17) θα αναφερόμαστε σε αυτά χωρίς επεξήγηση.

```

router.get( path: '/check-session', handlers: (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj>, res : Response<ResBody, LocalsObj>) : void => {
    if (req.session.user) {
        res.send( body: { loggedIn: true, username: req.session.user.username });
    } else {
        res.send( body: { loggedIn: false });
    }
});

router.post( path: '/logout', handlers: (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj>, res : Response<ResBody, LocalsObj>) : void => {
    req.session.destroy();
    res.status( code: 200 ).send( body: { message: 'User logged out successfully' });
});

```

Από πλευράς client, χρησιμοποιήθηκε AngularJS για τον έλεγχο των φορμών και ανάδειξη κατάλληλων alert/notification σε περίπτωση error/success.

Ο κώδικας περιληπτικά:

```

<script>
angular.module('myApp', [])
//REGISTER
.controller('RegistrationController', function ($scope, $http, $timeout) {
    $scope.registerUser = function () {
        const username = document.querySelector('.username').value;
        const password = document.querySelector('.password').value;

        console.log('registerUser function called');

        // Make a POST request to the correct route (/register)
        $http.post('/register', { username: username, password: password })
            .then(function (response) {
                console.log(response);
                // Display the success message
                showNotification();

                // Redirect after 3 seconds
                $timeout(function () {
                    window.location.href = '/page-event-list.html';
                }, 3000);
            })
            .catch(function (error) {
                if (error.status === 409) {
                    alert('This username already exists. Please choose a different one.');
                    window.location.reload();
                } else {
                    // Display the error message
                    alert('An error occurred during registration.');
                    console.error('Error making POST request:', error);
                }
            });
    };
});

```

Διαχείριση register φορμών.

```

31 //LOGIN CONTROLLER -- SHOW SUCCESS OR FAIL NOTIFICATION
32 .controller('loginController', function ($scope, $http, $timeout) {
33   $scope.loginUser = function () {
34     const username = document.querySelector('.loginUsername').value;
35     const password = document.querySelector('.loginPassword').value;
36
37     console.log('loginUser function called');
38
39     // Make a POST request to the /Login route
40     $http.post('/login', { username: username, password: password })
41       .then(function (response) {
42         console.log(response);
43
44         showsuccessNotification();
45
46         // Display the success message
47         //alert('You have successfully logged in. You are being redirected...');
48
49         // Redirect after 3 seconds
50         $timeout(function () {
51           window.location.href = '/page-event-list.html';
52         }, 3000);
53       })
54       .catch(function (error) {
55         // Display the error message
56         showfailNotification();
57         //alert('An error occurred during login.');
58         console.error('Error making POST request:', error);
59       });
60   } //SUCCESS TRIGGER
61   function showsuccessNotification() {
62     const notification = document.getElementById('succNotif');
63     notification.style.display = 'block';
64     const failnotification = document.getElementById('failNotif');
65     failnotification.style.display = 'none';
66   }
67 } //FAIL TRIGGER

```

Διαχείριση login φόρμας και εμφάνιση ανάλογων μηνυμάτων.

Ta /login /register controllers καλούνται κάθε φορά ανάλογα τη φορμα.

Επίσης στη σελίδα `sign-in-up-page.html` έχουμε έλεγχο session κατά την εκκίνηση της σελίδας, όταν ο χρήστης έχει active session (είναι συνδεδεμένος/εγγραμένος) τότε γίνεται redirect:

```

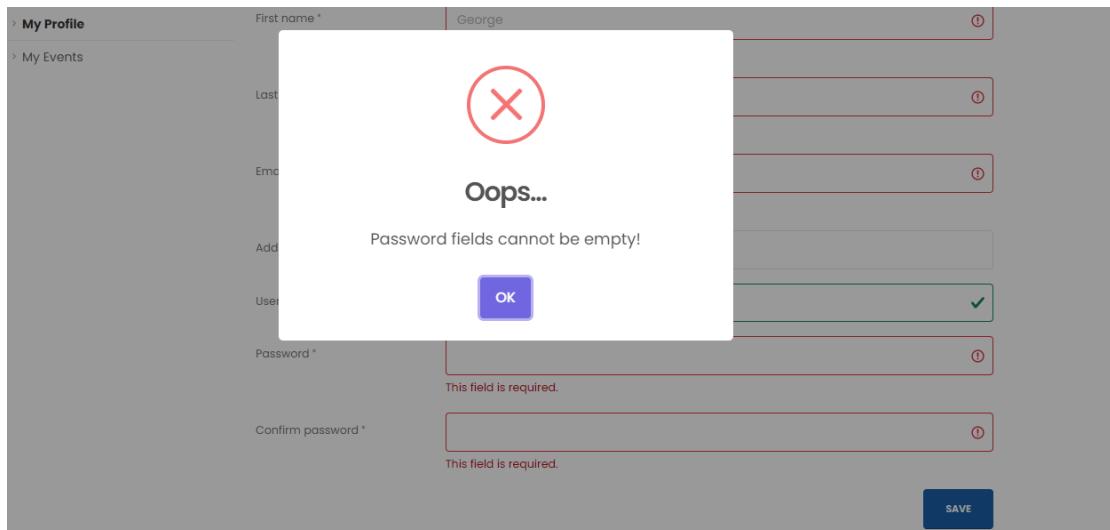
<!-- Session Check and Content Output -->
<script>
$(document).ready(function() {
  $.get('/check-session', function(data) {
    if (data.loggedIn) {
      window.location.href = "/page-event-list.html";
    }
    if (data.loggedIn) {
      $("#accountDropDown").show(); Εμφάνιση ανάλογων content ανάλογα το loggedIn Status
      $("#loginBtn").hide();
      $("#registerBtn").hide();
      $("#welcomeUsr").html('<strong class="text-color-dark text-4">Welcome ${data.username}!</strong>');
    } else {
      $("#accountDropDown").hide();
      $("#loginBtn").show();
      $("#registerBtn").show();
    }
  });
  $('#logoutBtn').click(function() {
    $.post('/Logout', function() {
      location.reload();
    });
  });
});

```

Χρησιμοποιεί τα routes /check-session /logout

4) Προφίλ Χρήστη (page-user-profile.html)

The screenshot shows the 'User Profile' page. At the top, there is a navigation bar with links to Home, About Us, Events, and Admin. Below the navigation bar, the page title 'User Profile' is displayed above a breadcrumb trail 'HOME > MY PROFILE'. The main content area is titled 'My Profile' and contains fields for First name, Last name, Email, Address, Username, Password, and Confirm password. Each field has a placeholder and a required asterisk. A 'SAVE' button is located at the bottom right.



Username * ✓

Password * ✓

Confirm password * ✓

SAVE

NOTICE! Your passwords don't match. **Please try again!**.

Username * ✓

Password * ✓

Confirm password * ✓

SAVE

Done! You have successfully updated your Profile Details! **Reload!**.

Από πλευράς κώδικα, στο server-side (page-user-profile.js) έχουμε τα εξής endpoints:
[/getProfile](#)

```

1 const express : any = require('express');
2 const router = express.Router();
3 const { MongoClient } = require('mongodb');
4
5 const url : string = 'mongodb://127.0.0.1:27017/';
6 const client : MongoClient = new MongoClient(url, { family: 4 });
7 const dbName : string = 'campaignDB';
8
9 // Define the route for getting user profile
10 router.get( path: '/getProfile', handlers: (req: Request<P, ResBody, ReqBody, ReqQuery, LocalsObj>, res: Response<ResBody, LocalsObj>) : void => {
11   // Session User
12   const sessionUser = req.session.user;
13   //Session Check
14   if (!sessionUser || !sessionUser.username) {
15     res.status( code: 400 ).json( body: { error: 'No user is currently logged in.' } );
16     return;
17   }
18
19   client.connect() Promise<MongoClient>
20     .then(() => {
21       const db : Db = client.db(dbName);
22       const collection : Collection<Document> = db.collection( name: 'userList' );
23       return collection.findOne( filter: { username: sessionUser.username } );
24     })
25     .then(user : Document & {} => {
26       // Omit the password from the response
27       delete user.password;
28       res.json(user);
29     })
30     .catch(err => {
31       console.error('Error:', err);
32       res.status( code: 500 ).json( body: { error: 'An error occurred.' } );
33     })
34     .finally( onFinally: () : void => {
35       client.close();
36     });
37 });

```

/updateProfile

```

49 // Define the route for updating user profile
50 router.post( path: '/updateProfile', handlers: (req: Request<P, ResBody, ReqBody, ReqQuery, LocalsObj>, res: Response<ResBody, LocalsObj>) : void => {
51   // Session User
52   const sessionUser = req.session.user;
53
54   // Body data
55   const { firstName, lastName, email, address, username, password } = req.body;
56
57   // User data to update
58   const updatedUser : {} = {
59     firstName,
60     lastName,
61     email,
62     address,
63     username,
64     password
65   };
66
67   client.connect() Promise<MongoClient>
68     .then(() => {
69       const db : Db = client.db(dbName);
70       const collection : Collection<Document> = db.collection( name: 'userList' );
71       return collection.updateOne(
72         filter: { username: sessionUser.username },
73         update: { $set: updatedUser }
74       );
75     })
76     .then(() : void => {
77       res.status( code: 200 ).json( body: { message: 'User profile updated successfully' } );
78     })
79     .catch(err => {
80       console.error('Error:', err);
81       res.status( code: 500 ).json( body: { error: 'An error occurred.' } );
82     });
83   });

```

Στο client-side έχουμε:

Εμφάνιση των δεδομένων στα fields από τη database χρησιμοποιώντας το active session του /getProfile.

```

517 $(document).ready(function() {
518     // Fetch user data when the page loads
519     $.ajax({
520         url: '/getProfile',
521         method: 'GET',
522         success: function(data) {
523             // Fill the form with the user data
524             $('input[name="firstName"]').val(data.firstName);
525             $('input[name="lastName"]').val(data.lastName);
526             $('input[name="email"]').val(data.email);
527             $('input[name="address"]').val(data.address);
528             $('input[name="username"]').val(data.username);
529         },
530         error: function(err) {
531             console.error(err);
532         }
533     });

```

Event Listener στο form submit:

```

335     // Handle form submit event
336     $('form[name="updateUser"]').on('submit', function(e) {
337         // Prevent the form from submitting
338         e.preventDefault();
339
340         // Get the password and confirmation
341         var password = $('input[name="password"]').val();
342         var confirmPassword = $('input[name="confirmPassword"]').val();
343
344         // Check if the passwords are empty
345         if (password === '' || confirmPassword === '') {
346             Swal.fire({
347                 icon: 'error',
348                 title: 'Oops...',
349                 text: 'Password fields cannot be empty!',
350             });
351             return false;
352         }
353
354         // Check if they match
355         if (password !== confirmPassword) {
356             updateProfileFail();
357             //alert("Passwords do not match");
358             return false;
359         }
360
361         // If they match, send the form data to the server
362         var formData = $(this).serialize();
363         $.ajax({
364             url: '/updateProfile',
365             method: 'POST',
366             data: formData,
367             success: function(data) {
368                 // Handle success response
369                 updateProfileSuccess();

```

Πάλι έχουμε έλεγχο για active session, στην περίπτωση που ο χρήστης δεν είναι συνδεδεμένος, γίνεται redirect στη σελίδα σύνδεσης/εγγραφής

```

521 <!-- Session Check and Content Output -->
522 <script>
523   $(document).ready(function() {
524     $.get('/check-session', function(data) {
525       if (!data.loggedIn) {
526         window.location.href = "/sign-in-up-page.html";
527       }
528       if (data.loggedIn) {
529         $('#accountDropDown').show();
530         $('#loginBtn').hide();
531         $('#registerBtn').hide();
532         $('#welcomeUser').html( <strong class="text-color-dark text-4">Welcome ${data.username}!</strong> );
533     } else {
534       $('#accountDropDown').hide();
535       $('#loginBtn').show();
536       $('#registerBtn').show();
537     }
538   });
539
540   $('#logoutBtn').click(function() {
541     $.post('/logout', function() {
542       location.reload();
543     });
544   });
545 });
546 </script>

```

Όλη η αλληλεπίδραση γίνεται με το collection `userList` στη database:

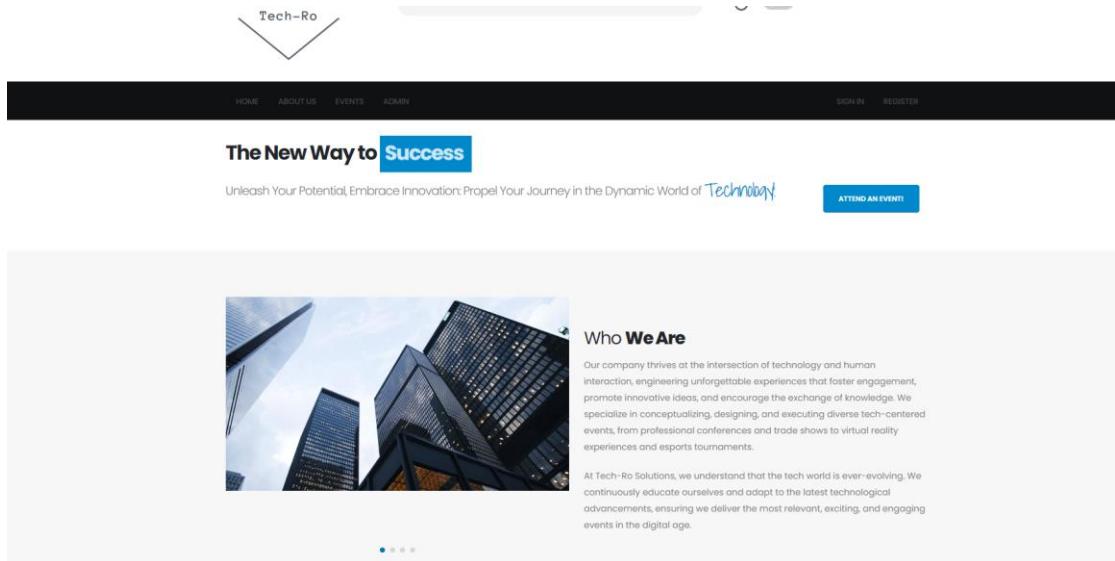
The screenshot shows the MongoDB Compass interface with the database name "campaignDB" and the collection name "userList". The "Documents" tab is selected. There is one document listed:

```

_id: ObjectId('648b02876a202bffa959e491')
username: "admin"
password: "admin"
email: ""
firstName: ""
lastName: ""
address: ""

```

5) Σελίδα Πληροφοριών καμπάνιας (aboutus.html)



Στο server-side (aboutus.js):

/getAboutUs route:

```

1 const express :any = require('express');
2 const router :Router = express.Router();
3 const { MongoClient } = require('mongodb');

4 const url :string = 'mongodb://127.0.0.1:27017/';
5 const client :MongoClient = new MongoClient(url, { family: 4 });
6 const dbName :string = 'campaignDB';

7 router.get( path: '/getAboutUs', handlers: (req :Request<P, ResBody, ReqBody, ReqQuery, LocalsObj>, res :Response<ResBody, LocalsObj>) :void => {
8   client.connect() :Promise<MongoClient>
9     .then(() :Promise<...> => {
10       const db :Db = client.db(dbName);
11       const collection :Collection<Document> = db.collection( name: 'aboutUsDynamic' );
12       return collection.findOne( filter: {} );
13     }) :Promise<...> |
14       .then(result :Document & ... => {
15         res.status( code: 200 ).json(result);
16         console.log(result);
17       }) :Promise<Promise<...>>
18       .catch(err => {
19         console.error('An error occurred:', err);
20         res.status( code: 500 ).send(err);
21       })
22       .finally( onFinally: () :void => {
23         client.close();
24       });
25     });
26   });

27 module.exports = function (app) :void {
28   app.use('/', router);
29 };

```

Στο client-side κομμάτι:

```

273 <script>
274     $.ajax({
275         url: "/getAboutUs", // Changed route here
276         method: "GET",
277         success: function(data) {
278             // Setting the text content for the "who we are" paragraphs
279             $('#whoAreWe1').text(data.whoAreWe1);
280             $('#whoAreWe2').text(data.whoAreWe2);
281
282             // Setting the images
283             $('#image1').attr('src', data.aboutUsImages.image1);
284             $('#image2').attr('src', data.aboutUsImages.image2);
285             $('#image3').attr('src', data.aboutUsImages.image3);
286             $('#image4').attr('src', data.aboutUsImages.image4);
287         }
288     });
289 </script>

```

Δυναμική εμφάνιση των πληροφοριών από το collection `aboutUsDynamic`:

campaignDB.aboutUsDynamic

- Documents Aggregations Schema Explain Plan Indexes Validation

Filter Type a query: { field: 'value' }

ADD DATA **EXPORT DATA**

```

_id: ObjectId('6495d0c1721a55021705b02c')
whoAreWe1: "Our company thrives at the intersection of technology and human intera..."
whoAreWe2: "At Tech-Ro Solutions, we understand that the tech world is ever-evolvi..."
aboutUsImages: Object
  image1: "img/aboutus/image1.png"
  image2: "img/aboutus/image2.png"
  image3: "img/aboutus/image3.png"
  image4: "img/aboutus/image4.png"

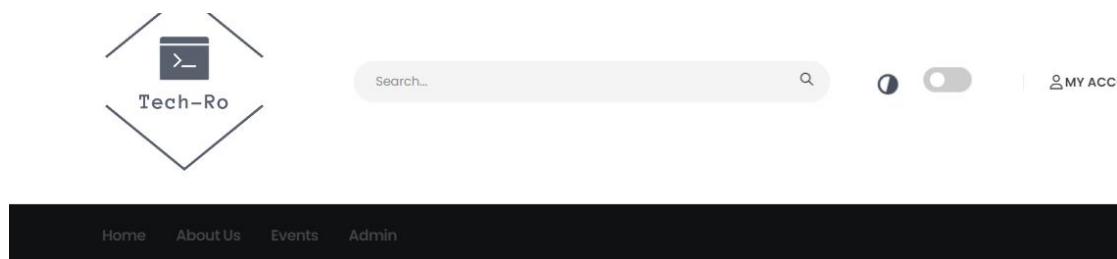
```

6) Σελίδα Εκδηλώσεων*

Η σελίδα εκδηλώσεων μας σπάει σε 2 επιμέρους σελίδες.

Σελίδα με τη λίστα των εκδηλώσεων και σελίδα με τις πληροφορίες της σελίδας που επιλέχθηκε.

Η πρώτη σελίδα είναι αυτή που τις εμφανίζει ως upcoming (μελλοντικές) ή expired (ολοκληρωμένες) και ονομάζεται `page-event-list.html`.



All Upcoming Events:



2023-12-25 21:20:00



2023-12-25 20:00:00



2023-12-25 20:00:00

Event 1

TICKET A: 5 TICKET B: 10
TICKET C: 15

Event 2

TICKET A: 0 TICKET B: 9
TICKET C: 9

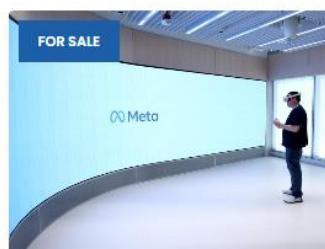
Event 2

TICKET A: 5 TICKET B: 10
TICKET C: 12

All Expired Events:



2022-12-25 20:00:00



2022-12-25 20:00:00

Event 2

TICKET A: 5 TICKET B: 8
TICKET C: 10

Event 2

TICKET A: 5 TICKET B: 10
TICKET C: 15

Λειτουργία hover με εμφάνιση map της τοποθεσίας της εκδήλωσης:

All Upcoming Events:

FOR SALE map

Athenaeum Ath

Sokratous

Ag. Panton

CHAROKOPOU ΧΑΡΟΚΟΠΟΥ Google

Map data ©2023 Terms of Use Report a map error

2023-12-25 21:20:00

Event 1

TICKET A: 5 TICKET B: 10

TICKET C: 15

2023-12-25 20:00:00

Event 2

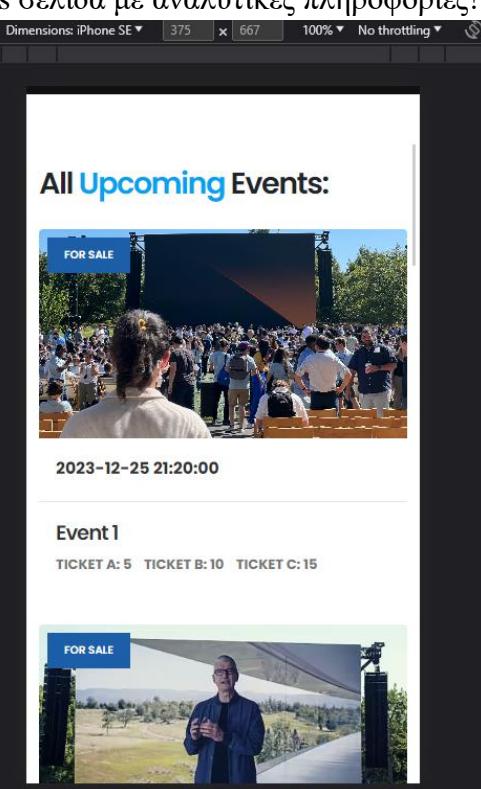
TICKET A: 0 TICKET B: 9

TICKET C: 9

Hover

Κατά το κλικ, ανοίγει google maps σελίδα με αναλυτικές πληροφορίες!

Όλα Responsive όπως ζητείται:



Από πλευράς κώδικα, στο server-side (page-event-list.js) έχουμε τα ακόλουθα endpoints:

/getEvents που παίρνει τα μη expired events

```
10 // Event Fetch
11 router.get( path: '/getEvents', handlers: (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj>, res : Response<ResBody, LocalsObj>) : void => {
12     client.connect() Promise<MongoClient>
13         .then(() => {
14             const db : Db = client.db(dbName);
15             const collection : Collection<Document> = db.collection( name: 'eventList' );
16             const now : Date = new Date();
17             return collection
18                 .find( filter: {
19                     eventDate: { $gt: now },
20                 } ) // Filter out events with dates greater than current date
21                 .toArray();
22         })
23         .then(events : Promise<...> => {
24             //console.log(events);
25             res.json(events);
26         })
27         .catch(err => {
28             console.error('Error:', err);
29             res.status( code: 500).json( body: { error: 'An error occurred.' });
30         })
31         .finally( onFinally: () : void => {
32             client.close();
33         });
34 }
```

/getExpiredEvents που παίρνει τα expired events

```
37 // Expired Event Fetch
38 router.get( path: '/getExpiredEvents', handlers: (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj>, res : Response<ResBody, LocalsObj>) : void => {
39     client.connect() Promise<MongoClient>
40         .then(() => {
41             const db : Db = client.db(dbName);
42             const collection : Collection<Document> = db.collection( name: 'eventList' );
43             const now : Date = new Date();
44             return collection
45                 .find( filter: {
46                     eventDate: { $lt: now },
47                 } ) // Filter out events with dates less than current date
48                 .toArray();
49         })
50         .then(expiredEvents : Promise<...> => {
51             //console.log(expiredEvents);
52             res.json(expiredEvents);
53         })
54         .catch(err => {
55             console.error('Error:', err);
56             res.status( code: 500).json( body: { error: 'An error occurred.' });
57         })
58         .finally( onFinally: () : void => {
59             client.close();
60         });
61 }
```

Στο Κομμάτι του client-side χρησιμοποιείται δυναμική εμφάνιση των expired events από την database με την χρήση του eventTemplate.html.

To Javascript:

```

371 <script>
372   $(document).ready(function() {
373     $.get("eventTemplate.html", function(template) {
374       //console.log("inside get template");
375       $.get("/getExpiredEvents", function(events) {
376         //console.log("inside getExpiredEvent");
377         for (let i = 0; i < events.length; i++) {
378           //console.log("inside for");
379
380           let date = new Date(events[i].eventDate);
381           let formattedDate = date.getFullYear() + '-' +
382             ('0' + (date.getMonth() + 1)).slice(-2) + '-' +
383             ('0' + date.getDate()).slice(-2) + ' ' +
384             ('0' + date.getHours()).slice(-2) + ':' +
385             ('0' + date.getMinutes()).slice(-2) + ':' +
386             ('0' + date.getSeconds()).slice(-2);
387
388           let eventHTML = $(template);
389           eventHTML.find("strong.text-4").text(formattedDate);
390           eventHTML.find("h4.text-5").text(events[i].eventName);
391           eventHTML.find("ul.list-unstyled li.list-inline-item:eq(0)").find("strong").text('Ticket A: ' + (events[i].tickets.A.totalSeats - events[i].tickets.A.bookedSeats.length));
392           eventHTML.find("ul.list-unstyled li.list-inline-item:eq(1)").find("strong").text('Ticket B: ' + (events[i].tickets.B.totalSeats - events[i].tickets.B.bookedSeats.length));
393           eventHTML.find("ul.list-unstyled li.list-inline-item:eq(2)").find("strong").text('Ticket C: ' + (events[i].tickets.C.totalSeats - events[i].tickets.C.bookedSeats.length));
394
395         }
396       }
397     }
398   })
399 
```

χρήση HTML template για ευκολία και πιο ευανάγνωστο κώδικα

Εργάσιμη της ημερομηνίας ευανάγνωστο format

Δυναμική εμφάνιση eventName και eventTickets σε κάθε loop της for

Χρησιμοποιείται το "eventTemplate.html" καθε επανάληψη

To eventTemplate.html είναι το παρακάτω:

```

1 <div class="col-12 col-sm-6 col-md-3 pb-4 mb-1 event-item" data-event-id="${events[i]._id}">
2   <div class="card custom-card-info custom-card-info-shadow border-0">
3     <div class="card-body overflow-hidden p-relative z-index-1">
4       <a href="javascript:void(0)" class="text-decoration-none">
5         <span class="custom-card-info-type bg-primary text-color-light px-3 py-1 text-1 font-weight-semibold text">
6           <span class="custom-card-info-img d-block">
7             <img src="" class="img-fluid">-->
8             <div class="map-image" style="..."></div>
9           </span>
10          <span class="custom-card-info-header d-block p-relative">
11            <strong class="text-dark text-4"></strong>
12            
13          </span>
14          <span class="custom-card-info-content d-block">
15            <h4 class="text-dark mb-1 text-5"></h4>
16            <ul class="list list-unstyled list-inline mb-0">
17              <li class="list-inline-item me-2 mb-0">
18                <strong class="text-default text-uppercase text-3"></strong>
19              </li>
20              <li class="list-inline-item me-2 mb-0">
21                <strong class="text-default text-uppercase text-3"></strong>
22              </li>
23              <li class="list-inline-item me-0 mb-0">
24                <strong class="text-default text-uppercase text-3"></strong>
25              </li>
26            </ul>
27          </span>
28        </a>
29      </div>
30    </div>
31  </div>
32 
```

Υλοποίηση mouse hover για εμφάνιση map με το location του expired event στη κάθε εκδήλωση, πάντα δυναμικά από το collection `eventList`:

```

395   let divElement = eventHTML.find('.map-image');
396   let originalSrc = "url(" + events[i].eventImage + ")";
397   divElement.attr('style', "background-image: " + originalSrc + "; width: 100%; height: 200px; background-size: cover;");
398
399   // We create an iframe for the map and initially hide it
400   let mapSrc = "https://maps.google.com/maps?q=" + events[i].eventCoordinates.lat + "," + events[i].eventCoordinates.long + "&hl=es;z=14&output=embed";
401   let mapIframe = '<iframe src="${mapSrc}" width="100%" height="200px" style="border:1px solid black; display: none;" allowfullscreen="" loading="lazy"></iframe>';
402   divElement.append(mapIframe);
403
404   let eventHTMLString = $(<div>).append(eventHTML.clone()).html();
405   //console.log(eventHTMLString);
406   $('#events-expired-container').append(eventHTMLString);
407 }
408
409 $('#events-expired-container').on('mouseover', '.map-image', function() {
410   // We show the iframe when the mouse hovers over the div
411   $(this).find('iframe').show();
412 });
413
414 $('#events-expired-container').on('mouseout', '.map-image', function() { // Note the change of container here
415   // We hide the iframe when the mouse moves out of the div
416   $(this).find('iframe').hide();
417 });

```

Με ίδιο ακριβώς τρόπο γίνεται και η εμφάνιση των upcoming events, με την μόνη διαφορά ότι τα upcoming events μπορεί ο χρήστης να κάνει click πάνω τους και παίρνοντας το ID του event προχωράει στην επόμενη σελίδα, την “page-event-list-details.html?id=” με ID το ID του event που πατήθηκε.

```
//on click event on anchor
eventHTML.find("a").attr("href", `page-event-list-details.html?id=${events[i]._id}`);
```

Η δεύτερη σελίδα είναι αυτή στην οποία ο χρήστης μπορεί να δει πληροφορίες για το event το οποίο πάτησε, οι οποίες εμφανίζονται δυναμικά ανάλογα το event.

The screenshot shows a web page for Event 2. At the top, there's a navigation bar with links for Home, About Us, Events, and Admin. Below the navigation is a large image of a man speaking on stage to an audience. To the right of the image is a form for ticket booking. The form fields include:

- Event:** Event 2
- Location:** (Map Location)
- Type A:** Total Type A Tickets: 5
- Type B:** Total Type B Tickets: 10
- Type C:** Total Type C Tickets: 15
- Number of Tickets:** A dropdown menu set to 1.
- Ticket type:** Radio buttons for Type A, B, or C, with Type A selected.

Description

This is a description of Event 2.

Number
of
Tickets:

3

Ticket type: A B C

Event 2

EVENTS > EVENT DETAILS

Select your Seats Below:



Μόνο η σειρά που σπέλεξε είναι ενεργοποιημένη

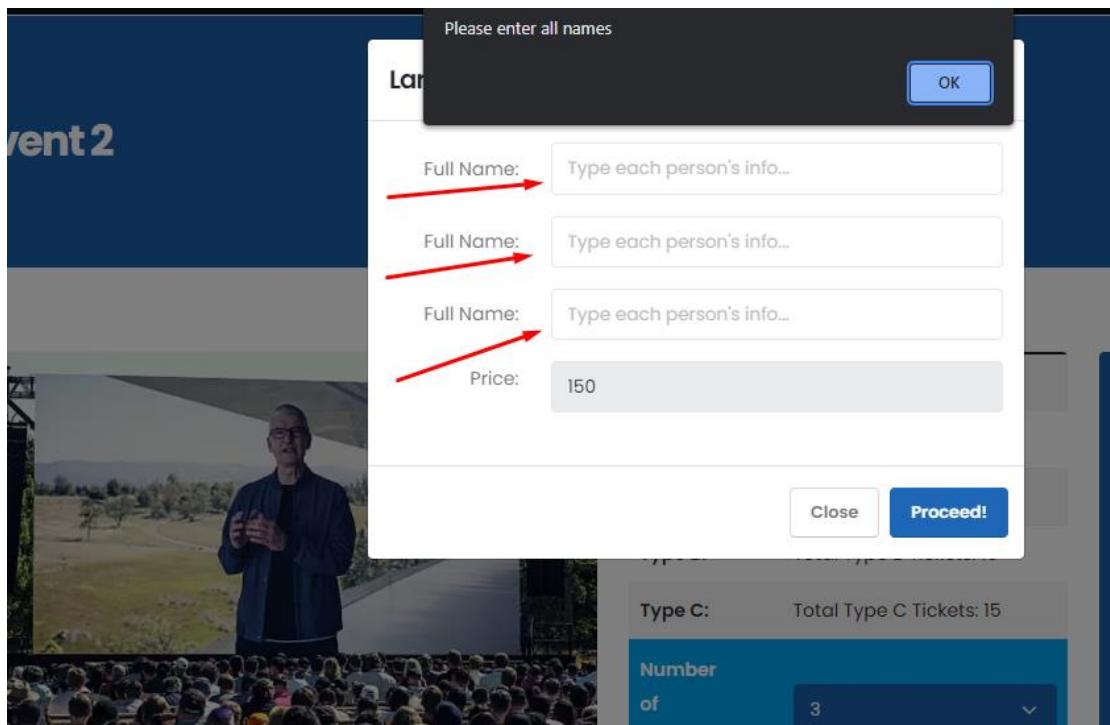
<input type="checkbox"/> A-1	<input type="checkbox"/> A-2	<input type="checkbox"/> A-3	<input type="checkbox"/> A-4	<input type="checkbox"/> A-5					
<input type="checkbox"/> B-1	<input type="checkbox"/> B-2	<input type="checkbox"/> B-3	<input type="checkbox"/> B-4	<input type="checkbox"/> B-5	<input type="checkbox"/> B-6	<input type="checkbox"/> B-7	<input type="checkbox"/> B-8	<input type="checkbox"/> B-9	<input type="checkbox"/> B-10
<input checked="" type="checkbox"/> C-1	<input checked="" type="checkbox"/> C-2	<input checked="" type="checkbox"/> C-3	<input checked="" type="checkbox"/> C-4	<input checked="" type="checkbox"/> C-5	<input checked="" type="checkbox"/> C-6	<input checked="" type="checkbox"/> C-7	<input type="checkbox"/> C-8	<input type="checkbox"/> C-9	<input checked="" type="checkbox"/> C-10

BOOK NOW!

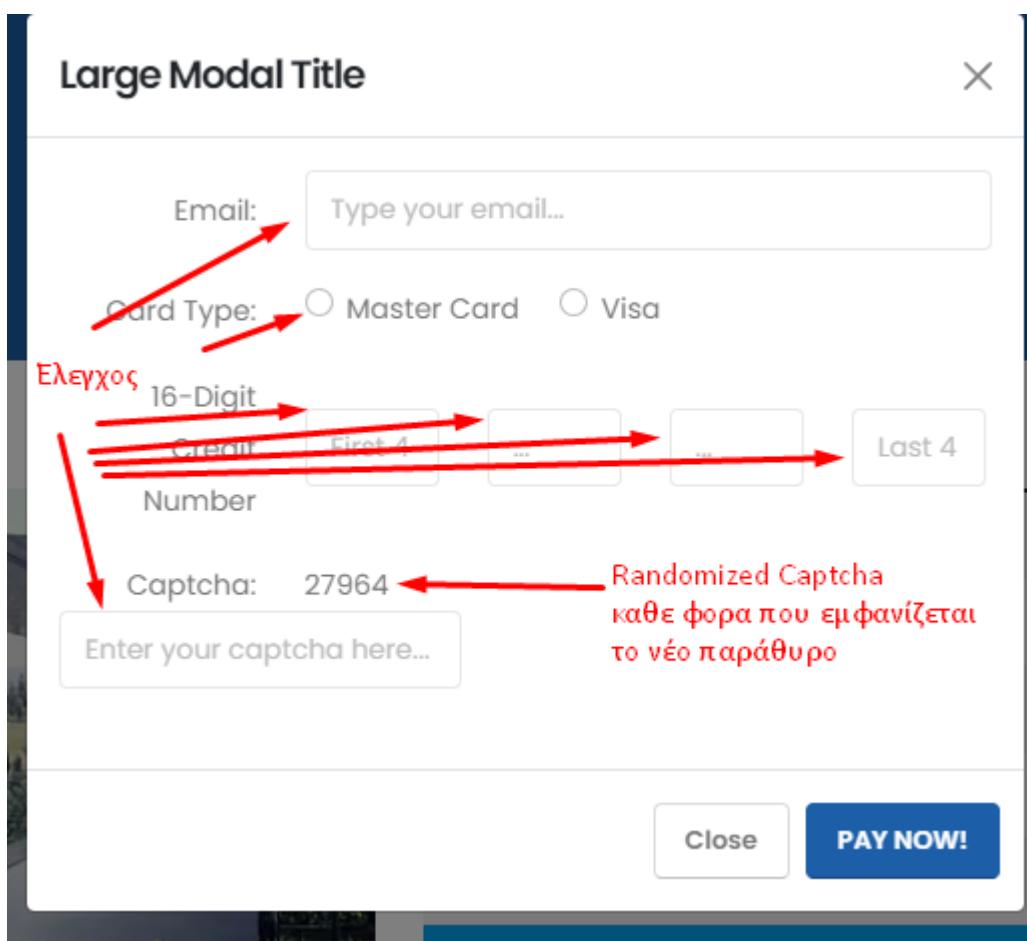
Ελέγχει τις διαθέσιμες θέσεις από τη βάση δεδομένων και ενεργοποιεί μόνο τα checkboxes που είναι διαθέσιμα εισιτήρια

Features

Πατώντας Book Now καλείται να βάλει τα ονόματα των θέσεων που θέλει να κλήσει με ελέγχους για συμπλήρωση των forms.



Με την συμπλήρωση των forms και Proceed εμφανίζεται το νέο modal:



Με την επιτυχημένη ολοκλήρωση των φορμών, και PAY NOW ολοκληρώνεται η κράτηση και αποστέλνεται mail με τις πληροφορίες:

Order Confirmation - Tech-Ro ➤ Inbox ×



techroconfirmorder@gmail.com

to me ▾

Hello admin,

Thank you for your purchase! Your order has been confirmed!

Order Details:

=====

Order ID: 64977ff19b0bc8edf337c3f3

Ticket 1:

Full Name: john doe

Seat: C-5

Ticket 2:

Full Name: jessica doe

Seat: C-6

Ticket 3:

Full Name: jess doe

Seat: C-7

Total Price: \$150

Credit Card: **** * 1234

=====

Thank you for choosing our services. Enjoy the event!

Best Regards,

Your Tech-Ro Foundation

Reply

Forward

Στο server-side (page-event-list-details.js) έχουμε τα εξής endpoints:

/getEventDetails route:

```
19 // GET /getEventDetails?id=eventID
20 router.get( path: '/getEventDetails', handlers: async (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res : Response<ResBody, LocalsObj> ) : Promise<void> => {
21   try {
22     await client.connect();
23     const db :Db = client.db(dbName);
24     const eventID = req.query.id;
25
26     const eventDetails :Document & {} = await db.collection( name: 'eventList').findOne( filter: { _id: new ObjectId(eventID) });
27     res.json(eventDetails);
28   } catch (error) {
29     console.error(error);
30     res.status( code: 500).json( body: { message: 'Error retrieving event details' });
31   }
32});
```

/createOrder route:

```
34 //Create Order Endpoint when user is paying
35 router.post( path: '/createOrder', handlers: async (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res : Response<ResBody, LocalsObj> ) : Promise<...> =>
36   const order :ReqBody = req.body;
37   try {
38     await client.connect();
39     const db :Db = client.db(dbName);
40
41     // Check if the user is logged in
42     if (!req.session || !req.session.user) {
43       return res.json( body: { success: false, error: 'Please Log in or Register' });
44     }
45
46     // Find the user's _id from the userList collection
47     const user :Document & {} = await db.collection( name: 'userList').findOne( filter: { username: req.session.user.username });
48     if (!user) {
49       return res.json( body: { success: false, error: 'User not found' });
50     }
51
52     // Add the userId to the order
53     order.userId = user._id;
54
55     // Convert the eventId to an ObjectId
56     order.eventId = new ObjectId(order.eventId);
57
58     // Log the order object
59     // console.log(order);
60
61     // Insert the order into the rentalList collection
62     const result :InsertOneResult<Document> = await db.collection( name: 'rentalList').insertOne(order);
```

Κατά την επιτυχημένη εισαγωγή document στο `rentalList` γίνεται αποστολή email όπως ζητείται

```

65     if (result.acknowledged) {
66         // Construct the message for the email
67         let message : string = `Hello ${user.username},\n\nThank you for your purchase! Your order has been confirmed!\n\n`;
68         message += 'Order Details:\n';
69         message += '=====\n';
70         message += `Order ID: ${result.insertedId}\n`;
71         message += `Ticket ${i+1}: \n\tFull Name: ${name}\n\tSeat: ${order.seats[i]}\n`.join('');
72         message += `Total Price: ${order.totalPrice}\n`;
73         message += `Credit Card: **** * ${order.creditCard.number.split(' ').pop()}\n`;
74         message += '=====\n';
75         message += 'Thank you for choosing our services. Enjoy the event!\n\n';
76         message += 'Best Regards,\nYour Tech-Ro Foundation';
77
78         // Set up mail options
79         const mailOptions : {} = {
80             from: 'techroconfirmorder@gmail.com', // sender address
81             to: order.email, // list of receivers
82             subject: 'Order Confirmation - Tech-Ro', // Subject line
83             text: message, // plain text body
84         };
85
86         transporter.sendMail(mailOptions, (err, info) => {
87             if(err)
88                 console.log(err)
89             else
90                 console.log(info);
91         });
92
93         return res.json( body: { success: true });

```

Κατά την εκτελημένη εισαγωγή document στο rentalList' αποστέλνεται email με χρήση "nodemailer" στη δηλωμένη διεύθυνση mail

/updateEventSeats route:

```

106    // Update booked seats for a particular event and ticket type
107    router.post( path: '/updateEventSeats', handlers: async (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj>, res : Response<ResBody, LocalsObj>) : Promise<...> => {
108        // console.log(req.body);
109        const { eventId, ticketType, seats } = req.body;
110
111        try {
112            await client.connect();
113            const db : Db = client.db(dbName);
114
115            // Convert eventId to ObjectId
116            const objectId = new ObjectId(eventId);
117
118            // Fetch the event from the database
119            const event : Document & {} = await db.collection( name: 'eventList').findOne( filter: { _id: objectId });
120
121            if (!event) {
122                return res.json( body: { success: false, error: 'Event not found' });
123            }
124
125            // Update the bookedSeats array in the event's ticket type
126            const updateResult : UpdateResult<Document> = await db.collection( name: 'eventList').updateOne(
127                filter: { _id: objectId },
128                update: { $push: { [ `tickets.${ticketType}.bookedSeats` ]: { $each: seats } } }
129            );
130
131            if (updateResult.acknowledged) {
132                return res.json( body: { success: true });
133            } else {
134                console.error(updateResult);
135                return res.json( body: { success: false, error: 'Failed to update event seats' });
136            }
137        } catch (error) {
138            console.error(error);
139            return res.json( body: { success: false, error: 'Failed to update event seats' });
140        }
141    });

```

Στο client-side, έχουμε 3 modals (τα 3 pop-up boxes) που εμφανίζονται ανάλογα την κατάσταση της παραγγελίας. Όπως ζητείται από την εκφώνηση όταν ο χρήστης επιλέγει το radio box για τον τύπο εισιτηρίου ανοίγει το πρώτο modal και ξεκινάει η διαδικασία της κράτησης.

Οπότε στη πλευρά του κώδικα έχουμε:

```

010 let selectedTicketType = '';
011 let selectedCheckboxes = [];  

012  

013 $(`input[name="inlineRadioOptions"]`).on('change', function() {  

014     // clear previously checked checkboxes of the selected type  

015     $('.seat-checkbox.ticket-type-${selectedTicketType}`).prop('checked', false);  

016     // clear the selectedCheckboxes array  

017     selectedCheckboxes = [];  

018  

019     selectedTicketType = $(this).val();  

020     const eventId = getEventIdFromUrl();  

021  

022     $.get(`/getEventDetails?id=${eventId}`, function(eventDetails) {  

023         // console.log(eventDetails);  

024         const imageUrl = getTicketTypeImageUrl(eventDetails, selectedTicketType);  

025         displayZoomedImage(imageUrl);  

026  

027         updateCheckboxStatus(eventDetails);  

028     });  

029 };  

030  

031 $('.form-select').on('change', function() {  

032     // clear previously checked checkboxes of the selected type  

033     $('.seat-checkbox.ticket-type-${selectedTicketType}`).prop('checked', false);  

034     // clear the selectedCheckboxes array  

035     selectedCheckboxes = [];  

036  

037     const eventId = getEventIdFromUrl();  

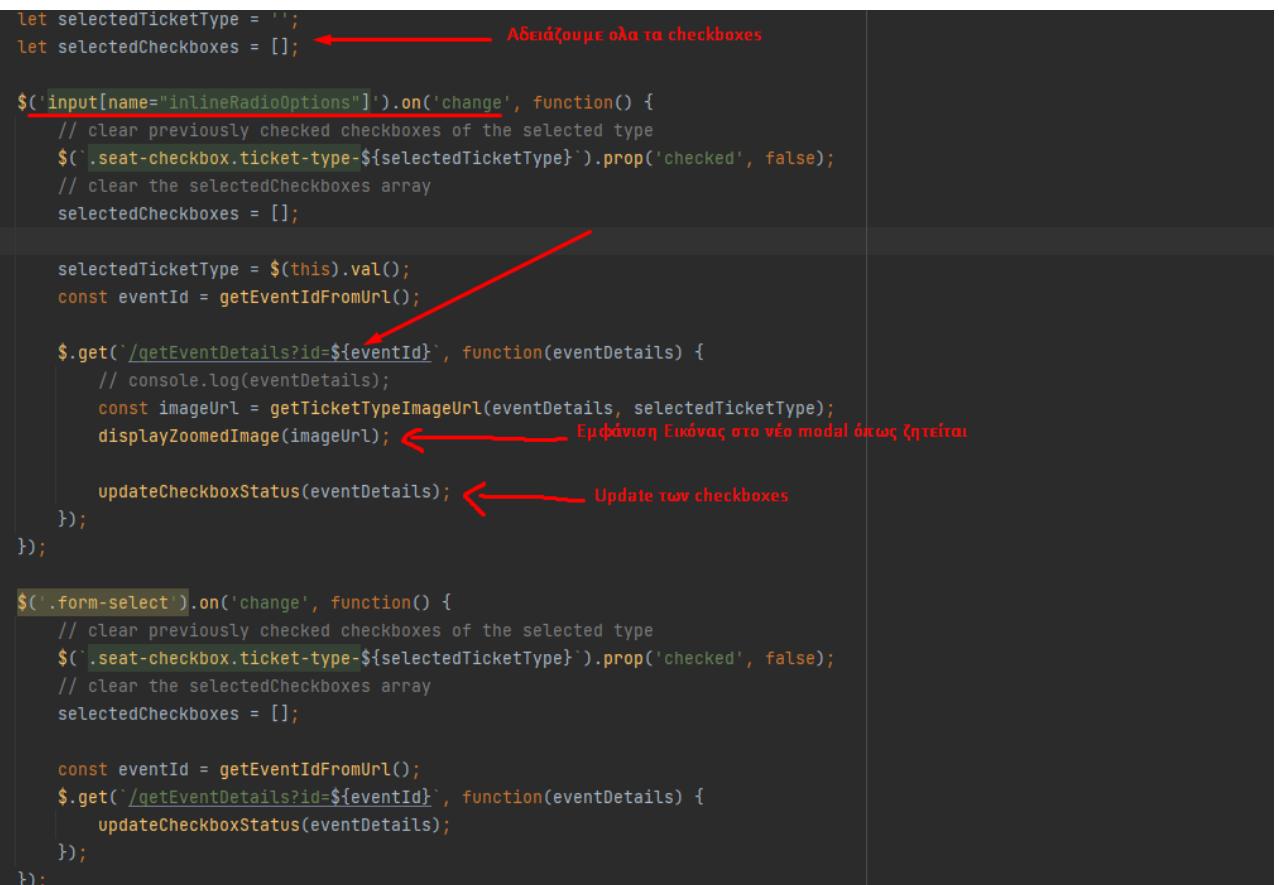
038     $.get(`/getEventDetails?id=${eventId}`, function(eventDetails) {  

039         updateCheckboxStatus(eventDetails);  

040     });  

041 });

```



```

//display ZOOMED image for first modal
1 usage
function displayZoomedImage(imageUrl) {
    $('#zoomedImage').attr('src', imageUrl);
    $('#zoomedImage').attr('width', '612');
    $('#zoomedImage').attr('height', '408');
    $('#zoomedInImageModal').modal('show');
}

//UPDATE STATUS for all checkboxes DEPENDING on the ticket type selected
3 usages
function updateCheckboxStatus(eventDetails) {
    const ticketCount = parseInt($('.form-select').val(), 10);
    const typeATotalSeats = eventDetails.tickets.A.totalSeats;
    const typeBTotalSeats = eventDetails.tickets.B.totalSeats;
    const typeCTotalSeats = eventDetails.tickets.C.totalSeats;

    // Disable all checkboxes
    $('input[type="checkbox"]').prop('disabled', true);

    if (selectedTicketType === 'A') {
        enableCheckboxesOfType('A', typeATotalSeats, ticketCount);
        disableBookedSeats(eventDetails.tickets.A.bookedSeats); //disable BOOKED seats found in DB
    } else if (selectedTicketType === 'B') {
        enableCheckboxesOfType('B', typeBTotalSeats, ticketCount);
        disableBookedSeats(eventDetails.tickets.B.bookedSeats); //disable BOOKED seats found in DB
    } else if (selectedTicketType === 'C') {
        enableCheckboxesOfType('C', typeCTotalSeats, ticketCount);
        disableBookedSeats(eventDetails.tickets.C.bookedSeats); //disable BOOKED seats found in DB
    }
}

```

//Generation Captcha Function 5-digit random number

```

1 usage
function generateCaptcha() {
    const captcha = Math.floor(10000 + Math.random() * 90000); // generates a 5 digit number
    $('#captchaDiv').html(captcha);
}

```

```

$.post('/createOrder', order, function(response) {
    if (response.success) {
        alert('Order created successfully!');
        // Updating the event collection with booked seats
        $.post('/updateEventSeats', { eventId, ticketType, seats }, function(updateResponse) {
            if (updateResponse.success) {
                // The event's bookedSeats have been updated successfully
                console.log("Event's bookedSeats updated successfully.");
            } else {
                // Failed to update the event's bookedSeats
                console.error('Failed to update event seats: ' + updateResponse.error);
            }
        });
    } else {
        if (response.error === 'Please Log in or Register') {
            alert(response.error);
            window.location.href = 'sign-in-up-page.html';
        } else {
            alert('Failed to create order: ' + response.error);
        }
    }
});

```

Όλες οι αλληλεπιδράσεις που έχουν μπει με **κόκκινο παραληλόγραμμο** στο server-sided κομμάτι και το αντίστοιχο request τους στο client κομμάτι γίνονται με τα collections `eventList` και `rentalList`.

Όπως και προηγούμενες σελίδες, έτσι και η σελίδα `page-event-list-details.html` ελέγχει την κατάσταση της σύνδεσης του χρήστη:

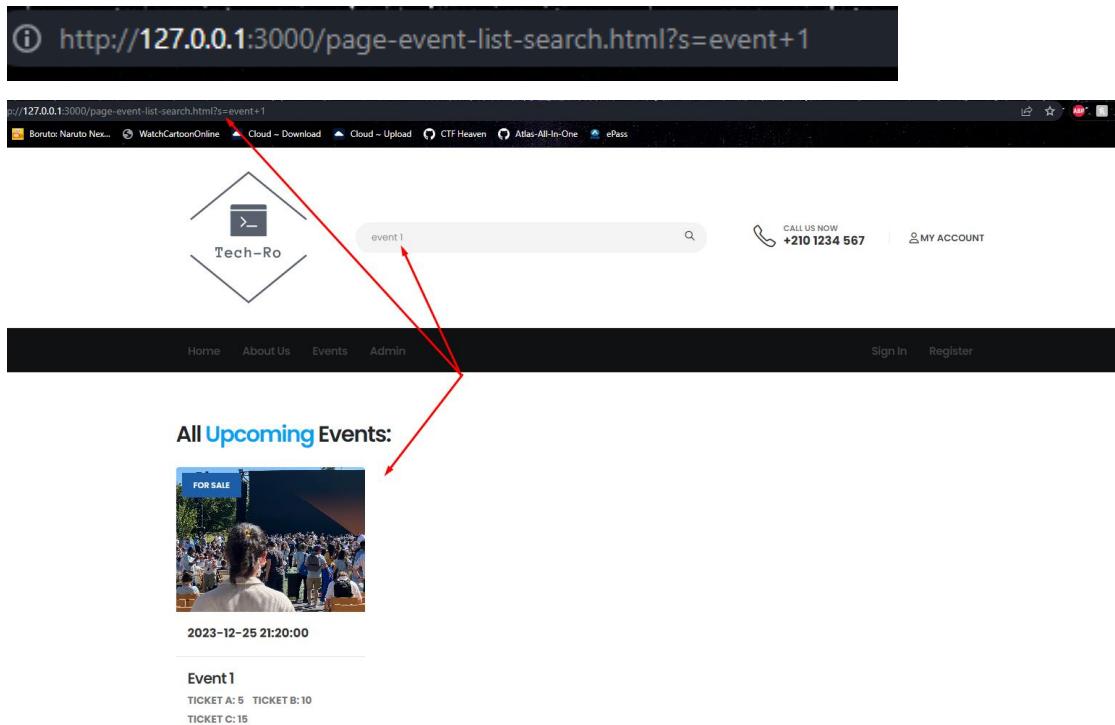
```
<!-- Session Check and Content Output -->
<script>
$(document).ready(function() {
    $.get('/check-session', function(data) {
        if (!data.loggedIn) {
            window.location.href = "/sign-in-up-page.html";
        }
        if (data.loggedIn) {
            $("#accountDropDown").show();
            $("#loginBtn").hide();
            $("#registerBtn").hide();
            $("#welcomeUsr").html('<strong class="text-color-dark text-4">Welcome ${data.username}!</strong>');
        } else {
            $("#accountDropDown").hide();
            $("#loginBtn").show();
            $("#registerBtn").show();
        }
    });
    $('#logoutBtn').click(function() {
        $.post('/logout', function() {
            location.reload();
        });
    });
});

```

8. Αναζήτηση εκδηλώσεων (page-event-list-search.html)

Ο χρήστης έχει τη δυνατότητα να αναζητά events από οποιαδήποτε σελίδα, πέραν την αρχικής, και τα αποτελέσματα εμφανίζονται πάντα στη σελίδα `page-event-list-search.html?s=` όπου “s” μπαίνει το search query.





Στο κομμάτι του κώδικα έχουμε:

Στο server-sided κομμάτι έχουμε τα routes τοποθετημένα στο αρχείο `page-event-list.js`:

/searchEvents route:

```

63 //Search Function
64 router.get( path: '/searchEvents', handlers: (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj>, res : Response<ResBody, LocalsObj>) : void =>
65   client.connect() Promise<MongoClient>
66   .then(() => {
67     const db :Db = client.db(dbName);
68     const collection :Collection<Document> = db.collection( name: 'eventList' );
69     const searchQuery = req.query.s;
70     const now :Date = new Date();
71     // We create a regex from the search query
72     // The 'i' flag makes the search case insensitive
73     const searchRegex :RegExp = new RegExp(searchQuery, flags: 'i');

74     return collection
75       .find( filter: {
76         eventDate: { $gt: now },
77         eventName: searchRegex,
78       })
79       .toArray();
80   }) Promise<Promise<....>>
81   .then(events => {
82     //console.log(events);
83     res.json(events);
84   })
85   .catch(err => {
86     console.error('Error:', err);
87     res.status( code: 500).json( body: { error: 'An error occurred.' });
88   })
89   .finally( onFinally: () :void => {
90     client.close();
91   });
92 });
93

```

/searchExpiredEvents

```
95 //Search Expired Events Function
96 router.get( path: '/searchExpiredEvents', handlers: (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj>, res : Response<ResBody, LocalsObj>) : void => {
97   client.connect() Promise<MongoClient>
98     .then(() : Promise<...> => {
99       const db:Dh = client.db(dbName);
100      const collection : Collection<Document> = db.collection( name: 'eventList' );
101      const searchQuery = req.query.s;
102      const now : Date = new Date();
103      // We create a regex from the search query
104      // The 'i' flag makes the search case insensitive
105      const searchRegex : RegExp = new RegExp(searchQuery, flags: 'i');
106
107      return collection
108        .find( filter: {
109          eventDate: { $lt: now },
110          eventName: searchRegex,
111        })
112        .toArray();
113    }) Promise<Promise<...>>
114    .then(events => {
115      //console.log(events);
116      res.json(events);
117    })
118    .catch(err => {
119      console.error('Error:', err);
120      res.status( code: 500 ).json( body: { error: 'An error occurred.' } );
121    })
122    .finally( onFinally: () : void => {
123      client.close();
124    });
125});
```

Από πλευράς client-side με παρόμοιο τρόπο όπως προηγουμένως στο `page-event-list.html` εμφανίζουμε δυναμικά τα αποτελέσματα της αναζήτησης από την βάση δεδομένων, ξεχωριστά για τα Upcoming και τα Expired Events.

Upcoming:

```
120 <script>
121 $(document).ready(function() {
122   const urlParams = new URLSearchParams(window.location.search);
123   const query = urlParams.get('s');
124
125   $.get("eventTemplate.html", function(template) {
126     $.get("/searchEvents?:" + query, function(events) {
127       console.log("inside getEvent");
128       for (let i = 0; i < events.length; i++) {
129         console.log("inside for");
130
131         let date = new Date(events[i].eventDate);
132         let formattedDate = date.getFullYear() + '-' +
133           ('0' + (date.getMonth()+1)).slice(-2) + '-' +
134           ('0' + date.getDate()).slice(-2) + ' ' +
135           ('0' + date.getHours()).slice(-2) + ':' +
136           ('0' + date.getMinutes()).slice(-2) + ':' +
137           ('0' + date.getSeconds()).slice(-2);
138
139         let eventHTML = $(template);
140         eventHTML.find("strong.text-4").text(formattedDate);
141         eventHTML.find("h4.text-5").text(events[i].eventName);
142         eventHTML.find("ul.list-unstyled li.list-inline-item:eq(0)").find("strong").text('Ticket A: ' + (events[i].tickets.A.totalSeats - events[i].tickets.A.bookedSeats.length));
143         eventHTML.find("ul.list-unstyled li.list-inline-item:eq(1)").find("strong").text('Ticket B: ' + (events[i].tickets.B.totalSeats - events[i].tickets.B.bookedSeats.length));
144         eventHTML.find("ul.list-unstyled li.list-inline-item:eq(2)").find("strong").text('Ticket C: ' + (events[i].tickets.C.totalSeats - events[i].tickets.C.bookedSeats.length));
145
146         //on click event on anchor
147         eventHTML.find("a").attr("href", "page-event-list-details.html?id=" + events[i].id);
148     });
149   });
150 });
```

Expired:

```

316 <script>
317   $(document).ready(function() {
318     const urlParams = new URLSearchParams(window.location.search) ←
319     const query = urlParams.get('s') ←
320
321     $.get("eventTemplate.html", function(template) {
322       $.get(`/_searchExpiredEvents?s=${query}`, function(events) {
323         // console.log("inside getExpiredEvent");
324         for (let i = 0; i < events.length; i++) {
325           // console.log("inside for");
326
327           let date = new Date(events[i].eventDate);
328           let formattedDate = date.getFullYear() + '-' +
329             ('0' + (date.getMonth() + 1)).slice(-2) + '-' +
330             ('0' + date.getDate()).slice(-2) + ' ' +
331             ('0' + date.getHours()).slice(-2) + ':' +
332             ('0' + date.getMinutes()).slice(-2) + ':' +
333             ('0' + date.getSeconds()).slice(-2);
334
335           let eventHTML = $(template);
336           eventHTML.find("strong.text-4").text(formattedDate);
337           eventHTML.find("h4.text-5").text(events[i].eventName);
338           eventHTML.find("ul.list-unstyled li.list-inline-item:eq(0)").find("strong").text('Ticket A: ' + (events[i].tickets.A.totalSeats - events[i].tickets.A.booked));
339           eventHTML.find("ul.list-unstyled li.list-inline-item:eq(1)").find("strong").text('Ticket B: ' + (events[i].tickets.B.totalSeats - events[i].tickets.B.booked));
340           eventHTML.find("ul.list-unstyled li.list-inline-item:eq(2)").find("strong").text('Ticket C: ' + (events[i].tickets.C.totalSeats - events[i].tickets.C.booked));
341
342           let divElement = eventHTML.find(".map-image");
343           let originalSrc = "url('" + events[i].eventImage + "')";
344           divElement.attr('style', "background-image: " + originalSrc + "; width: 100%; height: 200px; background-size: cover;");

```

9. Ιστορικό Εκδηλώσεων (page-user-orders.html)

> My Profile	Event Name:	Event 2
> My Events ←	Description:	This is a description of Event 2
	Event Date:	2023-12-25 20:00:00
	Order E-Mail:	zachkyria@gmail.com
	Full Name:	Lisa Doe
	Seat:	C-4

Event Name: Event 2

Description: This is a description of Event 2

Event Date: 2022-12-25 20:00:00

Order E-Mail: zachkyria@gmail.com

Full Name: Donkey Kong

Seat: B-4

Full Name: Melissa Ding

Seat: B-3

NOTICE! The event has ended. Stay tuned for the upcoming ones! .

Re-send Email >

Event Name: Event 2

Description: This is a description of Event 2

Event Date: 2023-12-25 20:00:00

Order E-Mail: zachkyria@gmail.com

Full Name: Lisa Doe

Seat: C-4

Full Name: Jessica Doe

Seat: C-3

Full Name: John Doe

Seat: C-2

Done! An e-mail has been sent with your ticket info! Please check your spam folder [HERE!](#) .

Re-send Email >

Και ο χρήστης λαμβάνει μέιλ στο δηλωμένο μέιλ κατά την παραγγελία:

Your Seat Rental Details for Event 2 Εισερχόμενα



techroconfirmorder@gmail.com

προς εγώ ▾

Dear Guest,

Thank you for your seat reservation for the event "Event 2".

Here are the details of your booking:

Event Name: Event 2

Description: This is a description of Event 2

Event Date: 12/25/2023, 8:00:00 PM

Seat(s) Details:

Full Name: John Doe, Seat: C-2

Full Name: Jessica Doe, Seat: C-3

Full Name: Lisa Doe, Seat: C-4

If you need to make any changes to your booking, please do not hesitate to get in touch with us.

Best regards,

Tech-Ro Team

 Απάντηση

 Προώθηση

Από πλευράς κώδικα, στο server-sided κομμάτι έχουμε τα εξής endpoints στο αρχείο `page-user-order.js`:

/userOrders route:

```
1// Define the route for getting user orders
18  router.get( path: '/userOrders', handlers: (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res : Response<ResBody, LocalsObj> ) : void => {
19    // Session User
20    const sessionUser = req.session.user;
21    console.log(sessionUser);
22
23    // Session Check
24    if (!sessionUser || !sessionUser.username) {
25        res.status( code: 400).json( body: { error: 'No user is currently logged in.' });
26        return;
27    }
28
29    let db; //use in multiple then() blocks
30
31    client.connect() Promise<MongoClient>
32        .then(() => {
33            db = client.db(dbName);
34            const userCollection :Collection<Document> = db.collection( name: 'userList');
35            // Use the username we got from the session to query the userList
36            return userCollection.findOne( filter: { username: sessionUser.username });
37        }) Promise<...>
38        .then(user :Document &{...} => {
39            const rentalCollection = db.collection('rentalList');
40            // Now use the _id we got from the previous query to query the rentalList
41            return rentalCollection.find({ userId: new ObjectId(user._id) }).toArray();
42        }) Promise<any> |
43        .then(rentals => {
44            const eventPromises = rentals.map(rental => {
45                const eventCollection = db.collection('eventList');
46                return eventCollection.findOne({ _id: new ObjectId(rental.eventId) });
47            });
48
49            return Promise.all(eventPromises).then(events :(Awaited<...>)[] => {
50                rentals.forEach((rental, index) :void => {
51                    rental.event = events[index];
52                });
53                return rentals;
54            });
55        }) Promise<...>
56        .then(rentals :(Awaited<...>)[] => {
57            // console.log(rentals);
58            rentals.forEach(rental :Awaited<unknown> => {
59                // Convert ObjectIds to strings
60                rental._id = rental._id.toString();
61                rental.userId = rental.userId.toString();
62                rental.eventId = rental.eventId.toString();
63                rental.event._id = rental.event._id.toString();
64            });
65            res.json(rentals);
66        })
67        .catch(err => {
68            console.error('Error:', err);
69            res.status( code: 500).json( body: { error: 'An error occurred.' });
70        })
71        .finally( onFinally: () :void => {
72            client.close();
73        });
74}
```

/rentalInfo route:

```
76 //Resend Email with Rental Info Endpoint
77 router.post( path: '/rentalInfo', handlers: (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj>, res : Response<ResBody, LocalsObj>) : any | undefined => {
78     // Validate request data
79     if (!req.body || !req.body.rental) {
80         return res.status( code: 400).json( body: { error: 'Rental data is missing in the request.' });
81     }
82
83     let rental = req.body.rental;
84     // Check if the rental is not expired
85     if (new Date(rental.event.eventDate) > new Date()) {
86         // If not expired, send the email with nodemailer
87         let seatsList = rental.seats.map((seat, index) : string => {
88             return `Full Name: ${rental.fullNames[index]}, Seat: ${seat}`;
89         }).join('\n');
90
91         let mailOptions : {} = {
92             from: 'techroconfirmorder@gmail.com',
93             to: rental.email,
94             subject: `Your Seat Rental Details for ${rental.event.eventName}`,
95             text: `Dear Guest,
96
97             Thank you for your seat reservation for the event "${rental.event.eventName}".
98
99             Here are the details of your booking:
100
101             Event Name: ${rental.event.eventName}
102             Description: ${rental.event.eventDescription}
103             Event Date: ${new Date(rental.event.eventDate).toLocaleString()}
104
105             Seat(s) Details:
106             ${seatsList}
```

```
114 transporter.sendMail(mailOptions, callback: (error, info) : any | undefined => {
115     if (error) {
116         console.error('Error:', error);
117         return res.status( code: 500).json( body: { error: 'Failed to send the email.' });
118     }
119     console.log('Email sent: ' + info.response);
120     res.json( body: { message: 'Email sent successfully.' });
121 };
122 } else {
123     res.json( body: { message: 'Rental is expired. No email sent.' });
124 }
```

Στο Client-Side έχουμε δυναμική εμφάνιση των orders του συνδεδεμένου χρήστη και δυναμική εμφάνιση των εισιτηρίων του (+ Άλλων πληροφοριών) σε φόρμες.

```

304 <script>
305     $(document).ready(function() {
306         $.get("/userOrders", function(rentals) {
307             // console.log("Received rentals: ", rentals);
308
309             rentals.forEach((rental, index) => {
310                 // Clone the template form and set a unique id
311                 let rentalForm = $("#rentalHistory").clone();
312                 rentalForm.attr("id", "rentalHistory" + index);
313
314                 // Populate the form with the rental and event data
315                 // console.log("Processing rental: ", rental);
316                 rentalForm.find("input[name='eventName']").val(rental.event.eventName);
317                 // console.log("Set eventName to: ", rental.event.eventName);
318
319                 rentalForm.find("input[name='eventDescription']").val(rental.event.eventDescription);
320                 // console.log("Set eventDescription to: ", rental.event.eventDescription);
321
322                 let date = new Date(rental.event.eventDate);
323                 let formattedDate = date.getFullYear() + '-' +
324                     ('0' + (date.getMonth() + 1)).slice(-2) + '-' +
325                     ('0' + date.getDate()).slice(-2) + ' ' +
326                     ('0' + date.getHours()).slice(-2) + ':' +
327                     ('0' + date.getMinutes()).slice(-2) + ':' +
328                     ('0' + date.getSeconds()).slice(-2);
329                 rentalForm.find("input[name='eventDate']").val(formattedDate);
330                 // console.log("Set eventDate to: ", formattedDate);
331
332                 rentalForm.find("input[name='email']").val(rental.email);
333                 // console.log("Set email to: ", rental.email);
334
335             // Add full name and seat for each seat in the rental
336             rental.seats.forEach((seat, index) => {
337                 // Create a new input for each seat wrapped in a form-group div
338                 let fullNameInput = $(<div class="form-group row"><label class="col-lg-3 col-form-label form-control-label line-height-9 pt-2 text-2">Full Name:</label><div class="col-lg-9 col-form-control form-control line-height-9 pt-2 text-2"><input type="text" name="fullNames" value=""/></div></div>);
339                 let seatInput = $(<div class="form-group row"><label class="col-lg-3 col-form-label form-control-label line-height-9 pt-2 text-2">Seat:</label><div class="col-lg-9 col-form-control form-control line-height-9 pt-2 text-2"><input type="text" name="seats" value=""/></div></div>);
340
341                 fullNameInput.find("input[name='fullNames']").val(rental.fullNames[index]);
342                 // console.log("Set fullName to: ", rental.fullNames[index]);
343
344                 seatInput.find("input[name='seats']").val(seat);
345                 // console.log("Set seat to: ", seat);
346
347                 // Insert the new inputs after the "Order E-Mail" field
348                 rentalForm.find('.form-group:has(input[name="email"]')).after(fullNameInput, seatInput);
349             });
350
351             // Add the form to the page
352             $('#rentalsContainer').append(rentalForm);
353             // console.log("Added form to rentalsContainer");
354
355             let divider = $('<div class="divider divider-xs divider-style-4 taller appear-animation" data-appear-animation="bounceIn" data-appear-animation-delay="300"><i class="fa fa-angle-down" style="font-size: 1.5em; color: #ccc; margin-right: 10px;"/></div>');
356             $('#rentalsContainer').append(divider);
357
358             rentalForm.find('button[name="resendEmail"]').click(function(event) {
359                 event.preventDefault(); // Prevent the default form submission which causes a page refresh
360
361                 $.post('/rentalInfo', { rental: rental }, function(response) {
362                     if(response.message === 'Email sent successfully.') {
363                         rentalForm.find('#emailSuccess').show(); // Show the success div
364                         rentalForm.find('#emailFail').hide(); // Ensure the failure div is hidden
365                     } else if(response.message === 'Rental is expired. No email sent.') {
366                         rentalForm.find('#emailFail').show(); // Show the failure div
367                         rentalForm.find('#emailSuccess').hide(); // Ensure the success div is hidden
368                     }
369                 });
370             });

```

For για κάθε order που βρήκε στο ID του συνδεδεμένου χρήστη

For μεσά στη πάνω for, που εμφανίζει δυναμικά τα στοιχεία του κάθε εισιτηρίου Ονομα+Θέση

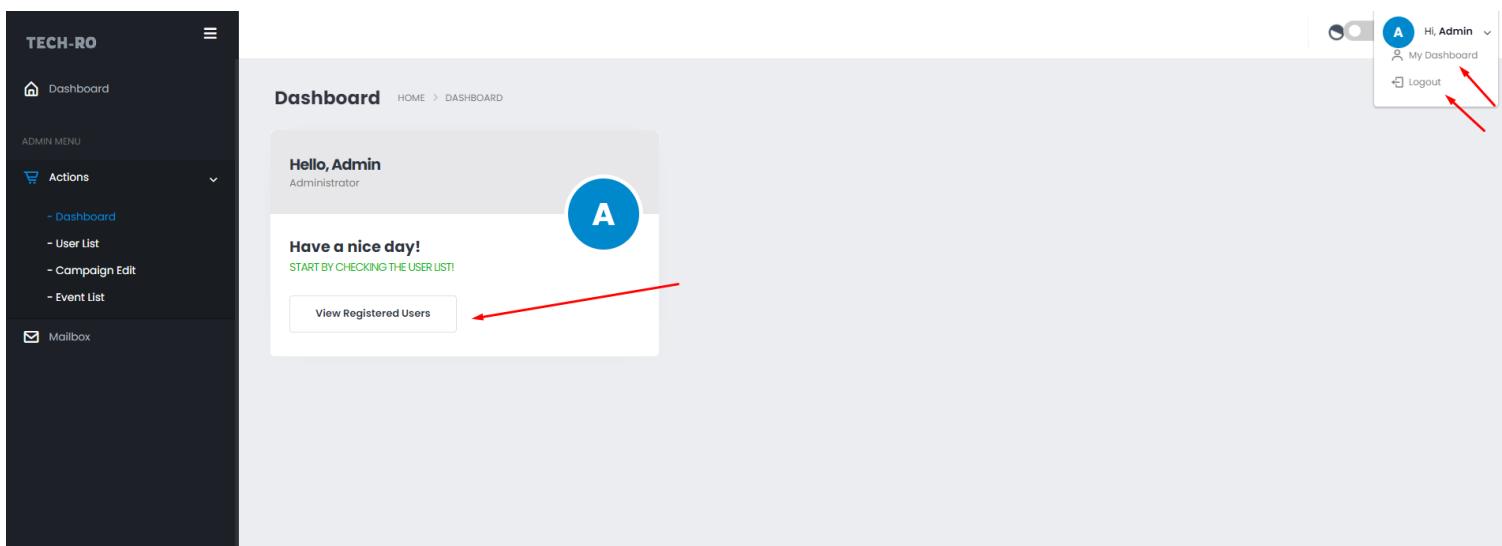
Όπως πάντα έλεγχος για active session και ανάδειξη σωστού content ή redirect:

```
508 | <!-- Session Check and Content Output -->
509 | <script>
510 |   $(document).ready(function() {
511 |     $.get('/check-session', function(data) {
512 |       if (!data.loggedIn) {
513 |         window.location.href = "/sign-in-up-page.html";
514 |       }
515 |       if (data.loggedIn) {
516 |         $("#accountDropDown").show();
517 |         $("#loginBtn").hide();
518 |         $("#registerBtn").hide();
519 |         $("#welcomeUsr").html(`<strong class="text-color-dark text-4">Welcome ${data.username}</strong>`);
520 |       } else {
521 |         $("#accountDropDown").hide();
522 |         $("#loginBtn").show();
523 |         $("#registerBtn").show();
524 |       }
525 |     });
526 |
527 |     $('#logoutBtn').click(function() {
528 |       $.post('/logout', function() {
529 |         location.reload();
530 |       });
531 |     });
532 |   });
533 | </script>
```

Συνεχίζουμε στο directory my_files<public<admin

10. Αρχική Σελίδα Διαχείρισης (page-admin-dashboard.html)

Η αρχική μας σελίδα διαχείρισης απαιτεί ο διαχειριστής να είναι συνδεδεμένος (ως διαχειριστής) για να αποκτήσει πρόσβαση. Όποτε, λειτουργικά, πρώτα συνδέεται και μετά προχωράει στην αρχική. (η σύνδεση θα αναλυθεί στο 11)



Η αρχική σελίδα διαχείρισης δεν έχει κάποια αλληλεπίδραση με την βάση δεδομένων.

Οπότε δεν χρησιμοποιούνται routes. Στο client-side κομμάτι έχουμε τα εξής:

```
57     <script>
58         $(document).ready(function() {
59             // Check if admin is logged in
60             $.get('/check-admin-session', function(data) {
61                 if (!data.loggedIn) {
62                     // If admin is not logged in, redirect to login page
63                     window.location.href = '/admin/page-admin-sign-in.html';
64                 }
65             });
66         });
67     </script>
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95     <!--Logout Button-->
96     <script>
97         $(document).ready(function() {
98             $('#adminLogout').click(function(event) {
99                 // Prevent the default action
100                event.preventDefault();
101
102                 $.post('/admin-logout', function(data) {
103                     // If the server responds successfully, redirect to the login page
104                     window.location.href = 'page-admin-sign-in.html';
105                 });
106             });
107         });
108     </script>
```

11. Σύνδεση Διαχειριστή (page-admin-sign-in.html)



Από πλευράς κώδικα, στο server-side κομμάτι έχουμε τα εξής endpoints:

/admin-Login route:

```
9  router.post( path: '/admin-login', handlers: (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj>, res : Response<ResBody, LocalsObj>) : void => {
10    const { adminLogin, adminPass } = req.body;
11
12    client.connect() Promise<MongoClient>
13      .then(() => {
14        const db : Db = client.db(dbName);
15        const collection : Collection<Document> = db.collection( name: 'adminLogin' );
16
17        // Try to find the admin with the provided username
18        return collection.findOne( filter: { username: adminLogin } );
19      }) Promise<>.
20      .then(admin : Document & ... => {
21        // If the admin doesn't exist or the password is incorrect, send an error response
22        if (!admin || admin.password !== adminPass) {
23          res.status( code: 401 ).json( body: { error: 'Invalid username or password' } );
24          // Stop further execution in this callback chain
25          throw new Error('Invalid username or password');
26        }
27
28        // The admin exists and the password is correct. Start a new session.
29        req.session.admin = { username: adminLogin };
30        res.status( code: 200 ).json( body: { message: 'Admin logged in successfully' } );
31      }) Promise<Promise<>>.
32      .catch(err => {
33        if (err.message !== 'Invalid username or password') {
34          console.error('Error:', err);
35          res.status( code: 500 ).json( body: { error: 'An error occurred.' } );
36        }
37      })
38      .finally( onFinally: () : void => {
39        client.close();
40      });
41};
```

/check-admin-session route:

```
43  router.get( path: '/check-admin-session', handlers: (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj>, res : Response<ResBody, LocalsObj>) : void => {
44    if (req.session.admin) {
45      res.send( body: { loggedIn: true, username: req.session.admin.username } );
46    } else {
47      res.send( body: { loggedIn: false } );
48    }
49  });
50};
```

/admin-logout route:

```
51  router.post( path: '/admin-logout', handlers: (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj>, res : Response<ResBody, LocalsObj>) : void => {
52    req.session.destroy();
53    res.status( code: 200 ).send( body: { message: 'Admin logged out successfully' } );
54});
```

Αξίζει να σημειωθεί ότι το session που δημιουργείται κατά την είσοδο είναι διαφορετικό από αυτό του User Interface κομματιού της διαδικτυακής εφαρμογής.

Στο client-side κομμάτι της σελίδας σύνδεσης έχουμε:

Διασταύρωση στοιχείων με collection `adminLogin`:

```
122 <script>
123     $(document).ready(function() {
124         $('#adminLoginForm').on('submit', function(e) {
125             e.preventDefault(); // Prevent the form from submitting normally
126
127             var adminLogin = $('#adminLogin').val();
128             var adminPass = $('#adminPass').val();
129
130             $.ajax({
131                 url: '/admin-login',
132                 method: 'POST',
133                 data: { adminLogin: adminLogin, adminPass: adminPass },
134                 success: function(response) {
135                     // Login was successful, redirect to the admin dashboard
136                     window.location.href = '/admin/page-admin-dashboard.html';
137                 },
138                 error: function(xhr, status, error) {
139                     // Login failed, show an error message
140                     alert('Login failed: ' + xhr.responseText.error);
141                 }
142             });
143         });
144     });
145 </script>
```

The screenshot shows the MongoDB interface for the `campaignDB.adminLogin` collection. The top navigation bar includes tabs for `Documents`, `Aggregations`, `Schema`, `Explain Plan`, `Indexes`, and `Validation`. The `Documents` tab is selected. Below the navigation is a search bar with the placeholder "Type a query: { field: 'value' }". At the bottom of the interface are two buttons: `ADD DATA` and `EXPORT DATA`. A single document is displayed in a modal-like box at the bottom:

```
_id: ObjectId('6494c086b29db4a36648615c')
username: "admin"
password: "admin"
```

Αν ο διαχειριστής είναι ήδη συνδεδεμένος, δεν μπορεί να ξανακάνει sign in:

```
38      <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
39
40      <script>
41          $(document).ready(function() {
42              // Check if admin is already logged in
43              $.get('/check-admin-session', function(data) {
44                  if (data.loggedIn) {
45                      // If admin is logged in, redirect to dashboard
46                      window.location.href = '/page-admin-dashboard.html';
47                  }
48              });
49      </script>
```

12. Σελίδα Επεξεργασίας Χρηστών

Η σελίδα επεξεργασίας χρηστών «σπάει» σε 2 κομμάτια

- Σελίδα Λίστας όλων των Χρηστών με δυνατότητα επιλογής συγκεκριμένου χρήστη (page-admin-user-list.html)
- Σελίδα Πληροφοριών επιλεγμένου Χρήστη (page-admin-user-details.html?id=)

Ξεκινάμε από την πρώτη, page-admin-user-list.html

ID	Full Name	E-mail	Events Booked
648b02876a202bffa959e491	Not provided	Not provided	4
6495c81fa7124c4937eb6a10	Not provided	Not provided	0
6496d379971e8679e9c5649a	Not provided	Not provided	0
6497410da0d473579269235e	Not provided	Not provided	0
649748cda0d473579269235f	Not provided	Not provided	0

Από πλευράς κώδικα, έχουμε στο server-sided κομμάτι τα εξής endpoints:

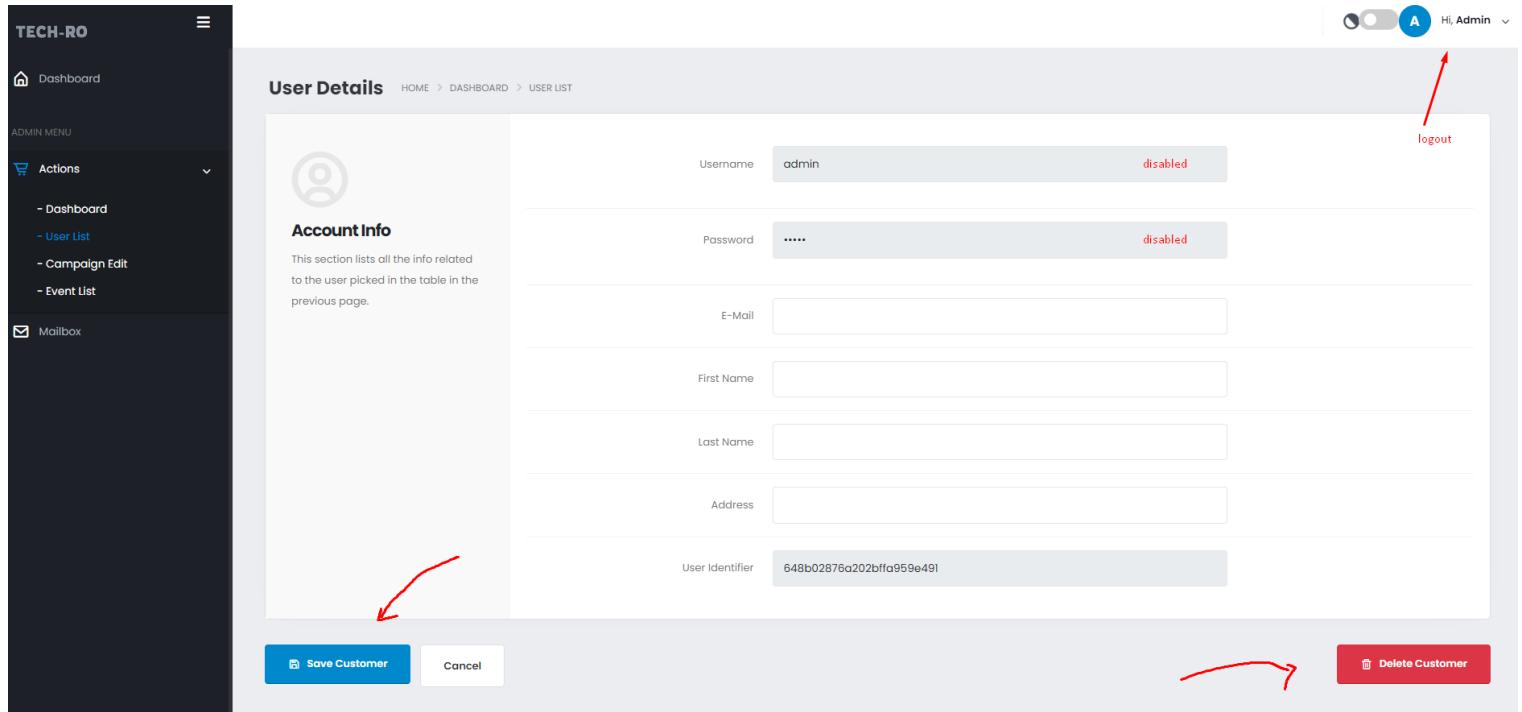
/get-Users route:

```
9  //Get Users to output to admin user list
10 router.get( path: '/get-users', handlers: async (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res : Response<ResBody, LocalsObj> ) : Promise<void> => {
11   try {
12     const db :Db = client.db(dbName);
13     const userCollection :Collection<Document> = db.collection( name: 'userList');
14     const rentalCollection :Collection<Document> = db.collection( name: 'rentalList');
15
16     // Get all users from the collection
17     const users :(WithId<...>)[] = await userCollection.find( filter: {}).toArray();
18
19     // Loop over each user to count the events booked
20     for(let user :WithId<Document> of users){
21       const count :number = await rentalCollection.countDocuments( filter: {userId: new ObjectId(user._id)});
22       user.eventsBooked = count;
23     }
24
25     // Return the list of users
26     res.status( code: 200).json(users);
27   } catch(err) {
28     console.error('Error:', err);
29     res.status( code: 500).json( body: { error: 'An error occurred.' });
30   }
31 }
```

Από πλευράς client-side, έχουμε αντίστοιχα:

```
318 $(document).ready(function() {
319   var table = $('#datatable-ecommerce-list').DataTable();
320
321   1 usage
322   function getUsers() {
323     $.ajax({
324       type: "GET",
325       url: '/get-users',
326       dataType: "json",
327       success: function (response) {
328         // Clear DataTable
329         table.clear();
330
331         $.each(response, function (index, user) {
332           var fullName = (user.firstName || user.lastName) ? `${user.firstName} ${user.lastName}` : 'Not provided';
333
334           // Add row through DataTable API
335           table.row.add([
336             `
```

Συνεχίζουμε με τη δεύτερη σελίδα, τη page-admin-user-details.html



Από πλευράς κώδικα, και συγκεκριμένα server-side έχουμε τα εξής endpoints:

/get-user route:

```

9 //Get User route to display to admin form
10 router.get( path: '/get-user/:id' , handlers: async (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res : Response<ResBody, LocalsObj> ) : Promise<void> => {
11   try {
12     const db :Db = client.db(dbName);
13     const collection :Collection<Document> = db.collection( name: 'userList');
14
15     const user :Document & {...} = await collection.findOne( filter: {_id: new ObjectId(req.params.id)} );
16
17     res.status( code: 200).json(user);
18   } catch(err) {
19     console.error('Error:', err);
20     res.status( code: 500).json( body: { error: 'An error occurred.' });
21   }
22 })
  
```

/update-user route:

```

24 //update user route from admin form
25 router.put( path: '/update-user/:id' , handlers: async (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res : Response<ResBody, LocalsObj> ) : Promise<void> => {
26   try {
27     const db :Db = client.db(dbName);
28     const collection :Collection<Document> = db.collection( name: 'userList');
29
30     // Update the user
31     await collection.updateOne(
32       filter: {_id: new ObjectId(req.params.id)},
33       update: {$set: req.body}
34     );
35
36     res.status( code: 200).json( body: { message: 'User updated.' });
37   } catch(err) {
38     console.error('Error:', err);
39     res.status( code: 500).json( body: { error: 'An error occurred.' });
40   }
41 })
  
```

/delete-user route:

```
43  //Delete route that deletes user from admin page
44  router.delete( path: '/delete-user/:id', handlers: async (req : Request<P, ResBody, ReqQuery, LocalsObj>, res : Response<ResBody, LocalsObj>) : Promise<void> => {
45    try {
46      const db : Db = client.db(dbName);
47      const collection : Collection<Document> = db.collection( name: 'userList' );
48
49      // Delete the user
50      await collection.deleteOne( filter: { _id: new ObjectId(req.params.id) });
51
52      res.status( code: 200).json( body: { message: 'User deleted.' });
53    } catch(err) {
54      console.error('Error:', err);
55      res.status( code: 500).json( body: { error: 'An error occurred.' });
56    }
57  });

```

Βλέπουμε ξεκάθαρα τις αλληλεπιδράσεις με το collection `userList`

Στο client-side έχουμε τα εξής:

```
328          // Wait until the document is fully loaded
329          $(document).ready(function() {
330
331            // Get the URL parameters
332            let params = (new URL(document.location)).searchParams;
333            // Get the ID from URL
334            let id = params.get('id');
335
336            // Fetch user data and populate form fields
337            $.ajax({
338              type: "GET",
339              url: '/get-user/' + id,
340              dataType: "json",
341              success: function (response) {
342                // Populate form fields
343                $('#userUsername').val(response.username);
344                $('#userPassword').val(response.password);
345                $('#userEmail').val(response.email);
346                $('#userFirstName').val(response.firstName);
347                $('#userLastName').val(response.lastName);
348                $('#userAddress').val(response.address);
349                $('#userId').val(response._id);
350
351            });
352        });

```

```
353     // Event handler for the "Save Changes" button
354     $('#saveUserBtn').on('click', function(event) {
355         event.preventDefault();
356
357         // Collect form data
358         let formData = {
359             email: $('#userEmail').val(),
360             firstName: $('#userFirstName').val(),
361             lastName: $('#userLastName').val(),
362             address: $('#userAddress').val(),
363         };
364
365         // Send a PUT request
366         $.ajax({
367             type: "PUT",
368             url: '/update-user/' + id,
369             data: JSON.stringify(formData),
370             contentType: "application/json",
371             success: function (response) {
372                 alert('User updated successfully!');
373                 location.reload();
374             }
375         });
376
377     });
378
```

```
379     // Event handler for the "Delete User" button
380     $('#deleteUserBtn').on('click', function(event) {
381         event.preventDefault();
382
383         // Confirm user action
384         if (confirm('Are you sure you want to delete this user? This action cannot be undone.')) {
385             // Send a DELETE request
386             $.ajax({
387                 type: "DELETE",
388                 url: '/delete-user/' + id,
389                 success: function (response) {
390                     alert('User deleted successfully!');
391                     window.location.href = '/admin/page-admin-user-list.html';
392                 }
393             });
394         }
395     });
396
```

13. Σελίδα Επεξεργασίας Πληροφοριών καμπάνιας (page-admin-campaign-edit.html)

The screenshot shows the 'Edit Campaign' interface. On the left is a sidebar with 'TECH-RO' branding and navigation links for Dashboard, Actions (Dashboard, User List, Campaign Edit, Event List), and Mailbox. The main area has a title 'Edit Campaign' with a breadcrumb path: HOME > DASHBOARD > CAMPAIGN. It contains sections for 'Campaign Info' (with a note about image format) and 'Product Description'. Two red arrows point from the 'Product Description' section to the text in the 'About Us' field. Another red arrow points from the bottom right of the 'About Us' field to the 'Save Changes' button.

Στο κομμάτι του κώδικα, πιο συγκεκριμένα στο server-side έχουμε τα παρακάτω endpoints:

/adminAboutUs get route:

```

10  router.get( path: '/adminAboutUs', handlers: (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res : Response<ResBody, LocalsObj> ) : void => {
11    client.connect() Promise<MongoClient>
12    .then(() => {
13      const db : Db = client.db(dbName);
14      const collection : Collection<Document> = db.collection( name: 'aboutUsDynamic' );
15      return collection.findOne( filter: {} );
16    }) Promise<...>
17    .then(data : Document & ... ) => {
18      res.json(data);
19    }) Promise<Promise<...>>
20    .catch(err => {
21      console.error('Error:', err);
22      res.status( code: 500).json( body: { error: 'An error occurred.' });
23    })
24    .finally( onFinally: () : void => {
25      client.close();
26    });
27  });

```

/adminAboutUs post route:

```
29   router.post( path: '/adminAboutUs' , handlers: (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res : Response<ResBody, LocalsObj> ) : void =>
30     client.connect() Promise<MongoClient>
31     .then(() => {
32       const db : Db = client.db(dbName);
33       const collection : Collection<Document> = db.collection( name: 'aboutUsDynamic');
34       return collection.updateOne(
35         filter: {},
36         update: { $set: req.body }
37       );
38     }) Promise<UpdateResult<...>>
39     .then(() : void => {
40       res.json( body: { message: "Successfully updated!" });
41     }) Promise<MongoClient>
42     .catch(err => {
43       console.error('Error:', err);
44       res.status( code: 500).json( body: { error: 'An error occurred.' });
45     })
46     .finally( onFinally: () : void => {
47       client.close();
48     });
49   );
});
```

Στο κομμάτι του client-side έχουμε:

```
321 <script>
322   // Fetching the existing values when the page loads
323   $(document).ready(function() {
324     $.ajax({
325       url: "/adminAboutUs",
326       method: "GET",
327       success: function(data) {
328         // Populate the form fields with the fetched data
329         $('#whoAreWe1').val(data.whoAreWe1);
330         $('#whoAreWe2').val(data.whoAreWe2);
331         $('#image1').val(data.aboutUsImages.image1);
332         $('#image2').val(data.aboutUsImages.image2);
333         $('#image3').val(data.aboutUsImages.image3);
334         $('#image4').val(data.aboutUsImages.image4);
335       }
336     });
337   });
});
```

```

339     // Sending the updated values to the server when the form is submitted
340     $('form').on('submit', function(e) {
341         e.preventDefault();
342
343         $.ajax({
344             url: "/adminAboutUs",
345             method: "POST",
346             data: {
347                 whoAreWe1: $('#whoAreWe1').val(),
348                 whoAreWe2: $('#whoAreWe2').val(),
349                 aboutUsImages: {
350                     image1: $('#image1').val(),
351                     image2: $('#image2').val(),
352                     image3: $('#image3').val(),
353                     image4: $('#image4').val(),
354                 }
355             },
356             success: function(response) {
357                 alert(response.message);
358             }
359         });
360     });

```

Όπως πάντα, γίνεται έλεγχος admin session, και ανάλογο redirect:

```

59
60         <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
61         <!--Session check or redirect-->
62         <script>
63             $(document).ready(function() {
64                 // Check if admin is logged in
65                 $.get('/check-admin-session', function(data) {
66                     if (!data.loggedIn) {
67                         // If admin is not logged in, redirect to login page
68                         window.location.href = '/admin/page-admin-sign-in.html';
69                     }
70                 });
71             });
72         </script>

```

14. Σελίδα Επεξεργασίας Εκδηλώσεων

Όπως και με την σελίδα επεξεργασίας χρηστών, έτσι και εδώ, η λειτουργικότητα 14 χωρίζεται σε 2 υποσελίδες:

- Σελίδα εμφάνισης όλων των διαθέσιμων εκδηλώσεων στην βάση δεδομένων δυναμικά (page-admin-event-list.html), και
- Σελίδα εμφάνισης πληροφοριών της επιλεγμένης εκδήλωσης, όπου ο διαχειριστής μπορεί να αλλάξει πληροφορίες και να τις διαγράψει αν το επιθυμεί

Ξεκινώντας με την πρώτη, page-admin-event-list.html έχουμε:

The screenshot shows the 'Event List' page from a web application. The left sidebar has sections for Dashboard, Admin Menu (Actions, Dashboard, User List, Campaign Edit, Event List), and Mailbox. The main area shows a table of events with columns: Event Name, Date, and Total Tickets. A red arrow points to the 'Event Name' column header. Another red arrow points to the first event row, which contains a link labeled 'Click για μεμφάνιση πληροφοριών'. A third red arrow points to the 'Total Tickets' column header. At the bottom, it says 'Showing 1 to 5 of 5 entries'. The top right shows a user profile with 'Hi, Admin' and a 'Logout' button.

Event Name	Date	Total Tickets
Event 1	Mon Dec 25 2023	0
Event 2	Mon Dec 25 2023	15
Event 3	Mon Dec 25 2023	3
Event 4	Sun Dec 25 2022	2
Event 5	Sun Dec 25 2022	0

Από πλευράς κώδικα, στο server-sided κομμάτι έχουμε τα εξής endpoints:

/eventList get route:

```
9   router.get( path: '/eventList', handlers: async (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj>, res : Response<ResBody, LocalsObj>) : Promise<void> => {
10     try {
11       await client.connect();
12       const db : Db = client.db(dbName);
13       const eventCollection : Collection<Document> = db.collection( name: 'eventList');
14
15       // Fetch all events
16       const events : (WithId<...>)[] = await eventCollection.find().toArray();
17
18       // For each event, fetch total rentals and append to the event object
19       for (let event : WithId<Document> of events) {
20         let totalRentals : number = 0;
21         // Iterate over each ticket type and sum up the booked seats
22         for (let ticketType in event.tickets) {
23           totalRentals += event.tickets[ticketType].bookedSeats.length;
24         }
25         event.totalRentals = totalRentals;
26       }
27
28       res.json(events);
29
30     } catch (err) {
31       console.error('Error:', err);
32       res.status( code: 500).json( body: { error: 'An error occurred.' });
33     }
34   });

```

Στην πλευρά του client έχουμε:

```
310 <script>
311 $(document).ready(function() {
312   let table = $('#datatable-ecommerce-list').DataTable();
313
314   $.ajax({
315     url: "/eventList",
316     method: "GET",
317     success: function(data) {
318       table.clear();
319
320       data.forEach(event => {
321         const row = [
322           `<a href="page-admin-event-details.html?id=${event._id}"><strong>${event._id}</strong></a>` ,
323           `<a href="page-admin-event-details.html?id=${event._id}"><strong>${event.eventName}</strong></a>` ,
324           `${new Date(event.eventDate).toDateString()}`,
325           `${event.totalRentals}`
326         ];
327
328         table.row.add(row);
329       });
330
331       table.draw();
332     }
333   });
334 });
335 </script>
```

Συνεχίζοντας με την δεύτερη σελίδα, page-admin-event-details.html

The screenshot shows the 'Event Details' page of a web application. The page has a header with 'Event Details' and a breadcrumb trail 'HOME > DASHBOARD > EVENTS'. On the left is a dark sidebar with 'TECH-RO' logo, 'Dashboard', 'Actions' (with 'User List', 'Campaign Edit', 'Event List'), and 'Mailbox'. The main content area has a form for event details. Fields include 'Event Name' (Event 1), 'Event Description' (This is a description of Event 1), 'Date Created' (2023-12-25), 'Latitude' (37.9570912), 'Longitude' (23.7127157), and 'Image' (img/events/event1.jpg). At the bottom are 'Save Event' and 'Cancel' buttons, and a 'Delete Event' button on the right. Red arrows highlight the 'Logout' links in the top right and bottom right corners.

Από πλευράς κώδικα έχουμε στο server-side τα εξής endpoints:

/eventList/:eventId get route:

```
9   router.get( path: '/eventList/:eventId' , handlers: async (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res : Response<ResBody, LocalsObj> ) : Promise<void> => {
10     try {
11       await client.connect();
12       const db :Db = client.db(dbName);
13       const eventCollection :Collection<Document> = db.collection( name: 'eventList' );
14
15       // Fetch the specific event
16       const event :Document &{...} = await eventCollection.findOne( filter: { _id: new ObjectId(req.params.eventId) });
17
18       // Send the event data to the client
19       res.json(event);
20
21     } catch (err) {
22       console.error('Error:', err);
23       res.status( code: 500).json( body: { error: 'An error occurred.' });
24     }
25   });
}
```

/eventList/:eventId put route:

```
28 router.put(path: '/eventList/:eventId', handlers: async (req: Request<P, ResBody, ReqBody, ReqQuery, LocalsObj>, res: Response<ResBody, LocalsObj>) : Promise<void> => {
29   try {
30     await client.connect();
31     const db: Db = client.db(dbName);
32     const eventCollection: Collection<Document> = db.collection(name: 'eventList');
33
34     // Update the specific event
35     await eventCollection.updateOne(filter: {_id: new ObjectId(req.params.eventId)}, update: { $set: req.body });
36
37     res.json(body: { message: 'Event updated successfully.' });
38
39   } catch (err) {
40     console.error('Error:', err);
41     res.status(code: 500).json(body: { error: 'An error occurred.' });
42   }
43 }
```

/eventList/:eventId delete route:

```
46 router.delete(path: '/eventList/:eventId', handlers: async (req: Request<P, ResBody, ReqBody, ReqQuery, LocalsObj>, res: Response<ResBody, LocalsObj>) : Promise<void> => {
47   try {
48     await client.connect();
49     const db: Db = client.db(dbName);
50     const eventCollection: Collection<Document> = db.collection(name: 'eventList');
51
52     // Delete the specific event
53     await eventCollection.deleteOne(filter: {_id: new ObjectId(req.params.eventId)});
54
55     res.json(body: { message: 'Event deleted successfully.' });
56
57   } catch (err) {
58     console.error('Error:', err);
59     res.status(code: 500).json(body: { error: 'An error occurred.' });
60   }
61 }
```

Στο client-side έχουμε, get:

```
395 <script>
396   $(document).ready(function() {
397     // Parse the event id from the url
398     const urlParams = new URLSearchParams(window.location.search);
399     const eventId = urlParams.get('id');
400
401     $.ajax({
402       url: '/eventList/${eventId}', // fetch data for specific event
403       method: "GET",
404       success: function(event) {
405         // Populate the form with the event data
406         $('#eventName').val(event.eventName);
407         $('#eventDescription').val(event.eventDescription);
408
409         const eventDate = new Date(event.eventDate);
410         $('#eventDate').val(eventDate.toISOString().split('T')[0]); // YYYY-MM-DD format
411         $('#eventHour').val(eventDate.getHours());
412         $('#eventMinute').val(eventDate.getMinutes());
413
414         $('#eventLatitude').val(event.eventCoordinates.lat);
415         $('#eventLongitude').val(event.eventCoordinates.long);
416         $('#eventImage').val(event.eventImage);
417     }
418   });
419 </script>
```

Put:

```
445 <script>
446     $(document).ready(function() {
447         const urlParams = new URLSearchParams(window.location.search);
448         const eventId = urlParams.get('id');
449
450         $('#saveEventBtn').click(async function(e) {
451             e.preventDefault();
452
453             const event = {
454                 eventName: $('#eventName').val(),
455                 eventDescription: $('#eventDescription').val(),
456                 eventDate: new Date($('#eventDate').val() + ' ' + $('#eventHour').val() + ':' + $('#eventMinute').val()).toISOString(),
457                 eventCoordinates: {
458                     lat: parseFloat($('#eventLatitude').val()),
459                     long: parseFloat($('#eventLongitude').val())
460                 }
461             };
462
463             const response = await fetch('/eventList/' + eventId, {
464                 method: 'PUT',
465                 headers: {
466                     'Content-Type': 'application/json'
467                 },
468                 body: JSON.stringify(event)
469             });
470         });
471     });
472 
```

Delete:

```
486 $('#deleteEventBtn').click(async function(e) {
487     e.preventDefault();
488
489     const confirmed = await Swal.fire({
490         title: 'Are you sure?',
491         text: "You won't be able to revert this!",
492         icon: 'warning',
493         showCancelButton: true,
494         confirmButtonColor: '#3085d6',
495         cancelButtonColor: '#d33',
496         confirmButtonText: 'Yes, delete it!'
497     });
498
499     if (confirmed.isConfirmed) {
500         const response = await fetch('/eventList/' + eventId, {
501             method: 'DELETE'
502         });
503
504         if (response.ok) {
505             Swal.fire(
506                 'Deleted!',
507                 'Event deleted successfully.',
508                 'success'
509             ).then(() => {
510                 window.location.href = 'page-admin-event-list.html';
511             });
512         } else {
513             Swal.fire(
514                 'Error!',
515                 'An error occurred while deleting the event.',
516                 'error'
517             );
518         }
519     }
520 });
521 
```

Όπως και προηγουμένως, έχουμε έλεγχο session και ανάλογο redirect:

```
56      <!--Session check or redirect-->
57      <script>
58          $(document).ready(function() {
59              // Check if admin is logged in
60              $.get('/check-admin-session', function(data) {
61                  if (!data.loggedIn) {
62                      // If admin is not logged in, redirect to login page
63                      window.location.href = '/admin/page-admin-sign-in.html';
64                  }
65              });
66      </script>
```

15. Media Queries

Γενικά στη διαδικτυακή εφαρμογή μας χρησιμοποιήθηκαν πολλές βιβλιοθήκες και media queries για να επιτευχθεί το responsiveness κομματι. Μερικά από τα media Queries που χρησιμοποιήθηκαν στην εργασία είναι τα παρακάτω, και βρίσκονται στο αρχείο `theme.css`:

```
1396      @media (max-width: 991px) {
1397          html #header.header-transparent .header-body {
1398              overflow: hidden;
1399      }
```

```
1161      @media (max-width: 991px) {
1162          #header.header-transparent .header-nav-features .header-nav-features-search .header-nav-features-dropdown.show {
1163              top: -50px !important;
1164              box-shadow: none !important;
1165          }
```

```

@media (min-width: 992px) {
    html.boxed #header .header-top.header-top-colored {
        margin-top: -18px;
        border-radius: 4px 4px 0 0;
    }

    html.boxed.sticky-header-active #header:not(.header-effect-shrink) .header-body {
        position: fixed !important;
        padding-left: 15px;
        padding-right: 15px;
    }

    html.boxed.sticky-header-active #header:not(.header-effect-shrink) .header-nav-bar {
        margin: 0 -15px -9px;
    }
}

```

```

@media (max-width: 1199px) {
    html.boxed .footer-reveal {
        border-bottom: none !important;
    }
}

```

16. Λειτουργικότητες Σύγχρονων Τάσεων

JavaScript Frameworks

Οπως αναφέρθηκε σε προηγούμενο κεφάλαιο, για λειτουργικότητα αυτης της τάσης χρησιμοποιήθηκε η AngularJS κυρίως στο κομμάτι του sign-in και register (βλέπε `sign-in-up-page.html`)

```

330 //SIGN-IN UP PAGE
331 //LOGIN CONTROLLER ~ SHOW SUCCESS OR FAIL NOTIFICATION
332 .controller('LoginController', function ($scope, $http, $timeout) {
333     $scope.loginUser = function () {
334         const username = document.querySelector('.loginUsername').value;
335         const password = document.querySelector('.loginPassword').value;
336     }
}

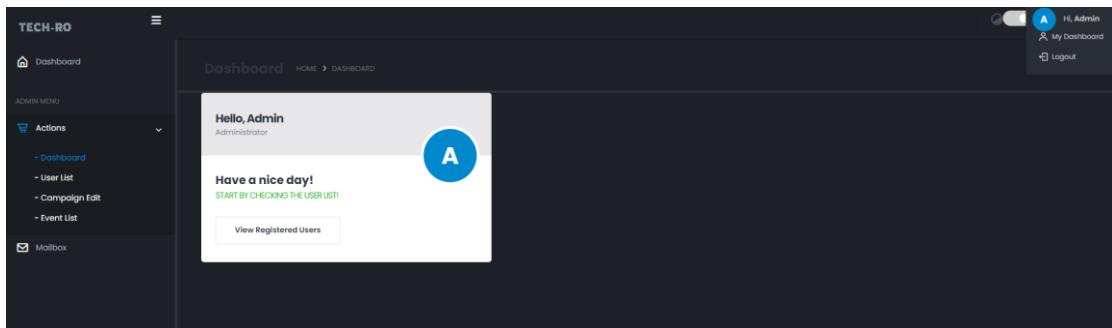
292 //REGISTER
293 //REGISTER CONTROLLER
294 .controller('RegistrationController', function ($scope, $http, $timeout) {
295     $scope.registerUser = function () {
296         const username = document.querySelector('.username').value;
297         const password = document.querySelector('.password').value;
298     }
}

```

Dark Mode

To dark-mode χρησιμοποιήθηκε καθ'όλη τη διαδικτυακή εφαρμογή με χρήση της μεταβλητής localStorage που δεν «χάνεται» με το κλήσιμο του browser. Έτσι ο χρήστης χωρίς να χρειάζεται να είναι συνδεμένος, καθ'όλη την διάρκεια της χρήσης των ιστοσελιδών έχει σταθερά ενεργοποιημένο το dark mode είτε στο user interface είτε στο admin interface. Ο τρόπος με τον οποίο έγινε αυτό είναι ο ακόλουθος και είναι ίδιος σε όλες τις σελίδες:

```
56      <script>
57          $(document).ready(function() {
58              // Get the saved dark mode state from LocalStorage, if it exists
59              var darkMode = localStorage.getItem('darkMode');
60
61              // If darkMode exists and is 'true', add the .dark class and check the switch
62              if (darkMode === 'true') {
63                  $('html').addClass('dark');
64                  $('#darkModeBtn').prop('checked', true);
65              }
66
67              // Listen for switch state changes
68              $('#darkModeBtn').change(function() {
69                  // If the switch is checked, add the .dark class and save the state
70                  if ($(this).is(':checked')) {
71                      $('html').addClass('dark');
72                      localStorage.setItem('darkMode', 'true');
73                  }
74                  // If the switch is not checked, remove the .dark class and save the state
75                  else {
76                      $('html').removeClass('dark');
77                      localStorage.setItem('darkMode', 'false');
78                  }
79              });
80      });
81  </script>
```



The screenshot shows the homepage of the Tech-Ro website. At the top left is the Tech-Ro logo, which consists of a stylized 'T' and 'R' inside a triangle. To the right is a search bar with a placeholder 'Search...', a magnifying glass icon, and a toggle switch. The main navigation menu includes 'Home', 'About Us', 'Events', and 'Admin' on the left, and 'Sign In' and 'Register' on the right. Below the menu, the heading 'All Upcoming Events:' is displayed in blue. Three event cards are shown in a grid:

- Event 1**: December 25, 2023, 21:20:00. **TICKET A: 5 TICKET B: 10 TICKET C: 15**
- Event 2**: December 25, 2023, 20:00:00. **TICKET A: 0 TICKET B: 9**
- Event 3**: December 25, 2023, 20:00:00. **TICKET A: 5 TICKET B: 10 TICKET C: 12**

The screenshot shows the login and register pages of the Tech-Ro website. Both pages have a dark background and feature the Tech-Ro logo at the top left. A search bar with placeholder 'Search...', a magnifying glass icon, and a toggle switch are positioned at the top right.

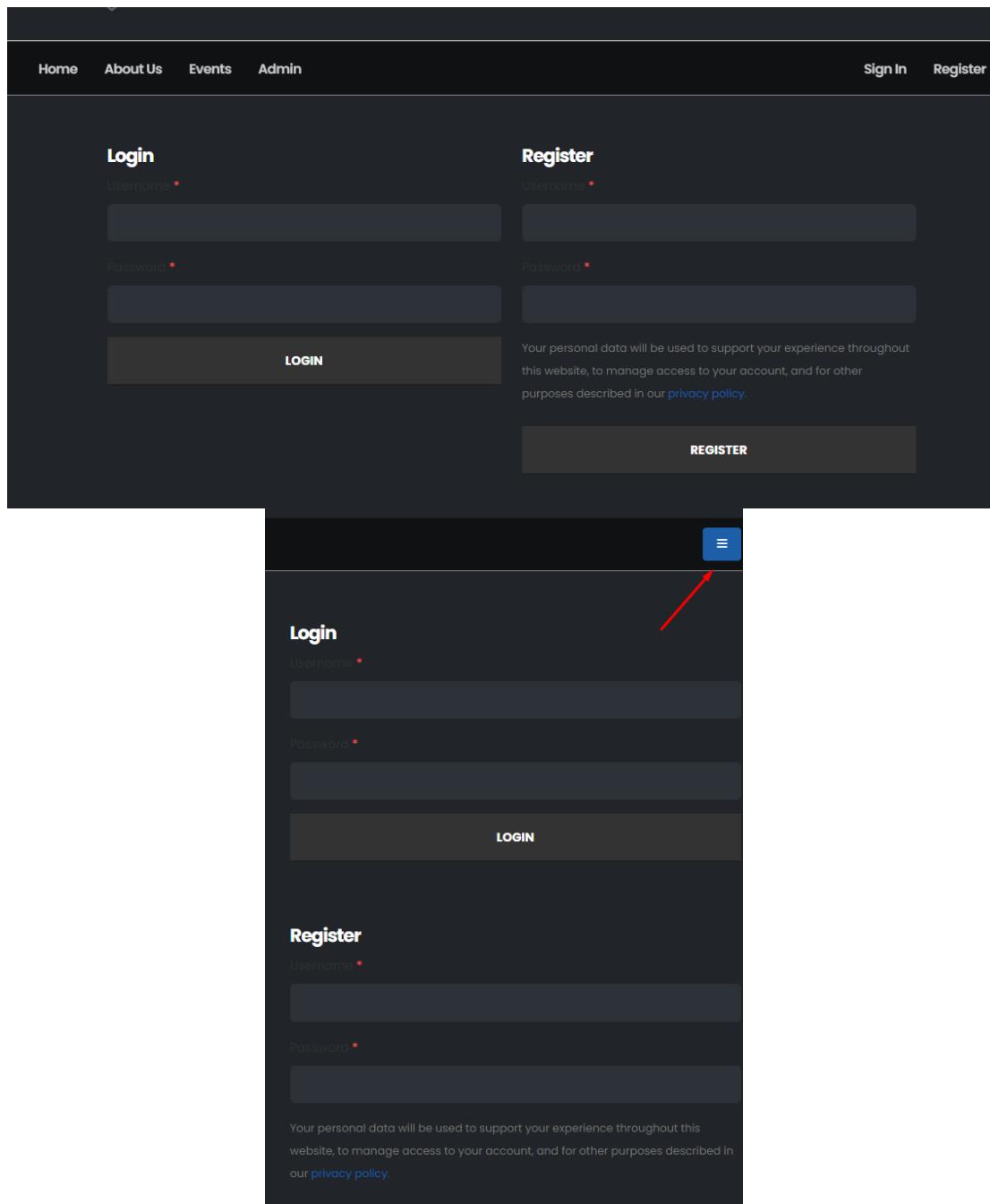
Login (Left):
A form with two input fields: 'Username *' and 'Password *'. Below the password field is a large grey button labeled 'LOGIN'.

Register (Right):
A form with two input fields: 'Username *' and 'Password *'. Below the password field is a note: 'Your personal data will be used to support your experience throughout this website, to manage access to your account, and for other purposes described in our [privacy policy](#)'. Below the note is a large grey button labeled 'REGISTER'.

Responsive Web Design

Παρομοίως με το Dark Mode, το RWD είναι κάτι που χρησιμοποιήθηκε καθ'όλη τη διαδικτυακή εφαρμογή* Όλες οι ιστοσελίδες είναι responsive για πολλά είδη συσκευών και αναλύσεων με κατάλληλα icons στη κάθε περίπτωση και σωστά grids έτσι ώστε να μην «χάνεται» ή συμπιέζεται υλικό.

`sign-in-up-page.html`



`aboutus.html`

The screenshot shows the homepage of a website named "Tech-Ro". The header features a logo with a stylized "T" and "R" inside a diamond shape, followed by the text "Tech-Ro". A search bar and a dark mode toggle are also in the header. Below the header is a navigation bar with links for "HOME", "ABOUT US", "EVENTS", and "ADMIN". On the right side of the header are "SIGN IN" and "REGISTER" buttons. The main content area has a blue header bar with the text "The New Way to Success". Below it is a sub-header: "Unleash Your Potential, Embrace Innovation: Propel Your Journey in the Dynamic World of Technology". A "ATTEND AN EVENT!" button is located on the right. The background of the main content area shows a photograph of several modern skyscrapers. To the right of the image is a section titled "Who We Are" with a paragraph of text describing the company's mission and services.

The New Way to Success

Unleash Your Potential, Embrace Innovation: Propel Your Journey in the Dynamic World of **Technology**.

ATTEND AN EVENT!

Who We Are

Our company thrives at the intersection of technology and human interaction, engineering unforgettable experiences that foster engagement, promote innovative ideas, and encourage the exchange of knowledge. We specialize in conceptualizing, designing, and executing diverse tech-centered events, from professional conferences and trade shows to virtual reality experiences and esports tournaments.

The screenshot shows the "About Us" page of the Tech-Ro website. It features a sidebar with dropdown menus for "HOME", "ABOUT US", "EVENTS", "ADMIN", "SIGN IN", and "REGISTER". The main content area contains a large image of a crowd of people at a technology event, with a prominent Apple logo in the background. Below the image is a section titled "Who We Are" with a paragraph of text describing the company's mission and services.

Who We Are

Our company thrives at the intersection of technology and human interaction, engineering unforgettable experiences that foster engagement, promote innovative ideas, and encourage the exchange of knowledge. We specialize in conceptualizing, designing,

The screenshot shows a web application interface for managing events. On the left, there is a sidebar with various menu items like 'Dashboard', 'Events', 'Campaign Edit', etc. The main area is titled 'Event List' and shows a table with five rows of event data. A blue button labeled '+ Add Order' is located at the top left of the table area.

ID	Event Name	Date	Total Tickets
6487b3851a7219db941dccc8	Event 1	Mon Dec 25 2023	0
6488689a6d62c5a722a20485	Event 2	Mon Dec 25 2023	15
648869736d62c5a722a20486	Event 3	Mon Dec 25 2023	3
648869756d62c5a722a20487	Event 4	Sun Dec 25 2022	2
6488697b6d62c5a722a2048a	Event 5	Sun Dec 25 2022	0

Showing 1 to 5 of 5 entries

This image shows the same event list data as the desktop screenshot, but presented in a mobile-friendly layout. At the top, there is a user profile icon with the text 'Hi, Admin' and a toggle switch. Below this is a large, semi-transparent title 'Event List'. The '+ Add Order' button is also present here.

ID	Event Name
6487b3851a7219db941dccc8	Event 1
6488689a6d62c5a722a20485	Event 2
648869736d62c5a722a20486	Event 3
648869756d62c5a722a20487	Event 4
6488697b6d62c5a722a2048a	Event 5

Showing 1 to 5 of 5 entries

17. Επιπλέον Λειτουργικότητες

i. Contact Us Page

Contact Us

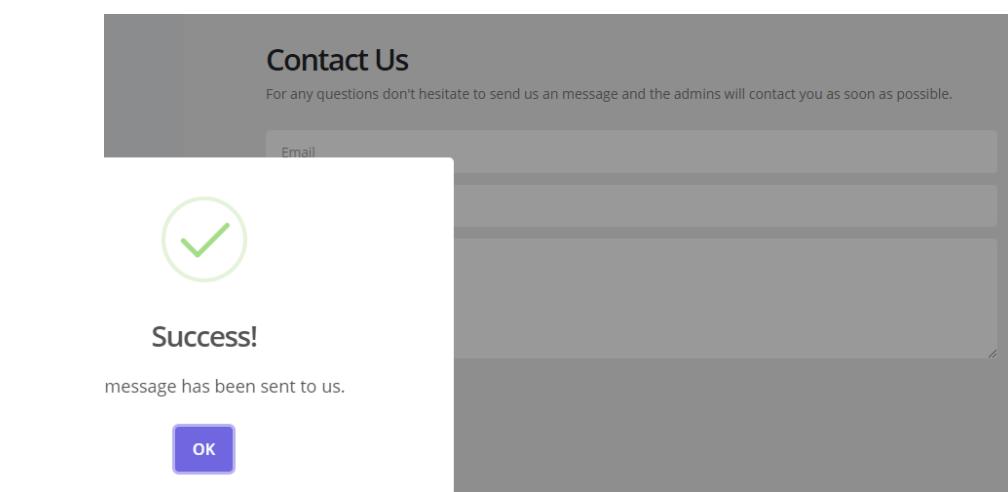
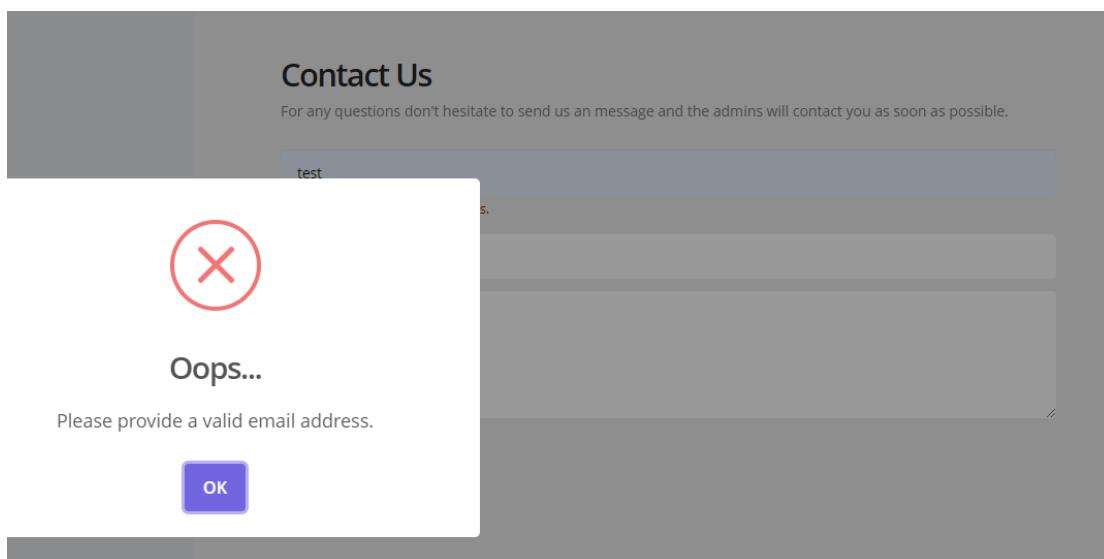
For any questions don't hesitate to send us a message and the admins will contact you as soon as possible.

Email

Subject

Message

SUBMIT



Από τη πλευρά κώδικα δημιουργήθηκε ένα collection `mailbox` στο οποίο αποστέλνονται τα μηνύματα. Πιο λεπτομερώς στο κώδικα:

```

763     // Send the request
764     $.ajax({
765         url: "/mailbox",
766         type: "POST",
767         data: {
768             email: $('#contactUsEmail').val(),
769             mailTitle: $('#contactUsTitle').val(),
770             mailContent: $('#contactUsContent').val(),
771             mailDate: new Date().toISOString()
772         },
773         success: function() {
774             // Show success notification and reset fields
775             Swal.fire(
776                 'Success!',
777                 'Your message has been sent to us.',
778                 'success'
779             );
780             $('#contactUsEmail').val('');
781             $('#contactUsTitle').val('');
782             $('#contactUsContent').val('');
783         },
784         error: function() {
785             // Show failure notification
786             Swal.fire({
787                 icon: 'error',
788                 title: 'Oops...',
789                 text: 'There was an error sending your message.'
790             });
791     }

```

```

1 router.post( path: '/mailbox', handlers: async (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj>, res : Response<ResBody, LocalsObj>) : Promise<void> => {
2     try {
3         await client.connect();
4         const db : Db = client.db(dbName);
5         const mailboxCollection : Collection<Document> = db.collection( name: 'mailbox' );
6
7         // Insert the new mail into the mailbox collection
8         const newMail : InsertOneResult<Document> = await mailboxCollection.insertOne( doc: {
9             email: req.body.email,
10            mailTitle: req.body.mailTitle,
11            mailContent: req.body.mailContent,
12            mailDate: req.body.mailDate,
13        },
14
15        // Send a success response to the client
16        res.json( body: { success: true, message: 'Mail sent successfully.' } );
17
18    } catch (err) {
19        console.error('Error:', err);
20        res.status( code: 500 ).json( body: { success: false, message: 'An error occurred.' } );
21    }
22 }

```

ii. Admin Mail Box για τα contact us mails

Mails HOME > DASHBOARD > MAILBOX

- Query Regarding "Blockchain & Cryptocurrency Symposium"** test1@test.com

Dear Event Organizers, I recently came across your event, "Blockchain & Cryptocurrency: The Future of Finance," and I'm highly intrigued by the concept. I'm particularly interested in the workshop titled "Leveraging Blockchain for Transparency in Transactions." Can you share more about the prerequisites and the knowledge level required for this workshop? Also, I would like to know if there will be any key figures or companies in the cryptocurrency industry speaking or presenting at the event. Thanks in advance for your response. Kind Regards,

6/24/2023, 10:06:51 PM
- Interested in "Artificial Intelligence in Healthcare Seminar"** test2@test.com

Hello, I'm writing to express my interest in your upcoming seminar on "Artificial Intelligence in Healthcare." This is a topic that greatly interests me, and I believe this seminar can provide valuable insights into the practical application of AI in medical practice. Can you please provide details on the main topics to be covered and any keynote speakers at the event? Moreover, I would like to inquire about any early bird discounts or group booking offers for the seminar. I look forward to hearing from you soon. Best regards,

6/24/2023, 10:07:35 PM
- Inquiring about "Emerging Trends in Cybersecurity Webinar"** test3@test.com

Dear Team, I've been following your organization's events for quite some time and am keen to participate in the upcoming "Emerging Trends in Cybersecurity" webinar. As I'm leading a team of security professionals, I believe this event could be highly beneficial. Could you please share more about the topics that will be covered, especially regarding cloud security and risk mitigation? Also, I would appreciate information about any networking opportunities during or after the event. Thank you in advance for your help. Sincerely,

6/24/2023, 10:07:49 PM
- Inquiry about "The Impact of AR & VR on Gaming Conference"** test4@test.com

Hi there, I'm reaching out regarding your conference on "The Impact of AR & VR on Gaming." I'm an avid gamer and game developer who is passionate about

6/24/2023, 10:08:09 PM
- Inquiry Regarding "Data Science & Machine Learning Summit"** test5@test.com

Dear Event Coordinators, I hope this message finds you well. I recently discovered your upcoming event, "Data Science & Machine Learning: Shaping the Future," and I

6/25/2023, 6:46:41 PM
- test** test@test.com

test

6/25/2023, 6:46:41 PM

Are you sure?

You won't be able to revert this!

No, cancel Yes, delete it!

The modal is covering the fifth email in the list.

6/24/2023, 10:06:51 PM 6/24/2023, 10:07:35 PM 6/24/2023, 10:07:49 PM

Mails HOME > DASHBOARD > MAILBOX

- Query Regarding "Blockchain & Cryptocurrency Symposium"** test1@test.com
- Interested in "Artificial Intelligence in Healthcare Seminar"** test2@test.com
- Inquiring about "Emerging Trends in Cybersecurity Webinar"** test3@test.com
- Inquiry about "The Impact of AR & VR on Gaming Conference"** test4@test.com
- Inquiry Regarding "Data Science & Machine Learning Summit"** test5@test.com

Από πλευράς κώδικα έχουμε server-side:

```
11   router.get( path: '/mailbox' , handlers: async (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res : Response<ResBody, LocalsObj> ) : Promise<void> => {
12     try {
13       await client.connect();
14       const db :Db = client.db(dbName);
15       const mailboxCollection :Collection<Document> = db.collection( name: 'mailbox' );
16
17       // Fetch all mails
18       const mails :WithId<...>[] = await mailboxCollection.find( filter: {} ).toArray();
19
20       // Send the mails data to the client
21       res.json(mails);
22
23     } catch (err) {
24       console.error('Error:', err);
25       res.status( code: 500).json( body: { error: 'An error occurred.' });
26     }
27   });

```

```
29   router.delete( path: '/mailbox/:mailId' , handlers: async (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res : Response<ResBody, LocalsObj> ) : Promise<void> => {
30     try {
31       await client.connect();
32       const db :Db = client.db(dbName);
33       const mailboxCollection :Collection<Document> = db.collection( name: 'mailbox' );
34
35       // Delete the specific mail
36       const result :DeleteResult = await mailboxCollection.deleteOne( filter: { _id: new ObjectId(req.params.mailId) });
37
38
39       // Check if any documents were deleted
40       if (result.deletedCount === 1) {
41         res.json( body: { success: 'Mail deleted successfully.' });
42       } else {
43         res.status( code: 404).json( body: { error: 'Mail not found.' });
44       }
45
46     } catch (err) {
47       console.error('Error:', err);
48       res.status( code: 500).json( body: { error: 'An error occurred.' });
49     }
50   });

```

Client-Side:

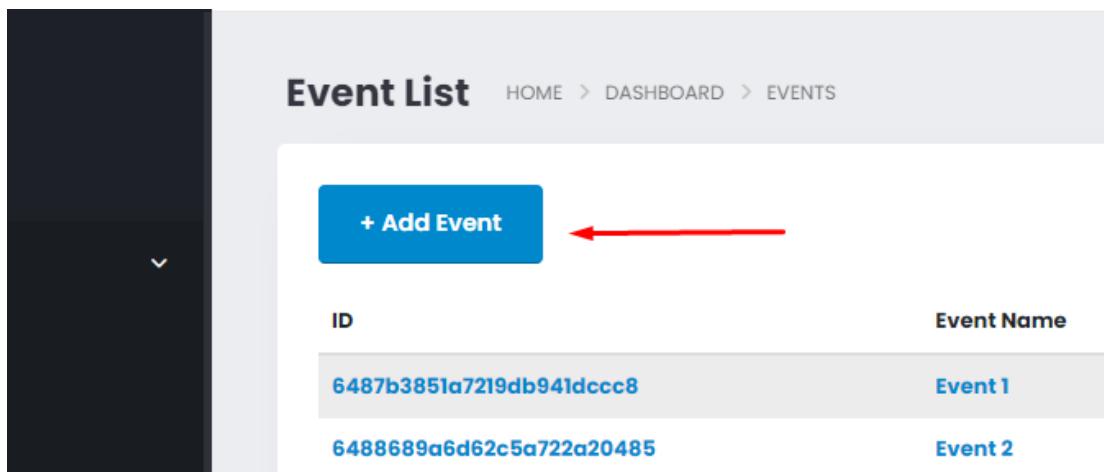
```
262 $(document).ready(function() {
263   $.ajax({
264     url: '/mailbox',
265     type: 'GET',
266     success: function(mails) {
267       // Loop over each mail
268       mails.forEach(mail => {
269         // Create a new card
270         var card = $`

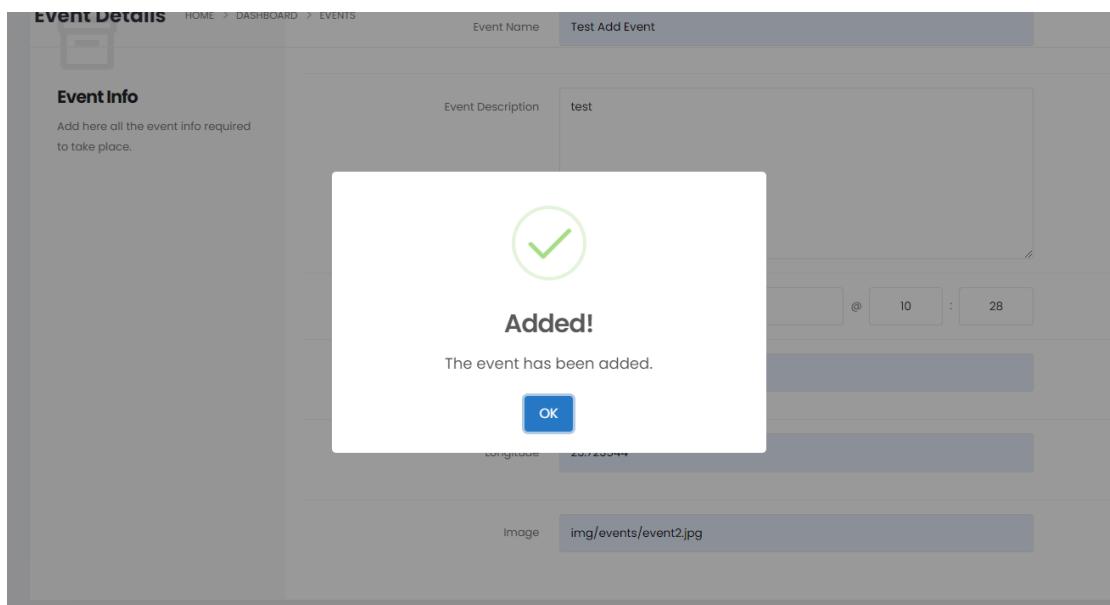
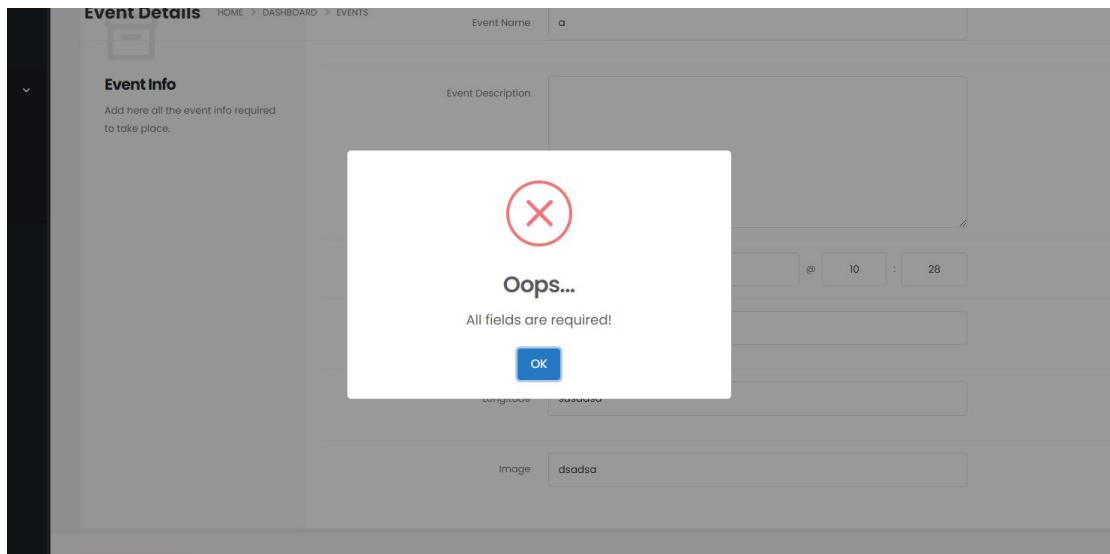
271           <div class="card card-warning mb-4">
272             <header class="card-header">
273               <div class="card-actions">
274                 <a href="#" class="card-action card-action-toggle" data-card-toggle></a>
275                 <a href="#" class="card-action card-action-dismiss deleteMail" data-id="${mail._id}"></a>
276               </div>
277             <h2 class="card-title text-color-success">${mail.mailTitle}</h2>
278             <p class="card-subtitle">${mail.email}</p>
279           </header>
280           <div class="card-body">
281             <div class="scrollable" data-plugin-scrollable style="...">
282               <div class="scrollable-content">
283                 <p>${mail.mailContent}</p>
284                 <p class="text-end text-muted">${new Date(mail.mailDate).toLocaleString()}</p>
285               </div>
286             </div>
287           </div>
288         </div>
289       `);
290     }
291   );
292   // Append the new card to the dynamicMails div
293 });


```

```
// Bind the delete event to the delete button of each card
$('.dynamicMails').on('click', '.deleteMail', function(e) {
    e.preventDefault();
    var mailId = $(this).data('id');
    Swal.fire({
        title: 'Are you sure?',
        text: "You won't be able to revert this!",
        icon: 'warning',
        showCancelButton: true,
        confirmButtonText: 'Yes, delete it!',
        cancelButtonText: 'No, cancel!',
        reverseButtons: true
    }).then((result) => {
        if (result.isConfirmed) {
            $.ajax({
                url: '/mailbox/' + mailId,
                type: 'DELETE',
                success: function() {
                    Swal.fire(
                        'Deleted!',
                        'Your mail has been deleted.',
                        'success'
                    );
                    // Remove the mail from the DOM
                    $('[data-id="' + mailId + '"]').parents('.col-lg-4').remove();
                },
                error: function(jqXHR, textStatus, errorThrown) {
                    console.log("AJAX Error: ", textStatus, errorThrown);
                    Swal.fire(
                        'Error!',
                        'There was an error deleting your mail.'
                    );
                }
            });
        }
    });
});
```

iii. Administrator Add New Event Page





iv. Experience Rating Feedback Box (page-event-complete.html)

Κατά την ολοκληρωση της κράτησης ζητείται από τον χρήστη να δώσει την κριτική του για την εμπειρία του στον ινστότοπο.

✓ Thank You. Your Order has been received.

Please Rate Your Experience!

How would you rate your experience with the loading times of the website?	How would you rate your experience with website friendliness?	How would you rate your overall experience with the events of the organization?
<div style="background-color: #009688; color: white; padding: 5px; text-align: center;">5</div>	<div style="background-color: #009688; color: white; padding: 5px; text-align: center;">5</div> <div style="background-color: #009688; color: white; padding: 5px; text-align: center;">5</div> <div style="background-color: #009688; color: white; padding: 5px; text-align: center;">4</div> <div style="background-color: #009688; color: white; padding: 5px; text-align: center;">3</div> <div style="background-color: #009688; color: white; padding: 5px; text-align: center;">2</div> <div style="background-color: #009688; color: white; padding: 5px; text-align: center;">1</div>	<div style="background-color: #009688; color: white; padding: 5px; text-align: center;">5</div>
Submit Form		

Αυτά τα στοιχεία δεν είναι εικονικά καθώς τοποθετούνται στην βάση δεδομένων στη συλλογή `experienceRating` την οποία μελλοντικά θα μπορούσε στο μέλλον να χρησιμοποιήσει για απόκτηση στοιχείων και λήψη αποφάσεων!

campaignDB.experienceRating

Documents Aggregations Schema Explain Plan Indexes Validation

Filter ⏳ ▾ Type a query: { field: 'value' }

+ ADD DATA ▾ ⏲ EXPORT DATA ▾

```
_id: ObjectId('64987eaab66db1eac1dc12ff')
websiteSpeed: "5"
websiteFriendliness: "5"
eventQuality: "5"
userId: ObjectId('648b02876a202bfffa959e491')
```