ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

"ΑΝΑΠΤΥΞΗ ΣΥΣΤΗΜΑΤΟΣ ΔΙΑΧΕΙΡΙΣΗΣ ΚΑΙ ΑΝΑΖΗΤΗΣΗΣ ΑΘΛΗΤΙΚΩΝ EVENTS ΜΕ ΧΡΗΣΗ .NET FRAMEWORK (ASP.NET - MVC και C#)"

ΟΝΟΜΑΤΕΠΩΝΥΜΟ ΦΟΙΤΗΤΡΙΑΣ: Σκούταρη Ισμήνη

ΑΜ: 2225

ΕΠΙΒΛΕΠΩΝ: Σωτήρης Χριστοδούλου

ΑΝΤΙΡΡΙΟ 2017-2018

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή

Αντίρριο, 2018

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. , Υπογραφή

2. , Υπογραφή

3. , Υπογραφή

Περίληψη

Σκοπός της παρούσας πτυχιακής εργασίας ήταν η ανάπτυξη ενός συστήματος το οποίο βοηθάει στην καταχώρηση και την αναζήτηση αθλητικών γεγονότων τόσο στην Ελλάδα όσο και σε άλλες χώρες. Στόχος ήταν να γίνει όσο το δυνατόν πιο εύχρηστη η διαχείριση της εφαρμογής τόσο για αυτούς που θα την διαχειρίζονται όσο και για αυτούς οι οποίοι απλά θέλουν να ενημερωθούν για τα events.

Το σύστημα χωρίζεται σε δύο μέρη. Το πρώτο κομμάτι είναι για τους διαχειριστές, στο οποίο καλούνται να καταχωρήσουν ορισμένες πληροφορίες σχετικά με τον αγώνα που θέλουν να προσθέσουν (όπως π.χ. το σημείο έναρξης και το σημείο τερματισμού, την ημ/νία διεξαγωγής, την απόσταση, κάποιες φωτογραφίες (προαιρετικό) καθώς και διάφορες άλλες πληροφορίες που αφορούν τον συγκεκριμένο αγώνα (βλ. Κεφ. 2)). Το δεύτερο κομμάτι αφορά τους επισκέπτες της εφαρμογής όπου θέλουν να ενημερωθούν σχετικά με τα events. Στο front-end έχουμε φροντίσει να τους παρέχουμε όσες περισσότερες ευκολίες γίνεται σχετικά με τον διαχωρισμό και την εύρεση των αγώνων, και για το λόγο αυτόν έχουν προστεθεί διάφορα φίλτρα όπως για παράδειγμα μπορούν να ορίζουν την απόσταση (σε χλμ), την ημ/νία διεξαγωγής των αγώνων που τους ενδιαφέρει, το τύπο του αγώνα, μπορούν ακόμη και εστιάζοντας πάνω στον χάρτη να επικεντρωθούν σε μία συγκεκριμένη περιοχή, και διάφορες άλλες λειτουργίες.

Στα πλαίσια της ανάπτυξης του συγκεκριμένου συστήματος έγινε χρήση της γλώσσας C# (βλ. Κεφ. 1 και 3) για την ανάπτυξη όλων των αλγορίθμων που χρησιμοποιήθηκαν (για τους οποίους θα μιλήσουμε εκτενέστερα στα ακόλουθα κεφάλαια). Επειδή όμως όταν μιλάμε για ένα τόσο μεγάλο σύστημα είναι λογικό μία μόνο γλώσσα προγραμματισμού να μην μπορεί να μας παρέχει όλα όσα θέλουμε προκειμένου να γίνει ένα ολοκληρωμένο σύστημα, όμορφο και εύχρηστο. Για τον λόγο αυτόν λοιπόν χρησιμοποιήθηκαν και κάποιες άλλες τεχνολογίες όπως η γλώσσα HTML, CSS, JavaScript καθώς και κάποιες τεχνολογίες όπως η KnockOut και τα Regural Expresions. Ο σωστός συνδυασμός λοιπόν όλων αυτών των λειτουργιών μας έδωσε το επιθυμητό αποτέλεσμα.

Ουσιαστικά πρόκειται για ένα σύστημα το οποίο κάνει πιο εύκολη την εύρεση αθλητικών γεγονότων σε ορισμένες ομάδες ανθρώπων που ασχολούνται συστηματικά με αυτόν τον τομέα καθώς περιλαμβάνει όλες τις απαραίτητες πληροφορίες που χρειάζονται και είναι πάρα πολύ απλό στην χρήση του τόσο από την μεριά των απλών χρηστών όσο και από την μεριά των διαχειριστών του συστήματος. Μπορεί εύκολα κάποιος να βρει αυτό που αναζητά καθώς είναι όλα μαζεμένα σε ένα σημείο και μπορεί να ανατρέξει τόσο σε events που γίνονται εντός Ελλάδας όσο και events που διοργανώνονται και σε χώρες του εξωτερικού.

Abstract

The aim of this thesis was to develop a system that helps to entry and search for sporting events both in Greece and other countries. The goal was to make administration of the application as user-friendly as possible for both those who manage it and those who just want to know about the events.

The system is divided into two parts. The first part is for the managers, who are asked to enter some information about the race they want to add (such as the starting point and the finish point, the date, the distance, some photos (optional) as well as various other information relating to the particular event (see Chapter 2)). The second part concerns the visitors of the application where they want to be informed about the events. At the front end, we've made sure to provide them with as much convenience as possible for separating and finding races, and for that reason various filters have been added, for example, they can specify the distance (in km), the date of the race of the event they are interested in, the type of match, they can even focus on the map to focus on a particular area, and various other functions.

The backend implementation is based on programming language C# (see Chapters 1 and 3). The C# language was used to develop all the algorithms used (for which we will talk more extensively in the following chapters). But when we talk about such a large system, it makes sense that only one programming language can not provide us with everything we want to become a complete system, beautiful and easy to use. For this reason, other technologies such as HTML, CSS, JavaScript, and some technologies such as KnockOut and Regular Expressions have been used. The right combination of all these functions gave us the desired result.

This is basically a system that makes it easier to find sporting events in some groups of people who are systematically engaged in this area as it includes all the necessary information they need and is too simple to use by both simple users and from the part of the system administrators. It is easy for someone to find what he is looking for as all sporting events are gathered together in the same place and can refer to events taking place inside Greece as well as events organized in foreign countries.

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω θερμά την οικογένεια μου, που καθ' όλη τη διάρκεια των σπουδών μου ήταν δίπλα μου και με στήριζε. Την πτυχιακή εργασία μου θα ήθελα να την αφιερώσω στους δύο πιο σημαντικούς άντρες της ζωής μου που με έκαναν να αγαπήσω αυτό το αντικείμενο και μου έδωσαν ερεθίσματα ώστε να το ακολουθήσω και να γίνει μέρος της ζωής μου. Την αφιερώνω λοιπόν στον Πατέρα μου Γεώργιο και τον Αδερφό μου Αλκιβιάδη. Επίσης θα ήθελα να ευχαριστήσω και τους καθηγητές μου, για την υπομονή και κατανόηση που έδειξαν απέναντί μου, καθώς και για την συμβολή τους και την όρεξη που έδειχναν στο να μου μάθουν νέα πράγματα πάνω στο αντικείμενο, καθώς και για την προθυμία τους να μου λύσουν κάθε απορία που είχα, σχετική ή μη με το μάθημα που δίδασκε ο κάθε ένας (ακόμη και τώρα που έχω τελειώσει από τη σχολή).

Πίνακας περιεχομένων

Κεφάλαιο 1 – Οι Τεχνολογίες Που Χρησιμοποιήθηκαν

1.1 Η Τεχνολογία MVC (Model-View-Controller)

1.1.1 Εισαγωγή Στο Μοντέλο MVC

Είναι αλήθεια ότι ο όρος MVC , ο οποίος σημαίνει Model-View-Controller, δεν είναι και τόσο νέος στον προγραμματισμό. Εμφανίστηκε για πρώτη φορά στην γλώσσα προγραμματισμού smalltalk.

Η smalltalk είναι μία αντικειμενοστραφής γλώσσα προγραμματισμού (oop) η οποία έχει την εξής δυνατότητα: κατά τη διάρκεια που τρέχουμε ένα πρόγραμμα η smalltalk μπορεί να αλλάζει τη δομή και την συμπεριφορά των δεδομένων του προγράμματος καθώς και διαφόρων συναρτήσεων αυτού κλπ. Υπάρχει από το 1978 και το project Smalltalk της εταιρίας XEROX.

Σε αυτό λοιπόν το project ορίστηκε για πρώτη φορά ο όρος MVC. Για να γίνει πιο κατανοητό το MVC είναι ένα είδος αρχιτεκτονικής ,είναι δηλαδή ένας συγκεκριμένος τρόπος με τον οποίο «χτίζουμε» ένα application. Ο τρόπος αυτός θα αναλυθεί με απλό τρόπο παρακάτω. (Από εδώ και κάτω θα υπάρξει ορολογία με αγγλικούς όρους. Θα προσπαθήσω να κάνω μία μετάφραση αλλά όλοι οι όροι που χρησιμοποιούνται στον προγραμματισμό είναι στα αγγλικά οπότε το να κάνουμε μεταφράσεις δεν είναι και η καλύτερη ιδέα.)

Πολύ σημαντικό ρόλο έχουν οι **Οντότητες - Entities** στην MVC αρχιτεκτονική. Τον όρο «οντότητα» ή στα Αγγλικά «entity» τον συναντούμε πολύ συχνά στο MVC αλλά και γενικότερα στον προγραμματισμό. Είναι όρος που προκύπτει κατά τη διαδικασία μοντελοποίησης ενός συστήματος και δίνει την δυνατότητα στον developer να δει από ποιες βασικές οντότητες απαρτίζεται η εφαρμογή του. Για παράδειγμα σε μία εφαρμογή με blogs, άρθρα, σχόλια, χρήστες οι οντότητες είναι: ο χρήστης, το blog, το comment, το άρθρο κλπ. Είναι πράγματα δηλαδή που υπάρχουν και πάνω σε αυτά βασίζουμε τις λειτουργίες του συστήματός μας. Συνήθως με τον όρο entity παρουσιάζουμε και ένα πίνακα από μία βάση δεδομένων. Δημιουργούμε «οντότητες» και μας βοηθά αυτό να δημιουργήσουμε την εφαρμογή μας πιο εύκολα και πιο σίγουρα.

Για να δημιουργήσουμε μία εφαρμογή σε MVC μπορούμε να χρησιμοποιήσουμε κάποιο framework, ανάλογα τη γλώσσα προγραμματισμού που θα χρησιμοποιήσουμε, το οποίο framework κάνει τη διαδικασία δημιουργίας του application πιο γρήγορη ,πιο ασφαλή - διότι σε ένα framework υπάρχουν πολλές λειτουργίες by default , δεν χρειάζεται να γράψουμε κώδικα - και πιο ευχάριστη. Κάποια γνωστά PHP frameworks είναι τα εξής: Codeigniter, CakePHP, Kohana ενώ γνωστά ASP.NET frameworks είναι τα : SignalR, NancyFX, Bootstrap, AngularJS.

1.1.2 Τι Σημαίνουν Τα Αρχικά MVC

Το **Model–View–Controller** (σε συντομογραφία αναφέρεται ως **MVC**) είναι ένα μοντέλο αρχιτεκτονικής λογισμικού το οποίο χρησιμοποιείται για την δημιουργία περιβαλλόντων αλληλεπίδρασης χρήστη. Στο μοντέλο αυτό η εφαρμογή διαιρείται σε τρία διασυνδεδεμένα μέρη. Τα μέρη αυτά είναι τα εξής :

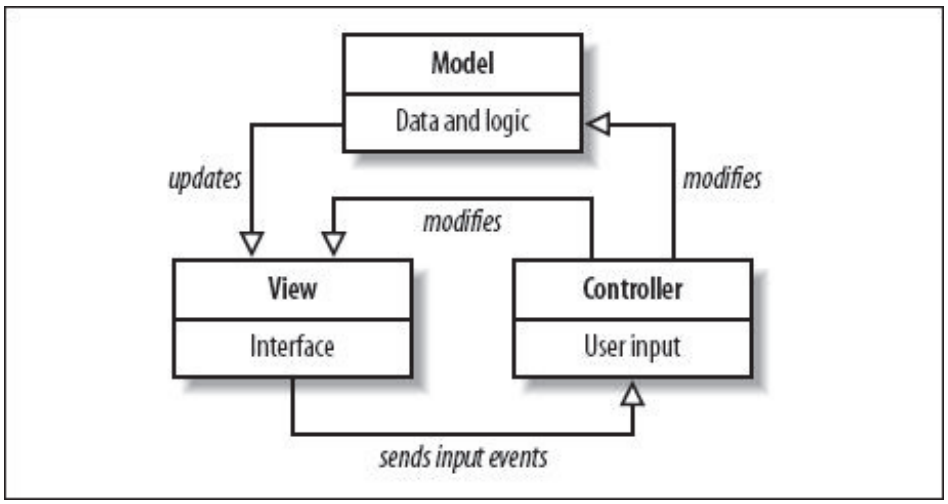
- **Model** : Στο model τοποθετούμε τις λειτουργίες της εφαρμογής που σχετίζονται με την πρόσβαση στη βάση δεδομένων. Οι λειτουργίες αυτές είναι με τη μορφή function (μεθόδων). Είναι κάποιες συναρτήσεις με τις οποίες εκτελούμε διάφορες λειτουργίες διαχείρισης των δεδομένων που λαμβάνουμε από τη βάση. Για παράδειγμα αν θέλουμε σε μία σελίδα να εμφανίσουμε κάποια δεδομένα από τη βάση δεδομένων ,το πρώτο βήμα είναι ότι στο model της εφαρμογής υπάρχει κάποια function για παράδειγμα «getAllData()» η οποία περιέχει κώδικα που μιλάει με τη βάση και τραβάει τα δεδομένα που θέλουμε. Ουσιαστικά ενημερώνει τις αντίστοιχες αναπαραστάσεις Views (όπου καλούνται με της σειρά τους να ενημερώσουν την γραφική απεικόνιση) και τους Controllers όταν υπάρχει αλλαγή στα δεδομένα

- **View** : Το View αναπαριστά με γραφικό τρόπο την πληροφορία που περιέχει το Model δημιουργώντας γραφική απεικόνιση στον χρήστη . Μέσα στη view δηλαδή υπάρχει το HTML της σελίδας της εφαρμογής μας. Είναι αυτό που βλέπουμε. Τις περισσότερες φορές μία View μιλάει με ένα controller και αφού ο controller κάνει τις διάφορες επεξεργασίες των δεδομένων στέλνει στη View συγκεκριμένα δεδομένα να εμφανίσει.

- **Controller** : Ο controller είναι ο μεσίτης μεταξύ Model και View. Ελέγχει το πώς «τρέχει» η εφαρμογή. Ο Controller μπορεί να στείλει εντολές στο Model και να ενημερώνει την κατάσταση του Model καθώς επίσης μπορεί να στείλει εντολές ώστε να γίνει η αντίστοιχη αναπαράσταση των δεδομένων του Model μέσω του View.

1.1.3 Παράδειγμα – Επεξήγηση

Ας δούμε μία συνολική εικόνα για το MVC :



Εικόνα 1.1 Τα μέρη από τα οποία αποτελείται το MVC

Στην παραπάνω περίπτωση συμβαίνουν τα εξής: Ο χρήστης δίνει κάποιο input στο site. Για παράδειγμα συμπληρώνει μία φόρμα και πατάει το κουμπί submit. Στη συνέχεια ο controller έχοντας λάβει το input του χρήστη μιλάει με το model χρησιμοποιώντας το input του χρήστη σαν μεταβλητή και ζητάει δεδομένα από το model τα οποία όταν τα λαμβάνει τα προσαρμόζει ανάλογα με αυτό που ζήτησε ο χρήστης. Στη συνέχεια ο controller με τα νέα δεδομένα πλέον , αλλάζει την view. Για παράδειγμα εάν σε ένα application βιβλιοπωλείου ο χρήστης συμπληρώσει ένα search form για βιβλία με συγκεκριμένη κατηγορία ο αντίστοιχος controller θα «μιλήσει» με κάποια function του model που θα ζητάει όλα τα βιβλία και θα δέχεται ως παράμετρο την κατηγορία. Το model θα βρίσκει τα βιβλία αυτά και μέσω της αντίστοιχης function στο controller θα τα επιστρέφει στη view, η οποία με τη σειρά της θα τα εμφανίζει στην οθόνη.

1.1.4 Βασικά Πλεονεκτήματα Του MVC

Χτίζοντας μία εφαρμογή με MVC έχουμε τα εξής βασικά πλεονεκτήματα :

- **Διαχωρισμός Προβλημάτων (Separation of Concers)**

Αυτό είναι και το πιο βασικό πλεονέκτημα του MVC. Ουσιαστικά δημιουργείται μία εφαρμογή η οποία έχει τρία επίπεδα, το επίπεδο των models , το επίπεδο των controllers και το επίπεδο των views όπου το κάθε επίπεδο επιτελεί ξεχωριστό έργο και ταυτόχρονα συνεργάζεται με τα άλλα επίπεδα. Μία σωστή MVC εφαρμογή είναι εκείνη που τα τρία επίπεδα είναι ξεκάθαρα καθορισμένα και δεν συμπλέκονται. Για παράδειγμα είναι λάθος στο επίπεδο των View να υπάρχει κώδικας που «μιλάει» με την βάση δεδομένων και «τραβάει» δεδομένα.

- **Επεκτασιμότητα**

Το δεύτερο πλεονέκτημα της MVC αρχιτεκτονικής είναι πολύ σημαντικό επίσης. «Επεκτασιμότητα» είναι η δυνατότητα που διαθέτει μία εφαρμογή , κατά την οποία μπορούμε μελλοντικά να προσθέσουμε λειτουργίες σε αυτή ή να αλλάξουμε κάποιες από τις ήδη υπάρχουσες λειτουργίες και να έχουμε άλλα αποτελέσματα. Για να το δούμε εντελώς απλά και κατανοητά, η πλατφόρμα WordPress είναι επεκτάσιμη με τη χρήση των διάφορων plugins διότι προσθέτουμε στις ήδη υπάρχουσες λειτουργίες και άλλες λειτουργίες. Τα προγράμματα που είναι φτιαγμένα με MVC αρχιτεκτονική έχουν βασικό χαρακτηριστικό ότι είναι επεκτάσιμα.

- **Ελεγξιμότητα (Testability)**

Αυτό είναι ένα πολύ κρίσιμο χαρακτηριστικό. Οι MVC εφαρμογές έχουν την δυνατότητα να είναι ελέγξιμες και με τον τρόπο αυτό συντηρούνται πιο εύκολα. Ας κάνουμε ένα απλό παράδειγμα. Έστω ότι έχουμε μία εφαρμογή η οποία διαθέτει μία λειτουργία login, δηλαδή ζητά από τον χρήστη να πληκτρολογήσει κάποια στοιχεία σε μία φόρμα και εν συνεχεία τον εισάγει μέσα στο σύστημα. Αυτή τη λειτουργία την ελέγχει κάποιος loginController ο οποίος περιέχει κώδικα που διαχειρίζεται τα δεδομένα αυτά που εισήχθησαν από τον χρήστη. Αυτός ο controller θεωρείται μία «μονάδα» ή αλλιώς unit. Στα MVC frameworks μπορούμε με πολλή ευκολία να γράψουμε απλό κώδικα-tests με τον οποίο τεστάρουμε αυτόν τον controller αλλά και κάθε μία από τις λειτουργίες του. Παίρνουμε τα αποτελέσματα και βλέπουμε αν η συγκεκριμένη μονάδα της εφαρμογής μας λειτουργεί σωστά.

- **<<Καθαρά>> URLs**

Τα περισσότερα MVC frameworks για web applications δίνουν τη δυνατότητα να έχουμε «καθαρά» urls. Ας κάνουμε ένα παράδειγμα. Έστω ότι έχουμε ένα blog και πατάμε το link για να διαβάσουμε ένα άρθρο. Ένα τυπικό URL θα μπορούσε να ήταν :

- http://example.com/article/Page.aspx?action=show&art_id=236

Με το MVC μπορούσαμε να έχουμε το εξής :

- http://example.com/articles/nice-link

Βλέπουμε λοιπόν ότι με το MVC μπορούμε να έχουμε πιο όμορφα και εύληπτα από το χρήστη αλλά και την μηχανή αναζήτησης URLs.

1.2 .NET Framework

1.2.1 Εισαγωγή

Το .NET Framework είναι ένα πλαίσιο λογισμικού (software framework) που προορίζεται για την πλατφόρμα των Windows. Αποτελείται από μια μεγάλη βιβλιοθήκη κλάσεων και υποστηρίζει μια μεγάλη γκάμα γλωσσών προγραμματισμού με τη δυνατότητα η μια να μπορεί να χρησιμοποιηθεί από την άλλη. Μέσα από τη σύνδεση της C# το .NET Framework προσφέρει δύο ενδιαφέρουσες έννοιες. Η μία είναι η CLR (Common Language Runtime), η οποία περιέχει μια εικονική μηχανή (virtual machine) που διαχειρίζεται την εκτέλεση ενός προγράμματος και παρέχει μια σειρά σημαντικών υπηρεσιών όπως ασφάλεια, διαχείριση μνήμης και διαχείριση εξαιρέσεων. Ουσιαστικά κάνει τα προγράμματα να είναι εύχρηστα και ενισχύει την δυνατότητα ανάμειξης πολλών γλωσσών προγραμματισμού. Η δεύτερη έννοια είναι .NET βιβλιοθήκη κλάσεων (class library). Αυτή η βιβλιοθήκη δίνει στα προγράμματα πρόσβαση στο περιβάλλον χρόνου εκτέλεσης (runtime environment).

Επίσης, βασικές λειτουργίες όπως οι γραφικές διεπαφές χρηστών (Graphical User Interfaces – GUIs), η επικοινωνία με βάσεις δεδομένων, η κρυπτογραφία, η ανάπτυξη web εφαρμογών και οι δικτυακές επικοινωνίες παρέχονται μέσω του Application Programming Interface (API) του .NET και μπορούν να συνδυαστούν με κώδικα από τους προγραμματιστές για τη δημιουργία ολοκληρωμένων εφαρμογών. Όσο ένα πρόγραμμα περιορίζεται στη χρήση της βιβλιοθήκης κλάσεων του .NET μπορεί να τρέχει οπουδήποτε υπάρχει εγκατεστημένο το περιβάλλον εκτέλεσης του .NET.

Το ASP.NET είναι ένα ενοποιημένο μοντέλο ανάπτυξης ιστού που περιλαμβάνει τις υπηρεσίες που είναι απαραίτητες για την ανάπτυξη εφαρμογών Web. Το ASP.NET είναι μέρος του .NET Framework και κατά την ανάπτυξη των εφαρμογών ASP.NET έχετε πρόσβαση σε όλες τις κλάσεις του .NET Framework. Μπορείτε να χρησιμοποιήσετε οποιαδήποτε γλώσσα είναι συμβατή με το CLR (Common Language Runtime) συμπεριλαμβανομένου και των Visual Basic και C#.

Δημιουργήθηκε τον Ιανουάριο του 2002 και είναι ο διάδοχος της τεχνολογίας Active Server Pages (ASP). Είναι ένα σύνολο τεχνολογιών όπου δίνουν την δυνατότητα υλοποίησης Web εφαρμογών και XML Web Services. Εκτελείται στον server και παράγει HTML, WML ή XML όπου αποστέλλονται σε κάποιο desktop ή mobile browser (περιέχει server-side logic). Επίσης χρησιμοποιεί ένα event-driven προγραμματιστικό μοντέλο όπου βελτιώνει την απόδοση και επιτρέπει τον διαχωρισμό μεταξύ της λογικής της εφαρμογής και της διεπαφής του χρήστη.

1.2.2 Χαρακτηριστικά Του .NET Framework

1. Υποστήριξη πολλαπλών γλωσσών

Στο .NET, η διαλειτουργικότητα μεταξύ των γλωσσών επιτυγχάνεται μέσω της δια-γλωσσικής κληρονομικότητας Μαζί με ένα ενοποιημένο σύστημα τύπων, η συνένωση κώδικα γραμμένου σε διαφορετικές γλώσσες είναι πλέον εύκολη υπόθεση.

2. Ανάπτυξη βασισμένη σε components

Η τέχνη της δημιουργίας ανεξάρτητων κομματιών λογισμικού που περιέχουν βιβλιοθήκες κλάσεων και διαμοιράσιμα τμήματα λειτουργικότητας μιας εφαρμογής έχει γίνει πολύ πιο εύκολη από παλιότερα.

3. Μεταφερσιμότητα

Η μεταφερσιμότητα (portability) επιτυγχάνεται μέσω της εικονικής μηχανής στο CLR. Η εικονική μηχανή αυτή είναι η προδιαγραφή μιας αφηρημένης εικονικής μηχανής για την οποία κώδικας γραμμένος σε οποιαδήποτε από τις .NET γλώσσες μεταγλωττίζεται σε μια εν- διάμεση γλώσσα γνωστή ως Common

Intermediate Language (CIL) και περιέχεται σε ένα assembly. Ένα assembly είναι είτε ένα εκτελέσιμο αρχείο (exe) είτε ένα dll. Η μεταγλώττιση σε αυτήν την ενδιάμεση γλώσσα είναι που εξασφαλίζει και την ανεξαρτησία από την πλατφόρμα.

4. Απλή εγκατάσταση εφαρμογών
5. Επικοινωνία με υπάρχων κώδικα

Παρόλο που η πλατφόρμα .NET προορίζεται να είναι ένα υποκατάστατο των παλαιότερων τεχνολογιών, υποστηρίζει τη χρήση κομματιών λογισμικού που έχουν αναπτυχθεί σε παλιότερες τεχνολογίες.

6. Αξιοπιστία

Το .NET είναι σχεδιασμένο για τη δημιουργία λογισμικού υψηλής αξιοπιστίας (robustness). Πραγματοποιεί εκτεταμένους ελέγχους κατά τη μεταγλώττιση αλλά και κατά την εκτέλεση των προγραμμάτων. Τα χαρακτηριστικά του .NET από μόνα τους ωθούν τους προγραμματιστές στην απόκτηση αξιόπιστων προγραμματιστικών συνηθειών.
Το μοντέλο διαχείρισης της μνήμης είναι απλό: η δέσμευση μνήμης γίνεται με μια λέξη (new), δεν υπάρχουν δείκτες προς θέσεις μνήμης σαν τύποι δεδομένων, ούτε αριθμητική δεικτών ενώ η αποδέσμευση μνήμης γίνεται αυτόματα μέσω του μηχανισμού garbage collection. Αυτό το απλό μοντέλο διαχείρισης μνήμης απαλείφει ολόκληρες κατηγορίες λαθών που απασχολούν τους προγραμματιστές σε C και C++. Η ανάπτυξη προγραμμάτων γίνεται με την πεποίθηση ότι αρκετά λάθη θα εντοπιστούν από το σύστημα πολύ πριν την ολοκλήρωσή της.

7. Αυτόματη διαχείριση μνήμης

Άλλο ένα σημαντικό χαρακτηριστικό του .NET Framework είναι η δυνατότητα αυτόματης διαχείρισης της μνήμης μέσω του Garbage Collector. Ιστορικά η διαχείριση της μνήμης ήταν από τις δυσκολότερες εργασίες που είχε να κάνει ένας προγραμματιστής. Οι παλαιότερες προσεγγίσεις στη διαχείριση μνήμης ήταν είτε χαμηλού επιπέδου (όπως η malloc/free στη C και η new/delete στη C++), προκαλώντας διάφορα bugs, είτε αρκετά πολύπλοκη. Ένας από τους στόχους του .NET και του CLR πιο συγκεκριμένα ήταν να απαλείψει από τους προγραμματιστές την ανάγκη διαχείρισης της μ νήμης.

Το CLR με το μηχανισμό Garbage Collector διαχειρίζεται τη μνήμη ερευνώντας πότε μπορεί να ελευθερωθεί με ασφάλεια μνήμη που δεν χρειάζεται άλλο σε ένα πρόγραμμα. Η μνήμη δεσμεύεται με την αρχικοποίηση των διαφόρων τύπων (αντικειμένων) από ένα διαθέσιμο τμήμα της μνήμης που διαχειρίζεται το CLR γνωστό ως heap (σωρός). Όσο υπάρχει κάποια άμεση ή έμμεση (μέσω του γράφου των αντικειμένων) αναφορά προς ένα αντικείμενο, τότε αυτό θεωρείται πως χρησιμοποιείται. Στην αντίθετη περίπτωση, όταν δεν μπορεί να αναφερθεί από κάποιο άλλο τμήμα του κώδικα τότε είναι διαθέσιμο για καταστροφή. Ο garbage collector τρέχει ανά τακτά χρονικά διαστήματα σε ένα ξεχωριστό thread, εντοπίζει ποια αντικείμενα είναι έτοιμα προς καταστροφή και αποδεσμεύει τη μνήμη που αυτά χρησιμοποιούσαν.

8. Ασφάλεια

Το .NET είναι σχεδιασμένο να λειτουργεί σε κατανεμημένα περιβάλλοντα, η ασφάλεια (security) των οποίων είναι υψίστης σημασίας. Με μέτρα που είναι ενσωματωμένα στις γλώσσες αλλά και κατά την εκτέλεση των προγραμμάτων, το .NET επιτρέπει τη δημιουργία προγραμμάτων που δεν μπορούν να προσβληθούν από έξω. Σε δικτυακά περιβάλλοντα οι εφαρμογές .NET είναι ασφαλείς από εισβολές μη εξουσιοδοτημένου κώδικα που επιχειρεί να δράσει στο παρασκήνιο και να εγκαταστήσει κακόβουλο λογισμικό ή να εισβάλει σε συστήματα αρχείων. Το περιβάλλον εκτέλεσης του .NET έχει ενσωματωμένο έναν μηχανισμό ασφάλειας με το όνομα Code Access Security (CAS) που απομονώνει τον κώδικα ανάλογα με την προέλευσή του, τον δημιουργό του ή άλλες πολιτικές.

9. Απόδοση

Η ενδιάμεση γλώσσα στην οποία μεταγλωττίζονται τα προγράμματα στο .NET, μεταγλωττίζεται πάντα με τη μέθοδο Just-in-Time (JIT). Ο μεταγλωττιστής JIT μεταγλωττίζει κάθε τμήμα του κώδικα στην ενδιάμεση γλώσσα όταν αυτό καλείται (just in time). Όταν έχει ήδη μεταγλωττιστεί κάποιο τμήμα κώδικα, το αποτέλεσμα αποθηκεύεται μέχρι η εφαρμογή να τερματίσει έτσι ώστε να μην χρειάζεται να ξανά μεταγλωττιστεί όταν ζητηθεί το συγκεκριμένο τμήμα. Η διαδικασία αυτή είναι αποδοτικότερη σε σύγκριση με την απευθείας μεταγλώττιση ολόκληρου του κώδικα, γιατί υπάρχει περίπτωση μεγάλα τμήματα κώδικα να μην εκτελεστούν ποτέ στην πραγματικότητα. Με τον μεταγλωττιστή JIT τέτοιος κώδικας δεν θα μεταγλωττίζονταν ποτέ.

10. Υποστήριξη με επαγγελματικά εργαλεία

Το .NET Framework συνοδεύεται από ένα σύνολο εργαλείων για την υποστήριξη της ανάπτυξης λογισμικού σε αυτό, όπως το σύστημα σχεδιασμού, τα frameworks για unit testing, την πλατφόρμα για build MSBuild και το σύστημα debugging, μέσα από το Visual Studio. Το Visual Studio είναι ένα από τα καλύτερα Integrated Development Environments (IDEs). Χρησιμοποιείται για την ανάπτυξη όλων των ειδών εφαρμογών που μπορούν να αναπτυχθούν στο .NET: από console εφαρμογές και εφαρμογές με γραφικές διεπαφές (GUIs) μέχρι web εφαρμογές και web services σε managed αλλά και εγγενή κώδικα. Οι Premium και Ultimate εκδόσεις του Visual Studio στοχεύουν στην ομαδική ανάπτυξη μαζί με τον Team Foundation Server (TFS) που χρησιμοποιείται για την παρακολούθηση ενός έργου .NET, τη διαχείριση του κώδικα, τη διαχείριση των αναφορών αλλά και το διαμοιρασμό κώδικα, τεκμηριώσεων ή άλλων εγγράφων.

1.2.3 Common Language Runtime (CLR)

Το Common Language Runtime (CLR) είναι μία υλοποίηση του πρότυπου Common Language Infrastructure (CLI) από τη Microsoft. Το CLI προσδιορίζει την υποδομή εκτέλεσης που απαιτείται για τη χρήση managed κώδικα, έτσι ώστε διαφορετικές βιβλιοθήκες και γλώσσες να συνεργάζονται. Υπάρχουν και άλλες υλοποιήσεις CLI από τρίτους παρόχους όπως το Mono και το DotGNU Portable.NET αλλά και της Microsoft με το όνομα Shared Source CLI (SSCLI) γνωστό και ως Rotor, το οποίο τρέχει σε πλατφόρμες όπως το FreeBSD. Αυτό θεωρητικά σημαίνει ότι ο κώδικας που στοχεύει στο .NET όχι μόνο επιτρέπει τη διαλειτουργικότητα μεταξύ διαφορετικών γλωσσών προγραμματισμού αλλά και την εκτέλεση σε διαφορετικές πλατφόρμες.

Η προδιαγραφή CLI και κατά συνέπεια το CLR αποτελείται από τρία κύρια στοιχεία :

- το Virtual Execution System (VES)
- το Common Type System (CTS) και
- το Common Language Specification (CLS)

Η VES είναι η καρδιά του CLR και είναι αυτή που εξασφαλίζει τη μεταφερσιμότητα των προγραμμάτων σε .NET. Η VES αντιλαμβάνεται το format της ενδιάμεσης γλώσσας Common Intermediate Language (CIL) και τα matadata που περιγράφουν τον κώδικα σε αυτήν την ενδιάμεση μορφή. Ο κώδικας σε CIL μορφή είναι ίδιος ανεξάρτητα από το υλικό ή την πλατφόρμα όπου εκτελείται. Η VES είναι ουσιαστικά μία εικονικά μηχανή διεργασίας (process virtual machine). Τέτοιες μηχανές τρέχουν σαν κανονική εφαρμογή μέσα στο λειτουργικό σύστημα στο οποίο φιλοξενούνται και υποστηρίζονται από μία διεργασία. Η λειτουργία τους ξεκινά με την έναρξη της διεργασίας και παύει με τον τερματισμό της.

Ο σκοπός των εικονικών μηχανών είναι να παρέχουν ένα περιβάλλον εκτέλεσης προγραμμάτων, ανεξάρτητο συνήθως από την πλατφόρμα που τρέχουν, η οποία κρύβει τις λεπτομέρειες του υποκείμενου υλικού ή λειτουργικού συστήματος, επιτρέποντας έτσι στα προγράμματα να τρ έχουν με τον ίδιο τρόπο σε κάθε πλατφόρμα. Δεν πρέπει να συγχέονται με τις εικονικές μηχανές συστήματος (system virtual machines) που παρέχουν μια πλήρη πλατφόρμα για την εκτέλεση ενός ολόκληρου λειτουργικού συστήματος όπως η VirtualBox, VMWare ή η Hyper-V.

Για να εκτελεστεί ο κώδικας μιας γλώσσας στο CLR, πρέπει πρώτα να μεταγλωττιστεί σε μια μορφή η οποία μπορεί να αναγνωριστεί κατά την εκτέλεση. Αυτή η μορφή είναι η Common Intermediate Language (CIL συχνά και σκέτο IL). Ο κώδικας που εκτελείται στο CLR είναι γνωστός ως managed κώδικας, σε αντίθεση με τον εγγενή κώδικα (native code) που εκτελείται κατευθείαν στον επεξεργαστή της μηχανής χωρίς επιπρόσθετη υποδομή εκτέλεσης. Εκτός από τις εντολές σε IL για την εκτέλεση μιας εφαρμογής το CLR χρειάζεται μεταδεδομένα (metadata) που περιγράφουν τη δομή και την οργάνωση των τύπων δεδομένων. Η χρήση των μεταδεδομένων απαλείφει την ανάγκη ύπαρξης header files και βοηθά διάφορα εργαλεία να επιθεωρήσουν τους τύπους. Ένα τέτοιο παράδειγμα είναι το Intellisense τ ου Visual Studio. Η ενδιάμεση γλώσσα CIL περιέχεται μέσα στα assemblies τα οποία, εκτός από το ρόλο τους στην εκτέλεση ενός προγράμματος, αποτελούν και της βασικές μονάδες εγκατάστασης στο .NET. Εξαιτίας της αυτοπεριγραφικής φύσης τους που παρέχεται από τα μεταδεδομένα, τα assemblies δεν χρειάζονται την βοήθεια πολύπλοκων οδηγών εγκατάστασης (installers) για την εγγραφή τους στο registry, αφού μια απλή αντιγραφή τους αρκεί, ενώ είναι δυνατή η ύπαρξη πολλαπλών εκδόσεων των assemblies αφού ενσωματώνουν τον αριθμό έκδοσής τους.

Ένα από τα σημαντικότερα κομμάτια του CLR είναι το CTS γιατί εξασφαλίζει τη δια- λειτουργικότητα μεταξύ των γλωσσών. Το CTS ορίζει τους προκαθορισμένους τύπους δεδομένων που είναι διαθέσιμοι στην ενδιάμεση γλώσσα έτσι ώστε όλες οι γλώσσες του .NET να παράγουν μεταγλωττισμένο κώδικα που βασίζεται σε αυτούς τους τύπους. Για παράδειγμα ο τύπος δεδομένων Integer της Visual Basic 2010 είναι στην πραγματικότητα ένας 32-bit προσημασμένος ακέραιος, ο οποίος αντιστοιχεί στην ενδιάμεση γλώσσα στον τύπο Int32. Επειδή ο μεταγλωττιστής της C# κατανοεί τον τύπο αυτόν δεν υπάρχει κάποιο πρόβλημα στη χρήση από τη C# ενδιάμεσου κώδικα παραγμένο από VB.

Το τελευταίο σημαντικότερο κομμάτι του CLR είναι το Common Language Specification (CLS). Το CLS είναι ένα σύνολο κανόνων στο οποίο βασίζονται όλες οι γλώσσες του .NET για να εξασφαλίσουν ότι δεν τίθενται προς χρήση τύποι δεδομένων τους οποίους δεν μπορούν να χειριστούν μερικές γλώσσες. Αν τηρούνται όλοι οι κανόνες ένας τύπος ορίζεται ως συμβατός με το CLS και κατά συνέπεια μπορεί να χρησιμοποιηθεί από όλες τις γλώσσες. Ένα παράδειγμα ενός κανόνα είναι ένας τύπος δεν μπορεί να έχει public μέλη που διακρίνονται μόνο από το αν οι χαρακτήρες τους είναι πεζοί ή κεφαλαίοι (π.χ. Bar και bar). Μερικές γλώσσες, όπως η Visual Basic, δεν κάνουν διάκριση σε πεζούς και κεφαλαίους χαρακτήρες και δεν θα μπορούσαν να αντιληφθούν τη διαφορά.

1.2.4 Αρχιτεκτονική Του .NET Framework

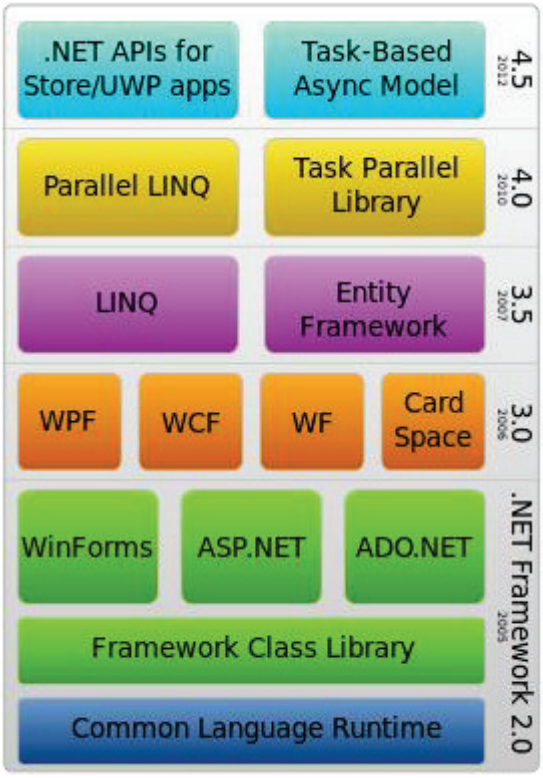
Ένας προγραμματιστής στην πλατφόρμα Windows έχει πρωταρχικά δυο μονοπάτια που μπορεί να ακολουθήσει κατά την ανάπτυξη εφα ρμογών. Το πρώτο είναι η ανάπτυξη εφαρμογών μη διαχειρίσιμου κώδικα (unmanaged code) και περιλαμβάνει τον προγραμματισμό κατευθείαν στο λειτουργικό σύστημα, χρησιμοποιώ ντας το Windows Application Programming Interface (Windows API) συνήθως με τη γλώσσα C++. Στόχος του Windows API είναι να δώσει τη δυνατότητα στους προγραμματιστές να αναπτύξουν προγράμματα με το λειτουργικό σύστημα για να αποφύγουν το γράψιμο κώδικα για τη δημιουργία components των Windows όπως φόρμες, κουμπιά μενού και άλλα. Με το Windows API ένας προγραμματιστής καλεί τις συναρτήσεις και αφήνει το λειτουργικό σύστημα να δημιουργήσει αυτά τα components. Ο κύριος λόγος χρήσης του Windows API είναι η απουσία υποστήριξης διαφόρων λειτουργιών από τη βασική έκδοση της χρησιμοποιούμενης γλώσσας προγραμματισμού. Η χρήση μιας λειτουργίας του Windows API από μια γλώσσα γίνεται με την κλήση της κατάλληλης συνάρτησης. Ωστόσο, οι κλήσεις τέτοιων συναρτήσεων είναι συνήθως μια πολύπλοκη διαδικασία που μπορεί να προκαλέσει απρόβλεπτες συνέπειες αν δεν γίνει σωστά. Η πολυπλοκότητα αυτή είναι ένας από τους λόγους που οδήγησε στη δημιουργία του δεύτερου μονοπατιού στην ανάπτυξη εφαρμογών, δηλαδή την ανάπτυξη εφαρμογών διαχειρίσιμου κώδικα.

Ο διαχειριζόμενος κώδικας εκτελείται κάτω από τον έλεγχο του CLR με την μεθοδολογία που αναλύθηκε. Από την στιγμή λοιπόν που εκτελείται κάτω από τον έλεγχο του CLR, ο διαχειριζόμενος κώδικας είναι αντικείμενο στους προκαθορισμένους περιορισμούς και αυτό παράγει αρκετά πλεονεκτήματα. Το .NET framework αποτελείται από ένα περιβάλλον εκτέλεσης το οποίο διαχειρίζεται την εκτέλεση κώδικα μέσω

κανόνων ασφαλείας, αυτόνομης διαχείρισης της μνήμης και αυστηρότητας τύπων, μειώνοντας έτσι σε μεγάλο βαθμό την πιθανότητα πρόκλησης λαθών από αυτούς τους παράγοντες.

Από την άλλη πλευρά ο μη διαχειριζόμενος κώδικας δεν εκτελείται κάτω από το CLR. Συνεπώς να μην παρέχονται όλοι αυτοί οι έλεγχοι με αποτέλεσμα να είναι πολύ πιο εύκολη η πρόκληση λαθών. Όλα τα προγράμματα που υπήρχαν πριν της έκδοσης του .NET Framework χρησιμοποιούσαν μη διαχειριζόμενο κώδικα.

Είναι πιθανό ο διαχειριζόμενος και μη κώδικας να εκτελούνται μαζί και αυτό καθιστά δυνατό τον συνδυασμό εφαρμογών οι οποίες δημιουργούνται τώρα να λειτουργούν με εφαρμογές οι οποίες προϋπήρχαν.



Εικόνα 1.2 .NET Framework overview

1.2.5 Τα Frameworks Που Προσφέρει Το ASP.NET

Τα τρία frameworks όπου προσφέρει το ASP.NET είναι :

1. **Web Forms (.aspx pages)**

Το Web Forms framework στοχεύει σε developers όπου προτιμούν control-based προγραμματισμό, όπως για παράδειγμα Windows Forms και WPF/XAML/Silverlight. Προσφέρει ένα WYSIWYG σχεδιαστικό μοντέλο ανάπτυξης (drag-and-drop), έτσι είναι δημοφιλές στους προγραμματιστές που αναζητούν ένα περιβάλλον γρήγορης ανάπτυξης εφαρμογών (RAD - rapid application development) για την ανάπτυξη ιστοσελίδων. Εάν είστε νέοι στο Web προγραμματισμό αλλά είστε εξοικειωμένοι με το περιβάλλον RAD τότε μπορείτε να δημιουργήσετε γρήγορα μία εφαρμογή web χωρίς να έχετε εμπειρία πάνω σε HTML και JavaScript.

Κάθε framework στοχεύει σε διαφορετικό κοινό και τύπο εφαρμογής. Το πιο θα επιλέξετε εξαρτάται από τον συνδυασμό της εμπειρίας πάνω σε web development, το framework με το οποίο είστε πιο εξοικειωμένοι και ποιο ταιριάζει καλύτερα στον τύπο εφαρμογής όπου καλείστε να υλοποιήσετε. Και τα τρία frameworks υποστηρίζονται και ενημερώνονται, καθώς υπάρχει και η συνεχής βελτίωσή τους με νεότερες εκδόσεις που βγαίνουν. Ειδικότερα, το μοντέλο Web Forms παρέχει τις ακόλουθες δυνατότητες :

- Ένα event-model που εκθέτει γεγονότα τα οποία μπορείτε να προγραμματίσετε σαν να προγραμματίσατε ένα client-application όπως WinForms ή WPF.
- Server controls that render HTML for you and that you can customize by setting properties and styles.
- Μια πλούσια ποικιλία ελέγχων για την πρόσβαση σε δεδομένα και την εμφάνιση δεδομένων.
- Αυτόματη διατήρηση των δεδομένων μεταξύ των αιτήσεων HTTP

Γενικά, για τη δημιουργία μιας εφαρμογής Web Forms απαιτείται λιγότερος προγραμματισμός από ότι για τη δημιουργία της ίδιας εφαρμογής χρησιμοποιώντας το framerork MVC ASP.NET. Ωστόσο, το Web Forms δεν είναι μόνο για γρήγορη ανάπτυξη εφαρμογών. Υπάρχουν πολλές σύνθετες εμπορικές εφαρμογές και πλαίσια εφαρμογών που βασίζονται σε αυτό το μοντέλο. Ένα από τα αρνητικά που έχει είναι ότι επειδή παρέχει αρκετά στοιχεία όπου παράγουν μόνα τους το HTML δεν προσφέρει λεπτομερή έλεγχο όπως τα άλλα frameworks του ASP.NET .

2. **ASP.NET Web Pages (.cshtml and .vbhtml files)**

Το Web Pages μοντέλο είναι κατάλληλο για προγραμματιστές όπου έχουν καλές σχέσεις με HTML αλλά δεν έχουν μεγάλη εμπειρία πάνω στον προγραμματισμό Επίσης είναι ένας εύκολος τρόπος κάποιος που ξέρει PHP ή έχει δουλέψει με παρόμοια frameworks να ενσωματωθεί με τη λογική του ASP.NET. Ουσιαστικά σε αυτό το μοντέλο δημιουργείτε σελίδες HTML και στην συνέχεια προσθέτετε server-based κώδικα προκειμένου να ελέγξετε την συμπεριφορά της σελίδας σας.

Όπως και το Web Forms, έτσι και το Web Pages μοντέλο είναι για γρήγορη ανάπτυξη εφαρμογών (RAD). Το μοντέλο αυτό παρέχει components τα οποία ονομάζονται helpers όπου σε βοηθούν να προσθέσεις λειτουργίες στην σελίδα σου χωρίς να χρειαστεί να γράψεις πολύ κώδικα όπου μπορεί να είναι είτε κουραστικό είτε περίπλοκο. Για παράδειγμα μερικοί helpers είναι, για την προβολή δεδομένων από τη βάση, τη σύνδεση στο Facebook, την προσθήκη χαρτών σε μία σελίδα κ.ο.κ.

Επίσης ο κώδικας σε μία τέτοια σελίδα .(cshtml ή .vbhtml) εκτελείται από πάνω προς τα κάτω όπως πχ και στην PHP.

3. **ASP.NET MVC**

Το μοντέλο αυτό στοχεύει σε προγραμματιστές οι οποίοι ενδιαφέρονται για test-driven προγραμματισμό. Επιτρέπει τον διαχωρισμό μεταξύ του επιπέδου της λογικής από το επίπεδο της παρουσίασης. Με τη διαφοροποίηση αυτή μπορεί να διευκολύνει την ανάπτυξη, την διαχείριση καθώς και την συντήρηση μίας μεγάλης και πολύπλοκης εφαρμογής. Το μοντέλο αυτό καθιστά δυνατή και τη συνεργασία πολλών ξεχωριστών ομάδων, καθώς διαφοροποιείται το back-end από το front-end.

Το μοντέλο MVC σάς δίνει τη δυνατότητα να αποκτήσετε πλήρη έλεγχο του τι ακριβώς κάνει η εφαρμογή σας και πώς συμπεριφέρεται στο περιβάλλον του διαδικτύου. Σχεδιάστηκε έτσι ώστε να είναι

επεκτάσιμο, δίνοντας με αυτόν τον τρόπο στους προγραμματιστές την δυνατότητα να προσαρμόζουν το framework σύμφωνα με τις απαιτήσεις της εφαρμογής όπου υλοποιούν.

Ουσιαστικά το ASP.NET MVC είναι ένα web application framework το οποίο υλοποιεί το μοτίβο model-view-controller και πρόκειται για open source λογισμικό. Όλες οι παλαιότερες εκδόσεις όπως ASP.NET, ASP.NET MVC, ASP.NET Web API, και ASP.NET Web Pages (πλατφόρμες όπου χρησιμοποιούσαν μόνο Razor σελίδες) ενσωματώθηκαν στο μοντέλο MVC 6.

Τα τρία frameworks δεν είναι εντελώς ανεξάρτητα και η επιλογή ενός δεν αποκλείει επίσης τη χρήση άλλου. Για παράδειγμα, οι σελίδες στο View (MVC μοντέλο) συχνά γράφονται ως αρχεία .cshtml ή .vbhtml (χρησιμοποιώντας τη σύνταξη "Razor"), πράγμα που σημαίνει ότι μπορούν να επωφεληθούν από ορισμένες από τις λειτουργίες των ιστοσελίδων όπως οι helpers. Δεδομένου ότι τα frameworks μπορούν να συνυπάρχουν στο ίδιο web application , δεν είναι ασυνήθιστο να βλέπουμε μεμονωμένα components μιας εφαρμογής να γράφονται με διαφορετικά frameworks.

1.2.6 Ιστορικό Κυκλοφορίας ASP.NET MVC

Version Number	CLR Version	Release Date	Support	Development Tool	Included in		Replaces
					Windows	Windows Server	
1.0	1.0	2002-02-13	2009-07-14	Visual Studio .NET	XP	N/A	N/A
1.1	1.1	2003-04-24		Visual Studio .NET 2003	N/A	2003	1.0
2.0	2.0	2005-11-07	1001-07-12	Visual Studio 2005	N/A	2003, 2003 R2, 2008 SP2, 2008 R2 SP1	N/A
3.0	2.0	2006-11-06	2001-07-12	Expressiona Blend	Vista	2008 SP2, 2008 R2 SP1	2.0
3.5	2.0	2007-11-19	2011-07-12 (except 3.5 SP1)	Visual Studio 2008	7, 8, 8.1, 10	2008 R2 SP1	2.0, 3.0
4.0	4	2010-04-12	2016-01-12	Visual Studio 2010	N/A	N/A	N/A
4.5	4	2012-08-15	2016-01-12	Visual Studio 2012	8	2012	4.0
4.5.1	4	2013-10-17	2016-01-12	Visual Studio 2013	8.1	2012 R2	4.0, 4.5
4.5.2	4	2014-05-05	N/A	N/A	N/A	N/A	4.0, 4.5.1
4.6	4	2015-07-20	N/A	Visual Studio 2015	10	N/A	4.0, 4.5.2

4.6.1	4	2015-11-30	N/A	Visual Studio 2015 Update 1	10 v1511	N/A	4.0, 4.6
4.6.2	4	2016-08-02	N/A	Visual Studio 2015	10 v1607	2016	4.0, 4.6.1
4.7	4	2017-04-05	N/A	Visual Studio 2017	10 v1703	N/A	4.0, 4.6.2
4.7.1	4	2017-10-17	N/A	Visual Studio 2017	10 v1709	N/A	4.0, 4.7

Πίνακας 1.1 Ιστορικό Κυκλοφορίας .NET Framework

Ημ/νία	Έκδοση		Ημ/νία	Έκδοση
10-12-2007	ASP.NET MVC CTP		17-10-2013	ASP.NET MVC 5
13-03-2009	ASP.NET MVC 1.0		17-01-2014	ASP.NET MVC 5.1
16-12-2009	ASP.NET MVC 2 RC		10-02-2014	ASP.NET MVC 5.1.1
04-02-2010	ASP.NET MVC 2 RC 2		04-04-2014	ASP.NET MVC 5.1.2
10-03-2010	ASP.NET MVC 2		22-06-2014	ASP.NET MVC 5.1.3
06-10-2010	ASP.NET MVC 3 Beta		01-07-2014	ASP.NET MVC 5.2.0
09-11-2010	ASP.NET MVC 3 RC		28-08-2014	ASP.NET MVC 5.2.2
10-12-1010	ASP.NET MVC 3 RC 2		09-02-2015	ASP.NET MVC 5.2.3
13-01-2011	ASP.NET MVC 3		18-11-2015	ASP.NET MVC 6.0.0-rc1
20-09-2011	ASP.NET MVC 4 Developer Preview		17-05-2016	ASP.NET Core MVC 1.0.0-rc2
15-02-2012	ASP.NET MVC 4 Beta		12-08-2016	ASP.NET Core MVC 1.0.0
31-05-2012	ASP.NET MVC 4 RC		17-08-2016	ASP.NET Core MVC 1.0.1
15-08-2012	ASP.NET MVC 4		17-11-2016	ASP.NET Core MVC 1.0.2
30-05-2013	ASP.NET MVC 4 4.0.30506.0		18-11-2016	ASP.NET Core MVC 1.1.0
26-06-2013	ASP.NET MVC 5 Preview		14-08-2017	ASP.NET Core MVC 2.0.0
23-08-2013	ASP.NET MVC 5 RC 1			

Πίνακας 1.2 Ιστορικό Κυκλοφορίας ASP.NET MVC

1.2.7 ASP.NET Vs PHP

Από την μία πλευρά η PHP είναι open source σε αντίθεση με το .NET που είναι μία πλατφόρμα επί πληρωμής της Microsft. Η PHP είναι μία μίξη μεταξύ πολλών γλωσσών προγραμματισμού και web framework ενώ το .NET είναι ένα application framework.

Ταχύτητα & Απόδοση

Αρχικά, μια κοινή εσφαλμένη αντίληψη σχετικά με την απόδοση και την ταχύτητα του ιστότοπου είναι ότι η γλώσσα που επιλέγετε να κωδικοποιήσετε καθορίζει τη συνολική απόδοση του ιστότοπού σας. Στην πραγματικότητα, ωστόσο, υπάρχει πολύ μικρή διαφορά μεταξύ των επιδόσεων των ιστοσελίδων της PHP και των ιστοσελίδων .NET.

Τόσο η ανάπτυξη ιστού PHP όσο και το .NET είναι εξίσου εξοπλισμένα για να έχουν πρόσβαση σε συστήματα αρχείων, να βρίσκουν εικόνες και να προβάλλουν σελίδες σε έναν διακομιστή ιστού και η ταχύτητα αυτών των επιδόσεων εξαρτάται πολύ περισσότερο από τον server της βάσης δεδομένων, τον υπολογιστή του τελικού χρήστη,το bandwidth κ.λπ.

Υποστήριξη

Η PHP είναι ανοικτού κώδικα και η ομάδα των προγραμματιστών της είναι πολύ μεγαλύτερη από το .NET. Παρόλα αυτά και οι δύο ομάδες δημοσιεύουν τακτικά σε φόρουμ στο διαδίκτυο, οπότε αν ψάχνετε για απαντήσεις σε προβλήματα, είναι πιθανό να βρείτε και τις δύο κοινότητες χρήσιμες.

Πλεονεκτήματα και μειονεκτήματα

PHP :

Πλεονεκτήματα :

1) Είναι ανοικτού κώδικα που σημαίνει ότι είναι ελεύθερο να χρησιμοποιείται και βελτιώνεται διαρκώς από μεγάλο αριθμό ανθρώπων και όχι από μία εταιρεία.
2) Η PHP έχει μεγάλη κοινότητα υποστήριξης.
3) Η PHP είναι ιδανική για μεγάλα έργα όπως το Facebook, οι ιστοσελίδες του Λευκού Οίκου.
4) Η PHP είναι ιδιαίτερα αποτελεσματική στην πρόσβαση και την επικοινωνία με διάφορους τύπους βάσεων δεδομένων. Αυτό τη καθιστά ιδανική για σενάρια που βασίζονται στο διαδίκτυο, όπως συστήματα διαχείρισης περιεχομένου ιστότοπων.
5) Είναι επεκτάσιμη και μπορείτε εύκολα να τη προσαρμόσετε ανάλογα με τις ανάγκες σας.
6) Υπάρχουν χιλιάδες προγραμματιστές Expert PHP που διατίθενται στην αγορά.
7) Χαμηλό κόστος και εύκολη εκμάθηση.

Μειονεκτήματα :

1) Ένα από τα κύρια μειονεκτήματα της PHP είναι ότι γενικά δεν είναι κατάλληλη για την κατασκευή εφαρμογών γραφείου.
2) Σε σύγκριση με άλλες γλώσσες, οι εφαρμογές PHP τείνουν να τρέχουν πιο αργά.
3) Ο χειρισμός σφαλμάτων PHP θεωρείται παραδοσιακά φτωχός σε σύγκριση με άλλες γλώσσες προγραμματισμού, πράγμα που σημαίνει ότι μπορεί να χρειαστεί περισσότερος χρόνος για να ανακαλύψετε γιατί ένα συγκεκριμένο κομμάτι κώδικα δεν λειτουργεί όπως αναμένεται.

.NET :

Πλεονεκτήματα :

1) Υποστηρίζει πολλές γλώσσες προγραμματισμού.
2) Ενημερώνει τον προγραμματιστή για σφάλματα πριν το compile.
3) Λειτουργεί εξαιρετικά καλά με τα Windows.

4) Εξαιρετική πλατφόρμα για εφαρμογές Enterprise.
5) Γρήγορη ανάπτυξη με πολλές προ-κωδικοποιημένες επιλογές.
6) Ένα εξαιρετικό UI για προγραμματιστές.
7) Έρχεται με πολλά εργαλεία και χαρακτηριστικά για την υποστήριξη προγραμματιστών.
8) Το .NET έχει μεγάλη δυνατότητα επεκτασιμότητας, σε σημείο που δημιουργεί ακόμη και έναν σκληρό ανταγωνισμό με τη PHP.

Μειονεκτήματα :

1) Μικρότερη κοινότητα υποστήριξης.
2) Λειτουργεί μόνο σε server της Microsoft
3) Είναι λίγο ακριβό λόγω της άδειας χρήσης της Microsoft.
4) Δεν είναι εύκολο να το μάθεις και πιο δύσκολο να το καταλάβεις.
5) Closed source technology.

Γενικά

1) Το ASP.NET χρησιμοποιεί VB.Net/Vb και C # για server side programming, ενώ στη php ο κώδικας αναμειγνύεται μεταξύ των html tags.
2) Το ASP. NET χρησιμοποιεί το IIS ως διακομιστή ιστού για να προβάλει / να μεταγλωττίσει τις σελίδες του σε πλατφόρμες windows ενώ η php χρησιμοποιεί τον apache ως web server, ο οποίος είναι προτιμότερο να τρέχει σε linux.

1.3 Η Γλώσσα Προγραμματισμού C#

1.3.1 Εισαγωγή

Ο αντικειμενοστραφής προγραμματισμός καθιερώθηκε τη δεκαετία του 1990 (η γλώσσα που συνεισέφερε αρκετά σε αυτό ήταν η Java). Εδώ το βασικό στοιχείο του προγράμματος είναι το αντικείμενο (object) που συνδυάζει ιδιότητες και διαδικασίες που επενεργούν σε αυτές αλλάζοντας τα δεδομένα τους. Κάθε αντικείμενο αποθηκεύει δεδομένα σε μεταβλητές και απαντά σε μηνύματα εκτελώντας τις διαδικασίες που καλούνται και μέθοδοι (methods). Το αντικείμενο αποτελεί δεσμευμένο κομμάτι της μνήμης (και συγκεκριμένα του σωρού) και καλείται και στιγμιότυπο (instance), ενός σύνθετου τύπου δεδομένων (data type) που καλείται κλάση (class). Η κλάση είναι μία αυτοτελής και αφαιρετική αναπαράσταση κάποιας κατηγορίας αντικειμένων, είτε φυσικών αντικειμένων του πραγματικού κόσμου είτε νοητών, εννοιολογικών αντικειμένων και στην ουσία αποτελεί την προδιαγραφή των δεδομένων και των διαδικασιών που επιδρούν πάνω σε αυτά.

Η C# είναι μια σχετικά καινούργια γλώσσα, ενώ έχει γίνει μια από τις πιο διαδεδομένες αντικειμενοστραφείς γλώσσες προγραμματισμού. Αναπτύχθηκε στη Microsoft, από μια ομάδα κάτω από την ηγεσία του Anders Hejlberg, σαν μέρος του .NET Framework. Η C# είναι μια συνεχώς εξελισσόμενη γλώσσα και με κάθε νέα της έκδοση προστίθενται νέα χαρακτηριστικά και συντακτικό, με μόνιμο στόχο να κάνει τα απλά πράγματα εύκολα και τα δύσκολα πράγματα εφικτά.

Παρατηρήσεις :

- **Η C# είναι Case sensitive**

Για να δηλώσουμε την Main() χρησιμοποιήσαμε κεφαλαίο Μ. Θα ήταν λάθος να λέγαμε static void main(). ο compiler εξ ορισμού καταλαβαίνει ότι το βασικό πρόγραμμα βρίσκεται στην μέθοδο Main() και όχι στην main(). Εννοείται πως **int y ≠ int Y**.

- **Keywords**

Στη C# υπάρχουν δεσμευμένες λέξεις-κλειδιά. Δεν μπορούμε να δηλώσουμε πχ μια μεταβλητή με όνομα Console/int/void/class, όπως επίσης δεν μπορούμε να βάζουμε αριθμό μπροστά από μεταβλητή.

Επίσης καλό είναι να αναλύσουμε λίγο την κληρονομικότητα και τους τύπους της που περιέχει η C# καθώς στο 3ο κεφάλαιο θα δούμε αρκετά μέρη του κώδικα του συστήματος όπου βασιζεται πάνω στην κληρονομικότητα των κλάσεων.

Αρχικά πρέπει να αναφέρουμε πως η C# δεν επιτρέπει πολλαπλή κληρονομικότητα (παρά μόνο (*κάπως)* με τα Interfaces). Δηλαδή μία κλάση μπορεί να κληρονομήσει μόνο από μία άλλη κλάση. Υπάρχουν οι κανονικές κλάσεις οι οποίες δημιουργούν από μόνες τους τις λειτουργίες-μεθόδους τους και υπάρχουν και οι Abstract. Οι abstract κλάσεις δεν μπορούν να χρησιμοποιηθούν απευθείας, πρέπει πρώτα σηλαδή να δημιουργηθεί κάποιο "στιγμιότυπο" αυτής της κλάσης, ουσιαστικά να κληρονομηθεί από κάποια άλλη.

Προστασία Πρόσβασης

1) **public** : Όλοι μπορούν να καλέσουν ή να έχουν πρόσβαση
2) **protected** : Μόνο τα μέλη έχουν πρόσβαση
3) **private** : Μόνο τα μέλη και μόνο αυτής της κλάσης
4) **sealed** : Δεν μπορεί να χρησιμοποιηθεί σαν βασική κλάση
5) **internal** : Public πρόσβαση μόνο μέσα στην assembly
6) **protected internal** : Protected στην assembly (σημαίνει protected OR internal)
7) **abstract** : δεν μπορεί να γίνει instantiate, αλλά μπορεί μόνο να κληρονομηθεί

Abstract classes :

1) υποχρεωτικά κληρονομούνται
2) Μπορούν να έχουν υλοποιήσεις μεθόδων, ή σκέτες δηλώσεις
3) Μπορούν να έχουν fields
4) Αν θέλουμε μια κλάση να *μην* μπορεί να κληρονομηθεί, τότε τη χαρακτηρίζουμε ως **sealed.**

Επίσης υπάρχει και ο τύπος Virtual. Η λέξη virtual χρησιμοποιείται σε Methods που ίσως να θέλουμε να κάνουμε override. Αυτό σημαίνει ότι όταν έχουμε μία A κλάση (στην οποία βρίσκεται έστω η public virtual void DoSomething()) την οποία κληρονομεί μία κλάση B, στη δεύτερη κλάση έχουμε τη δυνατότητα να αλλάξουμε την υλοποίηση της DoSomething(). Έστω ότι έχουμε μία κλάση Vehicle όπως παρακάτω:

```
1.  public class Vehicle
2.      {
3.          public virtual void Accelerate()
4.          {
5.              Console.WriteLine("Vehicle accelerating..");
6.          }
7.          public void Stop()
8.          {
9.              Console.WriteLine("Vehicle is stopping now..");
10.         }
11.     }
```

Δημιουργούμε ακόμα μία κενή κλάση με όνομα Car και το μόνο που κάνουμε είναι να κληρονομήσουμε την κλάση Vehicle. Αν δημιουργήσουμε ένα αντικείμενο τύπου Car στη Main τότε θα δούμε ότι έχουμε τη δυνατότητα να καλέσουμε τις μεθόδους Accelerate() και Stop().

```
1.  Car myCar = new Car();
2.              myCar.Accelerate();
3.              myCar.Stop();
4.  //Output:
5.  //Vehicle accelerating..
6.  //Vehicle is stopping now..
```

Η χρήση της λέξης virtual πριν από τη Method Accelerate() δηλώνει ότι μπορούμε να την κάνουμε override. Αυτό σημαίνει ότι θα αλλάξει η υλοποίηση της Accelerate() (δηλαδή ο κώδικας που θα εκτελείται).

Ανοίγοντας την κλάση Car γράφουμε τις λέξεις “public override” και βλέπουμε ότι το Intellisense μας δίνει την δυνατότητα να «επιλέξουμε» μία από τις μεθόδους Accelerate(), Equals(object obj), GetHashCode() και ToString() *. Επιλέγοντας την method Accelerate() το Visual Studio συμπληρώνει μόνο του την υλοποίηση της Accelerate(), συγκεκριμένα:

```
1.  public override void Accelerate()
2.          {
3.              base.Accelerate();
4.          }
```

Το base.Accelerate() σημαίνει ότι θα εκτελεστεί ο κώδικας που έχει υλοποιηθεί στην κλάση που κληρονομούμε (στην προκειμένη περίπτωση Vehicle). Αντικαθιστούμε την γραμμή base.Accelerate() με Console.WriteLine(“Car is accelerating..”); Τρέχοντας το πρόγραμμά μας έχουμε τα παρακάτω outputs:

```
1.  //Car is accelerating now..
2.  //Vehicle is stopping now..
```

Σημείωση : Η διαφορά μεταξύ των abstract και virtual methods είναι ότι τις πρώτες υποχρεούμαστε να τις υλοποιούμε όταν τις κληρονομούμε ενώ τις virtual έχουμε τη δυνατότητα να γράψουμε μία διαφορετική υλοποίηση αλλά δεν είναι υποχρεωτικό.

1.3.2 Σύντομο Ιστορικό Της C#

Οι αρχικοί δημιουργοί της γλώσσας C# ήταν οι Andrers Hejlsberg, Scott Wiltamuth και Peter Golde. Η πρώτη ευρέως διανεμημένη υλοποίηση της C# έγινε από την Microsoft τον Ιούλιο του 2000. Παρόλο που η υλοποίηση τηε C# από τη Microsoft βασίζεται στο CLI για υποστήριξη βιβλιοθήκης και runtime, άλλες υλοποιήσεις της C# δε χρειάζεται να το κάνουν αυτό, εφόσον υποστηρίζουν έναν εναλλακτικό τρόπο για να παίρνουν στο ελάχιστο CLI στοιχεία που απαιτούνται.

Ο βασικός στόχος της διαδικτυακής γλώσσας C# ήταν να είναι απλή, μοντέρνα, γενικής χρήσης, αντικειμενοστραφής γλώσσα προγραμματισμού. Η γλώσσα και οι υλοποιήσεις της θα πρέπει να παρέχουν υποστήριξη για software engineering, όπως έλεγχος τύπων, έλεγχος ορίων πινάκων, προσπάθειες χρήσης μη αρχικοποιημένων μεταβλητών, αυτόματη συλλογή gabage και άλλες. Η γλώσσα αυτή προορίζεται ουσιαστικά για χρήση στην ανάπτυξη software components κατάλληλων για διανεμημένα (distributed) περιβάλλοντα.

Η C# αποσκοπεί στο να είναι κατάλληλη για σύνταξη προγραμμάτων, τόσο για hosted όσο και για embedded συστήματα, ποικίλλοντας από τα πιο μεγάλα που χρησιμοποιούν εξειδικευμένα λειτουργικά συστήματα εώς τα πιο μικρά που χρησιμοποιούν απλούστερες λειτουργίες.

Η καλή σχέση της C# με το βασικό περιβάλλον .NET της Microsoft και ιδιαίτερα οι ευκολίας που παρουσιάζει εκεί όπου η Java δυσκολεύεται (π.χ. I/O) τις δίνουν ελπίδες, αν όχι σταδιακής επικράτησης, τουλάχιστον πολύχρονης συμβίωσης με τη μητέρα των διαδικτυακών προϊόντων, την Java.

1.3.3 Χαρακτηριστικά Της C#

1 Οικεία :

Η C# είναι μία διαχρονική συνέχεια της Java. Τα στοιχεία όμως και οι δυνατότητες που παρέχει, προέρχονται στην ουσία από άλλες γλώσσες προγραμματισμού, όπως Smalltalk, Fortran, Eiffel, Ada, Lisp, C++ και Cedar. Αυτό σημαίνει πως η C# δεν έιναι μόνο οικεία σε προγραμματιστές της Java, αλλά και σε όλους σχεδόν τους προγραμματιστές.

2 Απλή :

Η C# είναι μια απλή γλώσσα που μπορεί να χρησιμοποιηθεί χωρίς εντατική εκμάθηση, ενώ ταυτόχρονα είναι εναρμονισμένη με σύγχρονες προγραμματιστικές πρακτικές. Οι θεμελιώδεις αρχές της γλώσσας μπορούν να κατανοηθούν γρήγορα κάτι που σημαίνει ότι οι προγραμματιστές θα είναι παραγωγικοί σε σύντομο χρονικό διάστημα. Η C# έχει σχεδιαστεί έτσι ώστε να μειώνεται η πιθανότητα πρόκλησης λαθών από την πολυπλοκότητα του κώδικα, αφού τη μειώνει σε μεγάλο βαθμό με το απλουστευμένο συντακτικό της και την οργάνωση κώδικά της.

3 Κατανεμημένη :

Η C# επιτρέπει την προσπέλαση αντικειμένων που βρίσκονται σε διάφορες κορυφές του Διαδικτύου. Επίσης μπορεί να χρησιμοποιηθεί για δημιουργία προγραμμάτων που εκτελούνται σε επίπεδο Server και εξυπηρετούν μία ομάδα πελατών (clients). Τέτοια προγράμματα ονομάζονται servlets και υποστηρίζουν ολοκληρωμένα συστήματα στον Παγκόσμιου Ιστό

4 Αντικειμενοστραφής :

Η C# από τα θεμέλια της σχεδιάστηκε να είναι αντικειμενοστραφής. Ο αντικειμενοστραφής προγραμματισμός επικράτησε σαν προγραμματιστικό πρότυπο την προηγούμενη δεκαετία και παραμένει στις πρώτες προτιμήσεις των προγραμματιστών. Οι ανάγκες για κατανεμημένα συστήματα πελάτη-εξυπηρετητή συμπίπτουν με την ενθυλάκωση και την ανταλλαγή μηνυμάτων που είναι βασικά χαρακτηριστικά του αντικειμενοστραφούς προγραμματισμού. Κατά πολλούς ειδικούς στις γλώσσες προγραμματισμού, η επιτυχής λειτουργία των προγραμματιστικών συστημάτων σε δικτυακά περιβάλλοντα αυξανόμενης πολυπλοκότητας βασίζεται στην αντικειμενοστρέφεια. Η C# παρέχει μια ξεκάθαρη και αποδοτική αντικειμενοστραφή πλατφόρμα παρέχοντας στους προγραμματιστές μια συλλογή βιβλιοθηκών δοκιμασμένων αντικειμένων που παρέχουν λειτουργικότητα που ποικίλει από απλούς τύπους δεδομένων, σε διεπαφές εισόδου/εξόδου ή δικτυακές και εργαλεία για τη δημιουργία παραθυρικών εφαρμογών. Αυτές οι βιβλιοθήκες μπορούν να προσαρμοστούν στις ανάγκες του προγραμματιστή.

Επιπρόσθετα η C# υποστηρίζει και τον προγραμματισμό βασισμένο σε components (component-based programming) ο οποίος επιτρέπει τον προσδιορισμό αυτόνομων μονάδων λειτουργικότητας (components) που είναι απομονωμένα και τεκμηριωμένα, παρουσιάζοντας ένα μοντέλο με ιδιότητες, μεθόδους, events και μεταδεδομένα για το component. Η C# υποστηρίζει αυτά τα χαρακτηριστικά άμεσα κάνοντας έτσι τη διαδικασία δημιουργίας και χρήσης των components πολύ εύκολη.

5 Ερμηνευόμενη :

Η C# μπορεί να μεταφραστεί και σε γλώσσα μηχαν΄ς οποιουδήποτε CPU με τη χρήση ενός JIT (Just-in-time compiler).

6 Ασφαλής :

Η C# είναι από τις πλέον ασφαλείς γλώσσες προγραμματισμού. Διαθέτει ειδικό υποσύστημα ελέγχου ενδιάμεσου κώδικα με άμεσο στόχο τον εντοπισμό πιθανών σημείων αυτοκατάρρευσης του προγράμματος αλλά και κεντρικών καταρρεύσεων του όλου υπολογιστικού συστήματος (project).

7 Δυναμική :

Η C# είναι εφοδιασμένη με αυτόματη τεχνική που διευθετεί τη σύνδεση όλων των αναγκαίων αρχείων που συγκροτούν μία εφαρμογή τη διάρκεια της εκτέλεσης του προγράμματος. Αλλά και σε επίπεδο βιβλιοθηκών η C# είναι επίσης δυναμική, διότι χρησιμοποιεί την έννοια του πακέτου (Namespace).

8 Συμπαγής :

Έχει ισχυρό, αναγκαστικά σύστημα δηλώσεων, ισχυρό σύστημα τύπων και στον ενδιάμεσο κώδικα κάθε παράμετρος παίρνει τελικά το ορθό πεδίο ορισμού της ανεξάρτητα από την αντίστοιχη δήλωση του προγραμματιστή στο πηγαίο πρόγραμμα. Η C# ακολούθησε την κλασική τακτική της Java, όπου η χρησιμοποίηση δεικτών (pointers) είναι σχεδόν μηδενική, κάνοντας έτσι εύκολη τη ζωή των προγραμματιστών χωρίς κίνδυνο καταστροφής του χάρτη μνήμης σε μία δεδομένη στιγμή.

9 Μεταφέρσιμη :

Ο κώδικας της C# είναι ενδιάμεσος και ανεξάρτητος από συγκεκριμένες αρχιτεκτονικές ή δίκτυα υπολογιστών. Τα προγράμματα της C# είναι μεταφέρσιμα για όλες τις γνωστές σήμερα αρχιτεκτονικές και όλα τα γνωστά λειτουργικά συστήματα της Microsoft. Σύμφωνα με αυτήν την έννοια η C# είναι μία γλώσσα ανοικτής αρχιτεκτονικής και μεταφέρσιμη, διότι όπως πιστεύουν οι δημιουργοί της : « write once, run anywhere ».

10 Υψηλής Απόδοσης :

Η C# διαθέτει δύο ‘μυστικά’ όπλα για να καλυτερεύσει και στο θέμα της απόδοσης τα προγράμματά της. Το πρώτο όπλο της είναι η υποστήριξη πολλαπλών νημάτων (multi Threaded). Το δεύτερο όπλο της, για να καλυτερεύσει την απόδοσή της, είναι ο μεταγλωττιστής JIT, όπου στην ουσία μετατρέπεται ο ενδιάμεσος κώδικας και έχουμε τελικά εκτελέσιμο κώδικα.

11 Υποστήριξη Εφαρμογών :

Η C# προσφέρει πλήρη υποστήριξη σε SQL βάσεων δεδομένων που βρίσκονται διανεμημένες ή μη σε επίπεδο WEB. Επίσης, διαθέτει μία πασίγνωστη βιβλιοθήκη (Application Programming Interface - API), για υποστήριξη δομών δεδομένων, για πρωτόκολλα ασφάλειας προσπέλασης RDBMS, για επικοινωνία κόμβων στο δίκτυο ή και για υποστήριξη σύνθετων οπτικών καταστάσεων (multimedia).

Κεφάλαιο 2 – Ανάπτυξη Συστήματος Διαχείρισης και Αναζήτησης Αθλητικών Events με Χρήση .NET Framework (ASP.NET MVC και C#)

2.1 Σκοπός Του Συστήματος

Σκοπός της υλοποίησης της εφαρμογής αυτής είναι η ανάπτυξη ενός site μέσω του οποίου θα μπορεί ο διαχειριστής να καταχωρήσει και να επεξεργαστεί αθλητικά events καθώς θα μπορεί και κάποιος ως απλός επισκέπτης να ανατρέξει σε αυτά για να δει σχετικές πληροφορίες και να αναζητήσει κάποιο συγκεκριμένο που τον ενδιαφέρει. Το site αυτό επικεντρώνεται κυρίως σε αγώνες δρόμου (όλως των ειδών) και απευθύνεται κυρίως σε ανθρώπους όπου ασχολούνται με το συγκεκριμένο άθλημα (δρομείς). Έτσι λοιπόν, όπως θα αναλύσουμε και στις επόμενες ενότητες στόχος μας ήταν να γίνει όσο πιο απλή γίνεται η διαδικασία ενημέρωσης αυτής της ομάδας ανθρώπων για όλους τους αγώνες που διοργανώνονται (προς το παρόν στα πλαίσια της Ελλάδας) και να είναι εύκολη η εύρεση πληροφοριών τους (όπως πχ η ημ/νία διεξαγωγής, η τοποθεσία, η απόσταση (σε χλμ), καθώς και διάφορες άλλες σχετικές πληροφορίες). Έχοντας λοιπόν "μαζεμένα" σε ένα σημείο όλα τα events είναι εύκολο για κάποιον να ανατρέξει εκεί και να κάνει μία στοιχειώδη αναζήτηση με βάση τα κριτήρια τα οποία αυτός θα επιλέξει και να έχει μία άμεση ενημέρωση.

Για τη σωστή λειτουργία της εφαρμογής έγινε ο διαχωρισμός της σε δύο επιμέρους projects (τα οποία θα δούμε αναλυτικά στη συνέχεια). Το πρώτο κομμάτι **(εφαρμογή διαχειριστή – το Back-end του συστήματος)** είναι για την καταχώρηση και την διαχείριση των events, στο οποίο έχει access μόνο ο διαχειριστής της εφαρμογής, όπου καλείται να καταχωρήσει όλες τις πληροφορίες που αφορούν τον κάθε αγώνα. Ενώ αντίστοιχα το δεύτερο κομμάτι **(εφαρμογή χρηστών – το Front-end του συστήματος)** είναι αυτό το οποίο βλέπουν όλοι οι χρήστες. Τα δύο αυτά μέρη επικοινωνούν μεταξύ τους μέσω της βάσης δεδομένων (data base), όπου στο πρώτο μέρος ουσιαστικά γίνονται οι εγγραφές στην βάση ενώ στο δεύτερο μέρος γίνονται μόνο αναφορές ώστε να εμφανιστούν τα αντίστοιχα δεδομένα.

2.2 Απαιτήσεις – Προδιαγραφές

2.2.1. Εισαγωγή Στις Έννοιες

Σε αυτήν την ενότητα θα δούμε τις απαιτήσεις και της προδιαγραφές του συστήματος. Η ανάλυσή τους έγινε σε συνεργασία με τον διαχειριστή της εφαρμογής (τον πελάτη μας).Ο πελάτης είναι ο πρώτος που εντοπίζει ένα πρόβλημα και εκτιμά ότι υπάρχει ανάγκη λύσης του. Το πρόβλημα μπορεί να αφορά τον τρόπο με τον οποίο γίνονται κάποια πράγματα καθημερινά ή ένα παλαιότερο πληροφοριακό σύστημα που θα πρέπει να επεκταθεί ή να αντικατασταθεί. Ο πελάτης δεν είναι πάντα και χρήστης του συστήματος. Ο πελάτης χρησιμοποιεί την φυσική γλώσσα για να περιγράψει το πρόβλημα και τις υπηρεσίες που θα παρέχει το νέο πληροφοριακό σύστημα καθώς και τις συνθήκες που θα πρέπει να ικανοποιεί. Ουσιαστικά αποτελεί το πρώτο από τα ενδιαφερόμενα μέρη που περιγράφει τις απαιτήσεις από το σύστημα και το λογισμικό. Οι απαιτήσεις του πελάτη βέβαια διαφέρουν πάρα πολύ από τις απαιτήσεις του συστήματος.

Οι απαιτήσεις ουσιαστικά είναι μία σύντομη περιγραφή του συστήματος, απαντάει δηλαδή στο ερώτημα <<Τι κάνει το σύστημα ? >>, αλλά δεν περιλαμβάνει τον τρόπο υλοποίησής του. Αποτελεί μια προσπάθεια αποσαφήνισης του προβλήματος που στοχεύει να λύσει το πληροφοριακό σύστημα που θα δημιουργηθεί και όχι στην περιγραφή κάποιας λύσης. Οι απαιτήσεις καταγράφονται και τεκμηριώνονται με τέτοιο τρόπο ώστε να είναι κατανοητές από όλα τα ενδιαφερόμενα μέρη στην ανάπτυξη του συστήματος. Επιπλέον οι απαιτήσεις είναι μια σημαντική σταθερά που παραμένει ως ένας παράγοντας αξιολόγησης καθ 'όλη τη διάρκεια του κύκλου ζωής ενός προϊόντος. Είναι δηλαδή ένα είδος συμβολαίου που ελέγχεται ακόμα και μετά το τέλος της ανάπτυξης του προϊόντος.

Οι απαιτήσεις διακρίνονται σε δυο μεγάλες κατηγορίες. Στις λειτουργικές και στις μη λειτουργικές απαιτήσεις.

- Οι λειτουργικές απαιτήσεις περιγράφουν αναλυτικά την αλληλεπίδραση του συστήματος και του περιβάλλοντος. Πολλές φορές οι λειτουργικές απαιτήσεις αποδίδουν την έννοια του

επεξεργαστή πληροφορίας σε αυτό που περιγράφουν. Έτσι αναφέρονται στην είσοδο ή αλλιώς στα ερεθίσματα που μπορεί να δεχτεί το σύστημα και στον τρόπο που αντιδρά σε αυτά τα ερεθίσματα. Το σύστημα μπορεί να διενεργεί κάποιες επεξεργασίες ή να μεταβάλλει την κατάστασή του. Μετά την επεξεργασία προκύπτει η έξοδος του συστήματος δηλαδή τα επιθυμητά αποτελέσματα που παράγει σαν απόκριση στο ερέθισμα - είσοδο που δέχθηκε.

- Οι απαιτήσεις που δεν αναφέρονται σε κάποια λειτουργία του συστήματος ονομάζονται μη λειτουργικές. Οι μη λειτουργικές απαιτήσεις περιγράφουν κάποιες προδιαγραφές που πρέπει να έχει το σύστημα που ουσιαστικά πλαισιώνουν τις λειτουργικές απαιτήσεις. Είναι πολύ σημαντικές καθώς θέτουν κάποιους περιορισμούς στις επιλογές που έχουν οι κατασκευαστές στα στάδια σχεδιασμού και υλοποίησης. Οι μη λειτουργικές απαιτήσεις αναλύονται επιμέρους σε κατηγορίες. Οι πιο σημαντικές κατηγορίες είναι η ασφάλεια, η απόδοση, η αξιοπιστία, η χρηστικότητα και η υποστήριξη.

Η ανάπτυξης πληροφορικών συστημάτων και συγκεκριμένα η έννοια της ανάλυσης αναφέρεται στην μελέτη ενός τομέα της οικονομίας ή μιας εφαρμογής, η οποία οδηγεί στην δημιουργία προδιαγραφών για ένα νέο σύστημα, το οποίο θα οδηγήσει στην υλοποίηση του αντίστοιχου πληροφοριακού συστήματος. Το τελευταίο στάδιο της διαδικασίας προσδιορισμού απαιτήσεων είναι η προδιαγραφή απαιτήσεων. Σε αυτό το στάδιο καθορίζονται οι προδιαγραφές του υπό μελέτη συστήματος και καταγράφονται στο πιο σημαντικό παραδοτέο υλικό της ανάλυσης, το λεγόμενο Κείμενο Προδιαγραφών των Απαιτήσεων ή Κείμενο του Στόχου (Target Document). Επίσης απαραίτητο είναι να συνοδεύεται από ένα σύνολο μοντέλων αποτύπωσης του συστήματος σε μορφή Διαγραμμάτων Ροής Δεδομένων (Data Flow Diagrams), Οντοτήτων – Συσχετίσεων (Entity Relation) μαζί με τους Πίνακες Απόφασης/ Δένδρα Απόφασης και το Λεξικό Δεδομένων (Data Dictionary).

Οι προδιαγραφές αποτελούν τη δομημένη και λεπτομερή περιγραφή των απαιτήσεων του συστήματος, η οποία γίνεται με τη μορφή γραπτού λόγου και συνοδεύεται με τη μορφή διαγραμμάτων και πινάκων. Είναι σημαντικό να αναφερθεί ότι ελλείψεις, αστοχίες και λάθη όσων αναφέρονται σε αυτό θα μεταφερθούν σε όλη την υπόλοιπη διαδικασία κατασκευής του συστήματος και ασφαλώς θα έχουν επιπτώσεις στο τελικό προϊόν.

2.2.2. Απαιτήσεις – Προδιαγραφές Του Συστήματος

Η εφαρμογή αυτή παρουσιάζει μεγάλο πλήθος λειτουργικών και μη- λειτουργικών απαιτήσεων, οι οποίες δεν είναι δυνατόν να καταγραφούν και να παρουσιαστούν με πλήρη λεπτομέρεια στο παρόν παραδοτέο, καθώς πολλές από αυτές τις απαιτήσεις έγιναν εμφανείς κατά την διαδικασία ανάπτυξης του συστήματος σύμφωνα με την ακολουθούμενη μεθοδολογία κύκλου ζωής του υπό ανάπτυξη λογισμικού η οποία είναι επαναληπτική.

1. Εφαρμογή Διαχειριστή

- **<u>Λειτουργικές Απαιτήσεις</u>**
  - Η είσοδος του διαχειριστή στην εφαρμογή
  - Η προσθήκη/επεξεργασία/διαγραφή κάθε event
  - Να μπορεί να αποθηκεύει τις πληροφορίες που εισάγει ο διαχειριστής για το κάθε event
  - Να μπορεί να φέρνει τα αποτελέσματα από κάθε αναζήτηση που θα πραγματοποιείται

- **<u>Μη Λειτουργικές Απαιτήσεις</u>**
  - Ο χρόνος απόκρισης της εφαρμογής δεν θα πρέπει να είναι μεγάλος
  - Μόνο οι διαχειριστές θα μπορούν να έχουν πρόσβαση σε αυτή την εφαρμογή
  - Η εφαρμογή θα πρέπει να υποστηρίζει πολλές φυσικές γλώσσες
  - Θα πρέπει να υπάρχει η δυνατότητα επεκτασιμότητας και προσθήκη νέων χαρακτηριστικών
  - Η εφαρμογή πρέπει να είναι μεταφέρσιμη (να παίζει και σε Desktop αλλά και σε Mobile)

2. Εφαρμογή Χρηστών

- **Λειτουργικές Απαιτήσεις**
  - Να φέρνει και να εμφανίζει τα χαρακτηριστικά για κάθε event
  - Να μπορεί να φέρει τα αποτελέσματα από κάθε αναζήτηση που θα πραγματοποιείται από τους χρήστες
  - Να μπορεί να προσθέσει/αφαιρέσει στα "Αγαπημένα" τις επιλογές του αντίστοιχου χρήστη
  - Να μπορεί να εμφανίσει τα σημεία των αγώνων στον χάρτη
  - Να μπορεί να εμφανίσει στο ημερολόγιο τα events όπου αντιστοιχούν στην κάθε ημέρα
  - Να αφαιρεί αυτόματα από το ημερολόγιο τα events μετά το πέρασμα τις ημερομηνίας διεξαγωγής τους

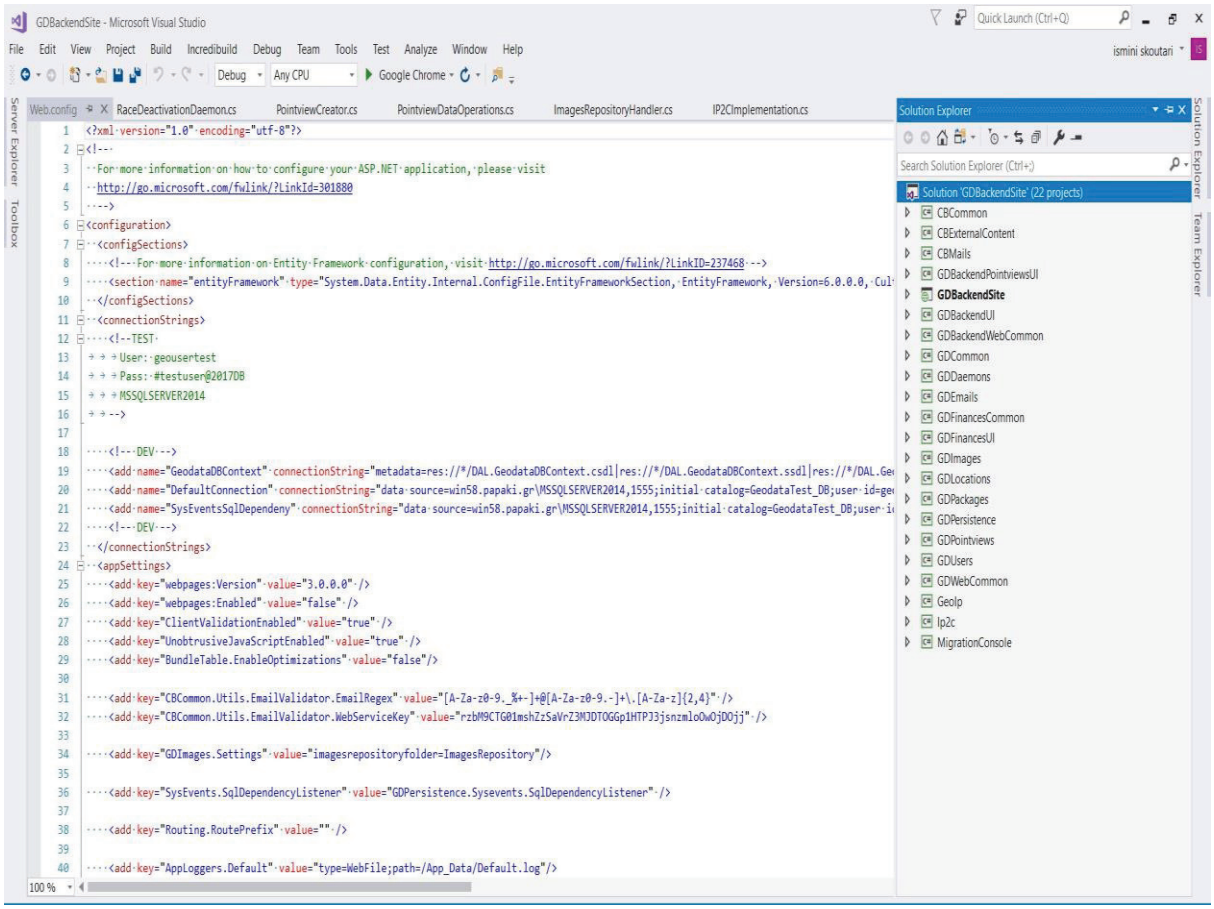
- **Μη Λειτουργικές Απαιτήσεις**
  - Η εφαρμογή πρέπει να παρέχει απόκριση πραγματικού χρόνου
  - Η εφαρμογή πρέπει να μπορεί να εξυπηρετεί αποδοτικά πολλούς χρήστες ταυτόχρονα
  - Η εφαρμογή πρέπει να είναι φιλικό στη χρήση
  - Μπορεί να έχει πρόσβαση οποιοσδήποτε σε αυτήν την εφαρμογή
  - Η εφαρμογή θα πρέπει να υποστηρίζει πολλές φυσικές γλώσσες
  - Θα πρέπει να είναι απλό και εύχρηστο
  - Θα πρέπει να υπάρχει η δυνατότητα επεκτασιμότητας και προσθήκη νέων χαρακτηριστικών
  - Η εφαρμογή πρέπει να είναι μεταφέρσιμη (να παίζει και σε Desktop αλλά και σε Mobile)

2.3 Τα Μέρη Από Τα Οποία Αποτελείται Το Σύστημα

Όπως αναφέραμε και προηγουμένως το σύστημά μας αποτελείται από δύο εσωτερικές εφαρμογές. Την εφαρμογή που απευθύνεται στους διαχειριστές του συστήματος (back-end) και την εφαρμογή η οποία απευθύνεται στους χρήστες του συστήματος (front-end). Οι δύο αυτές εφαρμογές επικοινωνούν μεταξύ τους μέσω της Βάσης Δεδομένων, όπου στην πρώτη περίπτωση γίνονται κυρίως οι εγγραφές σε αυτήν αλλά και μικρές αναζητήσεις, ενώ στην δεύτερη γίνονται μόνο αναζητήσεις προς την Βάση Δεδομένων. Αυτός ο διαχωρισμός έγινε προκειμένου να μπορέσει να γίνει πιο εύκολη η υλοποίηση του συνολικού συστήματος, καθώς και για τον παραλληλισμό όσον αφορά την ανάπτυξη αυτών των δύο εφαρμογών. Αυτό είχε ως αποτέλεσμα την μείωση του χρόνου παράδοσης του έργου, την απλή υλοποίησή του, καθώς καθιστά εύκολη και την συντήρηση του συστήματος. Επίσης καθιστά εύκολη και την επεκτασιμότητα του συστήματος μας, για τυχόν μελλοντικές αλλαγές – προσθήκες κάποιων λειτουργιών. Ας δούμε λίγο πιο αναλυτικά όμως τα συστατικά της κάθε εφαρμογής.

2.3.1. Το Back-End Του Συστήματος

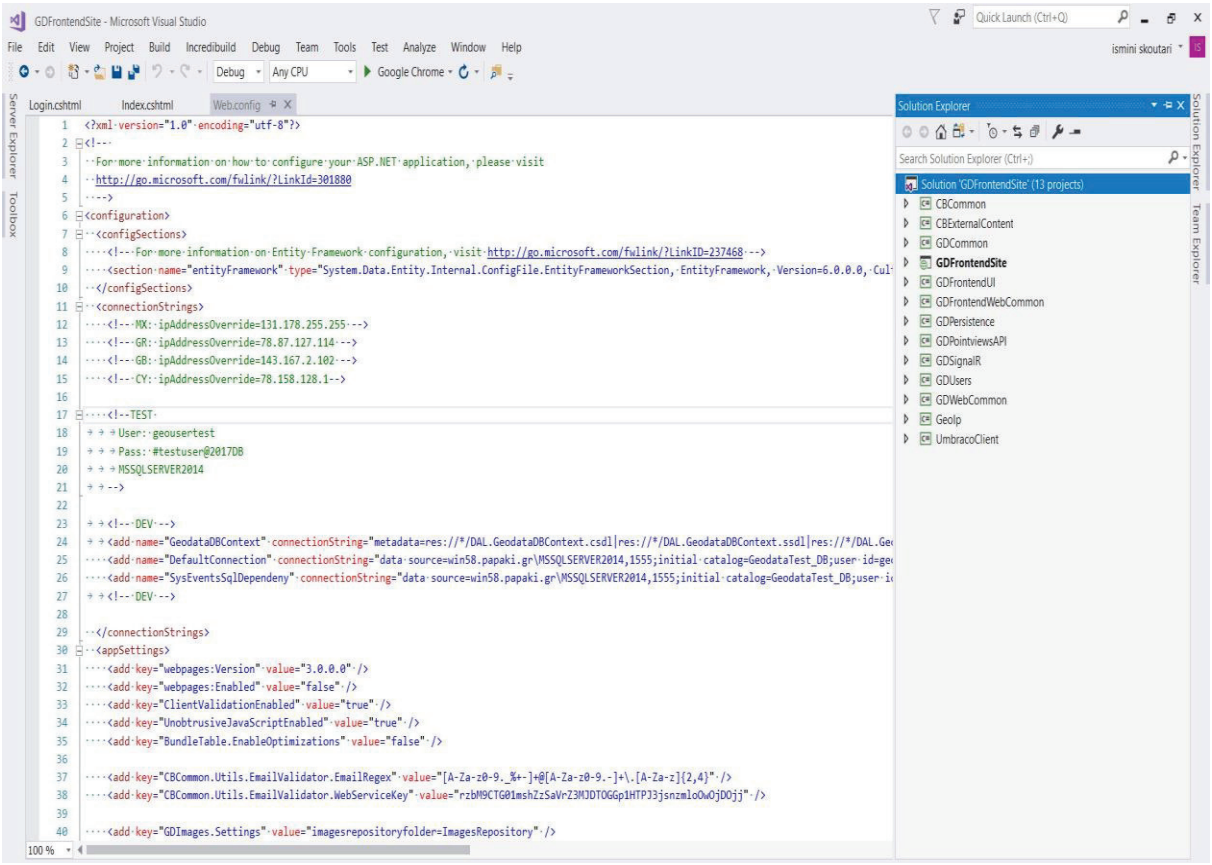
Το Back-End του συστήματος όπως αναφέραμε ήδη αφορά τους διαχειριστές του συστήματος. Εδώ λοιπόν καλείτε ο εκάστοτε διαχειριστής να καταχωρήσει τα events που θέλει να διαθέσει, καθώς να περάσει και όλες τις σχετικές με αυτά πληροφορίες. Η διαδικασία που καλείται να ακολουθήσει είναι σχετικά απλή. Αρχικά επιλέγει την κατηγορία στην οποία ανήκει το event όπου θέλει να καταχωρήσει, στην συνέχεια επιλέγει στον χάρτη το κύριο σημείο που αφορά το event (τις περισσότερες φορές αντιστοιχεί στο σημείο εκκίνησης του αγώνα) και στην συνέχεια περνάει όλες τις σχετικές πληροφορίες όπου επιθυμεί (να σημειώσουμε πως δεν είναι όλα τα πεδία από τις έξτρα πληροφορίες υποχρεωτικά, οπότε μπορεί να αφήσει και κενά κάποια πεδία όπου δεν θέλει να καταχωρήσει κάποια πληροφορία).



Εικόνα 2.1 Το περιβάλλον του Back-End του Συστήματος

2.3.2. Το Front-End Του Συστήματος

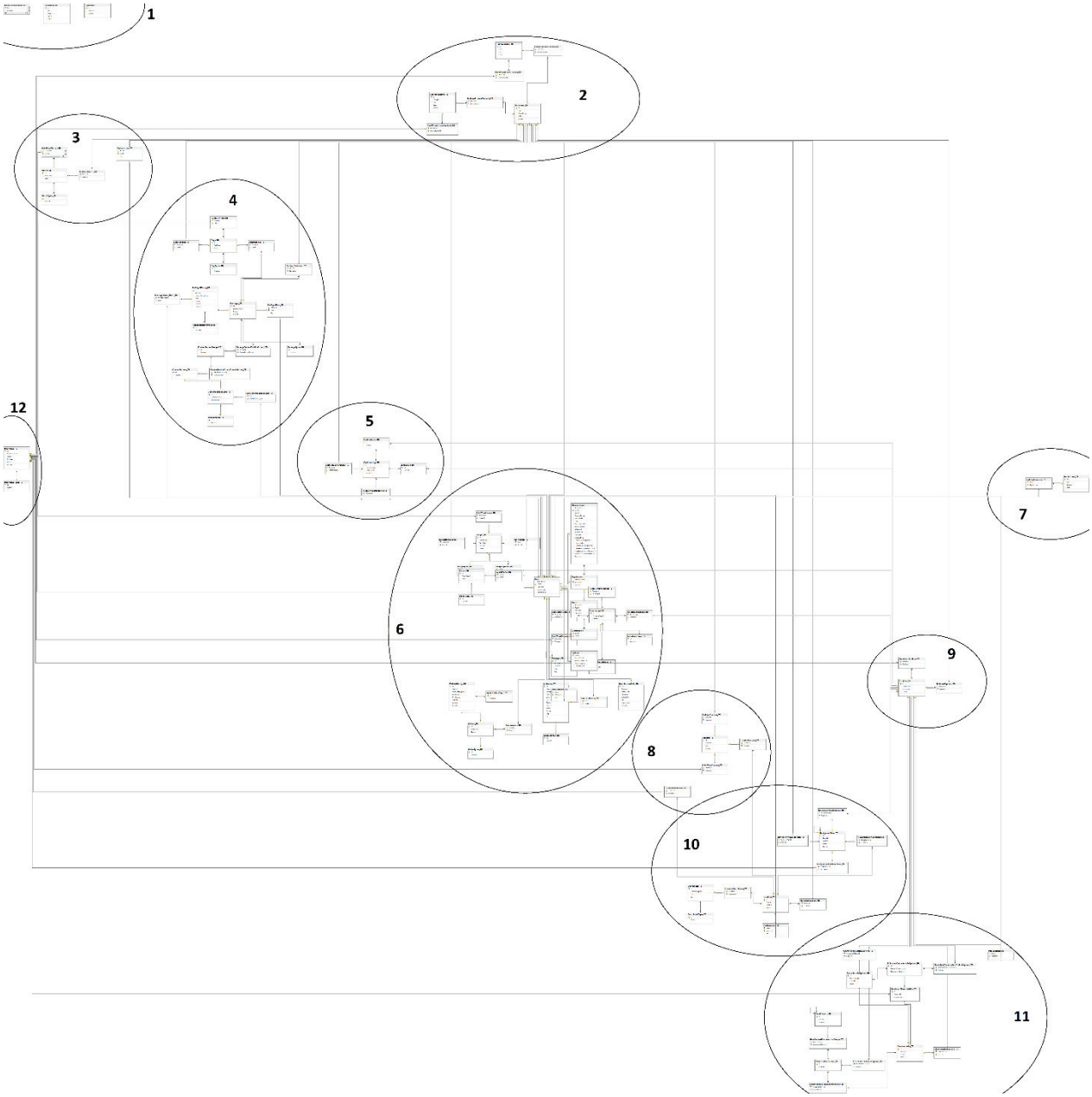
Από την άλλη μεριά υπάρχει και το Front-End του συστήματος όπου αυτό αναφέρεται στους χρήστες του συστήματος. Εδώ λοιπόν βρίσκεται η "καρδιά" του συστήματος αφού πρόκειται για την κύρια εφαρμογή που έχει άμεση επαφή με του χρήστες. Πρόκειται για την εφαρμογή στην οποία έχουν όλοι πρόσβαση και μπορούν να μπουν να βρουν όλα τα events όπου έχουν καταχωρηθεί από το Back-End που αναλύσαμε πριν. Έτσι λοιπόν φτάνουμε και στην σύνδεση αυτών των δύο εφαρμογών καθώς υπάρχει άμεση εξάρτηση μεταξύ τους, καθώς το Back-End μόνο του δεν θα μπορούσε να εμφανίσει κάπως τα δεδομένα ενώ το Front-End μόνο του δεν θα είχε δεδομένα να εμφανίσει. Η όλη επικοινωνία που υπάρχει μεταξύ των δύο γίνεται μέσω της Βάσης Δεδομένων όπου στην πρώτη περίπτωση καταχωρούμε τα δεδομένα και στην δεύτερη τα καλούμε ώστε να τα εμφανίσουμε στην τελική οθόνη του χρήστη. Η απεικόνιση γίνεται με πολλούς τρόπους, είτε με την εμφάνιση όλων των events πάνω στον χάρτη, είτε στο ημερολόγιο όπου είναι διαθέσιμο είτε στην λίστα που υπάρχει κάτω από τον χάρτη όπου εμφανίζει τα events από το πιο πρόσφατο ως την πιο παλιά καταχώρηση. Επίσης είναι διαθέσιμη και η λειτουργία της αναζήτησης όπου μπορεί ο κάθε χρήστης να επιλέξει τα events που θέλει με βάση κάποια συγκεκριμένα κριτήρια.



Εικόνα 2.2 Το περιβάλλον του Front-End του Συστήματος

2.4 Σχεδίαση Του Συστήματος

2.4.1. Μοντέλο (Ανάλυση) Κλάσεων – Διάγραμμα Βάσης Δεδομένων



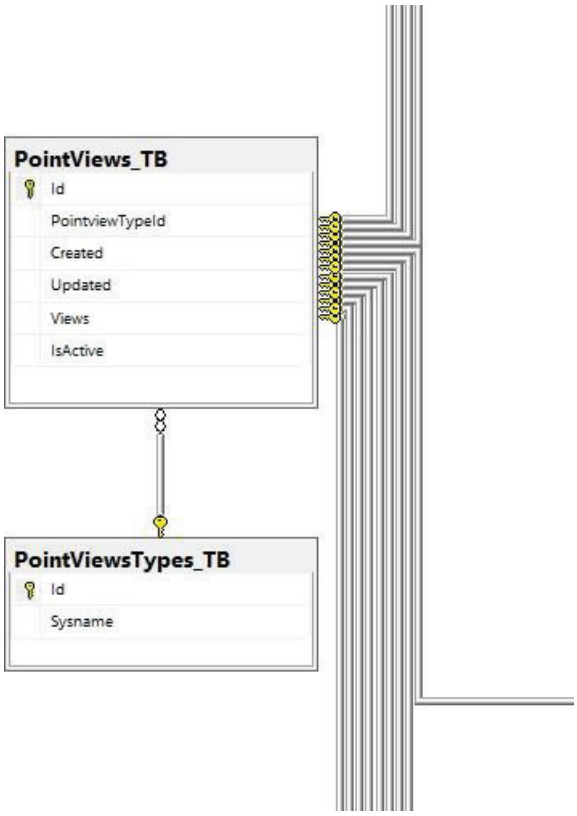
Εικόνα 2.3 Πλήρες Διάγραμμα Βάσης Δεδομένων

2.4.2. Περιγραφή Κλάσεων

Πριν ξεκινήσουμε την ανάλυση της βάσης και κατά συνέπεια των κλάσεων του συστήματος καλό είναι να σημειωθεί πως ακολουθείται μία συγκεκριμένη δομή για την δημιουργία των πινάκων. Όπως είδαμε και στην εικόνα 2.3 η βάση μας αποτελείται από κάποιες ενότητες όπου αντιστοιχούν με τη σειρά τους σε κάποια από τις λειτουργίες – χαρακτηριστικά του συστήματος. Κάθε μία από αυτές τις ενότητες λοιπόν περιέχει έναν πίνακα ο οποίος ονομάζεται “(όνομαΕνότητας)Types_TB”. Σε αυτόν τον πίνακα καταχωρούνται οι τύποι οι οποίοι είναι διαθέσιμοι για κάθε ενότητα όπου στον κάθε έναν στην συνέχεια αντιστοιχίζεται και ένα μοναδικό Id ώστε να μπορεί να γίνει και πιο εύκολη η αναφορά σε αυτά. Επίσης υπάρχει και η ενότητα με τα Captions η οποία περιέχει ότι έχει να κάνει με τα λεκτικά που χρησιμοποιούνται μέσα στο σύστημα, καθώς και σε όλες τις γλώσσες όπου είναι διαθέσιμες. Έτσι κάθε ενότητα πάλι με την σειρά της περιέχει και έναν πίνακα ο οποίος ονομάζεται “(όνομαΕνότητας)Captions_TB”, ο οποίος αναλαμβάνει την σύνδεση της εν λόγω ενότητας με τον πίνακα ο οποίος περιλαμβάνει τα Captoins ώστε να εμφανίζονται σωστά όλα τα λεκτικά στο interface του συστήματος.

Επίσης κάτι ακόμα το οποίο πρέπει να προσδιορίσουμε είναι πως κάποιοι Πίνακες ενώ έχουν δημιουργηθεί δεν αντιστοιχούν σε καμία από τις τρέχουσες λειτουργίες που εκτελεί το σύστημά μας και έχουν δημιουργηθεί για πιθανή μελλοντική χρήση όταν γινόταν ο σχεδιασμός του συστήματος. Οπότε δεν θα σταθούμε σε αυτούς και θα αναλύσουμε όσους χρησιμοποιούνται αυτή τη στιγμή. Ας ξεκινήσουμε λοιπόν την ανάλυση.

- Στο κομμάτι 12 της εικόνας 2.3 υπάρχουν ομαδοποιημένες δύο κλάσεις όπου επικοινωνούν μεταξύ τους και στην συνέχεια με το υπόλοιπο σύστημα.



Εικόνα 2.4 Μέρος της Βάσης Δεδομένων

Στο κάτω μέρος μπορούμε να διακρίνουμε τον Πίνακα PointViewsTypes_TB οποίος αποτελείται από το ID και το Sysname. Το ID χρησιμοποιείται ως key και είναι ένας ακέραιος αριθμός ενώ το Sysname είναι ένας varchar(32) το οποίο δείχνει τον τύπο του κάθε αγώνα. Όπως λοιπόν φαίνεται ο Πίνακας αυτός αναφέρεται στους τύπους όπου μπορούν να διακριθούν τα events που θα καταχωρηθούν.

Results Messages

	Id	Sysname
1	1	BUILDING
2	2	PLOT
3	3	RACE_ROAD
4	4	RACE_TRAIL
5	5	RACE_TRAIL_CITY

Εικόνα 2.5 Δεδομένα του Πίνακα PointViewsTypes_TB

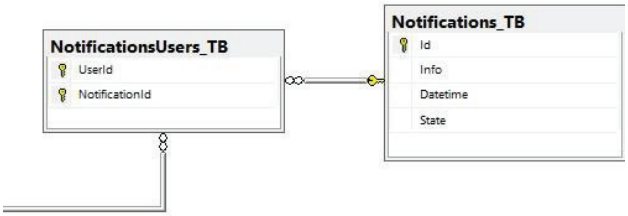
Αντίστοιχα στο πάνω μέρος διακρίνουμε τον Πίνακα PointViews_TB οποίος αποτελείται από 6 πεδία. Ένα χαρακτηριστικό ID, το PointviewTypeId το οποίο παίρνει τιμές σύμφωνα με τον Πίνακα PointViewsTypes_TB, τα πεδία Created και Updated χαρακτηρίζουν την ημ/νία καταχώρησης του event και την ημ/νία όπου ανανεώθηκε κάποια πληροφορία, το πεδίο Views αντιστοιχεί στο πόσα View έχει το αντίστοιχο event και τέλος το πεδίο IsActive το οποίο δείχνει αν το event είναι «ενεργοποιημένο», αν δηλαδή εμφανίζεται στον χάρτη. Με λίγα λόγια στον συγκεκριμένο πίνακα αποθηκεύονται πληροφορίες σχετικά με το κάθε event που καταχωρούμε.

Results Messages

	Id	PointviewTypeId	Created	Updated	Views	IsActive
1	2	4	2016-12-27 19:22:49.317	2016-12-27 19:22:51.527	0	0
2	6	5	2016-12-30 18:54:00.663	2016-12-30 18:54:02.720	0	0
3	7	4	2016-12-30 19:00:42.397	2017-01-01 09:03:23.483	0	0
4	8	5	2016-12-30 20:01:34.787	2017-02-10 06:23:26.860	0	0
5	36	3	2017-01-02 20:52:00.190	2017-01-15 18:12:31.237	0	0
6	37	5	2017-01-07 09:43:16.000	2017-02-25 10:07:01.790	0	0
7	39	4	2017-01-15 19:35:02.717	2017-02-22 16:12:36.930	0	0
8	40	5	2017-01-15 20:38:38.330	2017-01-16 05:24:38.480	0	0
9	41	3	2017-01-16 11:09:31.597	2017-02-22 16:11:04.720	0	0
10	43	3	2017-01-23 08:46:33.767	2017-02-25 18:49:55.730	0	0
11	46	4	2017-02-25 09:58:21.027	2017-02-25 09:58:21.197	0	1
12	47	5	2017-02-25 10:04:35.493	2017-02-25 10:07:40.207	0	0
13	48	4	2017-02-25 10:14:38.003	2017-02-25 10:14:38.020	0	0
14	51	3	2017-02-25 10:43:01.850	2017-02-25 10:43:03.517	0	0

Εικόνα 2.6 Δεδομένα του Πίνακα PointViews_TB

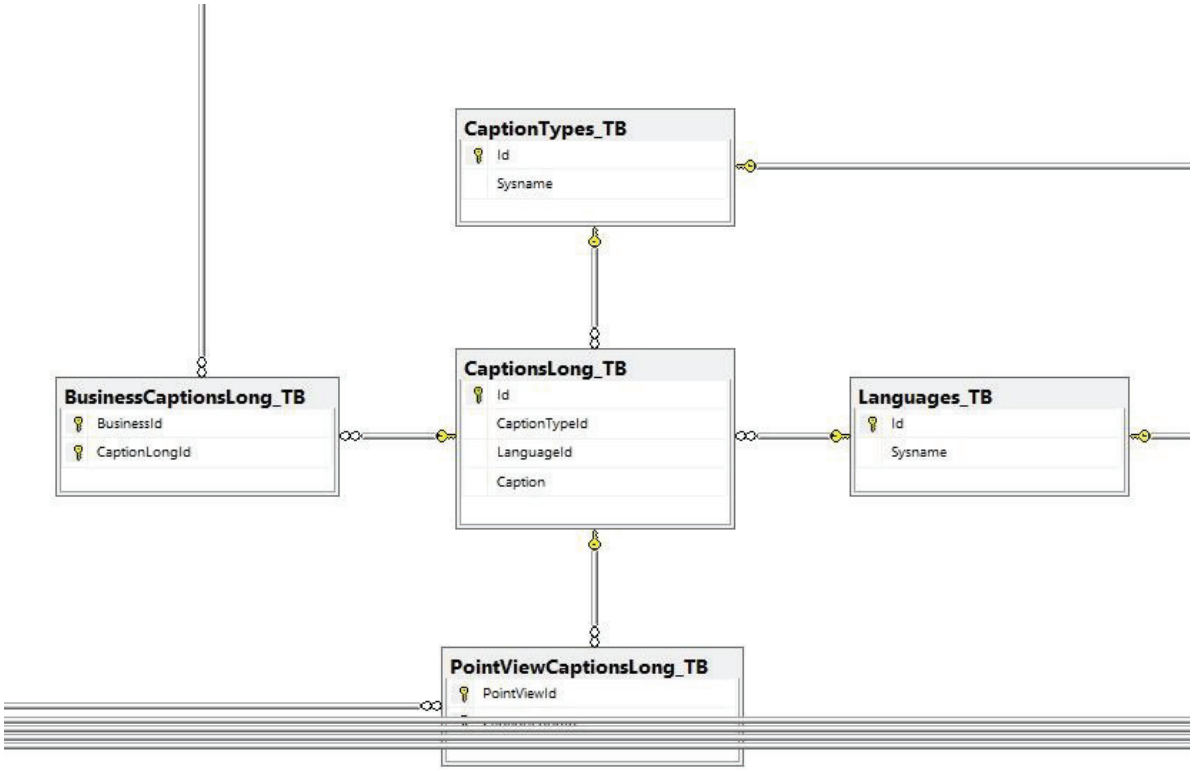
- Στο κομμάτι 7 της εικόνας 2.3 υπάρχουν ομαδοποιημένες δύο κλάσεις όπου επικοινωνούν μεταξύ τους και στην συνέχεια με το υπόλοιπο σύστημα.



Εικόνα 2.7 Μέρος της Βάσης Δεδομένων

Το συγκεκριμένο κομμάτι του Συστήματος δεν έχει τεθεί ακόμα σε λειτουργία και είναι προς υλοποίηση. Ουσιαστικά αφορά το κομμάτι όπου οι χρήστες θα μπορούν να αφήσουν κάποιο comment που όπως μπορούμε να διακρίνουμε και στην εικόνα 2.7 Ο Πίνακας NotificationsUser_TB αποτελείται από δύο πεδία όπου το ένα δηλώνει το ID του χρήστη (UserId) και το δεύτερο δηλώνει το Id του comment (NotificationId) όπου με βάση αυτό στην συνέχεια πηγαίνουμε στον δεύτερο πίνακα της εικόνας 2.7 τον Notifications_TB όπου υπάρχει πάλι αντίστοιχα το Id του comment (Id), το πεδίο Info όπου περιλαμβάνει τα σχόλια που έγραψε ο χρήστης και το πεδίο DateTime όπου αντιστοιχεί στην ημ/νία όπου υλοποιήθηκε το αντίστοιχο notification.

- Στο κομμάτι 5 της εικόνας 2.3 υπάρχουν ομαδοποιημένες πέντε κλάσεις όπου επικοινωνούν μεταξύ τους και στην συνέχεια με το υπόλοιπο σύστημα.



Εικόνα 2.8 Μέρος της Βάσης Δεδομένων

Στην εικόνα αυτή μπορούμε να διακρίνουμε 5 Πίνακες. Ένας εκ των οποίων είναι ο Languages_TB ο οποίος ουσιαστικά περιλαμβάνει τις γλώσσες οι οποίες είναι διαθέσιμες στο σύστημα και αποτελείται από δύο πεδία. Το ένα είναι το Id κάθε γλώσσας και το δεύτερο είναι το Sysname το οποίο είναι το language code της κάθε γλώσσας.

Results Messages

	Id	Sysname
1	1	en-US
2	2	el-GR

Εικόνα 2.9 Δεδομένα του Πίνακα Languages_TB

Στην συνέχεια υπάρχει ο Πίνακας CaptionTypes_TB ο οποίος αποτελείται και αυτός αντίστοιχα από δύο πεδία τα οποία είναι το Id και το Sysname. Στον Πίνακα αυτόν φαίνονται κάποια από τα χαρακτηριστικά που αφορούν τα events.

	Id	Sysname
1	1	TITLE
2	2	SUMMARY
3	3	CUSTOM_ADDRESS
4	4	WEBSITE_LINK
5	5	DESCRIPTION
6	6	EXTRA_INFO
7	7	CUSTOM_TIMESTAMP
8	8	NAME

Εικόνα 2.10 Δεδομένα του Πίνακα CaptionTypes_TB

Υπάρχει επίσης και ο Πίνακας PointViewCaptions_TB, ο οποίος αποτελείται από δύο πεδία. Το ένα είναι το PointViewId όπου δείχνει το Id του αντίστοιχου event και το δεύτερο είναι το CaptionId όπου δείχνει το Id του caption που αντιστοιχεί σε αυτό το event.

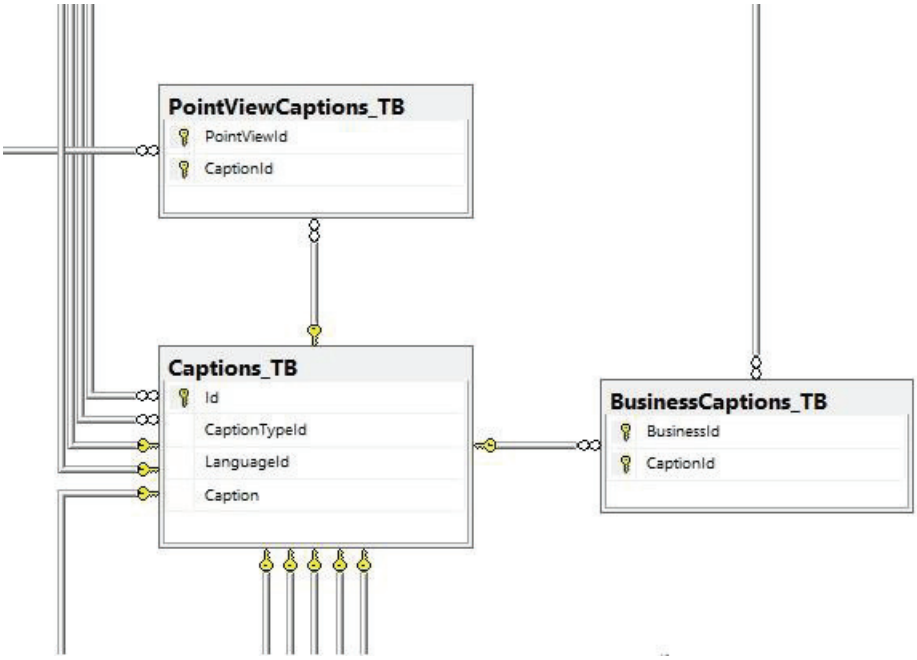
Results Messages

	PointViewId	CaptionId
1	2	206
2	2	207
3	2	208
4	2	209
5	6	220
6	7	340
7	8	751
8	8	752
9	8	753
10	8	754
11	36	545
12	37	1059
13	37	1060
14	37	1061

Εικόνα 2.11 Δεδομένα του Πίνακα PointViewCaptions_TB

Ακόμη μπορούμε να διακρίνουμε τους Πίνακες BusinessCaptionsLong_TB και CaptionsLong_TB. Ο CaptionsLong_TB ουσιαστικά συνδέει όλους τους προηγούμενους πίνακες που αναφέραμε αν και ακόμα οι δύο τελευταίοι δεν έχουν ενσωματωθεί σε κάποια λειτουργία του συστήματος. Έχουν σχεδιαστεί όμως για μελλοντική χρήση και κάλυψη της επεκτασιμότητας που αναφέραμε όταν περιγράφαμε τις απαιτήσεις και τις λειτουργίες του συστήματος.

- Στο κομμάτι 9 της εικόνας 2.3 υπάρχουν ομαδοποιημένες τρεις κλάσεις όπου επικοινωνούν μεταξύ τους και στην συνέχεια με το υπόλοιπο σύστημα.



Εικόνα 2.12 Μέρος της Βάσης Δεδομένων

Ο Πίνακας Captions_TB αποτελείται από τέσσερα πεδία όπου το ένα είναι το Id, το δεύτερο είναι το CaptionTypeId όπου αντιστοιχεί σε ένα από αυτά που περιλαμβάνει ο πίνακας που αναλύσαμε πριν (CaptionTypes_TB), το τρίτο είναι το LanguageId όπου αντιστοιχεί στην γλώσσα στην οποία είναι καταχωρημένη η πληροφορία και τέλος το Caption όπου είναι η περιγραφή.

Results Messages

	Id	CaptionTypeId	LanguageId	Caption
1	206	3	1	Tinos Island
2	207	1	1	Tinos Trail Race
3	208	4	1	novibet.com
4	209	2	1	Run at tinos island
5	220	1	1	The City Trail Race
6	340	1	1	Trail City 2
7	545	1	1	Tinos Trail
8	583	1	1	Test city trail race
9	584	6	1	Test
10	601	8	1	Sports
11	602	8	1	Racing
12	603	8	1	Road race
13	604	8	1	Trail race
14	605	8	1	City trail race

Εικόνα 2.13 Δεδομένα του Πίνακα Captions_TB

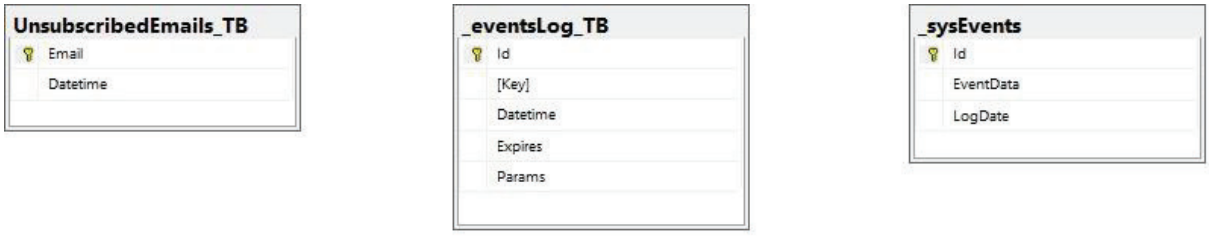
Στη συνέχει ο Πίνακας PointViewCaptions_TB αποτελείται από δύο πεδία τα οποία το κάθε ένα προσδιορίζει ένα Id, όπου αντιστοιχεί το event που έχει καταχωρηθεί με τις αντίστοιχες πληροφορίες που έχουμε καταχωρήσει.

Results Messages

	PointViewId	CaptionId
1	2	206
2	2	207
3	2	208
4	2	209
5	6	220
6	7	340
7	8	751
8	8	752
9	8	753
10	8	754
11	36	545
12	37	1059
13	37	1060
14	37	1061

Εικόνα 2.14 Δεδομένα του Πίνακα PointViewCaptions_TB

- Στο κομμάτι 1 της εικόνας 2.3 υπάρχουν ομαδοποιημένες τρεις κλάσεις όπου λειτουργούν αυτόνομα η κάθε μία και δεν έχουν καμία επικοινωνία με τις υπόλοιπες κλάσεις του συστήματός μας.



Εικόνα 2.15 Μέρος της Βάσης Δεδομένων

Ο Πίνακας _sysEvents πρόκειται για μία επικοινωνία του συστήματος ώστε να μπορεί να καταχωρεί κάποιες πληροφορίες και να αντίστοιχα να μπορεί να τις διαβάζει.

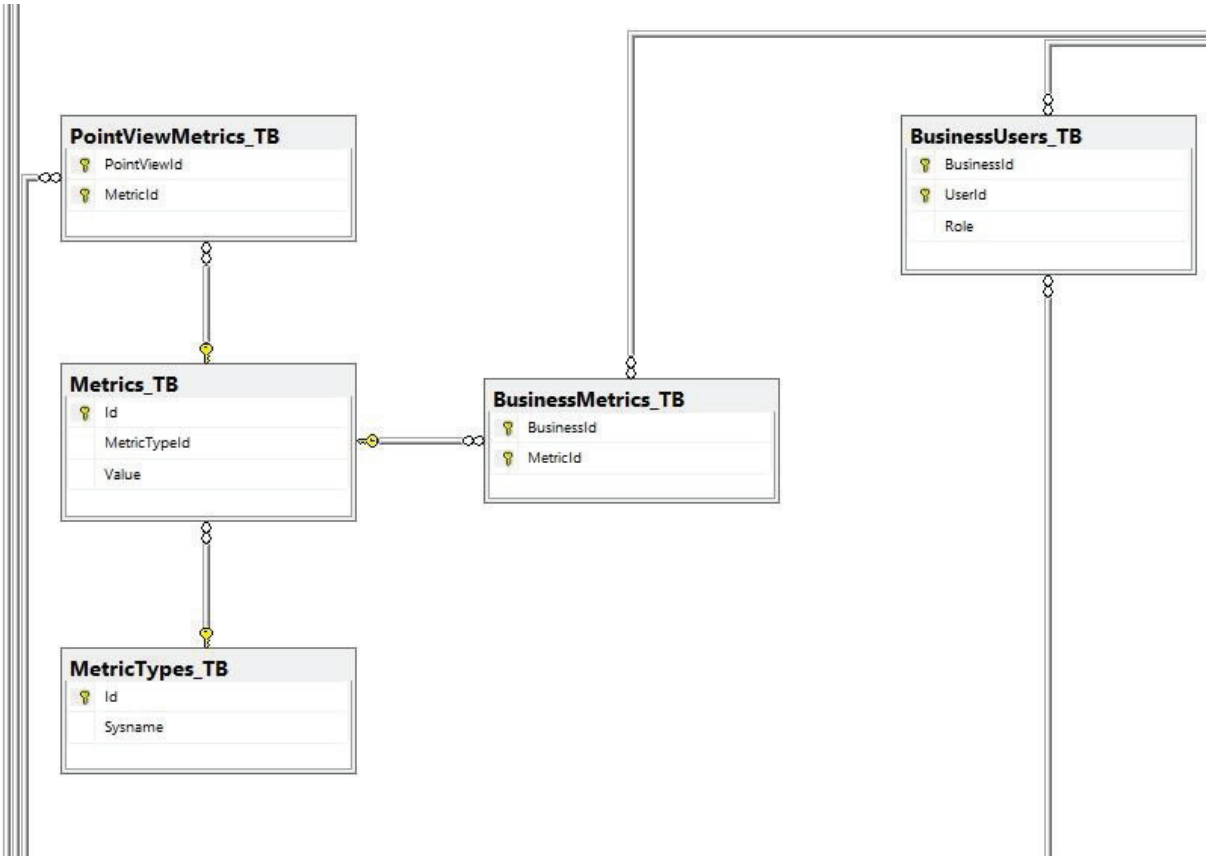
Results Messages

	Id	EventData	LogDate
1	2396	0x0001000000FFFFFFFF0100000000000000C0200000044...	2017-03-21 21:38:38.823
2	2397	0x0001000000FFFFFFFF0100000000000000C0200000044...	2017-03-21 21:40:26.107
3	2398	0x0001000000FFFFFFFF0100000000000000C0200000044...	2017-03-21 21:47:10.397
4	2399	0x0001000000FFFFFFFF0100000000000000C0200000044...	2017-04-13 20:20:11.510
5	2400	0x0001000000FFFFFFFF0100000000000000C0200000044...	2017-04-13 20:20:36.377
6	2401	0x0001000000FFFFFFFF0100000000000000C0200000044...	2017-04-21 13:05:13.923
7	2402	0x0001000000FFFFFFFF0100000000000000C0200000044...	2017-04-21 13:05:22.863
8	2403	0x0001000000FFFFFFFF0100000000000000C0200000044...	2017-04-21 13:05:33.490
9	2404	0x0001000000FFFFFFFF0100000000000000C0200000044...	2017-04-21 13:05:38.443

Εικόνα 2.16 Δεδομένα του Πίνακα _sysEvents

Τέλος ο Πίνακας UnsubscribedEmails_TB είναι αυτός όπου περιλαμβάνει τους χρήστες οι οποίοι επιθυμούν να μην λαμβάνουν αυτοματοποιημένα emails.

- Στο κομμάτι 3 της εικόνας 2.3 υπάρχουν ομαδοποιημένες πέντε κλάσεις όπου επικοινωνούν μεταξύ τους και στην συνέχεια με το υπόλοιπο σύστημα.



Εικόνα 2.17 Μέρος της Βάσης Δεδομένων

Δεν θα ασχοληθούμε ιδιαίτερα με τους Πίνακες BusinessMetrics_TB και BusinessUsers_TB καθώς δεν αντιστοιχούν σε κάποια από τις λειτουργίες που εκτελεί προς το παρόν το σύστημά μας αλλά αποτελούν μέρος του κομματιού της επεκτασιμότητας του συστήματος για μελλοντική χρήση.

Ο Πίνακας MetricTypes_TB αποτελείται από δύο πεδία όπως και αρκετοί από τους πίνακες όπου περιλαμβάνουν βασικά στοιχεία – τύπους για την λειτουργία του συστήματος. Το ένα από αυτά όπως σε όλες τις παρόμοιες περιπτώσεις είναι το Id και το δεύτερο είναι το Sysname.

Results Messages

	Id	Sysname
1	1	$SQMETERS
2	2	$DISTANCE
3	3	$ELEVATION_UP
4	4	$ELEVATION_DOWN

Εικόνα 2.18 Δεδομένα του Πίνακα MetricTypes_TB

Ο Πίνακας Metrics_TB αποτελείται από τρία πεδία και ουσιαστικά καταγράφει συγκεκριμένες πληροφορίες που αφορούν κάθε event, όπως π.χ. την απόσταση της διαδρομής, σε τι υψόμετρο βρίσκεται κτλ, όπου αυτές οι πληροφορίες καταγράφονται στο τρίτο πεδίο (Value).

Results Messages

	Id	MetricTypeId	Value
1	54	2	5.50
2	55	3	100.00
3	56	4	50.00
4	231	2	10.00
5	232	3	100.00
6	233	4	150.60
7	270	2	25.00
8	280	2	20.00
9	281	3	600.00
10	282	4	200.00
11	285	2	20.00
12	286	3	1000.00
13	287	4	1000.00
14	291	2	15.00

Εικόνα 2.19 Δεδομένα του Πίνακα Metrics_TB

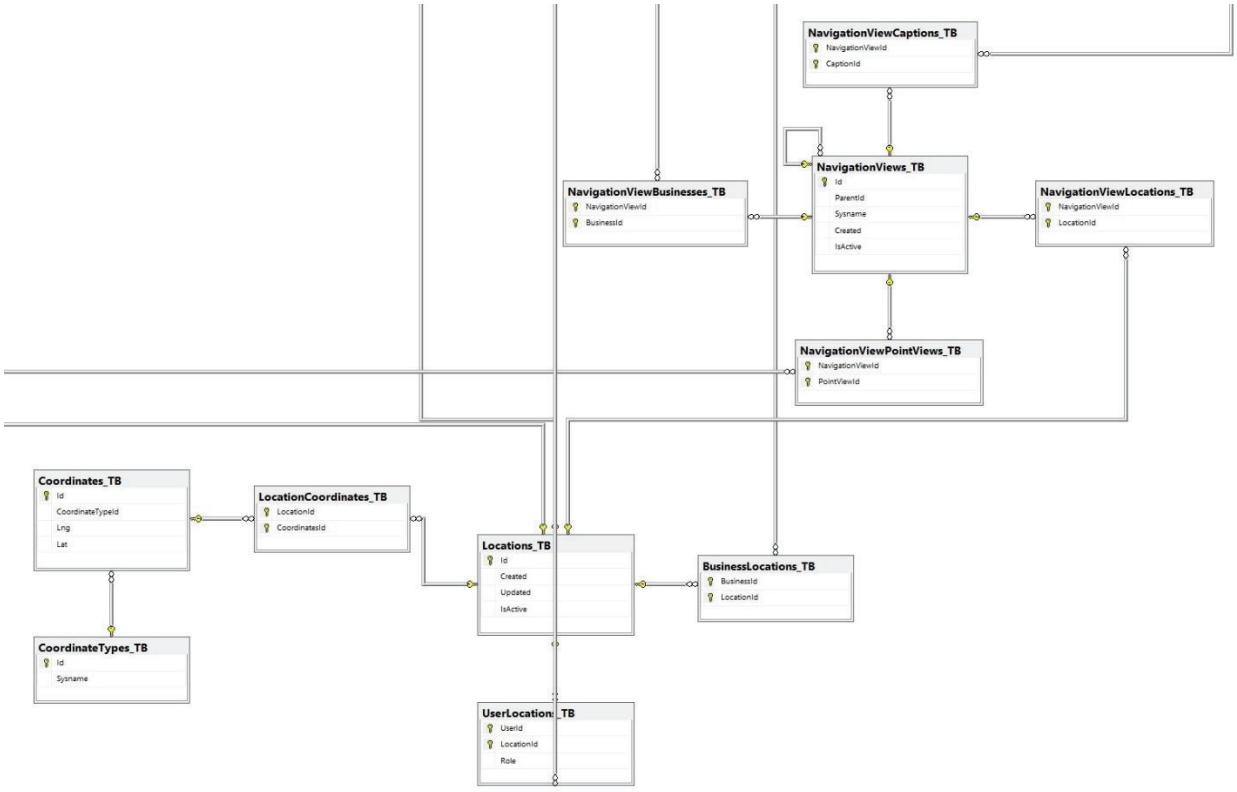
Τέλος ο Πίνακας PointViewMetrics_TB έχει αναλάβει την σύνδεση των events με τον πίνακα όπου περιγράψαμε πριν (Metrics_TB) που περιέχει τις πληροφορίες σχετικά με τον αγώνα.

Results Messages

	PointViewId	MetricId
1	2	54
2	2	55
3	2	56
4	8	231
5	8	232
6	8	233
7	37	291
8	37	292
9	37	293
10	39	280
11	39	281
12	39	282
13	41	270
14	43	461

Εικόνα 2.20 Δεδομένα του Πίνακα PointViewMetrics_TB

- Στο κομμάτι 10 της εικόνας 2.3 υπάρχουν ομαδοποιημένες έντεκα κλάσεις όπου επικοινωνούν μεταξύ τους και στην συνέχεια με το υπόλοιπο σύστημα.



Εικόνα 2.21 Μέρος της Βάσης Δεδομένων

Στον Πίνακα CoordinateTypes_TB μπορούμε να δούμε τους τύπους όπου χαρακτηρίζουν τις συντεταγμένες. Υπάρχουν δύο πεδία όπου το ένα είναι το Id και το δεύτερο το όνομα του τύπου. Ως Primary χαρακτηρίζεται το κύριο σημείο αναφοράς του event στον χάρτη και ως Border χαρακτηρίζονται τα σημεία τα οποία πλαισιώνουν το Primary σημείο και δημιουργούν κάτι ως 'φράκτη' γύρω από αυτό.

Results Messages

	Id	Sysname
1	1	PRIMARY
2	2	BORDER

Εικόνα 2.22 Δεδομένα του Πίνακα CoordinateTypes_TB

Στην συνέχεια στον Πίνακα Coordinates_TB καταγράφονται πληροφορίες όπως π.χ. οι συντεταγμένες του κάθε σημείου - event, όπως μπορούμε να δούμε και από τα πεδία από τα οποία αποτελείται. Πρώτο είναι το Id, δεύτερο είναι ο τύπος ο οποίος μπορεί να πάρει κάποιες από τις τιμές του Πίνακα CoordinateTypes_TB και στην συνέχεια είναι τα πεδία Lng και Lat όπου αντιπροσωπεύουν το 'γεωγραφικό πλάτος' και το 'γεωγραφικό μήκος' αντίστοιχα.

Results Messages

	Id	CoordinateTypeId	Lng	Lat
1	174	1	25.000000000000000000	37.000000000000000000
2	175	1	25.000000000000000000	37.000000000000000000
3	176	1	25.000000000000000000	37.000000000000000000
4	177	1	23.000000000000000000	38.000000000000000000
5	179	1	21.000000000000000000	34.000000000000000000
6	190	1	23.000000000000000000	38.000000000000000000
7	214	1	23.000000000000000000	38.000000000000000000
8	219	1	23.000000000000000000	38.000000000000000000
9	220	1	23.000000000000000000	38.000000000000000000
10	221	1	23.000000000000000000	38.000000000000000000
11	224	1	23.000000000000000000	38.000000000000000000
12	225	1	23.000000000000000000	38.000000000000000000
13	226	1	23.000000000000000000	38.000000000000000000
14	227	1	23.000000000000000000	38.000000000000000000

Εικόνα 2.23 Δεδομένα του Πίνακα Coordinates_TB

Στον Πίνακα Locations_TB καταχωρούνται τα σημεία που ορίζει ο διαχειριστής για κάθε event. Και καταγράφονται οι κύριες πληροφορίες όπως η ημ/νία που καταχωρήθηκε, πότε έγινε το τελευταίο update, αν είναι active. Και έτσι προκύπτουν και τα πεδία που τον χαρακτηρίζουν τα οποία είναι το Id, Created, Updated και IsActive.

Results Messages

	Id	Created	Updated	IsActive
1	128	2016-12-25 13:21:19.920	2016-12-25 13:21:19.920	1
2	129	2016-12-27 19:22:47.950	2016-12-27 19:22:47.950	1
3	130	2016-12-27 19:23:20.347	2016-12-27 19:23:20.347	1
4	131	2016-12-27 19:24:32.733	2016-12-27 19:24:32.733	1
5	132	2016-12-27 19:29:21.007	2016-12-27 19:29:21.007	1
6	133	2016-12-30 18:53:59.713	2016-12-30 18:53:59.713	1
7	134	2016-12-30 19:00:41.683	2016-12-30 19:00:41.683	1
8	135	2016-12-30 20:01:33.913	2016-12-30 20:01:33.913	1
9	136	2017-01-01 10:43:33.350	2017-01-01 10:43:33.350	1
10	137	2017-01-01 18:45:18.297	2017-01-01 18:45:18.297	1
11	138	2017-01-01 19:46:03.820	2017-01-01 19:46:03.820	1
12	139	2017-01-01 19:46:32.557	2017-01-01 19:46:32.557	1
13	140	2017-01-01 19:48:36.947	2017-01-01 19:48:36.947	1
14	141	2017-01-01 19:51:36.087	2017-01-01 19:51:36.087	1

Εικόνα 2.24 Δεδομένα του Πίνακα Locations_TB

Η χρήση του Πίνακα LocationCoordinates_TB είναι η σύνδεση του Πίνακα Locations_TB με τον Πίνακα Coordinates_TB ώστε να μπορούν να αντιστοιχηθούν οι καταχωρήσεις των events με τα αντίστοιχα σημεία όπου έχουν δηλωθεί στον χάρτη. Αποτελείται από δύο πεδία τα οποία είναι LocationId και CoordinatesId αντίστοιχα.

Results Messages

	LocationId	CoordinatesId
1	128	190
2	129	174
3	130	175
4	131	176
5	132	177
6	133	179
7	134	214
8	135	356
9	136	242
10	137	244
11	138	219
12	139	220
13	140	221
14	141	224

Εικόνα 2.25 Δεδομένα του Πίνακα LocationCoordinates_TB

Ο Πίνακας UserLocations_TB κρατάει πληροφορίες σχετικά με τον χρήστη ο οποίος καταχωρεί το συγκεκριμένο event. Ο Πίνακας αυτός αποτελείται από τρία πεδία όπου το πρώτο είναι το UserId, το δεύτερο είναι το LocationId και το τρίτο είναι το Role (τα δεδομένα αυτού του πίνακα θα αναλυθούν στην πορεία, αλλά γενικά περιγράφει τον ρόλο που έχει ο εκάστοτε χρήστης μέσα στο σύστημά μας). Στην ακόλουθη εικόνα θα βλέπουμε να επαναλαμβάνεται συνεχώς το ίδιο UserId και το ίδιο Role type καθώς το σύστημα διαθέτει έναν διαχειριστή ο οποίος είναι υπεύθυνος για την καταχώρηση των events.

Results Messages

	UserId	LocationId	Role
1	E451FF3B-FA3C-468A-95FF-66026576EEAB	128	1000
2	E451FF3B-FA3C-468A-95FF-66026576EEAB	129	1000
3	E451FF3B-FA3C-468A-95FF-66026576EEAB	130	1000
4	E451FF3B-FA3C-468A-95FF-66026576EEAB	131	1000
5	E451FF3B-FA3C-468A-95FF-66026576EEAB	132	1000
6	E451FF3B-FA3C-468A-95FF-66026576EEAB	133	1000
7	E451FF3B-FA3C-468A-95FF-66026576EEAB	134	1000
8	E451FF3B-FA3C-468A-95FF-66026576EEAB	135	1000
9	E451FF3B-FA3C-468A-95FF-66026576EEAB	136	1000
10	E451FF3B-FA3C-468A-95FF-66026576EEAB	137	1000
11	E451FF3B-FA3C-468A-95FF-66026576EEAB	138	1000
12	E451FF3B-FA3C-468A-95FF-66026576EEAB	139	1000
13	E451FF3B-FA3C-468A-95FF-66026576EEAB	140	1000
14	E451FF3B-FA3C-468A-95FF-66026576EEAB	141	1000

Εικόνα 2.26 Δεδομένα του Πίνακα UserLocations_TB

Το δεύτερο κομμάτι της εικόνας 2.21 (το πάνω δεξιά) όπου περιλαμβάνει τις υπόλοιπες πέντε κλάσεις περιέχει τις κλάσεις οι οποίες είναι υπεύθυνες για την αναζήτηση και την εμφάνιση των events στον χάρτη, καθώς συλλέγει τις πληροφορίες από όλους τους προηγούμενους πίνακες που αναφέραμε και ανάλογα με την απόσταση που υπάρχει μεταξύ των σημείων που έχουν δηλωθεί ομαδοποιεί τα event ανά περιοχή και είναι υπεύθυνο για την απεικόνισή τους.

Ο Πίνακας NavigationViewPointViews_TB αποτελείται από δύο πεδία, τα οποία είναι, το NavigationViewId και το PointViewId.

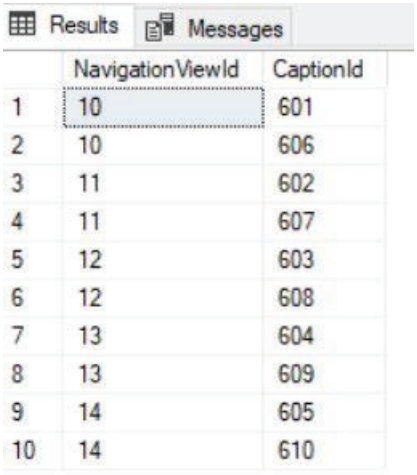
Results Messages

	NavigationViewId	PointViewId
1	1	229
2	1	230
3	1	231
4	1	232
5	1	233
6	1	234
7	1	235
8	1	236
9	1	237
10	1	238
11	1	239
12	1	240
13	1	241
14	8	2
15	8	6
16	8	7
17	8	36
18	8	40

Εικόνα 2.27 Δεδομένα του Πίνακα NavigationViewPointViews_TB

Ο Πίνακας NavigationViewLocations_TB αφορά το μενού που βρίσκεται στην πάνω αριστερή μεριά της εφαρμογής των χρηστών. Ουσιαστικά πρόκειται για το κύριο μενού της εφαρμογής. Το NvigationView με λίγα λόγια κάνει την σύνδεση των PointView (όπου είναι τα καταχωρημένα event) με την απεικόνισή τους στην οθόνη. Ανάλογα λοιπόν με το ποια επιλογή από το μενού έχεις επιλέξει κάνει την σύνδεση με το αντίστοιχο Location και στην συνέχει με όλα τα ViewPoint όπου αντιστοιχούν σε αυτό το Location, με κύριο στόχο να τα εμφανίσει στην οθόνη του χρήστη.

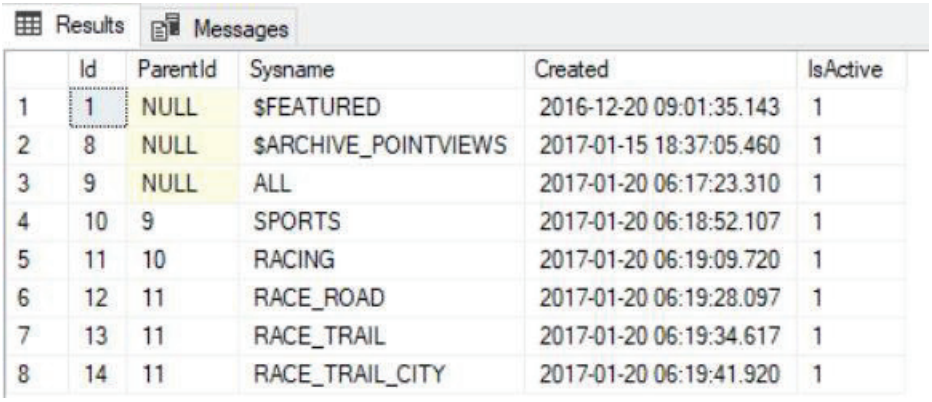
Στη συνέχεια είναι ο Πίνακας NavigationViewCaptions_TB ο οποίος αποτελείται επίσης από δύο πεδία τα οποία είναι τα NavigationViewId και το CaptoinId (το CaptionId αντιστοιχεί στον Πίνακα όπου είχαμε αναλύσει νωρίτερα, και αντιστοιχεί στην εικόνα 2.13 “Δεδομένα του Πίνακα Captions_TB”).


Results Messages

	NavigationViewId	CaptionId
1	10	601
2	10	606
3	11	602
4	11	607
5	12	603
6	12	608
7	13	604
8	13	609
9	14	605
10	14	610

Εικόνα 2.28 Δεδομένα του Πίνακα NavigationViewPointViews_TB

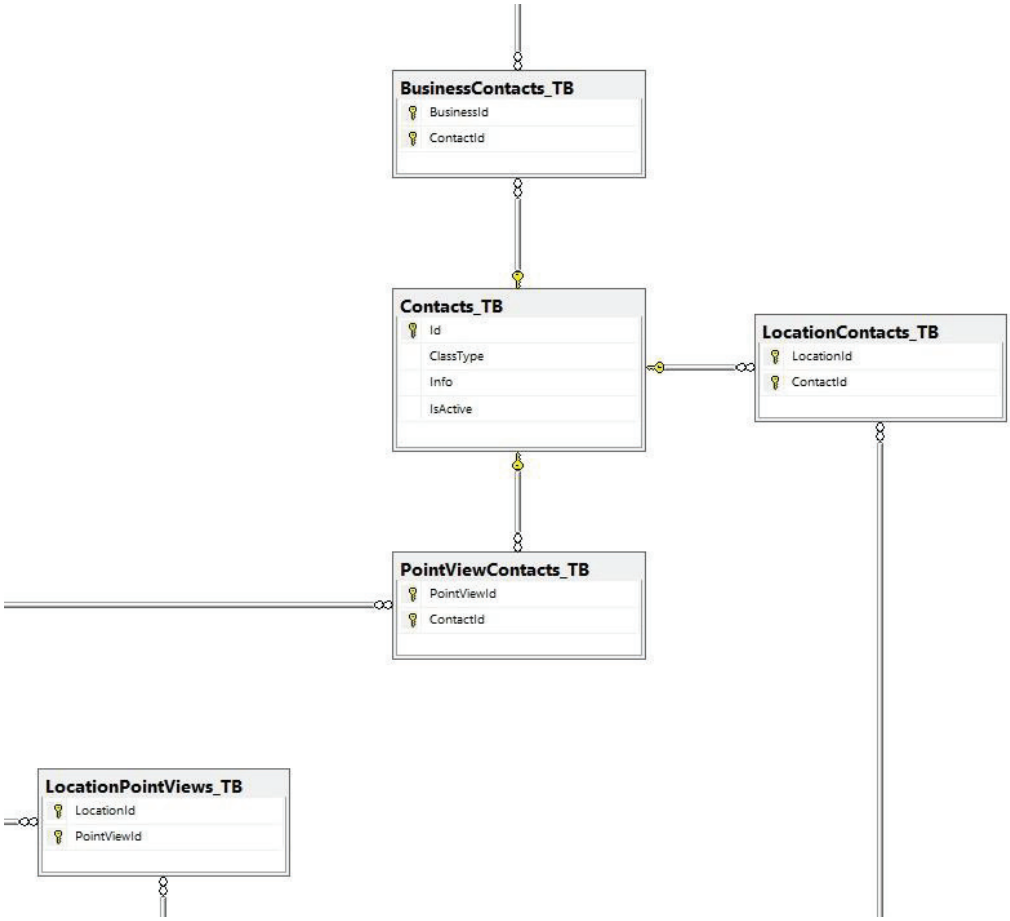
Τέλος έχουμε τον Πίνακα NavigationViews_TB ο οποίος συλλέγει τις βασικές πληροφορίες και ομαδοποιεί έτσι τα δεδομένα ώστε να μπορεί να γίνει σωστά στην συνέχεια η ομαδοποίηση των events κάτω από το κύριο μενού στην εφαρμογή των χρηστών. Αποτελείται από πέντε πεδία όπου είναι το Id, το ParentId, Created, IsActive και το Sysname το οποίο προσδιορίζει τον τύπο – ονομασία του κάθε επιπέδου στο μενού.


Results Messages

	Id	ParentId	Sysname	Created	IsActive
1	1	NULL	$FEATURED	2016-12-20 09:01:35.143	1
2	8	NULL	$ARCHIVE_POINTVIEWS	2017-01-15 18:37:05.460	1
3	9	NULL	ALL	2017-01-20 06:17:23.310	1
4	10	9	SPORTS	2017-01-20 06:18:52.107	1
5	11	10	RACING	2017-01-20 06:19:09.720	1
6	12	11	RACE_ROAD	2017-01-20 06:19:28.097	1
7	13	11	RACE_TRAIL	2017-01-20 06:19:34.617	1
8	14	11	RACE_TRAIL_CITY	2017-01-20 06:19:41.920	1

Εικόνα 2.29 Δεδομένα του Πίνακα NavigationViews_TB

- Στο κομμάτι 8 της εικόνας 2.3 υπάρχουν ομαδοποιημένες πέντε κλάσεις όπου επικοινωνούν μεταξύ τους και στην συνέχεια με το υπόλοιπο σύστημα.



Εικόνα 2.30 Μέρος της Βάσης Δεδομένων

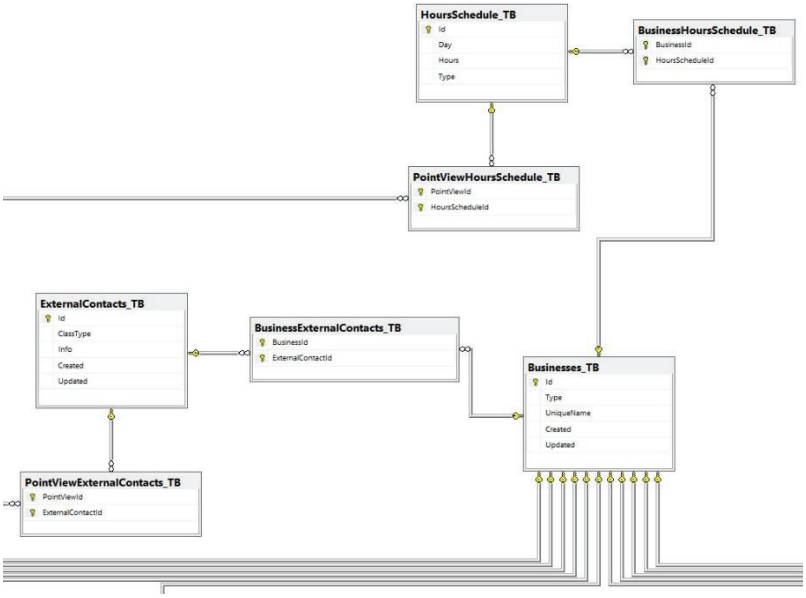
Ο Πίνακας LocationPointViews_TB μεσολαβεί ώστε να αντιστοιχίσει τα δεδομένα του πίνακα PointViews_TB με τα δεδομένα του πίνακα Location_TB.

Results Messages

	LocationId	PointViewId
1	129	2
2	133	6
3	134	7
4	135	8
5	163	36
6	164	37
7	167	39
8	168	40
9	169	41
10	172	43
11	175	46
12	176	47
13	177	48
14	181	51

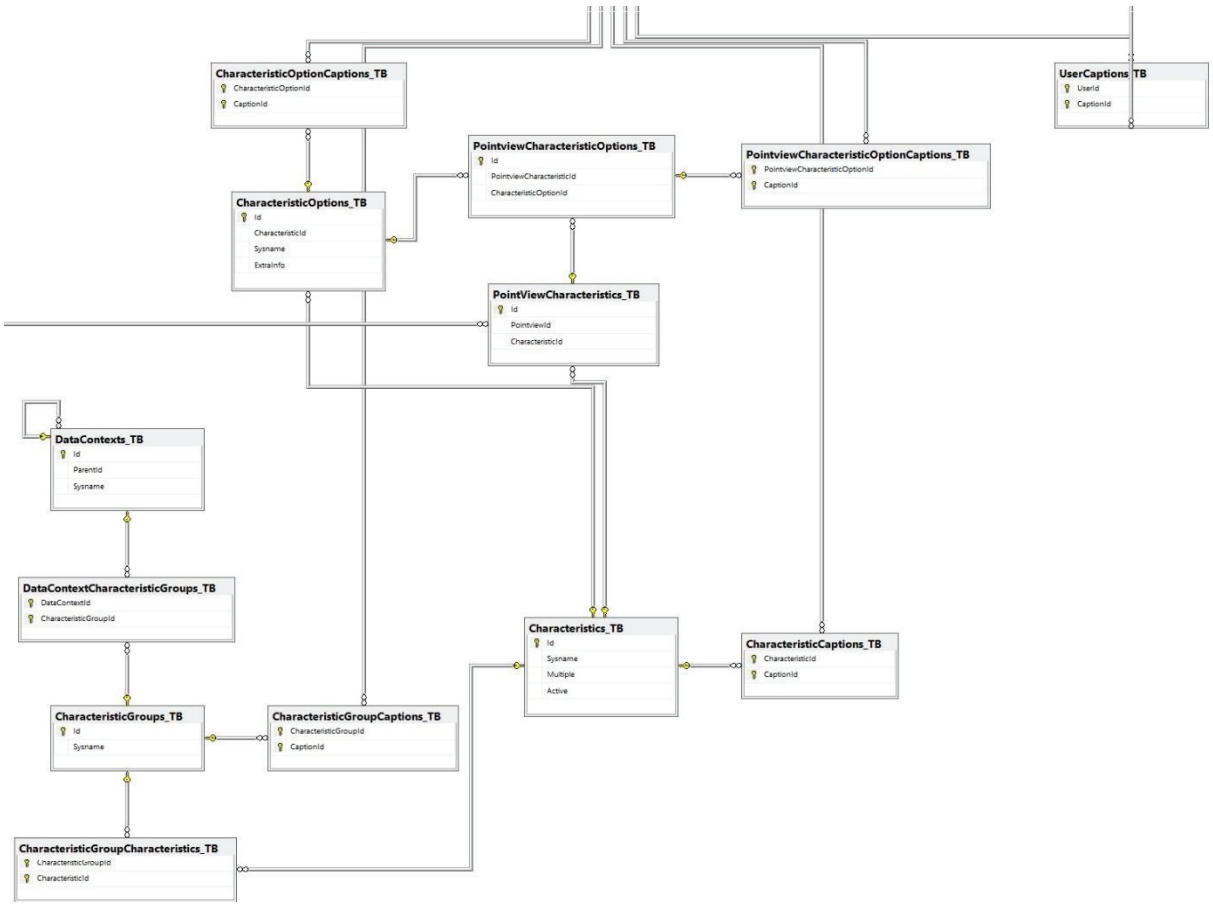
Εικόνα 2.31 Δεδομένα του Πίνακα LocationPointViews_TB

- Στο κομμάτι 9 της εικόνας 2.3 υπάρχουν ομαδοποιημένες εφτά κλάσεις όπου επικοινωνούν μεταξύ τους και στην συνέχεια με το υπόλοιπο σύστημα.



Εικόνα 2.32 Μέρος της Βάσης Δεδομένων

- Στο κομμάτι 11 της εικόνας 2.3 υπάρχουν ομαδοποιημένες 13 κλάσεις όπου επικοινωνούν μεταξύ τους και στην συνέχεια με το υπόλοιπο σύστημα.



Εικόνα 2.33 Μέρος της Βάσης Δεδομένων

Το κομμάτι αυτό αφορά κυρίως τα χαρακτηριστικά των events όπου καλείται ο διαχειριστής να καταχωρήσει. Είναι επιπλέον πληροφορίες όπως αν υπάρχει κάποιο μετάλλιο για τον νικητή, αν υπάρχει κάποιο T-shirt αναμνηστικό κ.α.

Για αρχή στον Πίνακα Characteristics_TB, ο οποίος αποτελείται από 4 πεδία, υπάρχουν όλα τα πεδία τα οποία καλείται ο διαχειριστής να συμπληρώσει (όπως μπορούμε να διακρίνουμε και στην ακόλουθη εικόνα). Αποτελείται από ένα Id, το Sysname όπου αντιπροσωπεύει το κάθε χαρακτηριστικό που προστίθεται, από το πεδίο Active το οποίο δηλώνει αν θα εμφανίζεται το αντίστοιχο πεδίο προς συμπλήρωση στο interface του διαχειριστή και τέλος το πεδίο Multiple το οποίο προσδιορίζει αν μπορεί να επιλέξει περισσότερες από μία επιλογές.

Results Messages

	Id	Sysname	Multiple	Active
1	1	TIMING	1	1
2	2	PHOTOSHOOTING	0	1
3	3	TSHIRT	0	1
4	4	MEDAL	0	1
5	5	CATEGORIES	0	1

Εικόνα 2.34 Δεδομένα του Πίνακα Characteristics_TB

Στη συνέχεια ο ρόλος του Πίνακα CharacteristicsCaptions_TB είναι να συνδέσει τις κατηγορίες που είδαμε στον προηγούμενο πίνακα με τα αντίστοιχα caption ώστε να εμφανίζονται σωστά τα πεδία στο σύστημα.

Results Messages

	CharacteristicId	CaptionId
1	1	652
2	1	653
3	2	654
4	2	655
5	3	656
6	3	657
7	4	658
8	4	659
9	5	660
10	5	661

Εικόνα 2.35 Δεδομένα του Πίνακα CharacteristicsCaptions _TB

Για να μπορέσουμε να κατανοήσουμε καλύτερα την ένωση αυτών των δύο πινάκων η παρακάτω εικόνα δείχνει τα δεδομένα τα οποία περιέχει ο πίνακας Captions_TB (τον οποίο είχαμε αναλύσει πιο πάνω) με το κομμάτι που αντιστοιχεί στα CaptionId που φαίνονται στην εικόνα 2.35 .

Results Messages

	Id	CaptionTypeId	LanguageId	Caption
52	654	8	1	Photoshooting
53	655	8	2	Φωτογράφιση
54	656	8	1	T-shirt
55	657	8	2	Μπλουζάκι
56	658	8	1	Medal
57	659	8	2	Μετάλλιο
58	660	8	1	Categories
59	661	8	2	Κατηγορίες
60	662	8	1	Event facilities
61	663	8	2	Παροχές αγώνα
62	751	1	1	Mobile trail city
63	752	3	1	Jdd
64	753	4	1	Xxbj
65	754	2	1	Bdnd

Εικόνα 2.36 Δεδομένα του Πίνακα Captions _TB

Ο Πίνακας CharacteristicGroups_TB βοηθάει στην ομαδοποίηση των χαρακτηριστικών που αφορούν τα events για την καλύτερη λειτουργία του συστήματος.

Results Messages

	Id	Sysname
1	1	EVENT_FACILITIES

Εικόνα 2.37 Δεδομένα του Πίνακα Captions _TB

Στην συνέχεια ο Πίνακας CharacteristicGroupCharacteristics_TB ενώνει τους δύο προηγούμενος πίνακες ώστε να αντιστοιχήσει το κάθε “χαρακτηριστικό” που έχουμε προσθέσει σε ένα από τα Group που υπάρχουν. Στην συγκεκριμένη περίπτωση υπάρχει μόνο ένα group και έτσι όλα ανήκουν στο ίδιο.

Results Messages

	CharacteristicGroupId	CharacteristicId
1	1	1
2	1	2
3	1	3
4	1	4
5	1	5

Εικόνα 2.38 Δεδομένα του Πίνακα CharacteristicGroupCharacteristics _TB

Ο Πίνακας DataContexts_TB περιέχει το μενού όπου είναι διαθέσιμο. Ουσιαστικά υπάρχει μία ιεραρχία όπου προσδιορίζεται από το πεδίο ParentId όπου υπάρχει στον πίνακα και προσδιορίζει το level κάθε καταχώρησης στο μενού. Για καλύτερη κατανόηση μπορούμε να δούμε την εικόνα που ακολουθεί.

Results Messages

	Id	ParentId	Sysname
1	1	NULL	ROOT
2	2	1	SPORTS
3	3	2	RACING
4	4	3	RACE_ROAD
5	5	3	RACE_TRAIL
6	6	3	RACE_TRAIL_CITY

Εικόνα 2.39 Δεδομένα του Πίνακα DataContexts _TB

Και τέλος, για το πρώτο κομμάτι που αφορά το μέρος 11 της εικόνας 2.3 υπάρχει και ο πίνακας DataCharacteristicGroups_TB ο οποίος είναι υπεύθυνος για την ένωση των προηγούμενων δύο.

Results Messages

	DataContextId	CharacteristicGroupId
1	4	1
2	5	1
3	6	1

Εικόνα 2.40 Δεδομένα του Πίνακα DataCharacteristicGroups _TB

Από το δεύτερο κομμάτι από το μέρος 11 της εικόνας 2.3 θα ξεκινήσουμε με τον πίνακα CharacteristicOptions_TB ο οποίος περιέχει για κάθε ένα από τα χαρακτηριστικά που έχουμε προσθέσει τις τρεις διαθέσιμες επιλογές που έχει ο διαχειριστής όπως αυτές φαίνονται στην ακόλουθη εικόνα.

Results Messages

	Id	CharacteristicId	Sysname	ExtraInfo
1	1	1	YES	1
2	2	1	NO	0
3	3	1	UNKNOWN	0
4	4	2	YES	0
5	5	2	NO	0
6	6	2	UNKNOWN	0
7	7	3	YES	0
8	8	3	NO	0
9	9	3	UNKNOWN	0
10	10	4	YES	0
11	11	4	NO	0
12	12	4	UNKNOWN	0
13	13	5	YES	0
14	14	5	NO	0

Εικόνα 2.41 Δεδομένα του Πίνακα CharacteristicOptions _TB

Έπειτα ακολουθεί ο πίνακας CharacteristicOptionCaptions_TB ο οποίος αντιστοιχεί στα δεδομένα του προηγούμενου πίνακα τα αντίστοιχα captions (ουσιαστικά τους αντιστοιχεί τα λεκτικά με τα οποία εμφανίζονται στο σύστημα).

Results Messages

	CharacteristicOptionId	CaptionId
1	1	616
2	1	617
3	2	618
4	2	619
5	3	620
6	3	621
7	4	628
8	4	629
9	5	630
10	5	631
11	6	632

Εικόνα 2.42 Δεδομένα του Πίνακα CharacteristicOptionCaptions _TB

Ο Πίνακας PointViewCharacteristics_TB αντιστοιχεί το κάθε event που έχουμε καταχωρήσει με κάθε ένα από τα χαρακτηριστικά όπου έχουμε καταχωρήσει.

Results Messages

	Id	PointviewId	CharacteristicId
1	101	41	1
2	102	41	2
3	103	41	3
4	104	41	4
5	105	41	5
6	116	39	1
7	117	39	2
8	118	39	3
9	119	39	4
10	120	39	5
11	131	46	1
12	132	46	2
13	133	46	3
14	134	46	4

Εικόνα 2.43 Δεδομένα του Πίνακα PointViewCharacteristics _TB

Στη συνέχεια υπάρχει ο Πίνακας PointViewCharacteristicOptions_TB ο οποίος συνδέει τα δεδομένα των δύο προηγούμενων πινάκων όπως μπορούμε να διακρίνουμε και στην ακόλουθη εικόνα.

Results Messages

	Id	PointviewCharacteristicId	CharacteristicOptionId
1	108	101	1
2	109	102	5
3	110	103	7
4	111	104	10
5	112	105	14
6	123	116	3
7	124	117	6
8	125	118	9
9	126	119	12
10	127	120	19
11	140	131	1
12	141	132	4
13	142	133	7
14	143	134	10

Εικόνα 2.44 Δεδομένα του Πίνακα PointViewCharacteristicOptions _TB

Προτελευταίος πίνακας αυτής της ενότητας είναι ο PointViewCharacteristicOptionCaptions_TB ο οποίος όπως έχουμε δει και για όλους τους προηγούμενους πίνακες που έχουν την ίδια κατάληξη (δηλαδή περιλαμβάνουν την λέξη Caption) αντιστοιχεί στα δεδομένα του προηγούμενου πίνακα τα αντίστοιχα λεκτικά για την σωστή εμφάνισή τους στο σύστημα.

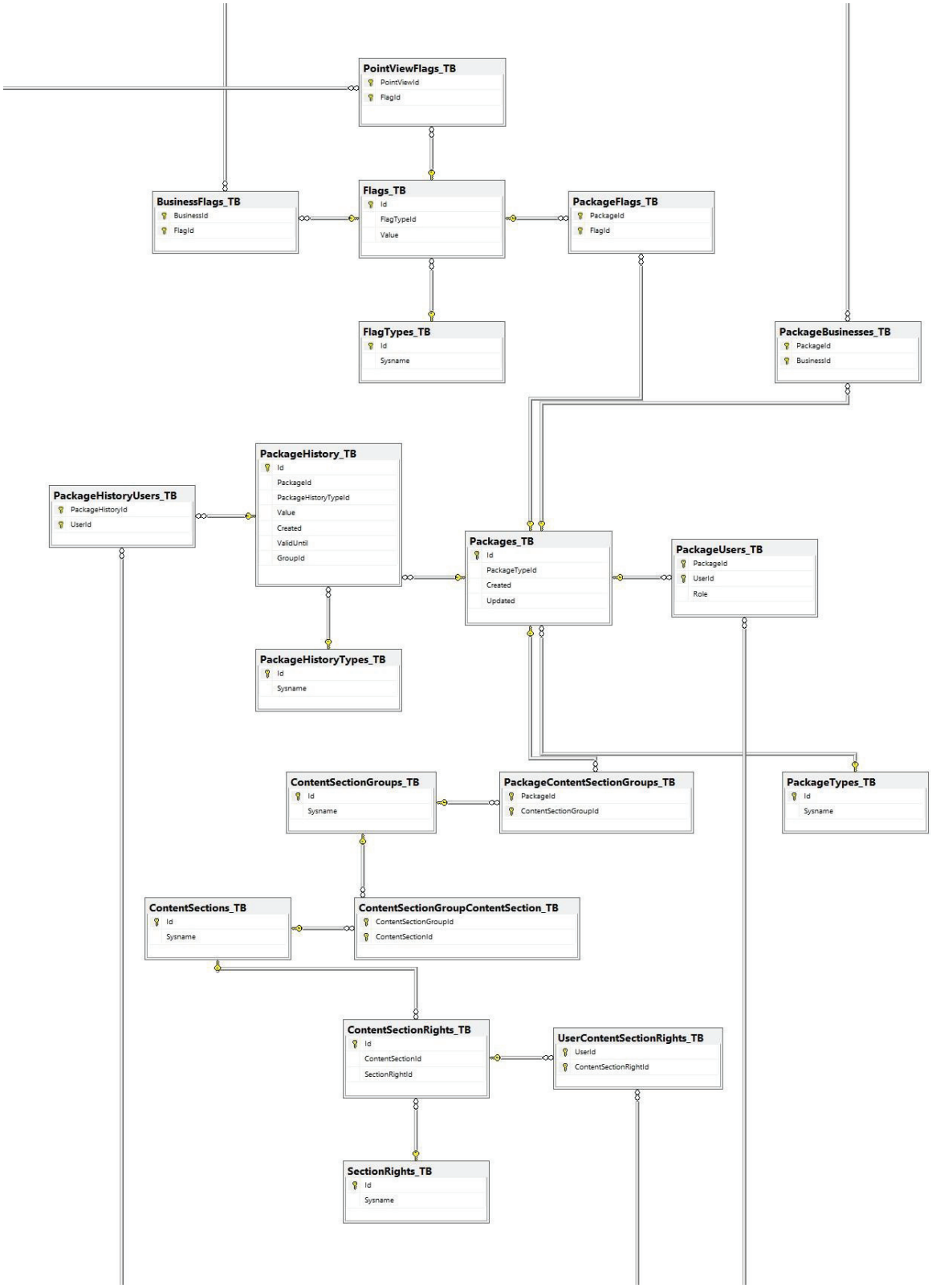
Results Messages

	PointviewCharacteristicOptionId	CaptionId
1	108	998
2	108	999
3	196	1618
4	196	1619

Εικόνα 2.45 Δεδομένα του Πίνακα PointViewCharacteristicOptionCaptions _TB

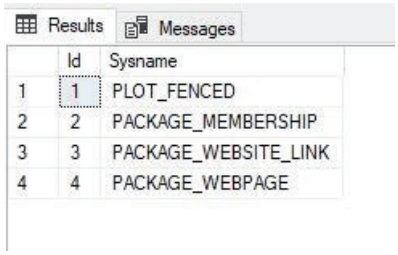
Τέλος υπάρχει και ο πίνακας UserCaptions_TB όπου αντιστοιχεί στον κάθε χρήστη τα αντίστοιχα λεκτικά. Στην συγκεκριμένη περίπτωση όμως αυτός ο πίνακας δεν εξυπηρετεί κάποια λειτουργία του συστήματος.

- Στη συνέχεια θα δούμε το προ-τελευταίο τμήμα από το οποίο αποτελείται η βάση δεδομένων. Στο κομμάτι 4 της εικόνας 2.3 υπάρχουν ομαδοποιημένες 19 κλάσεις όπου επικοινωνούν μεταξύ τους και στην συνέχεια με το υπόλοιπο σύστημα.



Εικόνα 2.46 Μέρος της Βάσης Δεδομένων

Αρχικά ας αναλύσουμε το πρώτο κομμάτι του τμήματος 4 το οποίο περιέχει 5 Πίνακες. Πρώτα ας δούμε τον Πίνακα FlagTypes_TB ο οποίος με βάση και την δομή που ακολουθούμε περιέχει τους τύπους οι οποίοι μπορούν να αντιστοιχηθούν σε αυτά τα δεδομένα.

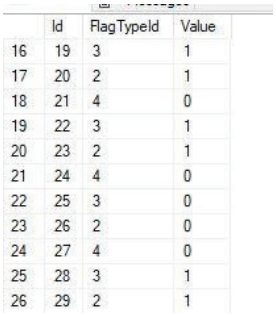

Results Messages

	Id	Sysname
1	1	PLOT_FENCED
2	2	PACKAGE_MEMBERSHIP
3	3	PACKAGE_WEBSITE_LINK
4	4	PACKAGE_WEBPAGE

Εικόνα 2.47 Δεδομένα του Πίνακα FlagTypes _TB

Ο συγκεκριμένος πίνακας είναι πιο γενικής χρήσης καθώς μπορεί να αντιστοιχηθεί με πολλούς από τους πίνακες του συστήματος. Ας δώσουμε μία περιγραφή για τους τέσσερις τύπους που βλέπουμε στην παραπάνω εικόνα. Ο πρώτος τύπος ουσιαστικά αναφέρεται στο αν κάποιο σημείο έχει σημεία τα οποία αναφέρονται στην περίφραξη του κύριου σημείου. Ο δεύτερος τύπος αναφέρεται στο αν μπορείς να προσθέσεις και άλλους χρήστες, ο τρίτος στο αν μπορείς να προσθέσεις κάποιο Link όπου σε οδηγεί σε εξωτερικό site, και τέλος ο τέταρτος τύπος αναφέρεται στο αν μπορείς να προωθείς το δικό σου site μέσα από το δικό μας σύστημα. Όπως θα δούμε και στον επόμενο πίνακα σε αυτούς τους τύπους αντιστοιχούνται τιμές 0 και 1 αντίστοιχα με το αν είναι active ή όχι.

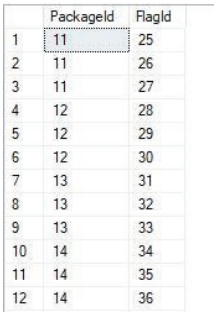
Ο Πίνακας Flags_TB είναι ο πίνακας ο οποίος ενώνει όλους τους πίνακες του πρώτου κομματιού που αναλύουμε ο οποίος αποτελείται από τρία πεδία, ένα χαρακτηριστικό ID, το FlagTypeId και το Value. Τα δύο πρώτα είναι προφανές σε τι αναφέρονται ωστόσο το τρίτο πεδίο μπορεί να πάρει μόνο τιμές 0 και 1 οι οποίες με τη σειρά τους αναφέρονται στο αν είναι διαθέσιμη αυτή η επιλογή ή όχι.



	Id	FlagTypeId	Value
16	19	3	1
17	20	2	1
18	21	4	0
19	22	3	1
20	23	2	1
21	24	4	0
22	25	3	0
23	26	2	0
24	27	4	0
25	28	3	1
26	29	2	1

Εικόνα 2.48 Δεδομένα του Πίνακα Flags _TB

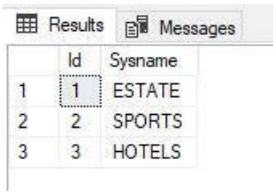
Τέλος στο πρώτο μέρος βλέπουμε ακόμα τον Πίνακα PackageFlags_TB ο οποίος αντιστοιχεί τα δεδομένα του πίνακα που είδαμε προηγουμένως με τον πίνακα Packages_TB όπου θα δούμε στην συνέχεια.



	PackageId	FlagId
1	11	25
2	11	26
3	11	27
4	12	28
5	12	29
6	12	30
7	13	31
8	13	32
9	13	33
10	14	34
11	14	35
12	14	36

Εικόνα 2.49 Δεδομένα του Πίνακα PackageFlags _TB

Το δεύτερο κομμάτι του μέρους 4 από τη βάση δεδομένων αναφέρεται στα Packages. Ας ξεκινήσουμε λοιπόν με τον πίνακα όπου συναντάμε σε όλους τους τομείς, ο οποίος είναι ο Πίνακας PackageTypes_TB.

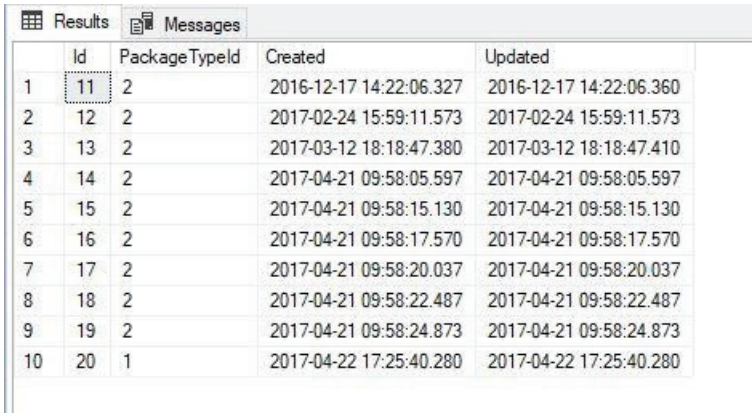


Results | Messages

	Id	Sysname
1	1	ESTATE
2	2	SPORTS
3	3	HOTELS

Εικόνα 2.50 Δεδομένα του Πίνακα PackageTypes _TB

Στην συνέχεια υπάρχει ο Πίνακας Packages_TB ο οποίος περιλαμβάνει τα "πακέτα" τα οποία έχουν δημιουργηθεί καθώς και πληροφορίες για αυτά όπως για παράδειγμα πότε δημιουργήθηκαν, πότε ήταν η τελευταία φορά όπου ενημερώθηκαν κτλ.

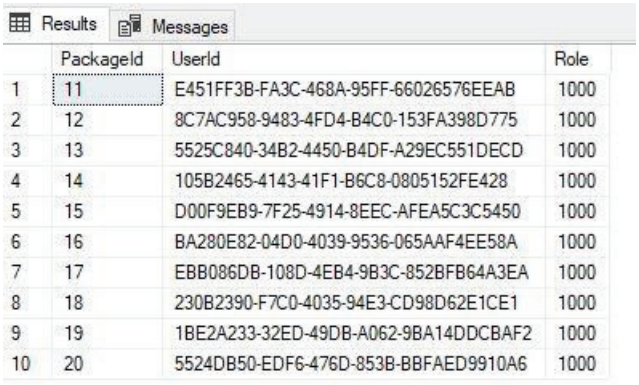


Results | Messages

	Id	PackageTypeId	Created	Updated
1	11	2	2016-12-17 14:22:06.327	2016-12-17 14:22:06.360
2	12	2	2017-02-24 15:59:11.573	2017-02-24 15:59:11.573
3	13	2	2017-03-12 18:18:47.380	2017-03-12 18:18:47.410
4	14	2	2017-04-21 09:58:05.597	2017-04-21 09:58:05.597
5	15	2	2017-04-21 09:58:15.130	2017-04-21 09:58:15.130
6	16	2	2017-04-21 09:58:17.570	2017-04-21 09:58:17.570
7	17	2	2017-04-21 09:58:20.037	2017-04-21 09:58:20.037
8	18	2	2017-04-21 09:58:22.487	2017-04-21 09:58:22.487
9	19	2	2017-04-21 09:58:24.873	2017-04-21 09:58:24.873
10	20	1	2017-04-22 17:25:40.280	2017-04-22 17:25:40.280

Εικόνα 2.51 Δεδομένα του Πίνακα Packages _TB

Επίσης υπάρχει ο Πίνακας PackageUsers_TB ο οποίος ουσιαστικά αναφέρει σε ποιο "πακέτο" ανήκει ο κάθε χρήστης του συστήματος.



Results | Messages

	PackageId	UserId	Role
1	11	E451FF3B-FA3C-468A-95FF-66026576EEAB	1000
2	12	8C7AC958-9483-4FD4-B4C0-153FA398D775	1000
3	13	5525C840-34B2-4450-B4DF-A29EC551DECD	1000
4	14	105B2465-4143-41F1-B6C8-0805152FE428	1000
5	15	D00F9EB9-7F25-4914-8EEC-AFEA5C3C5450	1000
6	16	BA280E82-04D0-4039-9536-065AAF4EE58A	1000
7	17	EBB086DB-108D-4EB4-9B3C-852BFB64A3EA	1000
8	18	230B2390-F7C0-4035-94E3-CD98D62E1CE1	1000
9	19	1BE2A233-32ED-49DB-A062-9BA14DDCBAF2	1000
10	20	5524DB50-EDF6-476D-853B-BBFAED9910A6	1000

Εικόνα 2.52 Δεδομένα του Πίνακα PackageUsers _TB

Τέλος σε αυτήν την ενότητα υπάρχουν και οι Πίνακες οι οποίοι κρατάνε κάποιο “Ιστορικό” για τα packges και προσδιορίζουν τα χαρακτηριστικά του καθενός. Ουσιαστικά πρόκειται για μία αντιστοιχία δυνατοτήτων στους χρήστες του συστήματος ανάλογα σε πιο πακέτο ανήκουν. Π.χ. αν ανήκει στο 1ο πακέτο μπορεί να καταχωρήσει μέχρι 3 αγώνες, αν ανήκει στο 2ο μπορεί να καταχωρήσει μέχρι 20 αγώνες αλλά από αυτούς μπορεί να έχει μόνο τους 5 active στον χάρτη. κ.ο.κ.

Results Messages

	Id	Sysname
1	1	POINTVIEWS_ACTIVE
2	2	POINTVIEWS_HISTORY
3	3	POINTVIEW_IMAGES
4	4	POINTVIEWS_PROMOTED
5	5	MEMBERS_CAPACITY

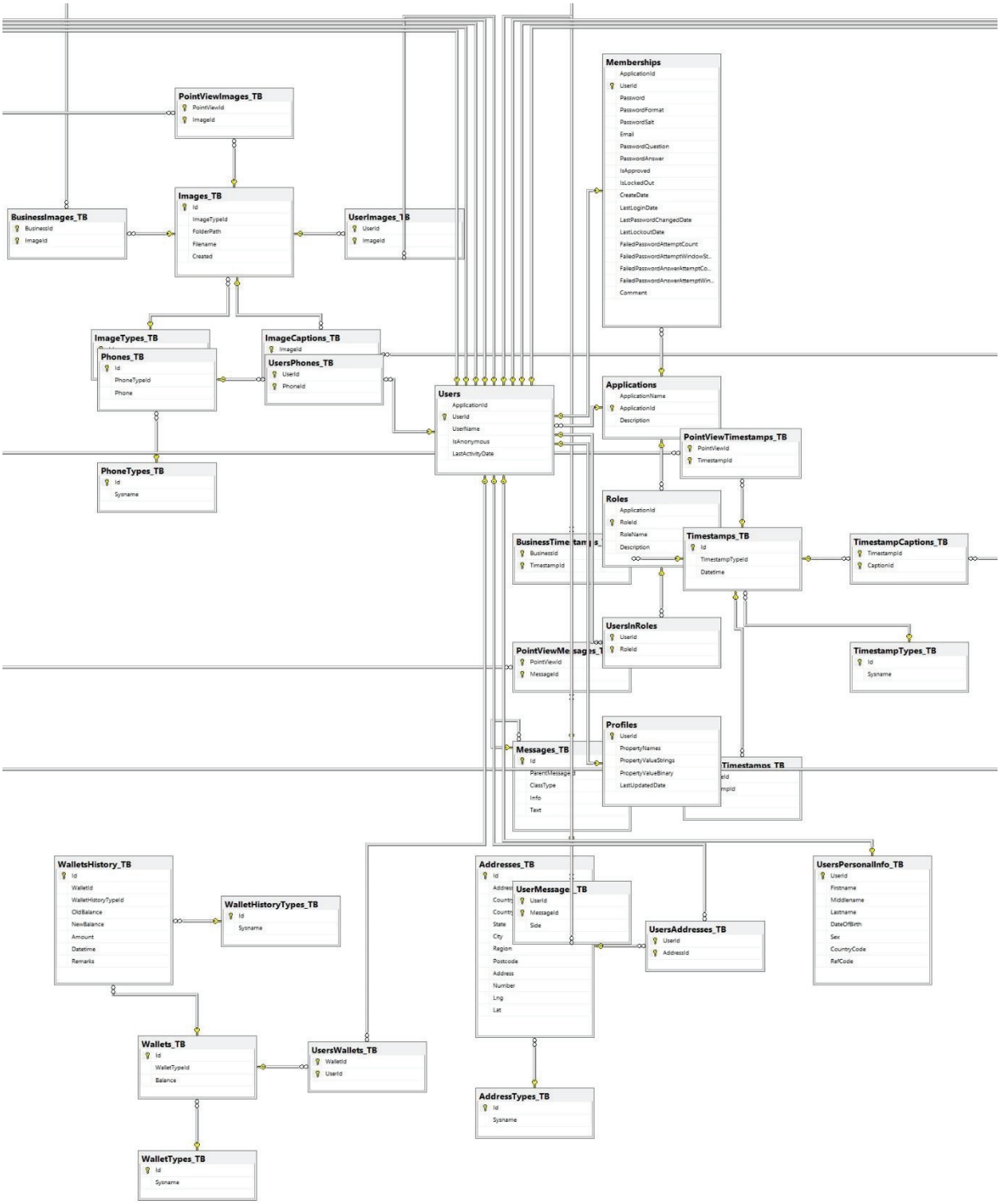
Εικόνα 2.53 Δεδομένα του Πίνακα PackageHistoryTypes _TB

Results Messages

	Id	PackageId	PackageHistoryTypeId	Value	Created	ValidUntil	GroupId
1	31	11	1	1.00	2016-12-17 14:22:07.280	NULL	NULL
2	32	11	2	5.00	2016-12-17 14:22:07.280	NULL	NULL
3	33	11	3	5.00	2016-12-17 14:22:07.280	NULL	NULL
4	34	11	4	3.00	2016-12-17 14:22:07.280	NULL	NULL
5	35	11	5	1.00	2016-12-17 14:22:07.280	NULL	NULL
6	36	11	2	6.00	2016-12-30 18:52:33.647	NULL	NULL
7	37	11	3	6.00	2016-12-30 18:52:42.850	NULL	NULL
8	38	11	2	10.00	2017-01-02 18:39:10.477	NULL	NULL
9	39	11	3	10.00	2017-01-02 18:39:18.613	NULL	NULL
10	40	11	1	3.00	2017-01-14 09:03:24.343	NULL	NULL
11	41	12	1	NULL	2017-02-24 15:59:11.697	2018-02-24 23:59:59.000	NULL
12	42	12	2	NULL	2017-02-24 15:59:11.697	NULL	NULL
13	43	12	3	50.00	2017-02-24 15:59:11.697	NULL	NULL
14	44	12	4	NULL	2017-02-24 15:59:11.697	NULL	NULL
15	45	12	5	10.00	2017-02-24 15:59:11.697	NULL	NULL
16	46	13	1	1.00	2017-03-12 18:19:05.323	NULL	NULL
17	47	13	2	5.00	2017-03-12 18:19:05.323	NULL	NULL
18	48	13	3	5.00	2017-03-12 18:19:05.323	NULL	NULL
19	49	13	4	0.00	2017-03-12 18:19:05.323	NULL	NULL

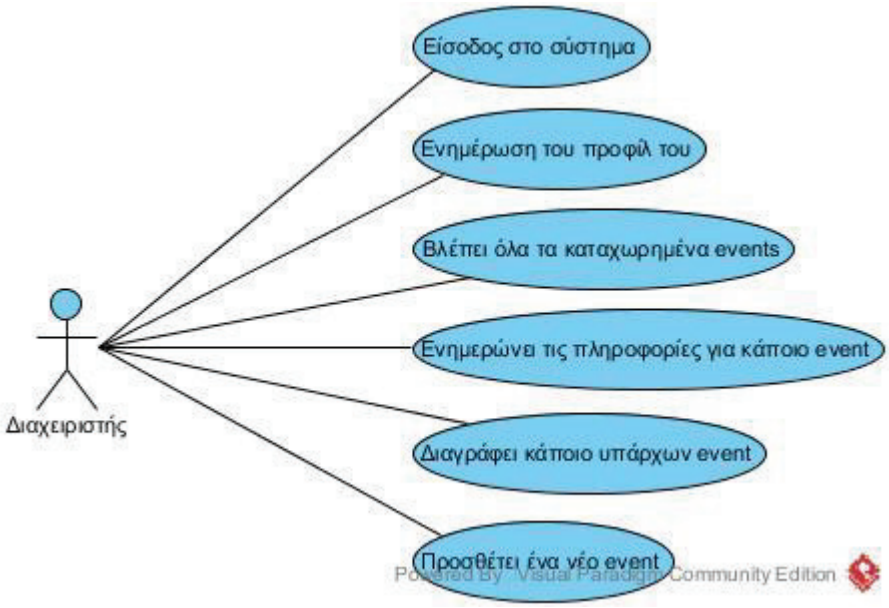
Εικόνα 2.54 Δεδομένα του Πίνακα PackageHistory _TB

- Και φτάνοντας στο τέλος της ανάλυσης της Βάσης του συστήματος αφήσαμε ένα από τα πιο σημαντικά κομμάτια το οποίο αφορά τους χρήστες του συστήματος. Στο 6ο κομμάτι λοιπόν και τελευταίο της εικόνας 2.3 υπάρχουν ομαδοποιημένες 33 κλάσεις όπου επικοινωνούν μεταξύ τους και στην συνέχεια με το υπόλοιπο σύστημα. Όλοι λοιπόν αυτοί οι πίνακες περιέχουν στοιχεία τα οποία αφορούν τους Users του συστήματος. Υπάρχει μόνο ένα μικρό κομμάτι σε αυτήν την ενότητα όπου αναφέρεται στις εικόνες όπου μπορούν να καταχωρηθούν σε κάποιο event συλλέγοντας πληροφορίες για αυτές.

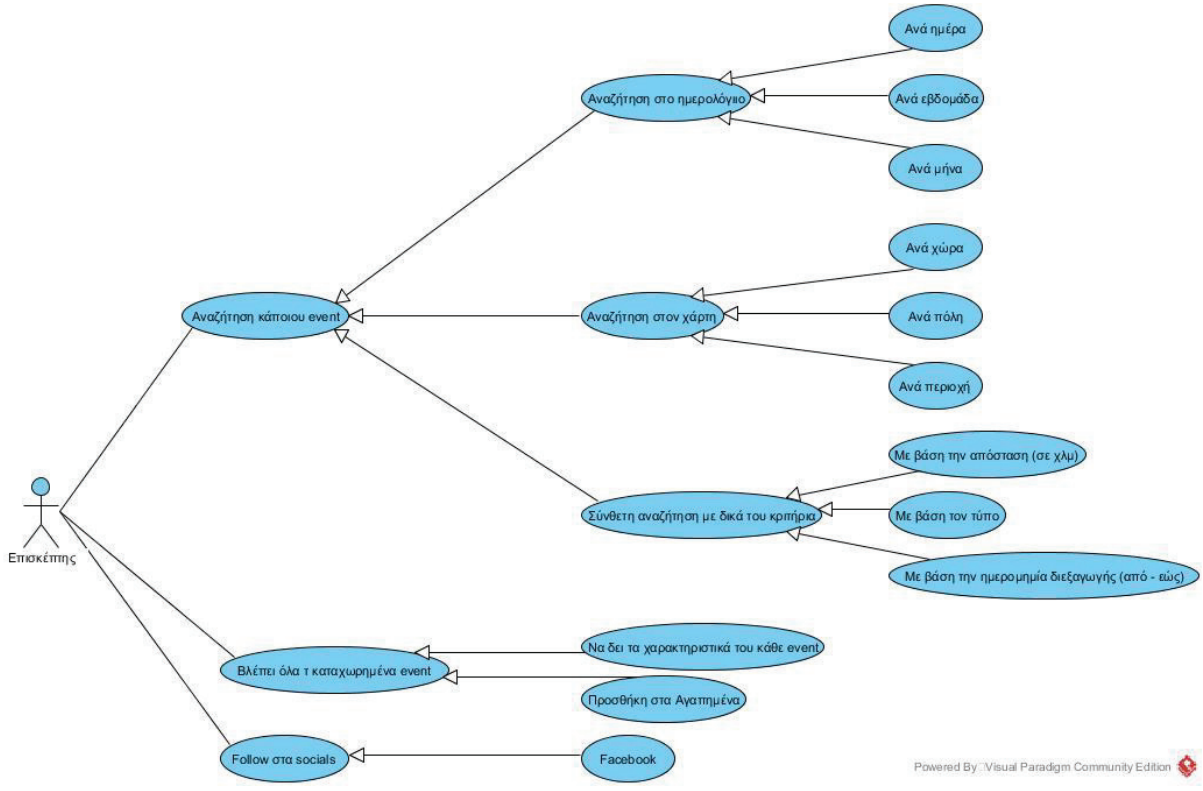


Εικόνα 2.55 Μέρος της Βάσης Δεδομένων

2.4.3. Συνολικό Μοντέλο Περιπτώσεων Χρήσης



Εικόνα 2.56 Διάγραμμα Περιπτώσεων Χρήσης του Διαχειριστή



Εικόνα 2.57 Διάγραμμα Περιπτώσεων Χρήσης του Επισκέπτη

2.4.4. Περιγραφή Περιπτώσεων Χρήσης

- Ας ξεκινήσουμε την ανάλυση από το Διάγραμμα Περιπτώσεων Χρήσης του Διαχειριστή. Ουσιαστικά στο διάγραμμα αυτό μπορούμε να δούμε μαζεμένες όλες τις λειτουργίες που μπορεί να εκτελέσει ο Διαχειριστής του συστήματος.

Αρχικά το πρώτο πράγμα που πρέπει να κάνει ο Διαχειριστής ανοίγοντας την αντίστοιχη εφαρμογή του συστήματος, είναι να συνδεθεί με το σύστημα ώστε να συνεχίσει στις υπόλοιπες ενέργειες. Το μόνο που χρειάζεται να κάνει είναι να προσθέσει στα αντίστοιχα πεδία, το username και το Password που διαθέτει.

Στην συνέχεια αναφέρονται οι υπόλοιπες λειτουργίες όπου μπορεί να εκτελέσει ο διαχειριστής. Μία από αυτές είναι η ενημέρωση του προφίλ του, όπου στην συγκεκριμένη περίπτωση, πατώντας το button όπου είναι διαθέσιμο στο View μπορεί να προσθέσει κάποια νέα εικόνα κτλ. Επίσης κάνοντας είσοδο στην εφαρμογή, στην πρώτη και κύρια σελίδα που έρχεται, μπορεί να δει όλα τα events τα οποία είναι καταχωρημένα, καθώς μπορεί και να τα ενημερώσει (αλλάζοντας κάποια ή κάποιες από τις πληροφορίες τους) είτε να τα διαγράψει ολοκληρωτικά. Τέλος μία ακόμη λειτουργία που μπορεί να εκτελέσει ο Διαχειριστής μέσα στην εφαρμογή είναι προφανώς και η δημιουργία ενός νέου event, περνώντας όλα τα χαρακτηριστικά που επιθυμεί να εμφανίζονται.

- Στη συνέχεια πάμε να δούμε το Διάγραμμα Περιπτώσεων Χρήσης ενός απλού Επισκέπτη της εφαρμογής. Όπως και πριν στο διάγραμμα αυτό μπορούμε να διακρίνουμε τις λειτουργίες όπου μπορεί να εκτελέσει κάποιος ο οποίος θα επισκεφτεί την εφαρμογή μας σαν ένας απλός χρήστης ο οποίος δεν χρειάζεται (τουλάχιστον σε πρώτη φάση) να κάνει κάποιου είδους είσοδο, βλέπει από την αρχή όλη την πληροφορία που του είναι απαραίτητη.

Το πιο απλό που μπορεί να κάνει είναι να κάνει follow στην αντίστοιχη σελίδα μας στο Facebook απλά πατώντας το button που υπάρχει στο View. Επίσης ανοίγοντας την εφαρμογή μπορεί να δει όλα τα καταχωρημένα events τα οποία εμφανίζονται καθώς να βρει και όλες τις πληροφορίες που αντιστοιχούν σε αυτό. Μία ακόμη λειτουργία είναι η προσθήκη κάποιου-κάποιων event-s στην λίστα με τα «Αγαπημένα» ώστε μετέπειτα να μπορεί να ανατρέξει σε αυτά πολύ πιο εύκολα και γρήγορα. Για το τέλος έχει μείνει μία από τις σημαντικότερες λειτουργίες που παρέχει η εφαρμογή η οποία είναι το search. Η αναζήτηση λοιπόν παρέχεται με τρεις διαφορετικούς τρόπους. Ο πρώτος τρόπος αναζήτησης είναι μέσα από το ημερολόγιο που διαθέτει η εφαρμογή των Χρηστών, στο οποίο μπορεί ο επισκέπτης να αναζητήσει τα events είτε ανά ημέρα, είτε ανά εβδομάδα είτε ανά μήνα. Δεύτερος τρόπος αναζήτησης είναι μέσω του χάρτη που διαθέτει η εφαρμογή, όπου εκεί τα events ομαδοποιούνται με βάση το Primary PointView όπου έχει δηλωθεί και ανάλογα του εύρους του χάρτη το οποίο εμφανίζεται εκείνη τη στιγμή στην οθόνη εμφανίζονται και τα αντίστοιχα events όπου το σημείο αναφοράς τους είναι μέσα στα όρια του χάρτη. Τρίτος λοιπόν και τελευταίος τρόπος αναζήτησης είναι με βάση τα κριτήρια όπου επιθυμεί ο επισκέπτης, όπου σε αυτήν την περίπτωση μπορεί να ορίσει τι χαρακτηριστικά θέλει να έχουν τα events που θα του εμφανιστούν (π.χ. την χιλιομετρική απόσταση της διαδρομής, την ημερομηνία διεξαγωγής είτε ένα ορισμένο εύρος ημερομηνιών, κα). Μπορεί να εκτελέσει την αναζήτηση είτε με κάποιο από τα κριτήρια όπου είναι διαθέσιμα είτε με τον συνδυασμό όλων – ή κάποιων – μαζί.

Κεφάλαιο 3 – Ανάλυση Αλγορίθμων Του Συστήματος

3.1 Μερικά σημεία επικοινωνίας του Συστήματος με την Βάση Δεδομένων

Σε αυτό το σημείο θα δούμε μερικά σημεία επικοινωνίας του Συστήματος με την Βάση Δεδομένων. Στα σημεία αυτά που θα αναφερθούμε είναι ουσιαστικά τα σημείο όπου μπορούμε να ενημερώσουμε τις τιμές που έχουμε καταχωρήσει σε κάποιους από τους πίνακες όπου αναλύσαμε στο προηγούμενο κεφάλαιο. Ο πίνακας στον οποίο αναφερόμαστε φαίνεται από το όνομα όπου έχει η κάθε μέθοδος καθώς ακολουθείται ένας συγκεκριμένος τύπος ονοματολογίας ο οποίος είναι _UpdateDB(table_name). Στην συνέχεια έχουμε παραθέσει όλο το κομμάτι του κώδικα που αφορά αυτή τη λειτουργία του Συστήματος. Ακολουθεί λοιπόν ο κώδικας και παρακάτω υπάρχουν κάποιες εικόνες οι οποίες απεικονίζουν το ίδιο μέρος μέσα στο περιβάλλον ανάπτυξης κώδικα όπου μπορούμε να διακρίνουμε καλύτερα τις μεθόδους και το εσωτερικό κάποιων από αυτών.

```
1.  #region Database Update
2.          protected virtual void _UpdatePointviewInfo(GeodataDBContext db)
3.          {
4.              var pv = db.PointViews_TB.Single(x => x.Id == _PointviewContext.PointviewId)
    ;
5.
6.              pv.Updated = DateTime.UtcNow;
7.          }
8.          protected virtual void _UpdateDbMetrics(IEnumerable<PointviewMetric> metrics, Ge
    odataDBContext db)
9.          {
10.             #region Delete old metrics
11.             var deleteableMetrics = db.Metrics_TB.Where(x => x.PointViews_TB.Any(p => p.
    Id == _PointviewContext.PointviewId));
12.             db.Metrics_TB.RemoveRange(deleteableMetrics);
13.             #endregion
14.
15.             #region Insert new metrics
16.             var dbpointview = db.PointViews_TB.Single(x => x.Id == _PointviewContext.Poi
    ntviewId);
17.             var insertables = metrics.Where(x => x.Value.HasValue);
18.             foreach (var m in insertables)
19.             {
20.                 Metrics_TB dbmetric = new Metrics_TB
21.                 {
22.                     MetricTypeId = db.MetricTypes_TB.FirstOrDefault(x => x.Sysname == m.
    MetricSysname).Id,
23.                     Value = m.Value.Value
24.                 };
25.
26.                 dbpointview.Metrics_TB.Add(dbmetric);
27.             }
28.             #endregion
29.         }
30.         protected virtual void _UpdateDbFlags(IEnumerable<PointviewFlag> flags, GeodataD
    BContext db)
31.         {
32.             #region Delete old flags
33.             var deleteableFlags = db.Flags_TB.Where(x => x.PointViews_TB.Any(p => p.Id =
    = _PointviewContext.PointviewId));
34.             db.Flags_TB.RemoveRange(deleteableFlags);
35.             #endregion
36.
37.             #region Insert new flags
```

```
38.             var dbpointview = db.PointViews_TB.Single(x => x.Id == _PointviewContext.Poi
    ntviewId);
39.             var insertables = flags.Where(x => x.Value.HasValue);
40.             foreach (var m in insertables)
41.             {
42.                 Flags_TB dbflag = new Flags_TB
43.                 {
44.                     FlagTypeId = db.FlagTypes_TB.FirstOrDefault(x => x.Sysname == m.Flag
    TypeSysname).Id,
45.                     Value = m.Value.Value
46.                 };
47.
48.                 dbpointview.Flags_TB.Add(dbflag);
49.             }
50.             #endregion
51.         }
52.         protected virtual void _UpdateDbTimestamps(IEnumerable<PointviewTimestamp> times
    tamps, GeodataDBContext db)
53.         {
54.             #region Delete old timestamps
55.             var deleteables = db.Timestamps_TB.Where(x => x.PointViews_TB.Any(p => p.Id
    == _PointviewContext.PointviewId));
56.             db.Timestamps_TB.RemoveRange(deleteables);
57.             #endregion
58.
59.             #region Insert new timestamps
60.             var dbpointview = db.PointViews_TB.Single(x => x.Id == _PointviewContext.Poi
    ntviewId);
61.             var insertables = timestamps.Where(x => x.Value.HasValue);
62.             foreach (var m in insertables)
63.             {
64.                 Timestamps_TB dbtimestamp = new Timestamps_TB
65.                 {
66.                     TimestampTypeId = db.TimestampTypes_TB.FirstOrDefault(x => x.Sysname
     == m.TimestampTypeSysname).Id,
67.                     Datetime = m.Value.Value.ToUniversalTime()
68.                 };
69.
70.                 if (m.Captions != null)
71.                     foreach (var capsgroup in m.Captions.Where(x => x.Value.Captions.Any
    (c => c.Caption != null)))
72.                         foreach (var caps in capsgroup.Value.Captions)
73.                         {
74.                             var lang = db.Languages_TB.Single(x => x.Sysname == caps.Lan
    guageSysname);
75.
76.                             dbtimestamp.Captions_TB.Add(new Captions_TB
77.                             {
78.                                 Caption = caps.Caption,
79.                                 CaptionTypeId = db.CaptionTypes_TB.FirstOrDefault(x => x
    .Sysname == caps.CaptionTypeSysname).Id,
80.                                 LanguageId = lang.Id
81.                             });
82.                         }
83.
84.                 dbpointview.Timestamps_TB.Add(dbtimestamp);
85.             }
86.             #endregion
87.         }
88.         protected virtual void _UpdateDbCaptionsShort(Dictionary<string, PointviewCaptio
    nsGroup> captionsShort, GeodataDBContext db)
89.         {
90.             #region Delete old captions
91.             var deleteables = db.Captions_TB.Where(x => x.PointViews_TB.Any(p => p.Id ==
     _PointviewContext.PointviewId));
92.             db.Captions_TB.RemoveRange(deleteables);
```

```
93.             #endregion
94.
95.             #region Insert new captions
96.             var dbpointview = db.PointViews_TB.Single(x => x.Id == _PointviewContext.Poi
    ntviewId);
97.
98.             foreach (var key in captionsShort.Keys)
99.             {
100.                    PointviewCaption[] Captions = captionsShort[key].Captions;
101.
102.                    foreach (var caps in Captions.Where(x => !string.IsNullOrWhiteSpace(
    x.Caption)))
103.                    {
104.                        var lang = db.Languages_TB.Single(x => x.Sysname == caps.Languag
    eSysname);
105.
106.                        Captions_TB dbcaps = new Captions_TB
107.                        {
108.                            Caption = caps.Caption,
109.                            LanguageId = lang.Id,
110.                            CaptionTypeId = db.CaptionTypes_TB.FirstOrDefault(ct => ct.S
    ysname == caps.CaptionTypeSysname).Id,
111.                        };
112.
113.                        dbpointview.Captions_TB.Add(dbcaps);
114.                    }
115.                }
116.                #endregion
117.            }
118.            protected virtual void _UpdateDbCaptionsLong(Dictionary<string, PointviewCap
    tionsGroup> captionsLong, GeodataDBContext db)
119.            {
120.                #region Delete old captions
121.                var deleteables = db.CaptionsLong_TB.Where(x => x.PointViews_TB.Any(p =>
     p.Id == _PointviewContext.PointviewId));
122.                db.CaptionsLong_TB.RemoveRange(deleteables);
123.                #endregion
124.
125.                #region Insert new captions
126.                var dbpointview = db.PointViews_TB.Single(x => x.Id == _PointviewContext
    .PointviewId);
127.
128.                foreach (var key in captionsLong.Keys)
129.                {
130.                    PointviewCaption[] Captions = captionsLong[key].Captions;
131.
132.                    foreach (var caps in Captions.Where(x => !string.IsNullOrWhiteSpace(
    x.Caption)))
133.                    {
134.                        var lang = db.Languages_TB.Single(x => x.Sysname == caps.Languag
    eSysname);
135.
136.                        CaptionsLong_TB dbcaps = new CaptionsLong_TB
137.                        {
138.                            Caption = caps.Caption,
139.                            LanguageId = lang.Id,
140.                            CaptionTypeId = db.CaptionTypes_TB.FirstOrDefault(ct => ct.S
    ysname == caps.LanguageSysname).Id,
141.                        };
142.
143.                        dbpointview.CaptionsLong_TB.Add(dbcaps);
144.                    }
145.                }
146.                #endregion
147.            }
```

```
148.           protected virtual void _UpdateDbCharacteristics(Dictionary<int, PointviewCha
    racteristic> characteristics, GeodataDBContext db)
149.           {
150.               #region Delete old characteristics
151.               var deleteables = db.PointViewCharacteristics_TB.Where(x => x.PointviewI
    d == _PointviewContext.PointviewId);
152.               db.PointViewCharacteristics_TB.RemoveRange(deleteables);
153.               #endregion
154.
155.               #region Insert new characteristics
156.               var dbpointview = db.PointViews_TB.Single(x => x.Id == _PointviewContext
    .PointviewId);
157.
158.               foreach (var key in characteristics.Keys)
159.               {
160.                   PointviewCharacteristic Characteristic = characteristics[key];
161.
162.                   var dbPointviewCharacteristic = new PointViewCharacteristics_TB
163.                   {
164.                       CharacteristicId = Characteristic.CharacteristicId,
165.                   };
166.
167.
168.                   foreach (var opt in Characteristic.Options)
169.                   {
170.                       var dbPointviewCharacteristicOption = new PointviewCharacteristi
    cOptions_TB
171.                       {
172.                           CharacteristicOptionId = opt.Id
173.                       };
174.
175.                       dbPointviewCharacteristic.PointviewCharacteristicOptions_TB.Add(
    dbPointviewCharacteristicOption);
176.
177.                       foreach (var capkey in opt.Captions.Keys)
178.                       {
179.                           var caps = opt.Captions[capkey];
180.
181.                           if (caps.Captions == null)
182.                               continue;
183.
184.                           foreach (var cap in caps.Captions.Where(x => !string.IsNullO
    rWhiteSpace(x.Caption)))
185.                           {
186.                               var lang = db.Languages_TB.Single(x => x.Sysname == cap.
    LanguageSysname);
187.
188.                               Captions_TB dbcaps = new Captions_TB
189.                               {
190.                                   Caption = cap.Caption,
191.                                   LanguageId = lang.Id,
192.                                   CaptionTypeId = db.CaptionTypes_TB.FirstOrDefault(ct
     => ct.Sysname == cap.CaptionTypeSysname).Id
193.                               };
194.
195.                               dbPointviewCharacteristicOption.Captions_TB.Add(dbcaps);

196.                           }
197.                       }
198.
199.                       dbpointview.PointViewCharacteristics_TB.Add(dbPointviewCharacter
    istic);
200.                   }
201.               }
202.               #endregion
203.           }
```

```
204.           protected virtual void _UpdateDbImageCaptions(Dictionary<int, PointviewCapti
    onsGroup> _PendingImageCaptions, GeodataDBContext db)
205.           {
206.               var imagesIds = _PointviewContext.Images.SelectMany(x => x.Value.Select(
    i => i.Id)).ToArray();
207.
208.               #region Delete old captions
209.               var deleteables = db.Captions_TB.Where(x => x.Images_TB.Any(p => imagesI
    ds.Contains(p.Id)));
210.               db.Captions_TB.RemoveRange(deleteables);
211.               #endregion
212.
213.               #region Insert new captions
214.               foreach (var key in _PendingImageCaptions.Keys)
215.               {
216.                   var dbimage = db.Images_TB.Single(x => x.Id == key);
217.                   PointviewCaption[] Captions = _PendingImageCaptions[key].Captions;
218.
219.                   foreach (var caps in Captions.Where(x => !string.IsNullOrWhiteSpace(
    x.Caption)))
220.                   {
221.                       var lang = db.Languages_TB.Single(x => x.Sysname == caps.Languag
    eSysname);
222.
223.                       Captions_TB dbcaps = new Captions_TB
224.                       {
225.                           Caption = caps.Caption,
226.                           LanguageId = lang.Id,
227.                           CaptionTypeId = db.CaptionTypes_TB.FirstOrDefault(ct => ct.S
    ysname == caps.CaptionTypeSysname).Id
228.                       };
229.
230.                       dbimage.Captions_TB.Add(dbcaps);
231.                   }
232.               }
233.               #endregion
234.           }
235.
236.           protected virtual void _DeleteRemovedImages(int[] removedImagesIds, GeodataD
    BContext db)
237.           {
238.               if (removedImagesIds == null)
239.                   return;
240.
241.               #region Delete removed images
242.               var deletables = db.Images_TB.Where(x => removedImagesIds.Contains(x.Id)
    );
243.               db.Images_TB.RemoveRange(deletables);
244.               #endregion
245.           }
246.           protected virtual string[] _UploadImages(UploadFileInfo[] images)
247.           {
248.               GenericPointviewImageUploader uploader = PointviewImageUploaders.Instanc
    e.CreateUploader(_PointviewContext.PointviewSysname, null);
249.
250.               return uploader.Upload(images, _PointviewContext.PointviewId);
251.           }
252.           #endregion
```

```
#region Database Update
protected virtual void _UpdatePointviewInfo(GeodataDBContext db)...
protected virtual void _UpdateDbMetrics(IEnumerable<PointviewMetric> metrics, GeodataDBContext db)...
protected virtual void _UpdateDbFlags(IEnumerable<PointviewFlag> flags, GeodataDBContext db)...
protected virtual void _UpdateDbTimestamps(IEnumerable<PointviewTimestamp> timestamps, GeodataDBContext db)...
protected virtual void _UpdateDbCaptionsShort(Dictionary<string, PointviewCaptionsGroup> captionsShort, GeodataDBContext db)...
protected virtual void _UpdateDbCaptionsLong(Dictionary<string, PointviewCaptionsGroup> captionsLong, GeodataDBContext db)...
protected virtual void _UpdateDbCharacteristics(Dictionary<int, PointviewCharacteristic> characteristics, GeodataDBContext db)...
protected virtual void _UpdateDbImageCaptions(Dictionary<int, PointviewCaptionsGroup> _PendingImageCaptions, GeodataDBContext db)...

protected virtual void _DeleteRemovedImages(int[] removedImagesIds, GeodataDBContext db)...
protected virtual string[] _UploadImages(UploadFileInfo[] images)...
#endregion
```

Εικόνα 3.1 Σημείο επαφής του Συστήματος με την Βάση Δεδομένων

```
#region Database Update
protected virtual void _UpdatePointviewInfo(GeodataDBContext db)
{
    var pv = db.PointViews_TB.Single(x => x.Id == _PointviewContext.PointviewId);

    pv.Updated = DateTime.UtcNow;
}
protected virtual void _UpdateDbMetrics(IEnumerable<PointviewMetric> metrics, GeodataDBContext db)
{
    #region Delete old metrics
    var deleteableMetrics = db.Metrics_TB.Where(x => x.PointViews_TB.Any(p => p.Id == _PointviewContext.PointviewId));
    db.Metrics_TB.RemoveRange(deleteableMetrics);
    #endregion

    #region Insert new metrics
    var dbpointview = db.PointViews_TB.Single(x => x.Id == _PointviewContext.PointviewId);
    var insertables = metrics.Where(x => x.Value.HasValue);
    foreach (var m in insertables)
    {
        Metrics_TB dbmetric = new Metrics_TB
        {
            MetricTypeId = db.MetricTypes_TB.FirstOrDefault(x => x.Sysname == m.MetricSysname).Id,
            Value = m.Value.Value
        };

        dbpointview.Metrics_TB.Add(dbmetric);
    }
    #endregion
}
```

Εικόνα 3.2 Σημείο επαφής του Συστήματος με την Βάση Δεδομένων

Στην εικόνα 3.2 μπορούμε να διακρίνουμε δύο από τις μεθόδους ανανέωσης των δεδομένων της Βάσης. Στην πρώτη μέθοδο ουσιαστικά περνάμε την τελευταία ημερομηνία όπου αντιστοιχεί στην τελευταία ενημέρωση που έγινε στο αντίστοιχο event. Ενώ στην δεύτερη ανανεώνουμε τα σημεία τα οποία έχουν καταχωρηθεί για κάποιο event. Όπως μπορούμε να δούμε ο τρόπος επικοινωνίας είναι σχετικά απλός και βασίζεται σχεδόν όλες τις φορές στο ID όπου έχει το κάθε event στην Βάση, και είπαμε σχεδόν όλες τις φορές καθώς υπάρχουν και περιπτώσεις όπου η αναζήτηση/καταχώρηση σε κάποιον πίνακα γίνεται με βάση το UserId ή οποιαδήποτε άλλο κύριο πεδίο για το σύστημα όπως π.χ. το Package, το PiontViewType κα.

Στην συνέχεια έχουμε προσθέσει ακόμη ένα μέρος του συστήματος που επικοινωνεί με την βάση το οποίο είναι το μέρος όπου γίνεται μία νέα καταχώρηση ενός event.

```
1.  abstract class PointviewCreator
2.      {
3.          protected abstract string PointviewSysname { get; }
4.          protected abstract PointviewContext Create(PointViews_TB pointview, GeodataDBCon
    text db);
5.
6.          public PointviewContext Create()
7.          {
8.              try
9.              {
10.                 PointViews_TB pv = new PointViews_TB
11.                 {
12.                     Created = DateTime.UtcNow,
13.                     Updated = DateTime.UtcNow,
14.                     IsActive = false,
15.                 };
16.
17.                 string[] navigationViewSysnames = DefaultNavigationViews.Instance.Get(Po
    intviewSysname);
18.
19.                 using (GeodataDBContext db = new GeodataDBContext())
20.                 {
21.                     pv.PointviewTypeId = db.PointViewsTypes_TB.Single(x => x.Sysname ==
    PointviewSysname).Id;
22.                     db.PointViews_TB.Add(pv);
23.
24.                     var navigationViews = db.NavigationViews_TB.Where(x => navigationVie
    wSysnames.Contains(x.Sysname));
25.                     foreach (var nv in navigationViews)
26.                         nv.PointViews_TB.Add(pv);
27.
28.                     db.SaveChanges();
29.
30.                     PointviewContext pc = Create(pv, db);
31.                     pc.PointviewId = pv.Id;
32.                     pc.Created = pv.Created;
33.                     pc.Updated = pv.Updated;
34.
35.                     return pc;
36.                 }
37.             }
38.             catch (Exception ex)
39.             {
40.                 LibAppLoggers.Instance.Pointviews.LogError($"PointviewCreator: {Pointvie
    wSysname} Create()", ex);
41.                 throw;
42.             }
43.         }
44.     }
```

3.2 Αυτόματη Αφαίρεση των Events από το Ημερολόγιο

Μία από τις λειτουργίες που μας παρέχει το σύστημα είναι η αυτόματη αφαίρεση των Events από το ημερολόγιο όταν περάσει η ημερομηνία διεξαγωγής τους. Ένα μέρος του κώδικα φαίνεται στην συνέχεια.

```
protected override void DoJob(object state)
        {
            try
            {
                int[] pointviewIds = null;
                using (GeodataDBContext db = new GeodataDBContext())
                {
                    DateTime now = DateTime.UtcNow.AddHours(-_PassedHours);

                    var pointviews = db.PointViews_TB
                        .Where(x => x.IsActive && _PointviewSysnames.Contains(x.PointViewsTypes_TB.Sysname)
                        && x.Timestamps_TB.FirstOrDefault(t => t.TimestampTypes_TB.Sysname == GDCommon.Config.Sysnames.TimestampTypes.SPORTS_START_TIME).Datetime < now);

                    foreach (var pv in pointviews)
                    {
                        PointviewPersistence.Deactivate(pv);
                        PointviewsStateCache.Instance.TryUpdate(pv.Id, (pointviewState) =>
                        {
                            pointviewState.IsActive = false;
                        });
                    }

                    pointviewIds = pointviews.Select(x => x.Id).ToArray();

                    if (pointviewIds.Length > 0)
                        db.SaveChanges();
                }

                if (pointviewIds.Length > 0)
                {
                    Tuple<int, bool>[] pointviewsState = pointviewIds.Select(x => new Tuple<int, bool>(x, false)).ToArray();

                    using (GeodataDBContext db = new GeodataDBContext())
                        db.LogSysEvent(new PointviewsActiveStateChanged(pointviewsState));
                }
            }
            catch (Exception ex)
            {
                LibAppLoggers.Instance.Daemons.LogError("RaceDeactivationDaemon.DoJob()", ex);
            }
        }

```

3.3 Χρήσιμα Σημεία Κώδικα

Στο Back-End του συστήματος υπάρχουν και κάποιες ακόμα λειτουργίες που εκτελούνται που αξίζει σε αυτό το σημείο να αναφέρουμε και να αναλύσουμε τον κώδικα που υπάρχει πίσω από αυτές. Μία από αυτές είναι η εύρεση διάφορων χρήσιμων πληροφοριών μέσω της IP του χρήστη. Τέτοιες πληροφορίες για παράδειγμα μπορεί να είναι το Country Code. Ο κώδικας που το υλοποιεί αυτό όπως θα δείτε και στην συνέχεια είναι αρκετά απλός και κατανοητός.

```
1.  class IP2CImplementation
2.      {
3.          string serviceUrl = "http://ip2c.org";
4.
5.          public IpInfo GetCountryCode(int num)
6.          {
7.              WebClient wc = new WebClient();
8.
9.              string response = wc.DownloadString($"{serviceUrl}/{num}");
10.
11.             return ParseResponse(response);
12.         }
13.
14.         public IpInfo GetCountryCode(string ip)
15.         {
16.             WebClient wc = new WebClient();
17.
18.             string response = wc.DownloadString($"{serviceUrl}/{ip}");
19.
20.             return ParseResponse(response);
21.         }
22.
23.         public IpInfo ParseResponse(string response)
24.         {
25.             string[] parts = response.Split(';');
26.
27.             IpInfo ipInfo = new IpInfo {
28.                 IpInfoResultType = (IpInfoResultType)Enum.Parse(typeof(IpInfoResultT
    ype), parts[0])
29.             };
30.
31.
32.             if (ipInfo.IpInfoResultType == IpInfoResultType.Success) {
33.                 ipInfo.CountryCode = parts[1];
34.                 ipInfo.CountryCode3Letters = parts[2];
35.                 ipInfo.CountryName = parts[3];
36.             }
37.
38.             return ipInfo;
39.         }
40.     }
```

Η δεύτερη λειτουργία που αξίζει να αναλύσουμε σε αυτό το σημείο αφορά τις εικόνες που έχουμε περασμένες στο σύστημα. Ουσιαστικά πρόκειται για μία μέθοδο στην οποία γίνεται κάποια επεξεργασία των καταχωρημένων εικόνων με σκοπό να μην επιβαρύνεται άσκοπα το σύστημα. Ο τρόπος που λειτουργεί είναι αρκετά απλός. Αν πχ ανεβάσει ο διαχειριστής μία εικόνα σε ένα x μέγεθος και με ZxW διαστάσεις και θέλεις στην συνέχεια να την εμφανίσεις και σε κάποιο άλλο σημείο του συστήματος στο οποίο όμως χρειάζεσαι αρκετά πιο μικρή διάσταση της εικόνας και κατά συνέπεια θα μπορούσε να είναι και αρκετά πιο μικρή σε μέγεθος, έρχεται αυτή η μέθοδος στην οποία δίνεις την ονομασία της εικόνας και σκοπός της είναι να στην γυρίσει στις διαστάσεις όπου την ζήτησες καθώς και σε πιο μικρό μέγεθος, με αποτέλεσμα να μπορεί να φορτωθεί και αρκετά πιο γρήγορα και δεν επιβαρύνεις με άσκοπα mb το site. Σε αυτήν την περίπτωση ο κώδικας είναι λίγο πιο περίπλοκος σε σχέση με πριν αλλά δεν παύει να

είναι απλός και κατανοητός. Γενικά την όλη δομή που ακολουθείται για την ανάπτυξη του συστήματος προσπαθούμε να την κρατάμε όσο πιο απλή γίνεται τόσο για ευκολία σε όσους δουλεύουν πάνω σε αυτό το σύστημα όσο και για αυτούς που πρόκειται μελλοντικά να αναλάβουν την υλοποίηση κάποιας επιπλέον λειτουργίας, καθώς επίσης θέλουμε να έχει απλή και εύκολη δομή ώστε να μπορεί να προστεθεί εύκολα στο μέλλον οποιαδήποτε λειτουργία ζητηθεί από τους πελάτες αλλά και να μπορούν να συντηρηθούν εύκολα και όσα έχουν υλοποιηθεί μέχρι στιγμής.

```
1.  public void ProcessRequest(HttpContext context)
2.          {
3.              Stopwatch sw = new Stopwatch();
4.              sw.Start();
5.
6.              string imageName = context.Request.RawUrl.Substring(URL_PREFIX.Length);
7.
8.              string fullFilePath = Path.Combine(RepositoryFolder, imageName);
9.
10.             Bitmap image = null;
11.             try
12.             {
13.                 lock (_Lock)
14.                 {
15.                     if (!File.Exists(fullFilePath))
16.                     {
17.                         if (!Regex.IsMatch(imageName, @"[a-zA-Z0-9]+_\d+X\d+.\w+"))
    //if image name is not in format "imagename_999X999.type" return not found and stop proc
    edure
18.                         {
19.                             context.Response.StatusCode = (int)HttpStatusCode.NotFound;

20.                             context.Response.SuppressContent = true;
21.                             return;
22.                         }
23.
24.                         image = CreateAndSaveImage(imageName);
25.                     }
26.                 }
27.             }
28.             catch (ImageNotFoundException) {
29.                 context.Response.StatusCode = (int)HttpStatusCode.NotFound;
30.                 context.Response.SuppressContent = true;
31.                 return;
32.             }
33.             catch (Exception ex)
34.             {
35.                 AppLoggers.Instance.Default.LogError("GDBackendUI.Handlers.ImagesReposit
    oryHandler.ProcessRequest()", ex, $"Request url: {context.Request.RawUrl}");
36.             }
37.
38.             Stream fs;
39.             if (image == null)//if we not have the image in memory get it from file syst
    em
40.                 fs = new FileStream(fullFilePath, FileMode.Open);
41.             else//in case we have the image in memory return it instead of load it from
    file system
42.                 fs = new MemoryStream(image.ToBytes());
43.
44.             context.Response.Clear();
45.             context.Response.ClearHeaders();
46.             context.Response.ContentType = FileUtils.GetMimeTypeByFilename(imageName);
47.
48.             int b;
49.             while ((b = fs.ReadByte()) != -1)
50.             {
51.                 context.Response.OutputStream.WriteByte((byte)b);
```

```
            }

            context.Response.OutputStream.Flush();
            fs.Close();

            sw.Stop();
            var ddd = sw.Elapsed;
        }
```

3.4 Ομαδοποίηση και Απεικόνιση των Events στον Χάρτη

Το συγκεκριμένο μέρος μπορεί να θεωρηθεί και ως η “καρδιά” του συστήματος αφού είναι και το πιο σημαντικό καθώς αυτό είναι υπεύθυνο για την επικοινωνία με την βάση ώστε να πάρει όλα τα καταχωρημένα και εν-ενεργά events προκειμένου να τα εμφανίσει μπροστά στον χάρτη στην εφαρμογή των χρηστών. Πρόκειται για ένα κομμάτι το οποίο αποτελείται από τους δικούς του Controllers τα δικά του models καθώς υπάρχει από πίσω ολόκληρος αλγόριθμος ο οποίος βασίζεται πάνω στα μαθηματικά και κυρίως στην γεωμετρία, μιας και μιλάμε για την απεικόνιση σημείων πάνω σε χάρτη.

Ας ξεκινήσουμε από τον Controller ο οποίος είναι και ο πιο απλός.

```
using System.Web.Mvc;
using GDFrontendWebCommon.Controllers;
using GDPointviewsAPI.ModelBuilders;
using GDPointviewsAPI.Models;
using System.Collections.Generic;
using GDWebCommon.Controllers;

namespace GDPointviewsAPI.Controllers
{
    [RoutePrefix("Api/Pointviews")]
    public class PointviewsController : GenericController
    {
        [HttpPost, LocaleRoute("GetMarkers")]
        public JsonResult GetMarkers(CoordinatesInputModel coordinates, double? radius = null, int? minPoints = null, int maxPoints = int.MaxValue)
        {
            JsonResult js = new JsonResult();

            FilterContextsModelBuilder.Instance.SetCoordinates(coordinates);

            js.Data = PointviewsModelBuilder.Instance.GetCurrentContextPointviews(radius, minPoints, maxPoints);

            return Json(js, JsonRequestBehavior.AllowGet);
        }

        [HttpPost, LocaleRoute("GetPointviews")]
        public JsonResult GetPointviews(List<int> ids)
        {
            JsonResult js = new JsonResult();

            js.Data = PointviewsModelBuilder.Instance.GetPointviews(ids.ToArray());

            return Json(js, JsonRequestBehavior.AllowGet);
        }

        [HttpGet, LocaleRoute("Search")]
        public JsonResult Search(string keyword)
        {
            JsonResult js = new JsonResult();

            js.Data = PointviewsModelBuilder.Instance.Search(keyword);
```

```
41.
42.             return Json(js, JsonRequestBehavior.AllowGet);
43.         }
44.     }
45. }
```

Ουσιαστικά πρόκειται για μία κλάση η οποία αποτελείται από τρεις μεθόδους. Η πρώτη στην γραμμή 14 είναι η GetMarkers η οποία είναι υπεύθυνη για την ομαδοποίηση των σημείων. Όπως μπορούμε να διακρίνουμε παίρνει κάποιες παραμέτρους οι οποίες είναι οι coordinates όπου αντιστοιχούν στις συντεταγμένες (αποτελείται από 4 σημεία όπου είναι τα LatBottom, LatTop, LatLeft και LatRight), τη radius η οποία είναι η ακτίνα που θέλουμε να έχει ο κύκλος που θα δημιουργηθεί καθώς και τα minPoints και maxPoints όπου είναι τα ελάχιστα και τα μέγιστα σημεία που θέλουμε να περιέχει ο κύκλος που θα δημιουργηθεί. Στη συνέχεια παρουσιάζεται η κλάση που αντιστοιχεί στην πρώτη παράμετρο που αναφέραμε. Στην γραμμή 20 ουσιαστικά επιστρέφονται τα PointViewsIDs όπου στην συνέχεια τα χρησιμοποιούμε στην επόμενη μέθοδο.

```
1.  namespace GDPointviewsAPI.Models
2.  {
3.      public class CoordinatesInputModel
4.      {
5.          public double LatBottom { get; set; }
6.          public double LatTop { get; set; }
7.          public double LngLeft { get; set; }
8.          public double LngRight { get; set; }
9.      }
10. }
```

Στην συνέχεια βλέπουμε την μέθοδο που καλείται στην γραμμή 20, την GetCurrentContextPointviews η οποία με την σειρά της παίρνει όλα τα events, τα ταξινομεί και στην συνέχεια αν έχει δοθεί ακτίνα καλεί την GroupMarkers η οποία αναλαμβάνει την ομαδοποίηση των events, καταχωρώντας τα ουσιαστικά σε κάποια Group. Ακολουθεί ο κώδικας αυτών των δύο μεθόδων.

```
1.  public IEnumerable<GenericMarker> GetCurrentContextPointviews(double? radius, int? minPo
    ints = 0, int maxPoints = int.MaxValue)
2.  {
3.      GenericFilterContext FilterContext = OperationContext.Current.FilterContext;
4.
5.      IEnumerable<GenericMarker> Markers = null;
6.
7.      ActiveCache.Instance.WithReadContext(() =>
8.      {
9.          Markers = FilterContext.GetPointviewEntities()
10.             .Select(x => GenericPointviewModelBuilder.GetBuilder(x.Sysname).GetMarker(x.
    PointviewInfo.PointviewId));
11.     });
12.
13.     if (radius.HasValue)
14.         Markers = FilterContext.GroupMarkers(Markers, radius.Value, minPoints.Value, max
    Points);
15.
16.     return Markers;
17. }
```

```
1.  public virtual IEnumerable<GenericMarker> GroupMarkers(IEnumerable<GenericMarker> marker
    s, double radius, int minPoints, int maxPoints)
2.  {
3.      var PointGroups = Points.GroupPoints(markers, radius, minPoints, maxPoints);
4.
5.      List<GenericMarker> GroupedMarkers = new List<GenericMarker>(markers.Count());
6.
7.      foreach (var pointGroup in PointGroups)
8.      {
9.          if (pointGroup.Count() > minPoints)
10.         {
11.             var AvarageLocation = PointsDbscanAlgorythm.AverageLocation(pointGroup);
12.
13.             var groupMarker = new GroupMarker
14.             {
15.
16.                 Location = new MarkerLocation
17.                 {
18.                     Lat = AvarageLocation.Y,
19.                     Lng = AvarageLocation.X,
20.                 },
21.                 ItemsCount = pointGroup.Count(),
22.                 Radius = PointsDbscanAlgorythm.MaxDistance(AvarageLocation.X, AvarageLoc
    ation.Y, pointGroup),
23.                 PointviewIds = pointGroup
24.                     .Where(x => x is PointviewMarker)
25.                     .Select(x => ((PointviewMarker)x).PointviewId)
26.                     .ToArray()
27.             };
28.
29.             GroupedMarkers.Add(groupMarker);
30.         }
31.         else
32.         {
33.             foreach (var marker in pointGroup)
34.             {
35.                 GroupedMarkers.Add((PointviewMarker)marker);
36.             }
37.         }
38.     }
39.
40.     return GroupedMarkers;
41. }
```

Η δεύτερη είναι στην γραμμή 26 η GetPointviews παίρνει την λίστα με τα PointViewsIDs όπου πήραμε από την προηγούμενη μέθοδο.

Και τέλος είναι η Search στην γραμμή 36 η οποία έχει αναλάβει ουσιαστικά την αναζήτηση κάποιου event με βάση κάποιων keywords. Η μέθοδος η οποία καλείται και βοηθάει στην υλοποίηση αυτής της λειτουργίας είναι η PointviewsModelBuilder.Instance.Search της οποίας η υλοποίηση φαίνεται παρακάτω.

```
public GenericPointviewModel[] Search(string keyword)
{
    return PointviewsApiEnvironment.Instance.PointviewsSearchProvider
        .Find(keyword)
        .Select(x => GenericPointviewModelBuilder.GetModel(x as GenericCacheEntity))
        .ToArray();
}
```

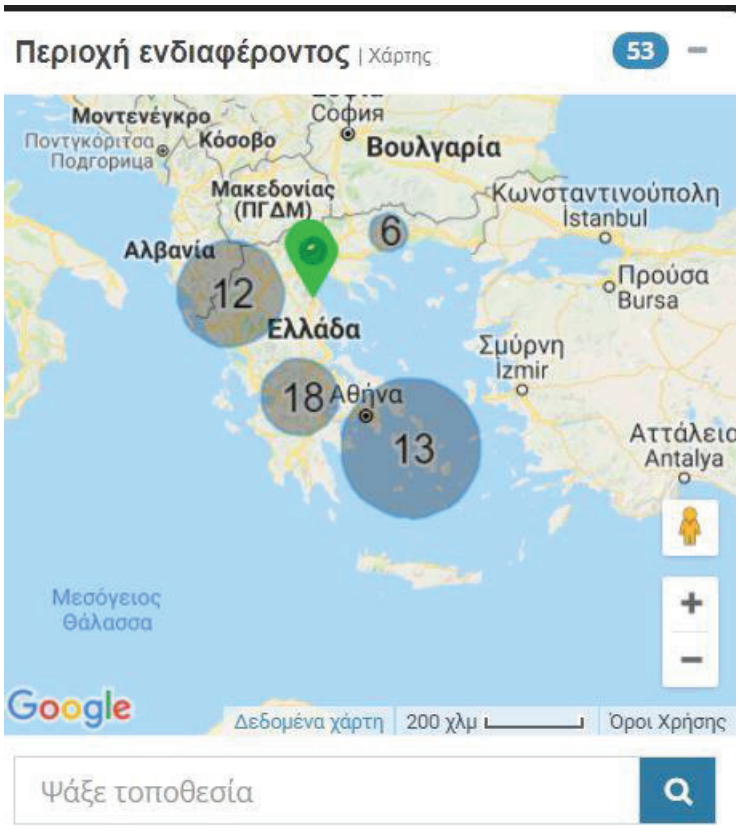
Στην συνέχεια θα δούμε κάποια σημεία όπου δημιουργούμε με javaScript κάτι σαν βιβλιοθήκη όπου κρατάμε όλες τις πληροφορίες σχετικά με την ακτίνα, τα ελάχιστα και τα μέγιστα σημεία ανάλογα με το ζουμ που έχει γίνει στον χάρτη.

Αρχείο Scripts-cb_custombindings-cb.custombindings.mampane.js

```
function _getLabelPositionAjustment(zoom) {
            switch (zoom) {
                case 15: return 0.005;
                case 14: return 0.005;
                case 13: return 0.005;
                case 12: return 0.005;
                case 11: return 0.005;
                case 10: return 0.015;
                case 9: return 0.018;
                case 8: return 0.04;
                case 7: return 0.09;
                case 6: return 0.18;
                case 5: return 0.3;
                case 5: return 0.5;
                case 4:
                case 3:
                case 2:
                case 1:
                    return 0.5;
                default: return 0;
            }
        }
```

```
    if (options.pageModel) {
        var mapContext = {
            map: map,
            getAllMarkers: function () ...,
            addMarker: function (options) ...,
            removeMarker: function (predicate) ...,
            addGroup: function (options) ...,
            clearGroups: function () ...,
            addLabel: function (options) {
                var mapLabel = new MapLabel({
                    text: options.text,
                    position: new google.maps.LatLng(options.lat + _getLabelPositionAjustment(map.zoom), options.lng),
                    map: map,
                    strokeWeight: options.strokeWeight || 3,
                    strokeColor: options.strokeColor || "#ffffff",
                    fontSize: options.fontSize || 20,
                    fontColor: options.fontColor || "#000000",
                    align: options.align || 'center'
                });

                _labels.push(mapLabel);
            },
            clearLabels: function () ...,
            setCenter: function (lat, lng) ...
        };
```



Εικόνα 3.3 Map Widget στην εφαρμογή των χρηστών

Και τέλος, για να κλείσουμε αυτή την ενότητα αξίζει να αναφέρουμε κάποια σημεία από τον μαθηματικό αλγόριθμο που χρησιμοποιήθηκε για να εξυπηρετήσει αυτή τη λειτουργία του συστήματος. Ας ξεκινήσουμε την αναφορά μας στην εύρεση της απόστασης μεταξύ δύο σημείων.

```
1.  public static double MaxDistance(double x1, double y1, IEnumerable<IPoint> points)
2.  {
3.      double max = 0;
4.
5.      foreach (var p in points)
6.      {
7.          double dist = Distance(x1, y1, p.X, p.Y);
8.          max = dist > max ? dist : max;
9.      }
10.
11.     return max;
12. }
```

```
1.  public static double Distance(double x1, double y1, double x2, double y2)
2.  {
3.      return Math.Sqrt(Math.Pow((x2 - x1), 2) + Math.Pow((y2 - y1), 2));
4.  }
```

Ουσιαστικά ο κώδικας αυτός είναι απλός. Δέχεται ως παράμετρο τέσσερα σημεία, όπου x1, y1 είναι οι συντεταγμένες του ενός σημείου και points είναι μία λίστα που περιέχει τις συντεταγμένες από όλα τα καταχωρημένα και ενενεργά events. Οπότε, δουλειά αυτής της μεθόδου είναι να βρει την απόσταση μεταξύ των σημείων με την βοήθεια της Distance, όπου ουσιαστικά η συγκεκριμένη μέθοδος δεν υλοποιεί τίποτα άλλο πέρα από τον μαθηματικό υπολογισμό για την εύρεση της απόστασης. Πάμε να καταλάβουμε λίγο καλύτερα την λογική που δουλεύει η ομαδοποίηση των events πάνω στον χάρτη.

Αρχικά ορίζουμε ένα σημείο, που θα είναι το σημείο αναφοράς μας. Στην συνέχεια ορίζουμε μία ακτίνα πάνω στην οποία θέλουμε να κινηθούμε. Έπειτα παίρνουμε ένα ένα τα events και υπολογίζουμε την απόσταση μεταξύ αυτού και του σημείου αναφοράς. Αν λοιπόν είναι εντός της ακτίνας και των ορίων που έχει σχεδιάσει ο κύκλος που ορίσαμε αρχικά μπαίνει μέσα στο group όπως και όλα τα υπόλοιπα σημεία που υπάγονται εντός των συγκεκριμένων ορίων. Επίσης ορίζεται και μία ελάχιστη απόσταση που πρέπει να έχουν μεταξύ τους τα σημεία. Μία ακόμη σημαντική λεπτομέρεια είναι ότι μπορούμε επίσης να ορίσουμε και τον ελάχιστον αριθμό events που θέλουμε να ομαδοποιηθούν στο ίδιο group καθώς επίσης και τον μέγιστο αριθμό.

Κάθε φορά που προστίθεται κάποιο σημείο σε ένα group ο υπολογισμός της απόστασης από το επόμενο γίνεται με βάση το κέντρο της απόστασης μεταξύ όλων των σημείων που υπάγονται στο αντίστοιχο group σε συνδυασμό με τις συντεταγμένες του νέου σημείου. Η μέθοδος που μας βοηθάει στην υλοποίηση αυτού είναι η ακόλουθη.

```
1.  public static Point AverageLocation(IEnumerable<IPoint> points)
2.  {
3.      double longitude = 0;
4.      double latitude = 0;
5.      double maxlat = 0, minlat = 0, maxlon = 0, minlon = 0;
6.      int i = 0;
7.      foreach (IPoint p in points)
8.      {
9.          latitude = p.Y;
```

```
        longitude = p.X;

        if (i == 0)
        {
            maxlat = latitude;
            minlat = latitude;
            maxlon = longitude;
            minlon = longitude;
        }
        else
        {
            if (maxlat < latitude)
                maxlat = latitude;
            if (minlat > latitude)
                minlat = latitude;
            if (maxlon < longitude)
                maxlon = longitude;
            if (minlon > longitude)
                minlon = longitude;
        }
        i++;
    }

    latitude = (maxlat + minlat) / 2;
    longitude = (maxlon + minlon) / 2;

    return new Point
    {
        X = longitude,
        Y = latitude
    };
}
```

Τέλος η αναφορά μας στην συγκεκριμένη λειτουργία του συστήματος θα κλείσει με ένα απόσπασμα του κώδικα που είναι υπεύθυνο για την ομαδοποίηση των events στα groups.

```
public HashSet<IPoint[]> ComputeGroups(IEnumerable<IPoint> points)
        {
            var AllPoints = points
                .Select(x => new AlgoPoint { Obj = x })
                .OrderBy(x => x.X)
                .ThenBy(x => x.Y)
                .ToArray();

            HashSet<AlgoGroup> AlgoGroupsFilled = new HashSet<AlgoGroup>();
            HashSet<AlgoGroup> AlgoGroups = new HashSet<AlgoGroup>();

            foreach (var point in AllPoints)
            {
                if (GetOrCreateCloserGroup(AlgoGroups, point, _radius, out AlgoGroup gro
up))
                    AlgoGroups.Add(group);

                group.Points.Add(point);

                if (group.Points.Count == _maxPoints)
                {
                    AlgoGroupsFilled.Add(group);
                    AlgoGroups.Remove(group);
                }
            }

```

```
            HashSet<AlgoGroup> AllGroups = new HashSet<AlgoGroup>(AlgoGroupsFilled.Union
(AlgoGroups));
            ProccessOrphanChildren(AllGroups);

            HashSet<IPoint[]> results = new HashSet<IPoint[]>();
            foreach (var group in AllGroups)
            {
                results.Add(group.Points.Select(x => x.Obj).ToArray());
            }

            return results;
        }
```

```
bool GetOrCreateCloserGroup(IEnumerable<AlgoGroup> algoGroups, AlgoPoint point, double r
adius, out AlgoGroup algoGroup)
        {
            foreach (var g in algoGroups)
            {
                if (Points.Distance(g.X, g.Y, point.X, point.Y) <= radius / 2)
                {
                    algoGroup = g;
                    return false;
                }
            }

            algoGroup = new AlgoGroup
            {
                Points = new HashSet<AlgoPoint>(),
                X = point.X,
                Y = point.Y
            };

            return true;
        }
```

Τέλος για τα events τα οποία δεν ανήκουν σε κάποιο group τα ορίζουμε ως Orphan και η κατηγοριοποίηση τους γίνεται με την χρήση της ακόλουθης μεθόδου.

```
void ProccessOrphanChildren(HashSet<AlgoGroup> algoGroups)
        {
            foreach (var openGroup in algoGroups.ToArray())
            {
                foreach (var orphanPoint in openGroup.Points.ToArray())
                {
                    if (!GetAverageOrCreateCloserGroup(algoGroups, openGroup.Points.Firs
t(), _radius, out AlgoGroup group, true))
                    {
                        group.Points.Add(orphanPoint);
                        openGroup.Points.Remove(orphanPoint);
                    }

                    if (openGroup.Points.Count == 0)
                        algoGroups.Remove(openGroup);
                }

            }
        }
```

3.5 Μικρές Αναφορές σε Κώδικα από όλες τις Τεχνολογίες που Χρησιμοποιήθηκαν για την Ανάπτυξη του Συστήματος

Αρχικά θα αναφερθούμε στο Style του συστήματος. Όλος ο σχεδιασμός έγινε με χρήση LESS. Είναι παρόμοια γλώσσα με τη CSS μόνο που σου παρέχει πολύ μεγαλύτερη ευκολία στην σύνταξη και στην ανάπτυξη του κώδικα καθώς σου επιτρέπει την δήλωση μεταβλητών και την χρήση συναρτήσεων όπου μπορείς στην συνέχεια να χρησιμοποιείς σε οποιοδήποτε σημείο του κώδικά σου όπως και σε οποιαδήποτε άλλη γλώσσα. Στην συνέχεια, αφού ολοκληρωθεί το αρχείο LESS παράγεται αυτόματα το αντίστοιχο CSS αρχείο το οποίο συνδέουμε στην κάθε εφαρμογή.

LESS αρχείο :

```
@fontface: 'Open Sans';
@fontfaceCondensed: 'Open Sans Condensed';

@blue_curious: #228bc5;
@cerise: #c622bf;
@green_forest: #23c639;
@mako: #404B4F;
@mercury: #e2e2e2;
@totemPole: #9d0505;

@main_width: 1150px;

body {
    font-family: @fontface !important;

    &.layout-boxed {
        background-image: url('/content/images/Fixtures/Backgrounds/site_bg.jpg');
    }
}
```

Παραγώμενο CSS αρχείο :

```
body {
  font-family: 'Open Sans' !important;
}
body.layout-boxed {
  background-image: url('/content/images/Fixtures/Backgrounds/site_bg.jpg');
}
.btn-dark {
  color: #fff;
  background-color: #222d32;
  border-color: #182023;
}
.btn-dark:focus,
.btn-dark.focus {
  color: #fff;
  background-color: #0c0f11;
  border-color: #000;
}
.btn-dark:hover {
  color: #fff;
  background-color: #2e3c42;
  border-color: #1e282c;
}
```

Στην συνέχεια θα δούμε δύο ειδών JavaScript αρχείων. Το πρώτο είδος αναφέρεται στα κλασικά αρχεία JavaScript σε συνδυασμό όμως με χρήση ενός Javascript library το KnockOut , ενώ το δεύτερο είδος αναφέρεται στα Resources που έχουμε στο site προκειμένου να υποστηρίζεται η πολυγλωσσία.

Κλασικά αρχεία JavaScript (για παράδειγμα αναφέρεται ο κώδικας του Login για την εφαρμογή των Χρηστών), ο συγκεκριμένος κώδικας είναι ακόμα υπό επεξεργασία, δεν έχει ολοκληρωθεί αυτή η λειτουργία ακόμη στην εφαρμογή :

```
cb.pages = cb.pages || {};
cb.pages.login = (function () {
    var _options;
    var _currentmodel;

    function getPageModelOptions() {
        return {};
    }

    function PageModel(options) {
        var self = this;
        ko.mapping.fromJS(options, getPageModelOptions(), self);
        cb.pages.base.buildbase(self, options);

        self.validateInputs = function () {
            if (self.username() == null || self.username().trim().length == 0) {
                self.allErrors(resx("validation.username_required"));
                return false;
            }

            if (self.password() == null || self.password().trim().length == 0) {
                self.allErrors(resx("validation.password_required"));
                return false;
            }

            return true;
        }

        self.toJS = function () {
            return ko.mapping.toJS(self, {
                'ignore': ["allErrors", "status", "validateInputs", "toJS"],
                'include': ["password", "username"]
            });
        }

        self.submit = function () {
            self.allErrors("");

            if (!self.validateInputs())
                return;

            cb.managers.popup.freezeScreen();

            $.post("/login", self.toJS())
                .always(cb.managers.popup.unfreezeScreen)
                .fail(requestfailedhandler)
                .done(function (response) {
                    if (response.data.status == cb.constants.status.FAILED)
                        self.allErrors(response.data.errors);
                    else
                        window.location = response.data.redirect;
                })

        }

        self.afterBinding = function () {
```

```
            $('input').iCheck({
                checkboxClass: 'icheckbox_square-blue',
                radioClass: 'iradio_square-blue',
                increaseArea: '20%' // optional
            });
        }

        return self;
    }


    function _init(options) {
        _options = options;

        _currentmodel = new PageModel(options);

        _currentmodel.bindModel("login");


        //cb.managers.popup.init({ selector: $("#popup_content") });
    }

    return {
        init: _init,
        model: _currentmodel,
    };
})();
```

Στη συνέχεια θα δείτε ένα μέρος από τα JavaScript Respurces (ελληνικά):

```
eventcaptions: {
        Raceroad: "Αγώνας δρόμου",
        Racetrail: "Αγώνας βουνού",
        Racetrailcity: "Αγώνας πόλης",
    },

    filters: {
        Searchfilters: 'Φίλτρα αναζήτησης',
        Sports: 'Αθλητικά',
        Racing: 'Αγώνες',
        Raceroad: 'Αγώνες δρόμου',
        Racetrail: 'Αγώνες βουνου',
        Racetrailcity: 'Αγώνες πόλης',
    },

    images: {
        Enterimagedescription: "Enter image's description",
        ImageUploadingCompleted: "Image upload was completed sucessfully!",
        YouShouldChooseAnImage: "You have to choose an image first!"
    },

    languages: {
        "en-US_short": "En",
        "en-US_long": "English",
        "el-GR_short": "Gr",
        "el-GR_long": "Greek"
    },
```

Και τέλος θα δείτε ένα μέρος από τα JavaScript Respurces (αγγλικά):

```
eventcaptions: {
        Raceroad: "Road race",
```

```
        Racetrail: "Trail race",
        Racetrailcity: "City trail race",
    },

    filters: {
        Searchfilters: 'Search filters',
        Sports: 'Sports',
        Racing: 'Racing',
        Raceroad: 'Road race',
        Racetrail: 'Trail race',
        Racetrailcity: 'City trail race',
    },

    images: {
        Enterimagedescription: "Enter image's description",
        ImageUploadingCompleted: "Image upload was completed sucessfully!",
        YouShouldChooseAnImage: "You have to choose an image first!"
    },

    languages: {
        "en-US_short": "En",
        "en-US_long": "English",
        "el-GR_short": "Gr",
        "el-GR_long": "Greek"
    },
```

Τέλος θα δούμε κάποια κομμάτια από HTML τα οποία συνδέονται και με τα Resources που αναφέραμε προηγουμένως. Αρχικά θα δούμε ένα Template που χρησιμοποιούμε και στη συνέχεια θα δούμε την κύρια σελίδα, ένα ουσιαστικά από τα Views που αναφέραμε στο πρώτο κεφάλαιο στα μέρη του ASP.NET MVC (Model-View-Controller).

Template :

```
<script type="text/html" id="tmpl_pointviews_tooltips_minimal_race_road">
    <div class="race_road">
        <div class="bg_logo" data-bind="backgroundimage: { imageurl: toImageUrl(imageLogo.fileName()) }"></div>
        <div class="section top">
            <span data-bind="text: title"></span>
            <span data-bind="text: distance"></span>
            <span data-bind="resx: 'eventcaptions.Raceroad'"></span>
        </div>
        <div class="section main">
            <div>
                <span data-bind="resx: 'sports.Startdate'"></span><span data-bind="text: startDate"></span>
            </div>
            <div>
                <span data-bind="resx: 'sports.Starttime'"></span><span data-bind="text: startTime"></span>
            </div>
            <div data-bind="visible: websiteLink() != null">
                <a href="" target="_blank" data-bind="attr: { 'href': websiteLink }, resx: 'shared.Websitelink'"></a>
            </div>
            <div>
                <button class="btn btn-default btn-flat btn-xs" data-bind="click: show, resx: 'shared.More'"></button>
                <i class="favorites_icon fa" data-bind="click:toggleFavorite,css:{'fa-heart':isFavorite(),'fa-heart-o':!isFavorite()}"></i>
            </div>
        </div>
    </div>
```

```
25. </script>
```

View (η κύρια σελίδα (home page) της εφαρμογής των χρηστών) :

```
1.  @model HomePageModel
2.  @{
3.      ViewBag.bodycss = "sidebar-mini layout-boxed  skin-black";
4.      ViewBag.Title = "SportsCalendar";
5.  }
6.
7.  @section metasection{
8.      @if (Model.FacebookMetadataModel != null)
9.      {
10.         <meta property="og:url" content="@Model.FacebookMetadataModel.Url" />
11.         <meta property="og:type" content="@Model.FacebookMetadataModel.Type" />
12.         <meta property="og:title" content="@Model.FacebookMetadataModel.Title" />
13.         <meta property="og:description" content="@Model.FacebookMetadataModel.Descri
    ption" />
14.         <meta property="og:image" content="@Model.FacebookMetadataModel.Image" />
15.     }
16. }
17.
18. @section Scripts {
19.     <script type="text/javascript">
20.         cb.pages.setCurrent(cb.pages.home, @Html.Raw(JsonModelConverter.Serialize(Mo
    del)));
21.     </script>
22. }
23.
24. <div class="banner home left">@Cms.Get("BANNERS_MAIN_HOME_LEFT").Render()</div>
25. <div class="banner home right">@Cms.Get("BANNERS_MAIN_HOME_RIGHT").Render()</div>
26. <div id="home" class="wrapper">
27.     <!-- ko with: headerModel -->
28.     <!-- ko template: template -->
29.     <!-- /ko -->
30.     <!-- /ko -->
31.
32.     <!-- ko with: leftMenuModel-->
33.     <!-- ko template: { name: template, afterRender: afterBinding } -->
34.     <!-- /ko -->
35.     <!-- /ko -->
36.
37.     <div id="main" class="content-wrapper home">
38.         <!-- ko template: pageTemplate -->
39.         <!-- /ko -->
40.     </div>
41.
42.     <!-- ko with: footerModel -->
43.     <!-- ko template: template -->
44.     <!-- /ko -->
45.     <!-- /ko -->
46. </div>
47.
48. <script type="text/html" id="tmpl_banners_home_top_main">
49.     @Cms.Get("BANNERS_MAIN_TOP").Render()
50. </script>
51.
52. <script type="text/html" id="tmpl_banners_home_left_menu_position_1">
53.     @Cms.Get("BANNERS_LEFTMENU_POSITION_1").Render()
54. </script>
```

View (σελίδα login από την εφαρμογή των χρηστών), κώδικας υπό επεξεργασία :

```
@model LoginModel
@{ ViewBag.bodycss = "hold-transition login-page"; }

@section Scripts {
    <script type="text/javascript">
        cb.pages.setCurrent(cb.pages.login, @Html.Raw(JsonModelConverter.Serialize(Model)));
    </script>
}

<div id="login" class="login-box">
    <div class="login-logo">
        <a href="/"><b>Sports</b>CALENDAR</a>
    </div>
    <div class="login-box-body">
        <p class="login-box-msg" data-bind="resx: 'login.promo'"></p>
        <p class="text-danger text-center" data-bind="text: allErrors"></p>
        <form>
            <div class="form-group has-feedback">
                <input type="email" class="form-control" data-bind="value: username, attr: { 'placeholder' : resx('shared.Email') }">
                <span class="glyphicon glyphicon-envelope form-control-feedback"></span>
            </div>
            <div class="form-group has-feedback">
                <input type="password" class="form-control" data-bind="value: password, attr: { 'placeholder' : resx('shared.Password') }">
                <span class="glyphicon glyphicon-lock form-control-feedback"></span>
            </div>
            <div class="row">
                <div class="col-xs-8">
                    @*<div class="checkbox icheck">
                        <label>
                            <input type="checkbox"> Remember Me
                        </label>
                    </div>*@
                </div>
                <div class="col-xs-4">
                    <button type="submit" class="btn btn-primary btn-block btn-flat" data-bind="click: submit, resx: 'login.submit'"></button>
                </div>
            </div>
        </form>
        @*<div class="social-auth-links text-center">
            <p>- OR -</p>
            <a href="#" class="btn btn-block btn-social btn-facebook btn-flat">
                <i class="fa fa-facebook"></i> Sign in using
                Facebook
            </a>
            <a href="#" class="btn btn-block btn-social btn-google btn-flat">
                <i class="fa fa-google-plus"></i> Sign in using
                Google+
            </a>
        </div>*@
        <a href="#" data-bind="resx: 'login.forgotpassword'"></a><br>
        <a href="register.html" class="text-center" data-bind="resx: 'login.register'"></a>
    </div>
</div>
```

Βιβλιογραφία

1. https://el.wikipedia.org/wiki/Model-view-controller
2. http://webapptester.com/mvc-framework-first-impression/
3. Οι ενότητες 1.3.2 & 1.3.3 είναι αποσπάσματα από το βιβλίο «Διαδικτυακός Προγραμματισμός C#» του Ιωάννη-Χρήστου Παναγιωτόπουλου, Καθηγητής Τμήματος Πληροφορικής Πανεπιστημίου Πειραιώς, Εκδόσεις Αθ. Σταμούλης.
4. http://whatis.techtarget.com/definition/model-view-controller-MVC
5. https://msdn.microsoft.com/en-us/library/dd381412%28v=vs.108%29.aspx?f=255&MSPPError=-2147217396
6. http://tostring.it/2014/06/30/top-must-know-frameworks-for-net-web-developers/
7. https://msdn.microsoft.com/en-us/library/aa479002.aspx?f=255&MSPPError=-2147217396
8. https://en.wikipedia.org/wiki/.NET_Framework
9. https://msdn.microsoft.com/en-us/library/4w3ex9c2.aspx?f=255&MSPPError=-2147217396
10. https://en.wikipedia.org/wiki/Common_Language_Runtime
11. http://www.dga.gr/web/publications/files/csharp.pdf
12. https://en.wikipedia.org/wiki/Map_matching

Ευρετήριο Πινάκων

Ευρετήριο Εικόνων