ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΗΛΕΚΤΡΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΩΝ ΔΙΑΤΑΞΕΩΝ ΚΑΙ
ΣΥΣΤΗΜΑΤΩΝ ΑΠΟΦΑΣΕΩΝ

Ανάπτυξη, Συντήρηση και Βελτιστοποίηση Πληροφοριακού Συστήματος Καταγραφής Διαδικασιών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΓΚΑΖΟΥΝΗ ΑΙΚΑΤΕΡΙΝΗ

Επιβλέπων : Δημήτρης Ασκούνης
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2016

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΗΛΕΚΤΡΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΩΝ ΔΙΑΤΑΞΕΩΝ ΚΑΙ
ΣΥΣΤΗΜΑΤΩΝ ΑΠΟΦΑΣΕΩΝ

Ανάπτυξη, Συντήρηση και Βελτιστοποίηση Πληροφοριακού Συστήματος Καταγραφής Διαδικασιών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΓΚΑΖΟΥΝΗ ΑΙΚΑΤΕΡΙΝΗ

Επιβλέπων : Δημήτρης Ασκούνης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 13η Ιουλίου 2016

Ασκούνης Δ.	Ψαρράς Ι.	Δούκας Χ.
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Επ. Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2016

.....
ΓΚΑΖΟΥΝΗ ΑΙΚΑΤΕΡΙΝΗ
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Η εκθετική ανάπτυξη του Παγκόσμιου Ιστού και η συνεχής χρήση του σε διάφορους τομείς της καθημερινότητας, έχει οδηγήσει στην ανάπτυξη μίας νέας γενιάς εφαρμογών, οι οποίες χαρακτηρίζονται πλέον από μεγάλο βαθμό πολυπλοκότητας. Η ανάπτυξη τέτοιων εφαρμογών είναι στην ουσία ένας συνδυασμός των παραδοσιακών Πληροφοριακών Συστημάτων με εφαρμογές Υπερμέσων (Hypermedia). Αυτός ο συνδυασμός θέτει νέες προκλήσεις στις υπάρχουσες προσεγγίσεις σχεδιασμού και παραγωγής λογισμικού. Ένα παράδειγμα μιας αντίστοιχης εφαρμογής είναι και το Πληροφοριακό Σύστημα Διαχείρισης Χρηματοδοτήσεων. Το Πληροφοριακό Σύστημα Διαχείρισης Χρηματοδοτήσεων υλοποιήθηκε στα πλαίσια του Επιχειρησιακού Προγράμματος "Ψηφιακή Σύγκλιση" για να καλύψει τις ανάγκες για διαχείριση της υποβολής χρηματοδοτήσεων αλλά και της υποβολής πιστώσεων.

Ο σκοπός της παρούσης διπλωματικής είναι πολλαπλός. Έπειτα από αρκετά χρόνια λειτουργίας του πληροφοριακού συστήματος η ανάγκη για συντήρηση είναι πλέον έκδηλη, καθώς αρκετές αστοχίες και λάθη λογισμικού είναι πλέον ορατές και επηρεάζουν την ομαλή λειτουργία του συστήματος. Επιπλέον, λαμβάνοντας υπόψιν τις καινούργιες τεχνολογίες που έχουν αναπτυχθεί κατά την πάροδο των χρόνων, η απόδοση του συστήματος δύναται να βελτιωθεί και να μην αποτελεί πλέον τροχοπέδη για την καθημερινή χρήση του συστήματος. Τέλος, η ανάγκη εκσυγχρονισμού και προσαρμογής του πληροφοριακού συστήματος στα σημερινά δεδομένα, τόσο σε επίπεδο προγραμματισμού αλλά και επίπεδο λειτουργίας, επιτάσσει την ανάπτυξη νέων δυνατοτήτων αλλά και νέων τεχνικών γραφής κώδικα.

Λέξεις Κλειδιά:
Πληροφοριακό Σύστημα, MVC Project, C#, Έλεγχος Database, Software Bug, Code Refactoring, Βελτίωση Απόδοσης

Abstract

The exponential growth of the Web and the continuous dispersion in various areas of everyday life, has fueled the development of a new generation of applications, which are now characterized by a high degree of complexity. The development of such applications is actually a hybrid that combines traditional Information Systems with Applications Hypermedia (Hypermedia). This combination poses new challenges to existing design and software development approaches. An example of a corresponding application is the Funding Information Management System. The Funding Information Management System was implemented under the Business Programme "Digital Plan" to meet the needs for management reporting funding and submitting credit.

The purpose of this project is multiple. After several years of information system operation, the need for maintenance is most evident, as several failures and software errors are more visible and affect the smooth operation of the system. Moreover, taking into account the new technologies that have been developed over the years, the system performance can be improved and no longer constitutes a hindrance to the daily use of the system. Finally, the need to update and adapt the information system up to date, both in programming level and operating level, requires the development of new possibilities and new techniques for writing code.

Keywords:

Information Management System, MVC Project, C#, Database Check, Software Bug, Code Refactoring, Performance Improvement

Ευχαριστίες

Η ολοκλήρωση της παρούσας διπλωματικής εργασίας υλοποιήθηκε με την υποστήριξη ενός αριθμού ανθρώπων, στους οποίους θα ήθελα να εκφράσω τις θερμότερες ευχαριστίες μου.

Αρχικά, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή ΕΜΠ κύριο Δημήτρη Ασκούνη, ο οποίος μου ανέθεσε αυτή την ενδιαφέρουσα εργασία.

Ιδιαίτερες ευχαριστίες θα ήθελα να απευθύνω στον υποψήφιο Διδάκτορα του Ε.Μ.Π., Ρωμανό Τσουροπλή, για την καθοδήγησή του και την άριστη συνεργασία σε οποιοδήποτε πρόβλημα αντιμετώπισα κατά τη διάρκεια εκπόνησης της εργασίας.

Τέλος, οφείλω ένα μεγάλο ευχαριστώ στην οικογένειά μου και στα αγαπημένα μου πρόσωπα για όλη τη βοήθεια, υποστήριξη, εμπιστοσύνη και αντοχή που έδειξαν κατά τη διάρκεια όλων των σπουδών μου.

Περιεχόμενα

1 Εισαγωγή 12

1.1 Γενικά – Το πρόβλημα & η ιδέα 13

1.2 Οργάνωση Εργασίας 14

2 Πληροφοριακά Συστήματα 16

2.1 Εισαγωγή 17

2.2 Το Πληροφοριακό Σύστημα 18

2.2.1 Ολοκληρωμένος Ορισμός Πληροφοριακού Συστήματος 18

2.2.2 Δομή ενός Πληροφοριακού Συστήματος 19

2.2.3 Ο Κύκλος ζωής ενός Πληροφοριακού Συστήματος 20

2.3 Πληροφοριακά Συστήματα σε μια Επιχείρηση- Οργανισμό 24

2.3.1 Βασικοί τύποι συστημάτων 24

2.3.2 Πλεονεκτήματα Χρήσης Πληροφοριακών Συστημάτων 26

2.3.3 Μειονεκτήματα Χρήσης Πληροφοριακών Συστημάτων 27

3 Σύστημα Διαχείρισης Χρηματοδοτήσεων 28

3.1 Το Πληροφοριακό Σύστημα Διαχείρισης Χρηματοδοτήσεων 29

3.2 Τεχνολογίες που χρησιμοποιήθηκαν κατά την υλοποίηση του 31

3.2.1 Εισαγωγή 31

3.2.2 Περιβάλλον Ανάπτυξης του Πληροφοριακού Συστήματος 31

3.2.2.1. Microsoft Visual Studio 31

3.2.2.2. .NET Framework 31

3.2.2.3. ASP.NET MVC 36

3.2.2.4. Η γλώσσα προγραμματισμού C# 39

3.2.2.5. AJAX και JQuery 41

3.2.2.6. CSS 42

3.2.2.7. JavaScript 42

3.2.2.8. Entity Framework 43

3.2.2.9. SQL Server 44

3.2.2.10. Σχεσιακό Διάγραμμα της Βάσης Δεδομένων του Πληροφοριακού Συστήματος *45*

4 Βελτιώσεις-Αλλαγές-Αποτελέσματα 47

4.1 Σκοπός των αλλαγών στο λογισμικό του Πληροφοριακού Συστήματος 48

4.2 Βάση Δεδομένων (Database) 49

4.2.1 Προβλήματα κακής σχεδίασης μιας Βάσης Δεδομένων 49

4.2.2 Έλεγχος χρήσης πινάκων σε Database Π.Σ.Δ.Χ 50

4.3 Διόρθωση λαθών (bugs) 57

4.3.1 Ανάθεση από Στέλεχος Β (μη εξουσιοδοτημένο) 57

4.3.2 Ανάθεση από Στέλεχος Γ (μη εξουσιοδοτημένο) 59

4.3.3 Ρόλοι 60

4.3.4 Αρχικοποίηση προέγκρισης 62

4.4 Code refactoring 64

4.4.1 Τι είναι το code refactoring 64

4.4.2 Τεχνικές για code refactoring 64

4.4.2.1. Τεχνικές που βασίζονται στην αρχή της αφαίρεσης (abstraction) *64*

4.4.2.2. Τεχνικές για το καταμερισμό του πηγαίου κώδικα σε πιο λογικά μέρη *65*

4.4.2.3. Τεχνικές για τη βελτίωση των ονομάτων και τη θέση του κώδικα *65*

4.5 CSS αλλαγές-βελτιώσεις 67

4.6 Βελτίωση απόδοσης 69

5 Επίλογος 71

5.1 Σύνοψη και Συμπεράσματα 72

5.2 Μελλοντικές Αλλαγές – Νέες Τεχνολογίες 73

5.2.1 Angular JS 73

5.2.2 PhoneGap 74

6 Βιβλιογραφία 75

Παράρτημα 78

Παράρτημα Α: SQL Query για έλεγχο μεγέθους βάσης 79

Παράρτημα Β: SQL Query για έλεγχο χρήσης βάσης 80

Παράρτημα Γ: Πίνακας Εικόνων 81

1 Εισαγωγή

1.1 Γενικά – Το πρόβλημα & η ιδέα

Η αυτοματοποίηση των διαδικασιών τόσο στη δημόσια διοίκηση αλλά και στο σύνολο των επιχειρήσεων εν έτει 2016 είναι ένα σύνηθες γεγονός. Γραφειοκρατικές διαδικασίες που απαιτούσαν πολύ χρόνο για την διεκπεραίωσή τους, με την εξέλιξη της τεχνολογίας και την εισαγωγή της Πληροφορικής στην καθημερινότητά μας, έχουν πλέον αυτοματοποιηθεί και δρομολογούνται σε λιγότερο χρονικό διάστημα, με μικρότερο κόστος, αλλά και με μεγαλύτερη διαφάνεια. Η αυτοματοποίηση των διαδικασιών επιτυγχάνεται κυρίως με τη χρήση Πληροφοριακών Συστημάτων.

Όπως θα αναφερθεί αναλυτικότερα και σε επόμενο κεφάλαιο, η ανάπτυξη του Πληροφοριακού Συστήματος Διαχείρισης Χρηματοδοτήσεων, έγινε στα πλαίσια του Επιχειρησιακού Προγράμματος «Ψηφιακή Σύγκλιση» και για να αυτοματοποιήσει την διαδικασία υποβολής αιτήσεων για χρηματοδότηση αλλά και υποβολής ετήσιων πιστώσεων. Το πληροφοριακό σύστημα κάλυψε σε μεγάλο βαθμό τις απαιτήσεις που προέκυψαν και συνεχίζει να χρησιμοποιείται ακόμη και σήμερα.

Ωστόσο, με μια απλή χρήση του πληροφοριακού συστήματος θα διαπιστώσει κανείς ότι δεν ανταποκρίνεται πλέον στα δεδομένα του σήμερα. Σφάλματα λογισμικού, προχειρότητα στη σχεδίαση, καθυστέρηση στην απόκριση της ιστοσελίδας, αλλά και τεχνολογίες που πλέον θεωρούνται ξεπερασμένες, συνθέτουν την ανάγκη αναπροσαρμογής και επανασχεδίασης του πληροφοριακού συστήματος.

Σε μια πρώτη προσέγγιση, θα πραγματοποιηθεί μια προσπάθεια επιδιόρθωσης των λαθών του κώδικα που επηρεάζουν πλέον και την λειτουργία του πληροφοριακού συστήματος, ένας έλεγχος της βάσης δεδομένων ώστε να διαπιστωθεί ποια στοιχεία της χρησιμοποιούνται και εφόσον αποκατασταθεί πλήρως η λειτουργικότητα του συστήματος, θα εξεταστεί η αύξηση της απόδοσης του.

Τέλος, θα εξεταστούν πιθανές λειτουργίες που μπορούν να αναπτυχθούν στο μέλλον, αλλά και μελλοντικές τεχνολογίες ώστε να καταστήσουν το Πληροφοριακό Σύστημα Διαχείρισης Χρηματοδοτήσεων ένα σύγχρονο και ευέλικτο εργαλείο της Δημόσιας Διοίκησης.

1.2 Οργάνωση Εργασίας

Η παρούσα εργασία αποτελείται, πλην του παρόντος, από τα ακόλουθα μέρη- κεφάλαια :

→ Στο δεύτερο κεφάλαιο γίνεται μια θεωρητική ανάλυση των πληροφοριακών συστημάτων. Παρουσιάζονται εν συντομία τα είδη των πληροφοριακών συστημάτων, οι εμπλεκόμενες μονάδες, ο σκοπός που εξυπηρετούνε, αλλά και τα πλεονεκτήματα και μειονεκτήματα της χρήσης τους.

→ Το τρίτο κεφάλαιο αποτελεί μια περιγραφή του Πληροφοριακού Συστήματος Διαχείρισης Χρηματοδοτήσεων και των αρμοδιοτήτων του αλλά και των δυνατοτήτων του. Στο πλαίσιο αυτό παρουσιάζεται και οι τεχνολογίες που χρησιμοποιήθηκαν κατά την υλοποίησή του.

→ Το τέταρτο κεφάλαιο εισάγει τον αναγνώστη στα προβλήματα που εντοπίστηκαν στο Πληροφοριακό Σύστημα Διαχείρισης Χρηματοδοτήσεων και τι αλλαγές πραγματοποιήθηκαν ώστε να διορθωθούν.

→ Στο πέμπτο κεφάλαιο αναλύονται τα αποτελέσματα των αλλαγών που πραγματοποιήθηκαν στο τέταρτο κεφάλαιο αλλά και πιθανές αλλαγές που θα μπορούσαν να πραγματοποιηθούν μελλοντικά.

→ Τέλος στο παράρτημα, που έπεται της βιβλιογραφίας παρουσιάζονται τα sql queries που χρησιμοποιήθηκαν κατά τον έλεγχο της βάσης δεδομένων.

2 Πληροφοριακά Συστήματα

2.1 Εισαγωγή

Στον σύγχρονο κόσμο των επιχειρήσεων τα πληροφοριακά συστήματα (Information systems) κατέχουν καθοριστικό ρόλο στην επιβίωση της επιχείρησης σ ένα περιβάλλον όπου ο ανταγωνισμός αυξάνεται καθημερινά. Μπορούμε να πούμε ότι αποτελούν τη σπονδυλική στήλη της Διοίκησης κάθε σύγχρονης επιχείρησης καθώς αποτελεί το συστατικό που συνδέει το φυσικό σύστημα παραγωγής με το σύστημα λήψης αποφάσεων. Βασική λειτουργία του είναι ο μετασχηματισμός των δεδομένων που υπάρχουν στο φυσικό σύστημα παραγωγής σε πληροφορίες (δεδομένα) που απαιτούν οι δραστηριότητες του συστήματος λήψης αποφάσεων. Όμως και αντίστροφα, διαβιβάζει (τις οδηγίες) δεδομένα που παράγει το σύστημα διοίκησης σε κατάλληλες πληροφορίες (δεδομένα) για το φυσικό σύστημα παραγωγής. Εκτός από τις παραπάνω βασικές λειτουργίες ένα ολοκληρωμένο Π.Σ. πρέπει να προσφέρει και επιπλέον δυνατότητες για:

:

- Την υποστήριξη των διοικητικών στελεχών όλων των επιπέδων στη λήψη έγκαιρων και σωστών αποφάσεων προς όφελος της εταιρείας.
- Την υποστήριξη της διαχείρισης της καθημερινής λειτουργίας της εταιρείας.
- Τον έλεγχο της λειτουργίας της επιχείρησης.
- Την υποστήριξη στον προγραμματισμό και τη δημιουργία της στρατηγικής ανάπτυξης της επιχείρησης ή του οργανισμού.
- Τη συνεισφορά στη δημιουργία αλλαγών ώστε η επιχείρηση ή ο οργανισμός να είναι σε θέση να προσαρμόζεται συνεχώς στο περιβάλλον του - ένα ολοκληρωμένο Π.Σ. πρέπει να είναι ευέλικτο και προσαρμόσιμο ώστε να ανταποκρίνεται στις αλλαγές και τις διαφορετικές απαιτήσεις διαφόρων ομάδων χρηστών.

Οι λειτουργίες αυτές πετυχαίνονται με την ολοκληρωμένη επεξεργασία των δεδομένων της επιχείρησης μέσα από την οποία παράγονται οι απαραίτητες πληροφορίες που καλύπτουν όλους τους τομείς της εταιρείας και οδηγούν στη λήψη σωστών αποφάσεων.

Τι ορίζουμε ωστόσο ως πληροφοριακό σύστημα;

2.2 Το Πληροφοριακό Σύστημα

2.2.1 Ολοκληρωμένος Ορισμός Πληροφοριακού Συστήματος

Ένα πληροφοριακό σύστημα είναι ένα σύνολο από διαδικασίες που συλλέγει, επεξεργάζεται, αποθηκεύει και μεταδίδει πληροφορίες με σκοπό την υποστήριξη λήψης αποφάσεων και του ελέγχου. Στις περισσότερες περιπτώσεις, τα συστήματα πληροφοριών είναι επίσημα συστήματα που βασίζονται σε υπολογιστικά συστήματα τα όποια και επιτελούν σημαντικό ρόλο στους οργανισμούς. Παρόλο που τα συστήματα πληροφοριών είναι βασισμένα σε υπολογιστικά συστήματα είναι σημαντικό να σημειώσουμε ότι κάθε υπολογιστικό σύστημα ή λογισμικό δεν είναι απαραίτητα ένα σύστημα πληροφοριών . Οι ηλεκτρονικοί υπολογιστές και τα σχετικά προγράμματα λογισμικού αποτελούν το τεχνικό υπόβαθρο, τα εργαλεία και το υλικό για τα σύγχρονα συστήματα πληροφοριών. Για την κατανόηση ωστόσο των συστημάτων πληροφοριών, απαιτείται κανείς να κατανοήσει τα προβλήματα τα οποία σχεδιάστηκαν να επιλύσουν, τις αρχιτεκτονικές και σχεδιαστικές λύσεις τους, καθώς και τις διαδικασίες του οργανισμού που οδηγούν σε αυτές τις λύσεις. Είναι σημαντικό να παρατηρηθεί πως το κάθε σύστημα πληροφοριών είναι μοναδικά σχεδιασμένο σύμφωνα με τις συγκεκριμένες απαιτήσεις του οργανισμού για τον οποίο αναπτύχθηκε. Κάθε οργανισμός ή επιχείρηση, προκειμένου να ελέγξει και να συντονίσει τον όγκο δεδομένων που συσσωρεύεται καθημερινά χρειάζεται κάποιο σύστημα, το οποίο να ανταποκρίνεται στα προβλήματα που αντιμετωπίζει άμεσα και αποτελεσματικά. Έτσι, με το απαραίτητο προσωπικό που θα είναι σωστά εκπαιδευμένο και το κατάλληλο πληροφοριακό σύστημα η διαχείριση και η επεξεργασία των πληροφοριών γίνεται απλούστερη και αποδοτική.

Ως σύστημα θα μπορούσαμε να ορίσουμε ένα σύνολο από συνιστώσες που αλληλεπιδρούν μεταξύ τους για να επιτύχουν κάποιο σκοπό. Οι συνιστώσες αυτές μπορεί να είναι:

- Όντα,
- Υλικά,
- Ιδέες,
- Αξίες.

Τα διάφορα μέρη ενός συστήματος είναι με τη σειρά τους συστήματα σε μικρότερη κλίμακα τα οποία αποτελούν υποσυστήματα του αρχικού συστήματος.

Επομένως κάθε σύστημα είναι υπερσύστημα κάποιων συστημάτων, αλλά αποτελεί παράλληλα και υποσύστημα κάποιου άλλου συστήματος. Όλα τα συστήματα περικλείονται από το περιβάλλον τους, δηλαδή με κάθε οντότητα που βρίσκεται έξω από τα όρια του συστήματος [1]

Σε κάθε σύστημα εισέρχονται δεδομένα από το περιβάλλον του, μετατρέπονται σε πληροφορίες και τέλος εξάγονται προς το περιβάλλον. Άρα κάθε σύστημα έχει μια Είσοδο (Input) , Επεξεργασία (Processing), και μια Έξοδο (Output). Ως δεδομένα (data) μπορούμε να ορίσουμε τα γεγονότα ή τις τιμές κάποιων χαρακτηριστικών που ανήκουν σε οντότητες και για να είναι χρήσιμα πρέπει να έχουν ακρίβεια, πληρότητα, σχετικότητα και διαθεσιμότητα.

Πληροφοριακό Σύστημα λοιπόν, είναι ένα σύνολο αλληλοσυνδεόμενων μερών που συνεργάζονται για τη συλλογή, επεξεργασία, αποθήκευση και διάχυση πληροφοριών με σκοπό την υποστήριξη της λήψης αποφάσεων, του συντονισμού, του ελέγχου και της ανάλυσης δεδομένων του δημοσίου τομέα ή ενός οργανισμού. [2]

2.2.2 Δομή ενός Πληροφοριακού Συστήματος

Κάθε Πληροφοριακό σύστημα αναλύεται στις παρακάτω συνιστώσες:

Α)<u>Άνθρωποι</u>

Υπάρχουν τρείς κατηγορίες σε αυτή τη συνιστώσα:

- Χρήστες(end users, user managers).
- Χρήστες που είτε εισάγουν στοιχεία στο σύστημα είτε συντηρούν το λογισμικό/ υλικό.
- Δημιουργοί(προγραμματιστές, εκπαιδευτές, αναλυτές, σχεδιαστές Β.Δ., ειδικοί δικτύων, project managers κ.λ.π)

Β)<u>Υλικό (Hardware)</u>

Οι προδιαγραφές υλικών και ο εξοπλισμός παίζουν πολύ σημαντικό ρόλο στη σύνθεση ενός Πληροφοριακού Συστήματος. Με τον όρο Υλικό αναφερόμαστε σε όλες τις συσκευές στις οποίες εκτελείται το πληροφοριακό σύστημα (π.χ. υπολογιστές, μονάδες αποθήκευσης πληροφορίας, δίκτυα κ.λ.π.)

Γ)Διαδικασίες (Procedures)

Αφορούν οδηγίες για τους εμπλεκόμενους στο σύστημα και διακρίνονται σε:

- Διαδικασίες για χρήστες(Εισαγωγή Δεδομένων)

- Διαδικασίες για χειριστές (Δημιουργία αντιγράφων ασφαλείας, Ανάκτηση Δεδομένων, Υπολογισμός στατιστικών στοιχείων, Κατασκευή γραφημάτων για απεικόνιση αποτελεσμάτων κ.λ.π)

Δ)Λογισμικό (Software)

Υπάρχουν διάφορες μορφές λογισμικού σε ένα οργανισμό. Πέρα από το λογισμικό που αφορά το πληροφοριακό σύστημα, υπάρχει συνήθως και λογισμικό για την κοστολόγηση, μισθοδοσία κλπ, αλλά και λογισμικό που διευκολύνει το χρήστη να αναπτύξει δικές του εφαρμογές.

Ε) Δεδομένα(Data)

Τα δεδομένα που είναι απαραίτητα για την κατασκευή ενός Πληροφοριακού Συστήματος είναι τα παρακάτω:

- Εικόνα
- Ήχος
- Κείμενο
- Σύμβολα

Σκοπός οποιουδήποτε πληροφοριακού συστήματος είναι να:

- Σχεδιάζει
- Ελέγχει
- Συντονίζει
- Διεκπεραιώνει

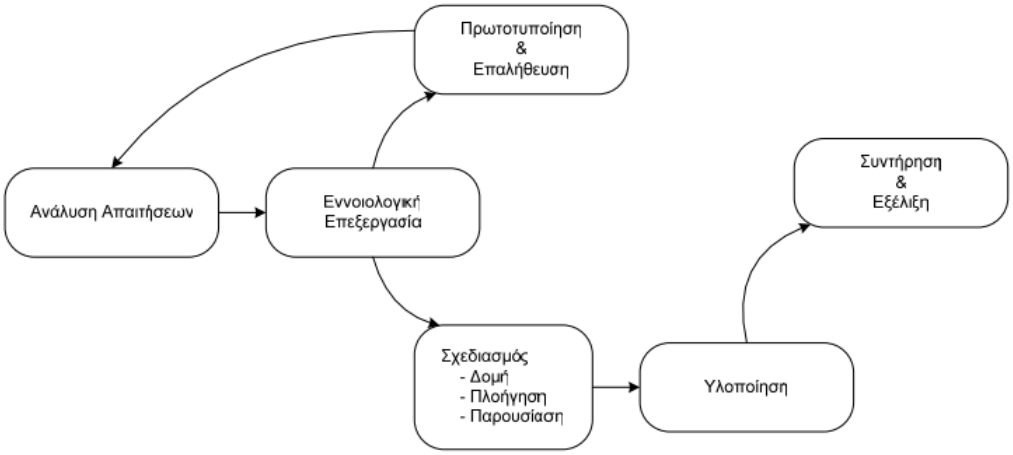
  Τις λειτουργίες ενός οργανισμού.

2.2.3 Ο Κύκλος ζωής ενός Πληροφοριακού Συστήματος

Παρόλο που μέχρι σήμερα δεν υπάρχει ομοφωνία σε σχέση με ένα γενικό μοντέλο του κύκλου ζωής (lifecycle) ενός Πληροφοριακού Συστήματος Παγκόσμιου Ιστού , ένα σχήμα των τυπικών δραστηριοτήτων οι οποίες εμπλέκονται στην κατασκευή μιας εφαρμογής του μπορεί να προκύψει από τη σύνθεση των μοντέλων κύκλων ζωής παραδοσιακών Πληροφοριακών συστημάτων και των προτάσεων για δομημένο σχεδιασμό Υπερμέσων [3].Στο Σχήμα 1 παρουσιάζεται το μοντέλο του κύκλου ζωής ενός Πληροφοριακού Συστήματος με βάση τα παραπάνω, όπως προτείνεται από τον (Fraternali, 1999). Με βάση το σχήμα αυτό, ο κύκλος ζωής ενός Πληροφοριακού Συστήματος αποτελείται από τα ακόλουθα στάδια:

- ***Ανάλυση Απαιτήσεων:*** Καθορίζεται ο στόχος του πληροφοριακού συστήματος, μέσω της αναγνώρισης των υποψήφιων χρηστών και του ορισμού της φύσης της βάσης πληροφοριών. Επιπρόσθετα της συνηθισμένης συλλογής των απαιτήσεων και των διαδικασιών αποτίμησης εφικτότητας, τα Πληροφοριακά Συστήματα Παγκόσμιου Ιστού που είναι σχεδιασμένες για καθολική πρόσβαση από μέρους των χρηστών, απαιτούν ιδιαίτερη προσοχή στον καθορισμό των απαιτήσεων αλληλεπίδρασης ανθρώπου-υπολογιστή, έτσι ώστε να καθοριστεί ο καταλληλότερος τύπος αλληλεπίδρασης για κάθε κατηγορία χρηστών και για κάθε τύπο συσκευής πρόσβασης.

- ***Εννοιολογική Επεξεργασία (Conceptualization):*** Το πληροφοριακό σύστημα αναπαρίσταται μέσα από ένα σύνολο από αφηρημένα μοντέλα, τα οποία επικοινωνούν τα κύρια συστατικά της συγκεκριμένης λύσης. [4]



Εικόνα 1 : Φάσεις ενός Πληροφοριακού Συστήματος

- ***Πρωτοτυποποίηση και Επαλήθευση*:** Απλοποιημένες εκδόσεις του πληροφοριακού συστήματος παρέχονται στους χρήστες δοκιμαστικά, ώστε να δοθεί η δυνατότητα ανάδρασης από αυτούς σε αρχικά στάδια της ανάπτυξης. Η σημαντικότητα της πρωτοτυποποίησης είναι ιδιαίτερα μεγάλη στην περίπτωση εφαρμογών Παγκόσμιου Ιστού, όπως και στην περίπτωση των Υπερμέσων, καθώς η ενδογενής πολυπλοκότητα του περιβάλλοντος διάδρασης απαιτεί έγκαιρη αξιολόγηση της συνολικής και ενιαίας αποτελεσματικότητας της δομής, της πλοήγησης και της παρουσίασης (Nielsen, 1996; Bachiochi et al., 1997). Τυπικά, κατασκευάζεται ένα πρωτότυπο, χρησιμοποιώντας μία απλοποιημένη αρχιτεκτονική πριν το σχεδιασμό του πληροφοριακού συστήματος για παράδειγμα σαν ένα σύνολο από υλοποιημένες ιστοσελίδες, οι οποίες περιέχουν ένα δείγμα του περιεχομένου του και προσομοιώνουν την επιθυμητή εμφάνιση και συμπεριφορά του.

- ***Σχεδιασμός:*** Τα εννοιολογικά σχήματα μετασχηματίζονται σε μία χαμηλότερου επιπέδου αναπαράσταση, η οποία βρίσκεται πιο κοντά στις ανάγκες της υλοποίησης, αλλά είναι ακόμα ανεξάρτητη από το πραγματικό περιεχόμενο της βάσης πληροφοριών. Τυπικά, το δομικό επίπεδο του πληροφοριακού συστήματος αποτυπώνεται στο σχήμα του αποθηκευτικού χώρου του περιεχομένου, το επίπεδο πλοήγησης σε ένα σύνολο δομικών στοιχείων πρόσβασης πάνω στο επίπεδο περιεχομένου και το επίπεδο παρουσίασης αποτυπώνεται σε ένα σύνολο οπτικών προδιαγραφών (στυλ), οι οποίες είναι ανεξάρτητες του περιεχομένου. Η τελευταία ενέργεια, η οποία ονομάζεται οπτικός-γραφικός σχεδιασμός (visual design), είναι πρωτεύουσας σημασίας για το σχεδιασμό και ανάπτυξη πληροφοριακών συστημάτων Παγκόσμιου Ιστού και έχει εμφανιστεί εδώ και αρκετά χρόνια ως ένας νέος αυτόνομος επιστημονικός κλάδος [5].

- ***Υλοποίηση:*** Ο αποθηκευτικός χώρος περιεχομένου γεμίζεται με νέο περιεχόμενο από ειδικούς σε σχέση με το θεματικό πεδίο του πληροφοριακού συστήματος και/ή από δεδομένα τα οποία βρίσκονται σε υπάρχοντα συστήματα. Τα περιβάλλοντα αλληλεπίδρασης (interfaces) με τους χρήστες (ιστοσελίδες στην περίπτωση των πληροφοριακών συστημάτων Παγκόσμιου Ιστού), κατασκευάζονται ενθέτοντας το περιεχόμενο σε συνδυασμό με εντολές πλοήγησης στο κατάλληλο στυλ παρουσίασης. Η απεικόνιση του σχεδιασμού στην υλοποίηση, απαιτεί την επιλογή της γλώσσας στην οποία το πληροφοριακό σύστημα θα αναπτυχθεί (π.χ., Java, HTML, ASP κλπ.) και την απόφαση του χρόνου "σύνδεσης" ανάμεσα στα δεδομένα και τις ιστοσελίδες του πληροφοριακού συστήματος. Το τελευταίο μπορεί να γίνει δυναμικά ή ασύγχρονα.

- ***Εξέλιξη και Συντήρηση:*** Μετά την παράδοση του πληροφοριακού συστήματος, πιθανές αλλαγές στις αρχικές απαιτήσεις, ή διόρθωση λαθών, μπορεί να οδηγήσει σε αναθεώρηση της δομής, της πλοήγησης, της παρουσίασης ή του περιεχομένου. Οι αλλαγές εφαρμόζονται σε όσο πιο υψηλό επίπεδο γίνεται στην "αλυσίδα" του κύκλου ανάπτυξης και διαχέονται έως το επίπεδο υλοποίησης.

Μια εξίσου ενδιαφέρουσα προσέγγιση σχετικά με τον κύκλο ζωής και τις φάσεις που περιλαμβάνει ένα πληροφοριακό σύστημα είναι η παρακάτω:

.

ΦΑΣΗ	ΕΡΩΤΗΜΑΤΑ	ΠΑΡΑΤΗΡΗΣΕΙΣ
Διερευνητική μελέτη (Καθορισμός Προβλήματος)	• Ποιο είναι το υπό εξέταση σύστημα; • Ποιο είναι το πραγματικό πρόβλημα; • Ποιες είναι οι υπάρχουσες εναλλακτικές λύσεις;	Ο χρήστης θα επιλέξει μία λύση για περαιτέρω εξέταση.
Μελέτη Σκοπιμότητας	• Είναι εφικτή η υλοποίηση της λύσης; • Υπάρχουν εναλλακτικοί τρόποι υλοποίησης; • Με ποιο κόστος/όφελος;	Περιγραφή της λύσης που επιλέχθηκε για υλοποίηση.
Ανάλυση Απαιτήσεων	• Ποιες είναι οι βασικές λειτουργίες του συστήματος; • Υπάρχουν ειδικές απαιτήσεις; • Ποια είναι τα κριτήρια αποδοχής των διαφόρων προϊόντων;	Περιγραφή του *τι* πρέπει να κάνει το σύστημα, ανεξάρτητα από την τεχνολογία υλοποίησης.
Σχεδιασμός Συστήματος	• Ποια θα είναι η δομή του συστήματος; • Ποιος θα είναι ο εξοπλισμός σε υλικό και λογισμικό; • Ποιες διαδικασίες απαιτούνται; • Με ποιον τρόπο θα πραγματοποιηθούν οι δοκιμές ελέγχου;	Αναλυτική περιγραφή του *πως* θα είναι το σύστημα. Τεχνικές προδιαγραφές για το υλικό και το λογισμικό που θα χρησιμοποιηθεί.
Υλοποίηση	• Λειτουργεί σωστά το υλικό; • Λειτουργεί σωστά το λογισμικό; • Εκτελούνται σωστά οι διαδικασίες;	Τεκμηρίωση του υλικού, του λογισμικού και των διαδικασιών που υλοποιήθηκαν.
Εγκατάσταση	• Δουλεύει το σύστημα ικανοποιητικά; • Πως θα γίνει η μετάπτωση από το παλιό στο νέο σύστημα;	Εγχειρίδια με οδηγίες χρήσης. Παράδοση – Παραλαβή του συστήματος.
Λειτουργία	• Τι προσθήκες, αλλαγές, τροποποιήσεις και βελτιώσεις απαιτούνται;	Προσπάθεια για ομαλή λειτουργία και συνεχή βελτίωση.

Εικόνα 2 : Κύκλος ζωής ενός Πληροφοριακού Συστήματος [6]

Θεωρητικά, οι φάσεις αυτές πρέπει να βρίσκονται διατεταγμένες σειριακά, δηλαδή η ολοκλήρωση κάθε φάσης να οδηγεί πάντα στην αμέσως επόμενή της. Αυτό, φυσικά, προϋποθέτει ότι τα αποτελέσματα κάθε φάσης θα οδηγούν κατά τρόπο αναμφισβήτητο στην επόμενη, δηλαδή το προϊόν που παράγει κάθε φάση θα γίνεται δεκτό όπως είναι χωρίς να υπάρχει η περίπτωση αλλαγής του αργότερα. Μόνο σε αυτή την περίπτωση είναι δυνατό να παγιωθεί η σειρά εκτέλεσης των διαφόρων φάσεων. Ειδικότερα, για την τελευταία φάση, τη φάση της λειτουργίας – συντήρησης, οι νέες ή/και αυξανόμενες απαιτήσεις που μπορεί να παρουσιαστούν εξετάζονται από την αρχή, δηλαδή ο κύκλος ζωής του Π.Σ. ολοκληρώνεται και αρχίζει ξανά από την πρώτη φάση. Αυτό όμως δύσκολα συμβαίνει στην πράξη, αν δεν ληφθούν τα κατάλληλα μέτρα. Συνήθως οι χρήστες για διάφορους λόγους αλλάζουν συνέχεια τις απαιτήσεις τους, με αποτέλεσμα οι αναλυτές του Π.Σ. να οδηγούνται σε μία συνεχή επανεξέταση των προηγουμένων φάσεων.

2.3 Πληροφοριακά Συστήματα σε μια Επιχείρηση- Οργανισμό

2.3.1 Βασικοί τύποι συστημάτων

Τρεις βασικές κατηγορίες πληροφοριακών συστημάτων εξυπηρετούν διαφορετικά επίπεδα ενός οργανισμού: πληροφοριακά συστήματα λειτουργικών διεργασιών (operational-level systems), πληροφοριακά συστήματα διοίκησης (management-level systems) και συστήματα στρατηγικού επιπέδου (strategic-level systems).

- **Λειτουργικού επιπέδου συστήματα:** Τα συστήματα αυτά υποστηρίζουν τους αντίστοιχους διευθυντές κρατώντας στοιχεία για τις βασικές συναλλαγές οτυ οργανισμού. Ο πρωταρχικός σκοπός των συστημάτων αυτού του επιπέδου είναι η απάντηση ερωτήσεων ρουτίνας και η παρακολούθηση των συναλλαγών του οργανισμού.

- **Διοικητικού επιπέδου συστήματα:** Τα συστήματα αυτά εξυπηρετούν την παρακολούθηση, τον έλεγχο, τη λήψη αποφάσεων και τις διαχειριστικές δραστηριότητες των διευθυντών μεσαίου επιπέδου.

- **Στρατηγικού επιπέδου συστήματα:** Τα συστήματα στρατηγικού επιπέδου επιτρέπουν στην ανώτερη διοίκηση να αντιμετωπίζει και να καθορίζει στρατηγικά θέματα και μακροχρόνιες τάσεις, τόσο εντός του οργανισμού όσο και από το εξωτερικό περιβάλλον.

Για κάθε επίπεδο διοίκησης ο οργανισμός διαθέτει τα αντίστοιχα συστήματα για την κάλυψη των αναγκών του.

1. **Συστήματα Επεξεργασίας Συναλλαγών- Transaction Processing Systems(TPS)**

Τα συστήματα αυτά χρησιμοποιούνται από το λειτουργικό προσωπικό για την διευκόλυνση των καθημερινών τους εργασιών και περιλαμβάνουν γεγονότα και συναλλαγές που λαμβάνουν χώρα στην επιχείρηση. Πωλήσεις, αποστολές, τιμολογήσεις, μισθοδοσία είναι κάποια από τα αντικείμενά τους. Τα συστήματα αυτά παράγουν αναλυτικές αναφορές για την ενημέρωση των χρηστών τους.

2. **Πληροφοριακά Συστήματα Διοίκησης- Management Information Systems (MIS)**

Τα συστήματα αυτά έχοντας πρόσβαση σε μεγάλο όγκο δεδομένων από πολλά τμήματα του οργανισμού παρέχουν στα μεσαία διοικητικά στελέχη αναλυτικές

αναφορές, οι οποίες δημιουργούνται με χρήση απλών μοντέλων διοίκησης όπως κατηγοριοποιήσεις και περιλήψεις.

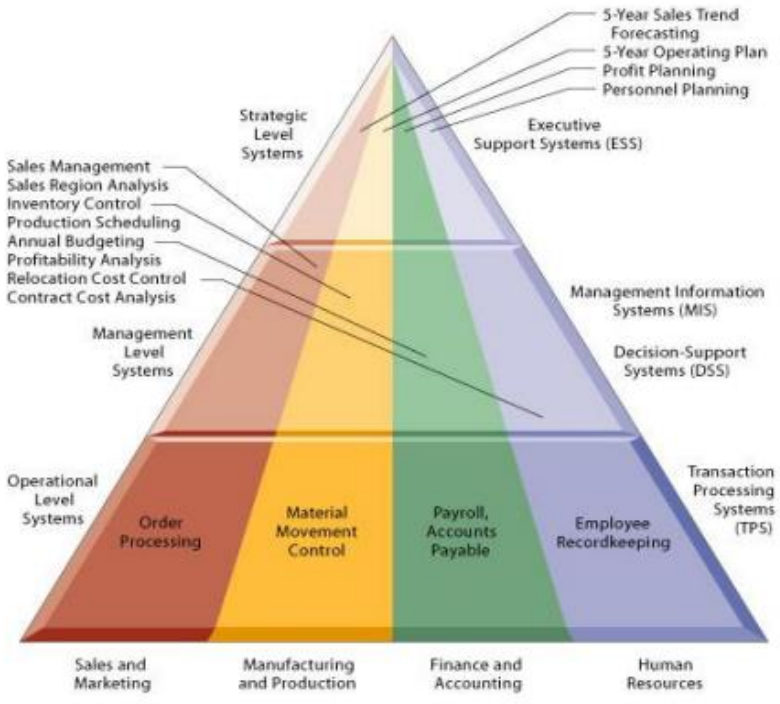
3. Συστήματα Υποστήριξης Αποφάσεων-Decision Support Systems(DSS)

Οι χρήστες τους ανήκουν επίσης στα μεσαία διοικητικά στελέχη και επιτρέπουν την συνεχή αλληλεπίδραση με τον χρήστη, χρησιμοποιούν δεδομένα από τα συστήματα επεξεργασίας συναλλαγών, από τα πληροφοριακά συστήματα διοίκησης καθώς και από εξωτερικές πηγές, και βοηθούν στην λήψη βέλτιστων αποφάσεων. Χρησιμοποιούν αναλυτικά μοντέλα και πολύπλοκα εργαλεία ανάλυσης δεδομένων για την εξαγωγή αποτελεσμάτων.

4. Συστήματα Υποστήριξης Επιτελικών Στελεχών -Executive Support Systems(ESS)

Τα συστήματα αυτά καλούνται να απαντήσουν στα ερωτήματα των επιτελικών διοικητικών στελεχών. Παρουσιάζουν συγκεντρωτικές αναφορές από τα επιμέρους τμήματα του οργανισμού, παρουσιάζουν τις μακροχρόνιες τάσεις της αγοράς και βοηθούν στον μακροπρόθεσμο σχεδιασμό της στρατηγικής του οργανισμού.

Στην εικόνα που ακολουθεί γίνεται μια ενδεικτική παρουσίαση των διαφορετικών συστημάτων ενός οργανισμού. Ο οργανισμός έχει συστήματα υποστήριξης επιτελικών στελεχών (ESS) για το στρατηγικό επίπεδο, πληροφοριακά συστήματα διοίκησης (MIS) και συστήματα υποστήριξης αποφάσεων (DSS) για το διοικητικό επίπεδο και τέλος συστήματα επεξεργασίας συναλλαγών (TPS) για το λειτουργικό επίπεδο. [7]



Εικόνα 3: Παρουσίαση συστημάτων ενός οργανισμού

2.3.2 Πλεονεκτήματα Χρήσης Πληροφοριακών Συστημάτων

Η χρήση των πληροφοριακών συστημάτων αντικατέστησε το μεγαλύτερο ποσοστό χειρόγραφων εγγράφων με αντίστοιχα ηλεκτρονικά. Έτσι έγινε πιο οικονομική και γρήγορη η πρόσβαση σε δεδομένα, ενώ έπαψε να είναι χρονοβόρα και πολύπλοκη η διαχείριση τους. Ακόμη, η επεξεργασία των στοιχείων με στατιστικά προγράμματα που παράγονται αυτόματα από τα πληροφοριακά συστήματα, προσφέρει νέες λύσεις και προτάσεις για την αντιμετώπιση προβλημάτων που αφορούν την επιχείρηση. Ο συνεχής έλεγχος και η ταξινόμηση των στοιχείων εξασφαλίζει ευελιξία και σωστή λειτουργία της επιχείρησης, καθώς και καλή εξυπηρέτηση των πελατών.
Ένα άλλο καίριο πλεονέκτημα των πληροφοριακών συστημάτων είναι η δυνατότητα διαλειτουργικότητας που προσφέρουν. Με τον όρο αυτό εννοούμε τη δυνατότητα μεταφοράς και χρήσης της πληροφορίας με ενιαίο και αποτελεσματικό τρόπο από διαφορετικούς οργανισμούς ανταλλαγής και ενοποίησης (integration) δεδομένων που προέρχονται από διαφορετικά πληροφοριακά περιβάλλοντα μέσω της υιοθέτησης κοινών προτύπων.

Συνοψίζοντας, τα πλεονεκτήματα των πληροφοριακών συστημάτων σε μια επιχείρηση Είναι τα ακόλουθα:

- **Διευκολύνει τον προγραμματισμό**: Το σύστημα βελτιώνει την ποιότητα των οργανισμών παρέχοντας την κατάλληλη πληροφόρηση για σωστές αποφάσεις. Εξαιτίας της αύξησης τόσο των μεγεθών όσο και της πολυπλοκότητας των οργανισμών, τα διευθυντικά στελέχη έχουν χάσει την προσωπική επαφή με τις βασικές λειτουργίες του οργανισμού.

- **Ελαχιστοποιεί την υπερπληροφόρηση**: Το σύστημα μετατρέπει τον μεγάλο όγκο δεδομένων σε συνοπτικές αναφορές, αποφεύγοντας την σύγχυση που ενδέχεται να προκύψει όταν τα στελέχη υπερφορτώνονται από πολλές λεπτομέρειες, πολλές φορές από διαφορετικά συστήματα.

- **Ενθαρρύνει την αποκεντροποίηση**: Η αποκεντροποίηση των ευθυνών καθίσταται εφικτή όταν το σύστημα είναι σε θέση να παρακολουθεί τις λειτουργίες ακόμα και σε χαμηλά επίπεδα της επιχείρησης. Το σύστημα χρησιμοποιείται αποτελεσματικά για την μέτρηση της αποδοτικότητας και για τις απαραίτητες αλλαγές στις διαδικασίες εντός του οργανισμού.

- **Συντονίζει**: Το σύστημα παρέχει την ενσωμάτωση εξειδικευμένων εργασιών, προσφέροντας σε κάθε τμήμα ενημέρωση για τα προβλήματα και τις απαιτήσεις των άλλων τμημάτων. Ενοποιεί τα κέντρα λήψεως αποφάσεων εντός του οργανισμού.

- **Διευκολύνει τον έλεγχο**: Το σύστημα υπηρετεί σαν σύνδεσμος μεταξύ διαχειριστικού σχεδιασμού και ελέγχου. Βελτιώνει την ικανότητα της διοίκησης να αξιολογεί και να βελτιώνει τις επιδόσεις της.

- **Το σύστημα συγκεντρώνει, επεξεργάζεται, αποθηκεύει, ανακτά, αξιολογεί και μεταδίδει την πληροφορία.**

2.3.3 Μειονεκτήματα Χρήσης Πληροφοριακών Συστημάτων

Παρά την τεράστια επιρροή των Πληροφοριακών Συστημάτων σε οργανισμούς-επιχειρήσεις, η χρήση τους δημιούργησε και αρκετά προβλήματα. Η ανάπτυξη ενός οποιουδήποτε ΠΣ έχει πολλές απαιτήσεις που πρέπει να ικανοποιηθούν για τη πλήρη λειτουργία του. Τα πλέον συνήθη προβλήματα που προκύπτουν είναι τα εξής:

- Πολλές φορές ένα πληροφοριακό σύστημα μπορεί να μην είναι εύχρηστο για χρήστες που δεν είναι εξοικειωμένοι αρκετά με την τεχνολογία, με αποτέλεσμα να δυσανασχετούν και να μην μπορούν να το χρησιμοποιήσουν παραγωγικά

- Ένα πληροφοριακό σύστημα-αν έχει ατέλειες- μπορεί να επιστρέψει περιττές πληροφορίες και ίσως δεν καταφέρει να ικανοποιήσει τις βασικές ανάγκες του χρήστη. Αυτό συμβαίνει συνήθως διότι είναι δύσκολος ο καθορισμός των πραγματικών απαιτήσεων μιας επιχείρησης όταν δημιουργείται το πληροφοριακό σύστημα

- Οι συνεχείς αλλαγές και αναβαθμίσεις στο λογισμικό ενδέχεται να έχουν μεγάλο οικονομικό κόστος στην επιχείρηση προκειμένου το λογισμικό της να είναι σύγχρονο.

- Η συντήρηση ενός πλήρους πληροφοριακού συστήματος χρειάζεται διαρκή έλεγχο και αναβάθμιση σε εξοπλισμό, καταρτισμένο προσωπικό και άμεση αποκατάσταση τυχών λαθών ώστε να αποφευχθούν περισσότερα προβλήματα.

3 Σύστημα Διαχείρισης Χρηματοδοτήσεων

3.1 Το Πληροφοριακό Σύστημα Διαχείρισης Χρηματοδοτήσεων

Το «Πληροφοριακό Σύστημα Διαχείρισης Χρηματοδοτήσεων» υλοποιήθηκε στα πλαίσια της Ειδικής Υπηρεσίας Διαχείρισης του Ε.Π «Ψηφιακή Σύγκλιση» η οποία είναι υπεύθυνη για τη συλλογή των προτάσεων προγραμματισμού της χρηματοδότησης των πράξεων από τους δικαιούχους.

Μέσω του πληροφοριακού συστήματος γίνεται η υποβολή του ετησίου πίνακα προγραμματισμού (προτάσεις ετησίων πιστώσεων) καθώς και για την υποβολή αιτημάτων χρηματοδότησης (πίνακας ΙΙ και πίνακας προβλεπόμενων πληρωμών).

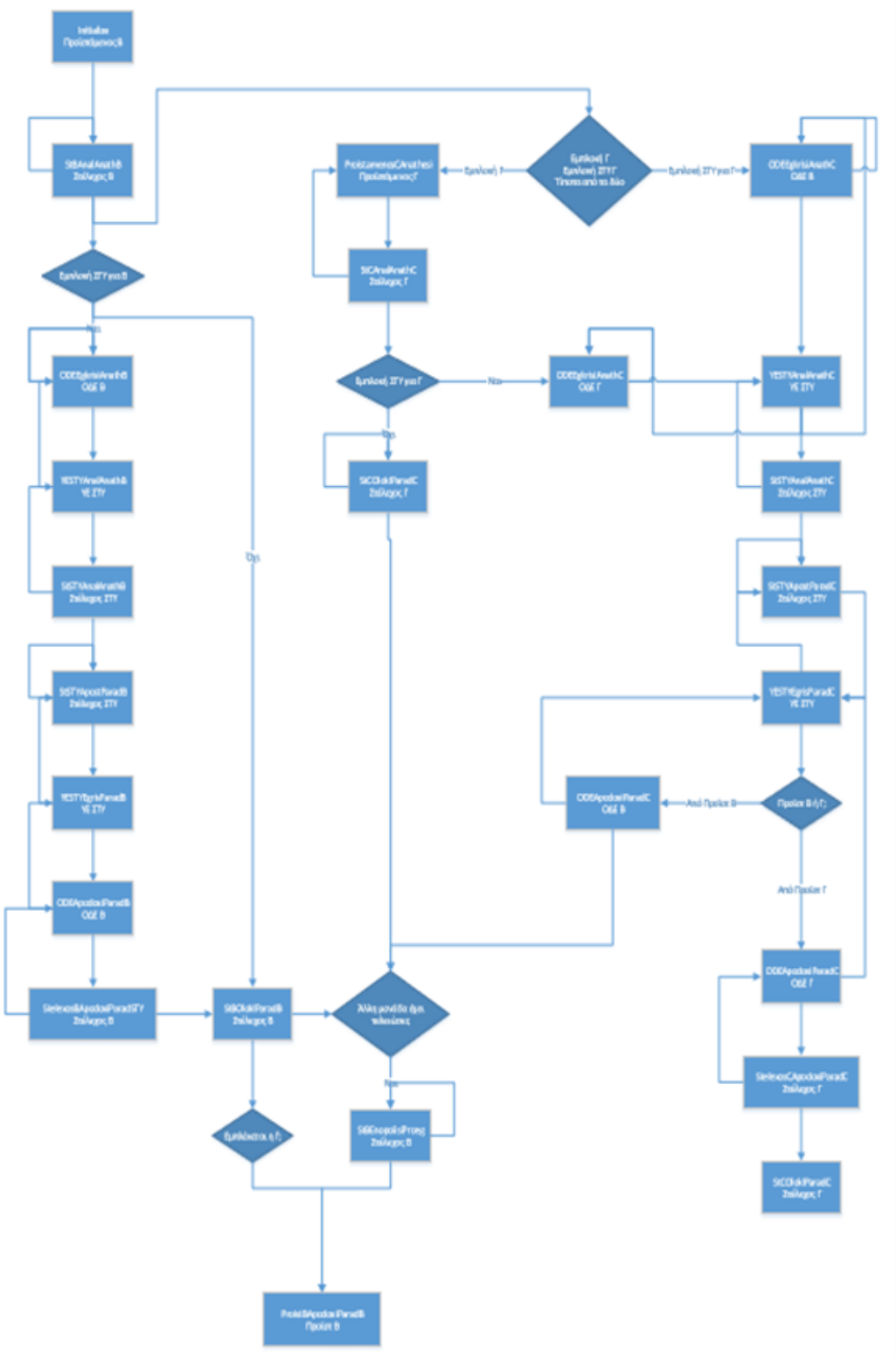
Η υποβολή των συγκεκριμένων στοιχείων γίνεται από τα στελέχη του φορέα τα οποία ορίζονται συμπληρώνοντας μια ειδική φόρμα . Τα στελέχη που ορίζονται ως υπεύθυνα για την υποβολή των εν λόγω αιτημάτων μπορεί να είναι οι υπεύθυνοι έργων όπως αυτοί έχουν ορισθεί στο εγκεκριμένο ΤΔΠ ή όποιο άλλο άτομο στο οποίο έχει δοθεί η συγκεκριμένη αρμοδιότητα (πχ Δ/νση Οικονομικού του δικαιούχου).

Ακόμη, το πληροφοριακό σύστημα καλύπτει ορισμένες αρμοδιότητες της Ε.Υ.Δ.Ε.Π «ΨΣ», οι οποίες σύμφωνα με τον Κανονισμό 1083/2006 και τον Νόμο 3614/2007 όπως αυτός έχει τροποποιηθεί και ισχύει καθορίζονται ως εξής:

- Επιλέγει τις προς χρηματοδότηση πράξεις σύμφωνα με τα κριτήρια που εφαρμόζονται στο Ε.Π «ΨΣ» και διασφαλίζει τη συμμόρφωση τους με τους ισχύοντες κοινοτικούς και εθνικούς κανόνες, καθ` όλη την περίοδο υλοποίησης του.
- Διενεργεί επαληθεύσεις κατά τα οριζόμενα στο άρθρο 8 του Νόμου 3614.
- Διασφαλίζει τη συλλογή και καταχώρηση στο ΟΠΣ των δεδομένων προγραμματισμού και υλοποίησης που απαιτούνται για τη χρηματοοικονομική διαχείριση, την παρακολούθηση, τις επαληθεύσεις, τους ελέγχους και την αξιολόγηση, καθώς και των λογιστικών εγγραφών για κάθε πράξη στο πλαίσιο του Ε.Π «ΨΣ». Είναι υπεύθυνη για την ακρίβεια, την ποιότητα και πληρότητα των στοιχείων που καταχωρούνται στο ΟΠΣ.

Μια άλλη λειτουργικότητα του Πληροφοριακού συστήματος είναι η διαδικασία των προεγκρίσεων ενός αιτήματος χρηματοδότησης, η οποία αποτυπώνεται σχηματικά στην Εικόνα 4 που ακολουθεί και δείχνει επιπλέον και τους διαφορετικούς ρόλους που έχουν οριστεί.

Τέλος, μέσω του πληροφοριακού συστήματος δίνεται η δυνατότητα εξαγωγής των σχετικών πινάκων σε μορφή .xls τα οποία μπορούν να αποθηκευτούν και να εκτυπωθούν για να είναι δυνατή η άμεση αποστολή τους. [8]



Εικόνα 4: Διάγραμμα ροής διαδικασιών σε Π.Σ.Δ.Χ

3.2 Τεχνολογίες που χρησιμοποιήθηκαν κατά την υλοποίηση του

3.2.1 Εισαγωγή

Τα τελευταία δέκα χρόνια, έχει γίνει μία πραγματική επανάσταση στον χώρο των διαδικτυακών εφαρμογών τόσο σε επίπεδο εμπορευματοποίησης όσο και επίπεδο τεχνολογικής προόδου. Έτσι, έχουν δημιουργηθεί ή/και εξελιχθεί πληθώρα γλωσσών και εργαλείων προσανατολισμένα στον διαδικτυακό προγραμματισμό όπως η ASP, Ruby, JavaScript, Ajax, JQuery, CSS, XTHML, HTML 5, Flash κ.ά. Κάθε μία από αυτές τις τεχνολογίες συμπληρώνει ή/και ανταγωνίζεται ή/και επεκτείνει τις άλλες μα στο σύνολό τους δημιουργούν ένα σύστημα το οποίο σκοπό έχει την επέκταση των δυνατοτήτων του ίδιου του Παγκόσμιου Ιστού.
Στο κεφάλαιο αυτό περιγράφονται συνοπτικά οι τεχνολογίες και τα εργαλεία ανάπτυξης που χρησιμοποιήθηκαν κατά την υλοποίηση της εφαρμογής.

3.2.2 Περιβάλλον Ανάπτυξης του Πληροφοριακού Συστήματος

3.2.2.1. Microsoft Visual Studio

Το Visual Studio είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (Integrated Development Environment - IDE) που χρησιμοποιείται για την ανάπτυξη διαδικτυακών και desktop εφαρμογών καθώς και διαδικτυακών υπηρεσιών (Web Services). Το Visual Studio εισάγει μια νέα διαδικασία ανάπτυξης προγραμμάτων, τον παραστατικό προγραμματισμό, που αλλάζει τον τρόπο εγγραφής και εκτέλεσης των προγραμμάτων, οδηγώντας σε αύξηση της παραγωγικότητας. Παρέχει προχωρημένα εργαλεία για τη διόρθωση λαθών, για τη τεκμηρίωση και εγγραφή κώδικα, για την ανάπτυξη διεπαφών χρήστη, για τη σχεδίαση κλάσεων καθώς και για τη σχεδίαση του σχήματος μίας βάσης δεδομένων (database schema). Επίσης υποστηρίζει πολλά plug-ins που προσφέρουν προηγμένες λειτουργίες όπως unit testing και refactoring. Οι ενσωματωμένες γλώσσες προγραμματισμού του Visual Studio είναι οι Visual C++, Visual C# και Visual Basic. Επίσης, παρέχεται υποστήριξη και για άλλες γλώσσες όπως τις F#, Python, Ruby οι οποίες εγκαθίστανται ξεχωριστά μέσω των language services. [9]

3.2.2.2. .NET Framework

Το .NET Framework είναι μια πλατφόρμα ανάπτυξης λογισμικού, για την δημιουργία εφαρμογών που προορίζονται για τα λειτουργικά συστήματα Windows, Windows phone, Windows Server και Windows Azure. Αποτελείται από την Common Language Runtime (CLR) , που ουσιαστικά είναι ένα εικονικό περιβάλλον εκτέλεσης των προγραμμάτων .NET, και την βιβλιοθήκη κλάσεων .NET Framework class library. [10] Γενικά, το .NET Framework παρέχει ένα ευέλικτο αλλά ταυτόχρονα ελεγχόμενο περιβάλλον εκτέλεσης εφαρμογών, ευκολία στην ανάπτυξη εφαρμογών και project, και

ταυτόχρονη χρήση πολλών γλωσσών προγραμματισμού, οι οποίες είναι οι .NET Languages. Πιο συγκεκριμένα οι βασικές ψηφίδες του .NET Framework είναι οι ακόλουθες:

- **.NET Framework Base Class Library**. Περιλαμβάνει πολλές βιβλιοθήκες κλάσεων με σκοπό να παρέχουν έτοιμο κώδικα στους προγραμματιστές, για λειτουργίες που θεωρείται ότι είναι συχνά χρησιμοποιούμενες, και μπορούν να μοντελοποιηθούν αποτελεσματικά. Παραδείγματα τέτοιων λειτουργιών είναι η εγγραφή και ανάγνωση σε αρχεία ή στην οθόνη, η επικοινωνία με βάσεις δεδομένων, η δημιουργία γραφικών στοιχείων κλπ.

- **.NET Languages.** Είναι ένα σύνολο γλωσσών προγραμματισμού που μπορούν να συνδυαστούν, και στο ίδιο Project, και η επικοινωνία μεταξύ τους. Αυτές οι γλώσσες είναι οι C#, Visual Basic, JS script, F#, J#, C++. Γενικά οι γλώσσες αυτές ακολουθούν τις αρχές του αντικειμενοστραφούς προγραμματισμού.

- **Language Integrated Query (LINQ).** Είναι μία ψηφίδα του .NET Framework που παρέχει την δυνατότητα διεξαγωγής ερωτημάτων σε δεδομένα (querying capabilities) στις γλώσσες του .NET. Αυτή η δυνατότητα είναι πολύ σημαντική, αφού τα περισσότερα πληροφοριακά συστήματα είναι στενά συνδεδεμένα με δεδομένα σε κάποια βάση δεδομένων, και το γεγονός ότι παρέχεται φυσική υποστήριξη από την γλώσσα (δυνατότητα αυτόματης συμπλήρωσης, επισήμανση λαθών, σύνθετες πράξεις με τα δεδομένα) αυξάνει σημαντικά την παραγωγικότητα των προγραμματιστών. [11]

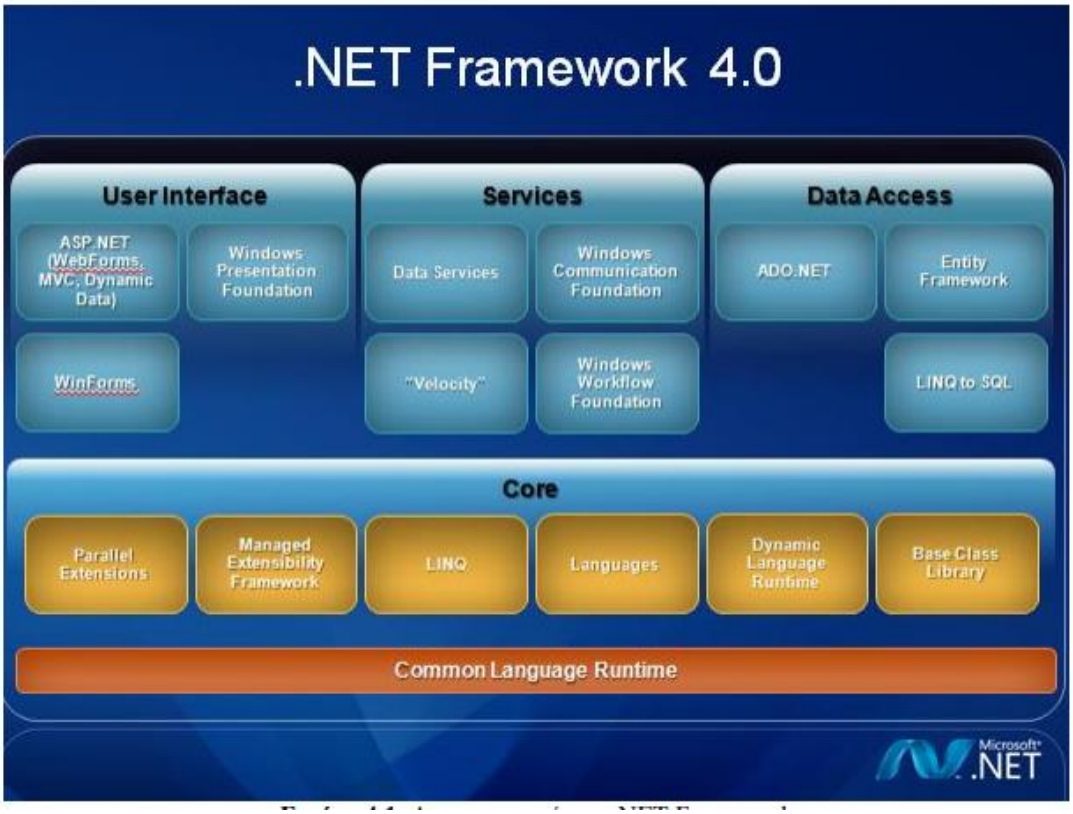
- **ASP.NET.** Το ASP.NET είναι ένα server-side Web application Framework που έχει σχεδιαστεί με σκοπό να παρέχει την ικανότητα στους Web Developers να δημιουργούν δυναμικές ιστοσελίδες. Είναι ένα σχετικά καινούριο Framework, που εμφανίστηκε το 2002. Είναι δομημένο στο Common Language Runtime (CLR), και επιτρέπει στους προγραμματιστές να χρησιμοποιήσουν την .NET γλώσσα προγραμματισμού της αρεσκείας τους. Το ASP.NET υποστηρίζει τρία διαφορετικά μοντέλα ανάπτυξης, που είναι τα Web Pages, Web Forms, και το MVC που είναι και αυτό που χρησιμοποιήθηκε για την ανάπτυξη του παρόντος πληροφοριακού συστήματος. [12]

- **Intermediate Language (IL).** Μία μορφή byte code, που αποτελεί ενδιάμεση μορφή μεταγλώττισης, σε ένα εικονικό περιβάλλον εκτέλεσης, γνωστό ως CLR, και σκοπό έχει ο κώδικας που παράγεται να είναι ανεξάρτητος της αρχιτεκτονικής του εκάστοτε επεξεργαστή.

- **Common Language Runtime (CLR).** Το εικονικό περιβάλλον (virtual machine), που ελέγχει και εκτελεί τις .NET εφαρμογές.

- **Garbage Collection.** Το εικονικό περιβάλλον CLR είναι υπεύθυνο για την αποδέσμευση των μη χρησιμοποιούμενων πόρων που δεσμεύουν οι .NET εφαρμογές.

- **Common Type System (CTS), και Common Language Specification (CLS).** Είναι ένα σύνολο από κανόνες οι οποίοι έχουν σκοπό να καταστήσουν δυνατή την επικοινωνία μεταξύ των διαφορετικών .NET Languages. Με αυτόν τον τρόπο είναι δυνατή ή εύκολη μετατροπή κώδικα από μια γλώσσα σε μία άλλη, αφού ουσιαστικά οι τύποι των δεδομένων (τα αντικείμενα) υπακούουν σε κοινούς κανόνες, και έχουν το ίδιο interface προς τον προγραμματιστή.



Εικόνα 5 : Αρχιτεκτονική του NET Framework

Πριν επεκταθούμε σε μια πιο αναλυτική παρουσίαση της δομής της πλατφόρμας .NET και των τεχνολογιών που την περιβάλλουν και χρησιμοποιήθηκαν στην ανάπτυξη του παρόντος πληροφοριακού συστήματος είναι σκόπιμο να παρουσιάσουμε τα σημαντικότερα χαρακτηριστικά και πλεονεκτήματά της:

- **Υποστήριξη πολλαπλών γλωσσών.** Στο .NET, η διαλειτουργικότητα μεταξύ των γλωσσών επιτυγχάνεται μέσω της διαγλωσσικής κληρονομικότητας. Μαζί με ένα ενοποιημένο σύστημα τύπων, η συνένωση κώδικα γραμμένου σε

διαφορετικές γλώσσες είναι πλέον εύκολη υπόθεση. Αυτό διευκολύνει και άλλα προγραμματιστικά πρότυπα όπως το συναρτησιακό της F#.

- **Ανάπτυξη βασισμένη σε components.** Η δημιουργία ανεξάρτητων κομματιών λογισμικού που περιέχουν βιβλιοθήκες κλάσεων και διαμοιράσιμα τμήματα λειτουργικότητας μιας εφαρμογής έχει γίνει πολύ πιο εύκολη από ότι ήταν παλιότερα. Η διαμοιράσιμη μονάδα κώδικα στο .NET βασίζεται στην έννοια του assembly, το οποίο μεταφέρει πληροφορίες που απαιτούνται για τη χρήση της όπως η έκδοση της.

- **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 ενδιάμεση 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, η οποία ωστόσο μένει μόνο στη θεωρία, αφού η μοναδική πλήρης υλοποίηση του .NET είναι διαθέσιμη μόνο για την πλατφόρμα των Windows. Όμως είναι διαθέσιμη μια μερική υλοποίηση ανοιχτού κώδικα με το όνομα Mono.

- **Απλή εγκατάσταση εφαρμογών.** Αντίθετα με τα components που βασίζονται στο COM, δεν χρειάζεται καμία καταχώρηση στο registry του συστήματος για την εγκατάσταση των assemblies καθώς μια απλή αντιγραφή αρκεί για τη χρήση τους. Επιπρόσθετα τα πιθανά προβλήματα της χρήσης των Dynamic Link Libraries (DLLs) απαλείφθηκαν με την υποστήριξη της ύπαρξης πολλαπλών εκδόσεων του ίδιου component. Το ίδιο το .NET Framework είναι ένα καλό παράδειγμα αυτής της δυνατότητας, αφού είναι δυνατόν να υπάρχουν πολλαπλές εκδόσεις του framework εγκατεστημένες στο ίδιο μηχάνημα.

- **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 όπως της COM και παρέχει πρόσβαση σε λειτουργίες χαμηλού επιπέδου του λειτουργικού συστήματος μέσω του μηχανισμού P/Invoke.

- **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.

- **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 (όπως στην περίπτωση του COM). Ένας από τους στόχους του .NET και του CLR πιο συγκεκριμένα ήταν να απαλείψει από τους προγραμματιστές την ανάγκη διαχείρισης της μνήμης. 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.

- **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. Το μοντέλο ασφάλειας συμβαδίζει με τους μηχανισμούς ασφάλειας που παρέχει το λειτουργικό σύστημα όπως οι Access Control Lists και τα Windows security tokens.

- **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 ξαναμεταγλωττιστεί 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.

- **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 2010. 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. [13]

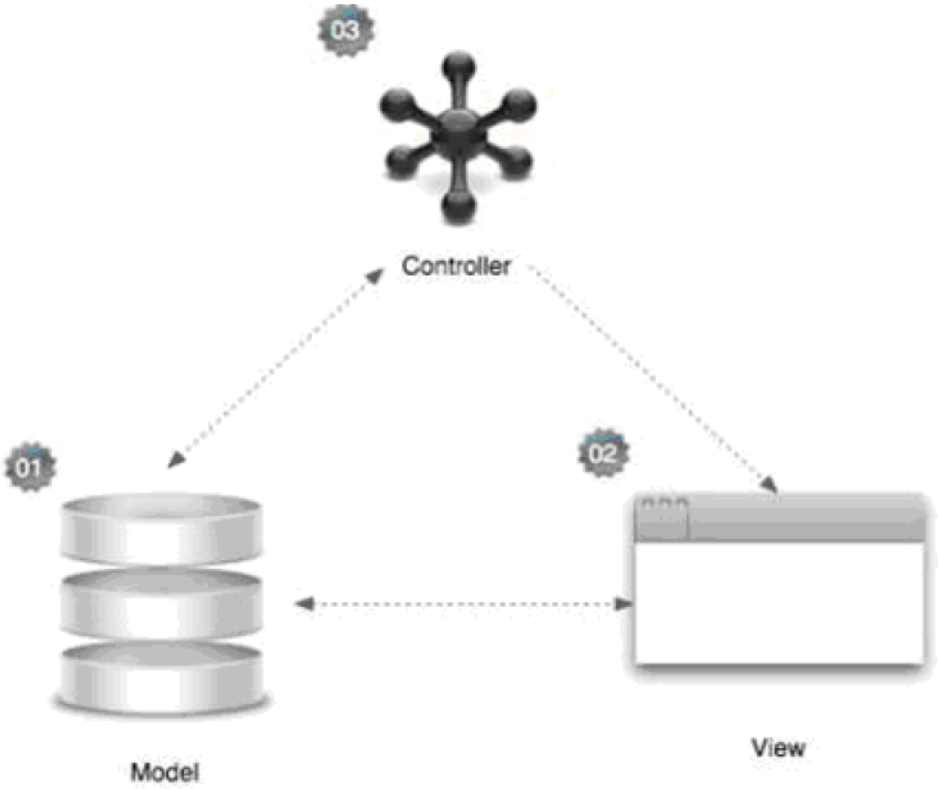
3.2.2.3. ASP.NET MVC

Το MVC (Model – View – Controller) είναι ένα μοντέλο αρχιτεκτονικής, που διαχωρίζει την εφαρμογή σε τρία κύρια συστατικά: το Model, το View, και τον Controller. Το ASP.NET MVC Framework ακολουθεί αυτήν την αρχιτεκτονική ανάπτυξης και αποτελεί ένα ελαφρύ και με μεγάλες δυνατότητες συντήρησης και ελεγξιμότητας Framework, που διατηρεί όλα τα πλεονεκτήματα του ASP.NET όπως τα λεγόμενα Master Pages ή το membership bashed authentication. Ήδη, αρκετοί προγραμματιστές προτιμούν αυτήν την αρχιτεκτονική και έχουν υιοθετήσει το συγκεκριμένο Framework αφού η χρήση του ενδείκνυται για πλήθος εφαρμογών. [14] Οι τρεις βασικές ψηφίδες, όπως αναφέρθηκαν και στην προηγούμενη παράγραφο, είναι οι ακόλουθες:

- **Model.** Είναι τα αντικείμενα εκείνα που υλοποιούν το business logic της εφαρμογής. Αποτελούν δηλαδή τις κύριες οντότητες της εφαρμογής, αφού η κατάσταση των αντικειμένων αυτών προσδιορίζει την συμπεριφορά της εφαρμογής. Τα αντικείμενα του μοντέλου συχνά αντιστοιχούν σε πίνακες κάποιας βάσης δεδομένων, οπότε ανακτούν τις τιμές από την βάση δεδομένων, ενεργούν πάνω στα δεδομένα, και στέλνουν πίσω τις ανανεωμένες πληροφορίες.

- **Views**. Είναι οι ψηφίδες που είναι υπεύθυνες για την παρουσίαση του User Interface (UI) της εφαρμογής. Τυπικά, αυτό το UI παράγεται από το μοντέλο.

- **Controllers.** Είναι οι ψηφίδες που είναι υπεύθυνες για την αλληλεπίδραση με τον χρήστη. Λαμβάνουν δηλαδή την είσοδο του χρήστη, επεξεργάζονται το μοντέλο (model) βάση αυτής, και επιλέγουν ποιο view θα εμφανιστεί τελικά στον χρήστη. Ο controller δηλαδή χειρίζεται την είσοδο από τον χρήστη, την μεταφέρει στο μοντέλο, που με την σειρά του ενεργεί πάνω στα δεδομένα (στην βάση δεδομένων). Τέλος επιλέγεται ποιο View θα εμφανιστεί στον χρήστη.



Εικόνα 6 : Τα τρία διακριτά τμήματα του προτύπου MVC

Το MVC Framework όπως είδαμε κάνει σαφή διαχωρισμό των διαφορετικών τμημάτων κάθε εφαρμογής, δηλαδή των Input logic, business logic και UI Logic, και ταυτόχρονα προσφέρει έναν εύκολο και ευέλικτο τρόπο επικοινωνίας μεταξύ τους. Παρέχει δηλαδή ένα είδος καθοδήγησης στον προγραμματιστή σχετικά με το που πρέπει να βρίσκεται το κάθε τμήμα μέσα στην εφαρμογή, γεγονός που οδηγεί σε ευανάγνωστο και εύκολα συντηρήσιμο κώδικα, και ελαττώνει την πολυπλοκότητα αφού επιτρέπει στον προγραμματιστή να επικεντρώνεται σε ένα κομμάτι της εφαρμογής κάθε φορά (model ή view ή controller). Ακόμη, η προσέγγιση αυτή καθιστά εύκολη την ανάπτυξη μεγάλων project από ομάδες πολλών ατόμων. Γενικά, το ASP.NET MVC προσφέρει :

- Σαφή διαχωρισμό των διαδικασιών της εφαρμογής (Input Logic – Business Logic – UI Logic), Test Driven Development (TDD) και υψηλή συντηρισημότητα της εφαρμογής.

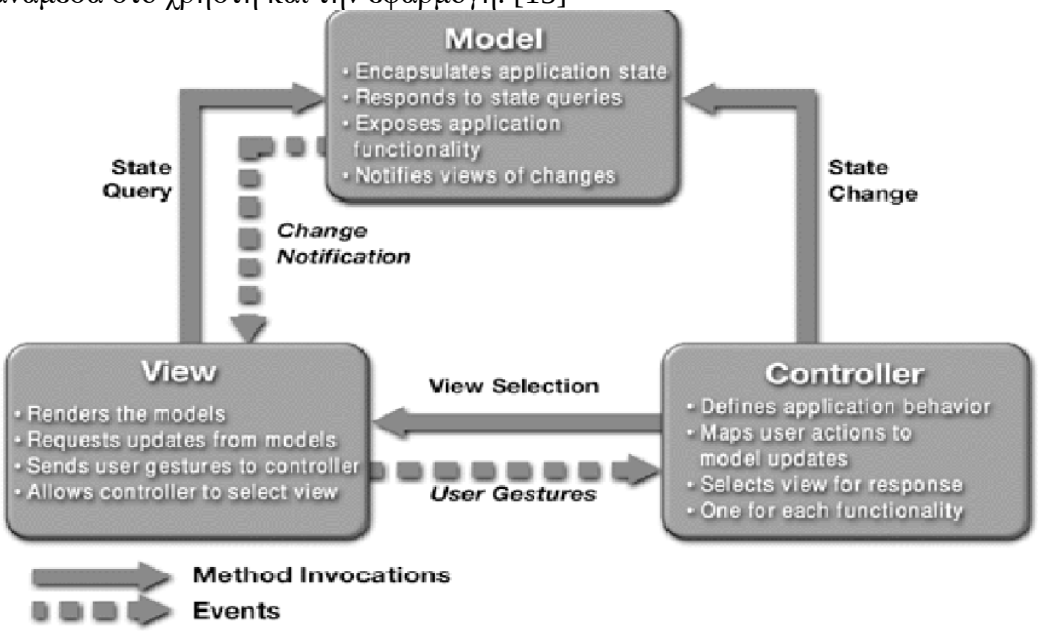
- Ένα επεκτάσιμο και παραμετροποιήσιμο Framework. Όλες οι ψηφίδες του είναι σχεδιασμένες με τέτοιο τρόπο, ώστε να μπορούν εύκολα να αντικαθίστανται ή να τροποποιούνται. Μπορούμε για παράδειγμα να κατασκευάσουμε και να συνδέσουμε την δικιά μας View engine, Routing Policy και γενικά διάφορες ψηφίδες.

- Έναν ισχυρό μηχανισμό URL – mapping , που ξεφεύγει από την κλασσική προσέγγιση της αναζήτησης κάποιου αρχείο σε απομακρυσμένο server. Τα URL's δεν αντιπροσωπεύουν τα αρχεία με τις επεκτάσεις τους (πχ home.php) , αλλά είναι ευανάγνωστα και εύκολα ανακτήσιμα, και σχεδιασμένα να υποστηρίζουν διάφορα URL naming patterns που επωφελούνται από το λεγόμενο Search Engine Optimization (SEO) και το Representational State Transfer (REST) addressing.

- Υποστήριξη για την χρήση ισχυρής γλώσσας προγραμματισμού σε κάποια ιστοσελίδα, Master Pages.

Αλληλεπίδραση μεταξύ των τριών τμημάτων

Η αλληλεπίδραση μεταξύ των τριών τμημάτων έχει κοινή δομή για όλες τις εφαρμογές που χρησιμοποιούν το πρότυπο MVC. Ο Ελεγκτής βρίσκεται στον πυρήνα της αρχιτεκτονικής και αλληλεπιδρά με τον εκάστοτε χρήστη ενώ στο παρασκήνιο υπάρχει επικοινωνία μεταξύ του στοιχείων του Μοντέλου και της Όψης. Το Μοντέλο έχει πρόσβαση στα δεδομένα έτσι ώστε ο Ελεγκτής να έχει τη δυνατότητα ενημέρωσης σε κάθε είσοδο που δέχεται από το χρήστη κάτι που αυξάνει το βαθμό αλληλεπίδρασης ανάμεσα στο χρήστη και την εφαρμογή. [15]



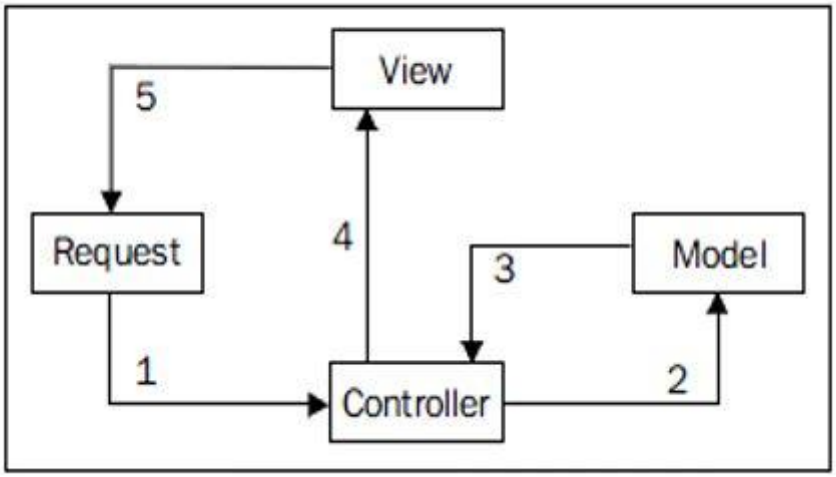
Εικόνα 7: Η αλληλεπίδραση ανάμεσα στα στοιχεία του MVC

Όπως φαίνεται στην παραπάνω εικόνα, κατά τη λήψη μίας αίτησης από το χρήστη, ο Ελεγκτής χειρίζεται το Μοντέλο και στη συνέχεια το Μοντέλο ενημερώνει την Όψη

ανανεώνοντας τη διεπιφάνεια χρήστη (UI) σε συνάρτηση με τις υφιστάμενες αλλαγές. Κάθε φορά που υπάρχει νέα αίτηση από το χρήστη ο κύκλος αυτός των ενεργειών εκτελείται εκ νέου. [16]

Παράδειγμα χειρισμού αίτησης με το MVC

Γενικά, ο τρόπος με τον οποίο γίνεται ο χειρισμός κάθε αίτησης φαίνεται στην ακόλουθη εικόνα και είναι ο παρακάτω:



Εικόνα 8: Γενικός τρόπος επεξεργασίας μίας αίτησης

- Η αίτηση κατευθύνεται στον ελεγκτή περιέχοντας τα δεδομένα που εισήγαγε ο χρήστης.
- Ο ελεγκτής επεξεργάζεται την αίτηση και καλεί το μοντέλο για να αποκτήσει πρόσβαση στα δεδομένα.
- Το μοντέλο απαντά στην κλήση του ελεγκτή αποστέλλοντας ή αποθηκεύοντας δεδομένα.
- Ο ελεγκτής στέλνει τα δεδομένα προς έξοδο στην όψη.
- Η όψη εμφανίζει τα δεδομένα στην κατάλληλη μορφή. Τα βήματα αυτά αποτελούν τη γενική δομή που ακολουθεί μία οποιαδήποτε αίτηση του χρήστη χωρίς όμως αυτό να σημαίνει ότι δε μπορούν να διαφέρουν στις εκάστοτε περιπτώσεις. [16]

3.2.2.4. Η γλώσσα προγραμματισμού C#

Η C# είναι μία αντικειμενοστραφής γλώσσα προγραμματισμού, συνεπώς είναι υψηλού επιπέδου. Αυτό σημαίνει πως είναι φιλική προς τον προγραμματιστή, αφού

αποκρύπτει τις λεπτομέρειες του υλικού του υπολογιστικού συστήματος, και παρέχει έναν ενιαίο τρόπο επικοινωνίας προς αυτό, μέσω των έτοιμων κλάσεων και βιβλιοθηκών που προσφέρει. Αποτελεί μια συνεχώς εξελισσόμενη γλώσσα, και σε κάθε νέα της έκδοση προστίθενται νέα χαρακτηριστικά και συντακτικό, με στόχο να κάνει τα πράγματα ευκολότερα για τον προγραμματιστή. Ακόμη, μέσω του ASP.NET μπορεί να ενσωματωθεί κώδικας σε C# ανάμεσα σε τυπικό κώδικα HTML με ξεκάθαρο και αποτελεσματικό τρόπο, βοηθώντας την κατασκευή δυναμικών ιστοσελίδων. [17] Εν συντομία, τα βασικά χαρακτηριστικά της είναι:

- **Απλότητα (Simplicity).** 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.

- **Αντικειμενοστρέφεια (Object – Orientation).** 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.

- **Οικειότητα (Familiarity).** Στο ξεκίνημα της υλοποίησής της, οι δημιουργοί της τεχνολογίας C# απέρριψαν την ολοκληρωτική χρήση της C++ σαν γλώσσα υλοποίησης. Στη νέα γλώσσα ωστόσο, κράτησαν αρκετά

χαρακτηριστικά της C++ αλλά και της Java και αφαίρεσαν την άχρηστη πολυπλοκότητα και των δύο. Έτσι έχοντας κρατήσει αρκετά από τα αντικειμενοστραφή χαρακτηριστικά και τη γενική φιλοσοφία της C++ αλλά και τη γενική ευκολία της Java βελτιώνοντας ορισμένα σημεία της, είναι σχετικά εύκολη τη «μετακόμιση» στη C# δεδομένου ότι η C η C++ αλλά και η Java είναι ευρέως γνωστές και χρησιμοποιούνται συχνά.

3.2.2.5. AJAX και JQuery

Με τον όρο AJAX (Asynchronous JavaScript and XML) αναφερόμαστε σε μια τεχνολογία που επιτρέπει την δυναμική ανανέωση μέρους ή ολόκληρης της ιστοσελίδας, χωρίς να χρειάζεται να την ανανεώσει ο χρήστης, αλλά μέσω των λεγόμενων ασύγχρονων requests. Ο πελάτης (client) ανακτά τα δεδομένα από τον εξυπηρετητή (server) στο παρασκήνιο, χωρίς να είναι εμφανής στον χρήστη αυτή η λειτουργία. Η τεχνολογία αυτή συνετέλεσε τα μέγιστα στην ανάπτυξη δυναμικών, υψηλής ποιότητας ιστοσελίδων. Επειδή συχνά παρεξηγείται ο όρος, να σημειωθεί πως το AJAX είναι απλά μια τεχνολογία, και όχι κάποια γλώσσα προγραμματισμού. Συγκεκριμένα τα λεγόμενα AJAX requests γράφονται σε JavaScript, και συνεπώς εκτελούνται στον περιηγητή του χρήστη. Στην παρούσα Web εφαρμογή η τεχνολογία αυτή χρησιμοποιήθηκε έτσι ώστε να υπάρχει δυναμική ανανέωση των περιεχομένων των πινάκων της βάσης δεδομένων, και συνεπώς να μην είναι υποχρεωμένος ο χρήστης να ανανεώνει συνεχώς την σελίδα. Ακόμη, ανανεώνεται δυναμικά το κεντρικό Master Page της ιστοσελίδας, όπου εμφανίζονται όλες οι ειδοποιήσεις (notifications) προς τον χρήστη. Η jQuery είναι ένα σύνολο από βιβλιοθήκες της γλώσσας JavaScript, οι οποίες ενσωματώνουν λειτουργίες που είναι πολύ πιθανό να θέλει κάποιος προγραμματιστής Web εφαρμογών να συμπεριλάβει στην ιστοσελίδα του. Είναι γραμμένες με αποδοτικό τρόπο, ευανάγνωστες, συντηρούνται και ανανεώνονται συνεχώς από την πολύ ενεργή κοινότητα που την αναπτύσσει. Ο σκοπός της είναι να κάνει την πλοήγηση στις ιστοσελίδες πιο ευχάριστη και διαδραστική γενικά να παρέχει μία ευχάριστη εμπειρία στον χρήστη μέσα από αυτήν. Ακόμη, παρέχει την δυνατότητα τροποποίησης των html κόμβων της ιστοσελίδας με εύκολο τρόπο, ενσωματώνοντας συντομογραφίες, συναρτήσεις και interfaces. Στην παρούσα Web εφαρμογή η jQuery χρησιμοποιήθηκε για την εναλλαγή και παρουσίαση εικόνων στην αρχική σελίδα, καθώς επίσης και για την επαλήθευση (validation) των δεδομένων που εισάγει ο χρήστης. Η επαλήθευση αυτή είναι κομβικής σημασίας, αφού αφ ενός απαγορεύουμε την έλευση λανθασμένων δεδομένων στον Server (ακόμα και όταν κάνει submit ο χρήστης τα δεδομένα δεν αποστέλλονται αν είναι λανθασμένα) , και εμφανίζεται κατάλληλο ενημερωτικό μήνυμα στον χρήστη ώστε να εισάγει σωστά δεδομένα, και αφ ετέρου δεν περιμένει ο χρήστης να αποσταλούν τα δεδομένα στον Server για να γίνει το validation, και μετά να απαντήσει ο Server με κάποιο μήνυμα λάθους. Βλέπουμε δηλαδή ότι η προσέγγιση αυτή είναι πολύ πιο αποδοτική. [18] [19]

3.2.2.6. CSS

Η CSS (Cascading Style Sheets) είναι μια γλώσσα που επιτρέπει την εύκολη μορφοποίηση του περιεχομένου (της HTML δηλαδή) των ιστοσελίδων. Η γλώσσα αυτή αναπτύχθηκε από την διεθνή κοινότητα W3C (World Wide Web Consortium) , και σε σύντομο χρονικό διάστημα προτυτοποιήθηκε και υποστηρίζεται από όλους τους μοντέρνους περιηγητές. Η ιδέα βάσει της οποίας αναπτύχθηκε αυτή η γλώσσα, είναι ο διαχωρισμός του περιεχομένου των ιστοσελίδων από την εμφάνισή τους. [20] Ο διαχωρισμός αυτός επιτρέπει την συγγραφή κώδικα ο οποίος καθορίζει την εμφάνιση των ιστοσελίδων ξεχωριστά από το περιεχόμενό τους, καθιστώντας δυνατή την παράλληλη ανάπτυξη και των δύο από διαφορετικούς προγραμματιστές. Ακόμη είναι δυνατή η συγγραφή των λεγόμενων templates, που είναι κώδικας CSS σε κάποιο αρχείο, και η χρήση τους σε διαφορετικές ιστοσελίδες. Στην παρούσα εφαρμογή χρησιμοποιήθηκε η CSS, μέσω του αρχείου Site.css έτσι ώστε η μορφοποίηση που επιβάλλει να υιοθετείται από όλα τα Views της εφαρμογής, και να κάνει πλοήγηση στον ιστότοπο ευχάριστη.

3.2.2.7. JavaScript

Στην αρχή της ιστορίας του Web, υπήρχε μόνο η HTML και το CGI (Common Gateway Interface). Το πρόβλημα με την HTML είναι η στασιμότητά της με την έννοια ότι με το που θα εκτυπωθεί στον browser του χρήστη δε μπορεί να μεταβληθεί από μόνη της. Για να επιτευχθεί κάτι τέτοιο πρέπει να υπάρξει επικοινωνία με τον server ώστε μέσω μία script γλώσσας προγραμματισμού στην πλευρά του server να αποσταλούν εκ νέου δεδομένα στην πλευρά του πελάτη. Κάτι τέτοιο απαιτούνταν ακόμα και για τις πιο απλές διαδικασίες όπως η απόκρυψη ή η εμφάνιση ενός στοιχείου του εγγράφου κάτι όμως που αποτελεί τροχοπέδη για την ανάπτυξη του διαδικτυακού προγραμματισμού αλλά και για τη λειτουργία των εξυπηρετητών αφού αυξάνεται σημαντικά ο υπολογιστικός φόρτος. Τη λύση σε αυτό το πρόβλημα έδωσε η γλώσσα JavaScript η οποία είναι μία γλώσσα προγραμματισμού που εκτελείται στην πλευρά του πελάτη .

Η JavaScript αναπτύχθηκε από την εταιρεία Netscape σε συνεργασία με την Sun Microsystems και πρωτοδημοσιεύτηκε το 1995 με το όνομα LiveScript. Αργότερα πήρε το τωρινό της όνομα εκμεταλλευόμενη τη δυναμική της πανίσχυρης γλώσσας προγραμματισμού Java αν και δεν έχει σχέση με αυτήν. [21]

Αποτελεί μία διερμηνευμένη (interpreted) γλώσσα προγραμματισμού με ιδιότητες αντικειμενοστραφούς γλώσσας προγραμματισμού, χωρίς όμως να μπορεί να χαρακτηριστεί ως πλήρης αντικειμενοστραφής αν και κάτι τέτοιο λέγεται ότι προβλέπεται στην επόμενη έκδοσή της.

Η γλώσσα JavaScript χρησιμοποιείται κυρίως (αλλά όχι μόνο) για την επίτευξη των παρακάτω στόχων:

- **Λιγότερος υπολογιστικός φόρτος στην πλευρά του server:** Ο έλεγχος και η επικύρωση των δεδομένων τα οποία εισάγονται από τους χρήστες γίνεται από τη μεριά του browser και κάθε δεδομένο το οποίο δεν είναι έγκυρο απορρίπτεται χωρίς να αποστέλλεται στον server. Παρόλα αυτά, είναι επιβεβλημένο να γίνει έλεγχος και στην πλευρά του server καθώς στους browser υπάρχει επιλογή απενεργοποίησης της JavaScript ενώ σε μερικούς δεν είναι ενσωματωμένη.

- **Αμεσότητα αλληλεπίδρασης με τους χρήστες:** Με την χρήση της JavaScript για τον έλεγχο των δεδομένων μειώνονται οι χρόνοι αναμονής του χρηστών αφού δεν υπάρχουν τα διαστήματα αναμονής επαναφόρτωσης της σελίδας σε περίπτωσης λάθους.

- **Αυξημένη χρηστικότητα και φιλικότητα προς τον χρήστη:** Κάτι τέτοιο επιτυγχάνεται επιτρέποντας στον χρήστη την αλληλεπίδραση με το γραφικό περιβάλλον χωρίς την επαναφόρτωση της σελίδας. Τέτοιο παράδειγμα είναι τα πτυσσόμενα μενού.

- **Αυξημένη δυνατότητα αλληλεπίδρασης:** Χαρακτηριστικό παράδειγμα είναι τα γεγονότα όπου κάθε γεγονός πυροδοτεί μία σειρά από άλλα γεγονότα.

- **Βελτιωμένα γραφικά περιβάλλοντα:** Χρησιμοποιώντας την JavaScript μπορούν να συμπεριληφθούν αντικείμενα με λειτουργίες drag-and-drop καθώς και plug-in, όπως είναι το Flash. Μάλιστα, αν και το Flash είχε γνωρίσει περιόδους άνθισης λόγω των πλεονεκτημάτων γραφική καλαισθησίας που παρείχε, σήμερα με την επικράτηση της JQuery επιτυγχάνονται τα ίδια, αν όχι καλύτερα, αποτελέσματα χωρίς την ανάγκη ύπαρξης plug-in στην πλευρά του πελάτη αλλά και χωρίς τους μεγάλους χρόνους αναμονής φόρτωσης των αρχείων Flash.

- **Ελαφρότερα περιβάλλοντα και μικρότεροι χρόνοι αναμονής**: Αντί της απαίτησης φόρτωσης ενός μεγάλου αρχείου Java applet ή ενός Flash movie, τα προγράμματα γραμμένα σε JavaScript είναι μικρά σε μέγεθος και αποθηκεύονται στη μνήμη cache του browser μόλις κατέβουν κάτι που ελαχιστοποιεί τους χρόνους αναμονής. Μάλιστα, λόγω του γεγονότος ότι πολλές ιστοσελίδες πλέον χρησιμοποιούν και φορτώνουν τη βιβλιοθήκη JQuery από κοινούς server, όπως αυτοί της Google, τα αρχεία αυτά υπάρχουν ήδη στην μνήμη του browser οπότε δε χρειάζεται να φορτωθούν ξανά. [22]

3.2.2.8. Entity Framework

Το Entity Framework επιτρέπει στους προγραμματιστές να προγραμματίζουν με βάση σχεσιακές βάσεις δεδομένων, χρησιμοποιώντας αντικείμενα .NET και Language

Integrated Query (LINQ). Διαθέτει πολλές νέες δυνατότητες, όπως παράβλεψη επιμονής και υποστήριξη POCO, συσχετισμούς εξωτερικών κλειδιών, αργή φόρτωση, υποστήριξη προγραμματισμού βάσει δοκιμής, λειτουργίες στο μοντέλο και νέους τελεστές LINQ. Οι επιπλέον δυνατότητες περιλαμβάνουν καλύτερη υποστήριξη n-tier με οντότητες αυτόματης παρακολούθησης, δημιουργία κώδικα με δυνατότητα προσαρμογής με χρήση προτύπων T4, πρώτη ανάπτυξη μοντέλου, βελτιωμένη εμπειρία σχεδίασης, καλύτερη απόδοση και πλουραλισμό συνόλων οντοτήτων.
Οι Υπηρεσίες δεδομένων WCF αποτελούν στοιχείο του .NET Framework, το οποίο σας επιτρέπει να δημιουργήσετε υπηρεσίες και εφαρμογές που βασίζονται σε REST και χρησιμοποιούν το πρωτόκολλο Open Data (OData) για έκθεση και χρήση δεδομένων στο Web. Οι Υπηρεσίες δεδομένων WCF διαθέτουν πολλές νέες δυνατότητες, όπως βελτιωμένη υποστήριξη BLOB, σύνδεση δεδομένων, καταμέτρηση γραμμών, προσαρμογή τροφοδοσίας, προβολές και βελτιώσεις στη διοχέτευση αιτήσεων. Με την ενσωματωμένη ενοποίηση με το Microsoft Office 2010, είναι πλέον δυνατή η έκθεση δεδομένων του Microsoft Office SharePoint Server ως τροφοδοσία OData και η πρόσβαση σε αυτή την τροφοδοσία δεδομένων με χρήση της βιβλιοθήκης-πελάτη των Υπηρεσιών δεδομένων WCF. [23]

3.2.2.9. SQL Server

Η Βάση Δεδομένων του πληροφοριακού συστήματος αναπτύχθηκε με κάποια έκδοση του Microsoft SQL Server. Ο Microsoft SQL Server είναι ένα σχεσιακό μοντέλο διαχείρισης βάσεων δεδομένων (RDBMS) του οπίου η υλοποίηση της γλώσσας SQL ονομάζεται T-SQL. [24] Οι λόγοι που χρησιμοποιούμε τα συστήματα διαχείρισης βάσεων δεδομένων είναι επειδή προσφέρουν:

- **Ανεξαρτησία δεδομένων.** Τα δεδομένα είναι ανεξάρτητα από την εφαρμογή που τα χρησιμοποιεί. Μπορούν να δημιουργηθούν εφαρμογές οι οποίες να προσαρμόζονται με σχετική ευκολία σε διαφορετικές πηγές δεδομένων.

- **Ακεραιότητα δεδομένων.** Εξασφαλίζεται ότι δεν υπάρχουν ασυνέπειες μεταξύ εγγραφών που αντιπροσωπεύουν την ίδια πληροφορία, τα δεδομένα 40 συμμορφώνονται με τους περιορισμούς που επιβάλλονται από το σχήμα της βάσης, και οι τιμές των μεταβλητών συμμορφώνονται με το πεδίο τιμών και τον τύπο δεδομένων που έχει οριστεί.

- **Ταυτόχρονη προσπέλαση και κεντρικό έλεγχο.** Εξασφαλίζει την ταυτόχρονη προσπέλαση των δεδομένων και απαλείφει τα λεγόμενα αδιέξοδα (deadlocks).

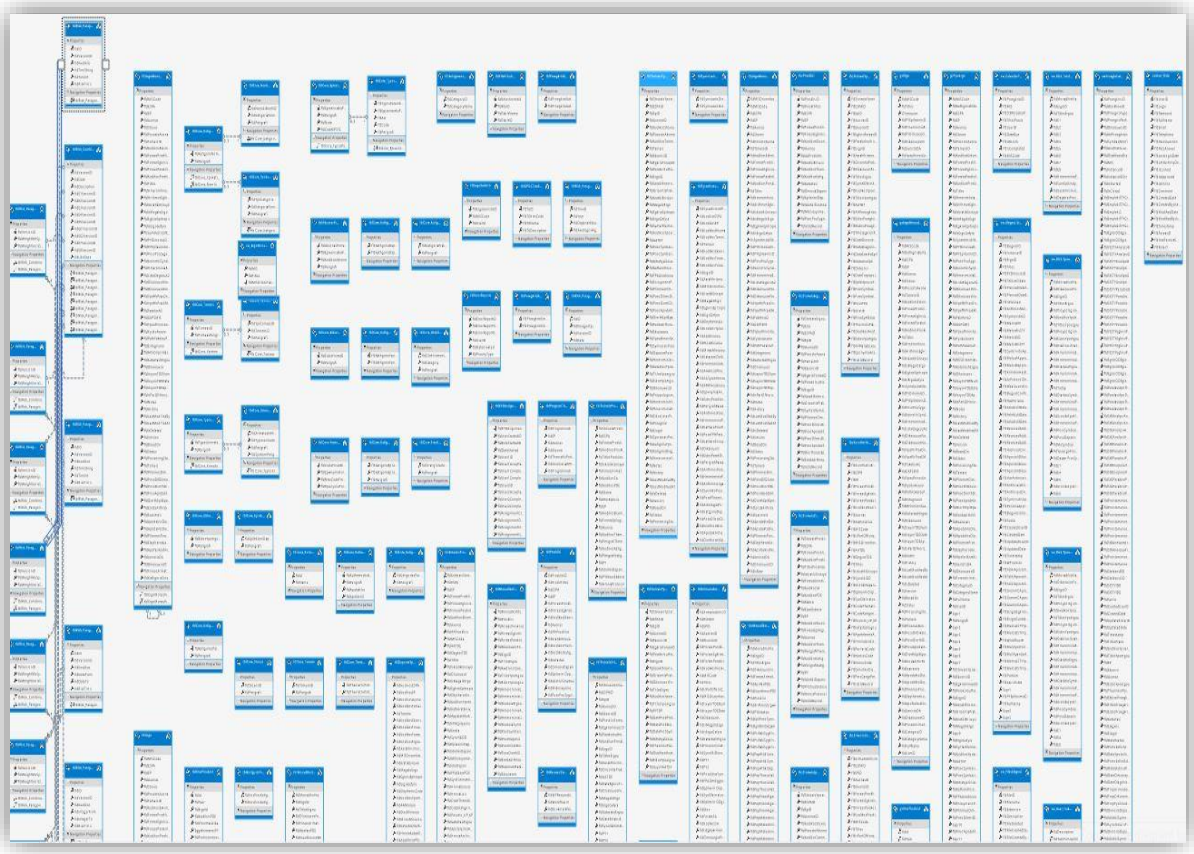
- **Αποδοτική αποθήκευση και προσπέλαση των δεδομένων.** Χρησιμοποιεί αποδοτικά τον διαθέσιμο χώρο του υπολογιστικού συστήματος, και εισάγει

δομές (ευρετήρια, hash tables κλπ) για να επιτυγχάνεται γρήγορη πρόσβαση στα δεδομένα

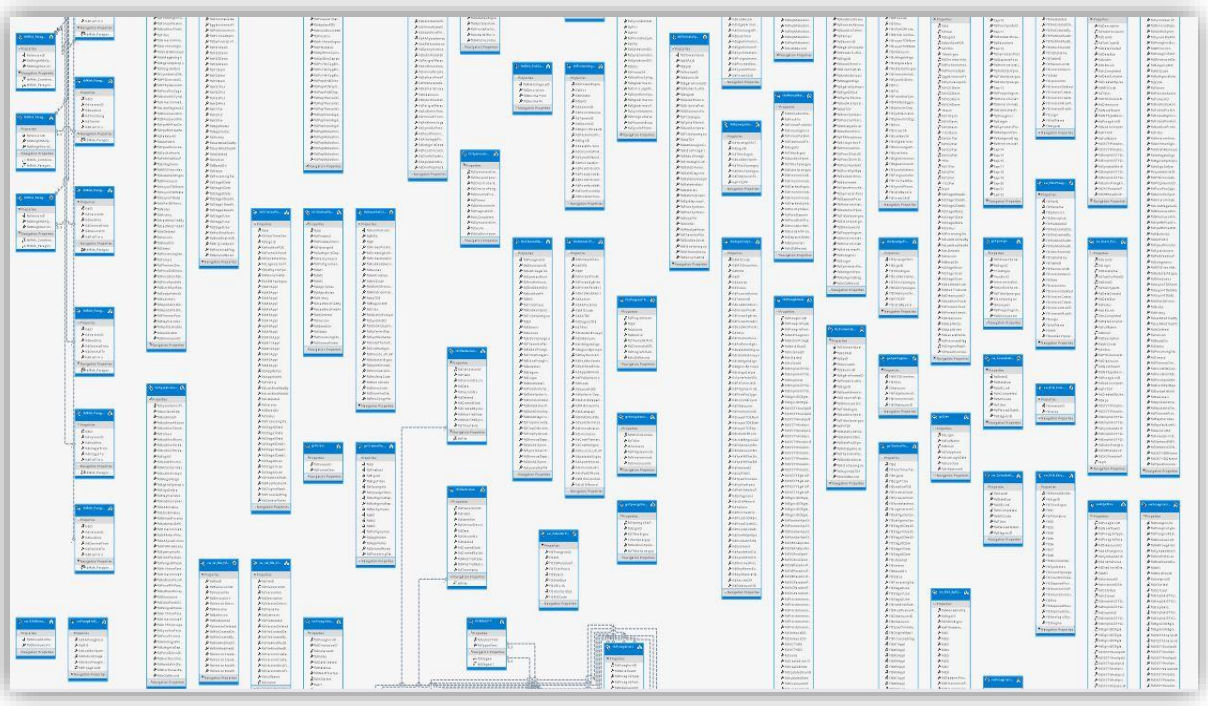
- **Αποφυγή πλεονάζουσας πληροφορίας (Non- Redundancy) των δεδομένων.** Το σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων επιχειρεί να εξασφαλίσει ότι δεν αποθηκεύεται πληροφορία η οποία είτε έχει αποθηκευτεί ήδη, είτε μπορεί να εξαχθεί από άλλες, ήδη αποθηκευμένες πληροφορίες. Η δημιουργία και τροποποίηση της βάσης δεδομένων έγινε με την βοήθεια του SQL Server Management Studio. Το SQL Server Management studio είναι μία εφαρμογή με γραφικό περιβάλλον (GUI) και περιλαμβάνει όλα τα απαραίτητα εργαλεία για την επεξεργασία βάσεων δεδομένων, όπως script editors, object explorer και παρέχει αρκετές ευκολίες στον προγραμματιστή, όπως επισήμανση λαθών, Intellisence, αρκετό έτοιμο κώδικα κλπ. [25]

3.2.2.10. Σχεσιακό Διάγραμμα της Βάσης Δεδομένων του Πληροφοριακού Συστήματος

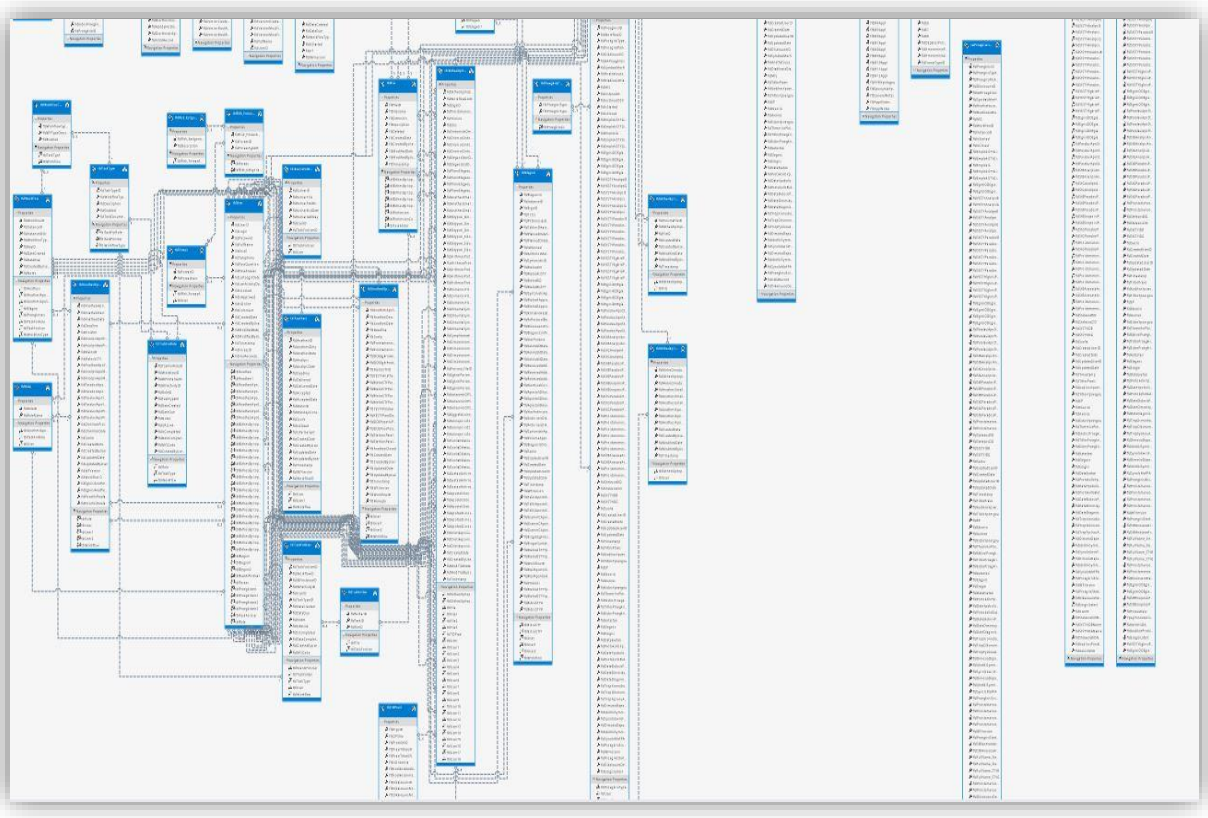
Σε αυτή την ενότητα απεικονίζεται για λόγους πληρότητας, το σχήμα της βάσης δεδομένων του πληροφοριακού συστήματος με το οποίο θα ασχοληθούμε αναλυτικά σε επόμενο κεφάλαιο, όπου και θα γίνει η πλήρης ανάλυσή του.



Εικόνα 9: Μέρος του σχεσιακού διαγράμματος της βάσης δεδομένων



Εικόνα 10: Μέρος ΙΙ του σχεσιακού διαγράμματος της βάσης δεδομένων



Εικόνα 11:Μέρος ΙΙΙ του σχεσιακού διαγράμματος της βάσης δεδομένων

4 Βελτιώσεις-Αλλαγές-Αποτελέσματα

4.1 Σκοπός των αλλαγών στο λογισμικό του Πληροφοριακού Συστήματος

Μία ιδιότητα του λογισμικού σε ένα πραγματικό περιβάλλον είναι η ανάγκη του να εξελίσσεται συνεχώς. Καθώς αναπτύσσεται, αλλάζει και προσαρμόζεται σε νέες απαιτήσεις, ο κώδικας καθώς και ο σχεδιασμός του γίνονται όλο και πιο περίπλοκοι και αποκλίνουν συνεχώς από τον αρχικό σχεδιασμό τους. Εξαιτίας αυτού του γεγονότος, το μεγαλύτερο μέρος του κόστους ανάπτυξής του αφιερώνεται σε διαδικασίες συντήρησης.

Καλύτερες μέθοδοι ανάπτυξης λογισμικού και εργαλεία δεν είναι δυνατό να λύσουν το συγκεκριμένο πρόβλημα, γιατί οι αυξανόμενες δυνατότητές τους χρησιμοποιούνται κατά κύριο λόγο για υλοποίηση νέων απαιτήσεων μέσα στο ίδιο χρονικό διάστημα, κάνοντας έτσι το λογισμικό για ακόμα μία φορά πιο περίπλοκο.

Για να αντιμετωπιστεί αυτή η αυξανόμενη πολυπλοκότητα, είναι επείγουσα η ανάγκη για τεχνικές μείωσης της πολυπλοκότητας του λογισμικού και παράλληλης αύξησης της εσωτερικής ποιότητάς του.

Η χρησιμοποίηση μιας μεθοδολογίας για το σχεδιασμό και ανάπτυξη μιας εφαρμογής ενισχύει την αποτελεσματικότητα, αλλά δεν εγγυάται τη βελτιστοποίηση στη όλη διαδικασία. Επιπλέον, οι περισσότερες εφαρμογές αναπτύσσονται από μεγάλες ομάδες ατόμων, πράγμα το οποίο οδηγεί σε προβλήματα επικοινωνίας στη διαδικασία σχεδιασμού/ανάπτυξης και έτσι παράγονται προϊόντα με υψηλό αριθμό προβλημάτων ευχρηστίας, αξιοπιστίας, απόδοσης, και άλλων χαρακτηριστικών ποιότητας. Στις περισσότερες από τις περιπτώσεις λόγω της έλλειψης χρόνου, οι προγραμματιστές επαναχρησιμοποιούν την προηγούμενη γνώση και εμπειρία τους, χωρίς να προσπαθήσουν να την προσαρμόσουν πλήρως στις απαιτήσεις του τρέχοντος έργου. Εξασκούν έτσι με “ακατάλληλο” τρόπο την έννοια της επαναχρησιμοποίησης (reuse). Είναι προφανές ότι στις περισσότερες περιπτώσεις υπάρχει ανάγκη για αναδόμηση (restructuring) / αναδιάταξη (refactoring) των εφαρμογών, ακόμη και αρκετό χρονικό διάστημα μετά την υλοποίησή τους.

Οι αλλαγές που πραγματοποιήθηκαν στο Πληροφοριακό Σύστημα Διαχείρισης Χρηματοδοτήσεων μπορούν να διαχωριστούν σε δυο βασικές κατηγορίες: αλλαγές για τον χρήστη του πληροφοριακού συστήματος και αλλαγές για την ομάδα προγραμματιστών, και στοχεύουν αντίστοιχα σε:

Από πλευράς του χρήστη:

- Βελτίωση απόδοσης του πληροφοριακού συστήματος.
- Διόρθωση λαθών κώδικα (bugs) που οδηγούν σε errors.

- Ανάπτυξη νέων δυνατοτήτων.
- Βελτίωση Οπτικής Εικόνας-Ποιότητας.

Από πλευράς προγραμματιστή:

- Αναδιαμόρφωση του κώδικα ώστε να είναι εύκολα διαχειρίσιμος και κατανοητός από οποιονδήποτε προγραμματιστή.

Στις επόμενες ενότητες παρουσιάζονται αναλυτικά οι αλλαγές που πραγματοποιήθηκαν.

4.2 Βάση Δεδομένων (Database)

4.2.1 Προβλήματα κακής σχεδίασης μιας Βάσης Δεδομένων

Είναι ευρέως αποδεκτό ότι οι βάσεις δεδομένων αποτελούν το σημαντικότερο κομμάτι κάθε πληροφοριακού συστήματος. Η διαδικασία που ακολουθείται για την υλοποίησης μιας βάσης δεδομένων είναι αντίστοιχη με τη διαδικασία υλοποίησης του πληροφοριακού συστήματος: ανάλυση απαιτήσεων, σχεδιασμός, υλοποίηση. Ωστόσο, μια κακή σχεδίαση της βάσης δεδομένων οδηγεί το πληροφοριακό σύστημα σε υπό-απόδοση αλλά και δυσκολία στην επεξεργασία των δεδομένων.

Αναλυτικότερα, τα προβλήματα μπορούν να κατηγοριοποιηθούν σε:

- **Πλεονασμός των δεδομένων (data redundancy).** Υπάρχει η περίπτωση να έχουμε επανάληψη των ίδιων δεδομένων σε διαφορετικούς πίνακες (tables).
- **Ασυνέπεια των δεδομένων (data inconsistency).** Αυτό μπορεί να συμβεί όταν υπάρχουν τα ίδια στοιχεία μιας εγγραφής στη βάση σε διαφορετικά σημεία και χρειασθεί να γίνει κάποια αλλαγή στην εγγραφή, οπότε είναι πολύ πιθανό να γίνει η διόρθωση μόνο στο ένα σημείο και όχι και στο άλλο.
- **Αδυναμία μερισμού δεδομένων (data sharing).** Μερισμός δεδομένων είναι η δυνατότητα για κοινή χρήση των στοιχείων κάποιων πινάκων (tables). Η αδυναμία μερισμού δεδομένων δημιουργεί καθυστέρηση στη λήψη αποφάσεων και στην εξυπηρέτηση των χρηστών.

- **Αδυναμία προτυποποίησης.** Έχει να κάνει με την ανομοιομορφία και με την διαφορετική αναπαράσταση και οργάνωση των δεδομένων στα αρχεία των εφαρμογών. Η αδυναμία αυτή δημιουργεί προβλήματα προσαρμογής των χρηστών καθώς και προβλήματα στην ανταλλαγή δεδομένων μεταξύ διαφορετικών συστημάτων.

4.2.2 Έλεγχος χρήσης πινάκων σε Database Π.Σ.Δ.Χ

Στο τέλος του 3ου κεφαλαίου, απεικονίσαμε το σχεσιακό διάγραμμα της βάσης δεδομένων του Πληροφοριακού Συστήματος Διαχείρισης Χρηματοδοτήσεων. Είναι φανερό ότι πρόκειται για ένα εξαιρετικά μεγάλο μοντέλο βάσης που αποτελείται από 140 πίνακες, δημιουργώντας την απορία κατά πόσο χρησιμοποιούνται στην εφαρμογή. **Με δεδομένο ότι το project στήθηκε με EF Database First MVC, θα διερευνηθεί η υπόθεση ότι ένα μεγάλο μέρος της database είχε σχεδιαστεί κατά τη φάση της ανάπτυξης αλλά εν τέλει δεν χρησιμοποιήθηκε.**

Η διαδικασία που ακολουθήθηκε για την εξέταση της database περιλαμβάνει τα ακόλουθα βήματα:

1) Ο πρώτος έλεγχος που πραγματοποιήθηκε στη database αφορούσε το μέγεθος της σε πλήθος εγγραφών και τι όγκο δεδομένων είχε κάθε πίνακας. Μια πρώτη προσέγγιση θα μπορούσε να είναι ότι πίνακες με μηδενικό όγκο δεδομένων μετά από χρόνια λειτουργίας του πληροφοριακού συστήματος, είναι πιθανό να μην χρησιμοποιούνται από την εφαρμογή, χωρίς ωστόσο αυτή η συνθήκη να είναι ικανή για την εξαγωγή συμπεράσματος. Για τον έλεγχο αυτό, χρησιμοποιήθηκε το SQL Query του Παραρτήματος 1 με τα αποτελέσματα να επισυνάπτονται εμφανίζονται στον Πίνακα 1 που ακολουθεί.

Table Name	Number OfRows	SizeinKB
tblStoixeiaErgwn	65535	11592
tbzStoixeiaErgwn	65535	11592
tbzSysxetismoiPliromon	47009	32584
tblSysxetismoiPliromon	47009	32456
tblStoixeiaYpoergwn	25770	10376
tblForeas	18914	1928
tblRisk_ForeasAndType	18821	648
tblEntaxeisStoixeiaErgou	16792	10504
tblXorothetisiErgou	10766	5000
tbzXorothetisiErgou	10462	4808
tblErgo	8967	6728
tbzEntaxeisStoixeiaErgou	8868	5576
tbzStoixeiaYpoergwn	8859	3272
tbzErgoAnalytical	8802	8264
tblErgoMonadikotita	7890	5576

tbzErgoMonadikotita	7890	2056
tblStoixeiaYpoergwnTDP	4516	3336
tbzStoixeiaYpoergwnTDP	4199	3016
tblTaskForUser	2624	904
tblProegkriseis	2172	3144
tbzProegkriseis	2145	2952
tblXrimatodotisiErgou	2069	1608
tblTaskForRole	1970	712
tblStoixeiaProsklisis	1829	1608
tblStoixeiaYpoergwnTDY	1742	2248
tbzStoixeiaYpoergwnTDY	1741	2264
tbzStoixeiaProsklisis	1582	1160
tblMonadikotitaYpoergou	1372	1352
tblYpoergaStoTDP	1355	1992
tblFile	1144	1232
tblFileAttachments	1056	272
tblCore_ForeasDimou	917	136
tbzXrimatodotisiErgou	896	648
tblStoixeiaXrimatodotisisErgwn	871	648
tblWorkflow	854	200
tblFileVersion	850	248
tblMonadikotitaErgou	754	584
tblProsklisi	664	136
tbzProsklisi	612	328
tblUser	442	360
tblCore_KatigoriaPraksewnEspa	439	136
tblUserRole	438	40
InstancesTable	385	1776
tblFileVersionCopy	352	299376
tblResumableBookmark	311	128
tblCore_EpileksimiDapani	216	40
tblAksonasProteraiotitas	107	40
tblProgramThemEkx	80	32
tblTaskType	80	16
tbzProgramThemEkx	80	16
tblElegxoi	54	112
tblCore_NomikiMorfi	51	16
tblCore_KathestosKratikwnEnisxysewn	33	40
tblRole	32	32
tblCore_TyposParembasisEKT	31	16
tblRisk_Paragontas_E1_DiadikasiaAnathesis_Times	27	16
tblRisk_Paragontas_E7_KatigoriaTelikouDikaiouxou_Times	27	16
tblCore_OlokliromeniKateythGrammiAnaptApasx	24	16
tblRisk_Paragontas_E4_EidosYpoergoy_Times	24	16
tblCore_EpixeirisiakoProgramma	20	16
tblCore_StratigikiKateuthyntiriaGrammiSynoxi	19	16
tblRisk_Paragontas_E2_ProypologismosPraxis_Times	18	16
tblRisk_Paragontas_E3_PolyplokotitaPlithosYpoergwn_Times	18	16
tblRisk_Paragontas_E8_YpoergaMeSymbasi_Times	18	16
tblTriminoPinakas2A	16	16

tblRisk_Paragontas_E5_DapanesProsSymbaseisTimes	15	16
tblCore_Ypotomeas	12	16
tblRisk_Paragontas_E6_XronodiagrammaYpoergaMeSymbasi_Times	12	16
tblRisk_Paragontas_E6_XronodiagrammaYpoergaXorisSymbasi_Times	12	16
tblCore_EidosYpoergou	11	16
tblCore_KatastasiYpoergou	11	16
tblCore_XrimatodotikoMeso	11	16
tblRisk_ParagontasCurrentVersion	11	16
tblErrorReports	10	16
tblCore_KatigoriaNomPros	9	16
tblCore_KatigoriaParastatikwnAnadoxou	9	16
tblCore_KatigoriaParastatikwnPliromisForea	9	16
tblRisk_KatigoriaForea	9	16
tblTriminoPinakas2	8	40
tblCore_EidosiForea	7	16
tblCore_KathgoriaDapanwn	6	16
tblRisk_RiskCategories	6	16
tblCore_TyposXrimatodotisis	5	16
tblCore_YpokatigoriaNomPros	5	16
tblWorkflowType	5	16
tblAnathesi	4	32
tblCore_KatigoriaEpileksimonDapanwn	4	16
tblCore_Tomeas	4	16
tblCore_KathgoriaYpologou	3	16
tblCore_Stoxos	3	16
tblCore_Tameio	3	16
tblProegkrisiType	3	16
tblRisk_CombinationVersions	3	16
tblRisk_Paragontas_E1_DiadikasiaAnathesis	3	16
tblRisk_Paragontas_E2_ProypologismosPraxis	3	16
tblRisk_Paragontas_E3_PolyplokotitaPlithosYpoergwn	3	16
tblRisk_Paragontas_E4_EidosYpoergoy	3	16
tblRisk_Paragontas_E5_DapanesProsSymbaseis	3	16
tblRisk_Paragontas_E6_XronodiagrammaYpoergaMeSymbasi	3	16
tblRisk_Paragontas_E6_XronodiagrammaYpoergaXorisSymbasi	3	16
tblRisk_Paragontas_E7_KatigoriaTelikouDikaiouxou	3	16
tblRisk_Paragontas_E8_YpoergaMeSymbasi	3	16
tblSymvaseisOrosima	3	16
tblSymvaseisOrosimaTypos	3	16
tblAnathesiApoStelexos	2	32
tblEidosSTY	2	16
LockOwnersTable	1	80
DefinitionIdentityTable	1	48
IdentityOwnerTable	1	16
SqlWorkflowInstanceStoreVersionTable	1	16
tblAnathesiApoProistEYD	1	16
tblRisk_Paragontas_EksagwgiDeigmatos_Times	1	16
DBProperties	0	0
InstanceMetadataChangesTable	0	0
InstancePromotedPropertiesTable	0	0

KeysTable	0	0
RunnableInstancesTable	0	0
ServiceDeploymentsTable	0	0
tblCore_ThematikiProteraiotita	0	0
tblDapanesYpoergonSysxetiseis	0	0
tblEkthesiEpitopiasEpalitheysis	0	0
tblEkthesiEpitopiasEpalitheysis_MelosOmadasElegxou	0	0
tblEkthesiEpitopiasEpalitheysis_PorismaFile	0	0
tblEntagmenaKathgoria	0	0
tblErgoXeiristis	0	0
tblEventsForUser	0	0
tblEYDAssignments	0	0
tblImportedRaw_Foreas	0	0
tblImportedRaw_KatigoriaPraxis	0	0
tblKatigoriaPraxisESPA	0	0
tblOPDCTreeElement	0	0
tblProegkrisiEisigisi	0	0
tblProegkrisiKatastasi	0	0
tblTaskFolder	0	0
tblTDPraxi	0	0
tbzCleanedImportProegkriseis	0	0
tbzMonadikotitaErgou	0	0
tbzMonadikotitaYpoergou	0	0
tbzYpoergaStoTDP	0	0

Πίνακας 1: Αποτελέσματα εφαρμογής SQL Query για μέγεθος βάσης

2) Στη συνέχεια, με βάση το Εγχειρίδιο Χρήσης του Πληροφοριακού Συστήματος, πραγματοποιήθηκε ένας πλήρης κύκλος λειτουργίας της εφαρμογής με όλους τους πιθανούς ρόλους-εργασίες, έτσι ώστε να γίνει εφαρμογή του SQL Query του Παραρτήματος 2. Το συγκεκριμένο SQL Query, αποτυπώνει πότε έγινε τελευταία φορά η πρόσβαση σε κάθε πίνακα της βάσης δεδομένων. Ως λογικό επακόλουθο, οι πίνακες στους οποίους δεν έγινε προσπέλαση, δεν χρησιμοποιούνται άμεσα από την εφαρμογή.

TableName	LastSelect
DBProperties	NULL
DefinitionIdentityTable	NULL
IdentityOwnerTable	NULL
InstanceMetadataChangesTable	NULL
InstancePromotedPropertiesTable	NULL
InstancesTable	NULL
KeysTable	NULL
LockOwnersTable	NULL
RunnableInstancesTable	NULL
ServiceDeploymentsTable	NULL

SqlWorkflowInstanceStoreVersionTable	NULL
tblAksonasProteraiotitas	NULL
tblAnathesi	2016-05-20 20:03:01.423
tblAnathesiApoProistEYD	2016-05-20 20:02:23.545
tblAnathesiApoStelexos	2016-05-20 20:01:25.877
tblCore_EidosiForea	2016-05-20 20:01:10.004
tblCore_EidosYpoergou	2016-05-20 20:01:26.537
tblCore_EpileksimiDapani	NULL
tblCore_EpixeirisiakoProgramma	NULL
tblCore_ForeasDimou	NULL
tblCore_KatastasiYpoergou	2016-05-20 20:01:27.215
tblCore_KathestosKratikwnEnisxysewn	NULL
tblCore_KathgoriaDapanwn	2016-05-20 20:07:12.325
tblCore_KathgoriaYpologou	NULL
tblCore_KatigoriaEpileksimonDapanwn	NULL
tblCore_KatigoriaNomPros	NULL
tblCore_KatigoriaParastatikwnAnadoxou	NULL
tblCore_KatigoriaParastatikwnPliromisForea	NULL
tblCore_KatigoriaPraksewnEspa	NULL
tblCore_NomikiMorfi	NULL
tblCore_OloklиromeniKateythGrammiAnaptApasx	NULL
tblCore_Stoxos	NULL
tblCore_StratigikiKateuthyntiriaGrammiSynoxi	NULL
tblCore_Tameio	NULL
tblCore_ThematikiProteraiotita	NULL
tblCore_Tomeas	NULL
tblCore_TyposParembasisEKT	NULL
tblCore_TyposXrimatodotisis	NULL
tblCore_XrimatodotikoMeso	NULL
tblCore_YpokatigoriaNomPros	NULL
tblCore_Ypotomeas	NULL
tblDapanesYpoergonSysxetiseis	NULL
tblEidosSTY	2016-05-20 19:17:28.250
tblEkthesiEpitopiasEpalitheysis	2016-05-20 20:13:12.605
tblEkthesiEpitopiasEpalitheysis_MelosOmadasElegxou	NULL
tblEkthesiEpitopiasEpalitheysis_PorismaFile	NULL
tblElegxoi	2016-05-20 20:02:45.437
tblEntagmenaKathgoria	2016-05-20 19:57:04.837
tblEntaxeisStoixeiaErgou	NULL
tblErgo	2016-05-20 19:50:41.513
tblErgoMonadikotita	2016-05-20 20:02:54.637
tblErgoXeiristis	NULL
tblErrorReports	NULL
tblEtosPinakas1	2016-05-20 19:24:22.425
tblEventsForUser	2016-05-20 19:19:02.430
tblEYDAssignments	NULL
tblFile	2016-05-20 19:43:09.673

tblFileAttachments	2016-05-20 20:03:00.870
tblFileVersion	2016-05-20 19:43:09.673
tblFileVersionCopy	NULL
tblForeas	2016-05-20 19:50:41.513
tblImportedRaw_Foreas	NULL
tblImportedRaw_KatigoriaPraxis	NULL
tblKatigoriaPraxisESPA	NULL
tblMonadikotitaErgou	NULL
tblMonadikotitaYpoergou	NULL
tblOPDCTreeElement	NULL
tblProegkriseis	2016-05-20 20:03:00.010
tblProegkrisiEisigisi	NULL
tblProegkrisiType	NULL
tblProgramThemEkx	NULL
tblProsklisi	NULL
tblResumableBookmark	2016-05-20 18:59:28.280
tblRisk_CombinationVersions	2016-05-20 18:57:24.432
tblRisk_ForeasAndType	NULL
tblRisk_KatigoriaForea	NULL
tblRisk_Paragontas_E1_DiadikasiaAnathesis	NULL
tblRisk_Paragontas_E1_DiadikasiaAnathesis_Times	2016-05-20 18:56:21.522
tblRisk_Paragontas_E2_ProypologismosPraxis	NULL
tblRisk_Paragontas_E2_ProypologismosPraxis_Times	NULL
tblRisk_Paragontas_E3_PolyplokotitaPlithosYpoergwn	NULL
tblRisk_Paragontas_E3_PolyplokotitaPlithosYpoergwn_Times	NULL
tblRisk_Paragontas_E4_EidosYpoergoy	NULL
tblRisk_Paragontas_E4_EidosYpoergoy_Times	NULL
tblRisk_Paragontas_E5_DapanesProsSymbaseis	NULL
tblRisk_Paragontas_E5_DapanesProsSymbaseisTimes	NULL
tblRisk_Paragontas_E6_XronodiagrammaYpoergaMeSymbasi	NULL
tblRisk_Paragontas_E6_XronodiagrammaYpoergaMeSymbasi_Times	NULL
tblRisk_Paragontas_E6_XronodiagrammaYpoergaXorisSymbasi	NULL
tblRisk_Paragontas_E6_XronodiagrammaYpoergaXorisSymbasi_Times	NULL
tblRisk_Paragontas_E7_KatigoriaTelikouDikaiouxou	NULL
tblRisk_Paragontas_E7_KatigoriaTelikouDikaiouxou_Times	NULL
tblRisk_Paragontas_E8_YpoergaMeSymbasi	NULL
tblRisk_Paragontas_E8_YpoergaMeSymbasi_Times	NULL
tblRole	2016-05-20 20:04:02.800
tblStoixeiaErgwn	NULL
tblStoixeiaProsklisis	NULL
tblStoixeiaXrimatodotisisErgwn	NULL
tblStoixeiaYpoergwn	2016-05-20 19:57:08.487
tblStoixeiaYpoergwnTDP	2016-05-20 20:02:54.637
tblStoixeiaYpoergwnTDY	2016-05-20 19:47:53.323
tblSymvaseisOrosima	2016-05-20 19:49:47.250
tblSymvaseisOrosimaTypos	2016-05-20 19:47:53.317
tblSysxetismoiPliromon	2016-05-20 19:32:10.250
tblTaskFolder	2016-05-20 19:19:02.430
tblTaskForRole	2016-05-20 20:02:54.637
tblTaskForUser	2016-05-20 20:02:54.437

tblTaskType	2016-05-20 20:02:54.637
tblTDPraxi	2016-05-20 20:23:34.507
tblTriminoPinakas2	2016-05-20 20:12:02.104
tblTriminoPinakas2A	2016-05-20 20:14:23.524
tblUser	2016-05-20 20:04:02.800
tblUserRole	2016-05-20 20:04:02.800
tblWorkflow	2016-05-20 19:53:39.987
tblWorkflowType	2016-05-20 20:02:54.637
tblMonadikotitaYpoergou	NULL
tblOPDCTreeElement	NULL
tblProegkriseis	2016-05-20 20:03:00.010
tblXorothetisiErgou	NULL
tblXrimatodotisiErgou	NULL
tblYpoergaStoTDP	2016-05-20 20:12:02.210
tbzCleanedImportProegkriseis	NULL
tbzEntaxeisStoixeiaErgou	NULL
tbzErgoAnalytical	NULL
tbzErgoMonadikotita	NULL
tbzMonadikotitaErgou	NULL
tbzMonadikotitaYpoergou	NULL
tbzProegkriseis	NULL
tbzProgramThemEkx	NULL
tbzProsklisi	NULL
tbzStoixeiaErgwn	NULL
tbzStoixeiaProsklisis	NULL
tbzStoixeiaYpoergwn	NULL
tbzStoixeiaYpoergwnTDP	NULL
tbzStoixeiaYpoergwnTDY	NULL
tbzSysxetismoiPliromon	NULL
tbzXorothetisiErgou	NULL
tbzXrimatodotisiErgou	NULL
tbzYpoergaStoTDP	NULL

Πίνακας 2: Αποτελέσματα SQL Query για προσπέλαση στη βάση

3) Τέλος, πραγματοποιήθηκε για λόγους επαλήθευσης, με βάση τα αποτελέσματα του SQL Query του Παραρτήματος 2, για τους πίνακες στους οποίους δεν υπήρχε προσπέλαση, έλεγχος στον κώδικα (code search) για να αποκλειστεί το ενδεχόμενο χρήσης τους.

Κατά τον έλεγχο στον πηγαίο κώδικα του project, επαληθεύτηκαν τα αποτελέσματα του 2^{ου} SQL query ότι δηλαδή οι πίνακες που δεν έγινε η προσπέλαση στη βάση δεν βρέθηκαν να χρησιμοποιούνται ούτε κατά το code search. Συνεπώς, οι πίνακες αυτοί δεν χρησιμοποιούνται από το πληροφοριακό σύστημα και μπορούν να αφαιρεθούν από το σχεσιακό διάγραμμα.

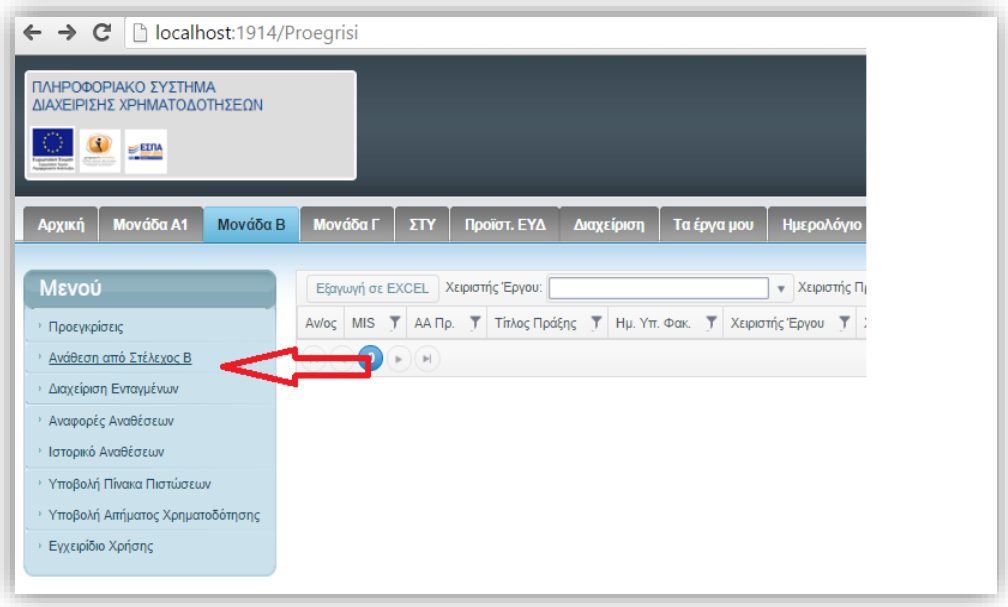
4.3 Διόρθωση λαθών (bugs)

Software bug είναι ένα λάθος, σφάλμα, αποτυχία, ή ελάττωμα σε ένα πρόγραμμα λογισμικού που το οδηγεί σε ανεπιθύμητη συμπεριφορά (π.χ. εμφάνιση λανθασμένου αποτελέσματος). Τα περισσότερα bugs προέρχονται από ανθρώπινα λάθη ή σφάλματα που γίνονται είτε στον πηγαίο κώδικα είτε στον σχεδιασμό /αρχιτεκτονική του προγράμματος, και μερικά προέρχονται από την εσφαλμένη παραγωγή κώδικα από έναν μεταγλωττιστή. Αναφορές που λεπτομερώς καταγράφουν τα bugs σε ένα πρόγραμμα αποκαλούνται συνήθως **αναφορές bugs**, αναφορές σφαλμάτων, αναφορές προβλημάτων, αναφορές αλλαγών, και τα λοιπά. [26]

Τα bugs μπορεί να έχουν μια ποικιλία επιδράσεων, με διάφορα επίπεδα δυσχέρειας προς τον χρήστη του προγράμματος. Μερικά bugs έχουν μόνο μια λεπτή επίδραση στην λειτουργικότητα του προγράμματος, και μπορούν έτσι να παραμείνουν μη ανιχνευθέντα για πολύ καιρό. Σοβαρότερα bugs μπορεί να προκαλέσουν στο πρόγραμμα πάγωμα που οδηγεί σε αδυναμία εκτέλεσης.

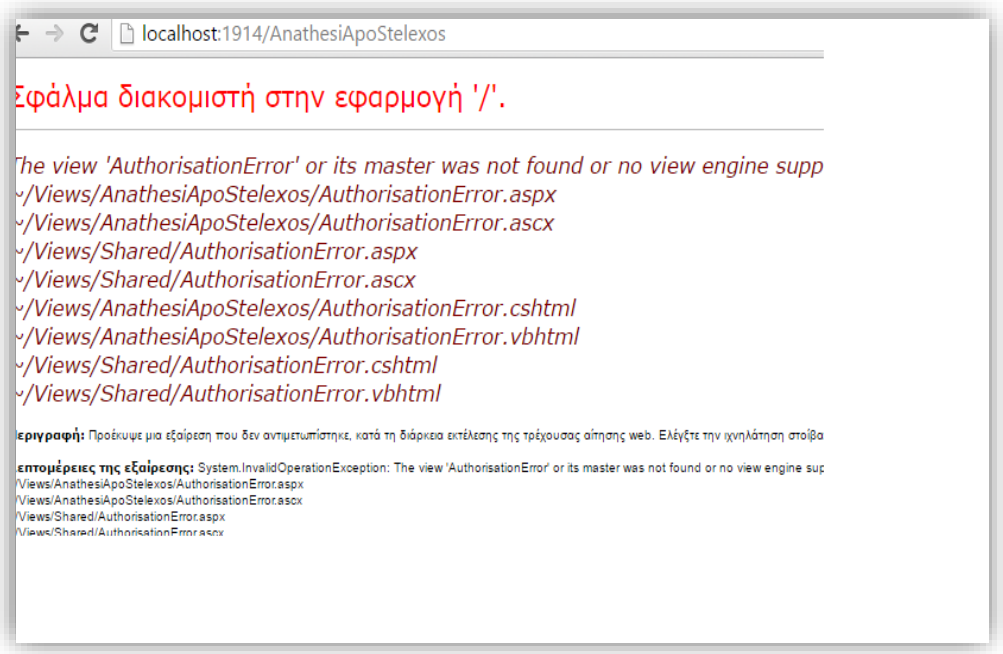
Στο Πληροφοριακό Σύστημα Διαχείρισης Χρηματοδοτήσεων τα software bugs που παρατηρήθηκαν είχαν κυρίως ως αποτέλεσμα τη δυσλειτουργία του συστήματος και τη παραπομπή σε system exceptions. Ορισμένα από αυτά παρουσιάζονται στις εικόνες που ακολουθούν.

4.3.1 Ανάθεση από Στέλεχος Β (μη εξουσιοδοτημένο)



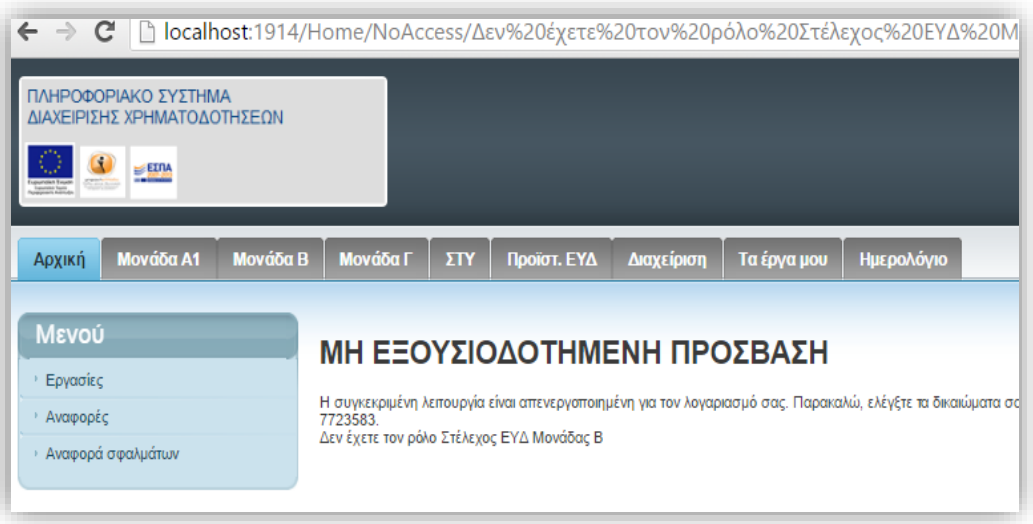
Εικόνα 12: Εντολή για Ανάθεση από Στέλεχος Β

Το κόκκινο βέλος παρουσιάζει την εντολή που δόθηκε και το αποτέλεσμα πραγματοποίησης της εντολής παρουσιάζεται στην επόμενη εικόνα.



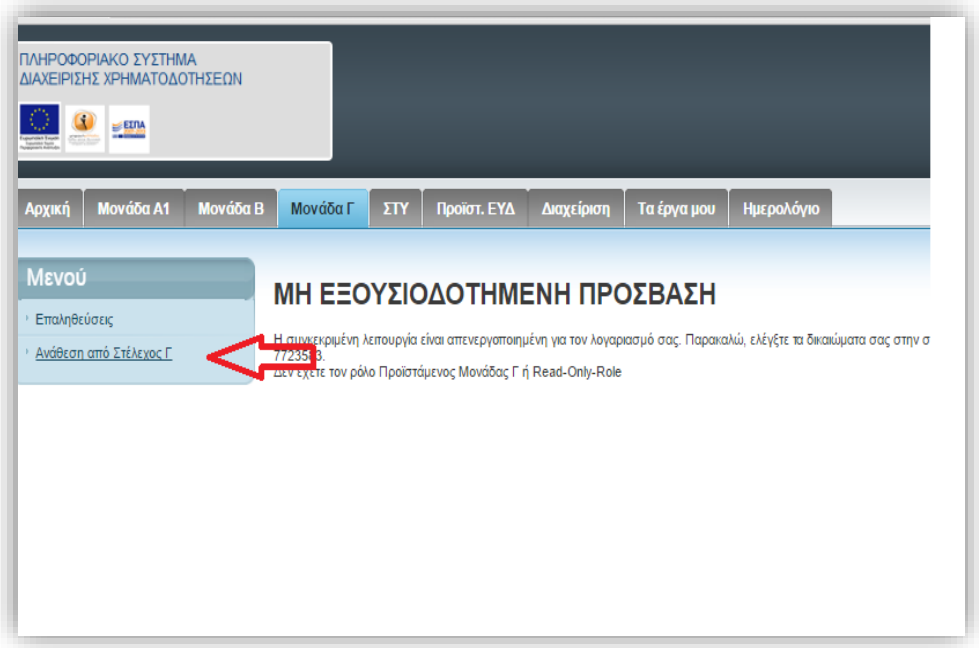
Εικόνα 13 : Αποτέλεσμα bug εντολής Ανάθεσης

Μετά τις απαραίτητες διορθώσεις στον κώδικα που οδήγησαν στο συγκεκριμένο bug, η εντολή Ανάθεση από Στέλεχος Β (μη εξουσιοδοτημένο) οδηγεί στην ακόλουθη οθόνη:



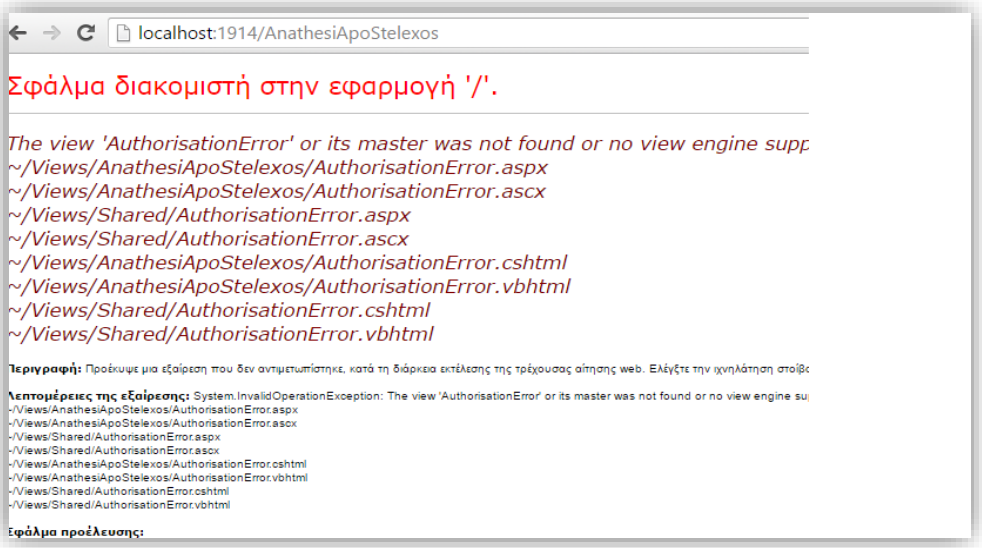
Εικόνα 14 : Διορθωμένη Εντολή Ανάθεσης Στελέχους Β

4.3.2 Ανάθεση από Στέλεχος Γ (μη εξουσιοδοτημένο)



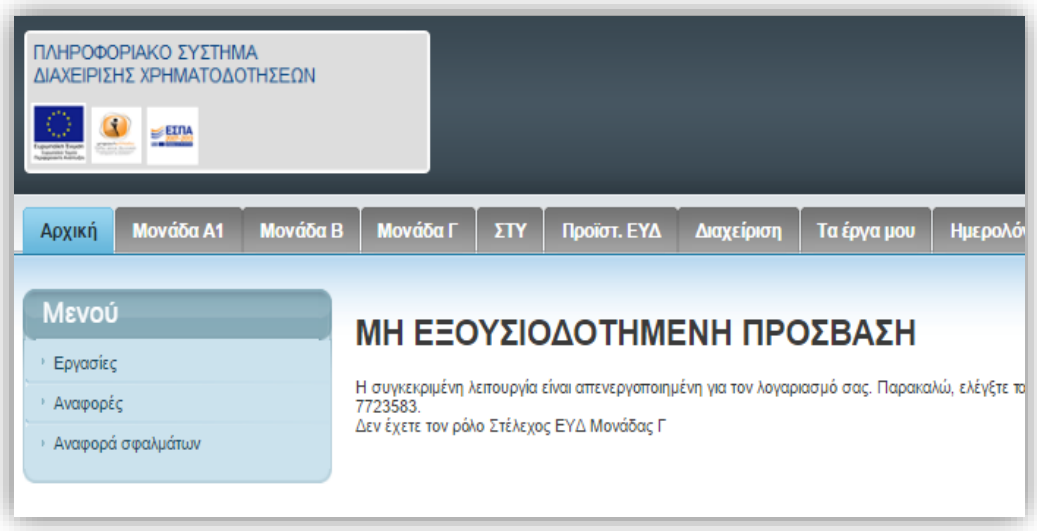
Εικόνα 15: Εντολή για Ανάθεση από Στέλεχος Γ

Αντίστοιχα, κατά την εκτέλεση της εντολής για «Ανάθεση από Στέλεχος Γ», σε περίπτωση μη εξουσιοδοτημένου χρήστη, το αποτέλεσμα φαίνεται στην παρακάτω εικόνα.



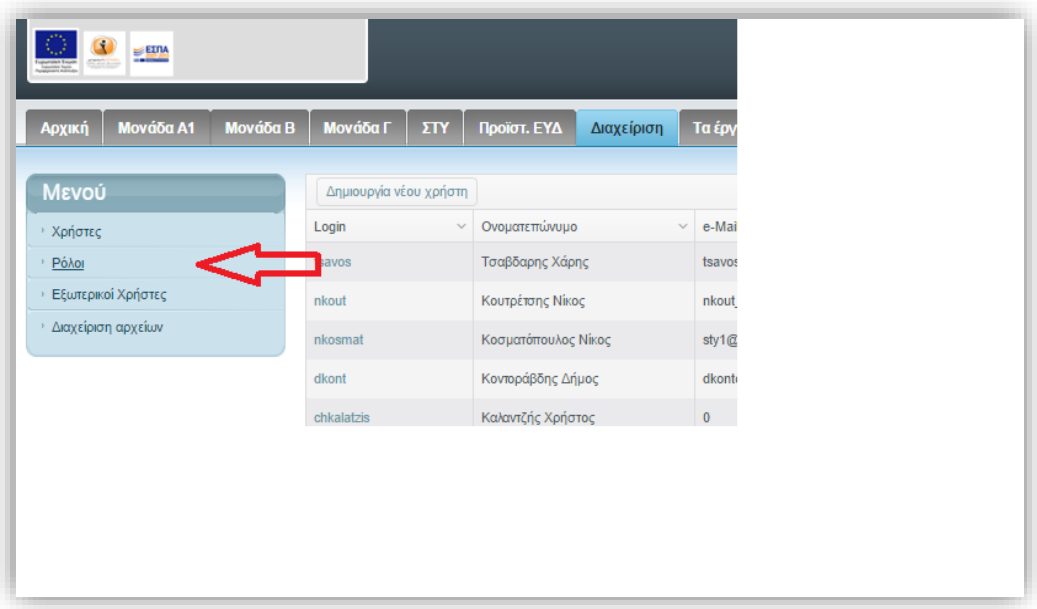
Εικόνα 16: Αποτέλεσμα bug εντολής Ανάθεσης Στελέχους Γ

Μετά τις απαραίτητες διορθώσεις του κώδικα:



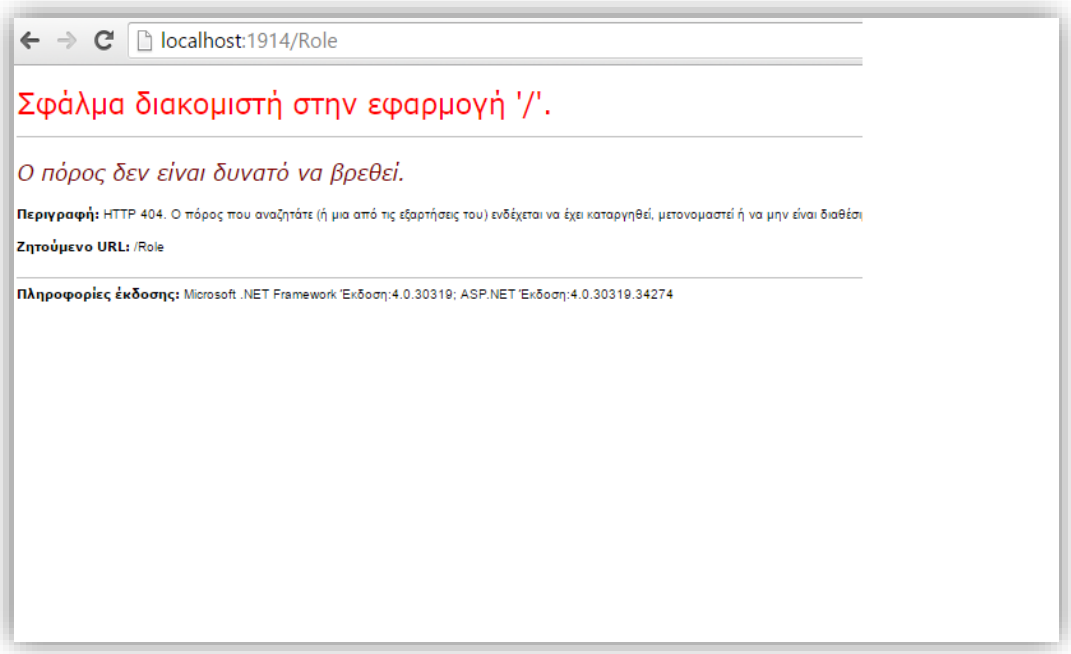
Εικόνα 17 : Διορθωμένη Εντολή Ανάθεσης Στελέχους Γ

4.3.3 Ρόλοι



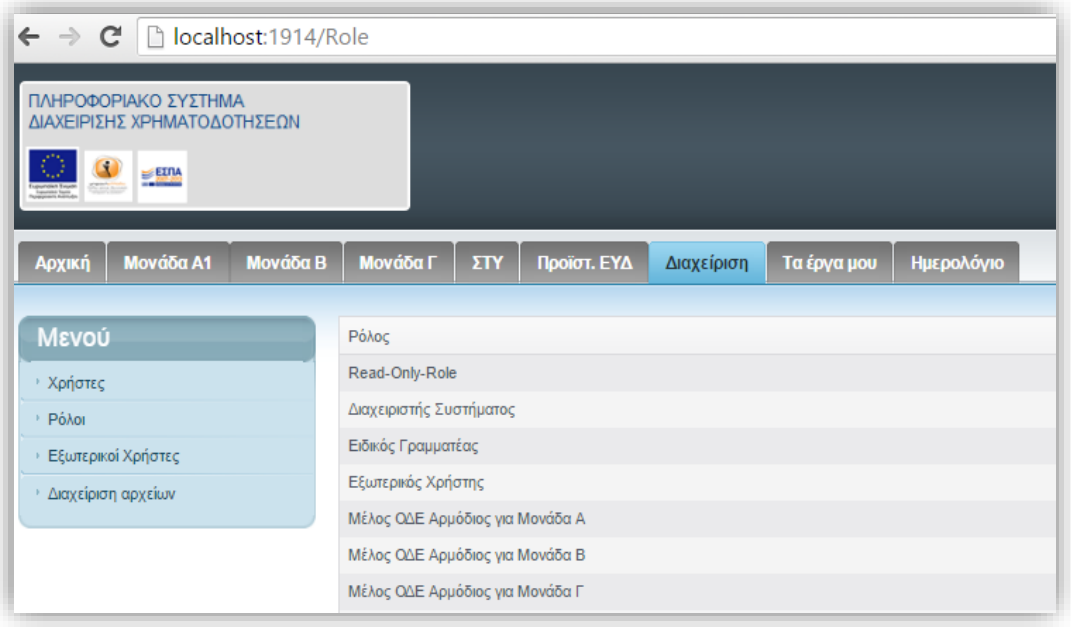
Εικόνα 18: Εντολή "Ρόλοι"

Στην περίπτωση της εντολής «Ρόλοι» διαπιστώνεται ότι δεν υπάρχει κάποια υλοποίηση και συνεπώς εμφανίζεται το επόμενο error.



Εικόνα 19: Αποτέλεσμα bug εντολής "Ρόλοι"

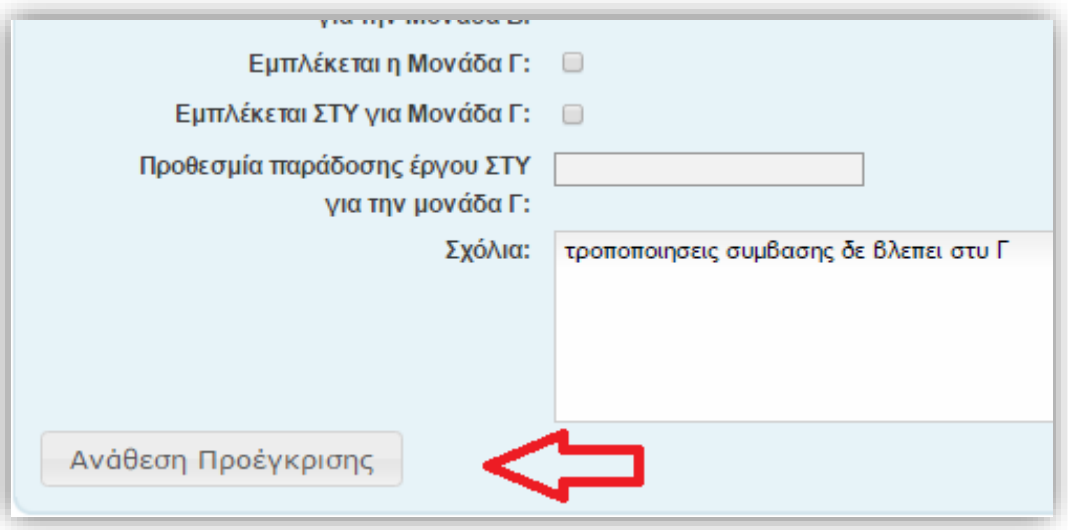
Εφόσον δεν υπήρχε κάποια υλοποίηση για την εντολή «Ρόλοι», δημιουργήθηκε μια οθόνη που απεικονίζει όλους τους πιθανούς ρόλους του πληροφοριακού συστήματος, όταν ο χρήστης έχει δικαιώματα διαχειριστή. Σε διαφορετική περίπτωση, εμφανίζεται μήνυμα Μη εξουσιοδοτημένης πρόσβασης.



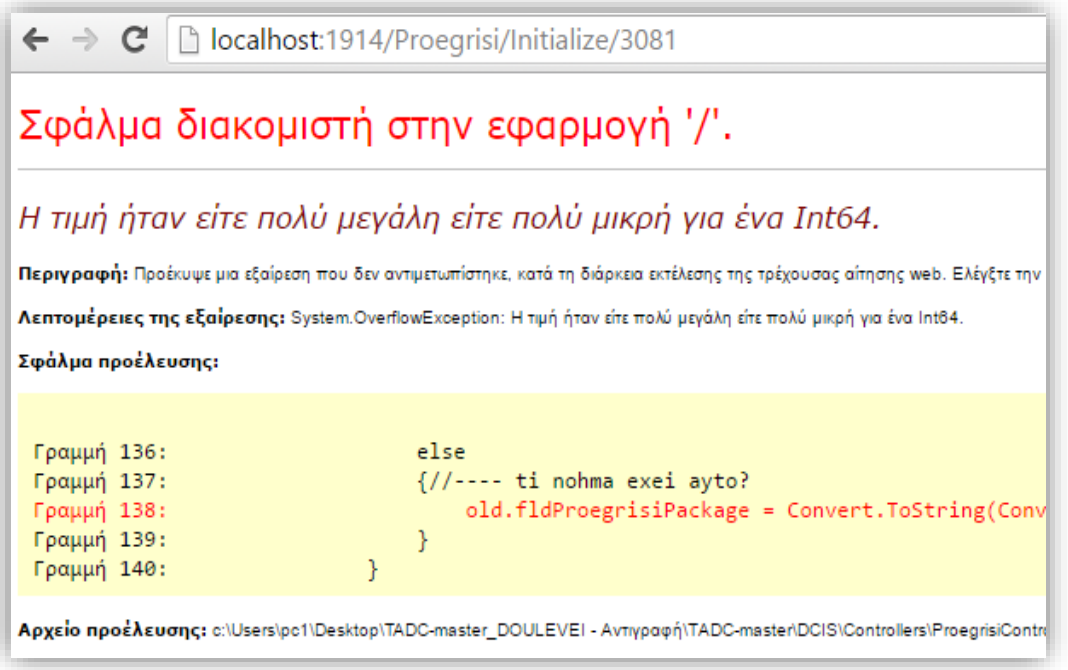
Εικόνα 20: Υλοποίηση view για εντολή "Ρόλοι"

4.3.4 Αρχικοποίηση προέγκρισης

Κατά τη διαδικασία της αρχικοποίησης προέγκρισης, εμφανίζεται το ακόλουθο bug που οφείλεται στην λανθασμένη υλοποίηση κώδικα. Αναλυτικότερα, το πεδίο fldProegkrisiPackage είναι ορισμένο στη βάση με πλήθος χαρακτήρων nvarchar(30), ωστόσο λόγω λανθασμένης χρήσης της εντολής Convert.ToInt64, το πλήθος των χαρακτήρων ξεπερνά το αντίστοιχο στη βάση δεδομένων, με αποτέλεσμα κατά την αποθήκευση να εμφανίζεται το ακόλουθο error.

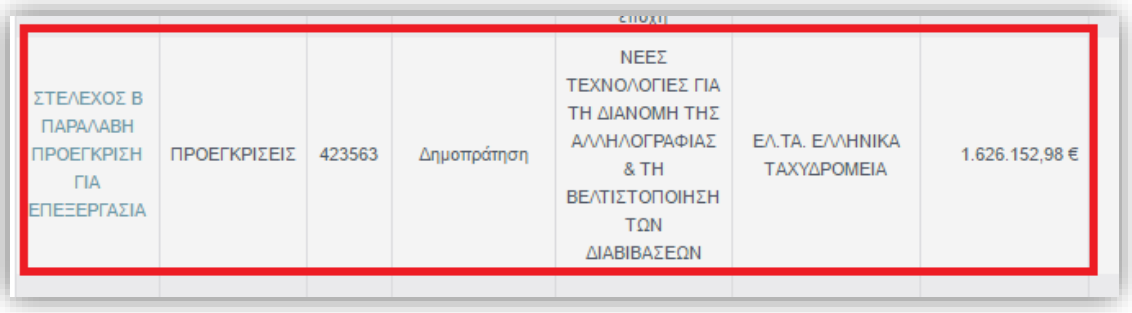


Εικόνα 21: Αρχικοποίηση προέγκρισης



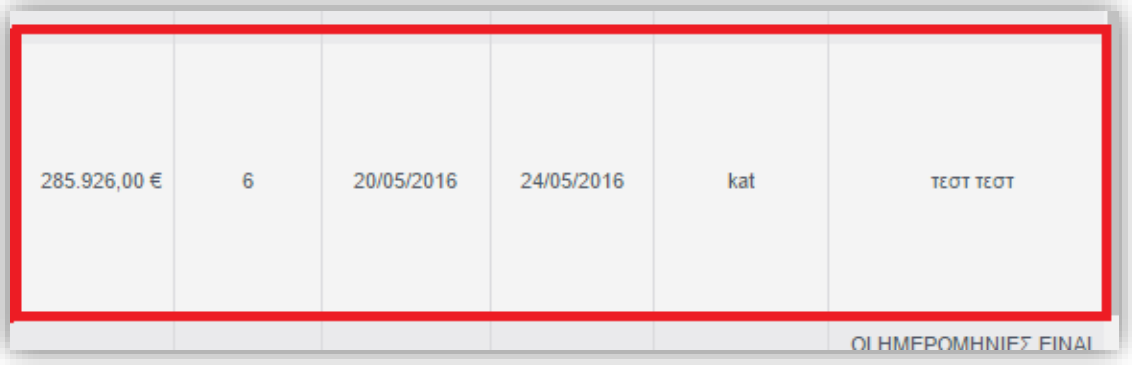
Εικόνα 22: Bug σε Αρχικοποίηση προέγκρισης

Μετά τη διόρθωση του κώδικα, και συγκεκριμένα της χρήσης της εντολής ConvertToInt64 η διαδικασία των αρχικοποιήσεων πραγματοποιείται κανονικά, όπως φαίνεται στην ακόλουθη εικόνα.



ΣΤΕΛΕΧΟΣ Β ΠΑΡΑΛΑΒΗ ΠΡΟΕΓΚΡΙΣΗ ΓΙΑ ΕΠΕΞΕΡΓΑΣΙΑ	ΠΡΟΕΓΚΡΙΣΕΙΣ	423563	Δημοπράτηση	ΝΕΕΣ ΤΕΧΝΟΛΟΓΙΕΣ ΓΙΑ ΤΗ ΔΙΑΝΟΜΗ ΤΗΣ Α/\/\Η/\ΟΓΡΑΦΙΑΣ & ΤΗ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗ ΤΩΝ ΔΙΑΒΙΒΑΣΕΩΝ	ΕΛ.ΤΑ. Ε/\/\ΗΝΙΚΑ ΤΑΧΥΔΡΟΜΕΙΑ	1.626.152,98 €

Εικόνα 23: Εμφάνιση Προέγκρισης (1)



285.926,00 €	6	20/05/2016	24/05/2016	kat	τεστ τεστ

Εικόνα 24: Εμφάνιση Προέγκρισης (2)

4.4 Code refactoring

4.4.1 Τι είναι το code refactoring

Code refactoring είναι η διαδικασία της αναδιάρθρωσης υπάρχοντα κώδικα που αλλάζει το factoring, χωρίς να αλλάζει την εξωτερική συμπεριφορά της εφαρμογής. Ουσιαστικά, το refactoring βελτιώνει μη λειτουργικές ιδιότητες του λογισμικού. Τα πλεονεκτήματα περιλαμβάνουν βελτιωμένη αναγνωσιμότητα κώδικα και μειωμένη πολυπλοκότητα: βελτιώνεται ουσιαστικά η συντηρησιμότητα του κώδικα και δημιουργείται μια πιο εκφραστική εσωτερική αρχιτεκτονική ή μοντέλο για τη βελτίωση της επεκτασιμότητας. [27] Τυπικά, το refactoring εφαρμόζει μια σειρά τυποποιημένων βασικών μικρο-refactorings, καθένα από τα οποία είναι (συνήθως) μια μικρή αλλαγή στον πηγαίο κώδικα ενός προγράμματος που είτε διατηρεί την συμπεριφορά του λογισμικού, ή τουλάχιστον δεν τροποποιεί τη συμμόρφωση του προς τις λειτουργικές απαιτήσεις. Πολλά περιβάλλοντα ανάπτυξης παρέχουν αυτοματοποιημένη υποστήριξη για την εκτέλεση των βασικών τεχνικών για refactoring. Αν πραγματοποιηθεί σωστά, το code refactoring μπορεί επίσης να επιλύσει τα κρυφά, ή άγνωστα σφάλματα υπολογιστή ή αδύναμα σημεία του συστήματος, με την απλούστευση της λογικής και την εξάλειψη των περιττών επιπέδων πολυπλοκότητας. Ωστόσο, αν δεν πραγματοποιηθεί σωστά, μπορεί να φέρει τα αντίστροφα αποτελέσματα αλλάζοντας ουσιαστικά τις εξωτερικές λειτουργίες και/ή εισάγοντας νέα σφάλματα. [26]

4.4.2 Τεχνικές για code refactoring

4.4.2.1. Τεχνικές που βασίζονται στην αρχή της αφαίρεσης (abstraction)

Η αρχή της αφαίρεσης στον προγραμματισμό (abstraction) είναι μια τεχνική για την διαχείριση της πολυπλοκότητας των υπολογιστικών συστημάτων. Εστιάζοντας περισσότερο στην σωστή αλληλεπίδραση του προγραμματιστή με το σύστημα και με τη θέσπιση επιπέδου πολυπλοκότητας, καταστέλλει τις πιο περίπλοκες λεπτομέρειες κάτω από το τρέχον επίπεδο.

- **Encapsulate Field – Χρήση κώδικα με μεθόδους getter και setter.** Στον προγραμματισμό, το field encapsulation περιλαμβάνει μεθόδους που μπορούν να χρησιμοποιηθούν για να διαβάσουν ή να γράψουν σε κάποιο πεδίο χωρίς να γίνει άμεση πρόσβαση σε αυτό. Μερικές φορές, αυτές οι μέθοδοι ονομάζονται getX και setX (όπου X το όνομα του πεδίου).

- **<u>Generalize Type</u> - δημιουργία περισσότερων γενικών τύπων για να καταστεί δυνατή η κοινή χρήση του κώδικα.** Το type generalization είναι μια τεχνική που χρησιμοποιείται συχνά στην αναμόρφωση κώδικα. Αξιοποιούνται με αυτό τον τρόπο τα οφέλη του object-orientation, χρησιμοποιώντας πιο γενικευμένους τύπους, επιτρέποντας έτσι την κοινή χρήση κώδικα, με αποτέλεσμα την καλύτερη συντήρηση και τη μείωση του όγκου κώδικα.

- **Αντικατάσταση συνθηκών με χρήση πολυμορφισμού.** Ο πολυμορφισμός είναι η δημιουργία ενός ενιαίου interface με τη χρήση διαφορετικών γλωσσών προγραμματισμού. Ένα από τα οφέλη του πολυμορφισμού είναι ότι οι τιμές από μια δραστηριότητα μπορεί να εφαρμοστεί σε κάποιον άλλο τύπο γλώσσας. [26]

4.4.2.2. Τεχνικές για το καταμερισμό του πηγαίου κώδικα σε πιο λογικά μέρη

- Η **τεχνική Componentization** χωρίζει τον κώδικα σε επαναχρησιμοποιήσιμες σημασιολογικές μονάδες που παρουσιάζουν σαφές, καλά καθορισμένο, και απλό στη χρήση interface.

- **Η Extract Class τεχνική** μεταφέρει τμήματα του κώδικα από μια υπάρχουσα κλάση σε μια νέα κλάση. Στον προγραμματισμό, η Extract Class refactoring εφαρμόζεται όταν μια κλάση περιλαμβάνει υπερβολικά πολλές μεθόδους και ο σκοπός της δεν είναι πλέον ξεκάθαρος. Περιλαμβάνει τη δημιουργία μιας νέας κλάσης και τη μεταφορά μεθόδων και/ή δεδομένων στη νέα κλάση.

- **Η Extract Method τεχνική,** μετατρέπει ένα μέρος μιας μεγαλύτερης μεθόδου σε μια νέα μέθοδο. Με την καταμερισμό του κώδικα σε μικρότερα κομμάτια, γίνεται πιο κατανοητός. Αυτό ισχύει επίσης και για τις λειτουργίες. [26]

4.4.2.3. Τεχνικές για τη βελτίωση των ονομάτων και τη θέση του κώδικα

- **Μετακίνηση μεθόδου ή Μετακίνηση πεδίου** – μετακίνηση σε μια καταλληλότερη κλάση ή αρχείο κώδικα

- **Μετονομασία Μέθοδου ή Μετονομασία πεδίου** – αλλαγή ονόματος σε ένα νέο που αποκαλύπτει καλύτερα το σκοπό του.

- **Pull Up τεχνική** - σε αντικειμενοστραφή γλώσσα προγραμματισμού, μετακίνηση σε μια υπερκλάση. Η Push Up τεχνική περιλαμβάνει την μετακίνηση ενός μέρος μιας κλάσης, για παράδειγμα μιας μεθόδου, από μια υποκλάση σε μια υπερκλάση.

- **Push Down** – σε αντικειμενοστραφή γλώσσα προγραμματισμού, μετακίνηση σε μια υποκλάση. Η Push Down τεχνική περιλαμβάνει την μετακίνηση ενός μέρος μιας κλάσης, για παράδειγμα μιας μεθόδου, από μια υπερκλάση σε μια υποκλάση.

Κατά τη διαδικασία refactoring του συγκεκριμένου project, χρησιμοποιήθηκαν κατά κύριο λόγο και σε μεγαλύτερη έκταση τεχνικές για τον καταμερισμό του κώδικα σε λογικά τμήματα, αλλά και type generalization, καθώς παρατηρήθηκε το φαινόμενο χρήσης μεγάλων μεθόδων με επαναλαμβανόμενα κομμάτια κώδικα.

Ένα χαρακτηριστικό παράδειγμα ανάγκης για code-refactoring στο Πληροφοριακό Σύστημα Διαχείρισης Χρηματοδοτήσεων αποτελεί η επανάληψη ένα μέρος του κώδικα που στέλνει ενημερωτικά emails σχετικά με την ανάθεση εργασιών στους αντίστοιχους χρήστες ή την αποστολή mail σε ρόλους. Παρατηρήθηκε στο αρχείο AnathesiApoStelexosController το συγκεκριμένο κομμάτι κώδικα να εμφανίζεται ως μέρος των μεθόδων 13 φορές. Με την Extract Method για τον καταμερισμό του κώδικα, δημιουργήθηκε νέα μέθοδος αποκλειστικά για την αποστολή emails και γίνεται η κλήση της μεθόδου στα σημεία του κώδικα που χρειάζεται.

Η νέα μέθοδος που δημιουργήθηκε :

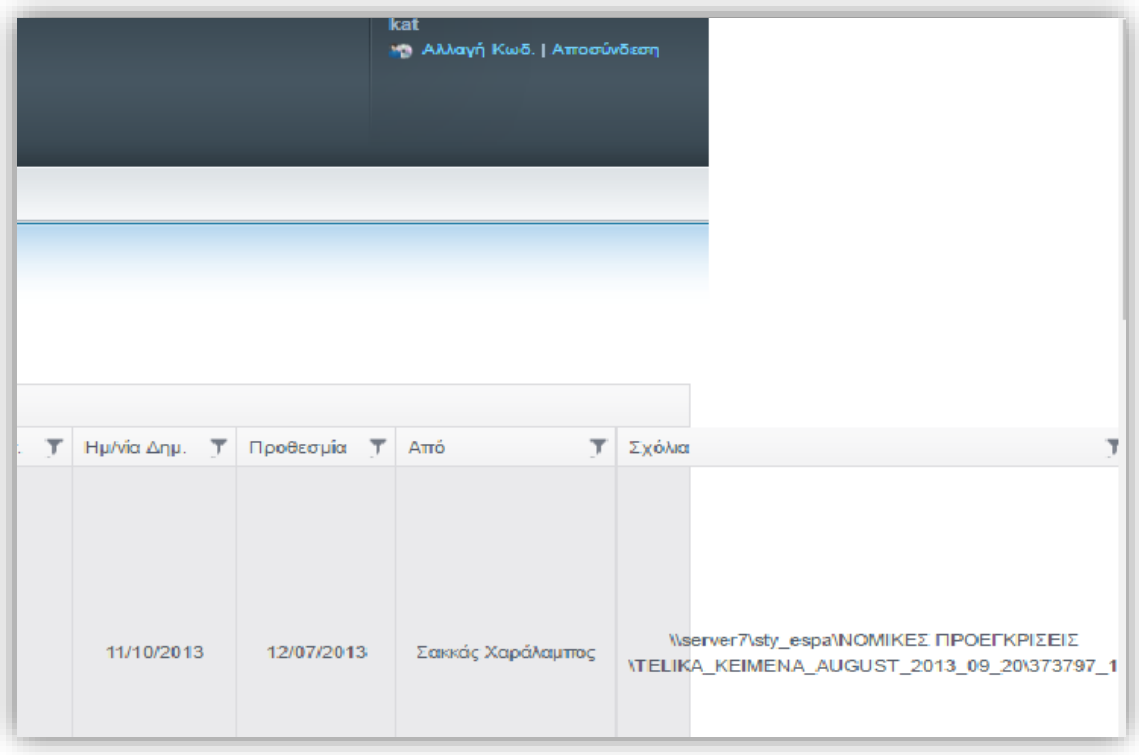
```
  private void SendMail(string subject, string acName, long curUserorRoleID, string
fldAnathesi, bool user)
      {
         string mailTemplate = "mailNotif_1.htm";
         string host = string.Format("{0}://{1}:{2}", Request.Url.Scheme,
Request.Url.Host, Request.Url.Port);
         string url = host + "/anathesi/" + acName + "/" + fldAnathesi;
         string[] pairs = { "name", "", "url", url };
         if(user)
            DCISMailSender.SendMailtoUser(curUserorRoleID, subject,
mailTemplate, pairs);
         else
            DCISMailSender.SendMailtoRole(curUserorRoleID, subject,
mailTemplate, pairs);

      }
```

4.5 CSS αλλαγές-βελτιώσεις

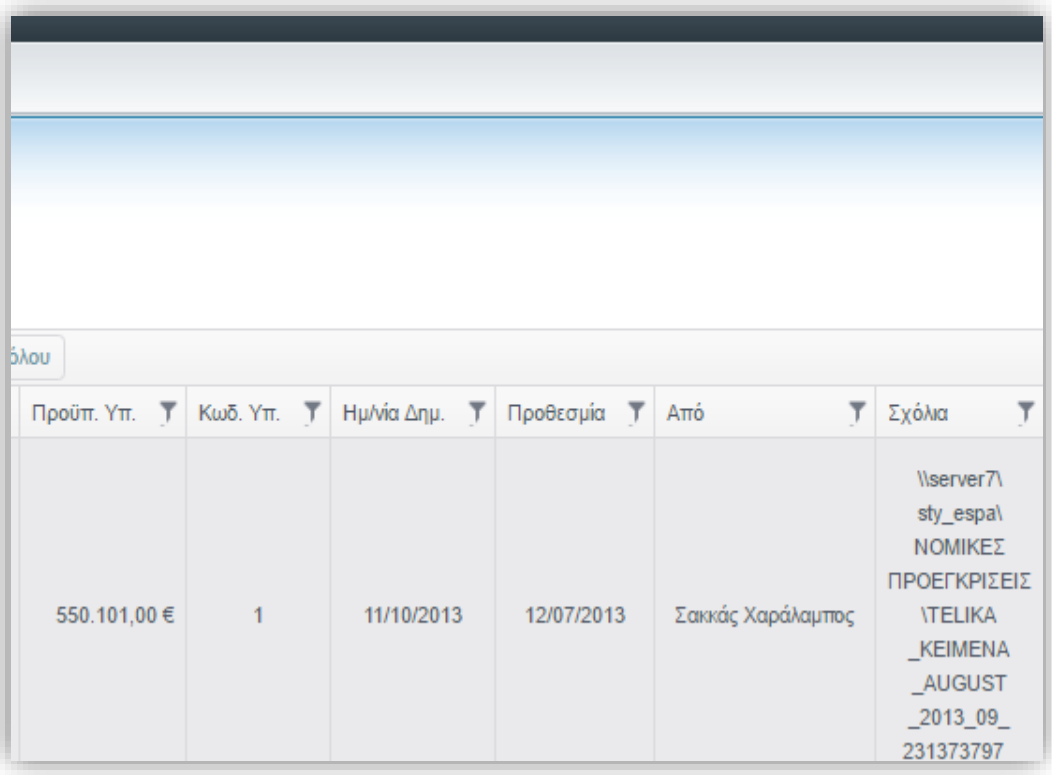
Όπως αναλύθηκε και διεξοδικότερα στο 3ο κεφάλαιο, το CSS είναι υπεύθυνο για την εμφάνιση των ιστοσελίδων. Λάθη ή αστοχίες στο CSS μιας ιστοσελίδας έχουν ως αποτέλεσμα όχι μόνο την αλλοίωση του αισθητικού αποτελέσματος μιας ιστοσελίδας, αλλά και σε ορισμένες περιπτώσεις προκαλούν προβλήματα λειτουργικότητας.

Ένα χαρακτηριστικό παράδειγμα αστοχίας CSS στο Πληροφοριακό Σύστημα Διαχείρισης Δαπανών απεικονίζεται στην εικόνα που ακολουθεί:



Εικόνα 25: Αστοχία CSS σε αρχικό grid

. Όπως είναι φανερό, δεν έχουν ορισθεί διαστάσεις για το μέγεθος κάθε στήλης του grid, με αποτέλεσμα την αστοχία της εικόνας. Με τις απαραίτητες διορθώσεις στο CSS, το αποτέλεσμα είναι η οπτική βελτίωση της ιστοσελίδας όπως φαίνεται στην εικόνα που ακολουθεί.



Εικόνα 26: Διόρθωση CSS σε αρχικό grid

4.6 Βελτίωση απόδοσης

Εκτός από την ευκολία χρήσης του Πληροφοριακού Συστήματος, μια άλλη βασική παράμετρος, αποτελεί η απόδοση του και κατά πόσο ανταποκρίνεται γρήγορα στις απαιτήσεις των χρηστών. Για τη μέτρηση της απόδοσης, χρησιμοποιήθηκε η υπηρεσία Pingdom, μια online εφαρμογή που υπολογίζει τον χρόνο απόκρισης, τον χρόνο διακοπής και την απόδοση των δικτυακών τόπων. [28]

Σύμφωνα με την υπηρεσία Pingdom, ο βαθμός απόδοσης του Πληροφοριακού Συστήματος είναι 73%, ο χρόνος απόκρισης 3,75 sec και το μέγεθος της σελίδας είναι 166.7 kB.

Αναλυτικότερα, το πρόβλημα της απόδοσης δημιουργείται κυρίως στο Leverage browser caching, στην έλλειψη cookies σε στατικό domain αλλά και στην ύπαρξη άμεσου redirect σε http σελίδες. Επιπλέον, επισημαίνεται ότι το 55,2% είναι javascript, 22,2% CSS, 19,3% εικόνες, και 3,3% HTML.

Λαμβάνοντας υπόψιν τα αποτελέσματα και εστιάζοντας στα βασικότερα προβλήματα, τα βήματα που ακολουθήθηκαν για να βελτιωθεί η απόδοση είναι τα εξής:

- Μείωση του άμεσου redirect. Οι ανακατευθύνσεις (redirects) προκαλούν ένα πρόσθετο κύκλο HTTP αίτησης -απόκρισης και καθυστερούν την απόδοση σελίδων.

- Χρήση συμπίεσης HTTP σελίδων (compression). Όλα τα σύγχρονα προγράμματα περιήγησης υποστηρίζουν τη συμπίεση gzip για όλες τα HTTP requests.Η συμπίεση gzip μπορεί να μειώσει το μέγεθος του μεταβιβασθέντος ανταπόκριση έως και 90 % ,το οποίο μπορεί να μειώσει σημαντικά την ποσότητα του χρόνου απόκρισης. [29]

- Η χρήση μιας πολιτικής προσωρινής αποθήκευσης δεδομένων του χρήστη, να καθοριστεί δηλαδή πόσο χρονικό διάστημα οι browsers θα αποθηκεύουν τοπικά εικόνες, css και τα javascripts της σελίδας. Ένα παράδειγμα ια να επιτευχθεί αυτό είναι η χρήση του παρακάτω κώδικα στο .htaccess file:

```
## EXPIRES CACHING ##
<IfModule mod_expires.c>
ExpiresActive On
ExpiresByType image/jpg "access 1 year"
ExpiresByType image/jpeg "access 1 year"
ExpiresByType image/gif "access 1 year"
ExpiresByType image/png "access 1 year"
ExpiresByType text/css "access 1 month"
ExpiresByType application/pdf "access 1 month"
ExpiresByType application/javascript "access 1 month"
ExpiresByType application/x-javascript "access 1 month"
ExpiresByType application/x-shockwave-flash "access 1 month"
ExpiresByType image/x-icon "access 1 year"
```

```
ExpiresDefault "access 2 days"
</IfModule>
## EXPIRES CACHING ##
```

- Αντικατάσταση της εκτεταμένης χρήσης των render-blocking Javascripts με JQuery. [30]

Τέλος, είναι σημαντικό να αναφερθούμε σε τρόπους βελτίωσης της απόδοσης του Entity Framework καθώς είναι το Framework που χρησιμοποιείται στο Πληροφοριακό Σύστημα Διαχείρισης Χρηματοδοτήσεων. [31] Ορισμένες τεχνικές είναι οι ακόλουθες:

- Διατήρηση του context ανοιχτό (var context = new EFContext()), μόνο για όσο διάστημα χρειάζεται.
- Βελτιστοποίηση των queries που είναι μόνο για read-only σκοπούς.
- Χρήση μεθόδου Select για τα πεδία που χρειάζονται, και όχι load ολόκληρου του αντικειμένου.
- Προσεκτική και μειωμένη χρήση των IQueryable.

Πραγματοποιώντας τις αναφερθείσες αλλαγές, διαπιστώνουμε ότι η απόδοση της web εφαρμογής, χρησιμοποιώντας πάλι την υπηρεσία Pingdom, έχει βελτιωθεί σημαντικά. Πιο συγκεκριμένα, ο βαθμός απόδοσης του Πληροφοριακού Συστήματος αυξήθηκε στο 82%, ο χρόνος απόκρισης 3,2 sec.

Λαμβάνοντας υπόψιν ότι περίπου οι μισοί web χρήστες θεωρούν ως ανεκτό χρόνο καθυστέρησης τα 3,5 sec, η βελτίωση του χρόνου απόκρισης θα συμβάλλει σημαντικά στην συνολική αύξηση της αξιοπιστίας του Πληροφοριακού Συστήματος**.**

5 Επίλογος

5.1 Σύνοψη και Συμπεράσματα

Έχοντας πραγματοποιήσει τις αλλαγές που αναλύονται στο 4ο Κεφάλαιο, και αποτυπώνονται αναλυτικότερα στον πηγαίο κώδικα, το Πληροφοριακό Σύστημα Διαχείρισης Χρηματοδοτήσεων έχει μετατραπεί σε ένα σύγχρονο, λειτουργικό και αποδοτικό εργαλείο.

Συγκεκριμένα, η απάλειψη των software λαθών, έχει αποκαταστήσει πλήρως τη λειτουργικότητα της εφαρμογής, χωρίς να παρουσιάζονται πλέον system exceptions και errors που εμποδίζαν την ομαλή διεξαγωγή των βασικών λειτουργιών του αλλά και αποτρέπαν την συστηματική χρήση του χρήστη.

Επιπλέον, ο έλεγχος στην βάση δεδομένων του πληροφοριακού συστήματος προσέφερε τη δυνατότητα αναπροσαρμογής του σχεσιακού διαγράμματος, αφαιρώντας το πλήθος των πινάκων που δεν χρησιμοποιούνται από την εφαρμογή και διατηρώντας μόνο όσα έδειξαν τα αποτελέσματα του αντίστοιχου sql query. Τα οφέλη από αυτόν τον έλεγχο είναι και η αύξηση της απόδοσης της εφαρμογής αλλά και η καλύτερη διαχείριση από πλευράς προγραμματιστή μιας και πλέον το σχεσιακό διάγραμμα απαρτίζεται μόνο από πίνακες που χρησιμοποιούνται στην εφαρμογή και όχι από όλους τους πίνακες της βάσης.

Ακόμη, αξίζει να αναφερθούμε και στις τεχνικές για αναδιάταξη του κώδικα. Οι τεχνικές αυτές έφεραν στην επιφάνεια λάθη λογισμικού αλλά και έβαλαν σε μια τάξη τον κώδικα. Κομμάτια κώδικα επαναλαμβανόντουσαν χωρίς κάποια χρησιμότητα, δυσχεραίνοντας την κατανόηση του και μειώνοντας την απόδοση της εφαρμογής.

Αρκετά σημαντικό είναι να αναφερθούμε και στις προσπάθειες που έγιναν για αύξηση της απόδοσης του συστήματος, και μείωση του χρόνου απόκρισης, που είχε ως αποτέλεσμα η απόδοση του συστήματος να αυξηθεί στο 82% και ο χρόνος απόκρισης να μειωθεί σε 3.2 sec.

Τέλος, εξίσου σημαντικό από πλευράς χρηστών είναι και η οπτική βελτίωση του πληροφοριακού συστήματος που οφείλονταν στις αστοχίες του CSS και διορθώθηκαν όπως για παράδειγμα απεικονίζονται στις αντίστοιχες εικόνες.

5.2 Μελλοντικές Αλλαγές – Νέες Τεχνολογίες

Οι νέες τεχνολογίες αποτελούν το μέλλον στο προγραμματισμό και για να παραμείνει το πληροφοριακό σύστημα ευέλικτο, λειτουργικό και ανταγωνιστικό θα πρέπει να ενσωματώσει τις νέες τεχνολογίες που προσφέρουν πλειάδα δυνατοτήτων.

5.2.1 Angular JS

Το ενδιαφέρον framework ανοιχτού λογισμικού της Google, αλλάζει τα δεδομένα στον τομέα στον διαδικτυακών εφαρμογών, καθώς έχει σχεδιαστεί με τους εξής στόχους:

- Αποδέσμευση του DOM από την λογική της εφαρμογής. Αυτό βελτιώνει την δυνατότητα ελέγχου του κώδικα.
- Ο έλεγχος της εφαρμογής έχει ίση σημασία με την διαδικασία υλοποίησης. Η δυσκολία των δοκιμών επηρεάζεται δραματικά από τον τρόπο που είναι δομημένος ο κώδικας.
- Αποσύνδεση της πλευράς του πελάτη (browser) από την πλευρά του διακομιστή. Αυτό επιτρέπει το έργο να προχωρήσει παράλληλα και επιτρέπει την επαναχρησιμοποίηση κώδικα και στις δύο πλευρές.
- Οδηγός για τους προγραμματιστές σε όλη τη διαδικασία υλοποίησης μιας εφαρμογής: από το σχεδιασμό του UI, στη δοκιμή.
- Ακολουθεί το μοτίβο MVC της μηχανικής λογισμικού και ενθαρρύνει τη χαλαρή σύνδεση μεταξύ παρουσίασης, δεδομένων και λογικής.
- Ένα μεγάλο μέρος της επιβάρυνσης για το backend μειώνεται, οδηγώντας σε πολύ ελαφρύτερες web εφαρμογές.

Γενικά, το AngularJS (http://angularjs.org/) δένει την HTML (views) σε αντικείμενα JavaScript (models). Όταν συμβαίνει μια αλλαγή στα models, η σελίδα ανανεώνεται αυτόματα. Συμβαίνει επίσης και το αντίθετο, όταν δηλαδή ένα text field ενημερώνεται στο view, το AngularJS χειρίζεται την κατάσταση αυτόματα χωρίς να χρειάζεται να γίνει κάποια επιπλέον ενέργεια στην HMTL, ή στην JQuery με την σύνδεση κάποιου event. [32]

5.2.2 PhoneGap

Τα τελευταία χρόνια, όμως, παρατηρείται μια κατακόρυφη αύξηση της χρήσης κινητών συσκευών και των εφαρμογών τους. Η JavaScript έχει αρχίσει να διεισδύει και σε αυτόν τον τομέα και μάλιστα με πολύ επιτυχημένα, πρώτα βήματα, όπως το PhoneGap (ή αλλιώς Apache Cordova).

Το PhoneGap είναι ένα framework για ανάπτυξη λογισμικού σε κινητές συσκευές που έχει πλέον αγοραστεί από την Adobe Systems και το οποίο δίνει την δυνατότητα στους προγραμματιστές, να δημιουργήσουν εφαρμογές για κινητές συσκευές με χρήση HTML5, CSS3 και JavaScript και όχι στις γλώσσες που υποστηρίζονται ανά συσκευή. Οι εφαρμογές που προκύπτουν είναι υβριδικές. Δηλαδή, δεν κάνουν χρήση των frameworks που ορίζει το λειτουργικό της κινητής συσκευής, αλλά ταυτόχρονα δεν αποτελούν καθαρά web-based εφαρμογές. Οι πλατφόρμες κινητών συσκευών, όπως το iOS, το Android και το Windows έχουν διαφορετικές απαιτήσεις, κανόνες και γλώσσες προγραμματισμού μεταξύ τους. Με την χρήση όμως του PhoneGap, η υλοποίηση πλέον μιας εφαρμογής γίνεται μια φορά για όλες τις πλατφόρμες. [33]

Παρόλα αυτά, το PhoneGap έχει κάποια μειονεκτήματα, όπως ότι δεν μπορεί να χρησιμοποιηθεί για την υλοποίηση εφαρμογών με πολλά γραφικά (π.χ. παιχνίδι) αλλά και το γεγονός πως δεν περιέχει κάποια βάση έτοιμου UI περιεχομένου, με αποτέλεσμα μερικές φορές η υλοποίηση να κρατάει περισσότερο.

Όμως, το πρώτο βήμα για την ενοποίηση όλων των πλατφόρμων που χρησιμοποιούνται στις κινητές συσκευές έγινε. Και έγινε με την βοήθεια της JavaScript.

Οι τελευταίες εξελίξεις στον χώρο του προγραμματισμού αναδεικνύουν κυρίαρχο του παιχνιδιου την JavaScript, η οποία γίνεται όλο και πιο δημοφιλής. Πλέον, έχουν δημιουργηθεί πανίσχυρες βιβλιοθήκες, οι οποίες διευκολύνουν την ανάπτυξη εφαρμογών σε web, πλατφόρμες και κινητά. Το καλύτερο όμως είναι, ότι η υλοποίηση τους θα μπορούσε να γίνει ενοποιημένα, γεγονός που βοηθάει τους προγραμματιστές στην δημιουργία νέων, καινοτόμων προγραμμάτων.

6 Βιβλιογραφία

[1] «.[Y.Yusufa, A.Gunasekaranb, and M.S.Abthorpe, 2004].».

[2] [Mouratidis H., Giorgini P, Manson G 2005]..

[3] *Garzotto et al., 1995; Isakowitz et al., 1995; Nanard and Nanard, 1995; Schwabe and Rossi, 1995.*

[4] J. O. a. G. Marakas, Management Information Systems, 2010.

[5] *Sano, 1996; Horton et al., 1996.*

[6] K. E, «Μεθοδολογίες Ανάλυσης και Σχεδιασμού Πληροφοριακών Συστημάτων,» [Ηλεκτρονικό]. Available: http://ecourse.uoi.gr/file.php/67/Kefalaio_1.pdf.

[7] K. L. a. J. Laudon, Management Information Systems, Managing the Digital firm, 11th Edition, 2009.

[8] [Ηλεκτρονικό]. Available: http://www.digitalplan.gov.gr/portal/resource/Hlektronikh-Ypobolh-Aithmatwn-Hrhmatodothshs-E.P.-PShfiakh-Sygklish.

[9] «Tools for every developer and every app,» [Ηλεκτρονικό]. Available: https://www.visualstudio.com/.

[10] «.NET Framework,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/.NET_Framework.

[11] «LINQ (Language-Integrated Query),» [Ηλεκτρονικό]. Available: http://msdn.microsoft.com/en-us/library/vstudio/bb397926.aspx.

[12] [Ηλεκτρονικό]. Available: http://www.asp.net/.

[13] «ASP.NET - Introduction,» [Ηλεκτρονικό]. Available: http://www.tutorialspoint.com/asp.net/asp.net_introduction.htm.

[14] «Learn About ASP.NET MVC,» [Ηλεκτρονικό]. Available: http://www.asp.net/mvc.

[15] M. A. Team, «ASP.NET MVC Overview,» [Ηλεκτρονικό]. Available: http://www.asp.net/mvc/tutorials/older-versions/overview/asp-net-mvc-overview.

[16] M. A. Team, «ASP.NET MVC Overview,» [Ηλεκτρονικό]. Available: http://www.asp.net/mvc/tutorials/older-versions/overview/asp-net-mvc-overview.

[17] «C# Station,» [Ηλεκτρονικό]. Available: http://www.csharp-station.com/Tutorial.aspx.

[18] «Ajax (programming),» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Ajax_%28programming%29.

[19] «About jQuery,» [Ηλεκτρονικό]. Available: http://learn.jquery.com/.

[20] B. Bos, «Cascading Style Sheets,» [Ηλεκτρονικό]. Available: https://www.w3.org/Style/CSS/.

[21] [Ηλεκτρονικό]. Available: https://www.javascript.com/about.

[22] «JavaScript Tutorial,» [Ηλεκτρονικό]. Available: http://www.w3schools.com/Js/.

[23] «Entity Framework,» [Ηλεκτρονικό]. Available: https://msdn.microsoft.com/en-us/data/ef.aspx.

[24] «Microsoft SQL Server,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Microsoft_SQL_Server.

[25] D. Schlichting, «MS SQL,» [Ηλεκτρονικό]. Available: http://www.databasejournal.com/features/mssql/article.php/3769211/What-is-SQL-Server.htm.

[26] J. Kerievsky, «Refactoring to Patterns,» [Ηλεκτρονικό]. Available: https://www.wikiwand.com/en/Code_refactoring.

[27] «Code refactoring,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Code_refactoring.

[28] «Pingdom Website Speed Test,» [Ηλεκτρονικό]. Available: https://www.pingdom.com/.

[29] [Ηλεκτρονικό]. Available: https://developers.google.com/speed/.

[30] A. Lumsden, «Best Practices for Increasing Website Performance,» 24 Oct 2015. [Ηλεκτρονικό]. Available: http://webdesign.tutsplus.com/articles/best-practices-for-increasing-website-performance--webdesign-9109.

[31] B. Emmett, «Entity Framework Performance and What You Can Do About It,» 16 December 2015. [Ηλεκτρονικό]. Available: https://www.simple-talk.com/dotnet/.net-tools/entity-framework-performance-and-what-you-can-do-about-it/.

[32] «Angular Templates,» [Ηλεκτρονικό]. Available: https://angularjs.org/.

[33] «PhoneGap Tutorial,» [Ηλεκτρονικό]. Available: http://www.tutorialspoint.com/phonegap/.

[34] Γ. Τζήμας, Βελτίωση Απόδοσης και Αποτελεσματικές Σχεδιαστικές Λύσεις για Εφαρμογές Παγκόσμιου Ιστού.

[35] [Ηλεκτρονικό]. Available: http://www.clickmedia.gr/behindtheclicks/2014/02/javascript-%CE%B7-%CE%BD%CE%AD%CE%B1-%CE%B4%CF%8D%CE%BD%CE%B1%CE%BC%CE%B7-%CF%83%CF%84%CE%BF-web-%CE%BA%CE%B1%CE%B9-%CF%8C%CF%87%CE%B9-%CE%BC%CF%8C%CE%BD%CE%BF/.

[36] [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Stored_procedure.

[37] [Ηλεκτρονικό]. Available: https://tools.pingdom.com

[38] «Code refactoring,» [Ηλεκτρονικό]. Available: https://www.wikiwand.com/en/Code_refactoring.

[39] «Analyze and optimize your website with PageSpeed tools,» [Ηλεκτρονικό]. Available: https://developers.google.com/speed/.

Παράρτημα

Παράρτημα Α: SQL Query για έλεγχο μεγέθους βάσης

Το query που δείχνει πόσες εγγραφές περιέχει κάθε πίνακας της βάσης δεδομένων και τι μέγεθος έχει, και χρησιμοποιήθηκε στο κεφάλαιο 4.2.2 είναι το εξής:

```
CREATE TABLE #RowCountsAndSizes (TableName NVARCHAR(128),rows CHAR(11),
       reserved VARCHAR(18),data VARCHAR(18),index_size VARCHAR(18),
       unused VARCHAR(18))

EXEC       sp_MSForEachTable 'INSERT INTO #RowCountsAndSizes EXEC sp_spaceused
''?'' '

SELECT     TableName,CONVERT(bigint,rows) AS NumberOfRows,
           CONVERT(bigint,left(reserved,len(reserved)-3)) AS SizeinKB
FROM       #RowCountsAndSizes
ORDER BY   NumberOfRows DESC,SizeinKB DESC,TableName

DROP TABLE #RowCountsAndSizes
```

Παράρτημα Β: SQL Query για έλεγχο χρήσης βάσης

Το query που χρησιμοποιήθηκε στο κεφ. 4.4.2 και ελέγχει σε ποιους πίνακες υπήρχε πρόσβαση κατά τη διάρκεια χρήσης της εφαρμογής:

```
WITH LastActivity (ObjectID, LastAction) AS
  (
       SELECT object_id AS TableName,
              last_user_seek as LastAction
         FROM sys.dm_db_index_usage_stats u
        WHERE database_id = db_id(db_name())
        UNION

       SELECT object_id AS TableName,
              last_user_scan as LastAction
         FROM sys.dm_db_index_usage_stats u
        WHERE database_id = db_id(db_name())
        UNION

       SELECT object_id AS TableName,
              last_user_lookup as LastAction
         FROM sys.dm_db_index_usage_stats u
        WHERE database_id = db_id(db_name())
  )

  SELECT OBJECT_NAME(so.object_id) AS TableName,
         MAX(la.LastAction) as LastSelect
    FROM sys.objects so
    LEFT JOIN LastActivity la
      on so.object_id = la.ObjectID
   WHERE so.type = 'U'

     AND so.object_id > 100
GROUP BY OBJECT_NAME(so.object_id)
ORDER BY OBJECT_NAME(so.object_id)
```

Παράρτημα Γ: Πίνακας Εικόνων

Εικόνα 1 : Φάσεις ενός Πληροφοριακού Συστήματος 21
Εικόνα 2 : Κύκλος ζωής ενός Πληροφοριακού Συστήματος [4] 23
Εικόνα 3: Παρουσίαση συστημάτων ενός οργανισμού 25
Εικόνα 4: Διάγραμμα ροής διαδικασιών σε Π.Σ.Δ.Χ 30
Εικόνα 5 : Αρχιτεκτονική του NET Framework 33
Εικόνα 6 : Τα τρία διακριτά τμήματα του προτύπου MVC 37
Εικόνα 7: Η αλληλεπίδραση ανάμεσα στα στοιχεία του MVC 38
Εικόνα 8: Γενικός τρόπος επεξεργασίας μίας αίτησης 39
Εικόνα 9: Μέρος του σχεσιακού διαγράμματος της βάσης δεδομένων 45
Εικόνα 10: Μέρος ΙΙ του σχεσιακού διαγράμματος της βάσης δεδομένων 46
Εικόνα 11:Μέρος ΙΙΙ του σχεσιακού διαγράμματος της βάσης δεδομένων 46
Εικόνα 12: Εντολή για Ανάθεση από Στέλεχος Β 57
Εικόνα 13 : Αποτέλεσμα bug εντολής Ανάθεσης 58
Εικόνα 14 : Διορθωμένη Εντολή Ανάθεσης Στελέχους Β 58
Εικόνα 15: Εντολή για Ανάθεση από Στέλεχος Γ 59
Εικόνα 16: Αποτέλεσμα bug εντολής Ανάθεσης Στελέχους Γ 59
Εικόνα 17 : Διορθωμένη Εντολή Ανάθεσης Στελέχους Γ 60
Εικόνα 18: Εντολή "Ρόλοι" 60
Εικόνα 19: Αποτέλεσμα bug εντολής "Ρόλοι" 61
Εικόνα 20: Υλοποίηση view για εντολή "Ρόλοι" 61
Εικόνα 21: Αρχικοποίηση προέγκρισης 62
Εικόνα 22: Bug σε Αρχικοποίηση προέγκρισης 62
Εικόνα 23: Εμφάνιση Προέγκρισης (1) 63
Εικόνα 24: Εμφάνιση Προέγκρισης (2) 63
Εικόνα 25: Αστοχία CSS σε αρχικό grid 67
Εικόνα 26: Διόρθωση CSS σε αρχικό grid 68