ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάπτυξη Κατανεμημένης Εφαρμογής Διαχείρισης Έργων με Χρήση Περιβάλλοντος Java Spring

---

Καραβίας Δημήτριος Εμμανουήλ
Α.Μ.: 3935

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Επιβλέπων: Καθηγητής Χρήστος Ζαρολιάγκης
Συνεπιβλέπων: Δρ. Ιωάννης Χατζηγιαννάκης

Πάτρα
Σεπτέμβριος 2012

Στη μητέρα μου

Περίληψη

Η παρούσα διπλωματική εργασία ασχολείται με την ανάπτυξη λογισμικού σε ένα ήδη υπάρχον πληροφοριακό σύστημα. Αρχικά θα κάνουμε μία σύντομη ανάλυση του πληροφοριακού συστήματος και των εργαλείων που χρησιμοποιήθηκαν για την ανάπτυξή του. Έπειτα θα χρησιμοποιήσουμε αυτά τα εργαλεία για την επέκταση του συστήματος αυξάνοντας τις δυνατότητές του. Τέλος θα κάνουμε μία αναφορά στις κρυφές μνήμες και στα οφέλη που αυτές έχουν σε συστήματα σαν και αυτό που θα μελετήσουμε και θα εφαρμόσουμε δύο τεχνολογίες κρυφών μνημών στο σύστημα και θα δούμε τα οφέλη αυτά στην πράξη.

Στο σημείο αυτό θα ήθελα να ευχαριστήσω τον Καθηγητή και επιβλέπωντα της διπλωματικής μου κ. Χρήστο Ζαρολιάγκη ο οποίος μου έδωσε την ευκαιρία να ασχοληθώ με το συγκεκριμένο αντικείμενο.

Επίσης θα ήθελα να ευχαριστήσω τον συνεπιβλέποντα της διπλωματικής μου, Δρ. Χατζηγιαννάκη Ιωάννη, ο οποίος με τον χαρακτήρα και την εμπειρία του βοήθησε με πολύ εποικοδομητικό τρόπο σε όλη την πορεία αυτής της διπλωματικής εργασίας.

Ακόμη θα ήθελα να ευχαριστήσω και τον κ. Σταθόπουλο Αναστάσιο, πρωτεργάτη του συστήματος το οποίο θα παρουσιάσουμε στην συνέχεια καθώς στο ξεκίνημά της ασχολίας μου με αυτή την διπλωματική εργασία με βοήθησε να μπω στη λογική της υλοποίησης που είχε γίνει.

Τέλος θα ήθελα να ευχαριστήσω την κ. Έφη Χήτα με την οποία είχα την τύχη να συνεργαστώ στα πλαίσια ανάπτυξης της υλοποίησης που περιλαμβάνει η διπλωματική εργασία, και η οποία ήταν εκεί να λύσει κάθε διαδικαστική απορία που αφορούσε τον οργανισμό στον οποίο επικεντρώνεται το σύστημα για το οποίο θα μιλήσουμε.

Περιεχόμενα

Εικόνες και πίνακες

Ι. Εισαγωγή

1.1 Κίνητρο και σημασία του θέματος

Οι πληροφορίες σήμερα είναι ζωικής σημασίας για κάθε οργανισμό. Κατεστραμμένα ή χαμένα δεδομένα μπορούν να προκαλέσουν διαταραχές στις κανονικές δραστηριότητες του οργανισμού και να οδηγήσουν σε οικονομικά προβλήματα. Τα πληροφοριακά συστήματα αποτελούνται από υλικό και λογισμικό τα οποία υλοποιούν τις πολύπλοκες λογικές διαδικασίες που απαιτούνται, δεδομένα τα οποία είναι απαραίτητα για τον οργανισμό, εφαρμογές για την διαχείριση αυτών των δεδομένων, ανθρώπους οι οποίοι συντηρούν αυτά τα συστήματα. Αυτά τα πληροφοριακά συστήματα βοηθούν έναν οργανισμό να κάνει καλύτερη διαχείριση των εσωτερικών του διαδικασιών, να παρέχει ασφάλεια και αξιοπιστία και προφύλαξη σε ζωτικές πληροφορίες για την εταιρία, πληροφορίες που μπορεί να αφορούν πελάτες ή εργαζόμενους σε αυτή.

Ένα άλλο βήμα στο οποίο η πληροφορική βοηθάει μία εταιρία είναι στην εσωτερική της δομή και στις εσωτερικές διαδικασίες που διέπουν την εταιρία. Για κάθε μεγάλη εταιρία είναι σημαντικό να έχει εσωτερική οργάνωση, να αποφύγει όσο το δυνατόν γίνεται τις γραφειοκρατικές διαδικασίες και να μην χωλαίνει σε διαδικασίες που σχετίζονται με τις εσωτερικές της πολιτικές και όχι με την παραγωγικότητά της. Υπάρχουν πολλές διαδικασίες μέσα σε έναν οργανισμό που μπορούν να αυτοματοποιηθούν με την χρήση πληροφοριακών συστημάτων. Οργανισμοί οι οποίοι οργανώνονται με βάση τα έργα αποτελούν ένα χαρακτηριστικό παράδειγμα αυτής της περίπτωσης. Τέτοιοι οργανισμοί καταμερίζουν τις εργασίες τους σε έργα. Κάθε έργο σε έναν οργανισμό είναι μία καλά ορισμένη δραστηριότητα η οποία θα πραγματοποιηθεί για την επίτευξη κάποιων συγκεκριμένων στόχων. Χαρακτηρίζεται από προκαθορισμένες ημερομηνίες έναρξης και λήξης, ενώ τόσο οι στόχοι όσο και η πόροι που θα καταναλωθούν για την επίτευξη τους περιορίζονται από χαρακτηριστικά όπως ο προϋπολογισμός του έργου και τα παραδοτέα που προκύπτουν. Επίσης κάθε έργο μπορεί να χωρίζεται σε επιμέρους στόχους οι οποίοι στο σύνολό τους υλοποιούν τους αρχικούς στόχους που τέθηκαν με την δημιουργία του έργου. Ένα έργο αποτελείται από έναν ή περισσότερους υπεύθυνους για την λειτουργία του, τους διαχειριστές του έργου. Οι διαχειριστές αυτοί έχοντας ως δεδομένο τους στόχους τους οποίους πρέπει το έργο να φέρει εις πέρας, είναι υπεύθυνοι για την σχεδίαση και την οργάνωση του έργου, καθώς και την κατευθυντήρια γραμμή την οποία αυτό θα ακολουθήσει. Είναι υπεύθυνοι για την διαχείριση των πόρων που έχουν στη διάθεσή τους και την σωστή κατανομή των εργασιών στην ομάδα εργασίας του έργου. Όταν πρόκειται για μεγάλα έργα με μεγάλο αριθμό εργαζόμενων και πολύπλοκους στόχους η παρακολούθηση και η οργάνωση των έργων είναι μία πρόκληση για κάθε διαχειριστή.

Τα πληροφοριακά συστήματα διαχείρισης έργων αποτελούν κομμάτι των πληροφοριακών συστημάτων διαχείρισης, και διαχειρίζονται πληροφορίες σε οργανισμούς οι οποίοι βασίζονται στα έργα. Αυτού του είδους τα πληροφοριακά συστήματα στόχο έχουν να βοηθήσουν στην οργάνωση των διαχειριστικών έργων που περιλαμβάνει την διαχείριση των οικονομικών και άλλων πόρων, τον καταμερισμό των έργων σε υποέργα, την ανάθεση εργασίας στην ομάδα του έργου. Επίσης βοηθούν στην εκτέλεση και την ολοκλήρωση των διαχειριστικών στόχων που αφορούν το έργο.

Τέτοιοι στόχοι μπορεί να είναι η σχεδίαση, η οργάνωση, ο έλεγχος των διαθέσιμων πόρων με σκοπό την ολοκλήρωση των στόχων του έργου. Χρησιμοποιούνται για να δημιουργήσουν ένα ακριβές πρόγραμμα και να ορισθεί η γραμμή που θα ακολουθήσει το έργο. Κατά την διάρκεια εκτέλεσης του έργου συλλέγονται πληροφορίες οι οποίες αποθηκεύονται σε μία βάση δεδομένων, το πληροφοριακό σύστημα χρησιμοποιείται για να συγκρίνει την γραμμή κατεύθυνσης του έργου με την πραγματική του κατεύθυνση με βάση τις πληροφορίες που συλλέχθηκαν για κάθε δραστηριότητα που περιλαμβάνει. Χρησιμοποιείται για την διαχείριση των υλικών, για την συλλογή οικονομικών στοιχείων καθώς και για την εξαγωγή αναφορών. Ενώ κατά την ολοκλήρωση χρησιμοποιείται για να επανεξεταστούν οι στόχοι του έργου που τέθηκαν αρχικά και κατά πόσο διεκπεραιώθηκαν.

Αυτή τη στιγμή υπάρχουν αρκετά πληροφοριακά συστήματα διαχείρισης έργων διαθέσιμα και τα οποία εταιρίες μπορούν να τα αγοράσουν και να τα χρησιμοποιήσουν. Τέτοια συστήματα είναι το Microsoft Project, το Micro-Planner Manager, Primavera Project Planner, Artemis Views, Open Plan, Enterprise PM, MicroPlanner X-Pert. Ωστόσο αυτά όπως και άλλα συστήματα που διατίθενται καλύπτουν βασικές ανάγκες των οργανισμών χωρίς να μπορούν να επικεντρωθούν στις ιδιαίτερες ανάγκες που μπορεί να έχει ένας συγκεκριμένος οργανισμός. Για αυτό είναι σύνηθες σε μεγάλους οργανισμούς οι οποίοι χρησιμοποιούν αυτού του είδους τα συστήματα να υπάρχει και μία ομάδα η οποία να αναπτύσσει το σύστημα, να το συντηρεί και να το εξελίσσει καθώς και ο ίδιος ο οργανισμός εξελίσσεται με την πάροδο του χρόνου και θα πρέπει να καλύπτει τις καινούριες ανάγκες που μπορεί να έχει.

1.2 Στόχοι της διπλωματικής εργασίας

Οι στόχοι αυτής εργασίας περιστρέφονται γύρω από το σύστημα διαχείρισης ερευνητικών έργων ΝΕΔΑ που χρησιμοποιείται από το Ινστιτούτο Τεχνολογίας Υπολογιστών & Εκδόσεων (Ι.Τ.Υ.Ε.) για την παρακολούθηση των ερευνητικών αναπτυξιακών έργων.

Ο πρώτος βασικός στόχος της διπλωματικής εργασίας είναι η ανάπτυξη λογισμικού χρησιμοποιώντας τα εργαλεία με τα οποία αναπτύχθηκε αυτό το σύστημα για την επέκταση του ώστε να είναι δυνατή η εισαγωγή και η έγκριση αιτημάτων μετακίνησης από τους εργαζόμενους του οργανισμού. Ταυτόχρονα θα γίνει ευκολότερη η παρακολούθηση των ταξιδιών καθώς θα είναι διαθέσιμα για επεξεργασία σε ηλεκτρονική μορφή και θα αυτοματοποιήσουμε την διαδικασία παραγωγής εντύπων απαραίτητων για την πραγματοποίηση ενός ταξιδιού.

Επόμενος και εξίσου σημαντικός στόχος είναι η χρήση κρυφών μνημών για την επιτάχυνση της επίδοσης του συστήματος, από την σκοπιά του χρόνου εξυπηρέτησης των αιτημάτων που δέχεται, χρησιμοποιώντας υλοποιήσεις κρυφών μνημών που μπορούν να εφαρμοστούν άμεσα στα υπάρχοντα περιβάλλοντα εργασίας που χρησιμοποιούνται από το σύστημα.

1.3 Συνεισφορά της διπλωματικής εργασίας

Η συνεισφορά της παρούσας διπλωματικής εργασίας χωρίζεται σε δύο σκέλη τα οποία αφορούν τις ανάγκες του οργανισμού Ε.Α.Ι.Τ.Υ.. Εργαζόμαστε πάνω στο πληροφοριακό σύστημα διαχείρισης έργων ΝΕΔΑ το οποίο χρησιμοποιείται από τον οργανισμό για την καλύτερη παρακολούθηση και οργάνωση των έργων και των

εργαζομένων σε αυτά. Στο πρώτο σκέλος επεκτείνουμε τις δυνατότητές αυτού του πληροφοριακού συστήματος ώστε να δώσουμε την δυνατότητα καλύτερης παρακολούθησης των δαπανών μετακίνησης των ταξιδιών που πραγματοποιούνται από τους εργαζόμενους του. Κάνουμε την διαδικασία εγκρίσεων αυτών ηλεκτρονική εισάγοντας τις ηλεκτρονικές υπογραφές ώστε να εξυπηρετήσουμε τις ανάγκες του οργανισμού, ενώ αυτόματα παρέχεται η δυνατότητα παρακολούθησης των ταξιδιών που χρεώνονται στα έργα του οργανισμού παρέχοντας άμεσο έλεγχο στις δαπάνες μετακίνησης που προκύπτουν σε κάθε έργο.

Το πληροφοριακό σύστημα ΝΕΔΑ είναι ένα σύστημα που μεγαλώνει και τα δεδομένα αυξάνονται με γρήγορο ρυθμό με αποτέλεσμα η συνολική ταχύτητα του συστήματος να μειώνεται. Το δεύτερο σκέλος της συνεισφοράς αυτής της διπλωματικής εργασίας είναι η αύξηση της ταχύτητας του συστήματος εφαρμόζοντας κρυφή μνήμη σε αυτό. Αυξάνουμε την ταχύτητα απόκρισης του συστήματος προσφέροντας καλύτερες επίδοσης και εμπειρία χρήσης στους εργαζόμενους του οργανισμού που το χρησιμοποιούν ενώ ταυτόχρονα και το μηχάνημα που φιλοξενεί το σύστημα μπορεί να ανταποκριθεί καλύτερα στις αιτήσεις των χρηστών.

1.4 Δομή της διπλωματικής εργασίας

Στο **κεφάλαιο 2** κάνουμε μία εισαγωγή στο σύστημα διαχείρισης ερευνητικών έργων πάνω στο οποίο ασχολείται αυτή η εργασία, αναφέρουμε τα χαρακτηριστικά του οργανισμού για τον οποίο σχεδιάστηκε το σύστημα ώστε να έχουμε μία εικόνα της δομής και της ιεραρχίας αυτού, στην συνέχεια αναφέρουμε την δομή του συστήματος και τα συστατικά του μέρη, τις οντότητες που περιέχει και τις λειτουργίες τις οποίες σχεδιάστηκε να κάνει.

Στο **κεφάλαιο 3** παρουσιάζουμε το πρώτο σκέλος αυτής της διπλωματικής εργασίας και αφορά την επέκταση του συστήματος διαχείρισης έργων ώστε μέσα από αυτό να γίνει ηλεκτρονική η διαδικασία υποβολής αιτημάτων δαπανών μετακινήσεων καθώς και η έγκριση αυτών. Αρχικά αναφέρουμε τις οντότητες που προστέθηκαν για να καλύψουν τις ανάγκες της πληροφορίας που θέλουμε να απεικονίσουμε για την καινούρια λειτουργία, στην συνέχεια παρουσιάζονται οι λειτουργίες που προστέθηκαν ώστε το σύστημα να είναι ολοκληρωμένο και λειτουργικό. Περιγράφονται τα διαγράμματα ροής της διαδικασίας έγκρισης, καθώς και η αυτόματη παραγωγή τωνεντύπων τα οποία πρέπει να συνοδεύουν την εντολή μετακίνησης.

Το **κεφάλαιο 4** αναφέρεται στις κρυφές μνήμες, κάνουμε μία αναφορά στις κρυφές μνήμες γενικά και επικεντρωνόμαστε στην βιβλιοθήκη κρυφών μνημών Ehcache για την Java την οποία και χρησιμοποιήσαμε. Παρουσιάζεται η λειτουργία μαζί με παράδειγμα χρήσης της βιβλιοθήκης συνδυαζόμενη με το περιβάλλον εργασίας Spring για την αποθήκευση μεθόδων (caching methods), καθώς επίσης και για την υλοποίηση της Ehcache για την δεύτερου επιπέδου κρυφή μνήμη της Hibernate. Τέλος παρουσιάζονται μετρήσεις και διαγράμματα που δείχνουν την επίδοση που επιτεύχθηκε με την χρήση των κρυφών αυτών μνημών.

Τέλος στο **κεφάλαιο 5** παρουσιάζονται τα συμπεράσματα που προέκυψαν με το πέρας αυτής της εργασίας καθώς και η βιβλιογραφία.

ΙΙ. Σύστημα διαχείρισης ερευνητικών έργων

Η διπλωματική εργασία περιστρέφεται γύρω από το πληροφοριακό σύστημα διαχείρισης ερευνητικών έργων ΝΕΔΑ. Πρόκειται για ένα σύστημα σχεδιασμένο να καλύψει τις απαιτήσεις του Ερευνητικό Ακαδημαϊκό Ινστιτούτο τεχνολογίας Υπολογιστών & Εκδόσεων (Ι.Τ.Υ.Ε.) και βρίσκεται σε χρήση από το 2009. Σχεδιασμένο αρχικά από τον Αναστάσιο Σταθόπουλο απόφοιτο της σχολής Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής ο οποίος το ανέπτυξε στα πλαίσια της δικής τους διπλωματικής εργασίας.

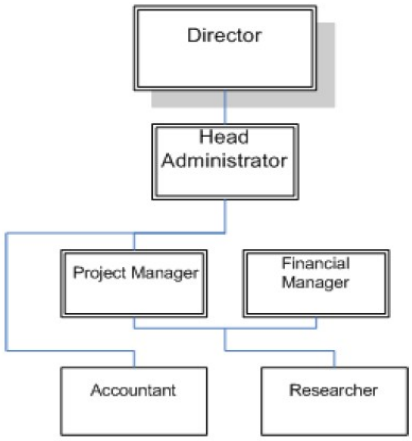
Το παρόν σύστημα πρόκειται για ένα πολύ καλό παράδειγμα πληροφοριακού συστήματος σχεδιασμένου με όλες τα πλέον σύγχρονα εργαλεία ανάπτυξης κώδικα. Είναι γραμμένο σε Java και κάνει ευρεία χρήση του περιβάλλοντος εργασίας Spring και τα modules που αυτό το περιβάλλον προσφέρει για να επεξεργάζεται τα αιτήματα των χρηστών του. Χρησιμοποιεί το περιβάλλον εργασίας Hibernate για υψηλού επιπέδου επικοινωνία του συστήματος με το σύστημα βάσεων δεδομένων που χρησιμοποιεί. Η βάση δεδομένων πρόκειται για μία σχεσιακή βάση δεδομένων MySQL.

Στην συνέχεια θα δούμε αναλυτικότερα τις προδιαγραφές πάνω στις οποίες σχεδιάστηκε το σύστημα, καθώς και μία περίληψη της δομής του οργανισμού για το οποίο σχεδιάστηκε. Επίσης θα παρουσιάσουμε την δομή του συστήματος όπως αναπτύχθηκε, παραδείγματα που δείχνουν πόσο απλουστευμένη και οργανωμένη είναι αυτή η δομή και επομένως πόσο απλή είναι δημιουργία επεκτάσεων αλλά και αλλαγών στο σύστημα. Επίσης θα κάνουμε μία αναφορά στις τεχνολογίες του συστήματος χωρίς να εμβαθύνουμε μιας και δεν είναι αυτός ο κύριος σκοπός της διπλωματικής εργασίας.

2.1 Δομή του οργανισμού και προδιαγραφές συστήματος.

Σε αυτό το σημείο θα δούμε την δομή που έχει ο οργανισμός για τον οποίο σχεδιάστηκε το σύστημα, και για τον οποίο ασχολείται και αυτή η διπλωματική με την επέκταση των δυνατοτήτων του συστήματος.

Στο κατώτερο επίπεδο του οργανισμού είναι οι ερευνητές οι οποίοι εργάζονται στα διάφορα έργα τα οποία έχει αναλάβει ο οργανισμός. Πάνω από αυτούς βρίσκονται οι διαχειριστές των έργων και οι οικονομικοί διαχειριστές των έργων. Οι διαχειριστές των έργων έχουν την ευθύνη να οργανώσουν τα έργα στα οποία είναι διαχειριστές και τους ανθρώπους σε αυτά, ενώ οι οικονομικοί διαχειριστές των έργων ευθύνονται για την σωστή διαχείριση του προϋπολογισμού των έργων που διαχειρίζονται. Πάνω από αυτούς βρίσκεται ο επικεφαλής διαχειριστής, αυτός έχει την ευθύνη της καλής πορείας όλων των έργων του οργανισμού. Στην κορυφή της πυραμίδας βρίσκεται ο διευθυντής. Επίσης υπάρχει και ο λογιστής του οργανισμού ο οποίος είναι υπεύθυνος για όλες τις λογιστικές διαδικασίες.



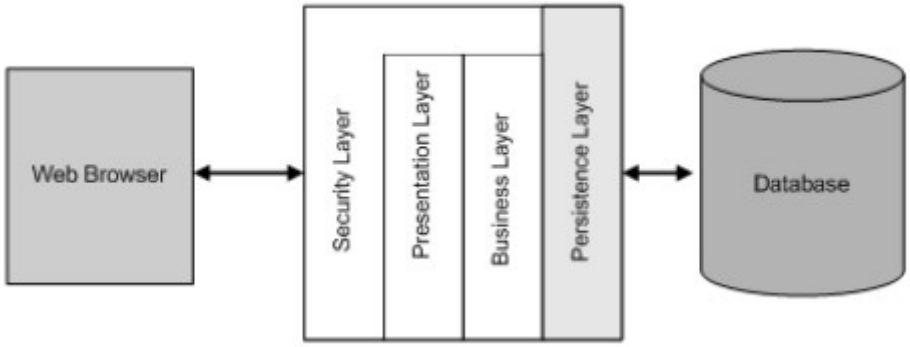
Σχήμα 2.1: Ιεραρχία οργανισμού

Το σύστημα σχεδιάστηκε για να παρέχει συγκεκριμένες δυνατότητες σε αυτούς τους ανθρώπους. Οι βασικές του λειτουργίες είναι η παρακολούθηση των έργων από τους διαχειριστές του, η παρακολούθηση των ερευνητών που εργάζονται στα έργα, των συμβάσεών τους και των ανθρωποωρών που έχουν εργαστεί. Πιο συγκεκριμένα δίνει τις εξής δυνατότητες ανά κατηγορία χρηστών:

- **Ερευνητές (Researchers) :** εμφάνιση όλων των έργων στα οποία εργάζονται, συμπλήρωση φόρμας δελτίου μηνιαίων ωρών απασχόλησης για τα έργα και τους μήνες στα οποία εργάζεται.
- **Διαχειριστής έργου (Project manager) :** όλα τα δικαιώματα των ερευνητών με επιπλέον την ανάθεση ενός ερευνητή σε ένα πακέτο εργασίας, την παρακολούθηση της δέσμευσης πόρων στο έργο. Έχει πλήρη πρόσβαση στα έργα που διαχειρίζεται και βλέπει όλες τις σελίδες και όλα τις αναφορές.
- **Οικονομικός διαχειριστής (Financial Manager) :** Έχει όλες τις δυνατότητες των ερευνητών και των διαχειριστών έργου.
- **Επικεφαλής διαχειριστής (Head Administrator) :** Έχει όλες τις δυνατότητες των ερευνητών και επιπλέον έχει πλήρη πρόσβαση σε όλα τα έργα του οργανισμού.
- **Διευθυντής (Director) :** Έχει όλες τις δυνατότητες των ερευνητών και επιπλέον έχει πλήρη πρόσβαση σε όλα τα έργα του οργανισμού.

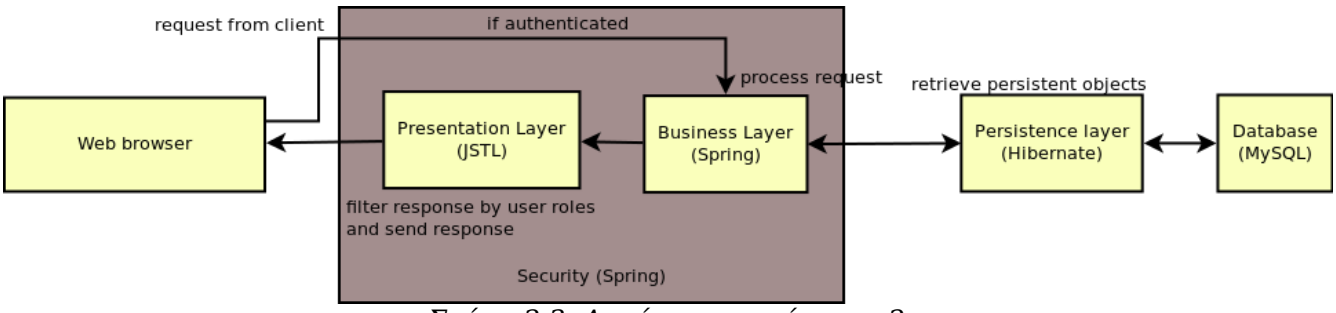
2.2 Δομή του συστήματος

Σε αυτή την ενότητα θα αναλύσουμε την δομή του συστήματος και στη συνέχεια θα παρουσιάσουμε τα εργαλεία με τα οποία επιτεύχθηκε αυτή η δομή. Η ΝΕΔΑ είναι ένα κατανεμημένο σύστημα το οποίο βασίζεται στο μοντέλο πελάτη-εξυπηρετητή (client-server model). Στην εικόνα που ακολουθεί φαίνεται αναλυτικά η δομή αυτού του συστήματος καθώς και τα επιμέρους μέρη που το αποτελούν.



Σχήμα 2.2: Δομή του συστήματος

Όπως παρατηρούμε στην παραπάνω εικόνα το σύστημα αποτελείται από τα εξής διαφορετικά επίπεδα, μία βάση δεδομένων για την αποθήκευση της πληροφορίας. Το επίπεδο επικοινωνίας με την βάση δεδομένων το οποίο το υλοποιεί η Hibernate, το επίπεδο λειτουργιών (business layer) το οποίο αναλαμβάνει την εξυπηρέτηση των αιτημάτων που πραγματοποιούν οι χρήστες υλοποιεί τις βασικά ερωτήματα που χρειάζονται για επικοινωνία με τη βάση δεδομένων μέσω των κλάσεων διαχειριστών (managers) οι οποίοι υλοποιούν ερωτήματα ανάγνωσης δεδομένων στη βάση, εγγραφής και ανανέωσης. Στην συνέχεια είναι το επίπεδο παρουσίασης το οποίο τροφοδοτείται από το επίπεδο λειτουργίας με πληροφορίες τις οποίες αναλαμβάνει να τις εμφανίσει στον χρήστη σε μορφή αναγνώσιμη από τον φυλλομετρητή. Στην συνέχεια υπάρχει το επίπεδο ασφαλείας το οποίο ελέγχει αν ο συγκεκριμένος χρήστης που κάνει το αίτημα στο σύστημα έχει το δικαίωμα να πραγματοποιήσει το συγκεκριμένο αίτημα. Μόνο αν έχει ο έλεγχος μεταφέρεται στα παρακάτω επίπεδα αλλιώς το αίτημα απορρίπτεται. Τέλος είναι το επίπεδο του φυλλομετρητή μέσω του οποίου ο χρήστης πραγματοποιεί αιτήματα.



Σχήμα 2.3: Δομή του συστήματος 2

Στην παραπάνω εικόνα φαίνεται αναλυτικότερα η δομή του συστήματος και το πως αυτό αποκρίνεται σε ένα αίτημα από έναν πελάτη. Ο πελάτης στέλνει ένα αίτημα, για παράδειγμα η προβολή ενός έργου. Το σύστημα μόλις λάβει το αίτημα θα εξετάσει αν ο συγκεκριμένος χρήστης έχει δικαίωμα να προβάλει την συγκεκριμένη σελίδα. Αν έχει δικαίωμα τότε ο αρμόδιος για αυτό το αίτημα Controller της Spring θα αναλάβει δράση και θα εκτελέσει τις εντολές που χρειάζονται για να εξυπηρετηθεί το αίτημα. Αν η εξυπηρέτηση του αιτήματος χρειάζεται συναλλαγές με τη βάση, στο παράδειγμά μας θα χρειαστεί να να ανακτήσουμε το έργο το οποίο ζήτησε ο χρήστης να δει, τότε ο Controller ζητάει από την Hibernate να πραγματοποιήσει αυτές τις συναλλαγές. Στην συνέχεια ο έλεγχος γυρίζει στον Controller ο οποίος δίνει τις πληροφορίες στο επίπεδο παρουσίασης που χρειάζεται για να επιστρέψει την απάντηση στον πελάτη. Στο επίπεδο προβολής επίσης τα αποτελέσματα μπορούν να υποστούν φιλτράρισμα αναλόγως τα δικαιώματα που έχει ο χρήστης πάνω στο συγκεκριμένο αίτημα. Στην συνέχεια τα αποτελέσματα επιστρέφονται σε μορφή αναγνώσιμη από τον φυλλομετρητή στον πελάτη.

Στις επόμενες ενότητες αυτού του κεφαλαίου θα δούμε πως αυτά τα επίπεδα υλοποιήθηκαν στο συγκεκριμένο σύστημα αναλυτικότερα.

2.2.1 Σχεσιακή βάση δεδομένων – Database

Η σχεσιακή βάση δεδομένων που χρησιμοποιείται είναι η MySQL στην έκδοση 5.5. Για την σύνδεση και την πραγματοποίηση ερωτημάτων και ενημερώσεων στη βάση χρησιμοποιήθηκε η HSQLDB (Hyper Structured Query Language Database). Πρόκειται για μία SQL μηχανή σχεσιακής βάσης δεδομένων η οποία είναι γραμμένη σε Java και περιλαμβάνει τον JDBC οδηγό.

Στην συνέχεια ακολουθούν οι οντότητες της σχεσιακής βάσης όπως σχεδιάστηκαν για να καλύψουν τις ανάγκες του συστήματος.

Χρήστες – Users

Περιγράφει τους χρήστες του συστήματος και έχει τις εξής ιδιότητες:

- Όνομα: το όνομα του χρήστη.

- Επώνυμο: το επώνυμο του χρήστη.
- Email: η ηλεκτρονική διεύθυνση αλληλογραφίας του χρήστη η οποία χρησιμοοποιείται και ως συνθηματικό για την είσοδό του στο σύστημα.
- Κωδικός πρόσβασης : Μαζί με το email επιτρέπουν την είσοδο του χρήστη στο σύστημα.
- Λίστα από έργα/ρόλοι : Για κάθε έργο στο οποίο ανήκει ο χρήστης δείχνει τα δικαιώματα του χρήστη σε αυτό το έργο.
- Rate: το Rate του χρήστη βάση του οποίου υπολογίζονται οι μηνιαίες αποδοχές.

Ερευνητικά έργα - Projects

Τα ερευνητικά έργα που διαχειρίζεται ο οργανισμός, έχουν τις εξής ιδιότητες:

- όνομα: το πλήρες όνομα του έργου.
- Σύντομο όνομα: το σύντομο όνομα του έργου το οποίο χρησιμοποιείται σε φόρμες για την γρήγορη αναφορά σε αυτό το έργο.
- Κατηγορία: η κατηγορία στην οποία ανήκει το έργο. Ο κωδικός του και η μονάδα στην οποία ανήκει.
- Ημερομηνία έναρξης: Η ημερομηνία έναρξης του έργου.
- Ημερομηνία λήξης: Αντίστοιχα η ημερομηνία λήξης του έργου.
- Website: πρόκειται για την ηλεκτρονική σελίδα του έργου, αν έχει.
- Υπεύθυνος έργου : ο χρήστης που είναι υπεύθυνος για όλες τις διαδικασίες του έργου όπως διαχείριση προσωπικού και ανάθεση ερευνητών σε πακέτα εργασίας.
- Οικονομικός διαχειριστής : ο χρήστης που διαχειρίζεται τα οικονομικά του έργου.
- Ερευνητική ομάδα : μία λίστα με τους ερευνητές που εργάζονται στο συγκεκριμένο έργο.
- Πακέτα εργασίας : Το έργο διαχωρίζεται σε πακέτα εργασίας, μικρότερα δηλαδή τμήματα εργασίας που πρέπει να γίνουν για να ολοκληρωθεί το συνολικό έργο και έτσι κατανέμεται καλύτερα η συνολική εργασία και οι εργαζόμενοι.

Πακέτα εργασίας – Work packages

Τα πακέτα εργασίας επημερίζουν το έργο σε μικρότερες δραστηριότητες και έχουν τις εξής ιδιότητες:

- Όνομα : το όνομα του πακέτου εργασίας.
- Συνολικοί ανθρωπομήνες: Το ποσό της εργασίας που θα δαπανηθεί σε αυτό το πακέτο εργασίας.
- Μήνας έναρξης: Ο αριθμητικός μήνας του έργου από τη στιγμή που που ξεκίνησε το έργο και εκφράζει την έναρξη του πακέτου εργασίας.
- Μήνας λήξης: αντίστοιχα είναι ο αριθμητικός μήνας του έργου από τη στιγμή που ξεκίνησε και εκφράζει το πότε θα λήξη το πακέτο εργασίας.

Μηνιαίες ώρες απασχόλησης - Timesheets

Τα timesheets αφορούν τις ώρες που θα δουλέψει ο εργαζόμενος έναν συγκεκριμένο μήνα σε ένα συγκεκριμένο έργο, και στο τέλος του μήνα ο χρήστης συμπληρώνει πόσες

ώρες δούλεψε καθημερινά. Οι ώρες που συμπληρώνει ο χρήστης σε άθροισμα θα πρέπει να συμφωνούν με τις ώρες που του ανατέθηκαν. Έχει τις εξής ιδιότητες:

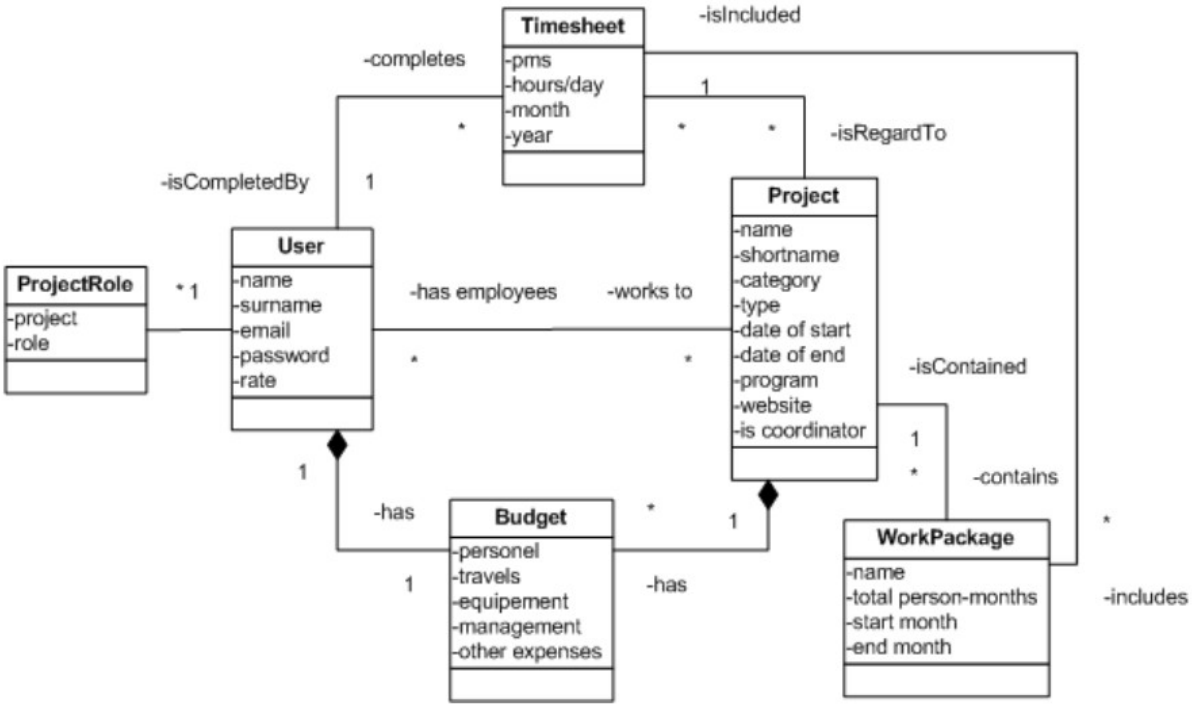
- Χρήστης: είναι ο χρήστης τον οποίο αφορά το timesheet, πόσες ώρες θα δουλέψει και πως τις δούλεψε κατά την διάρκεια του μήνα.
- Έργο : το έργο στο οποίο εργάζεται ο χρήστης και αφορά το timesheet.
- Ώρες ανά ημέρα ανά πακέτο εργασίας : οι ώρες που εισήγαγε ο χρήστης ότι εργάστηκε κάθε μέρα.
- Χρονολογία: Ο μήνας και το έτος για το οποίο συμπληρώνεται το timesheet.

Προϋπολογισμός – budget

Ένας προϋπολογισμός έχει τις παρακάτω ιδιότητες:

- Αμοιβές προσωπικού: Το ποσό του προϋπολογισμού που προορίζεται για τις αμοιβές των ερευνητών ενός έργου.
- Ταξίδια: Το ποσό του προϋπολογισμού που προορίζεται να καλύψει τις ανάγκες των ταξιδιών που θα γίνουν στα πλαίσια αυτού του έργου.
- Εξοπλισμός: Το ποσό του προϋπολογισμού που προορίζεται για την αγορά εξοπλισμού που χρειάζεται ένα έργο.
- Management: Το ποσό του προϋπολογισμού για την αμοιβή των διαχειριστών του έργου
- Λοιπά έξοδα: Το ποσό του προϋπολογισμού για όλα τα υπόλοιπα έξοδα που μπορεί να έχει το συγκεκριμένο έργο.

Στην συνέχεια ακολουθεί το διάγραμμα οντοτήτων συσχετίσεων όπως δημιουργείται από τις παραπάνω περιγραφές.



Σχήμα 2.4: Διάγραμμα οντοτήτων συσχετίσεων

2.2.2 Επίπεδο αποθήκευσης – persistence layer

Αυτό το επίπεδο μεσολαβεί μεταξύ της βάσης δεδομένων και του συστήματος, αποθηκεύει αντικείμενα απευθείας στη βάση και τραβάει από τη βάση δίνοντάς τα στο ανώτερο επίπεδο ως κανονικά αντικείμενα. Χρησιμοποιήθηκε η Hibernate στην έκδοση 3.2.0.ga.

Η Hibernate είναι μία Object-relational mapping (ORM) βιβλιοθήκη για την Java η οποία δίνει την δυνατότητα στους προγραμματιστές να αντιστοιχίσουν αντικείμενα του προγράμματός τους σε πίνακες της βάσης δεδομένων, ενώ εκτός από την MySQL υποστηρίζει όλες τις γνωστές βάσεις δεδομένων. Χρησιμοποιώντας την Hibernate μπορούμε να αλληλεπιδρούμε με βάσεις δεδομένων σε ένα πιο υψηλό προγραμματιστικό επίπεδο χωρίς να πέφτουμε στο επίπεδο της ίδιας της βάσης δεδομένων, καθώς όλη την επικοινωνία την αναλαμβάνει η Hibernate κρύβοντας της λεπτομέρειες από τον προγραμματιστή.

Η αντιστοιχία των αντικείμενων με την βάση γίνεται μέσω ρυθμίσεων που παρέχονται σε XML αρχεία ή με χρήση Annotations. Αφού έχουν οριστεί οι αντιστοιχίες αυτές η Hibernate αναπαράγει το σχήμα της βάσης ενώ υποστηρίζει και συσχετίσεις μεταξύ αντικειμένων τύπου one-to-many και many-to-many. Επίσης δίνει την δυνατότητα στον προγραμματιστή να ορίσει τους δικούς του τύπους δεδομένων και να τους κάνει αντιστοιχίσει σε τύπους βάσης δεδομένων. Η τελευταία δυνατότητα κάνει δυνατή την αλλαγή του προεπιλεγμένου SQL τύπου δεδομένων που ορίζει η Hibernate, την αντιστοιχία τιμών από Enum σε στήλες σαν να είναι κανονικές μεταβλητές, την αντιστοιχία μιας μεταβλητής σε περισσότερες στήλες. Επιπλέον προσφέρει στον

προγραμματιστή την δυνατότητα να κάνει queries στη βάση με χρήση Κριτηρίων (Criteria) σε υψηλό επίπεδο, ενώ υποστηρίζει και γλώσσα ερωτημάτων αντίστοιχη της SQL η οποία ονομάζεται Hibernate Query Language (HQL).

Στο συγκεκριμένο σύστημα η Hibernate ρυθμίστηκε ώστε να χρησιμοποιεί τον οδηγό της HSQLDB για να συνδέεται με τη βάση. Ενώ η ρύθμιση της Hibernate και των session και transactions γίνεται από το περιβάλλον εργασίας Spring.

2.2.3 Επίπεδο λειτουργιών – business layer

Είναι το επίπεδο στο οποίο αναπτύσσεται η λογική για την εξυπηρέτηση κάθε αιτήματος από τους χρήστες. Συγκεκριμένα για αυτό το επίπεδο χρησιμοποιείται η Spring. Όταν ένα αίτημα καταφθάνει αυτό περνάει μέσα από τον Dispacher Servlet, το αίτημα χαρακτηρίζεται από το URL και τις παραμέτρους. Ο dispatcher συμβουλεύεται τον HandlerMapping που έχουμε ορίσει για να βρεθεί ο κατάλληλος Controller ο οποίος περιέχει την λογική για την εξυπηρέτηση του αιτήματος. Επίσης σε αυτό το επίπεδο υπάρχουν και οι διαχειριστές, κλάσεις οι οποίες υλοποιούν όλα τα ερωτήματα που χρειάζεται το σύστημα για να επικοινωνεί με την βάση και χρησιμοποιούν το περιβάλλον εργασίας της Hibernate.

Για το επίπεδο αυτό χρησιμοποιήθηκε το Module της Spring Inversion of Control. Αποτελεί την κύρια λειτουργία του Spring. Επιτρέπει την διαμόρφωση και διαχείριση αντικειμένων Java χρησιμοποιώντας refletion. Reflection είναι η δυνατότητα μίας εφαρμογής να ελέγχει και να αλλάζει την δομή και τη συμπεριφορά ενός αντικειμένου κατά την εκτέλεση και όχι κατά το compile. Το module είναι υπεύθυνο να διαχειριστή τον κύκλο ζωής των αντικειμένων. Δημιουργεί τα αντικείμενα, κάνει την αρχικοποίησή τους και συνδέει αντικείμενα μεταξύ τους.

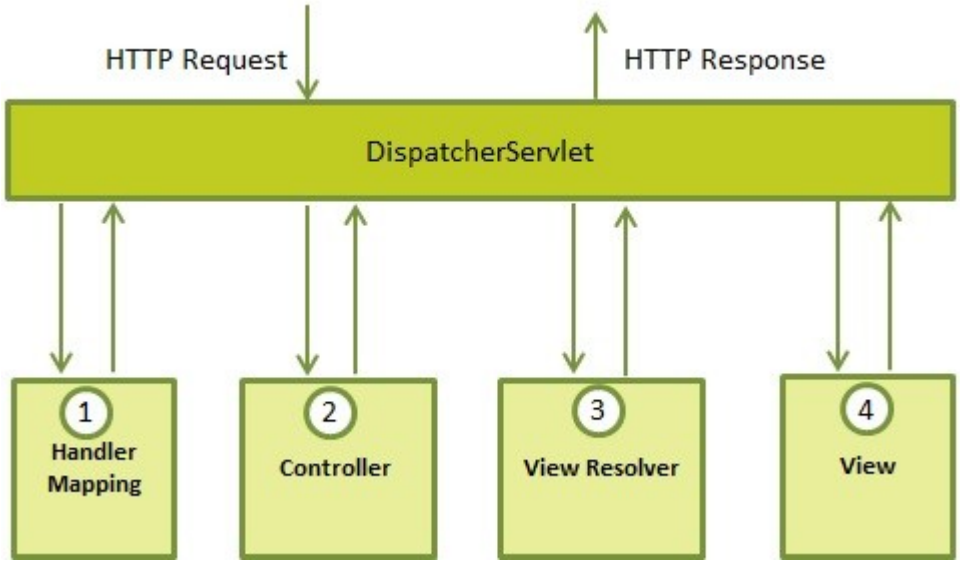
Τα αντικείμενα που δημιουργήθηκαν από το container επίσης αποκαλούνται “Managed Objects” ή “Beans”. Το container ρυθμίζεται χρησιμοποιώντας XML αρχεία τα οποία περιέχουν τις αναφορές στα Beans οι οποίες φέρουν και τις πληροφορίες που χρειάζονται για την δημιουργία αυτών των αντικειμένων. Τα αντικείμενα μπορούν να χρησιμοποιηθούν στη συνέχεια είτε με Εξαρτημένη αναζήτηση (Dependecy lookup) ή Εξαρτημένη Εισαγωγή (Dependecy injection). Με χρήση Εξαρτημένης αναζήτησης ο ενδιαφερόμενος ζητάει από το container για το αντικείμενο που τον ενδιαφέρει με βάση το συγκεκριμένο όνομα αυτού του αντικειμένου ή τον συγκεκριμένο τύπο του αντικειμένου. Με χρήση Εξαρτημένης εισαγωγής, το container δίνει απευθείας τα αντικείμενα με βάση το όνομα τους σε άλλα αντικείμενα που θα τα χρειαστούν, είτε με χρήση του δημιουργού (constructor) της κλάσης, είτε με χρήση factory methods.

Αν και υπάρχει η δυνατότητα να χρησιμοποιήσει κανείς το Spring framework για άλλες λειτουργίες χωρίς να κάνει χρήση του συγκεκριμένου module, η χρήση του Inversion of Control container έχει την τάση να κάνει το πρόγραμμα πιο κατανοητό, πιο οργανωμένο και ευκολότερα συντηρήσιμο και επεκτάσιμο. Ενώ μπορεί να χρησιμοποιηθεί για μικρές εφαρμογές μεχρι μεγάλες enterprise εφαρμογές.

Το Module αυτό έχει την τάση να κάνει την ανάπτυξη της λογικής του συστήματος απλή και μεθοδευμένη. Ο τρόπος με τον οποίο εξυπηρετούνται τα αιτήματα των χρηστών είναι ξεκάθαρος και οργανωμένος, και κάνει την επέκταση ενός τέτοιου συστήματος αρκετά απλή υπόθεση αφήνοντας τον προγραμματιστή να ασχοληθεί αποκλειστικά με το πρόβλημα της εξυπηρέτησης ενός αιτήματος παρά με την συνοχή της αρχιτεκτονικής του συστήματος γενικότερα.

2.2.4 Επίπεδο παρουσίασης – presentation layer

Πρόκειται για το επίπεδο το οποίο αναλαμβάνει να παρουσιάσει το αποτέλεσμα ενός αιτήματος στον χρήστη. Αυτό το επίπεδο χρησιμοποιεί το module SpringMVC. Όπως έχουμε περιγράψει και πιο πάνω το αίτημα περνάει μέσα από τον Dispatcher Servlet, στο servlet έχουμε ορίσει τα handler mappings που θα χρησιμοποιηθούν για τις αιτήσεις. Σε αυτά τα mappings υπάρχει η αντιστοιχία του αιτήματος με τον Controller που θα εξυπηρετήσει το αίτημα. Μόλις ο Controller ολοκληρώσει επιστρέφει πίσω στον DispatcherServlet ένα View μαζί με τα δεδομένα της επεξεργασίας. Στην συνέχεια ο DispatcherServlet αναζητά το View με το ίδιο όνομα που επέστρεψε ο Controller. Μόλις το βρει αυτό το View μαζί με τα δεδομένα που έδωσε ο Controller συνθέτουν το αποτέλεσμα που βλέπει ο χρήστης στην σελίδα του.



Σχήμα 2.5: Διαδικασία εξυπηρέτησης ενός αιτήματος

2.2.5 Επίπεδο ασφάλειας

Πρόκειται για το επίπεδο που εξασφαλίζει όπως περιγράψαμε στην αρχή αυτής της ενότητας την ασφάλεια σε δύο επίπεδα, πρώτον ασφάλεια στο επίπεδο της επεξεργασίας ενός αιτήματος, ελέγχοντας ότι ένας χρήστης έχει όντος δικαίωμα να πραγματοποιήσει ένα συγκεκριμένο αίτημα, και δεύτερον στην προβολή πληροφοριών, ελέγχοντας ότι ο χρήστης μπορεί να δει συγκεκριμένες πληροφορίες και πραγματοποιώντας κατάλληλο φιλτράρισμα στην οθόνη που βλέπει ο χρήστης. Για το

επίπεδο αυτό χρησιμοποιήθηκε το Spring Security.

Ο τρόπος με τον οποίο δουλεύει το SpringSecurity είναι αρκετά απλός. Έχει ρυθμιστεί με μία αντιστοιχία από URIs και δικαιώματα για κάθε URI. Μόλις ένας χρήστης συνδεθεί στο σύστημα τότε το SpringSecurity τραβάει από τη βάση όλα τα δικαιώματα αυτού του χρήστη στο σύστημα. Μόλις ένα αίτημα καταφθάσει στο σύστημα από αυτόν τον χρήστη περνάει μέσα από το SpringSecurity. Αν ο χρήστης που το έκανε έχει κάποιο από τα δικαιώματα που σημειώθηκε στο Configuration του SpringSecurity τότε ο χρήστης εξυπηρετείτε, διαφορετικά δεν μπορεί να έχει πρόσβαση. Επίσης στην κατάσταση που το response συνθέτεται μέσω του View που επέστρεψε ο Controller και των δεδομένων υπάρχει επιλογή μέσω κατάλληλων JSTL tags να φιλτραριστεί η απάντηση που θα λάβει ο χρήστης με βάση αυτά που επιτρέπεται να βλέπει.

2.3 Λειτουργίες του συστήματος

Το σύστημα αυτό όπως είπαμε σχεδιάστηκε για να ικανοποιήσει κάποιες ανάγκες του οργανισμού ΕΑΙΤΥ. Οι λειτουργίες οι οποίες σχεδιάστηκε να εκτελεί είναι οι εξής:

- Εισαγωγή ενός νέου χρήστη: Αφορά την εισαγωγή ενός νέρου χρήστη στο σύστημα. Πραγματοποιείται μόνο από τους Head Administrator και System Administrator. Ο διαχειριστής μπαίνει στην φόρμα εισαγωγής ενός καινούριου ατόμου και συμπληρώνει τα στοιχεία του, αυτά είναι το όνομα, το επώνυμο, το rate του, το email του, ο κωδικός πρόσβασης που θα χρησιμοποιεί.
- Επεξεργασία ενός χρήστη: Αφορά την επεξεργασία των στοιχείων ενός χρήστη που έχει ήδη εισαχθεί στο σύστημα. Πραγματοποιείται από τους Head Administrator και System Administrator. Ο διαχειριστής αφού εισέλθει στην φόρμα μπορεί να αλλάξει οποιαδήποτε από τα στοιχεία του χρήστη.
- Εισαγωγή έργου: Αφορά την εισαγωγή ενός νέου ερευνητικού έργου στο σύστημα. Πραγματοποιείται από τους Head Administrator και System Administrator. Ο διαχειριστής μπαίνει στη φόρμα εισαγωγής και εισάγει ένα νέο έργο στο σύστημα. Τα στοιχεία που εισάγει είναι το όνομα του έργου, το σύντομο όνομα του έργου, η ημερομηνία έναρξης και λήξης, ο τύπος του έργου, το πρόγραμμα του έργου, το website, τον διαχειριστή του έργου, τον οικονομικό διαχειριστή, την ερευνητική μονάδα και τα πακέτα εργασίας.
- Προσθήκη ερευνητών σε έργο: αφορά την εισαγωγή υπαρχόντων χρηστών στο σύστημα ως ερευνητές σε ένα συγκεκριμένο έργο. Πραγματοποιείται από τους Head Administrator και System Administrator.
- Προβολή συγκεκριμένου έργου: Προβάλλονται στην οθόνη λεπτομέρειες που αφορούν ένα συγκεκριμένο έργο όπως τα στοιχεία του έργου, τα μέλη της ερευνητικής ομάδας του έργου, τα πακέτα εργασίας.
- Καθορισμός person-months ερευνητή: Αφορά το πλάνο των ωρών που θα εργαστεί ένας συγκεκριμένος ερευνητής. Πραγματοποιείται από τον υπεύθυνο κάθε έργου, ο υπεύθυνος επιλέγει τον ερευνητή και τον μήνα στον οποίο θέλει να του αναθέσει τις ώρες. Στην συνέχεια εμφανίζεται μία φόρμα στην οποία

φαίνεται κάθε πακέτο εργασίας που είναι διαθέσιμο στο συγκεκριμένο μήνα, και για κάθε πακέτο συμπληρώνει πόσες ώρες θα εργαστεί ο ερευνητής.

- Κατάθεση timesheet: Αφορά την συμπλήρωση των ωρών ανά ημέρα που εργάστηκε ο ερευνητής. Πραγματοποιείται από όλους τους ερευνητές. Εισέρχονται στη φόρμα συμπλήρωσης του timesheet όπου για κάθε πακέτο εργασίας του έργου στο οποίο έχουν ώρες να εργαστούν, συμπληρώνουν για κάθε ημέρα του μήνα πόσες ώρες εργάστηκαν. Ο συνολικός αριθμός ωρών για κάθε πακέτο εργασίας πρέπει να συμφωνεί με το πλάνο που έγινε.
- Παρακολούθηση πακέτων εργασίας: Αφορά την παρακολούθηση της εξέλιξης του έργου με βάση τα πακέτα εργασίας. Δικαιώματα έχουν μόνο οι διαχειριστές του έργου και όχι οι απλοί ερευνητές. Μέσω αυτής της λειτουργίες εμφανίζονται οι μήνες που το έργο είναι ενεργό και για κάθε μήνα εμφανίζονται τα person-months τα οποία πλαναρίστηκαν από τον διαχειριστή του έργου μαζί με τα πραγματικά person-months που δήλωσαν οι ερευνητές ότι εργάστηκαν.

III. Σύστημα διαχείρισης ταξιδιών

Σε αυτό το κεφάλαιο θα ασχοληθούμε με μία από τις δύο κύριες συνεισφορές αυτής της διπλωματικής εργασίας. Θα επεκτείνουμε τις δυνατότητες του πληροφοριακού συστήματος που περιγράψαμε στο προηγούμενο κεφάλαιο. Η επέκταση που θα πραγματοποιήσουμε είναι η ψηφιοποίηση της διαδικασίας υποβολής και έγκρισης εντολής μετακίνησης και απόδοσης κατά την πραγματοποίηση ενός ταξιδιού για τον οργανισμό Ι.Τ.Υ.Ε..

Ο συγκεκριμένος οργανισμός ασχολείται με πολλά ερευνητικά έργα πολλά από αυτά ευρωπαϊκά και συνεργάζεται με άλλους οργανισμούς του εξωτερικού. Στα πλαίσια τέτοιων συνεργασιών οι εργαζόμενοι σε αυτά τα έργα συχνά καλούνται να ταξιδέψουν ώστε να έρθουν σε επαφή με συνεργάτες. Επίσης η παρακολούθηση συνεδρίων ή η παρουσίαση εργασιών είναι συχνοί λόγοι για τους οποίους ερευνητές πραγματοποιούν ταξίδια στα πλαίσια ενός τέτοιου οργανισμού. Ωστόσο για να πάει κάποιος ταξίδι χρειάζεται μία διαδικασία έγκρισης αυτής της πράξης. Σε αυτό το κεφάλαιο θα δούμε αυτή την διαδικασία και θα δούμε τα προβλήματα που έχει και την λύση που υλοποιήθηκε για αυτή την διπλωματική.

3.1 Ταξίδια – το πρόβλημα και η πρόταση λύσης

Για να μπορέσει ένας εργαζόμενος στον οργανισμό να πραγματοποιήσει ένα ταξίδι στα πλαίσια της δουλειάς του, μέχρι πρότινος η διαδικασία περιλάμβανε την συμπλήρωση ενός εντύπου όπου ο ταξιδιώτης συμπλήρωνε στοιχεία σχετικά με το ταξίδι που θέλει να πραγματοποιήσει. Αυτά τα στοιχεία περιλαμβάνουν γενικές πληροφορίες του ταξιδιού όπως προορισμός και ημερομηνίες, καθώς και τα έξοδα του ταξιδιού τα οποία είναι χωρισμένα σε κατηγορίες που θα αναλύσουμε στην συνέχεια. Αυτό το χαρτί στην συνέχεια έπρεπε να πάρει υπογραφή από τον διαχειριστή του έργου στο οποίο εργάζεται ο ταξιδιώτης, στη συνέχεια από τον οικονομικό διαχειριστή του έργου και από τον επικεφαλή διαχειριστή. Στην συνέχεια αφότου το ταξίδι εγκρινόταν και πραγματοποιούταν ο χρήστης έπρεπε να συμπληρώσει το έντυπο της απόδοσης του ταξιδιού στο οποίο αναφέρει τα έξοδα τα οποία είχε το ταξίδι παραδίδοντας ταυτόχρονα και τις αντίστοιχες αποδείξεις. Το έντυπο αυτό πάλι πρέπει να πάρει υπογραφές από τον λογιστή του οργανισμού και τον οικονομικό διαχειριστή του έργου. Στη συνέχεια τα δύο έντυπα μαζί με τις αποδείξεις αποθηκεύονται μαζί από τον λογιστή.

Σε αυτό το σενάριο υπάρχουν πολλά προβλήματα. Ένα βασικό πρόβλημα είναι η διαθεσιμότητα των ανθρώπων. Κάθε άνθρωπος που εμπλέκεται στο ταξίδι πρέπει να είναι παρόν για να δώσει την υπογραφή του, επίσης αυτό γίνεται ακόμα πιο πολύπλοκο αν το ταξίδι απορριφθεί σε μεταγενέστερο βήμα και πρέπει να ενημερωθούν για αυτό όλοι οι ενδιαφερόμενοι. Ένα ακόμα πρόβλημα είναι η παρακολούθηση των ταξιδιών. Για να παρακολουθήσεις ένα ταξίδι πρέπει να ψάξεις σε ένα μεγάλο πάκο με χαρτιά. Επίσης αν θες να δεις όλα τα ταξίδια που πραγματοποίησε ένας άνθρωπος θα πρέπει να αναζητήσεις όλα αυτά τα ταξίδια ένα ένα, αντίστοιχα αν θες να δεις όλα τα ταξίδια που πραγματοποιήθηκαν σε ένα έργο.

Πάνω στα προβλήματα που παρουσιάστηκαν θα σχεδιάσουμε ένα σύστημα

έγκρισης εντολής μετακίνησης και απόδοσης το οποίο θα επιτρέπει την αποθήκευση (upload) αποδείξεων και άλλων πιστοποιητικών απαραίτητων για το ταξίδι, θα κάνουμε ευκολότερη την παρακολούθηση των ταξιδιών ανά άνθρωπο και ανά έργο, θα αυτοματοποιήσουμε διαδικασίες όπως την εξαγωγή των εντύπων σε μορφή PDF η οποία στη συνέχεια μπορεί να εκτυπωθεί και τέλος θα υλοποιήσουμε την αποστολή μηνυμάτων ηλεκτρονικού ταχυδρομείου (e-mails) για την ενημέρωση των εμπλεκόμενων για την κατάσταση ενός συγκεκριμένου ταξιδιού.

Στην επόμενη ενότητα θα αναλύσουμε τις οντότητες που δημιουργήσαμε για το σύστημα και στην μεθεπόμενη θα δούμε την λειτουργικότητα.

3.2 Οντότητες συστήματος

Στην ενότητα αυτή θα παρουσιάσουμε της οντότητες του συστήματος, τα αντικείμενα δηλαδή τα οποία χρησιμοποιούμε για να αναπαραστήσουμε την απαραίτητη πληροφορία που χρειαζόμαστε. Αυτές οι οντότητες έρχονται να προστεθούν στις υπάρχουσες και να αλληλεπιδράσουν με αυτές.

Ταξίδι – Travel

Το ταξίδι είναι η βασική οντότητα του ταξιδιού και συγκεντρώνει τις βασικές πληροφορίες για ένα ταξίδι, οι ιδιότητες που έχει είναι οι εξής:

- **Αναγνωριστικό του ταξιδιού:** ο κωδικός που χρησιμοποιείται από την βάση δεδομένων και την Hibernate.
- **Κωδικός λογιστηρίου:** Είναι ο κωδικός της αντίστοιχης εγγραφής στη βάση του λογιστηρίου για το συγκεκριμένο ταξίδι.
- **Κωδικός λογιστηρίου προκαταβολής:** είναι ο κωδικός της αντίστοιχης εγγραφής στη βάση του λογιστηρίου για το κόστος της προκαταβολής για το ταξίδι.
- **Κωδικός λογιστηρίου εξόδων ταξιδιώτη:** είναι ο κωδικός της αντίστοιχης εγγραφής στη βάση του λογιστηρίου για το κόστος του ταξιδιώτη για το ταξίδι.
- **Περιγραφή:** Είναι το κείμενο στο οποίο ο ταξιδιώτης περιγράφει το σκοπό του ταξιδιού. Στη βάση αποθηκεύεται ως Text.
- **Σύνδεσμος:** πρόκειται για έναν σύνδεσμο ο οποίος μπορεί να περιέχει πληροφορίες σχετικά με το αντικείμενο του ταξιδιού, για παράδειγμα αν πρόκειται για συνέδριο μπορεί να είναι σύνδεσμος στη σελίδα του συνεδρίου.
- **Λίστα από αρχεία PDF που περιλαμβάνουν πληροφορίες για το ταξίδι:** Τα αρχεία αποθηκεύονται στο σύστημα όταν ο χρήστης τα κάνει upload, αυτό το πεδίο περιέχει μία λίστα με τα ονόματα όλων των σχετικών αρχείων.
- **Λίστα από αρχεία PDF που περιλαμβάνουν πληροφορίες για το πρόγραμμα του ταξιδιού:** Τα αρχεία αποθηκεύονται στο σύστημα όταν ο χρήστης τα κάνει upload, αυτό το πεδίο περιέχει μία λίστα με τα ονόματα όλων των σχετικών αρχείων.
- **Αρχείο PDF με την τελική Agenda για το ταξίδι.**

- **Πιστοποιητικό:** πρόκειται για αρχείο PDF που απαιτείται όταν ο ταξιδιώτης ταξίδεψε πριν εγκριθεί το ταξίδι και πιστοποιεί ότι είχε την έγκριση του διαχειριστή του έργου.
- **Τύπος:** Ο τύπος του ταξιδιού, οι τιμές που παίρνει είναι προεπιλεγμένες και μπορεί να είναι Conference, Excibition, School, Meeting, Other.
- **Αφετηρία:** πρόκειται για την αφετηρία του ταξιδιού, από που ξεκινάει ο χρήστης, περιγράφεται από το όνομα της χώρας και της πόλης.
- **Προορισμός:** πρόκειται για τον προορισμό του ταξιδιού, όμοια περιγράφεται από το όνομα της χώρας και της πόλης.
- **Ημερομηνία υποβολής :** Πρόκειται για την ημερομηνία στην οποία ο ταξιδιώτης υπέβαλε την αίτηση μετακίνησης.
- **Ημερομηνία απόδοσης :** Πρόκειται για την ημερομηνία στην οποία ο ταξιδιώτης συμπλήρωση και υπέβαλε την απόδοση.
- **Διεύθυνση προορισμού:** Είναι η ακριβής διεύθυνση του προορισμού.
- **Ημερομηνία αναχώρησης :** Κάθε ταξίδι έχει ημερομηνία αναχώρησης, πρόκειται για την μέρα που ο ταξιδιώτης ξεκινάει το ταξίδι. Στην βάση αποθηκεύεται ως Timestamp ενώ στην Java γίνεται αντιστοίχηση με το αντικείμενο java.util.Date.
- **Ημερομηνία επιστροφής :** Κάθε ταξίδι έχει και ημερομηνία επιστροφής, πρόκειτα για την ημερομηνία που ο ταξιδιώτης επιστρέφει. Όμοια με την ημερομηνία αναχώρησης στη βάση αποθηκεύεται ως Timestamp και στην Java γίνεται αντιστοίχηση σε αντικείμενο java.util.Date.
- **Ταξιδιώτης :** Πρόκειται για τον χρήστη ο οποίος θα πραγματοποιήσει το ταξίδι. Στη βάση αποτελεί ξένο κλειδί στον πίνακα των χρηστών (USERS) ενώ στην JAVA γίνεται απευθείς αντιστοίχηση στον χρήστη.
- **Έργο :** Κάθε ταξίδι στον οργανισμό χρεώνεται σε ένα έργο. Τα έργα είναι ήδη υλοποιημένα στο σύστημα και όπως συμβαίνει με τον χρήστη, στη βάση η εγγραφή αυτή είναι ένα ξένο κλειδί στον πίνακα των έργων (PROJECTS) ενώ στην Java γίνεται απευθείας αναφορά στο έργο.
- **Λίστα δαπανών :** πρόκειται για τις δαπάνες του ταξιδιού, ο αριθμός και το είδος αυτών των δαπανών διαφέρει από ταξίδι σε ταξίδι. Κάθε δαπάνη αποτελεί ένα ξεχωριστό αντικείμενο το οποίο θα μελετήσουμε στη συνέχεια.
- **Έγκριση ταξιδιώτη :** Όταν ο εργαζόμενος υποβάλει την εντολή μετακίνησης, αυτόματα δίνει και την έγκρισή του για αυτό το ταξίδι.
- **Έγκριση διαχειριστή έργου :** Ο διαχειριστής του έργου στο οποίο χρεώνεται το ταξίδι πρέπει να δώσει την έγκρισή του για αυτό το ταξίδι. Μπορεί είτε να εγκρίνει το ταξίδι είτε να το απορρίψει. Στην συνέχεια θα δούμε την ροή που ακολουθεί το ταξίδι κατά την διάρκεια των εγκρίσεων/απορρίψεων.
- **Έγκριση διαχειριστή οικονομικών του έργου :** ο διαχειριστής οικονομικών του έργου στο οποίο χρεώνεται το ταξίδι επίσης πρέπει να δώσει την έγκρισή του.

- **Έγκριση επικεφαλή διαχειριστή :** Πρόκειται για τον άνθρωπο ο οποίος ελέγχει όλα τα ταξίδια, και αυτός με την σειρά του πρέπει να εγκρίνει ή να απορρίψει αυτό το ταξίδι.
- **Έγκριση λογιστηρίου :** Για να αποζημειωθεί για το ταξίδι ο ταξιδιώτης και να πάρει τα λεφτά του πίσω χρειάζεται την έγκριση του λογιστηρίου.
- Έγκριση της αίτησης από τον διευθυντή.
- **Κωδικός δημοσίευσης :** Πολλά ταξίδια γίνονται επειδή εργαζόμενοι σε ερευνητικές ομάδες πραγματοποίησαν κάποια δημοσίευση την οποία και θα παρουσιάσουν σε κάποιο μέρος, σε αυτή την περίπτωση ο ταξιδιώτης θα πρέπει να προσθέσει και τον κωδικό της δημοσίευσής του.
- **Κατάσταση :** πρόκειται για την κατάσταση του ταξιδιού, σε πιο βήμα της ροής έχει φθάσει μετά από όσες εγκρίσεις/απορρίψεις έχει πάρει. Έχουμε δύο ειδών ταξίδια τα οποία διαφέρουν σε βήματα και θα μελετήσουμε στη συνέχεια.
- **Κωδικός ΑΔΑ :** Για κάθε ταξίδι υπάρχει ένα έγγραφο το οποίο αναρτάται στο σύστημα ΔΙΑΥΓΕΙΑ. Στη συνέχεια η ανάρτηση αυτή παίρνει κωδικό από το σύστημα και ο υπεύθυνος της ανάρτησης καταχωρεί τον κωδικό της διαύγειας στην εγγραφή αυτή.
- **Σχόλια :** Πρόκειται για ελεύθερο πεδίο στο οποίο ο χρήστης μπορεί να προσθέσει ότι σχόλιο μπορεί να έχει για το ταξίδι, μπορεί να είναι κάποια διευκρίνηση για τους επόμενους εγκρίνοντες που θα δουν την αίτηση.
- **Κατηγορία :** Πρόκειται για ένα πεδίο που περιγράφει αν το ταξίδι θα ακολουθήσει την μία ροή ή την άλλη, έχουμε δύο πιθανές περιπτώσεις. Αν το ταξίδι είναι ελληνικό και δεν περιέχει έξοδα τα οποία πληρώνει απευθείας ο οργανισμός τότε το ταξίδι ακολουθεί λιγότερα βήματα, επτά συγκεκριμένα καθώς χρειάζεται λιγότερες εγκρίσεις. Σε κάθε άλλη περίπτωση το ταξίδι χρειάζεται περισσότερες εγκρίσεις και τα βήματα που ακολουθεί είναι έντεκα.
- **Δομή :** τα έργα ανήκουν σε δομές, αυτό το πεδίο αφορά τον κωδικό δομής στο οποίο ανήκει το έργο. Υπάρχουν περιπτώσεις που ένα ταξίδι πραγματοποιείται κάτω από ένα έργο αλλά τελικά χρεώνεται σε διαφορετική δομή. Αυτό το πεδίο κρατά την πληροφορία για την δομή στην οποία χρεώθηκε.

Δαπάνη Ταξιδιού – Travel Expense

Πρόκειται για τις δαπάνες του ταξιδιού, και είναι τα αντικείμενα τα οποία εμπεριέχονται στην λίστα δαπανών της οντότητας Ταξίδι. Οι ιδιότητες που έχουν είναι:

1. **Αναγνωριστικό δαπάνης :** πρόκειται για τον μοναδικό κωδικό της δαπάνης όπως αυτή αποθηκεύεται στη βάση δεδομένων και χρησιμοποιείται στη συνέχεια από την Hibernate.
2. **Κωδικός λογιστηρίου :** Όλες οι δαπάνες οι οποίες χρεώνονται απευθείας στον οργανισμό φέρουν και έναν κωδικό λογιστηρίου ο οποίος είναι ο κωδικός της αντίστοιχης δαπάνης στην βάση του λογιστηρίου. Αν η δαπάνη δεν πληρώνεται απευθείας από τον οργανισμό αυτό το πεδίο είναι άδειο.

3. **Τύπος :** Περιγράφει το είδος της δαπάνης, οι τιμές που μπορεί να πάρει είναι ορισμένες εκ των προτέρων και είναι : Έξοδα εγγραφής, ξενοδοχείο, ημερήσια έξοδα, ημερήσια έξοδα στις μέρες αναχώρησης και επιστροφής, αεροπλάνο, αεροπλάνο με προσφορά από το Heraklio Travel, λεοφορίο, τραίνο, καράβι, ταξί, αυτοκίνητο, πάρκινγκ, διόδια και άλλο. Ανάλογα με την επιλογή του τύπου διαφορετικές από τις πληροφορίες της δαπάνης είναι απαραίτητες, για παράδειγμα στην ημερήσια αποζημείωση μας ενδιαφέρουν οι ημερομηνίες ώστε να γνωρίζουμε για πόσες μέρες θα αποζημειωθεί, στο αεροπλάνο μας ενδιαφέρουν η αφετηρία και ο προορισμός, ενώ στα διόδια δεν μας ενδιαφέρει τίποτα από αυτά.
4. **Ημερομηνία έναρξης :** Χρειάζεται στα ημερήσια έξοδα, σε αυτά τα έξοδα βάζουμε την ημερομηνία έναρξης και την ημερομηνία λήξης τους καθώς και την ημερήσια αποζημείωση που δικαιούται ο ταξιδιώτης. Στην συνέχεια από το γινόμενο των ημερών και της ημερήσιας αποζημείωσης υπολογίζεται το κόστος της δαπάνης.
5. **Ημερομηνία λήξης :** Χρησιμοποιείται μαζί με την ημερομηνία έναρξης.
6. **Αφετηρία :** Χρειάζεται σε έξοδα που περιγράφουν μετακινήσεις. Για παράδειγμα αυτοκίνητο, αεροπλάνο, τραίνο.
7. **Προορισμός :** Χρησιμοποιείται μαζί με την αφετηρία.
8. **Κόστος ταξιδιώτη :** Οι δαπάνες χωρίζονται σε τρεις κατηγορίες, αυτές τις οποίες πληρώνει ο ταξιδιώτης και παίρνει αποζημείωση μετά το πέρας του ταξιδιού, αυτές τις οποίες τις πληρώνει ο ταξιδιώτης αλλά παίρνει τα λεφτά από τον οργανισμό προκαταβολικά, και αυτές τις οποίες τις πληρώνει απευθείας ο οργανισμός. Στο πεδίο αυτό συμπληρώνεται μόνο αν η δαπάνη θα πληρωθεί από τον χρήστη και θα πάρει αποζημείωση μετά το πέρας του ταξιδιού.
9. **Κόστος στον οργανισμό :** Αντίστοιχα με το κόστος ταξιδιώτη που είδαμε, αυτό το πεδίο συμπληρώνεται με το ποσό της δαπάνης μόνο όταν πρόκειται για δαπάνη την οποία θα πληρώσει ο οργανισμός.
10. **Προκαταβολή :** αντίστοιχα με τα δύο προηγούμενα πεδία αυτό το πεδίο συμπληρώνεται μόνο αν το ποσό της δαπάνης θα το πάρει προκαταβολικά ο χρήστης πριν το ταξίδι.
11. **Τελικό ποσό :** όταν ο χρήστης επιστρέψει από το ταξίδι, θα κληθεί να συμπληρώσει την απόδοση του ταξιδιού, αυτό το πεδίο αντιπροσωπεύει το τελικό ποσό αυτής της δαπάνης το οποίο μπορεί να διαφέρει από την πρόβλεψη που έγινε αρχικά στην εντολή μετακίνησης.
12. **Σχόλια :** Σχόλια τα οποία μπορεί να προσθέσει ο χρήστης σχετικά με την δαπάνη ώστε να τα δουν οι εγκρίνοντες.
13. **Όνομα ξενοδοχείου :** Πρόκειται για πεδίο που χρησιμοποιείται όταν η δαπάνη είναι δαπάνη διαμονής σε ξενοδοχείο, σε αυτή την περίπτωση το όνομα του ξενοδοχείο είναι μία πληροφορία που πρέπει να φαίνεται.
14. **Λίστα αποδείξεων:** Στην φάση της έγκρισης του ταξιδιού ο χρήστης σε διάφορες δαπάνες καλείται να ανεβάσει κάποιο πιστοποιητικό που να δείχνει ότι αυτή είναι η πιο οικονομική λύση ή ότι πράγματι το ποσό είναι αυτό. Επίσης στην

φάση της απόδοσης ο χρήστης έχει πλέον την νόμιμη απόδειξη για την δαπάνη που πραγματοποιήθηκε. Αυτά τα αρχεία αφού ψηφιοποιηθούν αναρτώνται στο σύστημα και το πεδίο αυτό περιέχει την λίστα με τα αρχεία αυτά που αφορούν την συγκεκριμένη δαπάνη.

15. **Έγκριση δαπάνης από λογιστήριο :** Το ταξίδι παίρνει εγκρίσεις για να μπορέσει ο ταξιδιώτης να το πραγματοποιήσει, στην απόδοση χρειάζεται έγκριση κάθε δαπάνης ξεχωριστά καθώς κάποιες από αυτές μπορεί να απορριφθούν λόγο απουσίας αποδείξεων ή για άλλους λόγους. Αυτό το πεδίο κρατάει την ημερομηνία στην οποία εγκρίθηκε η δαπάνη από το λογιστήριο.
16. **Έγκριση δαπάνης από διαχειριστή οικονομικών έργου :** αντίστοιχα με το λογιστήριο και ο διαχειριστής οικονομικών του έργου στο οποίο πραγματοποιήθηκε το ταξίδι καλείται να εγκρίνει κάθε δαπάνη ξεχωριστά.
17. **Έγκριση πληρωμής :** Μόλις στην απόδοση μία δαπάνη εγκριθεί και από τον διαχειριστή οικονομικών του έργου και από τον λογιστή, ο λογιστής δίνει την τελική έγκριση για πληρωμή. Σε αυτό το σημείο ο κύκλος της συγκεκριμένης δαπάνης – και εφόσον έχουν εγκριθεί όλες οι δαπάνες τότε και του ταξιδιού – έχει ολοκληρωθεί.

Περιοχή – Place

Από ότι είδαμε παραπάνω τόσο το ταξίδι όσο και τα έξοδα χρειάζονται αφετηρία και προορισμό. Αυτά τα πεδία αν τα αφείσουμε ελεύθερα τότε δημιουργείται πρόβλημα ασυνέπειας των τιμών που παίρνουν, κάποιος μπορεί να γράψει μία περιοχή με διαφορετικό όνομα από κάποιον άλλον, με πεζά ή κεφαλαία ή με παραπάνω κενά. Αυτό το πρόβλημα δημιουργεί αδυναμία στο σύστημα να γνωρίζει ποιος πραγματικά ταξίδεψε που και να μπορεί με μονοσήμαντο τρόπο να απαντάει σε ερωτήματα που σχετίζονται με ταξίδια και προορισμούς. Για το λόγο αυτό ορίζουμε την οντότητα Περιοχή. Αυτή η οντότητα περιγράφει ένα μέρος, αποθηκεύεται στη βάση και στη συνέχεια ο χρήστης επιλέγει από τις ήδη υπάρχουσες περιοχές αυτή στην οποία θέλει. Αν η περιοχή δεν υπάρχει τότε θα πρέπει να προστεθεί από τον διαχειριστή. Οι ιδιότητες της είναι :

1. **Αναγνωριστικό περιοχής :** το μοναδικό αναγνωριστικό της οντότητας όπως αποθηκεύτηκε στη βάση δεδομένων και χρησιμοποιείται από την Hibernate.
2. **Χώρα :** Το όνομα της χώρας για αυτή την περιοχή. Είναι αλφαρηθμιτικό και στη βάση δεδομένων αποθηκεύεται ως VARCHAR(255).
3. **Πόλη :** Είναι το όνομα της πόλης για αυτή την περιοχή. Είναι επίσης αλφαρηθμιτικό και στη βάση δεδομένων αποθηκεύεται ως VARCHAR(255).

Έγκριση – Approval

Όπως είδαμε στα ταξίδια υπάρχουν πολλές εγκρίσεις. Εγκρίσεις από τον

ταξιδιώτη, τους διαχειριστές του έργου, τον επικεφαλή διαχειριστή και τον λογιστή. Η έγκριση έχει τις εξής ιδιότητες:

1. **Χρήστη :** είναι το πρόσωπο το οποίο θα δώσει την έγκρισή του ή θα απορρίψει το αίτημα.
2. **Ημερομηνία :** είναι η ημερομηνία στην οποία έγινε το αίτημα εγκρίθηκε ή απορρίφθηκε.
3. **Πράξη :** Περιγράφει αν έγινε έγκριση ή απόρριψη του αιτήματος.
4. **Σχόλια :** Ο χρήστης που εγκρίνει ή απορρίπτει ένα αίτημα μπορεί να έχει σχόλια πάνω σε αυτό. Ιδιαίτερα στην περίπτωση της απόρριψης μπορεί να γράψει σε σχόλια τον λόγο για τον οποίο απέρριψε το αίτημα, το οποίο θα το δει ο ταξιδιώτης και θα μπορέσει να πράξει τις κατάλληλες διορθωτικές κινήσεις για να ικανοποιήσει τα κριτήρια.

Απόσταση – Distance

Είναι αρκετά συχνό ταξιδιώτες του οργανισμού να χρησιμοποιούν αυτοκίνητο για ταξίδια που πραγματοποιούν εντός Ελλάδας και ιδιαίτερα στην Αθήνα. Όταν ο χρήστης χρησιμοποιεί αυτοκίνητο τότε η αποζημίωση που θα πάρει εξαρτάται από την απόσταση της διαδρομής και το κόστος ανά χιλιόμετρο που δικαιούται. Εφόσον το σύστημα δέχεται τις οντότητες Περιοχή, χρειάζομαστε μία οντότητα που να περιγράφει την απόσταση μεταξύ δύο περιοχών. Αυτή είναι η απόσταση και έχει τις παρακάτω ιδιότητες :

1. **Μοναδικό αναγνωριστικό :** πρόκειται για το μοναδικό αναγνωριστικό της οντότητας όπως αποθηκεύτηκε στη βάση και χρησιμοποιείται από την Hibernate.
2. **Περιοχή πρώτη :** Είναι αντικείμενο Περιοχή (Place), στη βάση είναι ξένο κλειδί στον πίνακα των περιοχών ενώ στην JAVA είναι αντικείμενο Περιοχή.
3. **Περιοχή δεύτερη :** Αντίστοιχα με την πρώτη περιοχή.
4. **Απόσταση :** Είναι ένας αριθμός κινητής υποδιαστολής ο οποίος απεικονίζει την απόσταση μεταξύ της πρώτης περιοχής και της δεύτερης.

Τύπος Ταξιδιού – Travel Type

Οι τύποι ταξιδιών όπως είπαμε είναι ορισμένοι εκ των προτέρων και θέλουμε να τους αποθηκεύουμε στη βάση δεδομένων. Για να το κάνουμε αυτό χρησιμοποιώντας τα πλεονεκτήματα της Hibernate, χρειαζόμαστε μία οντότητα η οποία να περιγράφει τα ίδια πεδία με τον πίνακα της βάσης ώστε να αντιστοιχισθεί. Αυτή η οντότητα είναι ο Τύπος Ταξιδιού – Travel Type, και περιλαμβάνει τις εξής ιδιότητες :

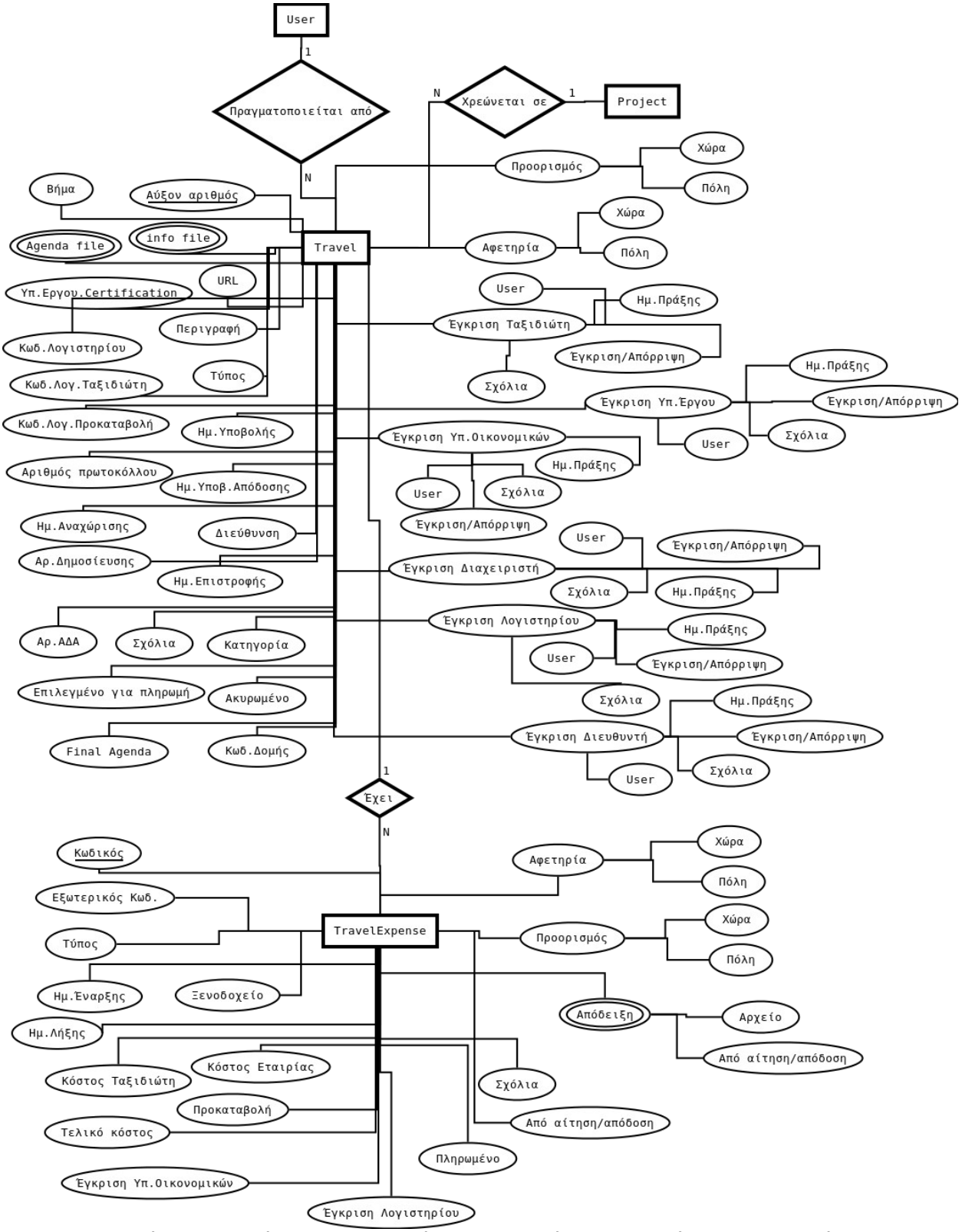
1. **Μοναδικό αναγνωριστικό :** πρόκειται για το μοναδικό αναγνωριστικό της οντότητας όπως αποθηκεύτηκε στη βάση δεδομένων και χρησιμοποιείται από την Hibernate.
2. **Τύπος :** το όνομα του τύπου του ταξιδιού. Παίρνει συγκεκριμένες τιμές τις οποίες αναφέραμε στην περιγραφή της οντότητας Ταξίδι

Τύπος Δαπάνης – Travel Expense Type

Αντίστοιχα με τους τύπους ταξιδιών και οι τύποι δαπανών είναι ορισμένοι εκ των προτέρων και θέλουμε να τους έχουμε αποθηκευμένους στη βάση δεδομένων. Η οντότητα που χρησιμοποιούμε για την αντιστοίχηση είναι ο Τύπος Δαπάνης – Travel Expense Type και περιλαμβάνει τις εξής ιδιότητες :

1. **Μοναδικό αναγνωριστικό :** το μοναδικό αναγνωριστικό όπως δώθηκε από την βάση δεδομένων και χρησιμοποιείται από την Hibernate.
2. **Τύπος :** Πρόκειται για το όνομα του τύπου στα αγγλικά, είναι αλφαρηθμιτικό και αποθηκεύεται στη βάση ως VARCHAR(255).
3. **Τύπος στα ελληνικά :** Στα έγγραφα της εντολής μετακίνησης και της απόδοσης τα οποία εξάγονται αυτόματα από το σύστημα εφόσον είναι ελληνικά κείμενα χρειαζόμαστε τις αντίστοιχες ονομασίες των δαπανών στα ελληνικά.

Ακολουθεί το διάγραμμα οντοτήτων συσχετίσεων όπως προκύπτει από τα παραπάνω:



Σχήμα 3.1: Διάγραμμα οντοτήτων συσχετίσεων συστήματος ταξιδιών

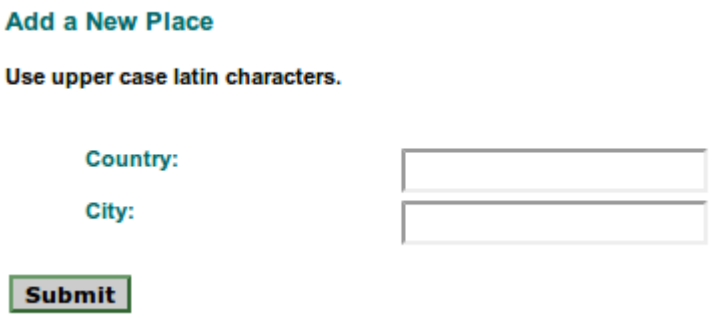
3.3 Λειτουργίες συστήματος

Στην προηγούμενη ενότητα είδαμε τα μοντέλα τα οποία δημιουργήσαμε για τις ανάγκες του συστήματος. Σε αυτή την ενότητα θα δούμε τις επιπλέον λειτουργίες τις οποίες θα προσθέσουμε στο σύστημα και με βάση τις παραπάνω οντότητες και τις ανάγκες του συστήματος

3.3.1 Εισαγωγή Περιοχής – Add Place

Ένα καινούριο ταξίδι που θέλει ένας χρήστης να πραγματοποιήσει μπορεί να έχει προορισμό που δεν υπάρχει στο σύστημα, ή να χρησιμοποιήσει αυτοκίνητο για να πάει σε μία περιοχή που δεν έχει πάει άλλος στο παρελθόν με αποτέλεσμα αυτή η περιοχή να μην έχει εισαχθεί στο σύστημα. Χρειάζομαστε λοιπόν μία φόρμα εισαγωγής Περιοχών στην οποίο ο διαχειριστής θα εισάγει Περιοχές μετά από απαίτηση των ταξιδιωτών.

- Δυνατότητα εισαγωγής Περιοχών έχουν μόνο οι διαχειριστές του συστήματος.
- Ο διαχειριστής επιλέγει ρυθμίσεις (Settings) και στην συνέχεια εισαγωγή περιοχής (Add Place).
- Εμφανίζεται στον χρήστη η φόρμα εισαγωγής καινούριας περιοχής.



Σχήμα 3.2: Φόρμα εισαγωγής περιοχής

- Ο χρήστης εισάγει το όνομα της χώρας και της πόλης που περιγράφουν την καινούρια περιοχή. Με χρήση Javascript η φόρμα μετατρέπει αυτόματα τους χαρακτήρες που πληκτρολογεί ο χρήστης σε κεφαλαία γράμματα ώστε να υπάρχει ομοιομορφία στον τρόπο γραφής. Επειδή είναι συχνό φαινόμενο να εισαχθεί πολλές φορές η ίδια χώρα για πολλά διαφορετικά μέρη, όπου απλά αλλάζει η πόλη, για να υπάρχει ομοιομορφία στον τρόπο που κάποιος εισάγει την χώρα, μόλις πληκτρολογήσει το όνομα της χώρας το πεδίο γίνεται αυτόματα autocomplete ώστε να υποδείξει στον χρήστη ότι αυτή η χώρα έχει ήδη εισαχθεί και να χρησιμοποιήσει ακριβώς την ίδια ονομασία.
- Περιορισμοί: αν ο χρήστης εισάγει μία περιοχή που ήδη υπάρχει, αυτό το καταλαβαίνουμε συγκρίνοντας τα ονόματα της χώρας και της πόλης, τότε το σύστημα δεν αποθηκεύει την περιοχή και τον ενημερώνει ότι η περιοχή ήδη υπάρχει.
- Αφού ικανοποιούνται οι περιορισμοί η καινούρια περιοχή αποθηκεύεται στη βάση.

3.3.2 Εισαγωγή απόστασης – Add Distance

Θα δούμε στην συνέχεια ότι όταν ένας ταξιδιώτης θέλει να χρησιμοποιήσει αυτοκίνητο για να μεταβεί από την αφετηρία σε κάποιο προορισμό τότε δικαιούται αποζημίωση για τα έξοδα του αυτοκινήτου. Η αποζημίωση αυτή υπολογίζεται από το κόστος ανά χιλιόμετρο που δικαιούται ο χρήστης, και υπολογίζεται με βάση το rate του χρήστη, και από τα χιλιόμετρα που διένυσε με το αυτοκίνητο. Θέλουμε επομένως να γνωρίζουμε την απόσταση μεταξύ δύο περιοχών στο σύστημα για να μπορούμε να αυτοματοποιήσουμε την διαδικασία υπολογισμού του κόστους αυτής της δαπάνης, αλλά και να κάνουμε ευκολότερο τον έλεγχο του τελικού ποσού από τους διαχειριστές οικονομικών που θα εγκρίνουν το ταξίδι. Χρειαζόμαστε επομένως μία φόρμα εισαγωγής της απόστασης.

- Δυνατότητα εισαγωγής απόστασης μεταξύ δύο περιοχών έχουν μόνο οι διαχειριστές του συστήματος.
- Ο διαχειριστής επιλέγει ρυθμίσεις (Settings) και στη συνέχεια εισαγωγή απόστασης (Add Distance).
- Εμφανίζεται η φόρμα εισαγωγής καινούριας απόστασης.

Σχήμα 3.3: Φόρμα εισαγωγής απόστασης.

- Ο χρήστης επιλέγει τις δύο περιοχές και στην συνέχεια εισάγει την απόσταση μεταξύ αυτών.
- Μόλις πατήσει Submit τότε γίενται έλεγχος αν η απόσταση αυτών των δύο περιοχών υπάρχει ήδη. Αν υπάρχει ανανεώνει την απόσταση στην υπάρχουσα εγγραφή και ενημερώνει τον χρήστη ότι έγινε ανανέωση της απόστασης. Αν δεν υπάρχει τότε προσθέτει μία καινούρια εγγραφή Απόστασης με τις τιμές που έδωσε ο χρήστης και τον ενημερώνει ότι έγινε εισαγωγή.

3.3.3 Επεξεργασία απόστασης - Edit Distance

Όπου υπάρχει ο ανθρώπινος παράγοντας πάντα υπάρχει και η πιθανότητα λάθους. Έτσι λοιπόν και στην περίπτωση που ένας διαχειριστής πρόσθεσε μία απόσταση μεταξύ δύο περιοχών δεν είναι απίθανο να έκανε λάθος στην απόσταση αυτών σε χιλιόμετρα. Είναι απαραίτητη επομένως η δυνατότητα διόρθωσης ενός τέτοιου λάθους. Ο διαχειριστής πρέπει να μπορεί να βλέπει όλες τις αποστάσεις που έχουν εισαχθεί στο σύστημα και εφόσον επιθυμεί να προβεί σε διορθωτικές παρεμβάσεις.

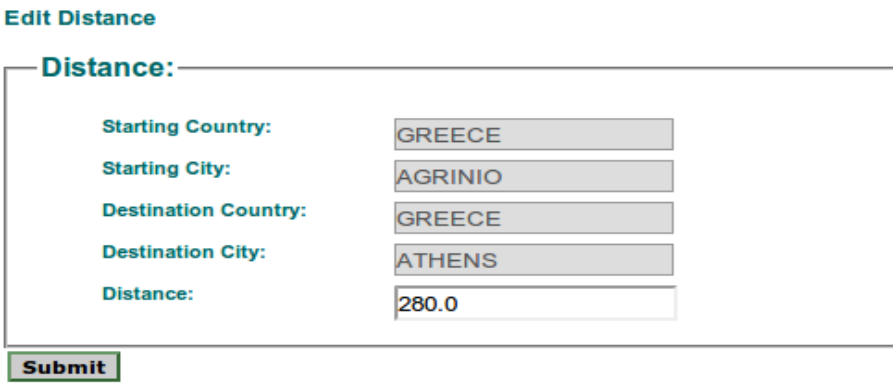
- Δυνατότητα προβολής και επεξεργασίας των αποστάσεων έχουν μόνο οι διαχειριστές του συστήματος.
- Ο διαχειριστής επιλέγει ρυθμίσεις (Settings) και στη συνέχεια επεξεργασία απόστασης (Edit distance).
- Εμφανίζεται στον χρήστη λίστα με όλες τις αποστάσεις που έχουν εισαχθεί στο σύστημα.

List of Distances

Starting Country	Starting City	Destination Country	Destination City	Distance	Edit?
GREECE	AGRINIO	GREECE	ATHENS	280.0	Yes
GREECE	AGRINIO	GREECE	NAFPAKTOS	66.4	Yes
GREECE	AMYNTAIO	GREECE	FLORINA	34.3	Yes
GREECE	AMYNTAIO	GREECE	THESSALONIKI	134.0	Yes
GREECE	ARGOLIKO MIDEAS	GREECE	XIROPIGADO ARKADIAS	23.0	Yes
GREECE	ATHENS	GREECE	ALIARTOS	95.0	Yes
GREECE	ATHENS	GREECE	ASPROPYRGOS	25.0	Yes
GREECE	ATHENS	GREECE	GASTOUNI	284.0	Yes
GREECE	ATHENS	GREECE	GKOURA	167.0	Yes
GREECE	ATHENS	GREECE	IOANNINA	424.0	Yes
GREECE	ATHENS	GREECE	KORINTHOS	94.0	Yes
GREECE	ATHENS	GREECE	PSACHNA	90.6	Yes
GREECE	ATHENS	GREECE	SERRES	583.0	Yes
GREECE	ATHENS	GREECE	THERMI THESSALONIKIS	514.0	Yes
GREECE	ATHENS	GREECE	THESSALONIKI	515.0	Yes
GREECE	ATHENS	GREECE	VEROIA	511.0	Yes
GREECE	ATHENS	GREECE	VOLOS	326.0	Yes
GREECE	CHANIA	GREECE	HERAKLIO	138.0	Yes

Σχήμα 3.4: Λίστα αποστάσεων

- Στη συνέχεια όπως φαίνεται και στην παραπάνω εικόνα ο χρήστης έχει την δυνατότητα να επεξεργαστεί την απόσταση που επιθυμεί. Επιλέγει επεξεργασία (Edit).
- Εμφανίζεται στον χρήστη η φόρμα επεξεργασίας της απόστασης.



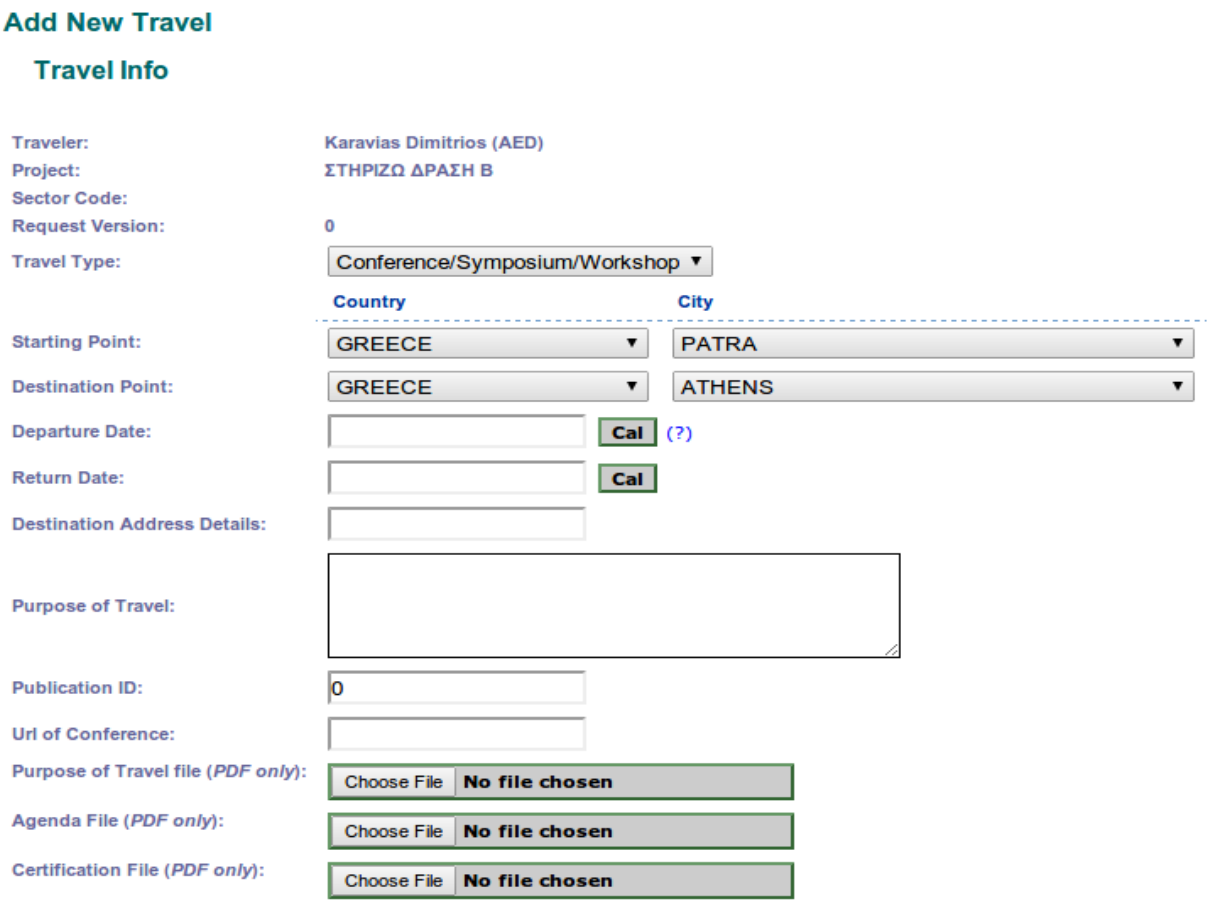
Σχήμα 3.5: Φόρμα επεξεργασίας απόστασης

- Ο διαχειριστής έχει την δυνατότητα να αλλάξει μόνο την χιλιομετρική απόσταση και όχι τις περιοχές.
- Μόλις πατήσει Submit το σύστημα ενημερώνει την αντίστοιχη εγγραφή στη βάση με την καινούρια τιμή.

3.3.4 Υποβολή αίτησης δαπάνης μετακίνησης

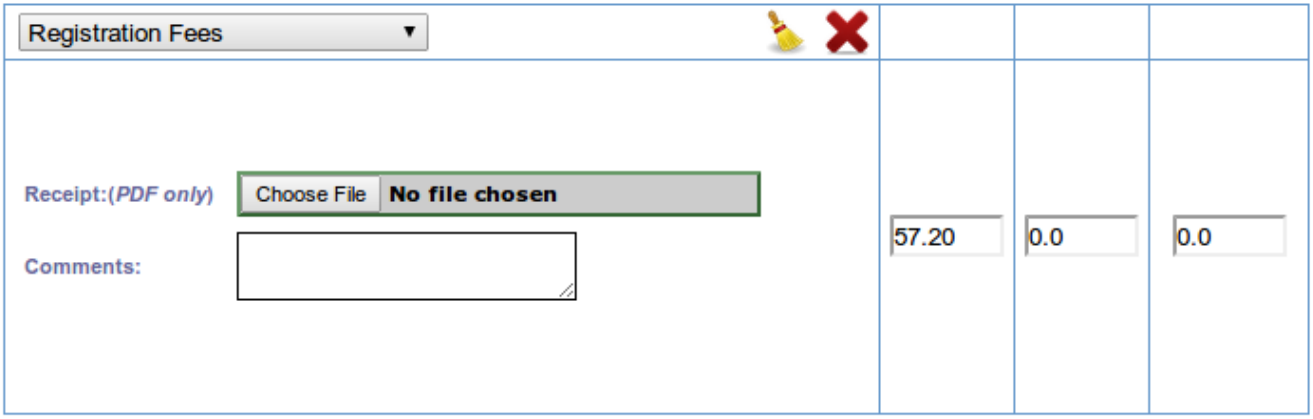
Η βασική λειτουργία του συστήματος. Ο χρήστης εισάγει μία καινούρια αίτηση εντολής μετακίνησης.

- Δυνατότητα εισαγωγής έχουν όλοι οι χρήστες του συστήματος, και μόνο για έργα στα οποία έχουν ενεργό συμβόλαιο.
- Ο χρήστης διαλέγει το έργο στο οποίο θέλει να χρεώσει το ταξίδι και επιλέγει εισαγωγή ταξιδιού από το μενού του έργου.
- Εμφανίζεται στον χρήστη η φόρμα εισαγωγής αίτησης ταξιδιού.
- Ο χρήστης συμπληρώνει τις γενικές πληροφορίες του ταξιδιού και στην συνέχεια τις δαπάνες. Οι γενικές πληροφορίες είναι οι εξής: Τύπος ταξιδιού, Αφετηρία, προορισμός, ημερομηνία αναχώρησης, ημερομηνία επιστροφής, διευθυνση προορισμού, σκοπό του ταξιδιού, υπερσύνδεσμο στην σελίδα του συνεδρίου αν πρόκειται για συνέδριο. Επίσης ο χρήστης πρέπει να αναρτήσει και αρχεία τα οποία να αποδεικνύουν τον σκοπό του ταξιδιού, το πρόγραμμα, και σε περίπτωση που το ταξίδι έχει γίνει σε ημερομηνία προγενέστερη της υποβολής του αιτήματος ένα πιστοποιητικό που αποδεικνύει ότι ο ταξιδιώτης είχε την έγκριση του διαχειριστή του έργου για να ταξιδέψει πριν προλάβει να υποβάλει την αίτηση.



Σχήμα 3.6: Φόρμα εισαγωγής αιτήματος μετακίνησης – Γενικά στοιχεία

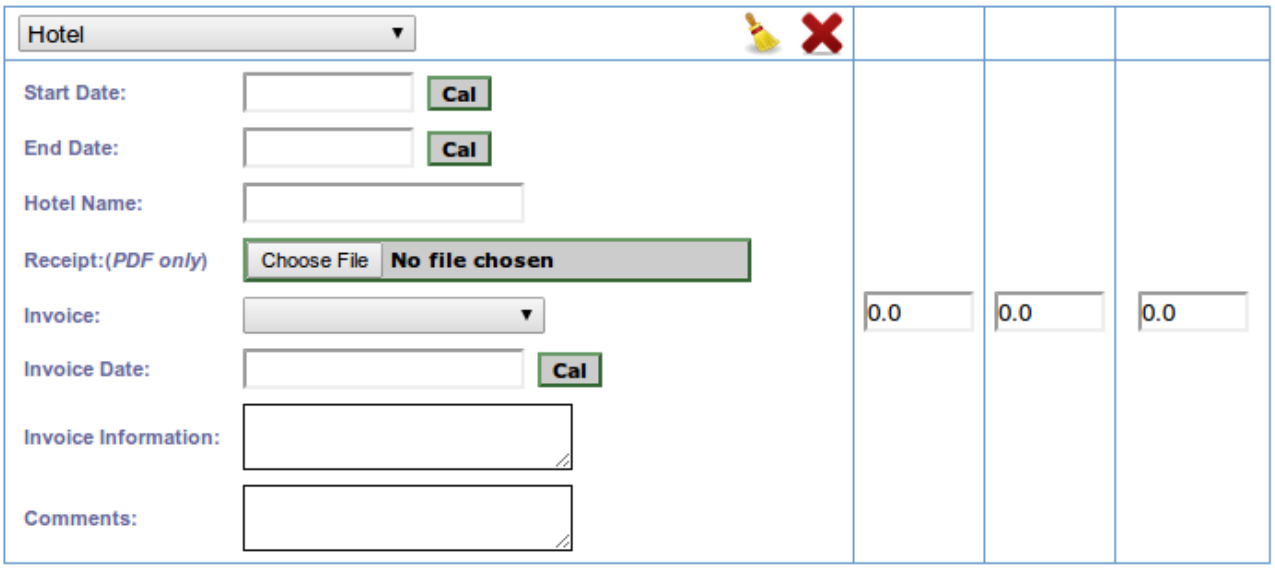
- Επιπλέον ο ταξιδιώτης συμπληρώνει τα έξοδα του ταξιδιού όπως προβλέπεται ότι θα έχουν. Τα έξοδα είναι μία δυναμική λίστα στην οποία ο χρήστης μπορεί να διαγράφει και να προσθέτει έξοδα κατά βούληση. Επιλέγει τον τύπο κάθε δαπάνης και συμπληρώνει τα απαραίτητα στοιχεία της. Τα στοιχεία αυτά διαφέρουν ανάλογα με τον τύπο. Έτσι έχουμε τις ακόλουθες περιπτώσεις:
  - **Έξοδα εγγραφής (Registration fees)**



Σχήμα 3.7: Εισαγωγή δαπάνης εγγραφής

Εδώ το μόνο που μας ενδιαφέρει είναι ο χρήστης να ανεβάσει (upload) ένα PDF που να αποδεικνύει το ποσό της εγγραφής. Ενώ ο χρήστης έχει τη δυνατότητα να προσθέσει κάποιο σχόλιο. Αυτή η δυνατότητα υπάρχει σε κάθε τύπο δαπάνης.

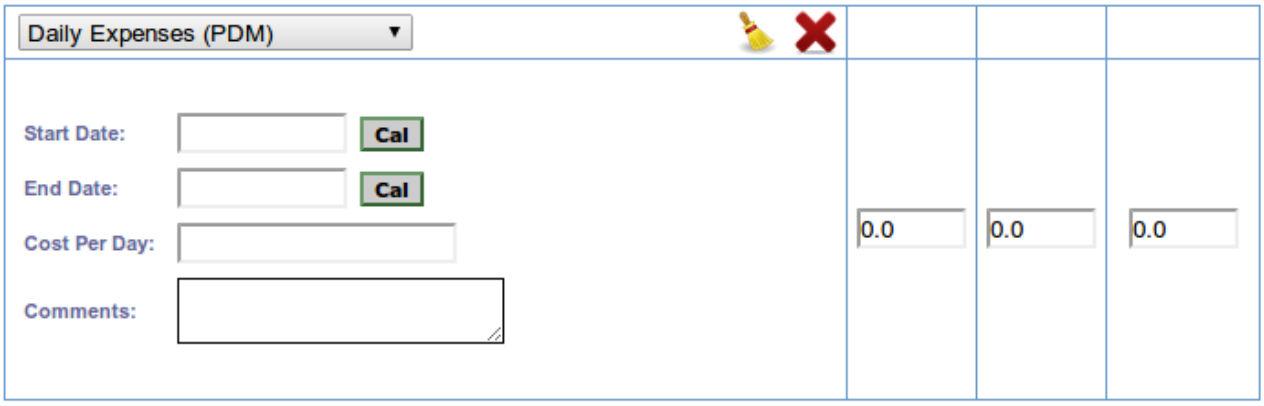
- **Ξενοδοχείο (Hotel)**



Σχήμα 3.8: Εισαγωγή δαπάνης ξενοδοχείου

Σε αυτή την περίπτωση χρειάζονται κάποιες πληροφορίες. Χρειάζεται η ημερομηνία που θα γίνει το check in στο ξενοδοχείο και η ημερομηνία του check out, το όνομα του ξενοδοχείου, τα στοιχεία της απόδειξης και κάποιο αρχείο που να αποδεικνύει το κόστος του ξενοδοχείου ή στην απόδοση την απόδειξη από το ξενοδοχείο. Και εδώ υπάρχει η δυνατότητα για προσθήκη σχολίων.

- **Ημερήσια έξοδα (Daily Expenses)**



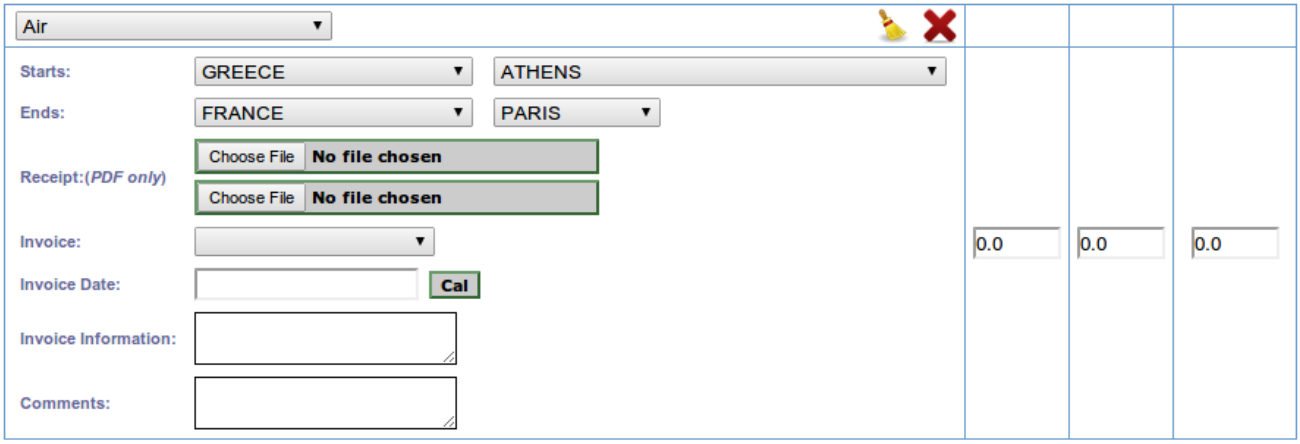
Σχήμα 3.9: Εισαγωγή δαπάνης ημερήσιων εξόδων

Αυτά είναι τα καθημερινά έξοδα διαβίωσης. Χωρίζονται σε έξοδα των ημερών ταξιδιού, αυτά αφορούν δύο μέρες, την ημέρα αναχώρησης και την ημέρα επιστροφής και στα έξοδα των ενδιάμεσων ημερών. Και στις δύο περιπτώσεις

ο χρήστης πρέπει να συμπληρώσει την ημερομηνία που ξεκινούν αυτά τα έξοδα και την ημερομηνία της τελευταίας μέρας που ισχύουν, το κόστος ανά ημέρα υπολογίζεται αυτόματα με βάση το rate του χρήστη ωστόσο υπάρχει δυνατότητα αλλαγής της τιμής. Τέλος πάντα έχει την δυνατότητα προσθήκης σχολίων.

Αυτά τα έξοδα έχουν το χαρακτηριστικό ότι μπορούν να υπολογιστούν αυτόματα μιας και όλες οι πληροφορίες υπάρχουν ήδη από τη στιγμή που ο χρήστης συμπληρώσει στα γενικά στοιχεία του ταξιδιού την ημέρα που ταξιδεύει και την ημέρα που επιστρέφει. Γνωρίζοντας έτσι τις ημερομηνίες αναχώρησης και επιστροφής μπορούμε να συμπληρώσουμε αυτόματα για τον χρήστη αυτά τα πεδία προσθέτοντας δύο έξοδα Travelling Daily Expenses, μία για την ημέρα αναχώρησης και μία για την ημέρα επιστροφής, και μία δαπάνη Daily Expenses για τις ενδιάμεσες ημέρες.

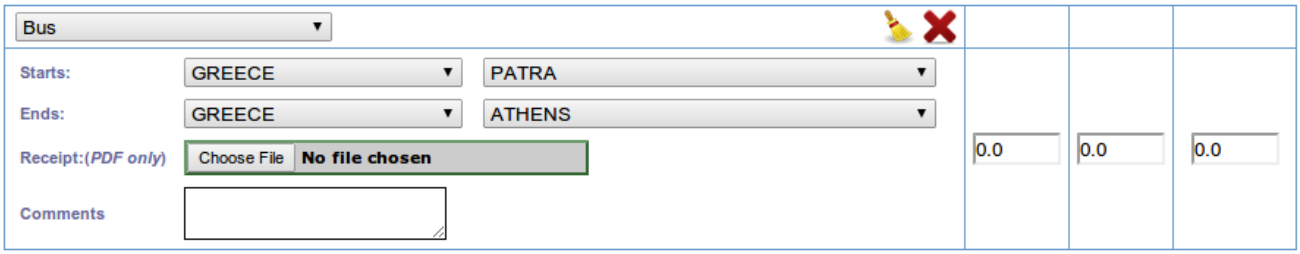
- **Αεροπλάνο (Air - Air / HeraklioTravel)**



Σχήμα 3.10: Εισαγωγή δαπάνης αεροπλάνου

Αυτή η δαπάνη αφορά ταξίδι με αεροπλάνο. Όπως βλέπουμε και στην παραπάνω εικόνα ο χρήστης πρέπει να συμπληρώσει τις περιοχές από που ταξιδεύει και σε ποια πόλη θα προσγειωθεί, θα πρέπει να προσθέσει απόδειξη για την τιμή του αεροπορικού εισιτηρίου καθώς και το είδος της απόδειξης, ενώ μπορεί να προσθέσει σχόλια. Ο οργανισμός για τις πτήσεις με αεροπλάνο συνεργάζεται με την εταιρία HeraklioTravel η οποία οργανώνει πτύσεις με καλύτερες προσφορές. Έτσι ο χρήστης έχει την δυνατότητα επιλογής κατηγορίας Air / HeraklioTravel αν πρόκειται για προσφορά, ή Air για κάθε άλλη περίπτωση.

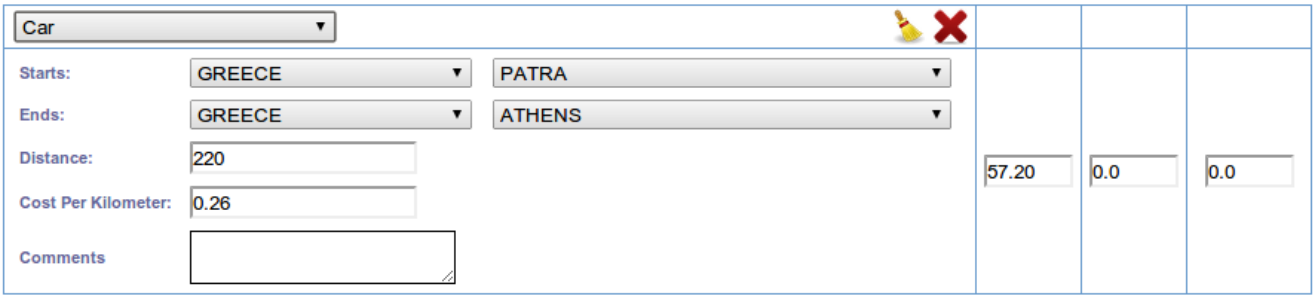
- **Έξοδα μετακινήσεων (Bus – Train – Boat)**



Σχήμα 3.11: Εισαγωγή δαπάνης μετακίνησης με μέσα μαζικής μεταφοράς

Αυτά είναι δαπάνες απλών μεταφορών, το μόνο που μας ενδιαφέρει εδώ είναι η αφετηρία, ο προορισμός και κάποιο PDF που να δικαιολογεί το κόστος, ενώ στην απόδοση θα πρέπει να μπει η κανονική απόδειξη εφόσον θα είναι διαθέσιμη.

- **Αυτοκίνητο (Car)**



Σχήμα 3.12: Εισαγωγή δαπάνης μετακίνησης με αυτοκίνητο

Πρόκειται για έξοδα μεταφοράς με αμάξι. Σε αυτά τα έξοδα μας ενδιαφέρει η αφετηρία, ο προορισμός και η απόσταση μεταξύ αυτών των δύο. Η απόσταση μεταξύ της αφετηρίας και του προορισμού δεν είναι ελεύθερο πεδίο, αντίθετα είναι η απόσταση όπως έχει αποθηκεύτει στο σύστημα στην οντότητα Distance, αν δεν υπάρχει στο σύστημα η απόσταση που θέλει ο χρήστης θα πρέπει να εισαχθεί από τον διαχειριστή. Στην συνέχεια προστίθεται το πεδίο κόστος ανά χιλιόμετρο το οποίο υπολογίζεται αυτόματα από το σύστημα με βάση το rate του χρήστη αλλά μπορεί να αλλαχθεί αν ο χρήστης το επιθυμεί. Η απόσταση πολλαπλασιασμένη με το κόστος ανά χιλιόμετρο μας δίνει και το τελικό ποσό που θα χρεωθεί στον ταξιδιώτη.

- **Άλλα έξοδα (Taxi – Parking – Tolls – Other)**

Σχήμα 3.13: Εισαγωγή λοιπών δαπανών

Πρόκειται για γενικά έξοδα τα οποία ο ταξιδιώτης μπορεί να γνωρίζει ότι θα τα κάνει αλλά στην φάση της εισαγωγής δεν έχει κάποιο στοιχείο να προσθέσει εκτός από κάποιο σχόλιο. Στην φάση της απόδοσης θα πρέπει να αναρτήσει την απόδειξη που εισέπραξε σε μορφή PDF.

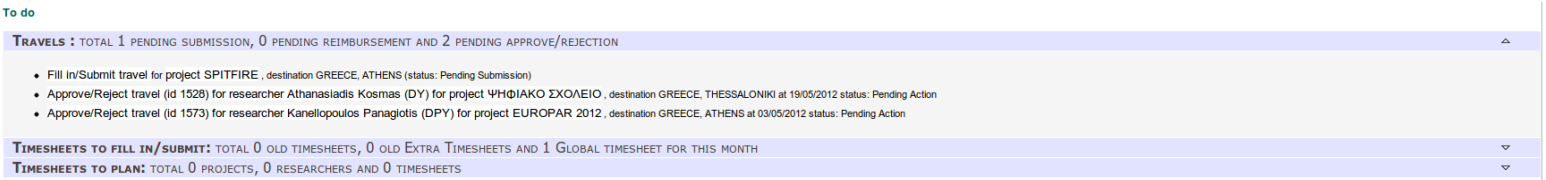
- Ο χρήστης αφού συμπληρώσει τα στοιχεία μπορεί να αποθηκεύσει όσες φορές θέλει το αίτημα χωρίς να το στείλει για έγκριση, και τέλος να το στείλει για έγκριση οπότε δεν μπορεί να το αλλάξει πλέον.

3.3.5 Έγκριση ταξιδιού

Από τις βασικές δυνατότητες του συστήματος. Σκοπός είναι όλη η διαδικασία της έγκρισης ενός ταξιδιού να γίνεται ηλεκτρονικά. Η έγκριση έχει δύο φάσης. Την έγκριση της εντολής μετακίνησης και την έγκριση της απόδοσης. Επίσης η διαδικασία της έγκρισης ενός ταξιδιού είναι διαφορετική όταν το ταξίδι είναι εσωτερικού και ο οργανισμός δεν πληρώνει κάποια δαπάνη για το ταξίδι εκ των προτέρων, και όταν όταν είναι εξωτερικού ή εσωτερικού με δαπάνη που ο οργανισμός πληρώνει εκ των προτέρων. Θα παρουσιάσουμε κάθε περίπτωση ξεχωριστά.

Έγκριση εντολής μετακίνησης για ταξίδια εξωτερικού ή με δαπάνες στον οργανισμό

- Εγκρίνουν: ο διαχειριστής του έργου, ο διαχειριστής των οικονομικών του έργου, ο επικεφαλής διαχειριστής και αν υπάρχει προκαταβολή στο ταξίδι και ο λογιστής του οργανισμού.
- Οι εγκρίσεις γίνονται με την σειρά, αυτός που έχει σειρά να εγκρίνει ενημερώνεται με email στο οποίο υπάρχει και ο υπερσύνδεσμος για να δει το αίτημα και να εγκρίνει ή απορρίψει. Επίσης στην αρχική σελίδα της ιστοσελίδας υπάρχει ειδική κατηγορία στα "TODO" για τα ταξίδια που πρέπει να εγκρίνει με τους αντίστοιχους συνδέσμους σε αυτά.



Σχήμα 3.14: Λίστα TODO

- Αφού ο χρήστης ακολουθήσει έναν από τους παραπάνω τρόπους, εμφανίζεται η φόρμα του ταξιδιού.
- Η φόρμα είναι ίδια με αυτή που συμπλήρωσε ο χρήστης με κάποιες ιδιαιτερότητες:
  - Ο διαχειριστής έργου βλέπει ακριβώς την ίδια φόρμα και μπορεί να πειράξει ακριβώς τα ίδια πεδία.
  - Ο διαχειριστής οικονομικών του έργου πρέπει επιπλέον να προσθέσει και τους κωδικούς λογιστηρίου στις δαπάνες και στο ταξίδι. Επίσης μπορεί να αλλάξει το έργο στο οποίο χρεώνεται το ταξίδι και να αλλάξει και την δομή στο οποίο χρεώνεται.
  - Ο επικεφαλής διαχειριστής βλέπει την ίδια φόρμα με τον διαχειριστή οικονομικών

    - Ο Λογιστής βλέπει την ίδια φόρμα.
- Σε κάθε έγκριση ενημερώνεται στην βάση η κατάσταση του ταξιδιού και η έγκριση που έλαβε χώρα.
- Μόλις όλοι οι χρήστες εγκρίνουν ο ταξιδιώτης είναι πλέον ελεύθερος να πάει το ταξίδι του.

Έγκριση απόδοσης μετακίνησης για ταξίδια εξωτερικού ή με δαπάνες στον οργανισμό

- Εγκρίνουν ο διαχειριστής οικονομικών του έργου και ο λογιστής του οργανισμού.
- Όπως και πριν οι χρήστες ενημερώνονται με email ότι πρέπει να εγκρίνουν την απόδοση, ενώ και πάλι στην ειδική κατηγορία στα TODO υπάρχει αυτή η πληροφορία.
- Ο χρήστης ακολουθεί έναν από τους παραπάνω τρόπους και εισέρχεται στην φόρμα της απόδοσης.
- Ο χρήστης επιλέγει τις δαπάνες που θέλει να εγκρίνει. Σε αυτό το στάδιο εγκρίνονται οι δαπάνες του ταξιδιού και όχι το ίδιο το ταξίδι.
- Μόλις ο χρήστης πατήσει Έγκριση (Approve) τότε ενημερώνονται στη βάση οι δαπάνες που ενέκρινε με την ημερομηνία έγκρισης.
- Περιορισμοί: ο χρήστης δεν μπορεί να πειράξει καμία πληροφορία από αυτές που μπήκαν στην φάση της έγκρισης της εντολής μετακίνησης. Μπορεί επίσης να προσθέσει καινούριες δαπάνες αλλά δεν μπορεί να διαγράψει παλιές δαπάνες.

Έγκριση εντολής μετακίνησης για ταξίδια εσωτερικού χωρίς δαπάνες στον οργανισμό

- Εγκρίνουν ο διαχειριστής του έργου.
- Ενημερώνεται με email ότι πρέπει να εγκρίνει ενώ εμφανίζεται και στα TODO στην αρχική σελίδα.
- Αφού ακολουθήσει έναν από τους παραπάνω τρόπους εισέρχεται στην φόρμα.
- Τροποποιεί αν επιθυμεί τα στοιχεί που έβαλε ο χρήστης και εγκρίνει ή απορρίπτει το αίτημα.
- Μόλις πατήσει Έγκριση ή απόρριψει το σύστημα ενημερώνεται με την ενέργειά του.

Έγκριση απόδοσης μετακίνησης για ταξίδια εσωτερικού χωρίς δαπάνες στον οργανισμό.

- Εγκρίνουν: ο διαχειριστής οικονομικών του έργου, ο επικεφαλής διαχειριστής και ο λογιστής του οργανισμού.
- Ο χρήστης που πρέπει να εγκρίνει ενημερώνεται με email και βλέπει αντίστοιχη ενημέρωση στα TODO στην αρχική σελίδα του συστήματος.
- Αφού ακολουθήσει έναν από τους παραπάνω τρόπους εισέρχεται στη φόρμα.

- Οι ενέργειες των χρηστών είναι οι παρακάτω:
  - Ο διαχειριστής οικονομικών του έργου συμπληρώνει τους κωδικούς λογιστηρίων των δαπανών και του ταξιδιού.
  - Ο επικεφαλής διαχειριστής εγκρίνει τα στοιχεία του ταξιδιού αφού βεβαιωθεί ότι και οι αποδείξεις είναι εντάξει.
  - Ο Λογιστής εγκρίνει μία μία τις δαπάνες για πληρωμή. Μόλις εγκριθούν όλες η διαδικασία έχει ολοκληρωθεί.
- Μόλις οι χρήστες κάνουν Έγκριση η Απόρριψη το σύστημα ενημερώνει τις αντίστοιχες εγγραφές με την ενέργειά τους.

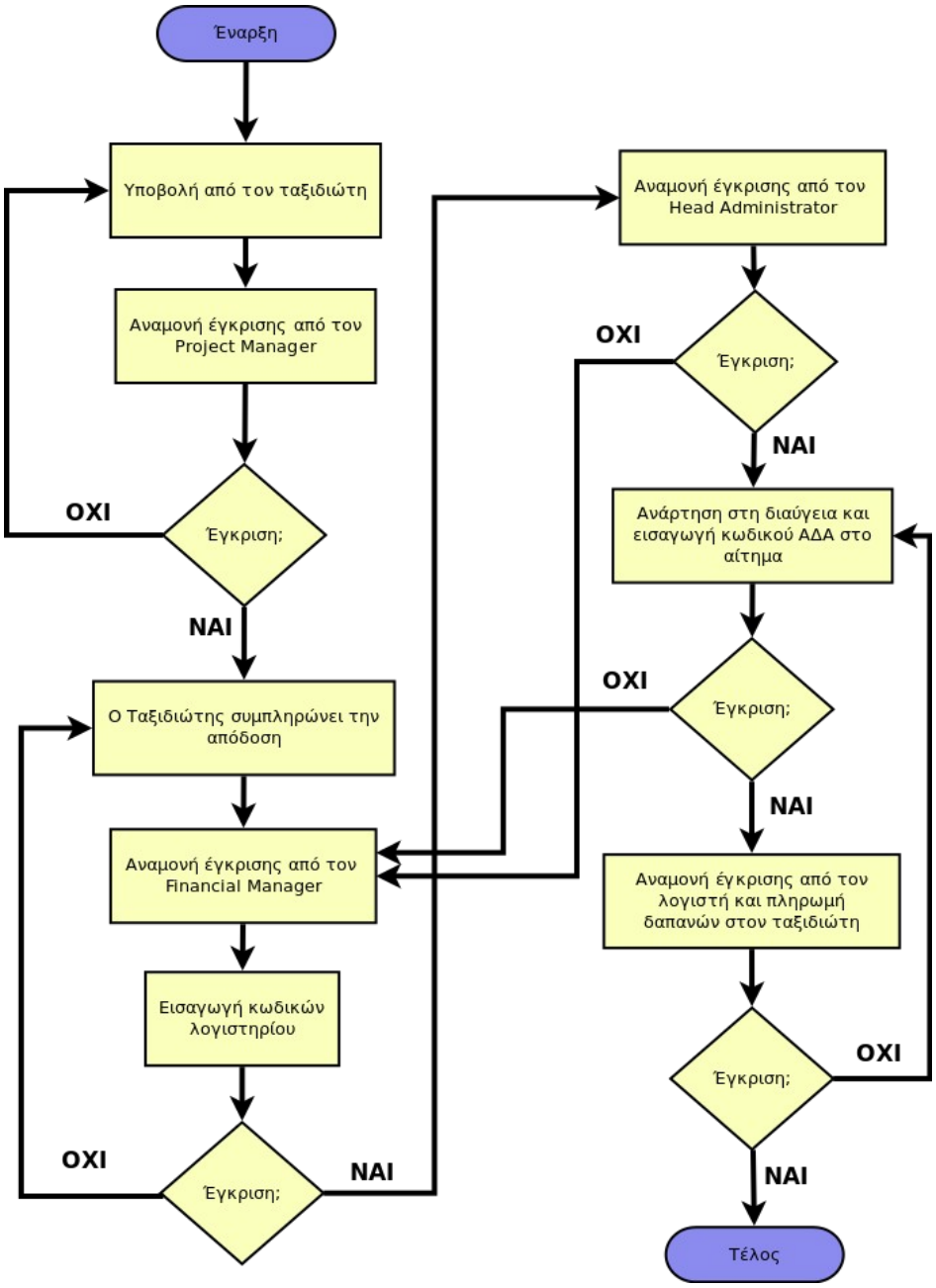
3.3.6 Ανάρτηση στη ΔΙΑΥΓΕΙΑ

Η διαύγεια είναι ένα βήμα το οποίο παρεμβάλλεται στη ροή του ταξιδιού. Ο οργανισμός είναι υποχρεωμένος να αναρτά, όπως και άλλες δαπάνες, τις δαπάνες μετακίνησης στο σύστημα Διαύγεια για την διαφάνεια των πράξεων του οργανισμού. Το πρόγραμμα Διαύγεια είναι ένα καινοτόμο πρόγραμμα το οποίο εφαρμόζεται από το 2010 και σκοπό έχει την ανάρτηση όλων των πράξεων κυβερνητικών φορέων και διοικητικών οργάνων στο Διαδίκτυο ώστε να ενισχύσει τον έλεγχο που ο απλός πολίτης μπορεί να έχει σε τέτοιες πράξεις καθώς και την υπευθυνότητα των φορέων άσκησης δημόσιας εξουσίας για τις πράξεις τους.

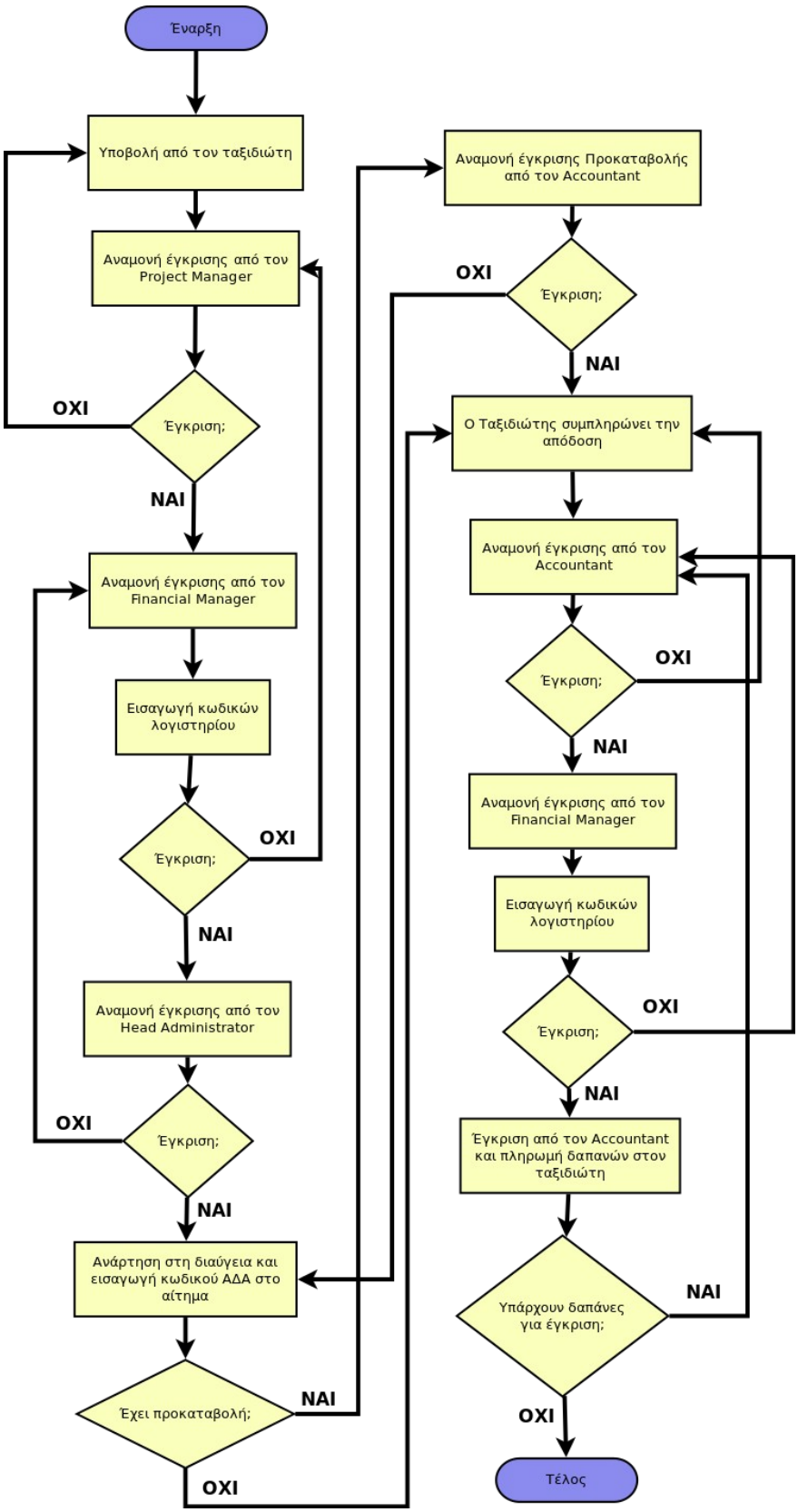
- Ο χρήστης που πραγματοποιεί την ανάρτηση στη διαύγεια είναι συγκεκριμένος και ορισμένος από τον οργανισμό.
- Ο χρήστης ενημερώνεται είτε με email είτε από τα TODO στην αρχική σελίδα του συστήματος ότι πρέπει να αναρτήσει κάποιο ταξίδι.
- Αφού ακολουθήσει έναν από τους παραπάνω τρόπος εισέρχεται στη φόρμα ταξιδιού.
- Αφού κατεβάσει το αρχείο της διαύγειας για το συγκεκριμένο ταξίδι, το οποίο αναπαράγεται από το σύστημα αυτόματα, στη συνέχεια αναρτά το αρχείο στο σύστημα ΔΙΑΥΓΕΙΑ το οποίο επιστρέφει ένα αναγνωριστικό κωδικό για την συγκεκριμένη ανάρτηση. Συμπληρώνει αυτή την ανάρτηση στη φόρμα του ταξιδιού. Δεν έχει δικαίωμα να πειράξει τίποτα άλλο στο ταξίδι.
- Το σύστημα αποθηκεύει τον κωδικό ΔΙΑΥΓΕΙΑΣ για το συγκεκριμένο ταξίδι.

3.4 Διαγράμματα ροής

Όπως είδαμε στην παραπάνω ενότητα ένα ταξίδι ακολουθεί διαφορετική πορεία εγκρίσεων στην εντολή μετακίνησης και στην απόδοση ανάλογα με το αν είναι ταξίδι εσωτερικού χωρίς δαπάνες στον οργανισμό ή έχει δαπάνες στον οργανισμό. Με βάση αυτά που είπαμε στην προηγούμενη ενότητα αλλά και τις ανάγκες του οργανισμού βγήκαν τα εξής διαγράμματα ροής για την διαδικασία έγκρισης του ταξιδιού:



Σχήμα 3.16: Διάγραμμα ροής έγκρισης ταξιδιών εσωτερικού χωρίς έξοδα στον οργανισμό



Σχήμα 3.15: Διάγραμμα ροής έγκρισης ταξιδιών με έξοδα στον οργανισμό

3.5 Έντυπα ταξιδιών

Στις βασικές ανάγκες του οργανισμού είναι και η εξαγωγή διαφόρων επίσημων εντύπων για τα ταξίδια, τα οποία πρέπει να έχουν διαθέσιμα σε περιπτώσεις ελέγχων αλλά και για πρακτική χρήση εσωτερικά στον οργανισμό. Τα έντυπο αυτά είναι:

1. Έντυπο δαπάνης μετακίνησης: είναι το έντυπο με τα στοιχεία του ταξιδιού στην φάση της έγκρισής του και πριν πραγματοποιηθεί το ταξίδι. Φαίνονται τα γενικά στοιχεία του ταξιδιού, τα έξοδα όπως προβλέπονται από τον ταξιδιώτη και οι ηλεκτρονικές υπογραφές των: ταξιδιώτη, διαχειριστή έργου, οικονομικό διαχειριστή έργου, επικεφαλή διαχειριστή του οργανισμού.
2. Έντυπο απόδοσης μετακίνησης: είναι το έντυπο με τα στοιχεία του ταξιδιού μετά την απόδοση. Φαίνονται τα στοιχεία του ταξιδιού, τα πραγματικά έξοδα αυτού, καθώς και οι υπογραφές των : ταξιδιώτη, οικονομικό διαχειριστή του έργου, λογιστή του οργανισμού.
3. Έντυπο διαύγειας: πρόκειται για το έντυπο το οποίο αναρτάται στο σύστημα διαύγεια και περιγράφει την νομιμότητα του ταξιδιού και τον σκοπό για τον οποίο πραγματοποιείται.

Μέχρι και πριν την ανάπτυξη της επέκτασης αυτής, τα έγγραφα αυτά βασισμένα σε υπάρχοντα πρότυπα συμπληρώνονταν με το χέρι από τους εργαζόμενους του οργανισμού. Στους στόχους του συστήματος ήταν και η αυτοματοποίηση της εξαγωγής αυτών των εντύπων με βάση τα στοιχεία που εισήχθησαν στο σύστημα.

Όλα τα έγγραφα είναι τυποποιημένα και δεν υπόκεινται σε επεξεργασία από τον χρήστη. Για το λόγο αυτό βγαίνουν σε μορφή PDF. Για την υλοποίηση αυτών των εγγράφων χρησιμοποιήσαμε το πακέτο itextpdf. Το itextPdf πρόκειται για ένα δυνατό εργαλείο για δημιουργία αρχείων PDF με πολλές δυνατότητες όπου ο χρήστης μπορεί να δημιουργήσει παραγράφους, λίστες, πίνακες, να προσθέσει εικόνες και πολλές άλλες δυνατότητες μορφοποίησης που το κάνουν αρκετά εύχρηστο εργαλείο και γρήγορο για υλοποίηση.

Ακολουθούν εικόνες από τα έγγραφα

Εντολή Μετακίνησης

B41X46941Δ-TAB
64.01.001.121.0441/001
ID 1395 / 156308

Προς: **Καραβίας Δημήτριος**
Παρακαλείσθε να μεταβείτε: **SERBIA, BELGRADE**
Ημ/νια αναχώρησης: 02/09/2012
Ημ/νία επιστροφής: 07/09/2012
Αιτιολογία: Συμμετοχή στο senZations Summer School στα πλαίσια της ενίσχυσης έρευνας της μονάδας.
Έργο: EM1 ΚΠ/ΕΕ
Τύπος: School

Υπεύθ. Έργου

Έγκριση Δαπάνης Μετακίνησης

Κατηγορία δαπανών	Πληρωτέα στον συνεργάτη (1)	Πληρωτέα από το Ε.Α.Ι.Τ.Υ(2)	Κωδικός εγγραφής
Κόστη Registration		415.00€	180722
Αεροπλάνο / HeraklioTravel		232.93€	180723
Έξοδα Διαβίωσης 1 μέρα με κόστος: 44.00€, 02/09/2012 - 02/09/2012 4 μέρες με κόστος: 30.00€, 03/09/2012 - 06/09/2012 1 μέρα με κόστος: 44.00€, 07/09/2012 - 07/09/2012	208.00€		
Λεωφορείο	14.20€		
Λεωφορείο	14.20€		
Σύνολο	236.40€	647.93€	
Γενικό σύνολο (1)+ (2)	884.33€		**Εγγραφές πληρωτέων στο συνεργάτη**
Προκαταβολή	0.00€		
Υπόλοιπο προς πληρωμή	236.40€		180724

Παρατηρήσεις:

Συνεργάτης: **Υπ. Έργου:** **Διαχειριστής:** **Δ.Δ.Υ.Ο:**

Σχήμα 3.17: Έντυπο εντολής μετακίνησης

Φύλλο Απόδοσης Ταξιδιών

B41X46941Δ-TAB
64.01.001.121.0441/001
ID 1395/156308/180724

Συνεργάτης: Καραβίας Δημήτριος
Προορισμός: SERBIA, BELGRADE
Ημ/νια αναχώρησης: 02/09/2012
Ημ/νία επιστροφής: 07/09/2012
Έργο: EM1 ΚΠ/ΕΕ
Ποσοστό Χρέωσης Έργου:
Τύπος: School

Κατηγορία Δαπανών	Σχόλια	Τελικό Ποσόν ΙΤΥΕ	Τελικό Ποσόν Ταξιδιώτη
Κόστη Registration		445.00€	
Αεροπλάνο / HeraklioTravel		232.93€	
Έξοδα Διαβίωσης (μέρα ταξιδιού) 1 ημέρες με κόστος: 44.00€			44.00€
Έξοδα Διαβίωσης 4 ημέρες με κόστος: 30.00€			120.00€
Έξοδα Διαβίωσης (μέρα ταξιδιού) 1 ημέρες με κόστος: 44.00€			44.00€
Λεωφορείο			14.20€
Λεωφορείο			14.20€
Ταξί	Από ξενοδοχείο (Mecavnik, Mokra Gora, Serbia) στο αεροδρόμιο (Βελιγράδι) 2 άτομα, 78.72 ευρο σύνολο, 39.36 δικα μου. (RSD9000*0,00874703)		39.36€
Άλλο	ΤΑΧΙ από αεροδρόμιο Αθήνα στα κτελ κηφισού (39,90) Metro προς Αεροδρόμιο (5,00)		44.90€
Λεωφορείο	Από αεροδρόμιο Βελιγράδι στο ξενοδοχείο		15.00€
	Σύνολο:	677.93€	335.66€
	Προκαταβολή:		
	Συνολικό κόστος ταξιδιού:	1013.59€	
	Υπόλοιπο προς πληρωμή στον συνεργάτη:		335.66€

Παρατηρήσεις:

Συνεργάτης: **Διαχειριστής:** **Έλεγχος λογιστηρίου:**

Σχήμα 3.18: Έντυπο φύλλου απόδοσης ταξιδιών

ΕΓΚΡΙΣΗ ΔΑΠΑΝΗΣ ΜΕΤΑΚΙΝΗΣΗΣ

Αρ. Αιτήματος 156308
Πάτρα, 30/07/2012

ΘΕΜΑ: Έγκριση δαπάνης μετακίνησης.

Έχοντας υπόψη:

1. Τον υπ' αρ. 3966 / 24.5.2011 νόμο του Υπουργείου Παιδείας, Δια Βίου Μάθησης & Θρησκευμάτων «Θεσμικό πλαίσιο των Πρότυπων Πειρ/κών Σχολείων, Ίδρυση Ινστιτούτου Εκπαιδευτικής Πολιτικής, Οργάνωση του Ινστιτούτου Τεχνολογίας Υπολογιστών και Εκδόσεων «ΔΙΟΦΑΝΤΟΣ» και λοιπές διατάξεις»
2. Την υπ' αρ. πρωτ. 76351/Η [ΦΕΚ 253 / 8.8.2011, Τεύχος Υπαλλήλων Ειδικών Θέσεων & Οργάνων Διοίκησης Φορέων του Δημόσιου & Ευρύτερου Δημόσιου Τομέα] Απόφαση του Υπουργείου Παιδείας, Δια Βίου Μάθησης & Θρησκευμάτων περί «συγκρότησης Δ.Σ. του Ινστιτούτου Τεχνολογίας Υπολογιστών & Εκδόσεων «ΔΙΟΦΑΝΤΟΣ»
3. Την «Διόρθωση Σφάλματος» στην υπ' αρ. πρωτ. 76351/Η [ΦΕΚ 286 / 29.8.2011, Τεύχος Υπαλλήλων Ειδικών Θέσεων & Οργάνων Διοίκησης Φορέων του Δημόσιου & Ευρύτερου Δημόσιου Τομέα] Απόφαση του Υπουργείου Παιδείας, Δια Βίου Μάθησης & Θρησκευμάτων περί «συγκρότησης Δ.Σ. του Ινστιτούτου Τεχνολογίας Υπολογιστών & Εκδόσεων «ΔΙΟΦΑΝΤΟΣ»
4. Την υπ. αρ. Δ 400/ 24-5-2011 Απόφαση με θέμα εξουσιοδοτήσεις του Προέδρου προς στελέχη & υπαλλήλους των Υπηρεσιών του ΙΤΥΕ.
5. Τον από 6/10/2010 εγκεκριμένο «Οδηγό Διαδικασιών/Διαχείρισης» (Απόφαση ΔΣ: 109/62§4.5)

Αποφασίζεται η μετακίνηση κ. Καραβίας Δημήτριος
Προορισμός: BELGRADE - SERBIA
Αιτία Μετακίνησης: Συμμετοχή στο senZations Summer School στα πλαίσια της ενίσχυσης έρευνας της μονάδας.
Ημερομηνία Αναχώρησης: 02/09/2012 – Ημερομηνία Επιστροφής: 07/09/2012

Η δαπάνη μετακίνησης καλύπτεται από τον προϋπολογισμό του έργου "Ερευνητική Μονάδα 1 ΚΠ / Ενίσχυση Έρευνας" με κωδικό λογιστηρίου 441 της Ερευνητικής Μονάδας 1. Η τελική δαπάνη θα προσδιορισθεί μετά την προσκόμιση των απαραίτητων δικαιολογητικών μετακίνησης και θα είναι σύμφωνη με τους όρους και περιορισμούς που θέτει το έργο και ο Οδηγός Διαδικασιών/Διαχείρισης του Ινστιτούτου.

Η Διευθύντρια των Διοικητικών και
Οικονομικών Υπηρεσιών ΙΤΥΕ

[1] Είδος Απόφασης: ΛΟΙΠΕΣ ΑΤΟΜΙΚΕΣ ΔΙΟΙΚΗΤΙΚΕΣ ΠΡΑΞΕΙΣ

Σχήμα 3.19: Έντυπο έγκρισης δαπάνης μετακίνησης για την ΔΙΑΥΓΕΙΑ

3.6 Μεταφόρτωση αρχείων

Όταν ένας ταξιδιώτης κάνει αίτηση μετακίνησης, για να εγκριθεί τελικά αυτό το ταξίδι, πρέπει να ο ταξιδιώτης να παρουσιάσει αποδείξεις τόσο για τον σκοπό του ταξιδιού όσο και για τα έξοδα. Συγκεκριμένα οι αποδείξεις που απαιτούνται είναι οι εξής:

- Στην φάση της έγκρισης της εντολής μετακίνησης: Πιστοποιητικό για τον σκοπό του ταξιδιού, το πρόγραμμα του συνεδρίου ή του σχολείου το οποίο θα παρακολουθήσει, το τελικό πρόγραμμα εφόσον αλλάξει, αν το ταξίδι έχει προηγηθεί του αιτήματος τότε απαιτείται έντυπο που να πιστοποιεί ότι ο ταξιδιώτης είχε την έγκριση του διαχειριστή του έργου στο να πραγματοποιήσει το ταξίδι. Επιπλέον αν το ταξίδι έχει πτήση με αεροπλάνο απαιτείται μεταφόρτωση της προσφοράς από το HeraklioTravel.
- Στην φάση της έγκρισης της απόδοσης: Αντίγραφα των νόμιμων αποδείξεων για όλες τις δαπάνες που πραγματοποίησε ο χρήστης.

Είναι φανερό επομένως ότι ένα ταξίδι κουβαλάει πολλά χαρτιά. Το σύστημα έρχεται να απλοποιήσει και να οργανώσει αυτή την κατάσταση επιτρέποντας τον χρήστη να ανεβάσει τα αρχεία αυτά στα σχετικά πεδία στο σύστημα κάνοντας άμεση την εύρεση της απόδειξης για μία συγκεκριμένη δαπάνη και γλιτώνοντας πολύτιμο χρόνο σε κάποιον που θέλει να ελέγξει το ταξίδι.

Για τη μεταφόρτωση των αρχείων χρησιμοποιήσαμε την Spring. Για να δώσουμε σε έναν Controller της Spring τη δυνατότητα να επεξεργαστεί τα αρχεία που θέλει ο χρήστης απαιτείται η χρήση ενός MultipartResolver. Συγκεκριμένα όταν ο DispatcherServlet αντιλαμβάνεται ένα multi-part αίτημα, ενεργοποιεί τον resolver που έχουμε ορίσει στην Spring. Στην συνέχεια ο resolver περικλύει το HttpServletRequest στο MultipartHttpServletRequest το οποίο υποστηρίζει την αποστολή αρχείων. Στην συνέχεια χρησιμοποιώντας το MultipartHttpServletRequest μπορούμε να έχουμε πρόσβαση στα αρχεία μέσα από τον Controller

Τα βήματα που απαιτούνται για να το κάνουμε αυτό είναι τα εξής:

1. Ορίζουμε στην Spring τον multipart resolver που θέλουμε να χρησιμοποιήσουμε.

```
  <!-- Configure the multipart resolver -->
      <bean id="multipartResolver"
class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
          <!-- one of the properties available; the maximum file size in bytes -->
          <property name="maxUploadSize" value="100000"/>
      </bean>
```

2. Στην συνέχεια ορίζουμε στην HTML ότι η φόρμα που θα χρησιμοποιηθεί θα αποστείλει αρχεία

```
<form method="post" action="upload.form" enctype="multipart/form-data">
    <input type="file" name="file"/>
    <input type="submit"/>
</form>
```

3. Μπορούμε να έχουμε πρόσβαση στο αρχείο μέσα από τον Controller με δύο τρόπους.
    1. Αν το έχουμε αντιστοιχήση στο αντικείμενο της φόρμας τότε μπορούμε να έχουμε πρόσβαση σε αυτό απευθείας από το αντικείμενο.

```
public class FileUploadBean {

    private byte[] file;

    public void setFile(byte[] file) {
        this.file = file;
    }

    public byte[] getFile() {
        return file;
    }
}
```

```
public class FileUploadController extends SimpleFormController {

    protected ModelAndView onSubmit(HttpServletRequest request,
HttpServletResponse response,
            Object command, BindException errors) throws ServletException,
IOException {

         // cast the bean
        FileUploadBean bean = (FileUploadBean) command;

         let's see if there's content there
        byte[] file = bean.getFile();
        if (file == null) {
             // hmm, that's strange, the user did not upload anything
        }

         // well, let's do nothing with the bean for now and return
        return super.onSubmit(request, response, command, errors);
    }

    protected void initBinder(HttpServletRequest request,
ServletRequestDataBinder binder)
        throws ServletException {
        // to actually be able to convert Multipart instance to byte[]
        // we have to register a custom editor
        binder.registerCustomEditor(byte[].class, new
ByteArrayMultipartFileEditor());
        // now Spring knows how to handle multipart object and convert them
    }
}
```

2. Μέσω του MultipartHttpServletRequest

```
final MultipartHttpServletRequest multipartRequest;
multipartRequest = (MultipartHttpServletRequest) request;
MultipartFile file = multipartRequest.getFile("file");
```

Αφού ο χρήστης ανεβάσει τα αρχεία που θέλει, στην συνέχεια ο Controller αποθηκεύει αυτά τα αρχεία στον σκληρό δίσκο και είναι διαθέσιμα μέσω της φόρμας κάθε στιγμή στους υπεύθυνους και τον ταξιδιώτη. Ωστόσο υπάρχουν και περιπτώσεις που μπορεί κάποιος να ανεβάσει κάτι το οποίο τελικά να είναι λάθος ή να μην χρειάζεται. Σε αυτή την περίπτωση θέλουμε να δώσουμε δυνατότητα διαγραφής των αρχείων από τον χρήστη. Ο χρήστης μπορεί να το κάνει αυτό μέσα από την φόρμα των ταξιδιών από την οποία τα κάνει και μεταφόρτωση. Στο κατώτερο μέρος της φόρμας αυτής υπάρχει ένα μενού με όλα τα αρχεία που ανέβηκαν και δίνεται η δυνατότητα διαγραφής. Αυτό φαίνεται και στην παρακάτω εικόνα:

Σχήμα 3.20: Δυνατότητα διαγραφής αρχείων

IV. Κρυφή μνήμη

Σε αυτό το κεφάλαιο θα ασχοληθούμε με τον δεύτερο στόχο της διπλωματικής εργασίας, την εφαρμογή κρυφών μνημών στο σύστημα διαχείρισης έργων. Ένα πληροφοριακό σύστημα μετά από ένα διάστημα χρήσης γεμίζει με πολλά δεδομένα. Όταν οι εγγραφές στην βάση δεδομένων αυξάνονται, αυτό έχει σαν συνέπεια ορισμένα αιτήματα στην βάση να αργούν πάρα πολύ. Ειδικότερα όταν ζητείται η εξαγωγή στατιστικών και αναφορών πάνω στα δεδομένα, τότε ζητούμε από την βάση ένα μεγάλο όγκο εγγραφών, πάνω στον οποίο εκτελούμε ένα μεγάλο αριθμό από πράξεις ώστε να ετοιμάσουμε τα δεδομένα για προβολή στην αναφορά. Επειδή η κίνηση των δεδομένων από την βάση στο σύστημα είναι πράξη εισόδου/εξόδου είναι αρκετές τάξης μεγέθους πιο αργή από ότι να είχαμε αυτά τα δεδομένα στη μνήμη.

Σε τέτοιες περιπτώσεις οι λύσεις που μπορείς να δώσεις μπορεί να είναι είτε στο υλικό (hardware), αναβαθμίζοντας την μηχανή πάνω στην οποία τρέχει το σύστημα, είτε προγραμματιστηκές. Στην ενότητα αυτή θα μελετήσουμε τις κρυφές μνήμες, θα δούμε συγκεκριμένα δύο υλοποιήσεις κρυφών μνημών, η πρώτη είναι η κρυφή μνήμη μεθόδων με χρήση Spring-annotations, και η δεύτερη η Δεύτερου Επιπέδου (Second-Level) κρυφή μνήμη της Hibernate. Και στις δύο περιπτώσεις θα χρησιμοποιήσουμε την EHCache, μία βιβλιοθήκη η οποία υλοποιεί και τα δύο αυτά ήδη κρυφών μνημών. Η Ehcache έκανε την εμφάνισή της το 2003 και παρέχει λύσεις για την αύξηση της απόδοσης και την μείωση του "βάρους" στην διαχείριση των πόρων από τις εφαρμογές. Παρέχει λύσεις, επιπλέον του δευτέρου επιπέδου κρυφής μνήμης της Hibernate και της κρυφής μνήμης μεθόδων, και για κατανεμημένες κρυφές μνήμες, και ιστοσελίδες.

4.2 Κρυφή μνήμη

Πρόκειται για μία περιοχή μνήμης η οποία κρατάει πληροφορίες οι οποίες προσπελάζονται με πολύ μεγάλη ταχύτητα ώστε την επόμενη φορά που ζητηθούν, η απόκριση να είναι άμεση. Η κρυφή μνήμη μεσολαβεί μεταξύ μίας αργής μηχανής και μίας πολύ γρήγορης μηχανής. Μπορεί να είναι μία πολύ μικρή ποσότητα μνήμης χρησιμοποιούμενη από μικρο επεξεργαστές, ή αρκετά μεγάλη ώστε να εξυπηρετήσει έναν ολόκληρο server ή cluster από servers για δεδομένα τα οποία χρησιμοποιούνται συχνά.

Ο αρχικός στόχος των κρυφών μνημών είναι να κρατήσουν μία πληροφορία που ζητήθηκε από έναν εξυπηρετητή διαθέσιμη για πρόσβαση σε μελλοντικό χρόνο. Όταν μία διεργασία χρειάζεται πληροφορίες αρχικά τις αναζητεί στην κρυφή μνήμη, ώστε αν είναι διαθέσιμη εκεί να λάβει την πληροφορία ταχύτερα από ότι αν την ζήταγε από τον σκληρό δίσκο ή κάποιο άλλο μέσο του οποίου η ταχύτητα είναι αρκετές τάξης μεγέθους αργότερη από την κρυφή μνήμη. Οι κρυφές μνήμες χαρακτηρίζονται από "χτυπήματα", hit rates, τα οποία μετράνε πόσο συχνά μία πληροφορία στην κρυφή μνήμη ζητείται από κάποια διεργασία. Επίσης μία πληροφορία στην κρυφή μνήμη όσο περισσότερο καιρό μένει σε αυτή χωρίς να ζητείται από κάποια διεργασία, τότε χάνει την αξία της και δεν χρειάζεται να παραμείνει στην κρυφή μνήμη ενώ μπορεί να αντικατασταθεί από κάποια άλλη.

Μία κρυφή μνήμη χαρακτηρίζεται από τον τρόπο με τον οποίο αντιστοιχεί τις πληροφορίες της κύριας μνήμης στις δικές της θέσης μνήμης. Έτσι η κρυφή μνήμη

μπορεί να έχει μία από τις παρακάτω ονομασίες:

- **Απευθείας συσχέτισης (Direct-mapped) :** Σε αυτή την περίπτωση κάθε θέση της κύριας μνήμης μπορεί να αποθηκευτεί μόνο σε μία συγκεκριμένη θέση στην κρυφή μνήμη. Αυτό σημαίνει ότι θα αντικαταστήσει την πληροφορία που υπήρχε σε αυτή την θέση. Αυτό σημαίνει ότι αν δύο πληροφορίες αντιστοιχούν στην ίδια θέση μνήμης και ζητούνται συνέχεια, τότε θα διώχνουν η μία την άλλη. Αν και η υλοποίησή της είναι απλή, για να ανταγωνιστεί της σχετιζόμενες μνήμες χρειάζεται να έχει μεγαλύτερο μέγεθος από αυτές.
- **Ν τρόπων συσχέτισης (N-associative) :** Σε αυτή την περίπτωση μία θέση της κύριας μνήμης μπορεί να αποθηκευτεί σε Ν θέσεις της κρυφής μνήμης. Ο τρόπος με τον οποίο αποφασίζεται ποιες θέσεις θα είναι υποψήφιες για αποθήκευση είναι η χρήση των λιγότερο σημαντικών ψηφίων για αντιστοιχία στην κρυφή μνήμη και κάθε αντιστοιχία περιέχει δύο εγγραφές.
- **Ολοκληρωμένης συσχέτισης (Full-associative) :** Πρόκειται για την περίπτωση που κάθε πληροφορία μπορεί να αποθηκευτεί σε οποιαδήποτε θέση της κρυφής μνήμης. Είναι η καλύτερη λύση όσον αφορά τα hits.

Όταν μία εγγραφή της κύριας μνήμης μπορεί να τοποθετηθεί σε περισσότερες θέσεις της κρυφής μνήμης, τότε, όταν αυτές οι θέσεις είναι γεμάτες πρέπει να αποφασιστεί πια εγγραφή θα φύγει για να δώσει την θέση της στην καινούρια πληροφορία. Στην περίπτωση της απευθείας συσχέτισης κάθε εγγραφή αντιστοιχεί σε μία θέση της κρυφής μνήμης οπότε το πια εγγραφή θα αντικατασταθεί είναι προκαθορισμένο. Ωστόσο στις κρυφές μνήμες Ν τρόπων συσχέτισης και Ολοκληρωμένης συσχέτισης χρειάζεται μία μέθοδος αντικατάστασης των εγγραφών της κρυφής μνήμης. Υπάρχουν αρκετές μέθοδοι αντικατάστασης πληροφοριών σε μία κρυφή μνήμη. Οι πιο κοινοί είναι οι εξής:

- **Τυχαία αντικατάσταση:** Πρόκειται για την πιο απλή μέθοδος. Όταν μία πληροφορία που δεν υπάρχει στην κρυφή μνήμη ζητείται, τότε η κρυφή μνήμη την προσθέτει αφαιρώντας μία τυχαία πληροφορία που είχε. Είναι πολύ απλό σε υλοποίηση αλλά έχει μεγάλο ρυθμό αστοχιών μνήμης.
- **Πρώτο μέσα, Πρώτο έξω (First in, first out):** Ικανοποιεί την ιδέα ότι η παλαιότερη πληροφορία που εισήχθηκε στην κρυφή μνήμη μάλλον είναι η λιγότερη πιθανή να ξαναζητηθεί στο μέλλον. Για την υλοποίηση αυτής της μεθόδου χρειάζεται η υλοποίηση μίας ουράς. Καθώς το πρώτο που θα εισαχθεί στην ουρά θα είναι το πρώτο που θα αφαιρεθεί από αυτή.
- **Αυτό που χρησιμοποιήθηκε τελευταίο (Least recently used):** Ικανοποιεί την ιδέα ότι η πληροφορία που έχει τον περισσότερο καιρό να ζητηθεί, θα είναι αυτή με την λιγότερη πιθανότητα να ξαναζητηθεί στο άμεσο μέλλον, επομένως και αυτή που θα αντικατασταθεί. Είναι μία καλή τεχνική και σχετικά πιο ακριβή στην υλοποίηση από τις άλλες δύο.

Στις επόμενες ενότητες θα μελετήσουμε την χρήση της EHCache, μίας υλοποίησης κρυφών μνημών η οποία παρέχει υλοποιήσεις που μπορούν να χρησιμοποιηθούν τόσο με την Spring για την αποθήκευση στην κρυφή μνήμη τα αποτελέσματα μεθόδων όσο και με την Hibernate με την ενεργοποίηση του δευτέρου

επιπέδου κρυφής μνήμης που παρέχει.

4.3 Caching methods with Spring 3 Annotations.

Σε αυτή την ενότητα θα δούμε πως μπορούμε να χρησιμοποιήσουμε την Ehcache σε συνδυασμό με την Spring για να δημιουργήσουμε κρυφές μνήμες οι οποίες θα αποθηκεύουν τα αποτελέσματα μεθόδων. Για να χρησιμοποιήσουμε την κρυφή μνήμη αρκεί να προσθέσουμε στις ρυθμίσεις της Spring τις παρακάτω γραμμές.

```
<ehcache:annotation-driven cache-manager="ehCacheManager"/>

<bean id="ehCacheManager"
class="org.springframework.cache.ehcache.EhCacheManagerFactoryBean">
      <property name="configLocation" value="classpath:ehcache.xml"/>
</bean>
```

Εδώ λέμε στην Spring ότι θα χρησιμοποιήσει κρυφή μνήμη που θα στηρίζεται σε Annotations και ότι η διαμόρφωση αυτής της μνήμης γίνεται στο αρχείο ehcache.xml. Σε αυτό το αρχείο ορίζουμε τις κρυφές μνήμες που θα χρησιμοποιήσουμε. Για να δημιουργήσουμε και να χρησιμοποιήσουμε μία κρυφή μνήμη θα πρέπει να της δώσουμε ένα όνομα με το οποίο θα αναφερόμαστε σε αυτή μέσω των annotations και να ορίσουμε κάποιες επιπλέον παραμέτρους που αφορούν την μνήμη.

```
<cache name="projectsCache"
        eternal="false"
        maxElementsInMemory="2000"
        overflowToDisk="false"
        diskPersistent="false"
        timeToIdleSeconds="0"
        timeToLiveSeconds="0"
        memoryStoreEvictionPolicy="LRU"
        statistics="true"/>
```

Στο παραπάνω παράδειγμα δείχνουμε την δημιουργία της κρυφής μνήμης που χρησιμοποιήθηκε για τα έργα. Οι ρυθμίσεις που δώσαμε περιγράφονται παρακάτω:

- eternal = false : Η ιδιότητα αυτή χρησιμοποιείται για να ορίσουμε αν τα αντικείμενα στη κρυφή μνήμη θα μπορούν να μείνουν σε αυτή για πάντα, αν δώσουμε τιμή αληθής τότε στα timeouts θα αγνοούνται και τα αντικείμενα θα παραμένουν στη μνήμη.
- MaxElementsInMemory = "2000" : Η ιδιότητα αυτή αφορά το μέγεθος της μνήμης. Εδώ λέμε ότι θέλουμε να έχουμε μέχρι 2000 αντικείμενα στη μνήμη.
- OverFlowToDisk = "false" : Αυτή η ιδιότητα αν είναι αληθής τότε, σε περίπτωση που τα αντικείμενα ξεπεράσουν τα maxElementsInMemory, χρησιμοποιείται ο σκληρός δίσκος για να αποθηκεύσει τα αντικείμενα που θα ακολουθήσουν.
- TimeToIdleSeconds = "0" : Ορίζει το χρόνο που ένα αντικείμενο μπορεί να παραμείνει στη μνήμη χωρίς να ζητηθεί. Η τιμή 0 δηλώνει ότι το αντικείμενο μπορεί να παραμείνει idle για πάντα.
- TimeToLiveSeconds= "0" : Ορίζει τον χρόνο που ένα αντικείμενο έχει από τη

στιγμή που μπήκε στην κρυφή μνήμη μέχρι να θεωρηθεί παλιό και να αφαιρεθεί από αυτή. Η τιμή 0 δηλώνει ότι τα αντικείμενα σε αυτή τη μνήμη δεν θα λήγουν ποτέ.

- MemoryStoreEvictionPolicy = "LRU" : Δηλώνει την πολύ αντικατάστασης που θα χρησιμοποιηθεί. Εμείς χρησιμοποιούμε την Least Recently Used, Κάθε φορά θα αφαιρείται το αντικείμενο που έχει περισσότερο καιρό να ζητηθεί.

Στην συνέχεια επιλέγουμε τις συναρτήσεις που θέλουμε να βάλουμε στην κρυφή μνήμη. Στον manager που περιέχει την λογική για τα ερωτήματα στη βάση για τα έργα υπάρχει η εξής συνάρτηση.

```
List<Project> listByUser(User value)
```

Αυτή η συνάρτηση επιστρέφει όλα τα έργα στα οποία ο χρήστης έχει κάποιο ρόλο. Θέλουμε αυτά τα έργα να μπαίνουν στην κρυφή μνήμη και αν ξαναζητήσουμε τα έργα για τον ίδιο χρήστη ξανά να έρθουν απευθείας από την κρυφή μνήμη. Για να το κάνουμε αυτό αρκεί να βάλουμε το εξής annotation.

```
@Cacheable(cacheName = PROJECTS_CACHE)
List<Project> listByUser(User value)
```

Η Ehcache χρησιμοποιεί τις συναρτήσεις που έχουν ορισθεί με το συγκεκριμένο annotation. Στη συνέχεια χρησιμοποιεί τις παραμέτρους των συναρτήσεων σαν κλειδιά για την κρυφή μνήμη που υλοποιεί. Συγκεκριμένα με βάση τις τιμές των παραμέτρων ελέγχει αν υπήρξε προηγούμενη κλήση με τις ίδιες τιμές. Αν υπήρξε τότε επιστρέφει τα αντικείμενα που επιστράφηκαν και την πρώτη φορά, μόνο που αυτή τη φορά δεν θα αφήσει την συνάρτηση να εκτελεστεί και τα αντικείμενα θα επιστραφούν άμεσα.

Έστω για παράδειγμα ότι ζητούμε τα έργα του χρήστη Χ για πρώτη φορά. Εφόσον δεν υπάρχει εγγραφή στην cache για τον χρήστη Χ για την συγκεκριμένη μέθοδο, θα εκτελεστεί ο κώδικας της συνάρτησης και θα επιστραφεί η λίστα με τα έργα. Τότε θα μπει στην κρυφή μνήμη η αντιστοιχία του χρήστη Χ και της λίστας με τα έργα. Αν ξ ανακαλέσουμε την συνάρτηση για τον χρήστη Χ, επειδή τώρα ο χρήστης Χ θα υπάρχει στην κρυφή μνήμη για αυτή τη μέθοδο, θα επιστραφεί η λίστα με τα έργα του χωρίς να χρειαστεί να κληθεί η συνάρτηση.

Πλεονεκτήματα

Στα πλεονεκτήματα της μεθόδου αποθήκευσης των αποτελεσμάτων των μεθόδων, είναι η απλότητα της χρήσης και ο ξεκάθαρος τρόπος με τον οποίο δουλεύει. Ενώ θα δούμε στην συνέχεια ότι είναι και ιδιαίτερα αποτελεσματική, συγκεκριμένα αν έχουμε hit στην κρυφή μνήμη τα αποτελέσματα επιστρέφονται σε σταθερό χρόνο.

Μειονεκτήματα

Υπάρχουν δύο βασικά μειονεκτήματα της μεθόδου αυτής.

- Το πρώτο βασικό μειονεκτήματα της μεθόδου είναι ότι όταν για παράδειγμα η μέθοδος μας επιστρέψει μία λίστα από αντικείμενα, τότε αυτή η λίστα θα καταλάβει μία θέση της κρυφής μνήμης. Αν στη συνέχεια ζητήσω από μία άλλη μέθοδο ένα αντικείμενο το οποίο υπάρχει στην λίστα που ήρθε από την προηγούμενη μέθοδος, τότε δεν υπάρχει τρόπος να το γνωρίζει, με αποτέλεσμα να αποθηκεύονται στην κρυφή μνήμη πολλαπλές φορές τα ίδια αντικείμενα σε λίστες επειδή επιστράφηκαν από διαφορετικές μεθόδους.
- Το δεύτερο βασικό μειονέκτημα είναι ότι αν ένα αντικείμενο το οποίο εμπεριέχεται σε ένα αντικείμενο που θέλουμε να κρατήσουμε στη κρυφή μνήμη (για παράδειγμα ο Χρήστης που εμπεριέχεται στο Έργο) ανανεωθεί στη βάση. Τότε λόγο του 1ου προβλήματος, ότι δηλαδή ίδια αντικείμενα (Έργα) μπορεί να είναι σε διάφορες θέσεις της κρυφής μνήμης, δεν μπορούμε να γνωρίζουμε με αποδοτικό τρόπο σε πια θέση υπάρχει το αντικείμενο του οποίο οι τιμές αλλάχθηκαν στη βάση και επομένως η κρυφή μνήμη δεν έχει την ενημερωμένη έκδοση. Για τον λόγο αυτό αναγκαζόμαστε να διαγράφουμε όλα τα στοιχεία από την κρυφή μνήμη αντί να διαγράφουμε ή να ενημερώνουμε μόνο τα στοιχεία που πρέπει.

Ωστόσο παρά τα μειονεκτήματα, η μέθοδος αυτή είναι ιδιαίτερα αποτελεσματική στην γενική περίπτωση και για αυτό την χρησιμοποιούμε ως την βασική μέθοδο κρυφής μνήμης στο συγκεκριμένο σύστημα.

4.4 Δεύτερο επίπεδο κρυφής μνήμης της Hibernate

Σε αυτή την ενότητα θα μελετήσουμε την Hibernate γενικά και τις λύσεις που υπάρχουν για υλοποίηση κρυφών μνημών. Στην συνέχεια θα δούμε πως δουλεύει η υλοποίηση της Ehcache για την Δεύτερου Επιπέδου κρυφή μνήμη της Hibernate.

Η Hibernate είναι αυτή που παρέχει την επικοινωνία του συστήματος με την βάση δεδομένων, για τις συναλλαγές στης με την βάση χρησιμοποιεί δύο ήδη κρυφών μνημών. Το **πρώτο επίπεδο (First-Level Cache)** σχετίζεται με το Session και είναι ενεργοποιημένο εκ των προτέρων. Χρησιμοποιείται για όλες τις συναλλαγές με τη βάση. Όταν ένα αντικείμενο βρίσκεται μέσα στο πρώτο επίπεδο κρυφής μνήμης τότε όταν ξαναζητηθεί δεν γίνεται ερώτημα στη βάση αλλά η Hibernate το παίρνει απευθείας από το πρώτο επίπεδο κρυφής μνήμης. Μόλις το Session κλείσει η κρυφή μνήμη πρώτου επιπέδου αδειάζει. Το **δεύτερο επίπεδο κρυφής μνήμης** της Hibernate σχετίζεται με το SessionFactory και υπάρχει για όλα τα Sessions που ανοίγουν. Όταν ένα αντικείμενο φορτώνεται από την βάση αποθηκεύεται στην κρυφή μνήμη και είναι διαθέσιμο σε όλη την εφαρμογή και όχι μόνο σε συγκεκριμένο session. Παρέχονται διάφορες υλοποιήσεις για το δεύτερο επίπεδο κρυφής μνήμης της Hibernate αυτές είναι οι εξής:

- EHCache (Easy Hibernate Cache)
  - είναι γρήγορη

- είναι ελαφριά
- εύκολη στη χρήση
- υποστηρίζει read-only και read/write κρυφή μνήμη.
- Υποστηρίζει αποθήκευση στη μνήμη και στο δίσκο.
- Δεν υποστηρίζει clustering

- OSCache (Open Sympony Cache)
  - Αποτελεί μία πολύ δυνατή υλοποίηση κρυφής μνήμης.
  - Υποστηρίζει read-only και read/write κρυφή μνήμη.
  - Υποστηρίζει αποθήκευση στην μνήμη και στο δίσκο.
  - Παρέχει βασικές υποστήριξη για clustering είτε με χρήση JavaGroups είτε JMS.
- SwarmCache
  - Είναι κρυφή μνήμη για clusters.
  - Υποστηρίζει read-only και nonstrict read/write κρυφή μνήμη.
  - Κατάλληλη για εφαρμογές που έχουν περισσότερες αναγνώσεις από την βάση παρά εισαγωγές.
- Jboss TreeCache
  - Ισχυρή replicated και transactional κρυφή μνήμη.

Εμείς όπως αναφέραμε θα χρησιμοποιήσουμε την EHCache. Το πρώτο βήμα είναι να ορίσουμε στις ρυθμίσεις της Hibernate ότι θα χρησιμοποιήσουμε το Δεύτερο Επίπεδο κρυφής μνήμης. Αυτό γίνεται με την παρακάτω ρύθμιση:

```
<prop key="hibernate.cache.use_second_level_cache">true</prop>
<prop key="hibernate.cache.provider_class">org.hibernate.cache.EhCacheProvider</prop>
<prop key="hibernate.cache.provider_configuration_file_resource_path">ehibcache.xml</prop>
```

Με αυτόν τον τρόπο λέμε ότι θα χρησιμοποιήσουμε το δεύτερο επίπεδο κρυφής μνήμης με τον EhCache provider, και η ρύθμιση της κρυφής μνήμης έχει γίνει στο αρχείο ehibcache.xml. Στην συνέχεια πρέπει να ορίσουμε στο ehibcache.xml όλες της κρυφές μνήμες που θα χρησιμοποιήσουμε. Μία για κάθε αντικείμενο με τον τρόπο που φαίνεται στον παρακάτω πίνακα:

```
<cache name="prman.model.project.Project"
       maxElementsInMemory="200"
       eternal="false"
       overflowToDisk="false"
       diskPersistent="false"
       diskExpiryThreadIntervalSeconds="120"
       memoryStoreEvictionPolicy="LRU"
         />
```

Οι ιδιότητες που ορίζουμε παρατηρούμε ότι είναι ίδιες με αυτές που

χρησιμοποιήσαμε στην περίπτωση της κρυφής μνήμης μεθόδων. Αυτό που διαφέρει ωστόσο είναι το όνομα που δίνουμε στην κρυφή μνήμη, σε αυτή την περίπτωση είναι όλο το μονοπάτι του αντικειμένου που θέλουμε, το πακέτο δηλαδή στο οποίο ανήκει η κλάση μαζί με το όνομα του αρχείου. Επόμενο και τελικό βήμα για να μπορέσουμε να αποθηκεύσουμε τα αντικείμενα Έργο στην κρυφή μνήμη είναι να ορίσουμε στο XML της αντιστοίχισης του αντικειμένου με την βάση ότι θα αποθηκευτεί στην κρυφή μνήμη και αυτόματα θα χρησιμοποιήσει την ρύθμιση που ορίσαμε παραπάνω για την μνήμη. Αυτό το κάνουμε προσθέτοντας την εξής γραμμή στο XML.

```
<class name="prman.model.project.Project" table="PROJECTS">
    <cache usage="read-write"/>
```

Με αυτή τη γραμμή λέμε στην Hibernate να χρησιμοποιήσει κρυφή μνήμη ανάγνωσης-εγγραφής για το αντικείμενο Project. Οι επιλογές για το είδος της κρυφής μνήμης που παρέχει ακόμη η Ehcache είναι οι εξής:

- Ανάγνωση μόνο (read-only): Χρησιμοποιείται για δεδομένα τα οποία δεν γίνονται ποτέ update.
- Nonstrict-read-write: Χρησιμοποιείται για δεδομένα τα οποία κάποιες φορές ενημερώνονται χωρίς ποτέ να κλειδώνουν την κρυφή μνήμη. Όταν χρησιμοποιείται αυτή η μέθοδος δεν είναι σίγουρο ότι ένα αντικείμενο που θα επιστραφεί από την κρυφή μνήμη είναι το πλέον ενήμερο με τις αλλαγές στη βάση.
- Ανάγνωση – εγγραφή (Read-write): Χρησιμοποιείται όταν έχουμε δεδομένα τα οποία ενημερώνονται στη βάση, ενώ τα αντικείμενα στη κρυφή μνήμη είναι ενήμερα με τις αλλαγές στη βάση.

Πλεονεκτήματα

Η Δεύτερου επιπέδου κρυφή μνήμη της Hibernate έρχεται να καλύψει τα μειονεκτήματα της κρυφής μνήμης των μεθόδων που είδαμε παραπάνω. Δηλαδή όταν ένα αντικείμενο είναι στην κρυφή μνήμη τότε η Hibernate δεν το ζητάει από την βάση δεύτερη φορά ακόμα και για διαφορετικά ερωτήματα, οπότε κάθε αντικείμενο είναι μία φορά μόνο στη μνήμη, ενώ αν ένα αντικείμενο ενημερωθεί στη βάση τότε δεν χρειάζεται να αδειάσει όλη η κρυφή μνήμη. Παρά μόνο το αντικείμενο που ενημερώθηκε.

Μειονεκτήματα

Σε αντίθεση με την χρήση την αποθήκευση των μεθόδων όταν οι κρυφές μνήμες της Hibernate είναι πολύ μεγάλες αυτό έχει σαν συνέπεια η επιτυχία στην αναζήτηση του αντικειμένου στην κρυφή μνήμη να είναι συγκριτικά πιο αργή. Ενώ αν η κρυφή μνήμη δεν είναι αρκετά μεγάλη σε περιπτώσεις όπου σε ερωτήματα για την εξαγωγή μίας αναφοράς ζητηθούν περισσότερα αντικείμενα από αυτά που μπορεί να χωρέσει, τότε ο χρόνος αντικατάστασης αυτών των αντικειμένων στη κρυφή μνήμη είναι πολύ μεγάλος με αποτέλεσμα η εφαρμογή να χρειάζεται πολύ περισσότερο χρόνο από αυτόν που θα χρειάζονταν χωρίς κρυφή μνήμη.

4.5 Μετρήσεις

Σε αυτή την ενότητα θα παρουσιάσουμε μετρήσεις της χρήσης των παραπάνω κρυφών μνημών σε ενδεικτικές σελίδες της ιστοσελίδας. Θα συγκρίνουμε σε κάθε σελίδα τέσσερις διαφορετικούς χρόνους:

1. Χωρίς χρήση κρυφής μνήμης
2. Χρήση κρυφής μνήμης για αποθήκευση αποτελεσμάτων μεθόδων
3. Χρήση του δεύτερου επιπέδου κρυφής μνήμης της Hibernate.
4. Χρήση κρυφής μνήμης για αποθήκευση αποτελεσμάτων μεθόδων και του δευτέρου επιπέδου κρυφής μνήμης της Hibernate.

Οι περιπτώσεις αυτές θα συγκριθούν για 5 επαναφορτώσεις της κάθε σελίδας που θα χρονομετρήσουμε. Αυτό που περιμένουμε να δούμε είναι ότι προφανώς στην πρώτη φόρτωση όπου οι κρυφές μνήμες θα είναι άδειες ο χρόνος σε όλες τις περιπτώσεις θα πρέπει να είναι σχεδόν ίδιος. Ενώ στις επόμενες τέσσερις θα πρέπει να δούμε βελτίωση στους χρόνους τουλάχιστον στην περίπτωση που χρησιμοποιούμε κρυφές μνήμες.

Επίσης να σημειωθεί ότι οι χρόνοι που θα παρουσιαστούν είναι χρόνοι φορτώματος της σελίδας, δηλαδή χρόνοι πραγματικής εμπειρίας του χρήστη όταν το σύστημα χρησιμοποιεί ή όχι κρυφές μνήμες, και όχι αποκλειστικά χρόνος απόκρισης κρυφής μνήμης. Ενώ και άλλες επεξεργασίες πάνω στα δεδομένα προσμετρώνται στο συνολικό χρόνο. Ωστόσο αυτό που μας ενδιαφέρει σε ένα τέτοιο σύστημα είναι ο χρόνος στον οποίο τα τελικά δεδομένα είναι έτοιμα για τον χρήστη και το κατά πόσο οι κρυφές μνήμες βελτιώνουν αυτόν τον χρόνο.

1. Αρχική σελίδα

Θα τρέξουμε το πείραμα στην κεντρική σελίδα της εφαρμογής όπου φορτώνονται όλα τα έργα από την βάση καθώς και οι εργαζόμενοι σε αυτά τα έργα.

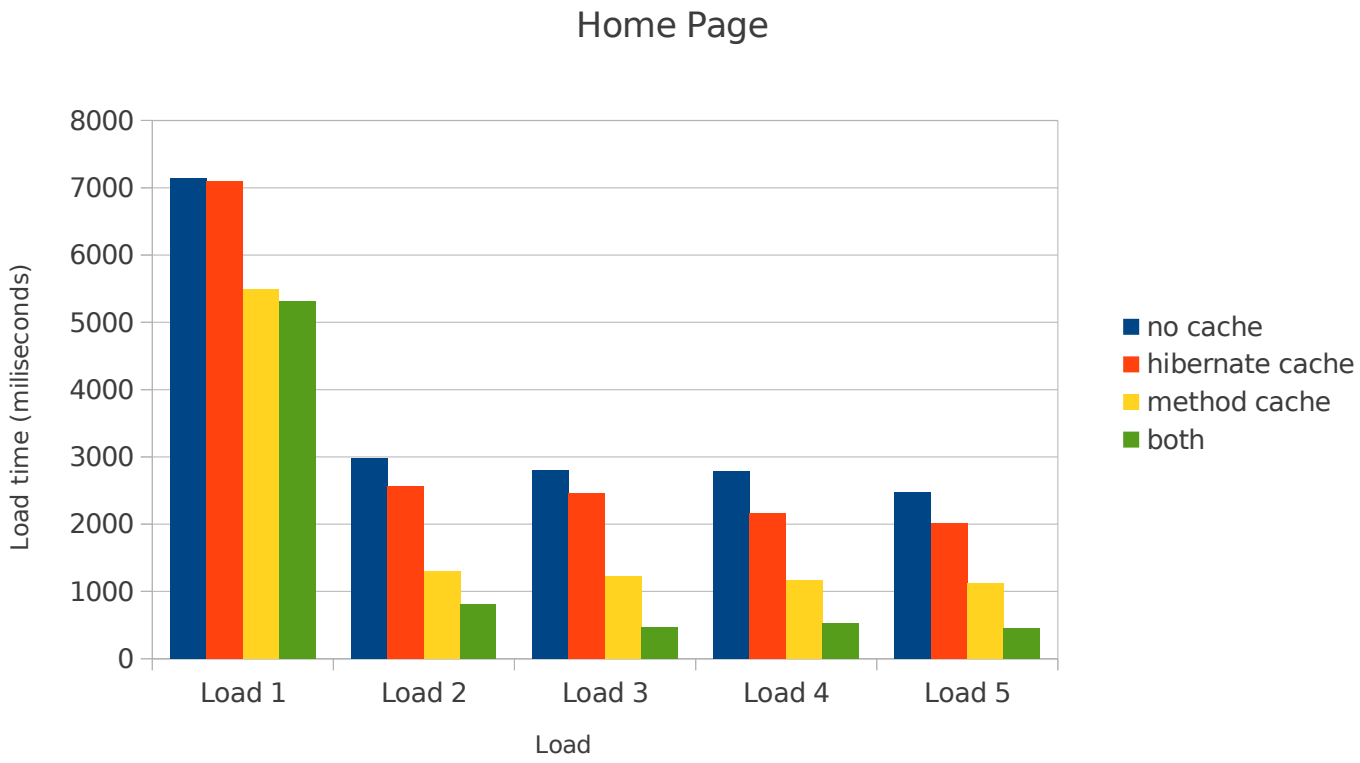
Οι μετρήσεις που πήραμε φαίνονται στον παρακάτω πίνακα:

Home Page

	no cache	hibernate cache	method cache	both
Load 1	7136	7096	5484	5310
Load 2	2980	2569	1297	813
Load 3	2796	2454	1226	474
Load 4	2786	2164	1173	529
Load 5	2472	2018	1120	452

Πίνακας 4.1:Μετρήσεις χρόνου φόρτωσης αρχικής σελίδας

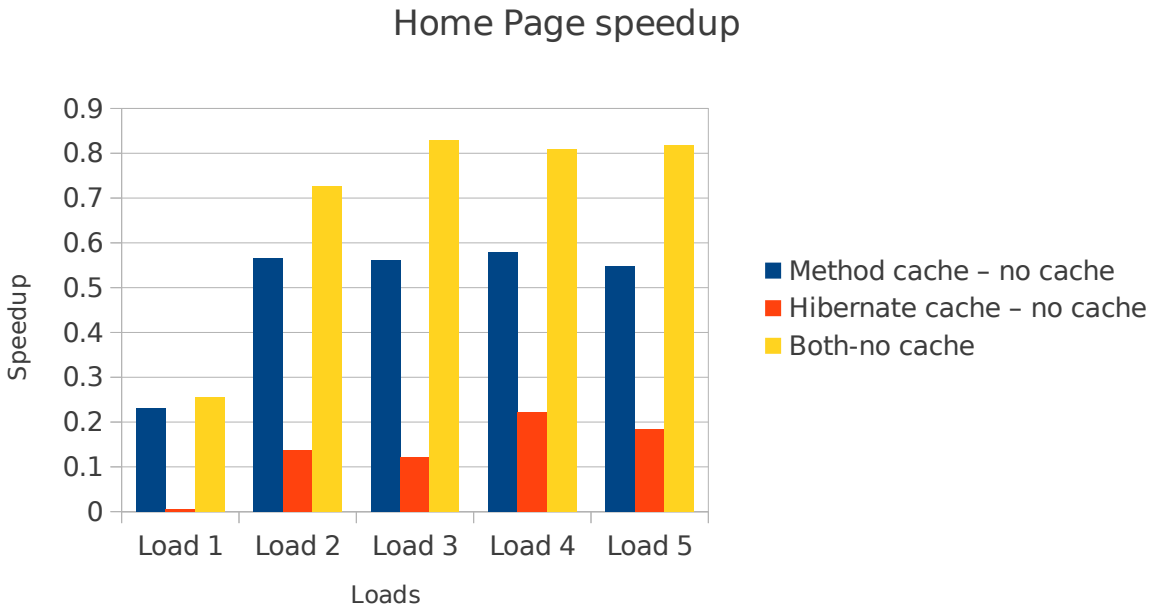
Ακολουθεί γραφική παράσταση των αποτελεσμάτων



Γραφική 4.1: Χρόνοι φόρτωσης αρχικής σελίδας

Παρατηρούμε ότι ακόμα και όταν δεν χρησιμοποιούμε κρυφή μνήμη η σελίδα ανοίγει περίπου 4 δευτερόλεπτα γρηγορότερα σε σχέση με την πρώτη φορά. Ενώ η χρήση της κρυφής μνήμης της Hibernate βελτιώνει την κατάσταση περίπου μισό δευτερόλεπτο σε κάθε φόρτωμα μετά το πρώτο. Ενώ η χρήση της κρυφής μνήμης για της μεθόδους βελτιώνει έως και 1μιση δευτερόλεπτο τον χρόνο του φορτώματος της σελίδας, παρατηρούμε ότι όταν χρησιμοποιούμε και τις δύο κρυφές μνήμες η απόδοση είναι ακόμη καλύτερη. Αυτό συμβαίνει γιατί εκτός από τα έργα τα οποία είναι στη μνήμη, ζητούνται και άλλα απαραίτητα στοιχεία από τη βάση που μπορεί να μην τους έχουμε δώσει κρυφή μνήμη μεθόδων, όπως ο συνδεδεμένος χρήστης. Δεν έχουμε αποκλειστική μνήμη στις μεθόδους μόνο για χρήστες. Ωστόσο η hibernate αποθηκεύει αυτόν τον χρήστη στην δική της κρυφή μνήμη και το επιστρέφει ταχύτατα στη μέθοδο που το ζητάει βελτιώνοντας την συνολική εικόνα.

Στην συνέχεια παρουσιάζουμε το ποσοστό βελτίωσης του χρόνου όταν χρησιμοποιούμε και τα δύο ήδη κρυφής μνήμης με την περίπτωση να μην χρησιμοποιούμε καθόλου κρυφή μνήμη, και με την περίπτωση να χρησιμοποιούμε μόνο κρυφή μνήμη μεθόδων, ώστε να δούμε την συμβολή της κρυφής μνήμης της Hibernate η οποία αν και όχι τόσο γρήγορη ωστόσο αποτελεσματική.



Γραφική 4.2: Επιτάχυνση φόρτωσης αρχικής σελίδας

Παρατηρούμε ότι η χρήση της κρυφής μνήμης μεθόδων επιτυγχάνει καλύτερα αποτελέσματα από την χρήση του Δευτέρου επιπέδου κρυφής μνήμης της Hibernate κάτι το οποίο θα συμβαίνει πάντα όταν τα δεδομένα υπάρχουν στην κρυφή μνήμη. Επίσης ο συνδυασμός των δύο κρυφών μνημών παρατηρούμε ότι επιτυγχάνει σημαντική επιτάχυνση στο σύστημα που αγγίζει και το 80% συγκριτικά με το να μην χρησιμοποιήσουμε καθόλου κρυφή μνήμη.

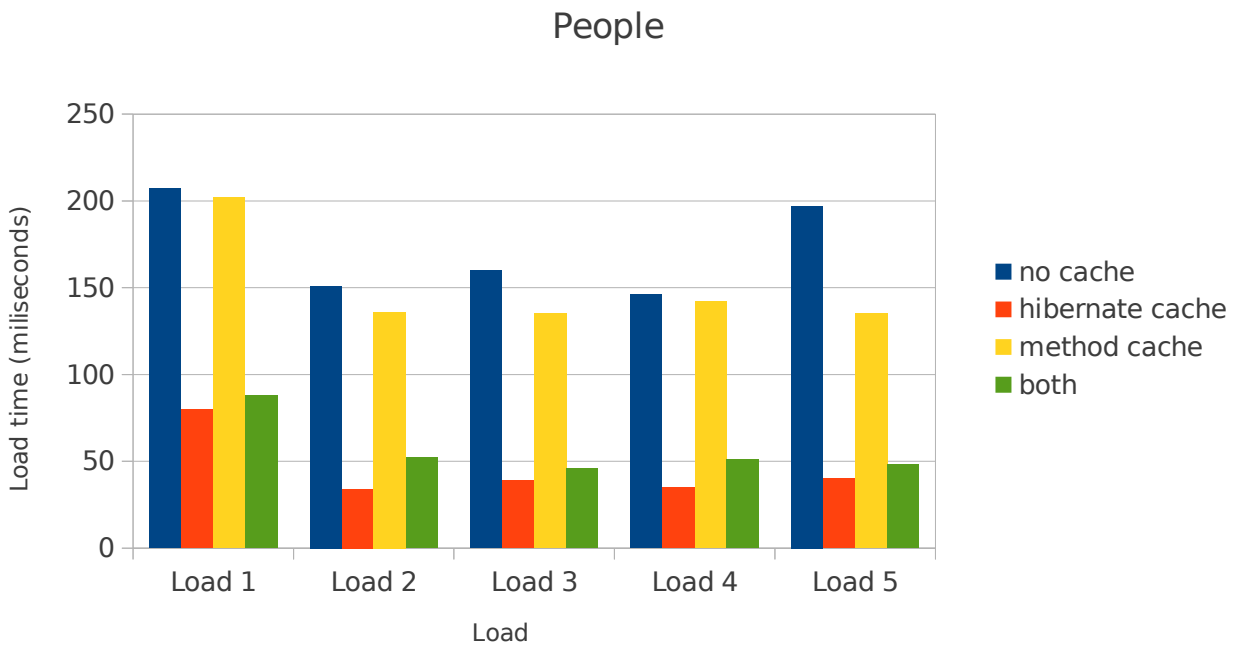
2. Λίστα ανθρώπων

Η δεύτερη σελίδα που θα χρονομετρήσουμε είναι μία σελίδα που παρουσιάζει όλους τους χρήστες του συστήματος και τα στοιχεία τους. Θυμίζουμε ότι οι χρήστες δεν χρησιμοποιούνται στην κρυφή μνήμη μεθόδων γιατί είναι πρωτογενής αντικείμενο και μας συμφέρει να τους παίρνουμε από άλλα αντικείμενα που εμπεριέχουν τους χρήστες, όπως τα έργα. Ωστόσο σε αυτή τη σελίδα χρειαζόμαστε αυτή την πρωτογενής πληροφορία. Ακολουθούν οι μετρήσεις:

People				
	no cache	hibernate cache	method cache	both
Load 1	207	80	202	88
Load 2	151	34	136	52
Load 3	160	39	135	46
Load 4	146	35	142	51
Load 5	197	40	135	48

Πίνακας 4.2: Μετρήσεις φόρτωσης σελίδας χρηστών

Ακολουθεί η γραφική παράσταση των αποτελεσμάτων:

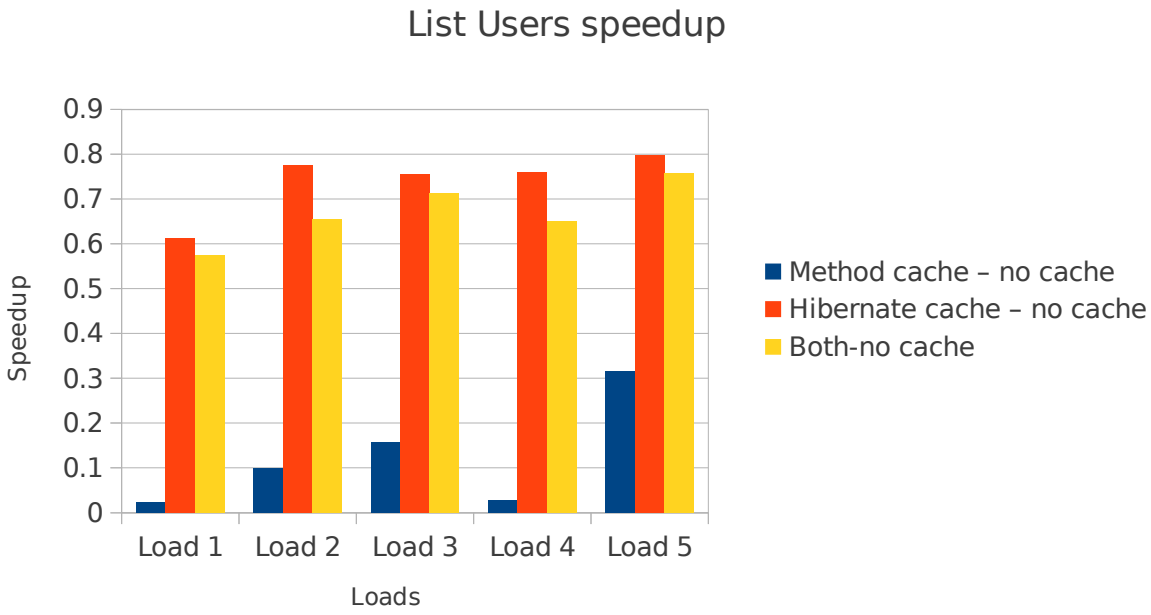


Γραφική 4.3: Χρόνοι φόρτωσης σελίδας χρηστών

Σε αυτή την περίπτωση παρατηρούμε ότι η χρήση της κρυφής μνήμης μεθόδων δεν προσφέρει παρά μόνο ελάχιστη βελτίωση από την απουσία κρυφής μνήμης και αυτό διότι τα κύρια αντικείμενα που φορτώνονται σε αυτή τη σελίδα δεν υπάρχουν στην κρυφή μνήμη.

Σε αντίθετη περίπτωση παρατηρούμε ότι η χρήση του δεύτερου επιπέδου κρυφής μνήμης της Hibernate βελτιώνει κατά πολύ τον χρόνο από το πρώτο κιόλας φόρτωμα της σελίδας. Αυτό οφείλεται στο ότι για να φθάσουμε σε αυτή τη σελίδα με τους χρήστες πρώτα περάσαμε από την σελίδα με την αρχική σελίδα με τα έργα. Στην αρχική αυτή σελίδα μαζί με τα έργα ήρθαν και οι χρήστες αυτών των έργων, δηλαδή όλοι οι χρήστες του συστήματος, έτσι αυτά τα αντικείμενα αποθηκεύτηκαν στη κρυφή μνήμη της Hibernate όταν ζητήσαμε τα έργα με αποτέλεσμα να μπορούν να επαναχρησιμοποιηθούν και τώρα. Κάτι που δεν συμβαίνει όπως είπαμε στην κρυφή μνήμη μεθόδων καθώς μόνο όταν τρέχουμε τις ίδιες μεθόδους με τις ίδιες παραμέτρους έχουμε επιτυχία στην κρυφή μνήμη. Ενώ παρατηρούμε επίσης ότι η χρήση και των δύο μεθόδων κρυφής μνήμης δεν είναι καλύτερη από την χρήση της Hibernate αποκλειστικά αλλά είναι συγκριτικά στα ίδια επίπεδα.

Ακολουθεί το ποσοστό βελτίωσης του χρόνου και σε αυτή την περίπτωση.



Γραφική 4.4: Επιτάχυνση φόρτωσης σελίδας χρηστών

Παρατηρούμε ότι σε αυτή την περίπτωση η χρήση της Δεύτερου επιπέδου κρυφής μνήμης της Hibernate έχει σαφώς καλύτερα αποτελέσματα από την κρυφή μνήμη μεθόδων. Αυτό συμβαίνει διότι τα βασικά αντικείμενα που ζητούνται από την βάση δεδομένων είναι οι χρήστες. Οι διαχειριστές της βάσης για την οντότητα Χρήστης δεν χρησιμοποιούνται στην κρυφή μνήμη μεθόδων διότι οι χρήστες κατά κύριο λόγο έρχονται μέσω άλλων αντικειμένων (έργων, ταξιδιών, ανθρωποωρών). Σε αντίθεση στην Hibernate έχουν την δική τους ξεχωριστή κρυφή μνήμη και η επιτάχυνση που επιτυγχάνεται είναι μεγάλη. Ο συνδυασμός των κρυφών μνημών δεν φαίνεται να έχει κάποια πλεονεκτήματα σε σχέση με την χρήση μόνο της κρυφής μνήμης της Hibernate σε αυτή την περίπτωση. Ωστόσο αξίζει να αναφερθεί όπως φαίνεται στις μετρήσεις με τους χρόνους ότι ο χρόνος φόρτωσης αυτής της σελίδας είναι της τάξης των 100-200 miliseconds που σημαίνει ότι ήδη η σελίδα ανοίγει ικανοποιητικά γρήγορα.

3. Προφίλ χρήστη

Η τρίτη σελίδα που θα χρονομετρήσουμε αφορά την σελίδα ενός συγκεκριμένου χρήστη. Σε αυτή τη σελίδα τα κύρια στοιχεία που φορτώνονται είναι όλα τα στοιχεία αυτού του χρήστη, όλα τα έργα στα οποία έχει εργαστεί και εργάζεται και όλα τα ταξίδια

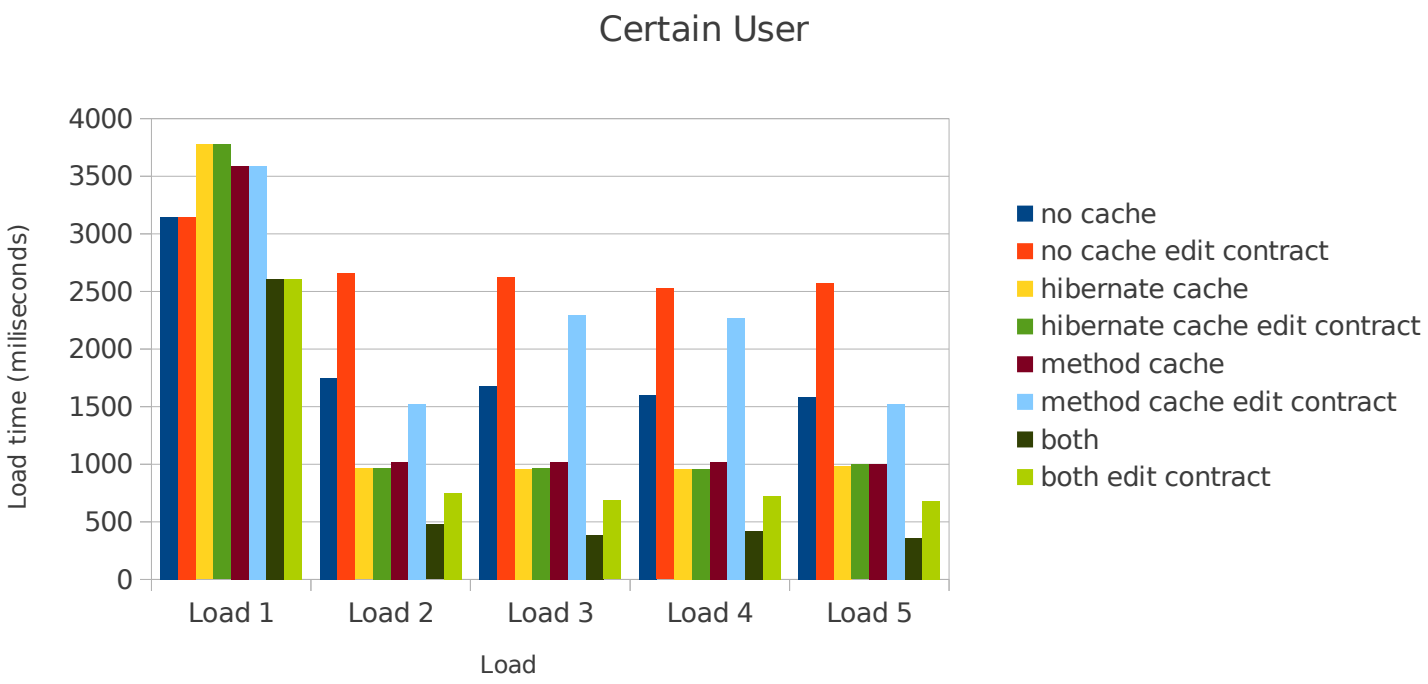
που έχει πραγματοποιήσει. Σε αυτή την περίπτωση θα συγκρίνουμε την απόδοση των κρυφών μνημών όπως και στις προηγούμενες περιπτώσεις αλλά επιπλέον και στην περίπτωση που αλλάζουμε τα στοιχεία του χρήστη και στη συνέχεια ξαναφορτώνουμε τη σελίδα. Όταν κάνουμε ανανέωση των στοιχείων του χρήστη στη βάση αυτό σημαίνει ότι όλες οι κρυφές μνήμες μεθόδων που εξαρτώνται από τον χρήστη θα αδειάσουν. Οπότε αυτό που περιμένουμε να δούμε είναι ότι με την χρήση της κρυφής μνήμης της Hibernate τα αποτελέσματα οι χρόνοι θα παραμένουν χαμηλά διότι η κρυφή μνήμη της Hibernate θα εξακολουθεί να έχει όλα τα δεδομένα στην μνήμη της και μόνο αυτά που άλλαξαν θα πρέπει να ξαναέρθουν, ενώ στην περίπτωση της κρυφής μνήμης μεθόδων οι χρόνοι δεν θα είναι τόσο χαμηλοί καθώς οι περισσότερες κρυφές μνήμες που θα χρησιμοποιήσει σε αυτή την περίπτωση θα αδειάσουν.

Οι χρόνοι είναι οι ακόλουθοι:

User								
	no cache	no cache edit contract	hibernate cache	hibernate cache edit contract	methods cache	ethods cache edit contra	both	both edit contract
Load 1	3147	3147	3775	3775	3592	3592	2606	2606
Load 2	1752	2658	962	971	1020	1525	476	749
Load 3	1682	2622	961	970	1022	2298	384	684
Load 4	1598	2526	957	959	1020	2268	424	724
Load 5	1581	2576	989	997	1003	1521	361	681

Πίνακας 4.3: Μετρήσεις φόρτωσης σελίδας προφίλ χρήστη

Ακολουθεί η γραφική παράσταση των αποτελεσμάτων:

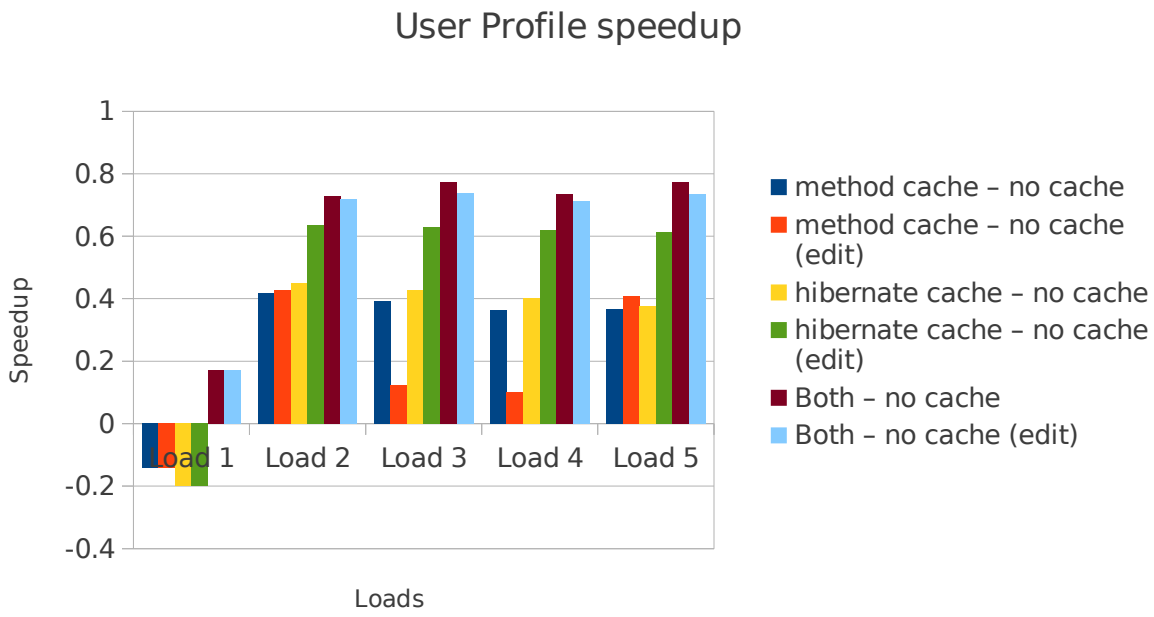


Γραφική 4.5: Χρόνοι φόρτωσης σελίδας προφίλ χρήστη

Όπως παρατηρούμε από το δεύτερο φόρτωμα και μετά της σελίδας, παρατηρούμε ότι όταν δεν χρησιμοποιούμε κρυφή μνήμη μετά από ανανέωση τιμών στη βάση η

σελίδα αργεί σχεδόν 2 δευτερόλεπτα περισσότερο από όταν δεν ανανεώνουμε τιμές στη βάση. Στην περίπτωση που χρησιμοποιούμε κρυφή μνήμη μεθόδων παρατηρούμε ότι ο χρόνος υπερδιπλασιάζεται όταν έχει προηγηθεί ανανέωση και καθάρισμα της κρυφής μνήμης από την περίπτωση που δεν έχει προηγηθεί. Όταν χρησιμοποιούμε κρυφή μνήμη της Hibernate παρατηρούμε ότι ο χρόνος φαίνεται να είναι σταθερά χαμηλός στο 1 δευτερόλεπτο και να μην επηρεάζεται σχεδόν καθόλου από την ανανέωση ή όχι των τιμών. Το εντυπωσιακό είναι όταν συνδυάζουμε και τις δύο κρυφές μνήμες όπου ο χρόνος κυμαίνεται από 0.5 δευτερόλεπτα σε 0.75.

Ακολουθεί η γραφική παράσταση της επιτάχυνσης που πραγματοποιήθηκε στις παραπάνω περιπτώσεις:



Γραφική 4.6: Επιτάχυνση φόρτωσης σελίδας προφίλ χρήστη

Αυτά που είπαμε φαίνονται και στην παραπάνω γραφική παράσταση στην οποία παρατηρούμε τα εξής:

- Όταν χρησιμοποιούμε την κρυφή μνήμη των μεθόδων χωρίς να χρειαστεί να επεξεργαστούμε τα στοιχεία του σεναρίου, τότε επιτυγχάνουμε μία βελτίωση της τάξης του 40%, ωστόσο αν επεξεργαστούμε το σενάριο και αδειάσει η κρυφή μνήμη η βελτίωση φαίνεται να πέφτει σημαντικά.
- Όταν χρησιμοποιούμε την κρυφή μνήμη της Hibernate τότε είτε επεξεργαζόμαστε είτε όχι το σενάριο, επειδή δεν αδειάζει ολόκληρη η κρυφή μνήμη, ο χρόνος παραμένει σταθερός οπότε και η η επιτάχυνση μεγαλώνει στην περίπτωση που επεξεργαζόμαστε τα στοιχεία.
- Στην περίπτωση τέλος που χρησιμοποιούμε και τις δύο κρυφές μνήμες μαζί βλέπουμε ότι συνδυάζουμε τα οφέλη και των δύο και τα αποτελέσματα είναι καλύτερα από το να χρησιμοποιήσουμε οποιαδήποτε κρυφή μνήμη ξεχωριστά.

4. Παρακολούθηση πακέτων εργασίας

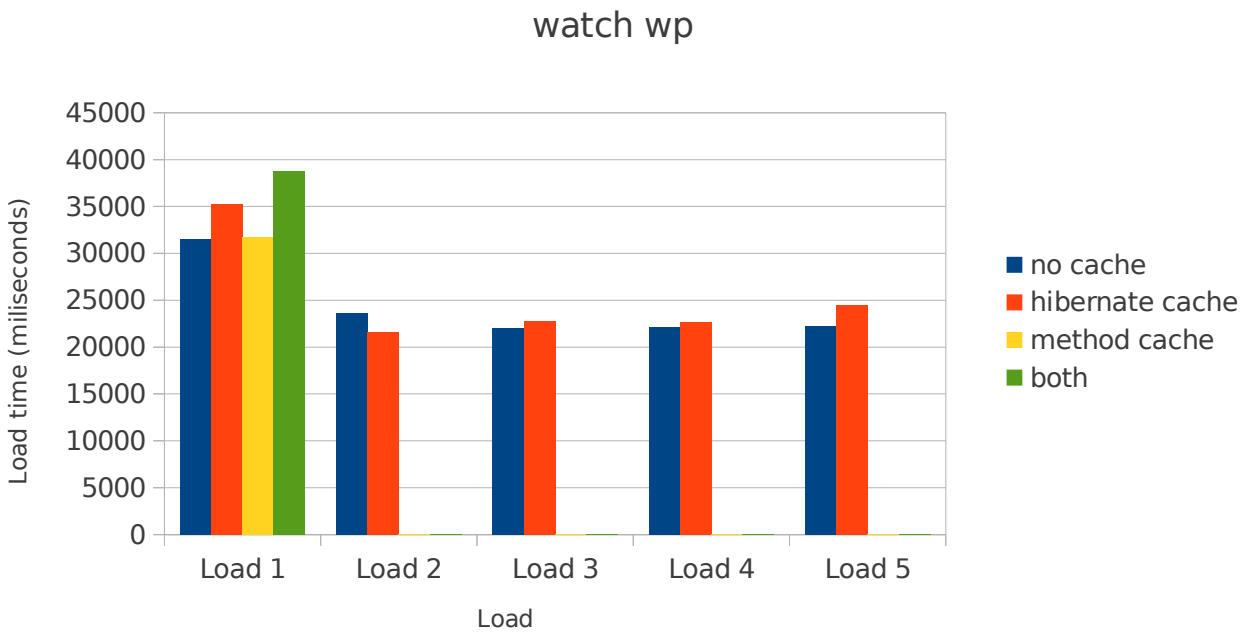
Τέλος θα εξετάσουμε τους χρόνους εκτέλεσης που χρειάζεται σε κάθε περίπτωση για ανοίξουμε μία σελίδα η οποία προβάλει αναφορές για ένα έργο, που αφορούν όλους τους εργαζόμενους του έργου και τις ώρες εργασίας τους. Αυτή η σελίδα φορτώνει ένα τεράστιο όγκο δεδομένων από την βάση και ο χρόνος που απαιτείται για την εξυπηρέτηση από την βάση είναι σημαντικός. Ακολουθούν τα δεδομένα των μετρήσεων:

watch wp

	no cache	hibernate cache	method cache	both
Load 1	31538	35290	31682	38829
Load 2	23632	21576	15	13
Load 3	22055	22758	16	14
Load 4	22141	22638	14	15
Load 5	22232	24507	15	15

Πίνακας 4.4: Μετρήσεις φόρτωσης σελίδας παρακολούθησης πακέτων εργασίας

Ακολουθεί η συγκριτική γραφική παράσταση.



Γραφική 4.7: Χρόνοι φόρτωσης σελίδας παρακολούθησης πακέτων εργασίας

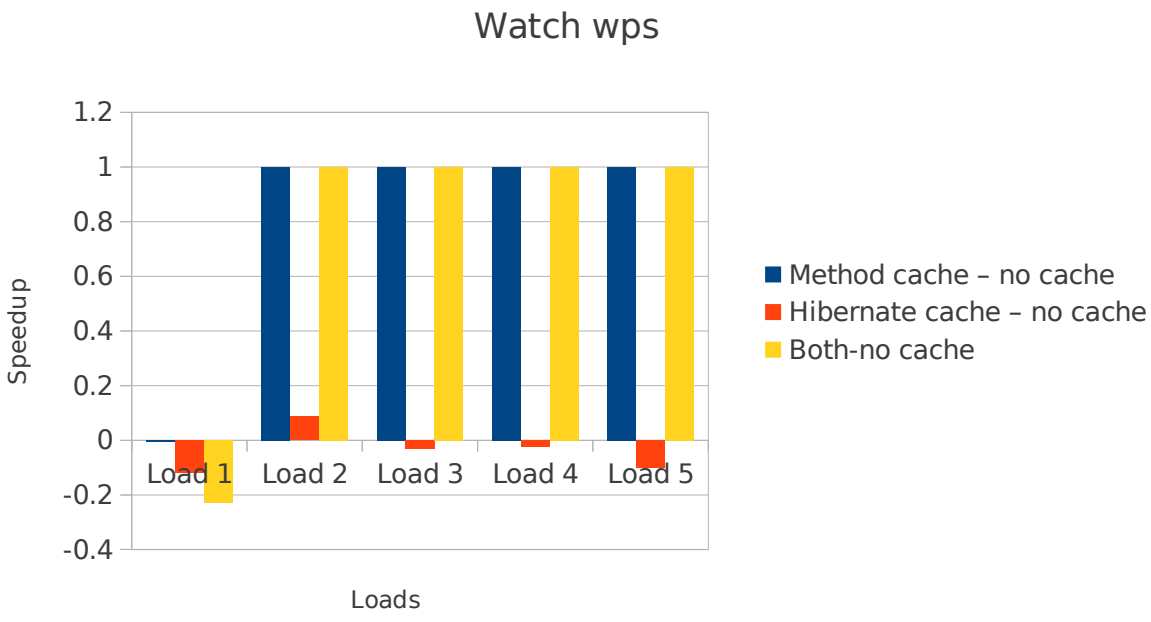
Σε αυτή τη σελίδα παρατηρούμε ένα ενδιαφέρον φαινόμενο. Στο πρώτο φόρτωμα όπως είναι αναμενόμενο οι χρόνοι είναι πολύ υψηλοί και φτιάνουν τα 30 δευτερόλεπτα, ενώ με χρήση της κρυφής μνήμης της Hibernate ο χρόνος αυξάνεται στα 35

δευτερόλεπτα, και στην περίπτωση που χρησιμοποιείται μαζί με την κρυφή μνήμη των μεθόδων φτάνει τα 39 δευτερόλεπτα. Στις επαναφορτώσεις της σελίδας παρατηρούμε ότι η χρήση μόνο της κρυφής μνήμης της Hibernate δεν έχει ευνοϊκές επιπτώσεις στο σύστημα αλλά φαίνεται να επιβαρύνει έως και 3 δευτερόλεπτα τον συνολικό χρόνο που απαιτείται για την εξυπηρέτηση του χρήστη. Ενώ όταν χρησιμοποιούμε την κρυφή μνήμη μεθόδων βλέπουμε ότι ο χρόνος είναι αμελητέος της τάξης των 15 miliseconds.

Σε αυτό το παράδειγμα βλέπουμε ότι η χρήση της κρυφής μνήμης της Hibernate μπορεί να μην είναι ευνοϊκή σε περιπτώσεις όπου πολλά δεδομένα απαιτούνται να προσκομισθούν από την βάση δεδομένων ενώ υπάρχουν περιορισμοί στο μέγεθος της κρυφής μνήμης που μπορούμε να διαθέσουμε.

Ακόμα χειρότερα είναι τα πράγματα όταν χρησιμοποιούμε την κρυφή μνήμη της Hibernate και παράγουμε αναφορές για διαφορετικά έργα με πολλούς εργαζόμενους και πολλά δεδομένα στη βάση. Σε αυτή την περίπτωση όταν η μνήμη γεμίσει στην επόμενη αναφορά που θα προσπαθήσουμε να παράγουμε επειδή όλα τα δεδομένα θα είναι καινούρια και δεν θα υπάρχουν στην κρυφή μνήμη θα πρέπει να αντικαταστήσουν υπάρχοντες εγγραφές. Ο χρόνος που απαιτούταν για την αντικατάσταση οδηγούσε σε μεγάλη αύξηση του συνολικού χρόνου απόκρισης της σελίδας με αποτέλεσμα η χρήση της κρυφής της Hibernate να μην εμφανίζεται σαν πρακτική λύση παρά όλες τις συνολικές βελτιώσεις και τα πλεονεκτήματα που φαίνεται να έχει.

Ακολουθεί η γραφική παράσταση επιτάχυνσης της απόδοσης με χρήση και των δύο κρυφών μνημών σε σχέση με την απουσία κρυφής μνήμης και την χρήση μόνο κρυφής μνήμης δεδομένων.



Γραφική 4.8: Επιτάχυνση φόρτωσης σελίδας παρακολούθησης πακέτων εργασίας

Παρατηρούμε στην παραπάνω γραφική παράσταση ότι η χρήση της κρυφής μνήμης των μεθόδων επιτυγχάνει καταπληκτική επιτάχυνση που τείνει να είναι σχεδόν στο 100% πιο αποδοτική από την απουσία κρυφής μνήμης. Η Hibernate δεν φαίνεται να συνεισφέρει καθόλου στην συγκεκριμένη περίπτωση, ενώ όπως αναφέραμε και πιο πάνω τα αποτελέσματα είναι ακόμα χειρότερα όταν την χρησιμοποιούμε και θέλουμε να παράγουμε αναφορές για περισσότερα έργα καθώς η αντικατάσταση των στοιχείων της όταν αυτή γεμίζει επιβαρύνει σημαντικά το σύστημα.

Η εφαρμογή της κρυφής μνήμης στο σύστημα φαίνεται να έχει ευεργετικές συνέπειες στον χρόνο εξυπηρέτησης των χρηστών από αυτό. Παρουσιάσαμε δύο ήδη κρυφών μνημών που μπορούν να εφαρμοστούν άμεσα στο σύστημα στο οποίο αναπτύσσουμε, το ένα είναι η κρυφή μνήμη δεδομένων και η άλλη η κρυφή μνήμη της Hibernate. Τα πλεονεκτήματα της κρυφής μνήμης δεδομένων είναι η ταχύτητά της και η ευκολία στην εφαρμογή, ενώ το βασικότερο μειονέκτημα είναι ότι όταν κάποια από τα δεδομένα αλλάζουν πρέπει να αδειάζουμε όλες τις μνήμες που σχετίζονται με τα δεδομένα που άλλαξαν. Τα πλεονεκτήματα της κρυφής μνήμης της Hibernate έρχονται να συμπληρώσουν τα μειονεκτήματα της προηγούμενης κρυφής μνήμης, με την αποθήκευση των αντικειμένων διαγραφή από την κρυφή μνήμη μόνο αυτών που δεν είναι ενημερωμένα με την βάση, ενώ φαίνεται να υστερεί σε ταχύτητα, ιδιαίτερα όταν πρόκειται για συναλλαγές με την βάση μεγάλου όγκου δεδομένων.

V. Συμπεράσματα και προοπτικές

Έχοντας φτάσει στην κατακλείδα της διπλωματικής αυτής εργασίας μπορούμε να βγάλουμε κάποια συμπεράσματα που προκύπτουν μετά από την εμπειρία της ανάπτυξης λογισμικού για ένα πληροφοριακό σύστημα.

Η ανάπτυξη τέτοιων λογισμικών για χρήση από έναν οργανισμό αποτελεί από μόνο του μία μεγάλη πρόκληση για τον κάθε έναν. Η ανάπτυξη λογισμικού τέτοιας φύσης μπορεί να είναι μία αρκετά πολύπλοκη διαδικασία ωστόσο τεχνολογίες όπως η Spring βοηθούν στο να έχεις έναν κώδικα οργανωμένο και με συνοχή που οι προσθήκες σε αυτό θα ακολουθούν την υπάρχουσα δομή κρατώντας τον κώδικα κατανοητό. Στην διπλωματική αυτή εργασία κληρονόμησα έναν μεγάλο σύστημα, με περίπου 100 χιλιάδες γραμμές κώδικα να υπάρχουν ήδη εκεί. Το δυσκολότερο κομμάτι ενός προγραμματιστή, τουλάχιστον στα πλαίσια της φοιτητικής του πορείας δεν είναι να αναπτύξει ένα δικό του τμήμα κώδικα αλλά να κατανοήσει ένα κώδικα που ήδη βρίσκεται εκεί και το ανέπτυξε κάποιος άλλος βάζοντας την δική του λογική πάνω στην υλοποίηση, λογική η οποία σπάνια συμφωνεί με αυτό που θα έκανε κάποιος άλλος, πόσο μάλλον όταν το μέγεθος του έργου είναι τόσο μεγάλο. Αυτό που έγινε γρήγορα ξεκάθαρο όμως είναι ότι όταν κάποιος χρησιμοποιεί τα κατάλληλα εργαλεία για την συγγραφή κώδικα και Frameworks για την μοντελοποίηση λογισμικού, όπως το Spring, το οποίο επιτρέπει ταχεία ανάπτυξη οργανωμένου κώδικα τότε το μέγεθος του έργου δεν παίζει τόσο δύσκολο ρόλο στο να καταλάβεις πως εσύ θα προσθέσεις τις δικές σου επεκτάσεις και θα συντηρήσεις το σύστημα γιατί όλα είναι οργανωμένα και ξεκάθαρα στον κώδικα και ξέρεις που ακριβώς να ψάξεις να διορθώσεις το κάθε πρόβλημα που μπορεί να προκύψει.

Στο ίδιο κλίμα όταν βιβλιοθήκες ORM, όπως η Hibernate, που δίνουν ένα αφηρημένο API για να επικοινωνεί μία εφαρμογή με την βάση δεδομένων κάνουν τη ζωή του προγραμματιστή εύκολη αφού μπορεί να ξεχάσει ότι το σύστημα από κάτω έχει μία βάση δεδομένων και να βλέπει τα πάντα σαν αντικείμενα. Αυτό επιταχύνει ακόμα περισσότερο την ανάπτυξη λογισμικού και κάνει την κατανόηση αυτού πολύ πιο απλή, καθώς οι συσχετίσεις μεταξύ των αντικειμένων είναι καλά ορισμένες μέσα στο σύστημα. Επίσης ο προγραμματιστής ξεχνάει τις ιδιαιτερότητες της κάθε βάσης αφού η Hibernate αναλαμβάνει να "μιλήσει" με αυτή.

Επιπλέον σε αυτά υπάρχουν πληθώρα εργαλείων σήμερα για την ανάπτυξη καλού κώδικα. Από τα IDE που ανεβάζουν την ταχύτητα της συγγραφής κώδικα με μεγάλο ρυθμό, μέχρι τα εργαλεία ελέγχου της ποιότητας του κώδικα τα οποία δίνουν συμβουλές στον προγραμματιστή για την βελτίωση της εικόνας του κώδικα, ώστε να είναι πιο κατανοήσημος, αλλά και της επίδοσης του κώδικα δίνοντας εναλλακτικές προτάσεις σε συγκεκριμένες γραμμές κώδικα που θα είχαν καλύτερη επίδοση.

Επιπλέον σε αυτά είδαμε τις θαυματουργικές επιπτώσεις της χρήσης κρυφών μνημών σε μεγάλα πληροφοριακά συστήματα στα οποία ο όγκος των δεδομένων μπορεί να τα κάνουν "βαριά". Είδαμε τις δύο λύσεις κρυφών μνημών που έχουμε με βάση τα εργαλεία στα οποία χτίστηκε το σύστημα και τα εφαρμόσαμε στην πράξη πετυχαίνοντας πολύ ενθαρρυντικά αποτελέσματα. Δεν καταφέραμε να χρησιμοποιήσουμε και τις δύο κρυφές μνήμες ο συνδυασμός των οποίων θα ήταν πολύ δυνατός κυρίως λόγω περιορισμένων μνήμης του συστήματος.

Ωστόσο η ιστορία δεν τελειώνει εδώ. Το σύστημα πάνω στο οποίο είχα την τύχη να συνεισφέρω έχει γίνει ένας ζωντανός οργανισμός. Το Ερευνητικό Ακαδημαϊκό Ινστιτούτο Τεχνολογίας Υπολογιστών το χρησιμοποιεί ήδη από το 2009. Τον Σεπτέμβρη του 2011 η επέκταση του συστήματος για την ψηφιοποίηση της έγκρισης των αιτημάτων μετακινήσεων τέθηκε σε δοκιμαστική λειτουργία και μετά από δύο μήνες καθιερώθηκε η χρήση του. Ωστόσο ο οργανισμός εξελίσετε και οι ανάγκες του μεγαλώνουν. Η δομή του συστήματος επιτρέπει την εύκολη τροποποίησή του ώστε να μπορέσει να εξελίσεται ταυτόχρονα με τον οργανισμό και να προσαρμόζεται στις ανάγκες του. Ενώ η προσθήκη περισσότερων επεκτάσεων μπορεί να γίνει σχεδόν τυφλοσούρτης.

Ακόμη περισσότερο ο τρόπος που σχεδιάστηκε το σύστημα και το κάνει εύκολα τροποποιήσιμο ανοίγει και προοπτικές για την χρήση του και από άλλους οργανισμούς. Οι λειτουργίες του συστήματος όπως τις περιγράψαμε είναι βασικές λειτουργίες που πολλοί οργανισμοί λίγο πολύ έχουν, ενώ η προσαρμογή των λειτουργιών και η επέκταση του με περισσότερες λειτουργίες είναι αρκετά εύκολη για τις ανάγκες συγκεκριμένου οργανισμού είναι αρκετά εύκολη υπόθεση.

Βιβλιογραφία

[1] Caucho, ‘‘Resin’’, http://www.caucho.com/resin/.

[2] Jet Brains Co, ‘‘Intellij idea’’, http://www.jetbrains.com/idea/

[3] Hibernate tutorial, http://docs.jboss.org/hibernate/orm/3.3/reference/en/html/tutorial.html

[4] Edgewall, ‘‘Trac’’, http://trac.edgewall.org/

[5] Βικιπαίδεια, ‘‘Hibernate framework’’ , http://el.wikipedia.org/ wiki/Hibernate_Framework.

[6] Wikipedia, ‘‘Apache ant’’, http://en.wikipedia.org/wiki/Apache_ Ant.

[7] Wikipedia, ‘‘Checkstyle’’, http://en.wikipedia.org/wiki/ Checkstyle.

[8] Wikipedia, ‘‘Hsqldb’’, http://en.wikipedia.org/wiki/HSQLDB.

[9] Wikipedia, ‘‘Object-relational mapping’’, http://en.wikipedia.org/ wiki/Object-relational_mapping.

[10] Wikipedia, ‘‘Pmd (software)’’, http://en.wikipedia.org/wiki/PMD_ (software).

[11] Wikipedia, ‘‘Subversion (software)’’, http://en.wikipedia.org/wiki/ Subversion_(software).

[12] Checkstyle 5.0, http://checkstyle.sourceforge.net/.

[13] Pro Spring, Rob Harrop and Jan Machacek

[14] Spring website, http://www.springsource.org/

[15] Maven website, http://maven.apache.org/

[16] Hibernate second level cache, http://ehcache.org/documentation/user-guide/hibernate

[17] Introduction to hibernate caching, http://www.javabeat.net/2007/10/introduction-to-hibernate-caching/

[18] Caching Methods with Spring 3 Annotations, http://ehcache.org/documentation/recipes/spring-annotations

[19] EHCAche, http://ehcache.org/documentation/index

[20] http://en.wikipedia.org/wiki/Project_management_information_system

[21] http://project-management-knowledge.com/definitions/p/project-management-information-system-pmis/