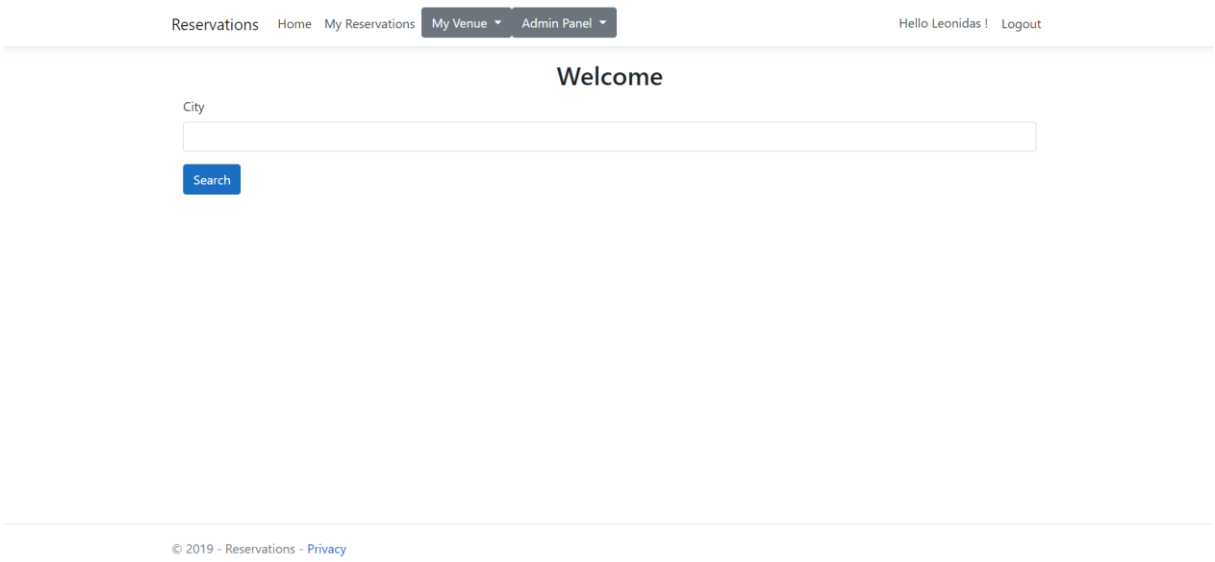
ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ
ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη Web-Based Συστήματος Διαχείρισης Κρατήσεων



Του φοιτητή
Αντωνιάδης Λεωνίδας
Αρ. Μητρώου: 144195

Επιβλέπων καθηγητής
Σαλαμπάσης Μιχαήλ

Θεσσαλονίκη 2020

Τίτλος Π.Ε. Ανάπτυξη Web-Based Συστήματος Διαχείρισης Κρατήσεων
Κωδικός Π.Ε. …
Ονοματεπώνυμο φοιτητή Αντωνιάδης Λεωνίδας.
Ονοματεπώνυμο εισηγητή …
Ημερομηνία ανάληψης Π.Ε. 5/11/2019
Ημερομηνία περάτωσης Π.Ε. …

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

ΠΕΡΙΛΗΨΗ

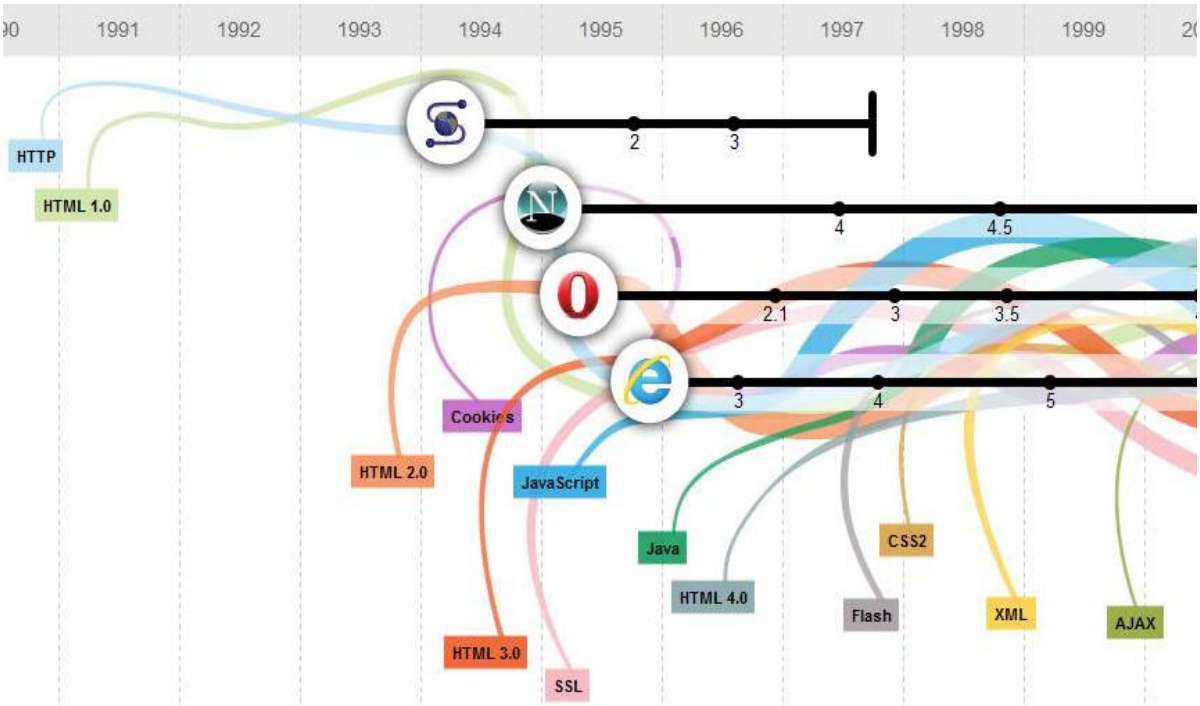
Η πιο βασική απαίτηση για κάθε μορφής website είναι να είναι αξιόπιστο αν πρόκειται για κάποιου είδους ηλεκτρονικό κατάστημα που πουλάει οτιδήποτε λογής αγαθά ή και όχι μόνο και να είναι όσο το δυνατόν πιο εύκολα προσβάσιμο και ευανάγνωστο προς το κοινό στο οποίο απευθύνεται. Είτε έχουμε δημιουργήσει μονοί μας κάποιο template είτε χρησιμοποιούμε κάποιο έτοιμο πρέπει να σιγουρευτούμε ότι ταιριάζει με το είδος της σελίδας που θέλουμε να δημιουργήσουμε. Το template είναι αυτό που θα καθορίσει μέσα σε λίγα δευτερόλεπτα αν ο χρήστης θέλει να συνεχίσει την πλοήγηση του μέσα στο website, πόσο μάλλον αν πρόκειται για κάποιο είδους μαγαζί να θελήσει να αγοράσει από αυτό. Τα γραφικά και η εμφάνιση είναι αυτά που βλέπουν οι καταναλωτές. Παραδόξως όσο σημαντική και να είναι η εμφάνιση μιας ιστοσελίδας για τους χρήστες όλη η ουσία βρίσκετε στο προγραμματιστικό κομμάτι για το πως δουλεύει και το πως γίνετε η αναπαράσταση όλον τον λειτουργιών. Συνεπώς ο στόχος αυτής της εργασίας είναι να δείξει την ανάπτυξη μια ιστοσελίδας κράτηση θέσεων σε οτιδήποτε τύπου καταστήματος και διαλέγοντας την θέση που επιθυμεί κάθε χρήστης. Επιπλέον η διαφορά σε αυτήν την σελίδα είναι ότι μπορείς να βρεις όλων των τύπων καταστήματος που έχει κράτηση και να κάνεις την κράτηση σου την ήμερα που έχει τον συγκεκριμένο event.

Περιεχόμενα

ΕΙΣΑΓΩΓΗ

Από την αρχή τις ανάπτυξης του διαδικτύου και τον ηλεκτρονικών σελίδων όπως ήταν το 1990 η πρώτη ιστοσελίδα στον world wide web που είχε αναπτύξει το CERN1 μέχρι και σήμερα ο τρόπος που γίνετε η αλληλεπίδραση έχει αναπτυχθεί σε τεράστιο βαθμό, κερδίζοντας όλο το ηλικιακό κοινό από μικρά παιδιά που μπορούν να ασχοληθούν με εκμάθηση και γνώση, μέχρι ότι άλλης λογής επικοινωνία μπορεί να υπάρξει.

Οι ιστοσελίδες πλέον έχουν φτάσει σε σημείο να είναι ένα αναπόσπαστο κομμάτι της καθημερινότητας κάθε χρήστη άσχετα με την ηλικία στην οποία ανήκει η της γραμματική μόρφωση που έχει. Ανάλογα σε τι κοινό απευθύνεται η κάθε μια ιστοσελίδα έχει και το ανάλογο κοινό. Αν τα πάρουμε τα πράγματα από την την αρχή από την γέννηση τον websites όλα βασιζόντουσαν επάνω στο http που αυτό που αναπαραστούσαν ήταν ένα απλό κείμενο που με την ανάλογη πλοήγηση μπορούσε να μεταφέρει τον χρήστη σε κάποιο άλλο ισότοπο με συγκεκριμένες για την εποχή τότε δυνατότητες. Χρειάστηκε βέβαια να περάσει λίγος καιρός μέχρι την διάδοση του www και έτσι αναπτύχθηκαν και άλλες γλώσσες προγραμματισμού που χρησιμοποιήθηκαν στην συνεχεία επάνω σε ένα τέτοιο πρωτόκολλο http όπως είναι η χρήση τον cookies, εκτέλεση JavaScript, χρήση ασφαλούς καναλιού επικοινωνίας με χρήση ssl, άλλα και άλλον τεχνολογιών όπως flash, css, ajax1.



Στις σημερινές ημέρες με την πρόοδο όλο και περισσότερων τεχνολογιών και γλωσσών προγραμματισμού η δημιουργία μιας σελίδας έχει γίνει αν δεν θέλουμε να χρησιμοποιήσουμε τον όρο εύκολη, τότε σίγουρα έγινε σε υπερβολικό βαθμό πιο δυναμική και αλληλεπιδραστική με τον χρήστη. Το WWW έχει φτάσει σήμερα να είναι για το κοινό τους ο τρόπος ψυχαγωγίας τους με (videogames, videos και μουσική και άλλα), ενημέρωσης τους από (ηλεκτρονικές εφημερίδες, άρθρα και εικόνες), εκμάθησης τους, η ακόμα και η ίδια τους η δουλειά σε συνεργασία με την επιχείρηση τους η ακόμα και δουλεύοντας κάπου ξεχωριστά από το σπίτι τους.

Ο κύριος σκοπός αυτής τις εργασίας είναι η περιγραφή μιας ηλεκτρονικής πλατφόρμας που μαζεύει όλους τους χώρους που μπορεί να επισκεφτεί ένα άτομο, ώστε να μπορέσει να τον διευκολύνει και να βρει αυτό που θέλει πιο γρήγορα και πιο αποτελεσματικά. Πιο συγκεκριμένα θα εξετάσουμε:

- Τη δομή και τους τρόπους ανάπτυξης μιας τέτοιας ιστοσελίδας.
- Την χρήση πολλαπλών γλωσσών προγραμματισμού και τεχνικών όπως είναι c#, asp.net, JavaScript (jquery), ajax, css, .net core 3, SQL, Entity Framework, Migrations, FabricJs, Identity Framework.
- Την λειτουργία ο χρήστης να διαλέγει την θέσει που επιθυμεί.
- Την λειτουργιά που ο ιδιοκτήτης σχεδίαση τον χωρώ του και τις θέσης όπως αυτός επιθυμεί.

ΚΕΦΑΛΑΙΟ 1

1.1 Ο ορισμός του προβλήματος

Το πρόβλημα που είδα είναι ότι δεν υπάρχει μια επιτυχής εφαρμογή η οποία μπορεί να εξυπηρέτηση ιδιοκτήτες και χρήστες για εύκολες γρήγορες κράτησης. Καθώς το διαδίκτυο έχει εισέλθει στις ζωές μας για να τις διευκολύνει και το χρησιμοποιήσουμε καθημερινά ακόμα και για τα πιο άπλα πράγματα (π.χ παραγγελιές φαγητού, προϊόντων, κράτησης εισιτήριον κ.α.) που θα μπορούσαν να γίνουν και με ένα τηλεφώνημα γίνονται μέσα από το διαδίκτυο. Για αυτό το λόγο αποφάσισα να δημιουργήσω αυτήν την εφαρμογή ώστε, οι ιδιοκτήτες να μπορούν να προωθήσουν τον χωρώ για μια κράτηση που μπορεί να είναι οτιδήποτε από μια συναυλία μέχρι και ένα εστιατόριο ή ένα μπαρ, και αυτός που αναζήτα να πάει σε ένα μέρος (π.χ ένας τουρίστας που δεν γνωρίζει ελληνικά)και να έχει την θέσει που έχει επιλέξει. Παροιμίες γνώστες εφαρμογές που υπάρχουν είναι το “e-table” από το “e-food”, και το “viva”.

1.2 Η διαφορά με άλλες εφαρμογές

Η σελίδα viva είναι μια παρόμοια σελίδα για κρατήσεις. Σε αυτήν την εφαρμογή ο χρήστης συνδέεται και μπορεί να κάνει κράτηση σε σινεμά, θέατρο, συναυλίες και μουσεία. Διαλέγεις ένα “event” που υπάρχει εκείνη την περίοδο ή θα γίνει σε μερικούς μήνες. Στη συνεχεία επιλεγείς την θέση σου (όχι σε όλες τις περιπτώσεις) και έχεις ανάλογη χρέωση ανάλογα την θέση καθώς μπορεί να είναι η θέση “VIP” αυτό συμβαίνει συνήθως όταν γίνεται κράτηση για θέατρο καθώς για αυτό τον λόγο χρησιμοποιούν πιο πολλοί την σελίδα. Αν κάνεις κράτηση για έναν σινεμά ή για κάποιο άλλο event μπορεί να μην έχεις τη δυνατότητα να επιλέξεις την θέση σου και άπλα μόνο να την πληρώσεις. Επομένως για να έχει κέρδος το viva παίρνει ποσοστά από της κράτησης που κάνει και κάποιοι που δεν θέλουν να χάνουν ποσοστά από τα κέρδη τους αυξάνουν την τιμή μέσα από το viva και έχουν της δίκες τους σελίδες πιο φτηνά και άπλα χρησιμοποιούν το viva για διαφήμιση. Τα θετικά αυτής της εφαρμογής είναι ότι έχει παρόμοια θεματολογία ώστε η χρήστες να μια μπερδεύονται. Τα αρνητικά της είναι ότι δεν έχει συνεχεία άλλα events διαλέγεις θέση σε άλλα όχι.

Μια άλλη παρόμοια εφαρμογή είναι το e-table από το e-food. Η συγκεκριμένη εφαρμογή απευθύνεται για κρατήσεις εστιατορίων. Ο χρήστης επιλεγεί την περιοχή που θέλει, τα άτομα και την μέρα και ανάλογα την μέρα έχει και διάφορες εκπτώσεις. Είναι μια απλή εφαρμογή κάνεις κράτηση χωρίς να επιλέξεις που θα καθίσεις και άπλα κερδίζεις μια έκπτωση 10%-15%. Τα θετικά αυτής της εφαρμογής βρίσκονται στην απλότητα της ο χρήστης ξέρει για τη βρίσκεται εκεί και θα πηγαίνει σε αυτόν τον χώρο για αυτόν το σκοπό. Τα αρνητικά είναι να μην μπορεί ο χρήστης να επιλέξει θέση και μπορεί να κάνει κράτηση και να τον βάλουν σε ένα σημείο μέσα στον χώρο που δεν είναι πολύ ευχάριστο.

Η διαφορά της δικής μου εφαρμογής από της άλλες είναι, ότι έχω μια μίξη τον δυο προηγούμενων εφαρμογών και πολλά περισσότερα καθώς μπορώ και εξυπηρετώ όλους το χώρους εστίασης. Άλλα έχω και πολλές διαφορετικές τεχνικές μέσα στην εφαρμογή που την κάνουν πιο εύκολη για χρήση, πρώτον ο ιδιοκτήτες έχει ένα δικό του περιβάλλων που μόνος μπορεί να δημιουργήσει το “event” που θέλει να κάνουν κράτησης και να δημιουργήσει και της θέσεις όπως επιθυμεί. Δεύτερων ο χρήστης έχει την δυνατότητα να κάνει κράτηση οποιαδήποτε στιγμή (εφόσον έχει δημιουργήσει τα event ο ιδιοκτήτης) μέσα από μια μεγάλοι ποικιλία χώρων ψυχαγωγίας, διασκέδασης και εστίασης. Τέλος παρατηρούμε ότι η διάφορα φαίνεται στην μεγάλη ποικιλία χώρων που μπορεί να κάνει κράτηση ο χρήστης. Τα θετικά της δίκης μου εφαρμογής είναι ότι μπορεί ο χρήστης να επιλέξει την θέση του, υπαρχή μεγάλη πληθώρα ιδιοκτήτων που μπορούν να προωθήσουν τον χώρο τους μέσα

από την εφαρμογή μου. Τα αρνητικά είναι ότι ίσος όλοι αυτή η πληροφορία από τους χώρους δεν θα μπορεί ο χρήστης να την επεξεργαστεί.

1.3 Γιατί web εφαρμογή

Για αυτή την εφαρμογή αποφάσισα να γίνει web και όχι desktop ή android. Ο λόγος είναι ότι στης μέρες μας ο καθημερινός κόσμος δεν χρησιμοποιήσει desktop εφαρμογές για να ανακαλύψει ψυχαγωγία τις χρησιμοποιήσει για εργασία. Επίσης μια desktop εφαρμογή για να διαφημιστή χρειάζεται μια ιστοσελίδα, άρα θα χρειάζονταν να κάνουμε δυο δουλείες για μια. Καλύτερη επιλογή θα ήταν μετά από την web εφαρμογή η android εφαρμογή. Παρόλο αυτά οι τεχνολογίες που χρησιμοποιώ επιτρέπουν στον κωδικά αυτό με λίγη έως και καθόλου τροποποίηση να γίνουν Android ή και desktop εφαρμογές με τη βοήθεια τον .net Core άλλα για αυτά θα μιλήσουμε σε επόμενα κεφαλαία.

ΚΕΦΑΛΑΙΟ 2

2.1 Ο Παγκόσμιος Ιστός (WORLD WIDE WEB)

Ο παγκόσμιος ιστός ή WORLD WIDE WEB είναι η υπηρεσία που έδωσε στο διαδίκτυο την σημερινή του αίγλη. Ο Παγκόσμιος ιστός είναι το δίκτυο των συνδεδεμένων υπολογιστών και δικτύων σε παγκόσμια κλίμακα,το οποίο χρησιμοποιεί συγκεκριμένη ομάδα πρωτοκόλλων επικοινωνίας. Το πρωτόκολλο επικοινωνίας αυτό ονομάζεται HTTP (Hypertext Transfer Protocol). Ο παγκόσμιος ιστός όπως είναι γνωστός στο ευρύ κοινό είναι ένα μοναδικό δίκτυο καθώς δεν υπάρχουν περισσότερα από ένα δίκτυα υπολογιστών παγκόσμιας κλίμακας.

2.2 Διαδίκτυο

Το Διαδίκτυο (Internet) είναι παγκόσμιο σύστημα διασυνδεδεμένων δικτύων υπολογιστών, οι οποίοι χρησιμοποιούν καθιερωμένη ομάδα πρωτοκόλλων, η οποία συχνά αποκαλείται "TCP/IP" (αν και αυτή δεν χρησιμοποιείται από όλες τις υπηρεσίες του Διαδικτύου) για να εξυπηρετεί εκατομμύρια χρηστών καθημερινά σε ολόκληρο τον κόσμο. Οι διασυνδεδεμένοι ηλεκτρονικοί υπολογιστές ανά τον κόσμο, οι οποίοι βρίσκονται σε ένα κοινό δίκτυο επικοινωνίας, ανταλλάσσουν μηνύματα (πακέτα) με τη χρήση διαφόρων πρωτοκόλλων (τυποποιημένοι κανόνες επικοινωνίας), τα οποία υλοποιούνται σε επίπεδο υλικού και λογισμικού. Το κοινό αυτό δίκτυο καλείται Διαδίκτυο.

2.3 Διαδικτυακά πρωτόκολλα και υπηρεσίες

Μερικά από τα πιο γνωστά διαδικτυακά πρωτόκολλα είναι τα TCP, IP, UDP, FTP, IMAP, POP3, SMTP, DNS, SMTP, HTTP, HTTPS, SSL, SSH ενώ μερικές από τις πιο γνωστές διαδικτυακές υπηρεσίες που κάνουν χρήση των πρωτοκόλλων αυτών είναι το ηλεκτρονικό ταχυδρομείο (e-mail), οι ομάδες συζητήσεων (newsgroups), η διαμοίραση αρχείων (file sharing), η μεταφορά αρχείων (file transfer), ο Παγκόσμιος Ιστός (World Wide Web) και η ροή μέσων σε πραγματικό χρόνο (streaming media) και η τηλεφωνία μέσω IP (voice telephony – VoIP). Από τις υπηρεσίες αυτές, το ηλεκτρονικό ταχυδρομείο και ο παγκόσμιος ιστός είναι οι πιο ευρέως χρησιμοποιημένες ενώ πολλές άλλες έχουν βασιστεί πάνω σε αυτές.

2.4 Υπηρεσίες Internet

Οι πιο δημοφιλείς και διαδεδομένες υπηρεσίες εφαρμογών του διαδικτύου είναι οι εξής:
• Παγκόσμιος Ιστός ή Ιστός (World Wide Web). Ο Ιστός δίνει τη δυνατότητα στους χρήστες να εμφανίζουν έγγραφα που περιέχουν κείμενο και γραφικά, καθώς και να ακολουθούν υπέρ- συνδέσμους για να κινηθούν μεταξύ των εγγράφων. Ο Ιστός αναπτύχθηκε σε τέτοιο βαθμό που έγινε η μεγαλύτερη πηγή διακίνησης πληροφοριών στο παγκόσμιο Internet, και εξακολουθεί να κατέχει κυρίαρχη θέση.

• Ηλεκτρονικό ταχυδρομείο (e-mail). Το ηλεκτρονικό ταχυδρομείο δίνει τη δυνατότητα στο χρήστη να συνθέσει ένα μήνυμα και να στείλει αντίγραφό του σε μεμονωμένα άτομα ή σε ομάδες ατόμων. Ένα άλλο τμήμα δίνει τη δυνατότητα στους χρήστες να διαβάσουν τα μηνύματα που έχουν λάβει
• Μεταφορά αρχείων (File transfer). Η εφαρμογή μεταφοράς αρχείων δίνει τη δυνατότητα στο χρήστη να στέλνει και να λαμβάνει αντίγραφα αρχείων δεδομένων. Η μεταφορά αρχείων είναι μια από τις πιο ευρέως χρησιμοποιού- μενες, υπηρεσίες εφαρμογών στο Internet. Παρά το ότι τα μικρά αρχείαμπορούν πλέον να επισυνάπτονται σε ένα μήνυμα ηλεκτρονικού ταχυδρομείου, η υπηρεσία μεταφοράς αρχείων εξακολουθεί να είναι απαραίτητη γιατί διαχείριση των μεγάλων αρχείων.

• Τηλεσύνδεση (Remote login). Η τηλεσύνδεση δίνει τη δυνατότητα σε ένα χρήστη να συνδεθεί με μια απομακρυσμένη μηχανή και να δημιουργήσει αλληλεπιδραστική ή περίοδο συνδεδεμένης λειτουργίας (login session). Μέσωτης τηλεσύνδεσης, η οθόνη του χρήστη συνδέεται κατευθείαν με την απομακρυσμένη μηχανή: κάθε πληκτρολόγηση του χρήστη μεταφέρεται στην απομακρυσμένη μηχανή και, αντιστοίχως, κάθε χαρακτήρας που στέλνει η απομακρυσμένη μηχανή εμφανίζεται στην οθόνη του χρήστη. Μόλις τερματίσει η περίοδος τηλεσύνδεσης, η εφαρμογή επαναφέρει το χρήστη στο τοπικό σύστημα.

2.5 Ιστοσελίδα (web page)

Η ιστοσελίδα είναι ένα αρχείο που περιέχει πληροφορίες που προορίζονται για δημοσίευση στον Παγκόσμιο Ιστό. Οι πληροφορίες του Παγκόσμιου Ιστού εμφανίζονται μορφοποιημένες με τη γλώσσα HTML (Hypertext Markup Language) σε μορφή ιστοσελίδων (web pages) και με την κατάληξη .htm ή .html. Υπάρχουν όμως και διαφορετικές μορφοποιήσεις ιστοσελίδων, όπως για παράδειγμα .aspx. Οι ιστοσελίδες μπορεί να περιέχουν εκτός από στατικό κείμενο, εικόνες, video, ήχο, κινούμενες εικόνες (animation), δυναμικό κείμενο κτλ. Ένα πλεονέκτημα τους είναι ότι αλληλοσυνδέονται μεταξύ τους και έτσι ο χρήστης μπορεί να μεταβεί με ένα κλικ από την μια ιστοσελίδα στην άλλη. Η κατασκευή ιστοσελίδων είναι κάτι που μπορεί να πραγματοποιηθεί εύκολα καθώς υπάρχουν αρκετά ελεύθερα προγράμματα καθώς και αυτοματοποιημένοι μηχανισμοί ιστοσελίδων που επιτρέπουν σε απλούς χρήστες εύκολα και γρήγορα να δημιουργήσουν την δική τους προσωπική ιστοσελίδα. Από την άλλη μεριά υπάρχουν και πολλές εταιρίες, που εξειδικεύονται στη δημιουργία ελκυστικών και λειτουργικών ιστοσελίδων που έχουν σαν στόχο να οδηγήσουν τους επισκέπτες στην αγορά κάποιου προϊόντος, στην επικοινωνία με τον ιδιοκτήτη του ιστοτόπου ή απλά στο ανέβασμα του εταιρικού προφίλ μιας επιχείρησης.

2.6 ΚΑΤΗΓΟΡΙΕΣ ΙΣΤΟΣΕΛΙΔΩΝ

Οι δύο βασικές κατηγορίες των ιστοσελίδων είναι οι δυναμικές (Dynamic) και οι στατικές (Static) ιστοσελίδες. Θα τις συναντήσουμε αναλυτικά πιο κάτω την κάθε μια ξεχωριστά και θα δούμε τις βασικές διαφορές τους.

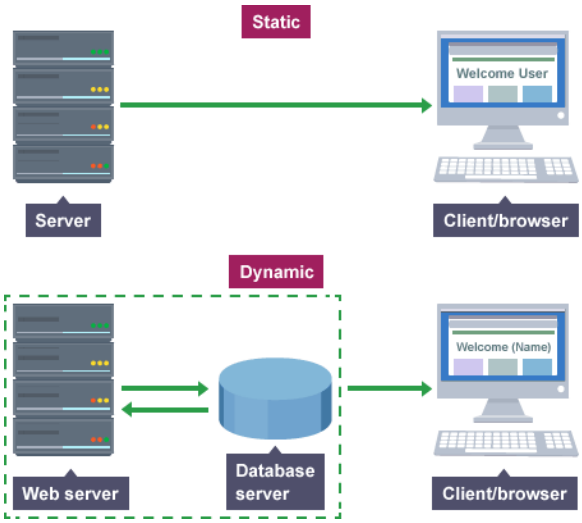
2.6.1 ΣΤΑΤΙΚΗ ΙΣΤΟΣΕΛΙΔΑ

Στατική ιστοσελίδα (Static Web Page) ονομάζεται μια ιστοσελίδα της οποίας το περιεχόμενο μεταφέρεται στον χρήστη ακριβώς στην μορφή που είναι αποθηκευμένο στον εξυπηρετητή ιστοσελίδων (Web Server). Με πιο απλά λόγια τα στατικά websites είναι αυτά που δεν ανανεώνονται συχνά και δεν προσφέρουν υπηρεσίες αλληλεπίδρασης με τους χρήστες. Οι στατικές ιστοσελίδες χρησιμοποιούν το πρωτόκολλο HTTP για να μεταφέρονται και αποθηκεύονται σε μορφή HTML. Βασικό πλεονέκτημα των στατικών ιστοσελίδων είναι ότι δεν χρειάζονται ιδιαίτερες γνώσεις και προγραμματιστικές δεξιότητες για να τις δημιουργήσεις καθώς και ότι η σελίδα μπορεί να διατίθεται στον φυλλομετρητή χωρίς να μεσολαβήσει κάποιος εξυπηρετητής ιστοσελίδων. Αυτό μπορεί να συμβεί με την χρήση ενός αποθηκευτικού μέσου όπως ένα CD-ROM ή USB. Επίσης μπορεί να κλωνοποιηθεί σε περισσότερους από έναν εξυπηρετητές. Στα αρνητικά των στατικών ιστοσελίδων παρουσιάζεται η άβολη χρήση τους απέναντι στον προγραμματιστή ή τον χρήστη καθώς και η δύσκολη διαχείρισή τους όταν ο αριθμός των ιστοσελίδων είναι μεγάλος. Σε αυτή την περίπτωση απαιτούνται αυτόματα εργαλεία διαχείρισης.

2.6.2 ΔΥΝΑΜΙΚΗ ΙΣΤΟΣΕΛΙΔΑ

Δυναμική ιστοσελίδα (Dynamic Web Page) είναι μια ιστοσελίδα η οποία δημιουργείται δυναμικά την στιγμή της πρόσβασης σε αυτή ή την στιγμή που ο χρήστης αλληλεπιδρά με τον εξυπηρετητή ιστοσελίδων. Με απλά λόγια, με τις δυναμικές ιστοσελίδες μπορούν συγκεκριμένοι χρήστες (διαχειριστές) να προχωρούν σε αλλαγές εύκολα και γρήγορα χρησιμοποιώντας έναν πίνακα διαχείρισης στον οποίο έχουν πρόσβαση μόνο αυτοί. Η έννοια ¨δυναμική¨ πηγάζει από την τεχνοτροπία με την οποία είναι κατασκευασμένη η ιστοσελίδα. Δυναμικά websites είναι τα ηλεκτρονικά καταστήματα, τα συστήματα διαχείρισης περιεχομένου (CMS), τα forums, τα blogs κ.ά. Βασικά πλεονεκτήματα της δυναμικής ιστοσελίδας είναι ότι έχουν την δυνατότητα άμεσης επέμβασης και τροποποίησης του περιεχομένου της από τον διαχειριστή της και επίσης ότι δεν υπάρχουν σχέσεις άμεσης εξάρτησης με κατασκευαστές και εταιρίες κατασκευής ιστοσελίδων. Δεν υπάρχουν περιορισμοί στον όγκο που μπορεί να αποκτήσει ένας ιστότοπος, η εξοικονόμηση πόρων και χρημάτων ακόμη είναι πολύ σημαντικός παράγοντας της χρήσης τους και τέλος υπάρχει διαδεδομένη τεχνογνωσία σε παγκόσμιο επίπεδο. Το βασικό τους μειονέκτημα είναι ότι υπάρχει μεγάλη εξάρτηση λειτουργίας της ιστοσελίδας από πλήθος ιδιοτήτων του διακομιστή στον οποίο πραγματοποιείται η φιλοξενία της ιστοσελίδας. Επίσης δυσκολότερη είναι και η αντιμετώπιση προβλημάτων καθώς και τεχνικών δυσκολιών.

2.6.3 ΔΙΑΦΟΡΕΣ ΔΥΝΑΜΙΚΩΝ – ΣΤΑΤΙΚΩΝ ΙΣΤΟΣΕΛΙΔΩΝ



Βασική διαφορά των δυναμικών ιστοσελίδων έναντι των στατικών είναι ότι στις δυναμικές είναι απεριόριστο το περιεχόμενο των πληροφοριών που μπορεί κάποιος να αποθηκεύσει ενώ στις στατικές το περιεχόμενο είναι πολύ περιορισμένο. Επίσης όπως αναφέρθηκε πιο πάνω οι στατικές ιστοσελίδες παρουσιάζουν ελάχιστη ακόμα και καμία αλληλεπίδραση με τους χρήστες ενώ στις δυναμικές ιστοσελίδες η αλληλεπίδραση με τους χρήστες είναι δεδομένη. Σε αντίθεση με τις στατικές οι οποίες είναι αρχεία ¨ανεβασμένα¨ σε έναν διακομιστή (server) οι δυναμικές ιστοσελίδες είναι ολόκληρες εφαρμογές που χρειάζονται μια βάση δεδομένων για να λειτουργήσουν. Πλεονέκτημα των δυναμικών ιστοσελίδων έναντι των στατικών είναι σε ότι αφορά το κόστος για την αλλαγή περιεχομένου καθώς στις πρώτες το κόστος είναι μηδαμινό ενώ των στατικών είναι αρκετά υψηλό. Σε ότι αφορά το κόστος κατασκευής όμως,τα πράγματα εδώ αντιστρέφονται, με τις στατικές ιστοσελίδες να υπερτερούν έναντι των δυναμικών αφού το αρχικό κόστος κατασκευής είναι πολύ μικρότερο. Τέλος όσο αφορά την εξέλιξη οι στατικές ιστοσελίδες δεν εξελίσσονται, ενώ οι δυναμικές εξελίσσονται μέρα με την μέρα.

2.7 Διακομιστής Ιστού (web server)

Κάθε ιστοσελίδα βρίσκεται με τη μορφή αρχείου σε κάποιον διακομιστή Ιστού (web Server). Οι διακομιστές Ιστού είναι ειδικοί υπολογιστές με ειδικό λογισμικό και κατάλληλες δικτυακές συνδέσεις, οι οποίοι επιτρέπουν τη διάθεση των ιστοσελίδων σε ολόκληρο τον κόσμο. Ο Web Server είναι, στην ουσία ο υπολογιστής εκείνος ο οποίος αναλαμβάνει να δημοσιεύει την ιστοσελίδα στο Διαδίκτυο. Ο χρήστης του Διαδικτύου που θέλει να δει μια ιστοσελίδα, τη ζητάει από τον διακομιστή Ιστού στον οποίο αυτή βρίσκεται, και ο διακομιστής Ιστού με τη σειρά του την στέλνει. Ο server θα

πρέπει να είναι σε συνεχή διαθεσιμότητα, ώστε να ανταποκρίνεται στις κλήσεις και να παρέχει τις ιστοσελίδες.

2.8 ΦΥΛΛΟΜΕΤΡΗΤΗΣ (WEB BROWSER)

Ένας φυλλομετρητής ιστοσελίδων ή περιηγητής ιστού είναι ένα λογισμικό που επικοινωνεί με τους διακομιστές ιστού (web servers) μέσω του πρωτοκόλλου HTTP. Αλληλεπιδρά με κείμενα, εικόνες, βίντεο και άλλες πληροφορίες που βρίσκονται συνήθως αναρτημένες σε μια ιστοσελίδα ενός ιστότοπου στον παγκόσμιο ιστό ή σε ένα τοπικό δίκτυο. Ο Web browser επιτρέπει στον χρήστη την γρήγορη και εύκολη πρόσβαση σε διάφορες ιστοσελίδες και ιστότοπους εναλλάσσοντας τις ιστοσελίδες μέσω των υπερσυνδέσμων. Τα κείμενα, οι εικόνες, τα βίντεο μπορεί να περιέχουν υπερσυνδέσμους προς άλλες ιστοσελίδες. Οι φυλλομετρητές χρησιμοποιούν τη γλώσσα μορφοποίησης HTML για την προβολή των ιστοσελίδων.
Κάποιοι από τους δημοφιλέστερους Web browsers είναι οι:

- Windows Internet Explorer
- Mozilla Firefox
- Opera
- Apple Safari
- Google Chrome

Για κάθε φυλλομετρητή υπάρχουν αρκετά πρόσθετα στοιχεία (add-ons, plug-ins) που βοηθούν στην αύξηση των δυνατοτήτων τους, την βελτίωση όσο αφορά την χρηστικότητα τους καθώς και την προστασία των πελατών σε θέματα ασφαλείας.

2.9 DomainName

Όνομα χώρου ή τομέα ή περιοχής (domain name) στο Διαδίκτυο είναι ένας περιορισμένος τομέας των διεθνών πόρων του Συστήματος Ονομάτων Χώρου (DNS) ο οποίος εκχωρείται για αποκλειστική χρήση σε ένα φυσικό ή νομικό πρόσωπο. Το όνομα τομέα / χώρου δεν ανήκει στο πρόσωπο που του έχει εκχωρηθεί αλλά έχει μόνο την αποκλειστική δυνατότητα χρήσης του για όσο διάστημα έχει καταβάλει τα τέλη -12- κατοχύρωσης. Ένα όνομα χώρου μπορεί να έχει διάφορες καταλήξεις όπως .com, .eu, .gr, .net, .org, .info, .biz, .de, .it, .es κ.λ.π., ανάλογα με τη χρήση και τη χώρα προέλευσής του.
Στα ονόματα χώρου επιτρέπεται μόνο η χρήση αλφαριθμητικών στοιχείων και παυλών. Για τα ονόματα χώρου με κατάληξη .gr υπάρχουν απαγορευμένες κατηγορίες. Αν ένα όνομα χώρου θεωρείται κοινόχρηστο ή γεωγραφικός όρος εκχωρείται μόνο στους αντίστοιχους οργανισμούς τοπικής αυτοδιοίκησης ανεξάρτητα από τον τρόπο γραφής του με λατινικά στοιχεία. Επίσης δεν επιτρέπεται η εκχώρηση ονομάτων χώρου με κατάληξη .gr που αποτελούν λέξεις κλειδιά στο Διαδίκτυο.
Τα κατοχυρωμένα ονόματα χώρου είναι συνήθως τα ονόματα των τριών ή τεσσάρων πρώτων επιπέδων. Τα υπόλοιπα ονόματα χώρου δεν χρειάζονται κατοχύρωση. Στα ονόματα χώρου κάθε τελεία δείχνει την αλλαγή επίπεδου ή αρχή ενός υποσυνόλου - υποτομέα και το σύνολο - χώρος που περιλαμβάνει όλα τα σύνολα είναι η πιο δεξιά τελεία που συνήθως παραλείπεται. Οι λύτες είναι το λογισμικό που μας βοήθα να χρησιμοποιήσουμε τα ονόματα χώρου. Οι λύτες διαβάζουν τα ονόματα του DNS από δεξιά προς τα αριστερά.

2.10 ΦΙΛΟΞΕΝΙΑ ΙΣΤΟΣΕΛΙΔΩΝ (WEB HOSTING)

Όταν μια ιστοσελίδα θα πρέπει να βρίσκεται συνεχώς αναρτημένη στο διαδίκτυο θα πρέπει πρώτα να αποθηκευτεί σε ένα Web server για να χρησιμοποιηθεί. Για αυτό υπάρχουν οι διαδικτυακές υπηρεσίες που επιτρέπουν σε εταιρίες και ιδιώτες να διαθέτουν μια ιστοσελίδα αναρτημένη στο διαδίκτυο συνεχώς και με την απαιτούμενη ασφάλεια. Γιατί θα ήταν αδύνατο ο κάθε χρήστης ξεχωριστά να διαθέτει τον απαιτούμενο εξοπλισμό. Η φιλοξενία ιστοσελίδων λοιπόν είναι αύτη η διαδικτυακή υπηρεσία που επιτρέπει όλο αυτό χωρίς επιβαρύνσεις με το κόστος του ανάλογου εξοπλισμού. Με πιο απλά λόγια μπορούμε να πούμε πως ο όρος Web Hosting είναι η ενοικίαση του χώρου στον ιδιοκτήτη μιας ιστοσελίδας σε υπολογιστές (διακομιστές) για να τοποθετήσει τα αρχεία του. Κάθε εταιρία ή ιδιώτης που παρέχουν φιλοξενία σε ιδιοκτήτες ιστοσελίδων διαθέτει τα δικά της πακέτα hosting με τιμές,ταχύτητες,διάρκεια και χαρακτηριστικά ανάλογα με τα θέλω του πελάτη. Οπότε σύμφωνα με την χρήση ή την κίνηση και την δημοφιλία της ιστοσελίδας του κάθε ενδιαφερόμενου επιλέγει το πακέτο που ταιριάζει καλύτερα σε αυτόν. Τα πακέτα φιλοξενίας όπως κατηγοριοποιούνται στις παρακάτω οικογένειες:

• **Shared Hosting**, όπου παρέχεται μέρος του διακομιστή στον οποίο φιλοξενούνται και άλλοι χρήστες.

• **Reseller Hosting**, όπου υπάρχει η δυνατότητα της μεταπώλησης εργαλείων φιλοξενίας ιστοσελίδων και χώρου.

• **Cloud Hosting**, όπου τα εισερχόμενα αιτήματα εξυπηρέτησης διαμοιράζονται σε μηχανήματα που έχουν το μικρότερο φόρτο εργασίας με την χρήση της τεχνολογίας διαμοιρασμού φόρτου εργασίας σε πολλούς διακομιστές ταυτόχρονα, ενώ παράλληλα τα αντίγραφα σε κάθε server εξασφαλίζουν την διαθεσιμότητα και την ακεραιότητα των αρχείων.

• **Dedicated Servers**, όπου παρέχεται ολόκληρος ο διακομιστής για αποκλειστική χρήση και διαχείριση από τον κάτοχο του ιστότοπου.

• **Virtual Private Server**, όπου μέσω λογισμικού εικονικοποίησης διακομιστή (virtualization) παρέχεται στον διακομιστή ένας απομονωμένος χώρος με δικούς του –αποκλειστικής χρήσης- πόρους συστήματος και κεντρική πρόσβαση.

2.11 ΕΞΥΠΗΡΕΤΗΤΗΣ ΙΣΤΟΥ (WEB SERVER)

Το λογισμικό που τρέχει σε έναν κόμβο του διαδικτύου ή στον υπολογιστή όταν εκτελεί προγράμματα εξυπηρετητές συνεχόμενα και για πολλές ώρες, ονομάζεται διακομιστής η εξυπηρετητής. Οι εξυπηρετήσεις αυτές αφορούν άλλα προγράμματα που ονομάζονται πελάτες (clients).Ο πιο δημοφιλής web server είναι ο Apache που θα τον δούμε αναλυτικά στην συνέχεια. Οι εξυπηρετητές-προγράμματα που συναντούμε συχνότερα σε περιβάλλον γραφείου είναι οι εξής:

• **Εξυπηρετητής αρχείων** (file server)

• **Εξυπηρετητής αντιγράφων ασφαλείας** (backup server)

• **Εξυπηρετητής βάσεων δεδομένων** (database server)

- **Εξυπηρετητής διαμεσολαβητή** (proxy server)
- **Εξυπηρετητής ηλεκτρονικού ταχυδρομείου** (mail server)
- **Εξυπηρετητής φαξ** (fax server)
- **Εξυπηρετητής εκτυπωτών** (printer server)

Οι εξυπηρετητές-προγράμματα που συναντούμε συχνότερα στο διαδίκτυο είναι οι εξής:

- **Εξυπηρετητής παγκοσμίου ιστού με το πρωτόκολλο http** (http server)
- **Εξυπηρετητής ηλεκτρονικού ταχυδρομείου** (mail server)
- **Domain Name System** (DNS server)
- **Εξυπηρετητής μεταφοράς αρχείων με το πρωτόκολλο ftp** (ftp server)

ΚΕΦΑΛΑΙΟ 3

3.1 Τι είναι HTML

Η Hyper Text Markup Language (Γλώσσα Σήμανσης Υπερκειμένου) είναι η κύρια γλώσσα σήμανσης για τις ιστοσελίδες και τα στοιχεία της είναι τα βασικά δομικά στοιχεία των ιστοσελίδων. Εμφανίστηκε το 1999. Γράφεται με την μορφή HTML στοιχείων τα οποία αποτελούνται από ετικέτες που ονομάζονται tags, οι οποίες 26 περικλείονται μέσα σε σύμβολα τύπου , μέσα στο περιεχόμενο της ιστοσελίδας. . Η HTML γλώσσα διαθέτει ένα πεπερασμένο αριθμό ετικετών που μπορούμε να χρησιμοποιήσουμε. Αυτός ο αριθμός όμως δεν παραμένει σταθερός. Η HTML γλώσσα μπορεί να χρησιμοποιηθεί και στις δυναμικές αλλά και στις στατικές ιστοσελίδες.

3.2 Τι είναι HTML 5

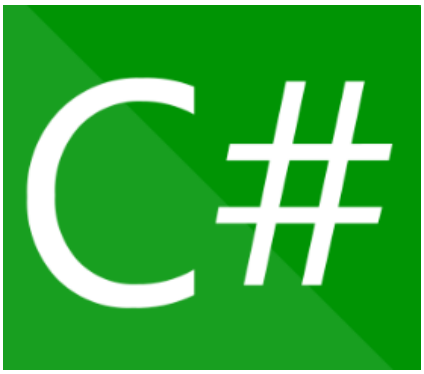
Η HTML 5 είναι η νέα έκδοση της γλώσσας προγραμματισμού HTML. Το W3C (World Wide Web Consortium) και το WHATWG (Web Hypertext Application Technology Working Group) συνεργάστηκαν και δημιούργησαν αυτή την νέα γλώσσα. Είναι ακόμα υπό ανάπτυξη ακόμη και όταν ετοιμαστεί πλήρως θα είναι η επόμενη μεγάλη έκδοση της HTML. Προορίζεται για αντικαταστάτης της HTML 4.0.1, της XHTML 1.0 και της DOM 2 HTML. Επίσης προορίζεται για την καλύτερη λειτουργία εντοπισμού λαθών, την πλήρη συμβατότητα ανεξαρτήτως συσκευής και την αντικατάσταση του scripting. Επιπλέον η μείωση της ανάγκης για ιδιόκτητα plug-in και άλλες διαδικτυακές εφαρμογές (RIA) είναι άλλος ένας βασικός κανόνας που έχει οριστεί για την HTML 5. Βάση για τα νέα χαρακτηριστικά είναι οι HTML, CSS, DOM και JavaScript. Περιέχει το πρότυπο Web Forms 2.0.
Κάποια από τα νέα χαρακτηριστικά της είναι:

- Το στοιχείο canvas για το drawing
- Τα νέα στοιχεία περιεχομένου όπως τα footer, header, section και nav
- Τα στοιχεία audio και video για την αναπαραγωγή των πολυμέσων
- Τα νέα στοιχεία δημιουργίας φόρμας όπως τα calendar, time, date, email, url και search.

3.3 Τι είναι SQL

Η SQL(αγγλ.αρκτ.από τοStructured Query Language) είναι μία γλώσσα υπολογιστών στις βάσεις δεδομένων, που σχεδιάστηκε για τη διαχείριση δεδομένων, σε ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων (Relational Database Management System, RDBMS) και η οποία, αρχικά, βασίστηκε στη σχεσιακή άλγεβρα. Η γλώσσα περιλαμβάνει δυνατότητες ανάκτησης και ενημέρωσης δεδομένων, δημιουργίας και τροποποίησης σχημάτων και σχεσιακών πινάκων, αλλά και ελέγχου πρόσβασης στα δεδομένα. Η SQL4 ήταν μία από τις πρώτες γλώσσες για το σχεσιακό μοντέλο του Edgar F. Codd, στο σημαντικό άρθρο του το 1970, και έγινε η πιο ευρέως χρησιμοποιούμενη γλώσσα για τις σχεσιακές βάσεις δεδομένων.

3.4 Τι είναι C#



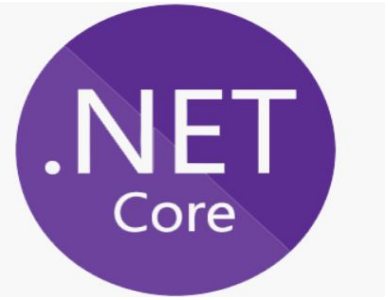
Η C# (C Sharp, ελληνική προφ. Σι Σαρπ) είναι μια γλώσσα προγραμματισμού Η/Υ. Δημιουργήθηκε από την Microsoft μέσα από την πλατφόρμα .NET και αργότερα αναγνωρίστηκε επισήμως από την Ecma (ECMA-334) και την ISO (ISO/IEC 23270:2018). Είναι μια από τις γλώσσες προγραμματισμού που δημιουργήθηκαν για την Κοινή Υποδομή Γλώσσας (αγγλικά: Common Language Infrastructure). Ο κύριος σκοπός της γλώσσας είναι να είναι απλή αντικειμενοστρεφής γλώσσα για γενική χρήση. Ο διοικητής της ομάδα που διαχειρίζεται την γλώσσα ονομάζεται Άντερς Χάιλσμπεργκ. Στις 15 Αυγούστου 2012 κυκλοφόρησε η έκδοση 5.0 η οποία είναι η πιο πρόσφατη μέχρι σήμερα.

3.5 Τι είναι ASP.NET

Το Asp.net είναι προγραμματιστικό περιβάλλον της εταιρείας Microsoft που δημιουργήθηκε για διαδικτυακό προγραμματισμό για την δημιουργία δυναμικών ιστοσελιδών στο διαδίκτυο. Αναπτύχθηκε από την Microsoft για να δώσει την δυνατότητα σε προγραμματιστές να δημιουργήσουν ιστοσελίδες, διαδικτυακές εφαρμογές και διαδικτυακές υπηρεσίες.
Δημιουργήθηκε τον Ιανουάριο του 2002 με την έκδοση 1.0 και είναι ο διάδοχος της τεχνολογίας Active Server Pages (ASP). Το ASP.NET δημιουργήθηκε με την (CLR) δίνοντας την ικανότητα στους προγραμματιστές να γράψουν την ASP.NET με οποιαδήποτε υποστηριζόμενη .NET γλώσσα. Η επέκταση ASP.NET SOAP επιτρέπει τους χρήστες της γλώσσας να κάνουν μηνύματα στην SOAP γλώσσα. Το 2014 η Microsoft άρχισε να διαθέτει το περιβάλλον .NET ως ανοικτό-κώδικα υπό την άδεια Apache 2.0. Το ASP.NET δεν χρησιμοποιείται στην εφαρμογή άπλα είναι ο “πατέρας” του .NET Core.

3.6 Τι είναι .NET Core

Το .NET Core είναι μια πλατφόρμα ανάπτυξης ανοιχτού κώδικα γενικού σκοπού. Μπορείτε να δημιουργήσετε εφαρμογές .NET Core για Windows, macOS και Linux για επεξεργαστές x64, x86, ARM32 και ARM64 χρησιμοποιώντας πολλές γλώσσες προγραμματισμού. Τα πλαίσια και τα API παρέχονται για cloud, IoT, UI πελάτη και μηχανική εκμάθηση.



Πρόκειται για ένα Framework που λειτουργεί τόσο στο πλήρες .NET Framework, στα Windows όσο και στο .NET Core πολλαπλών πλατφορμών. Ωστόσο, η έκδοση 3 του ASP.NET Core λειτουργεί μόνο σε .NET Core και σταμάτα να προσφέρει υποστήριξη για το .NET Framework. Το Framework είναι μια πλήρης επανεγγραφή που ενώνει το προηγούμενο ξεχωριστό ASP.NET MVC και ASP.NET Web API σε ένα μονό μοντέλο προγραμματισμού.

Παρόλο που είναι ένα νέο Framework, χτισμένο σε μια νέα στοίβα ιστού, έχει υψηλό βαθμό συμβατότητας με το ASP.NET. Οι εφαρμογές ASP.NET Core υποστηρίζουν side by side versioning , όπου διαφορετικές εφαρμογές, που εκτελούνται στον ίδιο υπολογιστή, μπορούν να στοχεύουν

διαφορετικές εκδόσεις του ASP.NET Core. Αυτό δεν είναι δυνατό με προηγούμενες εκδόσεις του ASP.NET.

3.7 Τι είναι το Identity Framework

ASP.NET Core Identity είναι ένα API που βοηθάει το user interface για την σύνδεση του χρήστη. Διαχειρίζεται τους χρήστες, κωδικούς, τα δεδομένα του προφίλ, ρόλους, τα tokens, email επιβεβαιώσει και πολλά άλλα. Οι χρήστες μπορούν να δημιουργήσουν λογαριασμούς και τα δεόμενα τους θα αποθηκευτούν στο identity ή σε έναν εξωτερικό login provider. Υποστηρίζεται από εξωτερικά logins providers όπως Facebook, Google, Microsoft Account και Twitter.

3.8 Τι είναι το Entity Framework

Το Entity Framework είναι μια ομάδα από τεχνολογίες στο ADO.NET που υποστηρίζει την αναπτύξει των δεδομένων(data-oriented) σε εφαρμογές. ΟΙ αρχιτέκτονες και η προγραμματιστές των data-oriented εφαρμογών έχουν δυσκολευτεί με την ανάγκη να πετύχουν δυο διαφορετικούς στόχους. Πρέπει να μοντελοποιήσουν της έννοιες(entities), τις σχέσεις(relationships), και τη λογική από τα "business" προβλήματα που λύνουν, και ακόμα πρέπει να δουλέψουν με data engines που χρησιμοποιούνται για να πάρουν δεδομένα. Τα δεδομένα μπορούν να έχουν πολλαπλά συστήματα αποθήκευσης, το κάθε ένα με το δικό του πρωτόκολλο, ακόμα εφαρμογές που λειτουργούν με ένα μέσο αποθήκευσης πρέπει να ισορροπήσουν τα προαπαιτούμενα του ανυποθήκευτου συστήματος ενάντιας τις απαιτήσεις του κάλου και διατηρητέου κωδικά μια εφαρμογής.

Το Entity Framework προσφέρει στου προγραμματιστές να δουλεύουν με τα δεδομένα στη μορφή του domain-specific objects και ιδιότητες, όπως σαν πελάτες και διεύθυνση πελάτη, χωρίς να χρειάζεται να ανησυχεί για του πινάκες τις βάσης δεδομένων και τις στήλες και το που αποθηκεύονται τα δεδομένα. Με το Entity Framework, οι προγραμματιστές μπορούν να δουλεύουν σε πιο υψηλό abstract επίπεδο όταν δουλεύουν με δεδομένα. Επειδή το Entity Framework είναι ένα στοιχείο του .NET Framework και του .NET Core, το Entity Framework μπορεί να λειτουργήσει σε όλους του υπολογιστές που έχουν το .NET Framework ή .NET Core εγκαταστημένο.

3.8 Τι είναι CSS

Η CSS (Cascading Style Sheets) είναι μια γλώσσα υπολογιστή που ανήκει στην κατηγορία των γλωσσών φύλλων στυλ που χρησιμοποιείται για τον έλεγχο της εμφάνισης ενός εγγράφου που έχει γραφτεί με μια γλώσσα σήμανσης. Χρησιμοποιείται δηλαδή για τον έλεγχο της εμφάνισης ενός εγγράφου που γράφτηκε στις γλώσσες HTML και XHTML, δηλαδή για τον έλεγχο της εμφάνισης μιας ιστοσελίδας και γενικότερα ενός ιστοτόπου. Η CSS είναι μια γλώσσα υπολογιστή προορισμένη να αναπτύσσει στυλιστικά μια ιστοσελίδα δηλαδή να διαμορφώνει περισσότερα χαρακτηριστικά, χρώμματα, στοίχιση και δίνει περισσότερες δυνατότητες σε σχέση με την html. Για μια όμορφη και καλοσχεδιασμένη ιστοσελίδα η χρήση της CSS κρίνεται ως απαραίτητη.

3.9 Τι είναι το Bootstrap

Το Bootstrap είναι ένα open-source CSS framework για responsive σελίδες, προορίζεται για front-end web development. Περιέχει CSS και javascript για τυπογραφία, φόρμες, κουμπιά, και άλλα εργαλεία για ένα interface. Ο κύριος λόγος για να προσθέσει στο project είναι για να χρησιμοποιήσουμε τα χρώματα του Bootstrap και τα layouts που προσφέρει. Επιπλέον οι προγραμματιστές μπορούν να πάρουν προς όφελος τους τις CSS κλάσεις και να επεκτείνουν το Bootstrap. Επιπροσθέτως το visual studio στο .NET CORE MVC που χρησιμοποιήθηκε στην εφαρμογή υπάρχει είδη εγκαταστημένο.

3.10 Τι είναι Ajax (Asynchronous JavaScript and XML)

Η τεχνολογία AJAX (Asynchronous Javascript and XML) αυτή τη στιγμή αποτελεί τη πιο σύγχρονη τεχνολογία στον προγραμματισμό στο internet, δίνοντας διαδραστικές δυνατότητες σε ένα δυναμικό site, μετατρέποντας το από ένα απλό site σε μια διαδικτυακή εφαρμογή. Ένας από τους κύριους εκφραστές αυτής της τεχνολογίας είναι και η jQuery.

Σίγουρα θα έχετε παρατηρήσει τα τελευταία χρόνια σε διάφορα sites την εντυπωσιακή εμφάνιση κειμένων, τα πρωτότυπα scrolling, τα δυναμικά ξεφυλλίσματα σε photogalleries. Αυτές οι υλοποιήσεις χρησιμοποιούν την τεχνολογία AJAX και κύριο χαρακτηριστικό τους δεν είναι οι εντυπωσιακές κινήσεις που κάνουν αλλά η αλλαγή τους χωρίς την επαναφόρτωση του site. Η AJAX δίνει τη δυνατότητα εμφάνισης νέων στοιχείων στο site, χωρίς τη φόρτωση νέας σελίδα. Μπορεί δηλαδή ο web developer να δημιουργήσει ένα site με μία μόνο σελίδα, στην οποία θα φορτώνονται διαφορετικά δεδομένα ανάλογα με τις επιλογές του χρήστη. Έτσι καταργεί τους ατελείωτους φακέλους με τα html αρχεία, στα οποία επαναλαμβάνεται το ίδιο κομμάτι κώδικα, βελτιώνοντας παράλληλα και την ασφάλεια του site καθώς καταργεί την αλλαγή του url στη μπάρα διευθύνσεων.

Όπως δηλώνει και στο όνομα της , η κύρια γλώσσα με την οποία εφαρμόζεται η AJAX είναι η JavaScript. Συνεπώς κάποιος που χρησιμοποιεί τη JavaScript μπορεί να την χρησιμοποιήσει για να εφαρμόσει τη τεχνολογία AJAX. Πέρα όμως από την JavaScript, τα τελευταία χρόνια έχουν κάνει την εμφάνισή τους και κάποιες πρόσθετες βιβλιοθήκες οι οποίες δίνουν τη δυνατότητα στον προγραμματιστή να γράψει κώδικα σε JavaScript με ποιο εύκολο, σύντομο και κατανοητό τρόπο. Τέτοιες βιβλιοθήκες είναι οι: jQuery, Dojo, YUI, MooTool, Prototype.

3.11 Τι είναι JavaScript

Η JavaScript (JS) είναι διερμηνευμένη γλώσσα προγραμματισμού για ηλεκτρονικούς υπολογιστές. Αρχικά αποτέλεσε μέρος της υλοποίησης των φυλλομετρητών Ιστού, ώστε τα σενάρια από την πλευρά του πελάτη (client-side scripts) να μπορούν να επικοινωνούν με τον χρήστη, να ανταλλάσσουν δεδομένα ασύγχρονα και να αλλάζουν δυναμικά το περιεχόμενο του εγγράφου που εμφανίζεται.

Η JavaScript είναι μια γλώσσα σεναρίων που βασίζεται στα πρωτότυπα (prototypebased), είναι δυναμική, με ασθενείς τύπους και έχει συναρτήσεις ως αντικείμενα πρώτης τάξης. Η σύνταξή της είναι επηρεασμένη από τη C. Η JavaScript αντιγράφει πολλά ονόματα και συμβάσεις ονοματοδοσίας από τη Java, αλλά γενικά οι δύο αυτές γλώσσες δε σχετίζονται και έχουν πολύ διαφορετική σημασιολογία. Οι βασικές αρχές σχεδιασμού της JavaScript προέρχονται από τις γλώσσες προγραμματισμού Self και Scheme. Είναι γλώσσα βασισμένη σε διαφορετικά προγραμματιστικά παραδείγματα (multi-paradigm), υποστηρίζοντας αντικειμενοστρεφές, προστακτικό και συναρτησιακό στυλ προγραμματισμού.

3.12 Τι είναι Jquery

Jquery είναι μια JavaScript βιβλιοθήκη σχεδιασμένη για να απλοποίηση το HTML DOM time για την διάσχισης και τα χειρισμό του, όπως και το event handling, CSS animation, και το Ajax. Είναι open-source λογισμικό. Από τον Μάιο του 2019, Jquery χρησιμοποιείτε από 73% από τα 10 εκατομμύρια πιο δημοφιλής ιστοσελίδες. Η Microsoft περιεχέι Jquery μαζί με το visual Studio για την χρήσει ASP.NET AJAX και ASP.NET MVC framework.

3.13 Τι είναι το FabricJs

Το FabricJs είναι μια ισχυρή βιβλιοθήκη JavaScript που κάνει την εργασία με τον καμβά (canvas) της HTML5 πιο ευκολότερη. Το Fabric παρέχει ένα μοντέλο αντικείμενου που λείπει από τον καμβά, καθώς και ένα πρόγραμμα μετατροπείς (parser) SVG, ένα επίπεδο αλληλεπίδρασης, και μια ολόκληρο πακέτο από άλλα εργαλεία. Πρόκειται για ένα πλήρως ανοιχτού κώδικα έργο, με άδεια από το MIT, με πολλές συνεισφορές τα τελευταία χρόνια.

3.14 Τι είναι το MVC

Το Model–view–controller (σε συντομογραφία αναφέρεται ως MVC) είναι ένα μοντέλο αρχιτεκτονικής λογισμικού το οποίο χρησιμοποιείται για τη δημιουργία περιβαλλόντων αλληλεπίδρασης χρήστη. Στο μοντέλο αυτό η εφαρμογή διαιρείται σε τρία διασυνδεδεμένα μέρη ώστε να διαχωριστεί η παρουσίαση της πληροφορίας στον χρήστη από την μορφή που έχει αποθηκευτεί στο σύστημα. Το κύριο μέρος του μοντέλου είναι το αντικείμενο *Model* το οποίο διαχειρίζεται την ανάκτηση/αποθήκευση των δεδομένων στο σύστημα. Το αντικείμενο *View* χρησιμοποιείται μόνο για να παρουσιάζεται η πληροφορία στον χρήστη (π.χ. με γραφικό τρόπο). Το τρίτο μέρος είναι ο *Controller* ο οποίος δέχεται την είσοδο και στέλνει εντολές στο αντικείμενο *Model* και στο *View.*

Εκτός από το να διαιρείται η εφαρμογή σε τρία μοντέλα, η σχεδίαση *model–view–controller* ορίζει και τις αλληλεπιδράσεις των μοντέλων.

- Ο **controller** μπορεί να στέλνει εντολές στο μοντέλο και να ενημερώνει την κατάσταση του μοντέλου. Μπορεί επίσης να στέλνει εντολές ώστε να γίνει η αντίστοιχη αναπαράσταση των δεδομένων του μοντέλου μέσω του *View*.

- Το **model** ενημερώνει τις αντίστοιχες αναπαραστάσεις *views* και τους *controllers* όταν υπάρχει αλλαγή στα δεδομένα. Αυτή η ενημέρωση επιτρέπει στα *views* να ενημερώνουν τη γραφική απεικόνιση.
- Το **view** αναπαριστά με γραφικό τρόπο την πληροφορία που περιέχει το *model* δημιουργώντας γραφική παρουσίαση στο χρήστη.

3.15 Γιατί χρησιμοποιήθηκαν

Ας τα πάρουμε από την αρχή για πιο λόγο χρησιμοποιώ κάθε μια από τις τεχνολογίες που βρίσκονται παραπάνω.

1. HTML : δεν υπάρχει και άλλη γλωσσά περιγραφής ιστοσελίδας.
2. SQL : είναι της Microsoft και κάνει πιο εύκολη την χρίση της στο visual studio παρόλα αυτά καθώς χρησιμοποίησα τα migrations θα μπορούσα να χρησιμοποίηση οποιαδήποτε SQL.
3. C#: είναι η γλωσσά της Microsoft και την χρησιμοποίησα για της δυνατότητες που έχει στο cross-platform.
4. .NET Core : είναι πιο γρήγορο πιο λειτουργικό και τρέχει σε όλες τις πλατφόρμες.
5. Identity framework : Υποστηρίζεται από εξωτερικά logins providers όπως Facebook, Google, Microsoft Account και Twitter.
6. Entity framework : το Entity Framework μπορεί να λειτουργήσει σε όλους του υπολογιστές που έχουν το .NET Framework ή .NET Core εγκαταστημένο.
7. CSS: Για μια όμορφη και καλοσχεδιασμένη ιστοσελίδα η χρήση της CSS κρίνεται ως απαραίτητη.
8. Bootstrap: καθώς είναι ήδη περασμένο από το visual studio, αλλιώς θα γινόταν χρίση του TailWind καθώς έχει πιο πολλά στοιχεία άλλα είναι πιο βαρύ.
9. Ajax : για να έχω ασύγχρονη επικοινωνία client-server
10. JavaScript – Jquery: είναι η πιο συνηθισμένη γλωσσά για front-end για ιστοσελίδες και το Jquery υποστηρίζεται άμεσα από το visual studio.
11. FabricJs : Ο καμβάς μας επιτρέπει να δημιουργήσουμε κάποια εντυπωσιακά γραφικά στον ιστό αυτές τις μέρες. Αλλά το API που παρέχει είναι απογοητευτικά χαμηλού επιπέδου. Είναι ένα πράγμα αν θέλουμε απλώς να σχεδιάσουμε μερικά βασικά σχήματα σε καμβά και να τα ξεχάσουμε. Αλλά μόλις υπάρχει ανάγκη για κάθε είδους αλληλεπίδραση, αλλαγή εικόνας σε οποιοδήποτε σημείο ή σχέδιο πιο περίπλοκων σχημάτων - η κατάσταση αλλάζει δραματικά. Το Fabirc λυνει κάποια από αυτά τα προβλήματα.
12. MVC : είναι μια από της καλύτερες αρχιτεκτονικές που μπορείς κάνεις μια εφαρμογή ώστε να μην κάνεις το λεγόμενο “spaghetti code”.

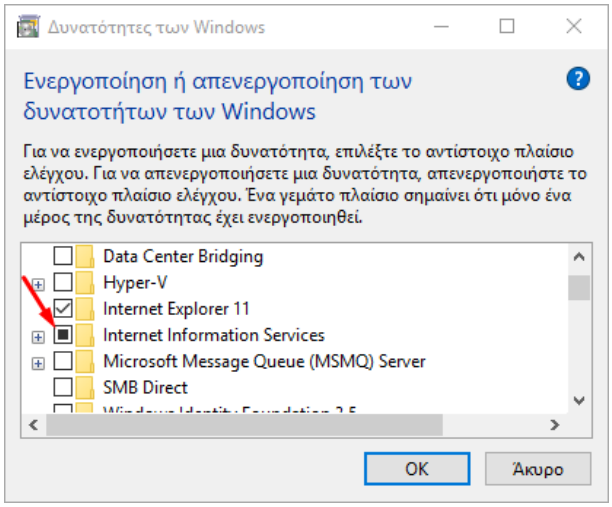
ΚΕΦΑΛΑΙΟ 4

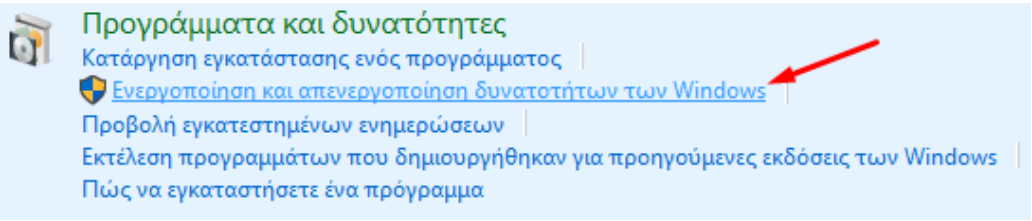
4.1 Προγράμματα που χρησιμοποιήθηκαν

Το βασικό πρόγραμμα που χρησιμοποιήθηκε για την ανάπτυξη της σελίδας είναι το Microsoft Visual Studio Entrerprise 2019 Version 16.3. Στην συγκεκριμένη έκδοση έγινε και η χρήση του .Net Core 3 Μέσα από την εμπειρία που απέκτησα από την χρήση αυτού του προγράμματος έχω να πω πως έχει κάνει μια πάρα πολύ καλή δουλειά η Microsoft έτσι ώστε να είναι πάρα πολύ εύχρηστο προς τον προγραμματιστή. Το Visual Studio είναι ένα πρόγραμμα που μπορεί κάποιος να κάνει project σε διάφορες γλώσσες προγραμματισμού. Στην περίπτωση αυτής της εργασίας έγινε η χρήση της γλώσσας HTML, JavaScript (jquery) η οποία είναι front-end γλώσσα και εκεί γράφουμε αυτά που θέλει να δει ο χρήστης δηλαδή για τα views αρχείο .cshtml. Για του controllers που είναι αρχεία .cs χρησιμοποιεί c#

4.2 Τοπικός Διακομιστείς

Η εγκατάσταση του τοπικού διακομιστή IIS(Internet Information Services) γίνετε αυτόματα αν κάποιος περάσει το Visual studio στον υπολογιστή του τότε είναι έτοιμος για testing του κώδικα. Αν σε περίπτωση θέλουμε να κάνουμε χειροκίνητα την εγκατάσταση του IIS η διαδικασία είναι πολύ απλή. Από τον πίνακα ελέγχου επιλέγουμε από την κατηγορία προγράμματα την επιλογή ενεργοποίηση και απενεργοποίηση δυνατοτήτων των windows. Στο παράθυρο που θα εμφανιστεί απλός ψάχνουμε και τσεκάρουμε την επιλογή που λέει Internet Information Services. Στην Συνέχεια πατάμε ok και μετά από επανακίνηση του συστήματος ο IIS τρέχει κανονικά στον υπολογιστή.





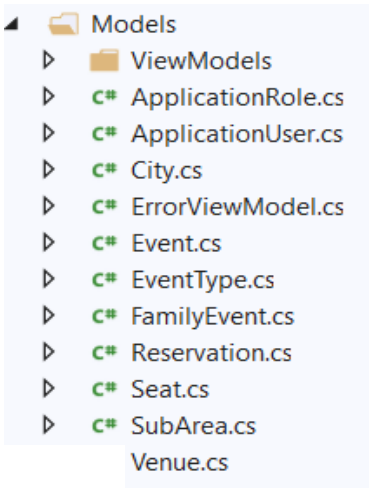
4.3 Βάση Δεδομένων (DataBase)

Για βάση δεδομένων έγινε η χρήση της SQL και η βάση δεδομένων βρισκόταν τοπικά. Η οποία δημιουργήθηκε με την χρήσει του Entity framework και Migrations. Για να γίνει η χρήσει του Entity framework πρέπει πρώτα να περαστούν τα nuget packages. Στη συνέχεια πρέπει να γίνει στο startup file στο ConfigureServices η πιο αρχείο θα χρησιμοποιήσω για να δηλώνω τη δεδομένα να μπουν στη βάση.

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddDbContext<ApplicationDbContext>(options =>
        options.UseSqlServer(
            Configuration.GetConnectionString("DefaultConnection")));
    services.AddSignalR();
```

Συνεχίζοντας δημιουργούμε τα μοντέλα μας καθώς χρησιμοποίησα MVC αρχιτεκτονική. Όταν έχουμε δημιουργήσει τα μοντέλα που θέλουμε τότε τα θα πάμε στο αρχείο που δηλώσαμε στα services στο startup file, και θα δημιουργήσουμε Dbset των μοντέλων. Το Θετικό με αυτή την τεχνική είναι ότι οποιαδήποτε στιγμή μπορούμε να προσθέσουμε ή να αφαιρέσουμε στοιχεία, πεδία ή και πινάκες χωρίς να υπαρχή μεγάλη δυσκολία ή κάπου πρόβλημα.

```
public DbSet<Venue> Venue { get; set; }
```



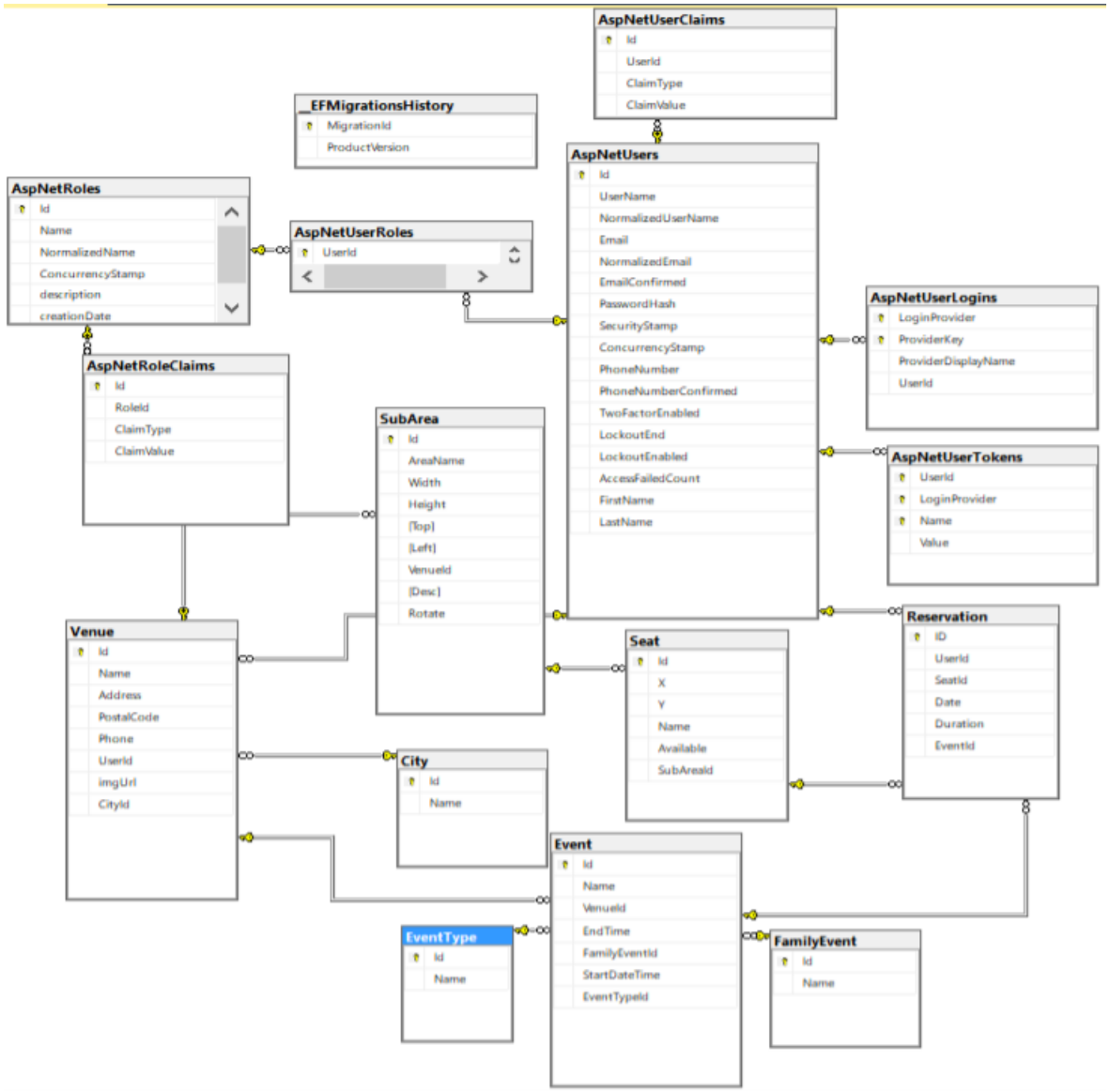
Επόμενο βήμα πρέπει να έχουμε δημιουργήσει μια βάση. δεδομένων από το SQL Server Object Explorer του visual studio να δημιουργήσουμε το connection string στο appsettings.json. Τέλος θα χρειαστεί ένα ακόμα nuget packages Microsoft.EntityFrameworkCore.Tools και στο package manager console παράθυρο που υπάρχει μέσα στο visual studio θα εκτελέσουμε δύο εντολές
`add-migration` και το όνομα που θέλουμε, αυτό θα δημιούργησε ένα αρχείο που περιγραφεί τα δεδομένα που

```
migrationBuilder.CreateTable(
    name: "Event",
    columns: table => new
    {
        Id = table.Column<int>(nullable: false)
            .Annotation("SqlServer:Identity", "1, 1"),
        Name = table.Column<string>(nullable: true),
        Date = table.Column<DateTime>(nullable: false),
        VenueId = table.Column<int>(nullable: false)
    },
    constraints: table =>
    {
        table.PrimaryKey("PK_Event", x => x.Id);
        table.ForeignKey(
            name: "FK_Event_Venue_VenueId",
            column: x => x.VenueId,
            principalTable: "Venue",
            principalColumn: "Id",
            onDelete: ReferentialAction.Restrict);
    });
```

θέλουμε να αποθηκεύσουμε στην βάση άλλα και τις σχέσης τους.

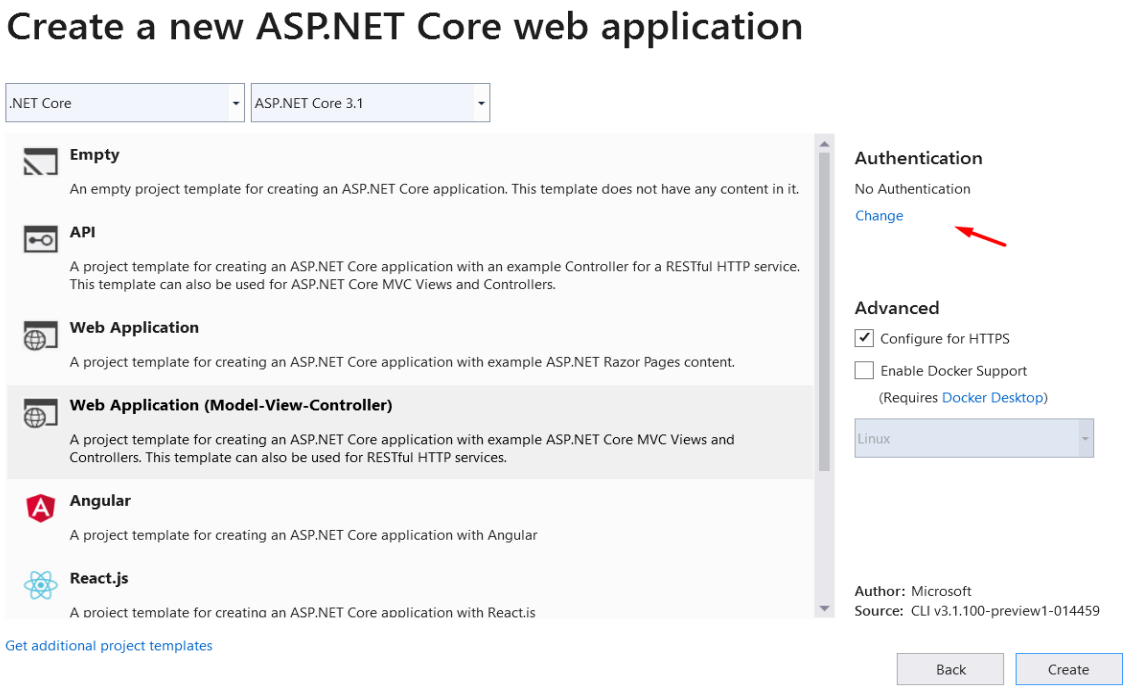
Άλλα για να δημιουργηθούν οι πινάκες της βάσης πρέπει να εκτελέσουμε αυτήν την εντολή αφού έχουμε δημιουργήσει το migration file μας, update-database αυτή η εντολή θα πάρει το migration file και θα το μετάφραση σε SQL ακόμη θα δημιουργήσει ένα migration πινάκα που αποθηκεύει τα

migration που έχουν γίνει ώστε να μην ξανά κάνει τα ίδια migration. Με αυτόν τον τρόπο μπορούμε με ένα ENTER να δημιουργήσουμε αυτήν τη βάση.



4.4 Χρήση του Identity Framework
4.4.1 Δημιουργία User και Roles

Οι χρήστες και οι ρόλοι δημιουργήθηκαν με την βοήθεια του identity framework που είδα στο



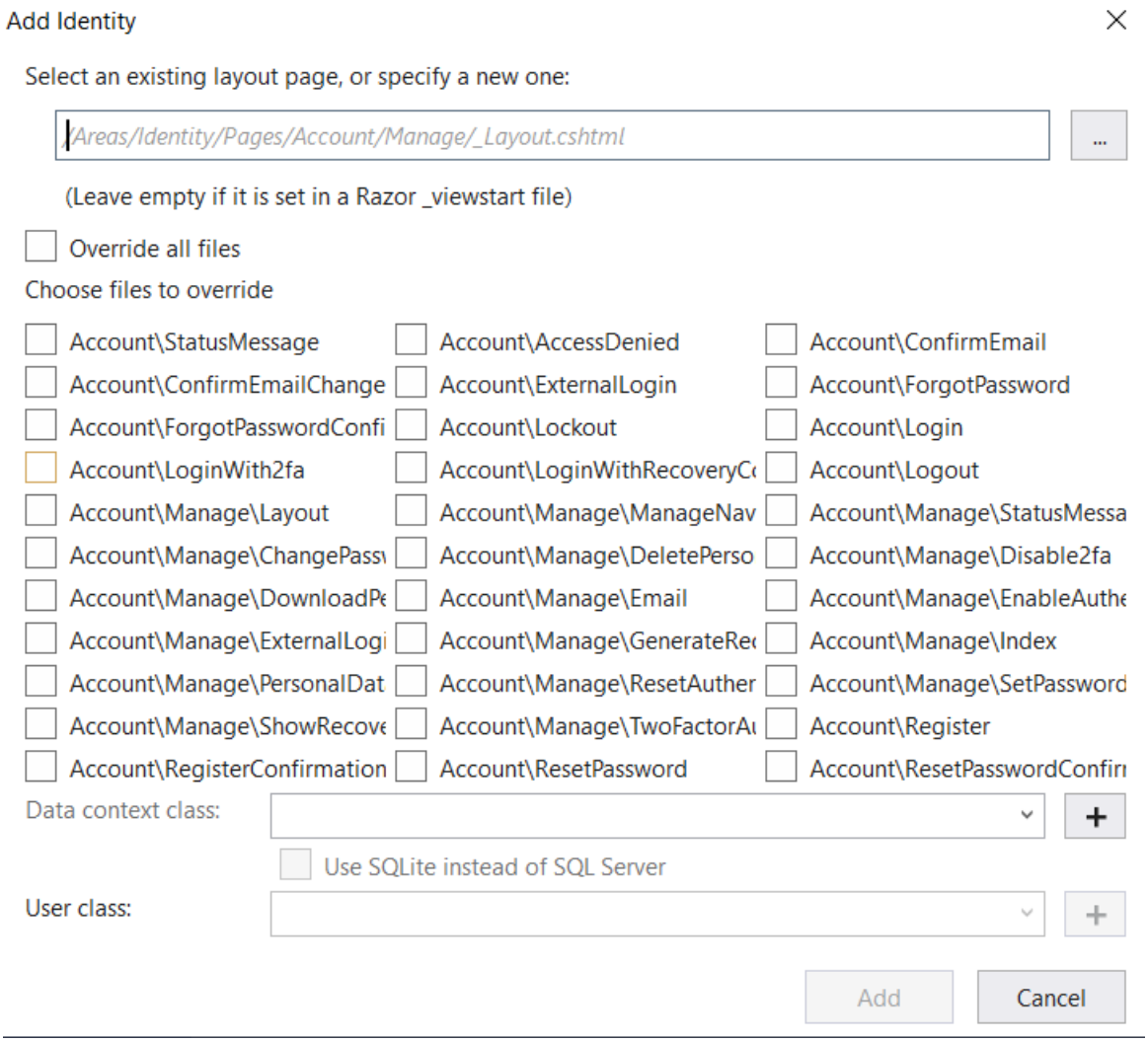
προηγούμενο κεφαλαίο. Υπάρχουν διάφοροι τρόποι για να προσθέσεις αυτό το framework άλλα όπιο εύκολος και αυτός που χρησιμοποίησα είναι όταν δημιουργείς το project να πρόσθεσης αυτήν την επιλογή. Θα μας εμφάνισή ένα παράθυρο και θα επιλέξουμε τον Individual user accounts και θα δημιουργήσει όλα όσα χρειάζονται για να υπάρχουν χρήστες.

4.4.2 Τροποποίηση User και Roles

Επειδή στα δημιουργεί δεν σημαίνει και ότι είναι έτοιμα για την χρήση που χρειάζεται. Υπάρχουν κάποια βασικά στοιχεία όπως e-mail, τηλέφωνο, κωδικό hash (είναι κρυπτογραφημένοι) και άλλα. Όμως δεν έχει όνομα και επίθετο που χρειάζεται η εφαρμογή μας και εμείς πρέπει να κάνουμε override με την βοήθεια του entity και να βάλουμε τα δικά μας πεδία και κάνουμε σχεδόν ότι κάνουμε για όλα τα πεδία. Σχεδόν γιατί πρέπει να πάμε στο start up και να βάλουμε το παρακάτω κομμάτι κώδικα.

```
services.AddIdentity<ApplicationUser, ApplicationRole>(
    options => options.Stores.MaxLengthForKeys = 128)
    .AddEntityFrameworkStores<ApplicationDbContext>()
    .AddDefaultUI()
    .AddDefaultTokenProviders();
```

Επιπλέον χρειάζεται να αλαλάξουμε τον κωδικά που δημιουργεί νέους χρήστες και εκεί που ο χρήστης τροποποιεί τα στοιχεία του και του ρόλους του καθώς θα τους χρειαστούμε στη συνεχεία. Πρώτα πρέπει να εμφανίσουμε τα αρχεία αυτά καθώς το framework τα είχε ¨κρυφά¨ και πρέπει να τα κάνουμε scaffold, όποτε κάνουμε add νέο scaffold αντικείμενο και διαλέγουμε το identity μας εμφανίζει αυτό το παράθυρο :



επιλέγουμε οποία αρχεία θέλουμε να επεξεργαστούμε στην δικιά μας περιπτώσει θα τα χρειαστούμε όλα για να είμαστε και μελλοντικά έτοιμοι.

4.4.3 Πρόσθεση προσαρμοσμένων User δεδομένων

Με αυτόν τον τρόπο έχω δημιουργήσει κάποια αρχεία που είναι front-end και back-end όπως της asp τα .aspx. Τα κυριότερα αρχεία που χρειάζονται αλλαγή είναι το register και index. Ακόμα είναι αναγκαίο να αλλάξει και τον front-end και back-end γιατί έχουμε περάσει να στοιχειά και χρειαζόμαστε ο χρήστης να μπορεί να βάλει τα στοιχεία και να αποθηκεύονται. Για τον front-end είναι πολύ απλό δημιουργό δυο πεδία “input” και στο back-end χρησιμοποιήσω το παρακάτω κώδικα :

```
var user = new ApplicationUser { UserName = Input.Email, Email = Inp
await _userManager.SetPhoneNumberAsync(user, Input.PhoneNumber);
var result = await _userManager.CreateAsync(user, Input.Password);
```

Εδώ στο μοντέλο μας δηλώνω τι τύπου είναι τα πεδία μου άλλα αυτό μόνο δεν αρκεί για να τα αποθηκεύσω.

Δημιουργώ έναν user του προσθέτω όλα τα στοιχεία που μου στέλνει ο χρήστης (τα στοιχειά είναι validate από το front-end) και με την εντολή createAsync ο χρήστης δημιουργείτε και αποθηκευτικέ στην βάση. Εγώ περνώ το result και αν όλα πήγαν σωστά περνώ το κωδικό του και τον κρυπτογραφώ και στέλνω για επιβεβαίωση e-mail.

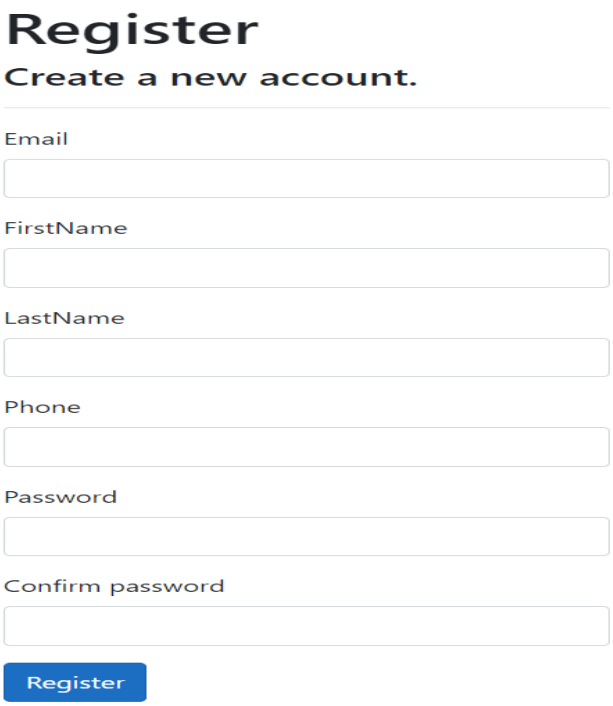
4.4.4 Επιλογές κωδικού του χρήστη

Κάθε εφαρμογή θέλει ο χρήστης να έχει έναν ασφαλή κωδικό. Με αυτό το σκεπτικό για να παραμετροποιήσεις τα passwords του identity, καθώς όταν μου τα στέλνει ο χρήστης δεν είναι σωστό να τα δω για να ελέγξω αν οι κωδικοί τηρούν τα κριτίκαρα ασφαλείας που πρέπει. Χρειαζόμαστε να πάω στο start up file και να αλλάξω τις επιλογές που μου δίνει το identity framework.

```
services.Configure<IdentityOptions>(options =>
{
    // Default Password settings.
    options.Password.RequireDigit = true;
    options.Password.RequireLowercase = false;
    options.Password.RequireNonAlphanumeric = true;
    options.Password.RequireUppercase = false;
    options.Password.RequiredLength = 8;
    options.Password.RequiredUniqueChars = 0;
});
```

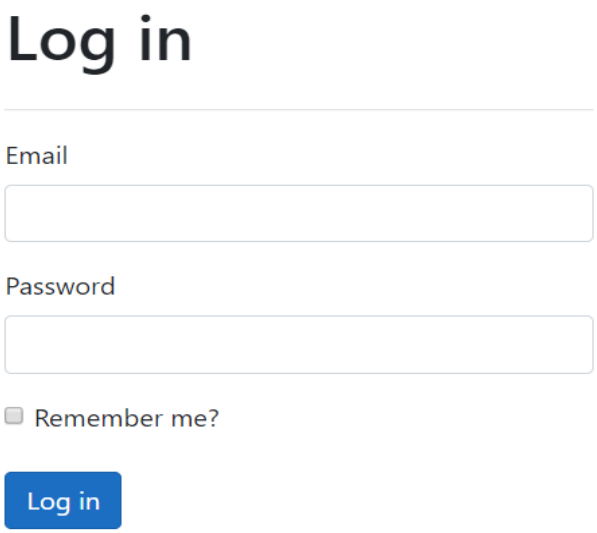
4.4.5 Εγγραφή που χρήστη

Ο χρήστης διαλέγει την επιλογή “register”
συμπληρώνει τα στοιχειά του και αν όλα είναι σωστά τότε συνδέεται στην σελίδα.



4.4.6 Σύνδεση του χρήστη

Στην συγκεκριμένη ιστοσελίδα το κοινό από χρήστες που υπάρχουν είναι 3 ειδών. Άλλα όλοι κάνουν σύνδεση με τον ίδιο τρόπο. Τα είδη τους χωρίζονται ανάλογα με τον ρόλο τους 1 Admin : που είναι ο ιδιόκτητης της σελίδας εγώ σε αυτήν την περιπτώσει και δίνω του ρόλους, 2 Venue : που αξιοποιεί τις υπηρεσίες της σελίδας και 3 Client : που τις χρησιμοποιεί, άλλα θα αναλύσουμε πιο πολύ τους ρολούς σε άλλα κεφαλαίο.



4.4.7 Έλεγχος ταυτότητας και ρόλος του χρήστη

Ο έλεγχος ταυτότητας του χρήστη γίνεται με τον sing In Manager του identity framework και όχι το κλασσικό session που χρησιμοποιήσαμε στο .net framework.

```
var result = await _signInManager.PasswordSignInAsync(Input.Email, Input.Password, Input.RememberMe, lockoutOnFailure: tru
if (result.Succeeded)
{
    _logger.LogInformation("User logged in.");
    return LocalRedirect(returnUrl);
```

Επιπλέον με ίδιο τρόπο. ελέγχουμε και για τον ρόλο του χρήστη που θέλω για να του δώσουμε επιπρόσθετα δικαιώματα.

4.5 Οι controllers

OI controllers διαχειρίζονται τα δεδομένα από τα μοντέλα και τα views άρα όσα μοντέλα έχουμε τόσους controllers θα χρειαστούν.

4.5.1 Home Controller

Ο Home Controller είναι αυτοδημιούργητος καθώς όταν δημιουργούμε ένα νέο project στο visual studio υπάρχει εξαρχής και διαχειρίζεται τη κύρια σελίδα “home”.

4.5.2 Venue Controller

Ο Venue Controller διαχειρίζεται τα venue, άρα φροντίζει για την δημιουργία τους, την επεξεργασία τους, την διαγραφή τους και την εμφάνιση τους. Για την δημιουργία έχουμε την φόρμα που συμπληρώνει ο χρήστης τα στοιχεία του venue και μετά τρέχει η συνάρτηση “create” για να δημιουργήσει το αντικείμενο στη βάση. Με μια παράμετρο δεχόμαστε τα στοιχεία που έβαλε ο χρήστη σαν αντικείμενο και δημιουργώ ένα νέο αντικείμενο και το αποθηκεύω στη βάση. Είναι αναγκαίο να δημιουργήσω νέο αντικείμενο γιατί από τον χρήστη δεν λαμβάνω venue,άλλα Venue View Model αντικείμενο. Ο λόγος που γίνεται αυτό είναι για την εικόνα. Στο venue μοντέλο η εικόνα. είναι ένα απλό αλφαριθμητικό στοιχείο άλλα ο χρήστης δεν μπορεί να μας στείλει την εικόνα. του σαν αλφαριθμητικό. Έτσι είναι αναγκαίοι να δημιουργεί νέο μοντέλο και να αποθηκευτεί η εικόνα. στο server με ένα μοναδικό όνομα και στη συνεχεία αυτό το όνομα να αποθηκευτεί στη βάση σαν imgUrl.Οι επεξεργασία λειτουργεί με τον ίδιο τρόπο.

```
Venue newshop = new Venue
{
    Name = model.Name,
    Address = model.Address,
    CityId = model.CityId,
    PostalCode = model.PostalCode,
    Phone = model.Phone,
    UserId = userManager.GetUserId(HttpContext.User),
    imgUrl = uniqueFileName

};
_context.Add(newshop);
await _context.SaveChangesAsync();
```

```
string uniqueFileName = null;
if(model.Photo != null)
{
    string uploadsFolder = Path.Combine(HostingEnviromnet.WebRootPath, "images");
    uniqueFileName = Guid.NewGuid().ToString() + "_" + model.Photo.FileName;
    string filePath = Path.Combine(uploadsFolder, uniqueFileName);
    model.Photo.CopyTo(new FileStream(filePath, FileMode.Create));
}
```

Η διαγραφή χρειάζεται να βρούμε το στοιχειά που θέλουμε να διαγράψουμε να δούμε αν άλλα στοιχειά εξαρτώνται από αυτό (πχ έχουν αποθηκευτεί κράτησης σε από το venue) και αν έχουν να τα διαγράψω και αυτά ώστε να μην δίνει λάθος η βάση.

```
public async Task<IActionResult> DeleteConfirmed(int id)
{
    var venue = await _context.Venue.FindAsync(id);
    _context.Venue.Remove(venue);
    await _context.SaveChangesAsync();
    return RedirectToAction(nameof(Index));
}
```

4.5.3 Sub Areas Controller

Ο Sub Areas Controller διαχειρίζεται τα Sub Areas του venue. Με παρόμοιο τρόπο θα αποθηκεύσουμε στην βάση, άλλα θα λάβουμε τα στοιχεία με εντελώς διαφορετικό τρόπο. Προηγούμενο τα λαμβάναμε από μια φόρμα τώρα επειδή η ιδιαιτερότητες του Sub Area δεν το επιτρέπου χρειάζεται να λάβουμε ένα JSON αρχείο που θα έχει τα στοιχεία τον Sub Areas και εμείς μετά θα τα λάβουμε στη βάση. Καθώς όμως ο controller δεν μπορεί να επικοινωνήσει με τον view πρέπει να κάνουμε χρήση του Ajax.

```
public asy   $.ajax(                                                    ;ubAreas)
            {
                    type: "Post",
                    url: '@Url.Action("Create", "SubAreas")',
                    dataType: 'json',
                    contentType: 'application/json; charset=utf-8',
                    data: JSON.stringify(SubAreas),
                    success: function (data) { alert("saved"); },
                    error: function (data) { alert("fail to save");},
                    accept: 'application/json'
                })
            });
```

Η δυσκολία σε αυτήν τη περίπτωση δεν ήταν στη αποστολή του αρχείο άλλα στη δημιουργία του καθώς πρέπει να πάρουμε τα στοιχεία που δημιουργεί ο χρήστη μέσα από τον καμβά και να τα κάνουμε αντικείμενο και να τα μετατρέψουμε σε JSON. Όλοι διαδικασία που ακολουθάει δεν συμβαίνει μέσα στον controller άλλα στο view front-end άλλα είναι αναγκαίο ώστε να δημιουργηθεί

```
$("#save").click(function () {
            var tmp = {};
             var SubAreas = [];

    var canvasObjects = canvas.getObjects();
            for (obj in canvasObjects) {
              if (canvasObjects[obj].get('type') == 'group') {
                 tmp.Top = canvasObjects[obj].get('top');
                 tmp.Left = canvasObjects[obj].get('left');
                 tmp.AreaName = canvasObjects[obj].get('id');
                 tmp.Rotate = canvasObjects[obj].get('angle');
                 tmp.Width = canvasObjects[obj].getScaledWidth();
                 tmp.Height = canvasObjects[obj].getScaledHeight();;
               }
               if (canvasObjects[obj].get('type') == 'text') {
                  tmp.AreaName = canvasObjects[obj].get('text');
                  SubAreas.push(tmp);
                  tmp = {};
               }
    }
```

το sub area. Αν η εργασία αυτή γινόταν από δυο άτομα και οι εργασίες χωρίζονταν σε front-end και back -end, το front-end θα έπρεπε να φροντίσει για αυτήν τη διαδικασία και θα αναλύσω αυτή τη διαδικασία πιο πολύ όταν θα αναφέρομαι στο front-end.

4.5.4 Seat Controller

Ο Seat Controller είναι υπεύθυνος για τη δημιουργία, αποθήκευση και διαγραφή θέσεων του συστήματος. Η δημιουργία γίνεται στο front-end όπως και τον sub areas και μέσα από Ajax δέχεται Al αντικείμενο το δημιουργεί και το αποθηκεύει στη βάση.

Για τη διαγραφή εδώ χρειάζεται να διαγράψουμε στοιχεία τα οποία δείχνουν τη θέση ώστε να διαγραφή και να μη βγάζει η βάση λάθος. Βρίσκουμε τη θέση που θέλουμε να διαγράψουμε βάση το id της. Στη συνεχεία είναι αναγκαίο να δούμε αν έχουν γίνει κράτησης σε αυτήν τη θέση, αν έχουν γίνει το τις διαγράφουμε. Τέλος διαγράφω την θέση που θέλω και αποθηκεύω την βάση. Όπως έχετε παρατηρήσει δεν υπάρχει πουθενά SQL κώδικας για χάρις την βοήθεια του entity framework. Επιπλέον υπάρχει λειτουργιά ώστε ο ιδιοκτήτης αν για κάποιο λόγο δεν μπορεί να προσφέρει την συγκεκριμένη θέση (πχ κάθετε κάποιος άλλος τώρα) υπάρχει δυνατότητα να κάνει την θέσει μη διαθέσιμη.

```
public async Task<IActionResult> DeleteConfirmed(int id)
{
    var seat = await _context.Seat.FindAsync(id);
    var res = _context.Reservation
        .Where(r => r.SeatId == seat.Id).ToList();
    if (res.Count != 0)
    {
        foreach (var r in res)
        {
            _context.Reservation.Remove(r);
        }
    }
    _context.Seat.Remove(seat);
    await _context.SaveChangesAsync();

    return RedirectToAction(nameof(ListOfMySeats));
}
```

```
public async Task<IActionResult> CreateSeatMap([FromBody] JsonSeatModel[] seats,int? subAreaId)
{
    if (ModelState.IsValid)
    {
        foreach (var s in seats)
        {
            Seat seat = new Seat
            {
                Name = s.Name,
                X = s.left,
                Y = s.top,
                SubAreaId = (int)subAreaId,
                Available = true

            };
            _context.Add(seat);
        }
    }
```

4.5.6 Reservation Controller

Ο Reservation Controller είναι υπεύθυνος για της κράτησης, την δημιουργία, την διαγραφή, τον έλεγχο και την εμφάνισή των κρατήσεων. Η δημιουργία και η διαγραφή γίνονται με παρόμοιο τρόπο όπως είναι και τα προηγούμενα, ο χρήστης δίνει τη θέση που θέλει να κάνει κράτηση την ημερομηνία και ώρα και την ώρα που πιστεύει θα φύγει, από η διάφορα τους βγαίνει η διάρκεια της κράτησης, το event και δημιουργείτε η κράτηση. Για την διαγραφή παίρνω το id και διαγράφω την συγκεκριμένη κράτηση.

Για την εμφανίσει όλων των κρατήσεων γίνεται φιλτράρισμα ώστε να γίνεται πιο εύκολα η αναζήτηση των κρατήσεων.

```
public async Task<IActionResult> Index(string sortOrder)
```

Όταν τρέχει ο κώδικας να εμφάνιση όλες τις κρατήσεις παίρνει μια παράμετρο την sortOrder που δηλώνει με ποιο στοιχείο θέλει ο χρήστης να κατατάξει των πινάκα του. Μέσα από τη λειτουργία του .Net core ViewData, περνάμε τα στοιχεία του view στο controller, δηλαδή αυτή η παράμετρο μας δίνει ένα ViewData.

```
ViewData["NameSortParm"] = sortOrder == "LastName" ? "LastName_desc" : "LastName";
ViewData["DateSortParm"] = sortOrder == "Date" ? "Date_desc" : "Date";
ViewData["VenueSortParm"] = sortOrder == "Venue" ? "Venue_desc" : "Venue";
ViewData["AreaNameSortParm"] = sortOrder == "AreaName" ? "AreaName_desc" : "AreaName";
ViewData["EventSortParm"] = sortOrder == "Event" ? "Event_desc" : "Event";
```

Ένα από αυτά τα ViewData θα μας δοθεί. Χρησιμοποιούμε το ViewData γιατί η μεταβλητή δεν χάνεται άλλα αποθηκεύετε. Με αυτόν τον τρόπο μπορώ να δώσει την τιμή που θέλω στο sortOrder ώστε αν το ViewData είναι κενό να του δώσει την λειτουργία “LastName” αν δεν είναι και είναι “LastName” να του δώσει “LastName_desc” ή το αντίστροφο. Ώστε να μπορούμε να αλλάζουμε την κατάταξη σε αύξουσα ή φθίνουσα. Αφού βρούμε τις κρατήσεις που θέλουμε να εμφανίσουμε τρέχουμε μια άλλη συνάρτηση που είναι αυτή που θα κάνει τη κατάταξη.

```
res = FilterRes(sortOrder, res);
return  View(res);
```

Η FilterRes είναι μια συνάρτηση που επιστρέφει κρατήσεις και μέσα είναι μια switch που ελέγχει το αλφαριθμητικό που της δίνουμε ώστε να κάνει την σωστή κατάταξη.

Μια ακόμα σημαντική ενεργεία που κάνει ο Reservation Controller είναι να βρίσκει αν υπάρχουν κρατήσεις στις θέσεις. Η συνάρτηση παίρνει τα στοιχεία που είναι απαραίτητα για να βρει που και πότε θέλει να κάνει ο χρήστης κρατήσεις από το front-end και επιστρέφει ένα Json File που λέει ποιες θέσεις. δεν μπορεί να κάνει κράτηση ο χρήστης. Πρώτα ελέγχω αν είναι θέση είναι μη διαθέσιμη για κάποιον λόγο που έχει δηλώσει ο ιδιοκτήτης. Στη συνεχεία ελεγείων αν υπάρχει κράτηση εκείνη την στιγμή η αν η κράτηση καλύπτει άλλη κράτηση όλα μπαίνουν σε έναν πινάκα και στέλνονται για να εμφανιστούν με κόκκινο στο front-end.

```
if (NowDateTime.Date == ResDate.Date)
{
    var seatsUnAvailable = _context.Seat.Where(s => s.SubAreaId == subArea.Id && s.Available == false).ToList();
    foreach (var s in seatsUnAvailable)
    {
        seatIds[i++] = s.Id;
    }
}
    reservations = reservations
         .Where(res => ResDate <= res.Date
        && res.Date <= ResDate.AddMinutes(Duration.TotalMinutes)
        || ResDate <= res.Date.AddMinutes(res.Duration.TotalMinutes)
        && res.Date.AddMinutes(res.Duration.TotalMinutes) >= ResDate.AddMinutes(Duration.TotalMinutes)).ToList();

foreach (var r in reservations)
{
    seatIds[i++] = r.SeatId;
}

return  Json(seatIds);
```

4.5.7 Events Controller

Ο events controller είναι υπεύθυνος για την δημιουργία, την διαγραφή και την εμφανίσει των events που γίνονται “σήμερα” στην εφαρμογή. Για τη δημιουργία η ιδιαιτερότητα εδώ ήταν ότι χρειάζεται μια επιπλέον συνάρτηση για γίνει η δημιουργία για τη ο χρήστης μας δίνει συγκεκριμένες μέρες που θέλει να γίνεται το event. Για αυτό το λόγο χρειάζεται να ελέγξουμε την κάθε μέρα αν είναι π.χ Δευτέρα, Τρίτη κ.α. Με αυτό το σκεπτικό πρέπει πριν δημιουργηθεί να ελέγχεται αν η μέρα είναι σωστή.

```
if (ev.Repeat.AfterNumTimes != "")
{
    for (int i = 0; i < int.Parse(ev.Repeat.AfterNumTimes)*7; i++)
    {
        if (CorrectDay(ev,i, everyNum) == true)
        {
            Event nEvent = new Event
            {
                Name = ev.Name,
                StartDateTime = ev.StartDateTime.AddDays(i+ (everyNum * 7))
                .ToLocalTime(),
                EndTime = ev.EndTime.AddDays(i+ (everyNum * 7))
                .ToLocalTime(),
                EventTypeId = int.Parse(ev.EventTypeId),
                VenueId = Venue.Id,
                FamilyEventId = FamilyEvent.Id

            };

             _context.Add(nEvent);
        }
```

Το AfterNumTimes δηλώνει ότι ο χρήστης έβαλε να επαναλαμβάνεται για ένα συγκεκριμένο αριθμό, η συνάρτηση CorrectDay ελέγξει την ήμερα, οι ήμερες αυξάνονται με τον μετρητή της for. Όταν δημιουργείται το event στην αρχική ημερομηνία προσθέτονται οι νέες μέρες και βγαίνει το σωστό αποτέλεσμα στις ημερομηνίες. Αν ο χρήστης δώσει μια συγκεκριμένη ημερομηνία που θέλει να τελειωνόσουν τα events τότε η δημιουργία τους γίνεται με χρήση while.

```
int i = 0;
while (ev.Repeat.UntilDate > ev.StartDateTime.AddDays(i))
```

Η συνάρτηση CorrectDay επιστρέφει ένα boolean εξετάζοντας την κάθε περιπτώσει αν ήμερα είναι Δευτέρα ή Τρίτη κ.α.Το Json που δέχεται είναι το event που έχει δημιουργήσει ο χρήστης στο front-end της σελίδας και έρχεται στον controller μεσώ Ajax.

```
public bool CorrectDay(JsonEventModel ev,int i,int everyNum)
{
    bool correctDay = false;
    if (ev.Repeat.M == true && ev.StartDateTime.AddDays(i+everyNum * 7)
        .DayOfWeek.ToString() == "Monday")
    {
        correctDay = true;
    }
    else if (ev.Repeat.Tu == true && ev.StartDateTime.AddDays(i+everyNum * 7)
        .DayOfWeek.ToString() == "Tuesday")
    {
        correctDay = true;
    }
```

Για την διαγραφή υπάρχουν οι δυνατότητες να διαγράψει ένα συγκεκριμένο event οι όλα τα event τα οποία ανήκουν στη συγκεκριμένοι οικογένεια. Για το να συγκεκριμένο event η διαδικασία είναι ίδια με τα άλλα αντικείμενα παίρνω το id και το διαγράφω, άλλα για την οικογένεια είναι λίγο διαφορετικά. Ελέγχω πρώτα αν έχει επίλεκτη να διαγράφουν όλα. Στη συνεχεία βρίσκω την οικογένεια που βρίσκεται το event. Ύστερα για κάθε event που υπάρχει στη λίστα που θα βρω χρειάζεται να διαγράφουν οι κρατήσεις που είναι πιθανό να είχαν και τέλος διαγράφω όλοι τη λίστα.

```
if (dAll == true)
{

    var familyEvents = _context.Event
        .Include(r => r.FamilyEvent).Include(r => r.EventType)
        .Where(e => e.FamilyEventId == ev.FamilyEventId).ToList();
    foreach(var @event in familyEvents)
    {
        var hasReservations = _context.Reservation
            .Where(r => r.EventId == @event.Id).ToList();
        _context.Reservation.RemoveRange(hasReservations);
    }
         _context.Event.RemoveRange(familyEvents);
}
    else
```

Για την εμφάνισή των event που γίνονται σήμερα άπλα παίρνω από την βάση τα event που έχουν την ίδια ημερομηνία με σήμερα και η τελική τους ώρα είναι μεγαλύτερη από το “τώρα”.

```
public async Task<IActionResult> EventsForToday(String City)
{
    var applicationDbContext = _context.Event
        .Include(r => r.FamilyEvent)
        .Include(r => r.Venue)
        .Where(e => e.StartDateTime.Date == DateTime.Now.Date
        && e.EndTime.TimeOfDay > DateTime.Now.TimeOfDay);
```

4.5.8 Application User Controller

Ο Application User Controller σε αντιθέσει με τους άλλους controllers δεν είναι υπεύθυνος για την δημιουργία των χρηστών καθώς για αυτό όπως είπαμε είναι για το identity framework. Άλλα χρειάζεται ώστε να εμφανίζει τους χρήστες και να μπορεί ο διαχειριστής της σελίδας να τους δώσει το δικαίωμα να δημιουργήσουν το δικό τους venue. Τη συνάρτηση μπορεί να την εκτέλεση μόνο διαχειριστής (Admin), βρίσκουμε το χρήστη που επιθυμούμε και αν είναι στο ρόλο που επιλέξαμε να τον βάλουμε τον αφαιρούμε αλλιώς τον προσθέτουμε, κάνουμε αποθήκευση στης αλλαγές και τέλος. Δώσαμε ρόλο σε έναν χρήστη ή αφαιρέσαμε.

4.6 Front-end

Το front-end μιας εφαρμογής είναι συνήθως το περιεχόμενο που βλέπει ο χρήστης και κάποιες άλλες λειτουργίες που χρειάζεται η εφαρμογή για να τρέξει (π.χ να πάρει κάποια δεδομένα μεσώ Ajax). Καθώς χρησιμοποιώ την αρχιτεκτονική λογισμικού MVC όλα όσα σχετίζονται με τον front-end βρίσκονται στα Views. Σε αυτούς του φακέλους υπάρχουν όλες οι σελίδες που τρέχει και μπορεί να δει κάποιος χρήστης. Οι σελίδες χωρίζονται στα κομμάτια που δεν αλλάζουν που βρίσκονται στον Shared φάκελο και τον κεντρικό περιχεόμενο αλλάζει ανάλογα την σελίδα που βρίσκεται ο χρήστης. Αυτό επιταχύνεται με το Bootstrap και CSS χωρίζουμε την σελίδα σε divs και στην κλάση “container” τρέχουμε μια εντολή του .Net Core ώστε να αλλάζει τα “container” με την σελίδα που επιθυμεί ο χρήστης.

4.6.1 Menu που δημιουργήθηκαν

```
<div class="container">
    <main role="main" class="pb-3 ">
        @RenderBody()
    </main>
</div>
```

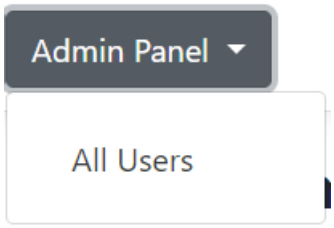
Καθώς η εφαρμογή δίνει επιπλέον δικαιώματα σε χρήστες που έχουν το ρόλο ιδιοκτήτη ή διαχειριστή, χρειάζεται να δημιουργηθούν δυο Menu ή control panel ώστε να μπορέσουν οι χρήστες να εκτελέσουν της δραστηρίοτητες τους με ευκολία. Πρώτα χρειάζεται να εκτελέσει αν όντως ο χρήστης βρίσκεται στον ρόλο που θέλω. Στη συνεχεία δημιουργώ μια λίστα και ένα div με την κλάση που υπάρχει είδη στο bootstap για dropdown λίστες. Επόμενο βήμα να δημιουργήσω το κουμπί που θα ανοίγει τη λίστα και μετά το κουμπί που θα είναι μέσα στη λίστα και θα κάνει μια λειτουργία. Το κουμπί με την λειτουργία θα την εκτελέσει, η λειτουργία είναι η Create από τον controller Venue. Όποτε η εφαρμογή θα πάει στο controller και θα τρέξει η συνάρτηση Create που επιστέφει την σελίδα για να δημιουργήσει ο χρήστης το Venue(η συνάρτηση που να την τρέξει μόνο χρήστης που έχει τον ρόλο Venue ώστε να μην μπορεί κάποιος να πάρει το link και να εισέλθει την σελίδα).

```
@if (SignInManager.IsSignedIn(User) && User.IsInRole("Venue"))
{
    <li class="nav-item">
        <div class="dropdown">
            <button class="btn btn-secondary dropdown-toggle"
                    type="button" id="dropdownMenu2"
                    data-toggle="dropdown"
                    aria-haspopup="true" aria-expanded="false">
                My Venue
            </button>
            <div class="dropdown-menu" aria-labelledby="dropdownMenu2">
                <button class="dropdown-item" type="button">
                <a class="nav-link text-dark" asp-controller="Venue"
                   asp-action="Create">
                    Create Venue</a></button>
```

Παροιμιωδώς γίνεται και για τον διαχειριστή άπλα αλλάζουμε τον ρόλο που ζητάμε.

```
@if (SignInManager.IsSignedIn(User) && User.IsInRole("Admin"))
{
    <li class="nav-item">
        <div class="dropdown">
            <button class="btn btn-secondary dropdown-toggle"
                    type="button" id="dropdownMenu2"
                    data-toggle="dropdown" aria-haspopup="true"
                    aria-expanded="false">
                Admin Panel
            </button>
```

Όταν τρέχω την εφαρμογή αυτή είναι η εμφάνιση του.



4.6.2 FabricJs και θέσεις

Για την δημιουργία και εμφανίσει των θέσεων και των sub area γίνεται με την βοήθεια της ανοικτής κώδικα βιβλιοθήκης της JavaScript FabricJs. Καθώς ο χρήστης μέσα σε έναν χώρο πρέπει να μπορεί να διαλέξει την θέση που επιθυμεί να κάτσει. Το FabricJs επεκτείνει τον καμβά της JavaScript.

```
var canvas = new fabric.Canvas('c');
```

Αν τρέξουμε μια σελίδα HTML με αυτό στο για script θα πάρουμε έναν κενό καμβά μέσα σε αυτόν τον καμβά μπορώ να σχεδιάσω ότι θέλω ή να βάλω μια εικόνα, στη συγκεκριμένη περιπτώσει θα βάλουμε ένα μικρό τετράγωνο που δηλώνει την θέσει. Το τετράγωνο έχει πλάτος, ύψος , το left δηλώνει πόσο μακριά είναι από το τέλος του καμβά από τα αριστερά αντίστοιχα το top για το πάνω μέρος του καμβά, fill το γέμισμα χρώματος, το scaleY πόσο μεγάλο να είναι ανάλογα τον άξονα Y,

ένα id, originX Οριζόντια προέλευση μετασχηματισμού, originY η κάθετη. Άλλα αυτόν δεν είναι αρκετό αν ο χρήστης θέλει να δημιουργεί εκατό θέσεις δεν μπορεί να κάνει εκατό κλικς και να βάζει σε σωστά σημεία τις θέσεις. Για αυτό εδώ παίρνουν ρόλο τα nextRow nextCol, ώστε με ένα κλικ ο χρήστης να δημιουργεί ένα χαρτί θέσεων και στη συνεχεία αν επιθυμεί να τον μετά βάλει όπως επιθυμεί.

```
$("#create").click(function () {
    if ($("#Row").val() != 0 && $("#Col").val() != 0) {
        var nextRow = 0;
         var nextCol = 0;
        for (var r = 0; r < $("#Row").val(); r++) {
            nextRow = 0;
            console.log("row:"+r);
            for (var c = 0; c < $("#Col").val(); c++) {
                 console.log("col:"+c);
                var name = "Row:" + r + "-Col:" + c;

                var rect = new fabric.Rect({
                    width: 50,
                    height: 50,
                    left: 100 + nextRow,
                    top: 100 +nextCol,
                    fill: 'grey',
                    scaleY: 0.5,
                    id:name,
                    originX: 'center',
                    originY: 'center'

                });
                nextRow = nextRow + 60;
                canvas.add(rect);
            }
            nextCol = nextCol+35;
        }
    }
});
```

Να εξηγήσω βήμα βήμα τη κάνει ο παραπάνω κώδικας. Όταν ο χρήστης κάνει κλικ στο πεδίο (το πεδίο είναι κουμπί άλλα θα μπορούσε να είναι οτιδήποτε ακόμα και παράγραφος) που έχει id create να εκτελέσει την παρακάτω συνάρτηση. Δημιουργώ δυο αριθμητές που τοποθετούν τις σειρές και στήλες στο σωστό σημείο πάνω στον καμβά ώστε να μην δημιουργείτε το κάθε στοιχείο πάνω στο άλλο. Στη συνέχεια για κάθε γραμμή που μου έχει δώσει ο χρήστης (υπάρχουν δυο πεδία input για να δώσει ο χρήστης πόσες γραμμές και στήλες θέλει από θέσεις) να μην αλλάξω γραμμή για αυτό το nextRow το μηδενίζω. Στη συνεχεία μπαίνουμε στο επόμενο for που είναι για της στήλες δημιουργώ ένα όνομα για την θέσει ανάλογα με το που βρίσκεται δημιουργώ το στοιχείο και προσθέτω αλλάζω σειρά και το κάνω αυτό μέχρι να τελειώσει η στήλη βγαίνω από το for και αλλάζω στήλη μηδενίζεται η γραμμή και όλα πάλι από την άρχει.

Επειδή έχουν εμφανιστεί στον καμβά δεν σημαίνει ότι έχουν αποθηκευτεί στην βάση δεδομένων καθώς τα δεδομένα αυτά πρέπει να σταλθούν στον controller για γίνουν οι διαδικασίες που είδαμε παραπάνω. Για αυτό το λόγο χρειάζεται να δημιουργήσω ένα κουμπί που θα αποθηκεύει τις θέσεις. Πρώτα όμως πρέπει να πάρουμε τα στοιχειά των θέσεων μέσα από τον καμβά, για να γίνει αυτό χρειάζεται να πάρω όλα τα στοιχεία από τον καμβά. Έπειτα να βρω μέσα από αυτήν τη λίστα αντικείμενων τα στοιχειά που δημιούργησα και είναι τύπου τετράγωνο, να αποθηκεύσω σε ένα προσωρινό αντικείμενο τα πεδία που θέλω να το προσθέσω σε έναν πινάκα να αδειάσω το αντικείμενο και έχω έναν πινάκα αντικείμενων με τα στοιχεία που χρειάζομαι να αποθηκεύσω (top,lef και name).Τέλος εκτελώ το Ajax είναι τύπου post, και θα τρέξει την συνάρτηση CreateTableMap που βρίσκεται στον controller θέση και του στέλνω και μια εξτρά παράμετρο το sub area id, ο τύπος δεδομένων είναι json, μετά δηλώνω τον τύπο περιεχόμενων, τα δεδομένα που θέλω

να στείλω αν γίνουν όλα σωστά τότε θα εμφανίσει ένα μήνυμα ότι αποθηκευτήκαν αλλιώς ότι απέτυχε η αποθήκευση.

```
$("#save").click(function () {
    var tmp = {};
     var seats = [];

     var canvasObjects = canvas.getObjects();
    for (obj in canvasObjects) {
      if (canvasObjects[obj].get('type') == 'rect') {
          tmp.top = canvasObjects[obj].get('top');
          tmp.left = canvasObjects[obj].get('left');
          tmp.Name = canvasObjects[obj].get('id');
          seats.push(tmp);

            tmp = {};
        }
    }
    console.log(JSON.stringify(seats));
 $.ajax(
{
    type: "post",
    url: '@Url.Action("CreateTableMap", "Seat",new { subAreaId = ViewBag.subAreaId })',
  dataType: 'json',
        contentType: 'application/json; charset=utf-8',
        data: JSON.stringify(seats),
        success: function (data) { alert("saved"); },
        error: function (data) { alert("fail to save");},
    accept: 'application/json'
    })
});
```

Για να εμφανιστούν όμως οι θέσεις στον “πελάτη” (σε χρήστη που θέλει να κάνει κράτηση) πρέπει να γίνει η αντιστροφή διαδικασία πάλι με την βοήθεια του Ajax.

```
$("document").ready(function () {
    $.ajax(
        {
            type: "get",
            url: '@Url.Action("get_data", "Seat")',
            dataType: 'json',
            data: { "SubAreaId": SubAreaId},
            contentType: 'application/json; charset=utf-8',
            success: function (data) {
                var seats = jQuery.parseJSON(JSON.stringify(data));
                for (var i = 0; i < seats.length; i++) {

                     var rect = new fabric.Rect({
                          width: 50,
                          height: 50,
                          fill: 'grey',
                          left: seats[i].x,
                          top: seats[i].y,
                          id:seats[i].id,
                          scaleY: 0.5,
                          selectable: false,
                          originX: 'center',
                          originY: 'center'

                      });
                    canvas.add(rect);
                }

            },
            error: function (data) { alert("fail to save"); },
            accept: 'application/json'
        });
});
```

Το Ajax. καλεί τη συνάρτηση get_date που βρίσκεται στον controller seat και στέλνει από πιο sub area id θέλει της θέσεις. Δέχεται τα στοιχεία σε json file και τα αντιστρέφω και εμφανίζω με την διάφορα ότι ο χρήστης δεν μπορεί να τα επιλέξει για να τα μετακίνηση. Όμως ούτε αυτό αρκεί είναι αναγκαίο να μη μπορεί να επιλέξει τις θέσεις που είναι κρατημένες.

Με αυτό το σκεπτικό χρειάζεται να καλέσουμε πάλι Ajax ώστε να δει το front-end αν υπάρχουν κρατήσεις και να κάνεις τις θέσεις αυτές με κόκκινο. Όποτε στέλνουμε στον controller τα στοιχεία που χρειάζεται για να βρει αν υπάρχουν κρατήσεις. Τα στοιχεία αυτά είναι eventId, SubAreaId, date, Duration και καλεί την συνάρτηση isFree από τον controller Reservation. Βρίσκω όλα τα αντικείμενα “rect” που δημιούργησα συγκρίνω τα id τους με αυτά που έλαβα και αν είναι ίδια του αλλάζω χρώμα.

```
$( window ).on( "load",function () {
    $.ajax(
        {
            type: "get",
            url: '@Url.Action("isFree", "Reservation")',
            dataType: 'json',
            data: { "EventId": EventId,"SubAreaId":SubAreaId, "ResDate": date,"Duration":Duration },
            contentType: 'application/json; charset=utf-8',
            success: function (data) {
                var res = jQuery.parseJSON(JSON.stringify(data));
                console.log(res);
                var canvasObjects = canvas.getObjects();
                for (obj in canvasObjects) {
                    if (canvasObjects[obj].get('type') == 'rect') {
                        for (var i = 0; i < res.length; i++) {
                            if (canvasObjects[obj].get('id') == res[i]) {
                                canvasObjects[obj].set('fill', 'red');
                            }
                        }
                    }
                }
                canvas.renderAll();
            },
            error: function (data) { alert("fail to save"); },
            accept: 'application/json'
        });
});
```

Όταν ένας χρήστης επιλεγεί μια θέση και αυτήν πρέπει να αλλάζει χρώμα. ώστε να δηλώνει ότι την έχει επιλέξει και μπορεί να διαλέξει περισσότερες από μια. Με αυτή τη λογική έχουμε έναν πινάκα με της θέσης που έχει διαλέξει και τρέχω μια συνάρτηση όταν κάνει κλικ στον καμβά. Παίρνω το στοιχείο που επέλεξε ο χρήστης βλέπω το χρώμα του αν είναι κόκκινο του πετάω μήνυμα ότι δεν μπορεί να κάνει κράτηση εκεί. Αν είναι πράσινο το κάνω πάλι γκρι με την λογική ότι άλλαξε γνώμη για της θέση και αφαιρώ τη θέση από τον πινάκα που έχω δημιουργήσει. Αν δεν είναι καμιά από τις δυο περιπτώσεις τον κάνω την θέση πράσινη και την προσθέτω στον πινάκα.
Όταν ο χρήστης είναι έτυμος να κάνει την κράτηση του πατεί τον κουμπί της κρατήσεις και στέλνει τα δεδομένα του στο back-end για να τα αποθηκεύσει στην βάση

```
$("#reservetion").click(function () {
    $.ajax(
        {
            type: "Post",
            url: '/Reservation/MakeRes/?EventId=' + EventId + '&Duration=' + Duration+ "&ResDate="+date,
            dataType: 'json',
            contentType: 'application/json; charset=utf-8',
            data: JSON.stringify(selectedSeats),
            success: function (data) { alert("saved"); },
            error: function (data) { alert("fail to save"); },
            accept: 'application/json'
        })

});
```

Παροιμία δημιουργούνται, αποθηκεύονται και εμφανίζονται και τα sub area η μόνη διάφορα είναι ότι έχουν παραπάνω στοιχειά που είναι το πλάτος και το ύψος καθώς ο χρήστης με το ρόλο venue μπορεί να βάλει το μέγεθος που θέλει να φαίνεται πάνω στον καμβά.

```
var canvasObjects = canvas.getObjects();
        for (obj in canvasObjects) {
            if (canvasObjects[obj].get('type') == 'group') {
                tmp.Top = canvasObjects[obj].get('top');
                tmp.Left = canvasObjects[obj].get('left');
                tmp.AreaName = canvasObjects[obj].get('id');
                tmp.Rotate = canvasObjects[obj].get('angle');
                tmp.Width = canvasObjects[obj].getScaledWidth();
                tmp.Height = canvasObjects[obj].getScaledHeight();;
            }
            if (canvasObjects[obj].get('type') == 'text') {
                tmp.AreaName = canvasObjects[obj].get('text');
                SubAreas.push(tmp);
                tmp = {};
            }
}
      $.ajax(
```

4.6.3 Events και calendar

Όταν ένας χρήστης θέλει να δημιουργήσει ένα event του ανοίγει μια σελίδα με ένα ημερολόγιο διαλέγει την ήμερα που θέλει να ξεκινάει το event. Για να προσέθεσα μια νέα βιβλιοθήκη java script και μια για το bootstrap για να έχω modal. Το modal είναι ένα παράθυρο που ανοίγει μέσα στην σελίδα με fade-in animation. Οι βιβλιοθήκη είναι από το cloudflare.com η fullcalendar για να δημιουργηθεί το πλαίσιο του ημερολογίου κάνω ένα div με id calendar. Για να δημιουργηθεί και να πάρει τα events που υπάρχουν είδη και να τα προσθέσει πάνω στο ημερολόγια πρέπει να πάρω τα events από τον controller μεσώ Ajax και να τα αποθηκεύσω σε έναν πινάκα που είναι global μεταβλητή. Επόμενο βήμα είναι να καλέσω την συνάρτηση που γεμίζει το ημερολόγια με τα events. Έπειτα “διαγράφω” τα παλιό ημερολόγιο και σχεδιάζω το νέο, το δίνω την σημερινή ημερομηνία και των πινάκα που δημιουργούσα καθώς στον πινάκα αντικείμενων των έχω δημιουργήσει έτσι ώστε να αναγνωρίζει την αρχική ημερομηνία δηλώνοντας το με start.

```
success: function (data) {
    $.each(data, function (i, v) {
        events.push({
            id: v.id,
            title: v.name,
            start: moment(v.startDateTime),
            end: v.endTime != null ? moment(v.endTime) : null,
            EventTypeId: v.eventTypeId,
            EventType: v.eventType
        });
    })
    GenerateCalender(events);
},
```

Συνεχίζοντας κάλω την συνάρτηση GenerateCalender για να δημιουργηθεί το ημερολόγια με όλα τα events του venue.

```
function GenerateCalender(events) {
    $('#calender').fullCalendar('destroy');
    $('#calender').fullCalendar({
        contentHeight: 400,
        defaultDate: new Date(),
        timeFormat: 'h(:mm)a',
        header: {
            left: 'prev,next today',
            center: 'Name',
            right: 'month,basicWeek,basicDay,agenda'
        },
        eventLimit: true,
        eventColor: '#378006',
        events: events,
```

Ο χρήστης έχει 2 επιλογές να κάνει κλικ σε κενό ή να κάνει κλικ πάνω σε event. Για αυτό το λόγο δημιουργώ ένα κενό modal και θα το γεμίσω ανάλογα με το κλικ που θα κάνει ο χρήστης.

```
<div id="myModal" class="modal fade" role="dialog">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal">&times;</button>
                <h4 class="modal-title"><span id="eventTitle"></span></h4>
            </div>
            <div class="modal-body">
                <p id="pDetails"></p>
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
            </div>
        </div>
    </div>
</div>
```

Όταν δημιούργησα το ημερολόγιο του έδωσα την εντολή όταν κάνουν κλικ σε event δημιουργεί ένα modal που ο χρήστης. βλέπει της λεπτομερείς του event και αν θέλει μπορεί να το διαγράψει ή να διαγράψει όλοι την οικογένεια από events.
Η διαγραφή γίνεται μεσώ Ajax στέλνω το id του event στον controller και αυτός κάνει τα υπόλοιπα.

Όταν ο χρήστης κάνει κλικ σε άδειο κομμάτι τότε πρέπει να μπορεί να δημιουργήσει event το ανοίγει ένα modal που δημιουργείται δυναμικά η διαφορά εδώ είναι ότι χρειάζεται να ανοίξει ένα ακόμα modal ώστε να μπορεί να επιλέξει αν θέλει να επαναλαμβάνεται το event.

```
dayClick: function (date, jsEvent, view) {
    $('#myModal #eventCreate').text("Create Event");
    var $description = $('<div class="row"><div class="col-md-4">' +
        ' <div class="form-group">' +
        '<b>Name:</b> <input id="Name"/></div >' +
        ' <div class="form-group">' +
        '<input id="Date" value=' + date.format() + ' type="hidden"/>' +
        '<b>Start Time:</b> <input id="StartTime" type="time"/>' +
        ' <div class="form-group">' +
        '<b>End Time:</b> <input id="EndTime" type="time"/>' +
        ' <div class="form-group">' +
        '<b>Event Type:</b> <select id="EventTypes" value="Restaurant"> </select>' +
        ' <div class="form-group">' +
        '<br><input type="submit" value="Repeat Time" id="RepeatTime" class="btn btn-primary" /></div>' +
        ' <div class="form-group">' +
        '<input type="submit" value="Create" id="Create" class="btn btn-primary" /> </div>' +
        '</div ></div >'
    );

    $('#myModal #pDetails').empty().html($description);
    $('#myModal').modal();
}
```

Το modal για την επανάληψη υπάρχει με HTML και όχι java script καθώς το κενό modal που άλλαζα είναι ήδη ανοικτό και πρέπει να ανοίξω ένα άλλο modal για να μην κλείνει το προηγούμενο. Όταν διαλέξει το ρυθμό επανάληψης το σύστημα αποθηκεύει αυτές τις επιλογές προσωρινά.

```
$("#saveRepeatTime").click(function () {
    repeat.NumOfRepeat = $("#numOfRepeat").val();
    repeat.SelectRepeat = $("#selectRepeat").val();
    repeat.Su = $("#su").is(":checked");
    repeat.M = $("#m").is(":checked");
    repeat.Tu = $("#tu").is(":checked");
    repeat.W = $("#w").is(":checked");
    repeat.Th = $("#th").is(":checked");
    repeat.F = $("#f").is(":checked");
    repeat.Sa = $("#sa").is(":checked");
    repeat.UntilDate = $("#untilDate").val();
    repeat.AfterNumTimes = $("#afterNumTimes").val();
     console.log($("#afterNumTimes").val());
    if ($("#untilDate").val() != "" && $("#afterNumTimes").val() != "") {
        alert("you cant have both until and after");
        $("#untilDate").val() = "";
        $("#afterNumTimes").val() = "";
        repeat = {};
    } else if ($("#untilDate").val() == "" && $("#afterNumTimes").val() == "") {
        alert("you need to put Until or After x occurances");
        $("#untilDate").val() = "";
        $("#afterNumTimes").val() = "";
        repeat = {};
    }
    if (repeat.AfterNumTimes != "" && repeat.UntilDate == "") {
        repeat.UntilDate = new Date();
    }
   $('#myModalRepeatTime').modal('toggle')
});
```

Όταν ο χρήστης είναι έτοιμος πατάει το κουμπί δημιουργία. Εκτελείται το Ajax στέλνει τα δεδομένα στον controller και τα αποθηκεύει στη βάση δεδομένων. Άλλα πριν σταλθούν τα δεδομένα πρέπει να αλλάξουμε λίγο της ημερομηνίες ώστε ο controller να καταβάλει ότι είναι ημερομηνίες. Επειδή η java script και το .Net core διαφέρουν στο πως διαχειρίζονται τις ημερομηνίες (παντού υπάρχουν validate για να ελέγχουν την ορθότητα των στοιχείων).

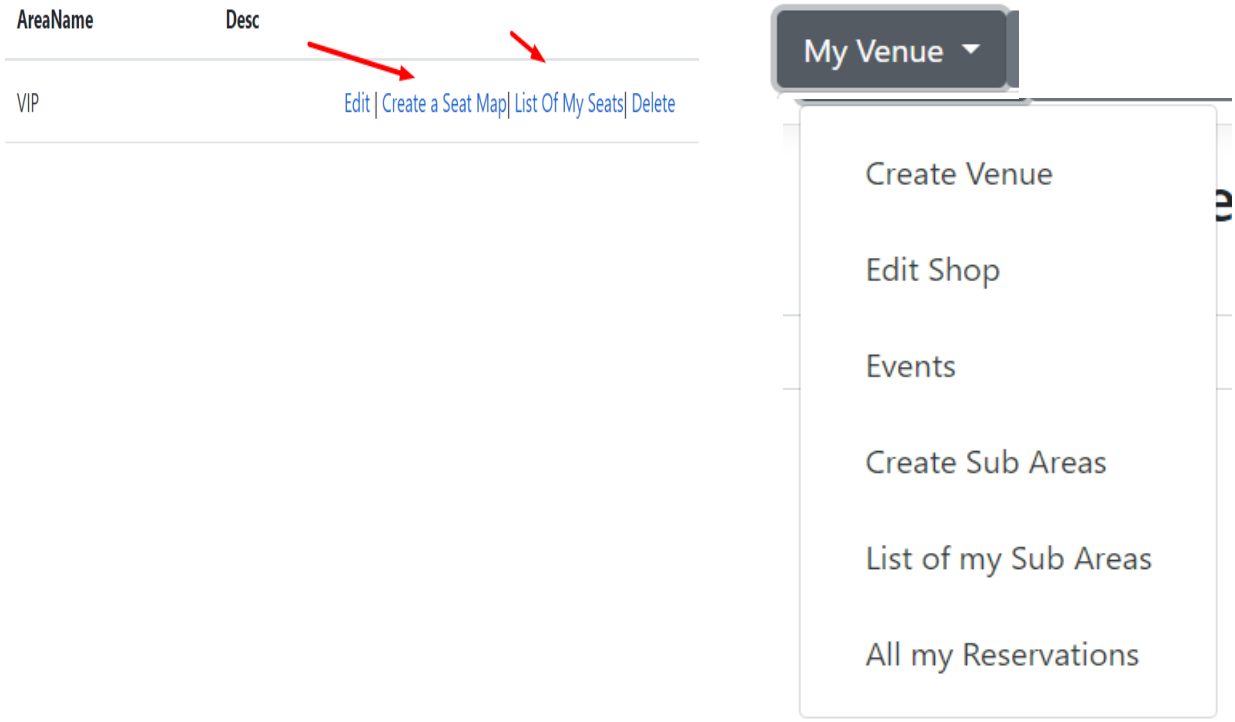
```
var end = new Date($("#Date").val() + 'T' + $("#EndTime").val());
var start = new Date($("#Date").val() + 'T' + $("#StartTime").val());
```

Για του χρήστης που δεν έχουν ρόλο venue υπάρχει σελίδα που μπορούν να δουν τα events σε μορφή ημερολογίου άλλα μπορούν μόνο να κάνουν κράτηση.

Κεφαλαιο 5

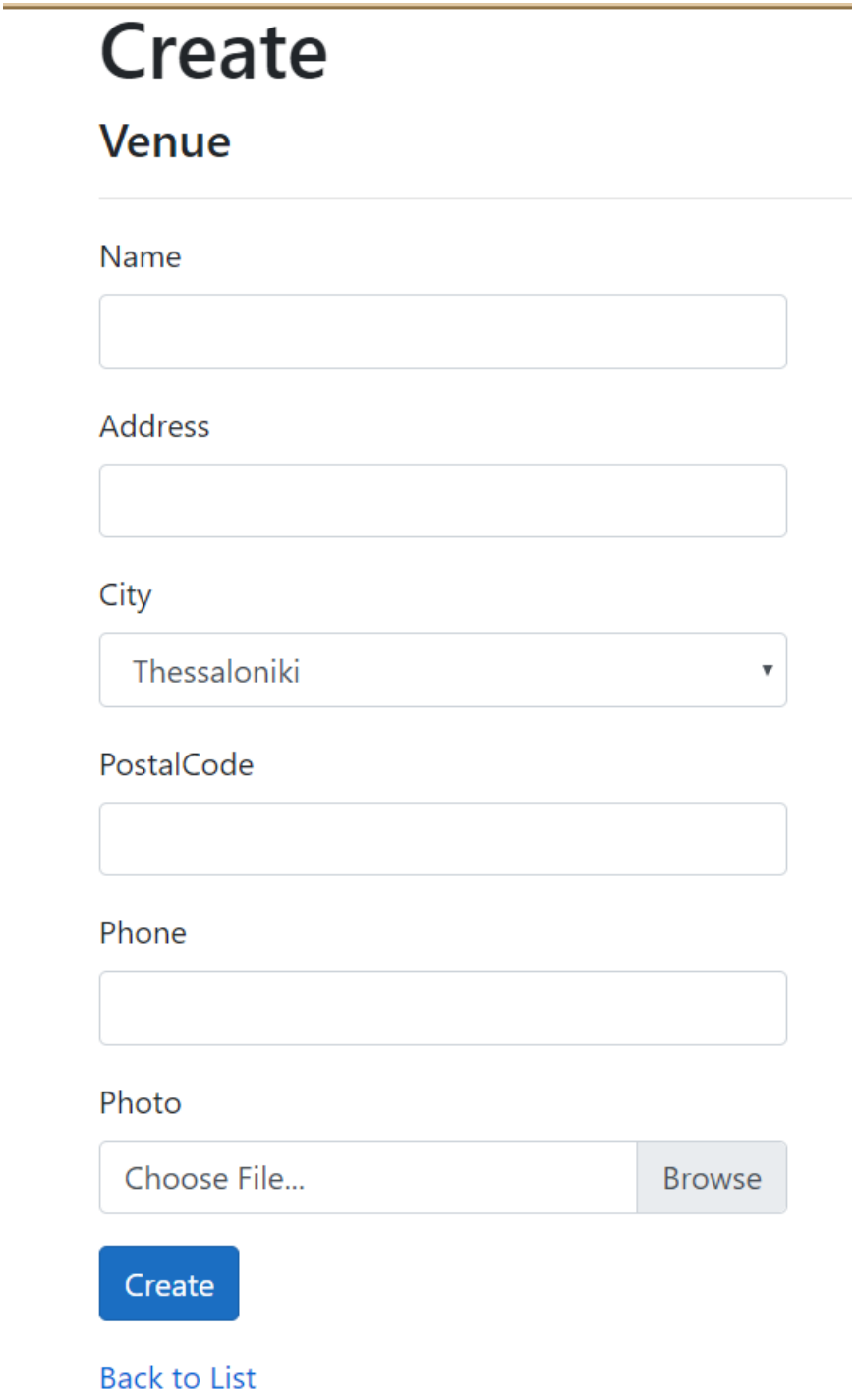
5.1 Επιλογές στο Venue menu panel

Οι επιλογές στο πάνελ είναι η δημιουργία του venue, η επεξεργασία του, η δημιουργία events, η δημιουργία sub area, η λίστα των sub area μέσα στη λίστα υπάρχει η επιλογή για να δημιουργήσει τη θέσης ή να δει την λίστα των θέσεων του sub area και όλες οι κρατήσεις



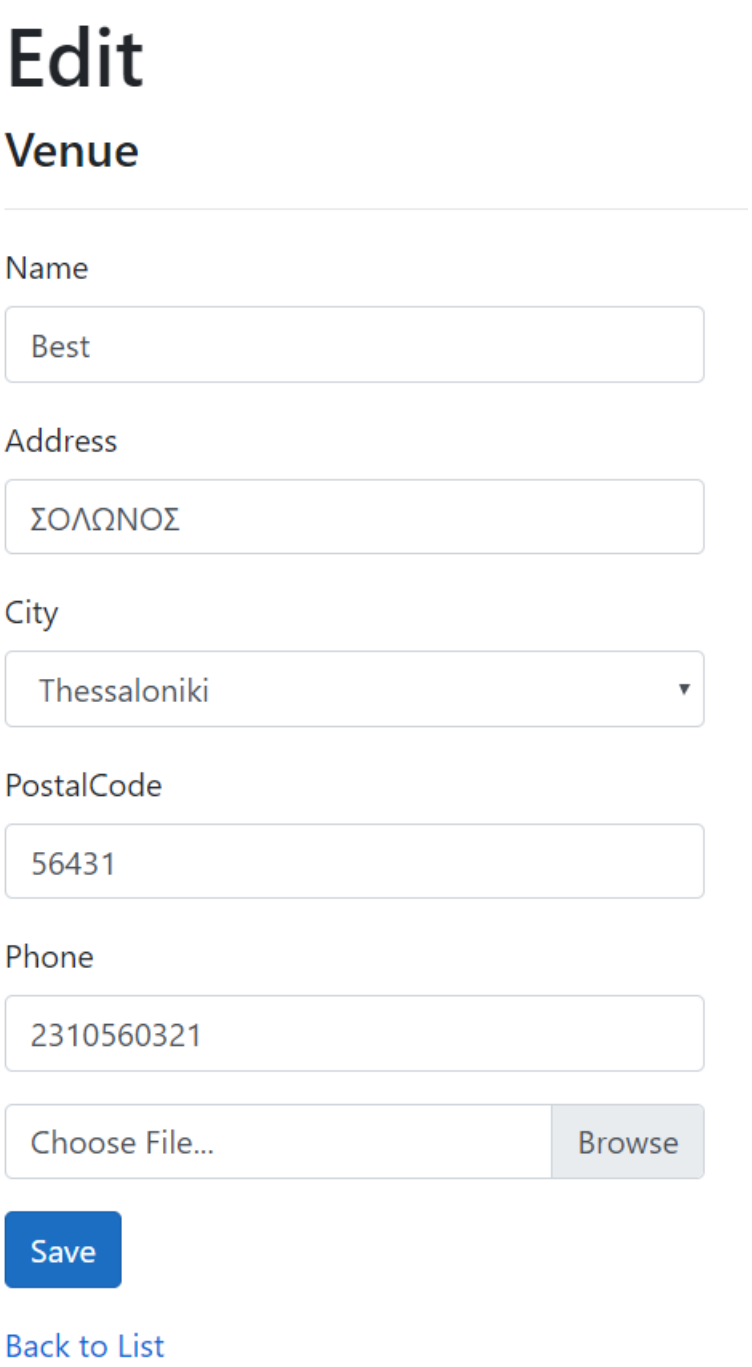
5.1.1 Δημιουργία venue

Για να δημιουργήσει κάποιος το venue πρέπει να διαλέξει την πρώτη επιλογή αυτόν θα τον μεταφέρει σε μια σελίδα που θα έχει μια φόρμα και θα πρέπει να συμπληρώσει τα στοιχεία που του ζητάει για να γίνει ή δημιουργία. Για το πεδίο πόλη υπάρχει μια λίστα από τις μεγαλύτερες πόλης της Ελλάδος.



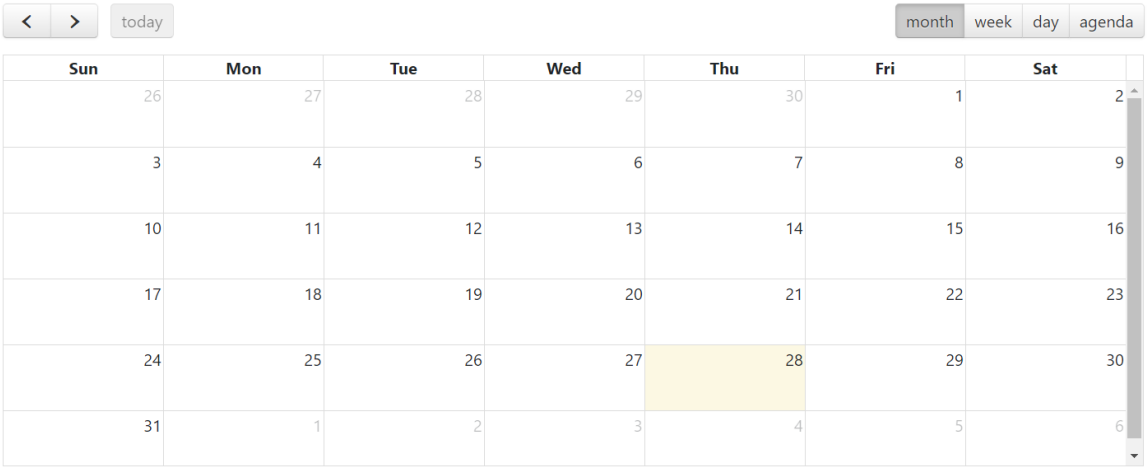
5.1.2 Επεξεργασία venue

Για την επεξεργασία του venue είναι ο χρήστης έκανε κάτι λάθος ή άλλαξε ονομασία ή κάποιο άλλο στοιχείο. Η διάφορα με την δημιουργία είναι ότι στην φόρμα αυτή τα στοιχεία που είχε βάλει ο χρήστης υπάρχουν πάνω στη φόρμα.

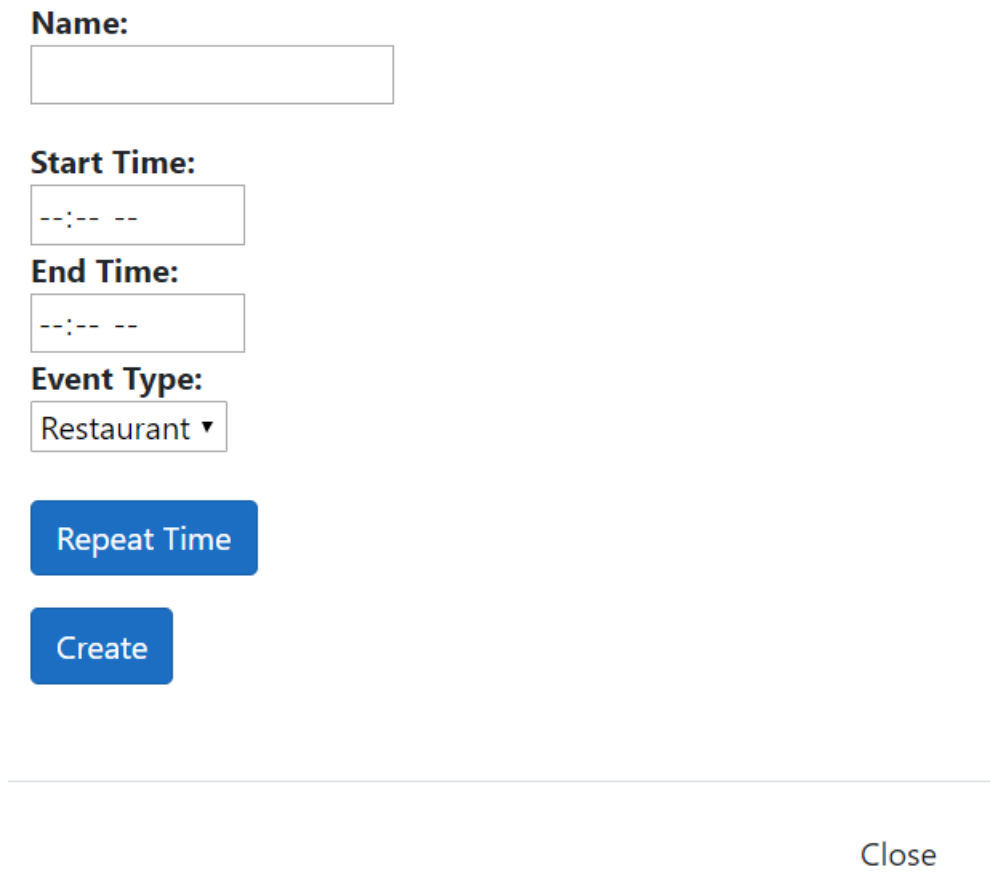


5.1.3 Δημιουργία event

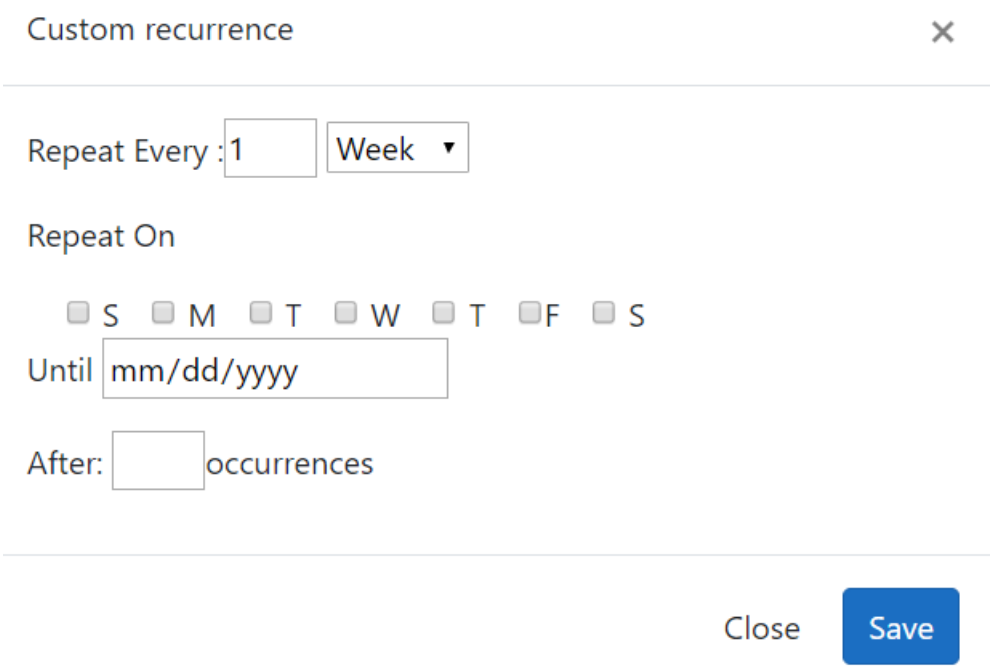
Πατάττοντας την τρίτη επιλογή μπορούμε να δημιουργήσουμε τα events. Θα εμφανιστεί μια σελίδα που υπάρχει ένα ημερολόγιο. Κάνοντας κλικ σε μια ημερομηνία που είναι ίση ή μεγαλύτερη από σήμερα ανοίγει ένα παράθυρο που περιέχει την φόρμα δημιουργίας του event.



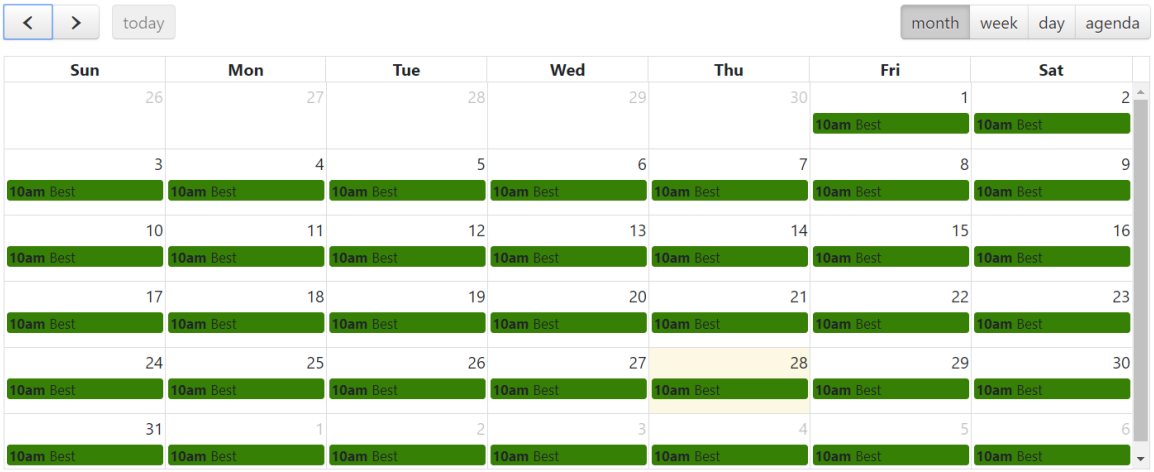
Σε αυτήν τη φόρμα ο χρήστης βάζει όνομα, ώρα αρχής, τέλους και τον τύπο του event. Έπειτα μπορεί να επιλέξει αν θέλει να επαναλαμβάνεται το event ή να πατήσει δημιουργία και να εμφανιστεί μόνο ένα event.



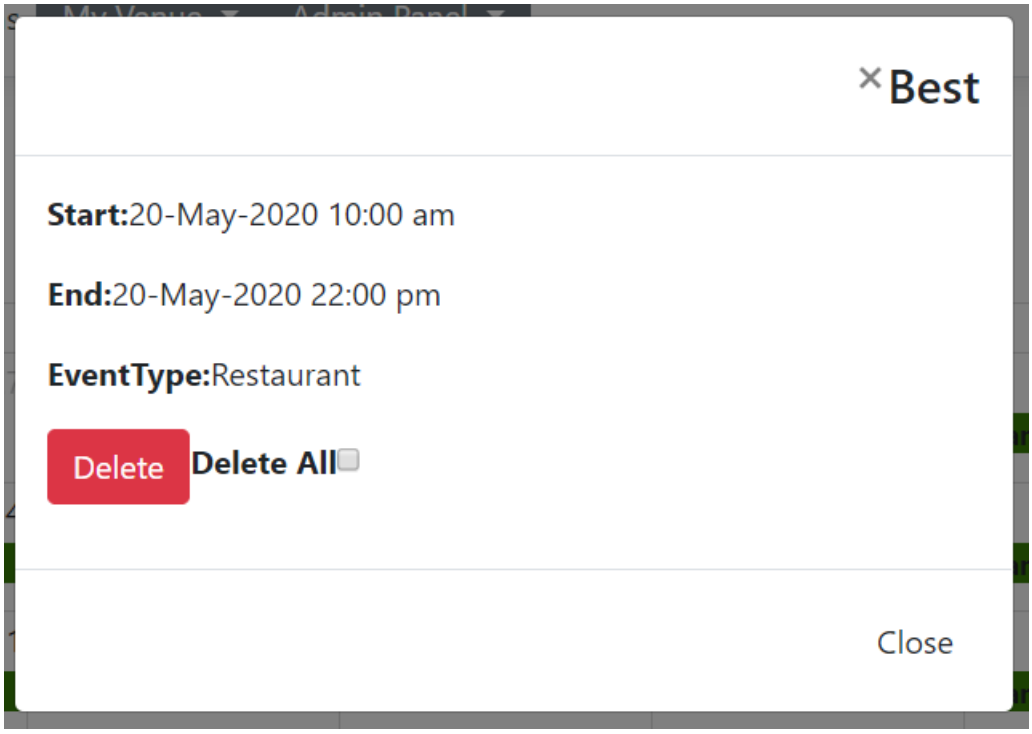
Αν επιλέξει να βάλει επανάληψη τότε θα εμφανιστεί ένα νέο παράθυρο. Με τις επιλογές: κάθε πότε να επαναλαμβάνεται “αριθμό” και αν αναφέρεται σε εβδομάδες ή μήνες ή μέρες. Δουλεύει όπως και στο “google calendar”.



Όταν ανανέωση την σελίδα θα εμφανιστούν τα event που έχει δημιουργήσει.

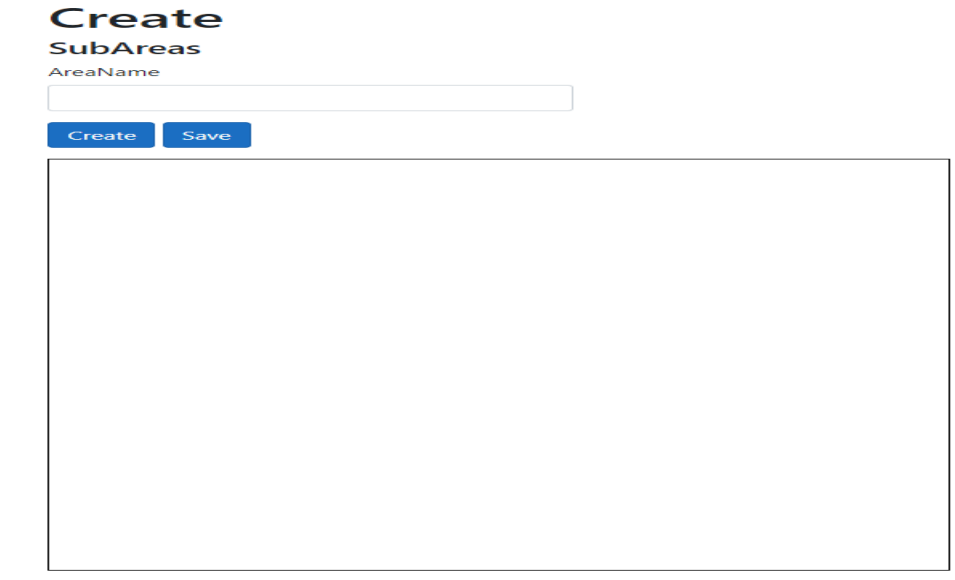


Κάνοντας κλικ πάνω σε ένα πράσινο ανοίγουν οι λεπτομερείς του event και μπορεί να το διαγράψει ή να διαγράψει όλα τα event που ανήκουν σε αυτήν την οικογένεια.

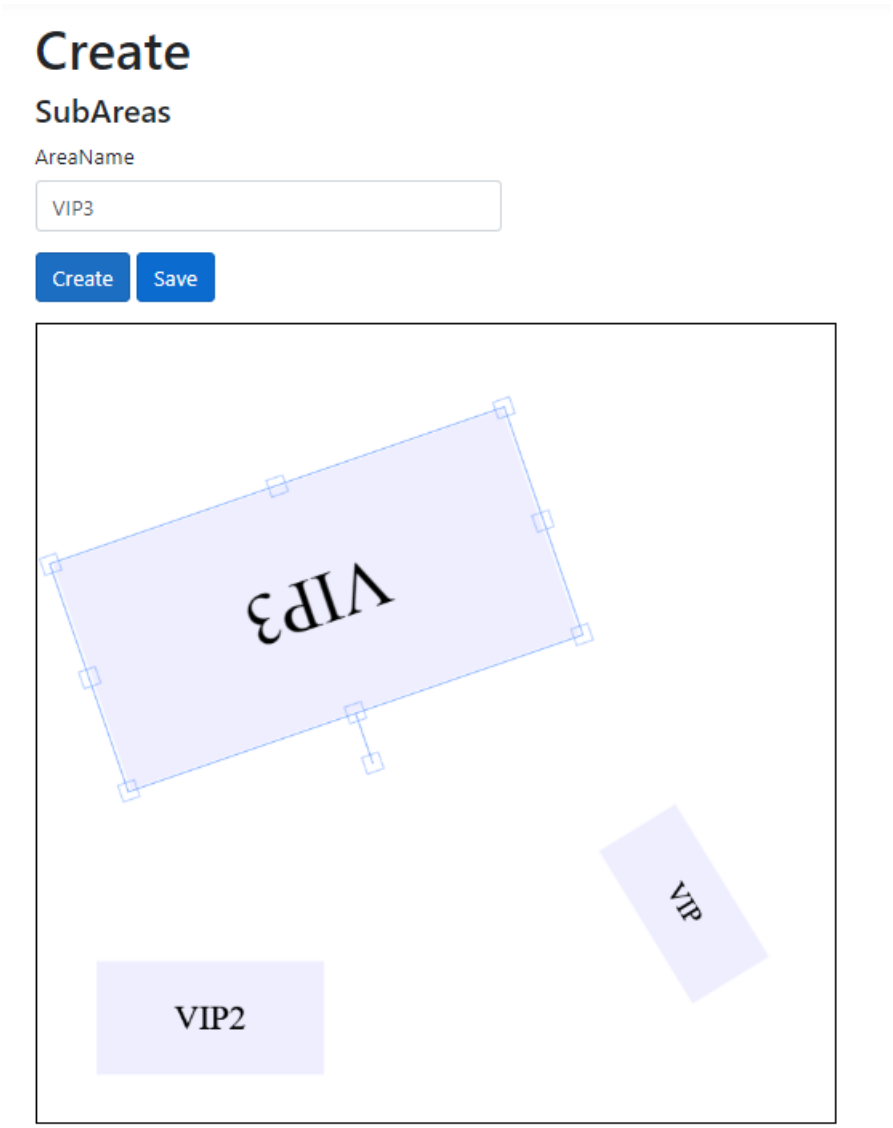


5.1.4 Δημιουργία Sub Area

Είναι η τετάρτη επιλογή και ανοίγει μια σελίδα που ο χρήστη βάζει όνομα και πατάει το κουμπί δημιουργία, μπορεί να δημιουργήσει όσα sub areas θέλει.

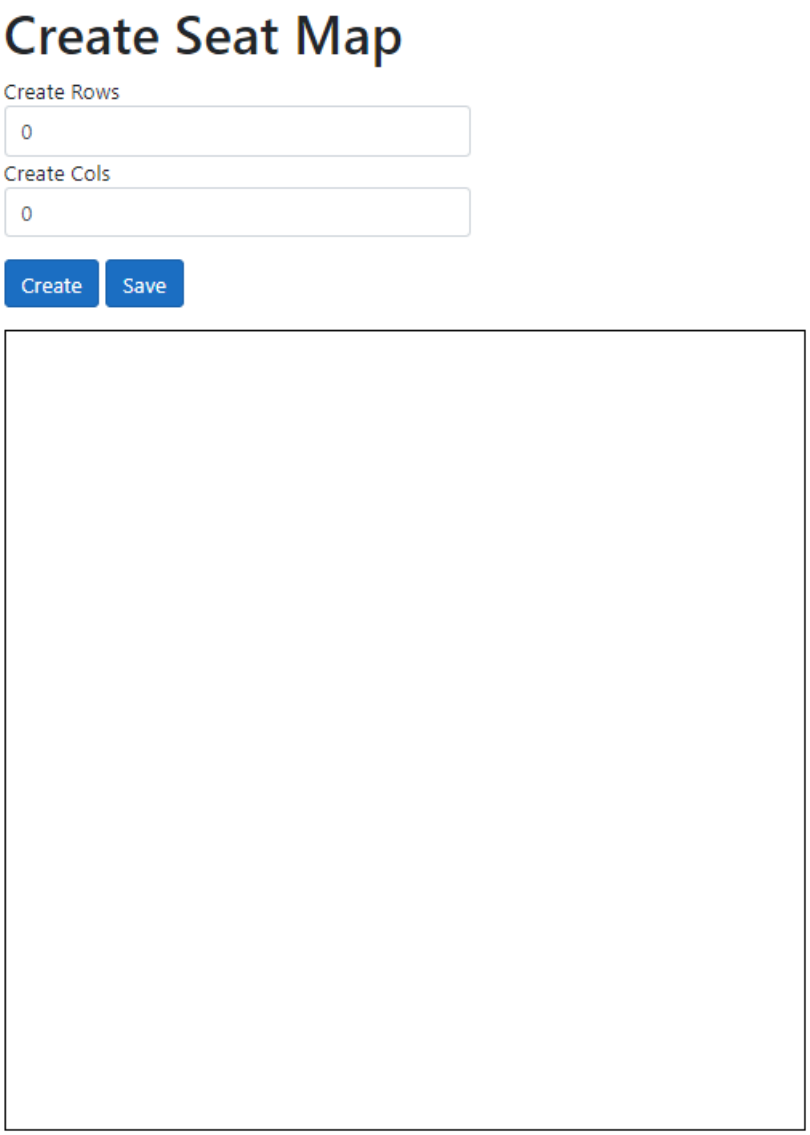


Όταν δημιουργήσει ένα μπορεί να το φέρει στο μέγεθος που θέλει. και να το βάλει σε οποία θέση θέλει μέσα στον καμβά.

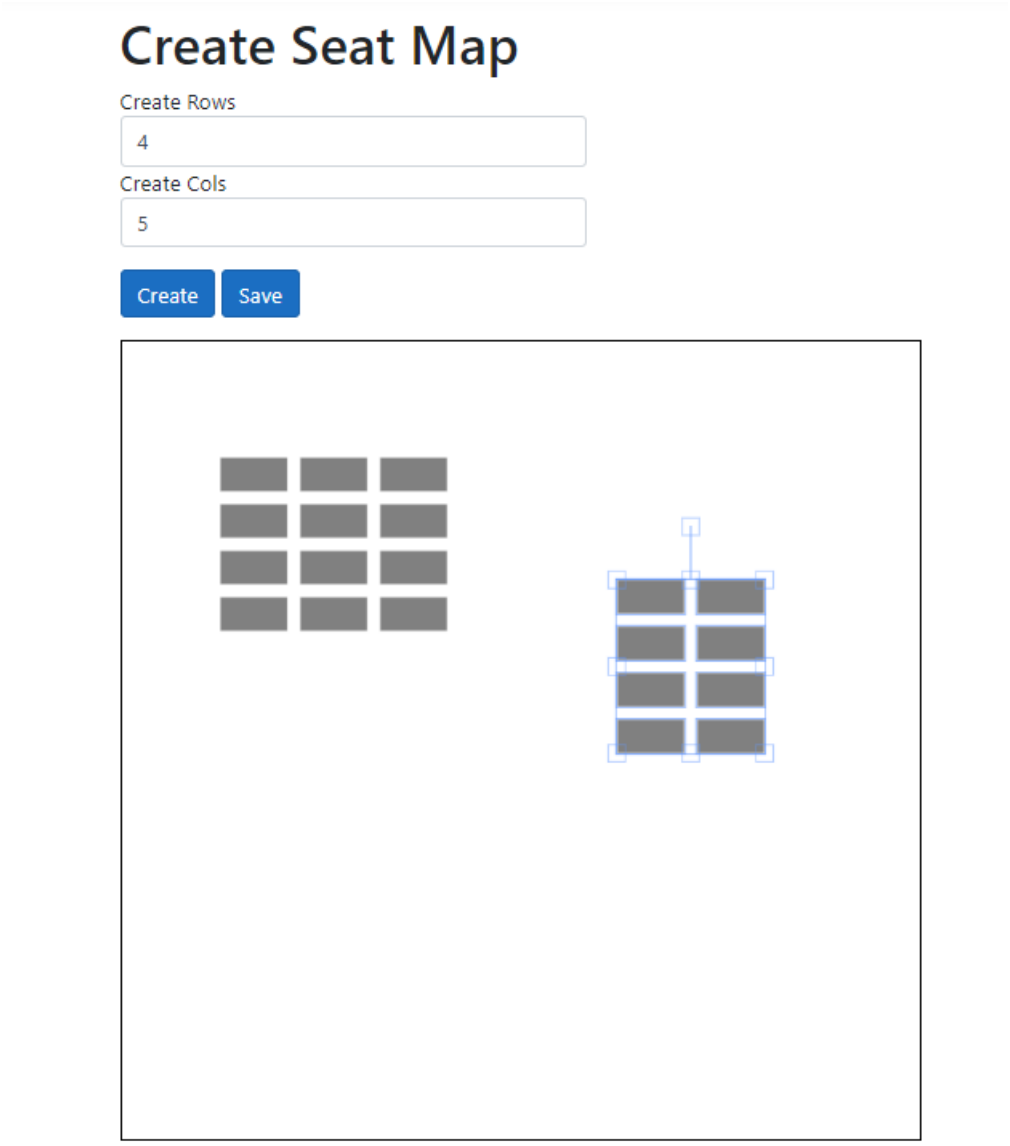


5.1.5 Δημιουργία θέσεων

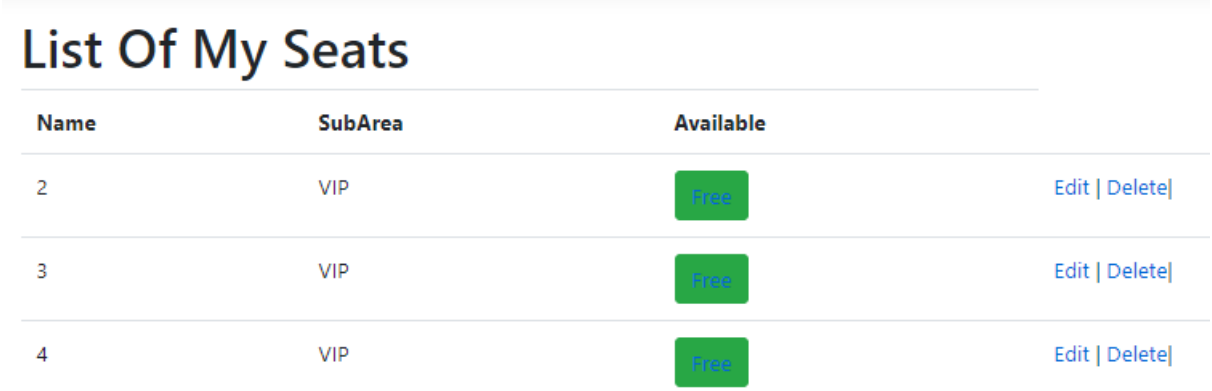
Πηγαίνοντας στη λίστα των sub area διαλέγει σε ποια περιοχή θέλει. να δημιουργήσει θέσεις. Θα εμφανιστεί μια σελίδα παρόμοια με την προηγούμενη η διαφορά είναι ότι θα επιλέξει πόσες στήλες και γραμμές από θέσεις. θέλει να έχει.



Αφού δημιουργήσει τις θέσεις μπορεί να τις βάλει σε οποίο σημείο στον καμβά θέλει επιλέγοντας τις θέσεις μια μια ή πολλές μαζί σαν ομάδα, όπως μπορεί κάποιος να επιλέξει πολλαπλά στοιχειά πάνω από την επιφάνεια εργασίας του.



Αν επιλέξει να δει την λίστα από θέσεις που έχει ένα sub area. Μπορεί να διαγράψει θέσεις ή να κάνει μια θέση μη διαθέσιμη για σήμερα.

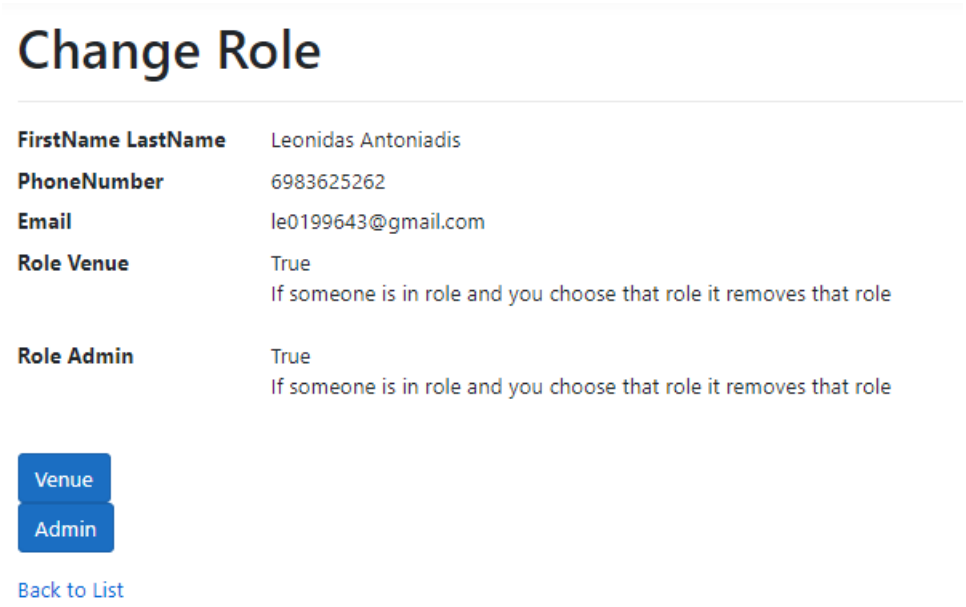


5.1.6 Εμφάνιση κρατήσεων

Η τελευταία επιλογή είναι η εμφανίσει όλων των κρατήσεων που μπορεί να την ταξινόμηση ανάλογα το στοιχείο που επιθυμεί.

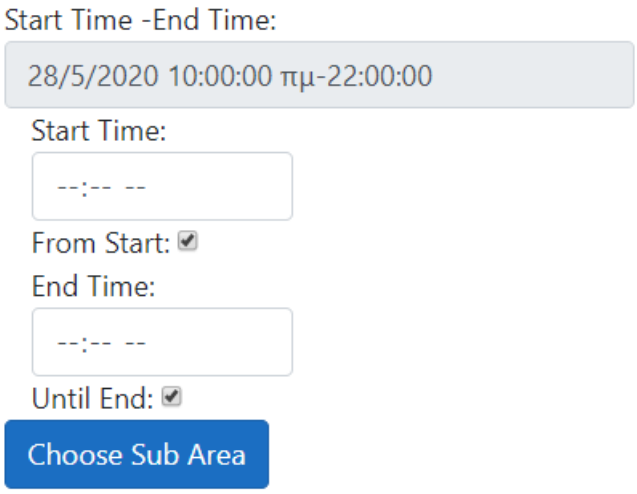
5.2 Πως αλλάζουμε ρόλους

Ρόλους μπορεί μόνο οι διαχειριστές να αλλάξουν μέσα από το δικό τους πάνελ. Εμφανίζουν όλους τους χρήστες διαλέγουν την επιλογή αλλαγή ρολού. Βλέπουν αν ο χρήστης είναι σε αυτόν το ρόλο και αν θέλουν τον αναφέρουνε αν δεν είναι τον προσθέτουν.

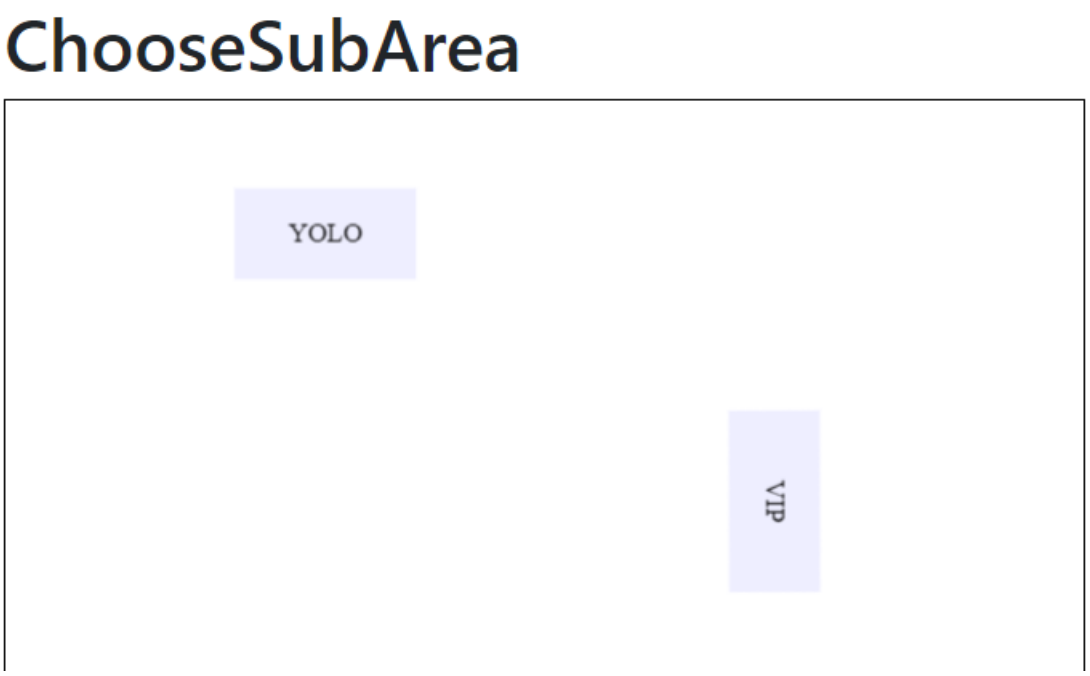


5.3 Πως κάνουμε κράτηση

Στην αρχική σελίδα γράφει ο χρήστης την πόλη που θέλει να βρει ένα event για να κάνει κράτηση σήμερα. (μπορεί να κάνει κράτηση και για άλλη μέρα αν πάει στο venue λεπτομέρειες events) και θα πατήσει κράτηση. Θα ανοίξει μια σελίδα και μια φόρμα να βάλει τη ώρα θα έρθει και τη ώρα θα φύγει (μπορεί να κάνει κράτηση. από την αρχή έως το τέλος).



Στη συνεχεία επιλεγεί το sub area που θέλει να κάνει κράτηση.



Και τέλος της θέσης που θέλει να κάνει κρατήσει και είναι διαθέσιμες και πατάει κράτηση.

Ο χρήστης κάνοντας κλικ στο My Reservations πάνω στο μενού μπορεί να δει τις κρατήσεις του και τις λεπτομερείς τους.

Αξιολόγηση

Σε ένα δείγμα τριάντα ατόμων που δοκιμάσαν την εφαρμογή απάντησαν σε ένα μικρο ερωτηματολόγιο για την εμπειρία τους που δοκιμάσαν τη εφαρμογή.

1. Πόσο τους άρεσε (εμφανισιακά) η εφαρμογή. Απάντησαν :
    1. Καθόλου : 0
    2. Λίγο : 0
    3. Μέτρια : 5
    4. Πόλη : 20
    5. Πάρα Πόλη : 5

2. Πόσο καλή είναι η λειτουργικότατα της εφαρμογής. Απάντησαν :
    1. Καθόλου : 0
    2. Λίγο : 0
    3. Μέτρια : 3
    4. Πόλη : 17
    5. Πάρα Πόλη : 10

3. Πόσο ικανοποιούμενος είναι από την εφαρμογή. Απάντησαν :
    1. Καθόλου : 0
    2. Λίγο : 1
    3. Μέτρια : 2
    4. Πόλη : 20
    5. Πάρα Πόλη : 7

4. Αν θα προτιμήσουν αυτήν την εφαρμογή από άλλες παραπλήσιες. Απάντησαν :
    1. Καθόλου : 0
    2. Λίγο : 3
    3. Μέτρια : 2
    4. Πόλη : 18
    5. Πάρα Πόλη : 7

5. Αν θα άλλαζαν κάτι στην εφαρμογή. Απάντησαν :
    1. Όχι : 17
    2. Ναι : 13

Συμπεράσματα

Μέρος του στόχου αυτής της εργασίας ήταν να παρέχει τεχνική επισκόπηση της δημιουργίας μιας ιστοσελίδας η οποία παρέχει σε πελάτες υπηρεσίες να δημιουργούν τον χώρο τους και να προσφέρουν υπηρεσίες κρατήσεων σε άλλους χρήστες. Από την αξιολόγηση παρατηρείται ότι υπάρχουν πολλές διορθώσεις – αναβαθμίσεις που μπορούν να γίνουν, άλλα σε μια τέτοια εφαρμογή τα εργαλεία, της τεχνικές και την αρχιτεκτονικές λογισμικού που μπορεί να προσθέσει κάποιος είναι ατελείωτα. Το μέλλον αυτής της εργασίας είναι να γίνουν οι αναγκαίες αναβαθμίσεις και να δημιουργηθεί για android εφαρμογή μόνο για τους χρήστες. που θέλουν να κάνουν κράτηση. Μια ενδιαφέρον αναβαθμίσει είναι να μπει Qr-code ώστε να διαβάσει της κράτηση. από το κινητό του χρήστη.

Βιβλιογραφία

https://groups.google.com/forum/#!topic/fabricjs/SRHqJmLCXnY

https://docs.microsoft.com/en-us/archive/msdn-magazine/2016/may/asp-net-writing-clean-code-in-asp-net-core-with-dependency-injection

http://fabricjs.com/fabric-intro-part-1

https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_panels_contextual&stacked=h

https://el.wikipedia.org/wiki/HTML

http://dnhost.gr/kb/article/AA-00274/0/Τι-είναι-η-MySQL-βάση- δεδομένων.html

https://el.wikipedia.org/wiki/ASP.NET
http://el.wikipedia.org/wiki/CSS

https://en.wikipedia.org/wiki/Ajax_(programming)

http://el.wikipedia.org/wiki/JavaScript

https://docs.microsoft.com/el-gr/aspnet/core/?view=aspnetcore-3.0

https://stackoverflow.com/questions/12998739/how-to-check-if-datetime-now-is-between-two-given-datetimes-for-time-part-only/12998855

https://www.youtube.com/watch?v=aoxEJii70_I

https://www.youtube.com/user/kudvenkat/videos

https://www.youtube.com/watch?v=miR_GymdSaw