24 Μαΐου 2022

john missas

[Επωνυμία εταιρείας]

[Διεύθυνση εταιρείας]

τεχνολογια λογισμικου

ειδικο θεμα 3

**Αντικειμενοστραφής Ανάπτυξη Λογισμικού με UML**

**βασισμένη στη διαδικασία RUP**

1. **Εισαγωγή**
   1. Στόχοι της εργασίας

Οι στόχοι της εργασίας είναι να φτιάξουμε μια εφαρμογή τύπου ημερολογίου-ατζέντας στην οποία θα μπορεί ο χρήστης να συνδεθεί για να κάνει τις ακόλουθες λειτουργίες :

* Εισαγωγή και διαγραφή ενός ραντεβού στην πλατφόρμα από έναν χρήστη.
* Σύνδεση και αποσύνδεση του χρήστη από το σύστημα.
* Τροποποίηση και ενημέρωση υπαρχόντων ραντεβού που έχει φτιάξει ο χρήστης.
* Προσθήκη και άλλων ατόμων σε ένα ραντεβού.
  1. Ορισμός του προβλήματος προς επίλυση

Το πρόβλημα είναι να φτιάξουμε ένα λογισμικό που θα βοηθάει τον χρήστη να οργανώσει την καθημερινότητά του, ένα διαδικτυακό ημερολόγιο. Ένα λογισμικό τόσο λειτουργικό, όσο και εύχρηστο. Η εφαρμογή επιτρέπει στον χρήστη να έχει πρόσβαση ανά πάσα στιγμή στο πρόγραμμα της ημέρας του, καθώς και να κάνει τις απαραίτητες αλλαγές σε αυτό όπως αυτός επιθυμεί.

1. **Σύντομη παρουσίαση της RUP (Rational Unified Process)**

Η RUP είναι μια Διαδικασία Τεχνολογίας Λογισμικού (Software Engineering Process). Ο στόχος της είναι να διασφαλίσει την παραγωγή λογισμικού υψηλής ποιότητας που ικανοποιεί τις ανάγκες των τελικών χρηστών μέσα σε ένα συγκεκριμένο χρονοδιάγραμμα και κόστος.

Ο κύκλος ζωής του λογισμικού (Software life-cycle) σκιαγραφεί τη ζωή του προγράμματος λογισμικού από τη στιγμή της γέννησής του μέχρι τη στιγμή της αντικατάστασης ή της εγκατάλειψής του.

Ο κύκλος ζωής του λογισμικού στην RUP υποδιαιρείται σε τέσσερις συνεχόμενες φάσεις. Οι τέσσερις φάσεις είναι:

* Η Φάση Σύλληψης (Inception Phase),
* Η Φάση Επεξεργασίας (Elaboration Phase),
* Η Φάση Κατασκευής (Construction Phase) και
* Η Φάση Μετάβασης (Transition Phase)

Ορισμοί σχετικοί με την RUP:

* **Κύκλος ανάπτυξης:** Ένα πέρασμα από τις 4 φάσεις. Κάθε πέρασμα από τις 4 φάσεις παράγει μια νέα γενιά (generation) λογισμικού.
* **Κύκλος εξέλιξης:** Οι διαδοχικοί κύκλοι στην περίπτωση που το προϊόν συνεχίσει να

εξελίσσεται στην επόμενη γενιά του, ονομάζονται κύκλοι εξέλιξης (evolution cycles).

1. **Φάση: Έναρξη (Inception)**

3.1 Σύλληψη απαιτήσεων

Παρακάτω καταγράφονται οι λειτουργικές απαιτήσεις της εφαρμογής, δηλαδή οι υπηρεσίες που πρέπει να παρέχει το σύστημα και πως εκείνο πρέπει να αντιδρά στις διάφορες καταστάσεις και εισόδους:

* Να προσαρμόζεται η εφαρμογή στο μέγεθος της οθόνης του χρήστη, είτε είναι υπολογιστής, είτε είναι smartphone.
* Να αποθηκεύει τους χρήστες σε μία βάση δεδομένων (δυνατότητα αποθήκευσης στοιχείων).
* Δυνατότητα σύνδεσης και αποσύνδεσης στο σύστημα με εντολή του χρήστη.
* Να γίνεται εισαγωγή και διαγραφή ραντεβού από τον χρήστη. Η διαγραφή γίνεται μόνο από εκείνον τον χρήστη που δημιούργησε το ραντεβού.
* Να μπορεί να γίνει τροποποίηση των στοιχείων του ραντεβού (ώρα, ημερομηνία, συμμετέχοντες) από τον χρήστη που έχει δημιουργήσει το συμβάν.
* Να μπορούν να προστεθούν και άλλα άτομα σε ένα ραντεβού από τον χρήστη που έφτιαξε το ραντεβού αυτό.
* Να μπορεί να γίνει δημιουργία λογαριασμού από ένα νέο χρήστη.
* Δυνατότητα σύνδεσης και αποσύνδεσης σε υπάρχοντα λογαριασμό από τον χρήστη.

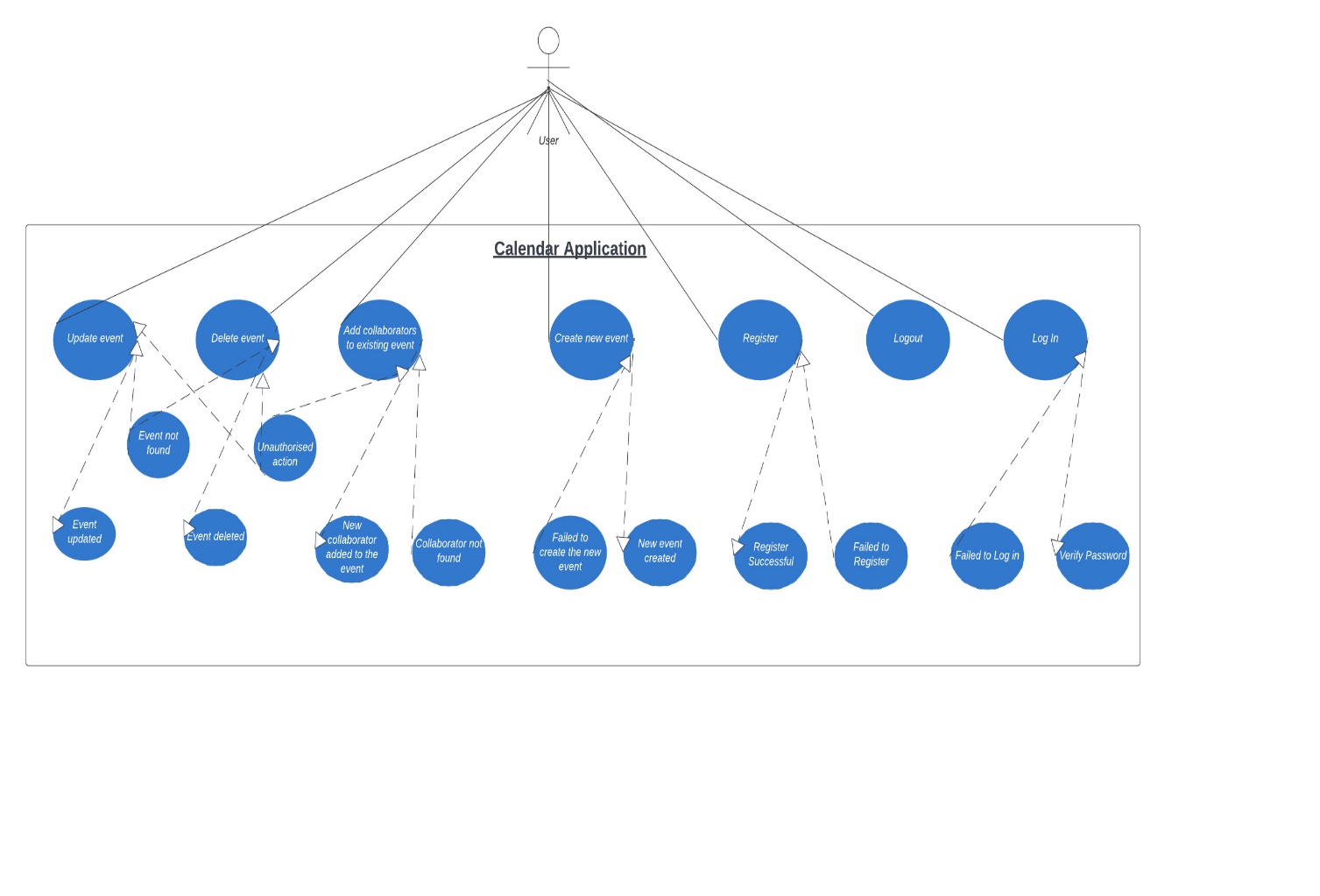
**3.2 Ανάλυση-Σχεδιασμός**

3.2.1 Διαγράμματα Περιπτώσεων Χρήσης

Σε αυτό το διάγραμμα περιγράφουμε τις λειτουργίες που είναι διαθέσιμες στον χρήστη όπως ορίζονται από την εκφώνηση καθώς και οι μετέπειτα καταλήξεις αυτών. Συγκεκριμένα ο χρήστης μπορεί να:

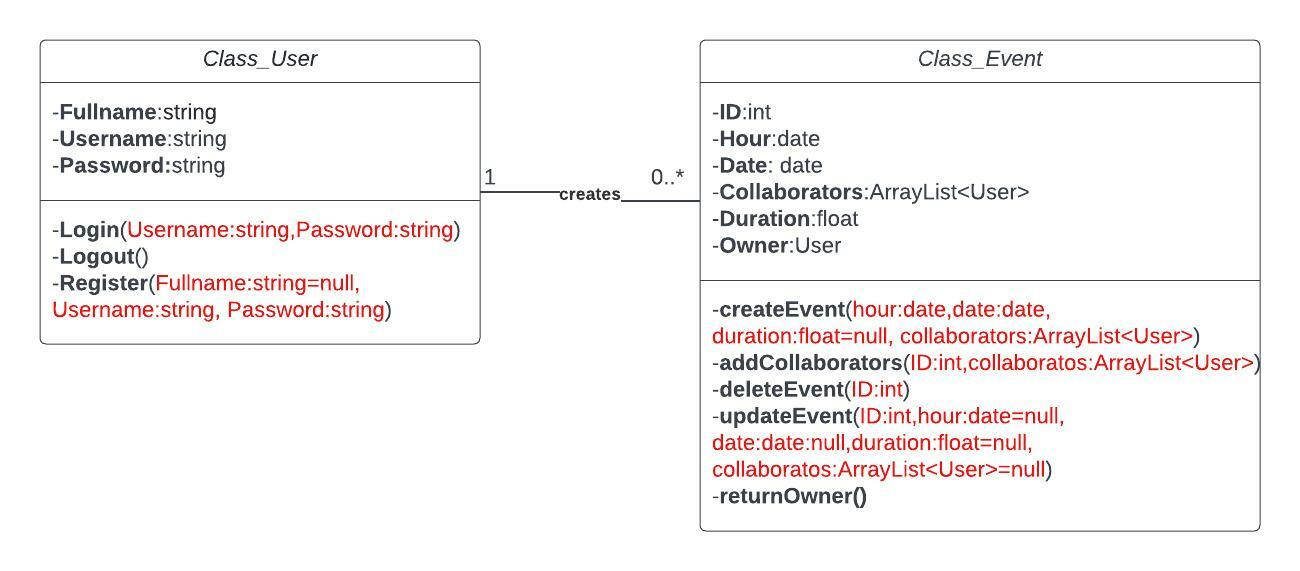
* Ενημερώσει ένα ραντεβού
* Διαγράψει ένα ραντεβού
* Προσθέσει συμμετέχοντες σε ένα ραντεβού
* Δημιουργήσει ένα ραντεβού
* Κάνει εγγραφή
* Κάνει έξοδο από την εφαρμογή
* Κάνει είσοδο στην εφαρμογή

Η φορά από τα βέλη δείχνει την χρονική συνέχεια εκτέλεσης. Για παράδειγμα ο χρήστης δεν μπορεί να κάνεις Verify Password αν δεν πετύχει η λειτουργία Login παρ’ όλα αυτά το Failed to Login μπορεί να πραγματοποιηθεί ανεξάρτητα από την επιτυχία του Login.



3.2.2 Διαγράμματα Τάξεων

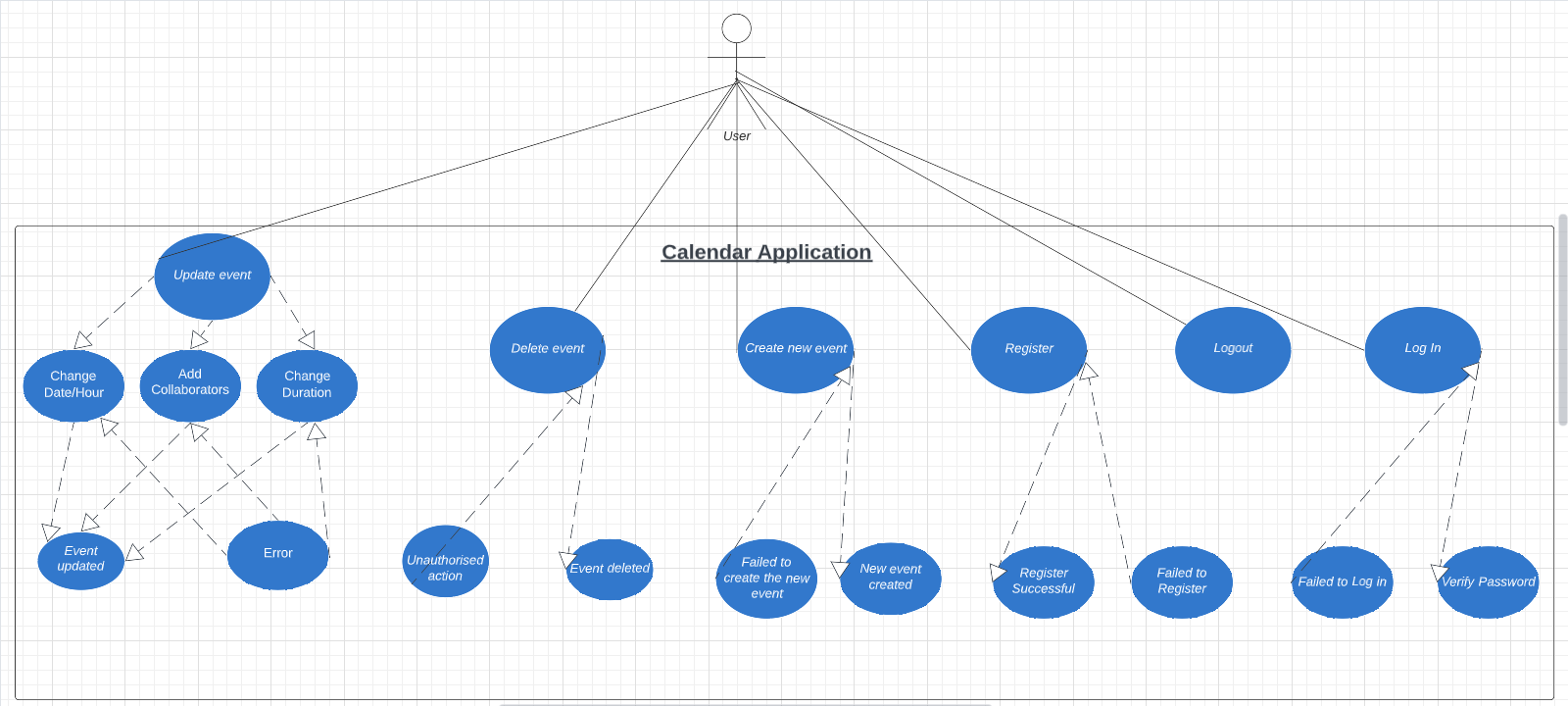
Σε αυτό το διάγραμμα αναλύουμε τις κλάσεις που θα χρησιμοποιήσουμε στην εφαρμογή μας. Έχουμε αποφασίσει να φτιάξουμε αρχικά δύο κλάσεις με τα παρακάτω χαρακτηριστικά και συναρτήσεις. Οι κλάσεις αυτές συνδέονται καθώς τα αντικείμενα Event φτιάχνονται από τα αντικείμενα User. Ένα αντικείμενο User μπορεί να φτιάξει όσα Event επιθυμεί



1. **Φάση: Εκπόνηση Μελέτης (Elaboration)**
   1. Ανάλυση-Σχεδιασμός
      1. Διαγράμματα Περιπτώσεων Χρήσης (2η έκδοση)

Στην 2η έκδοση του διαγράμματος περιπτώσεων χρήστη χρειάστηκε να:

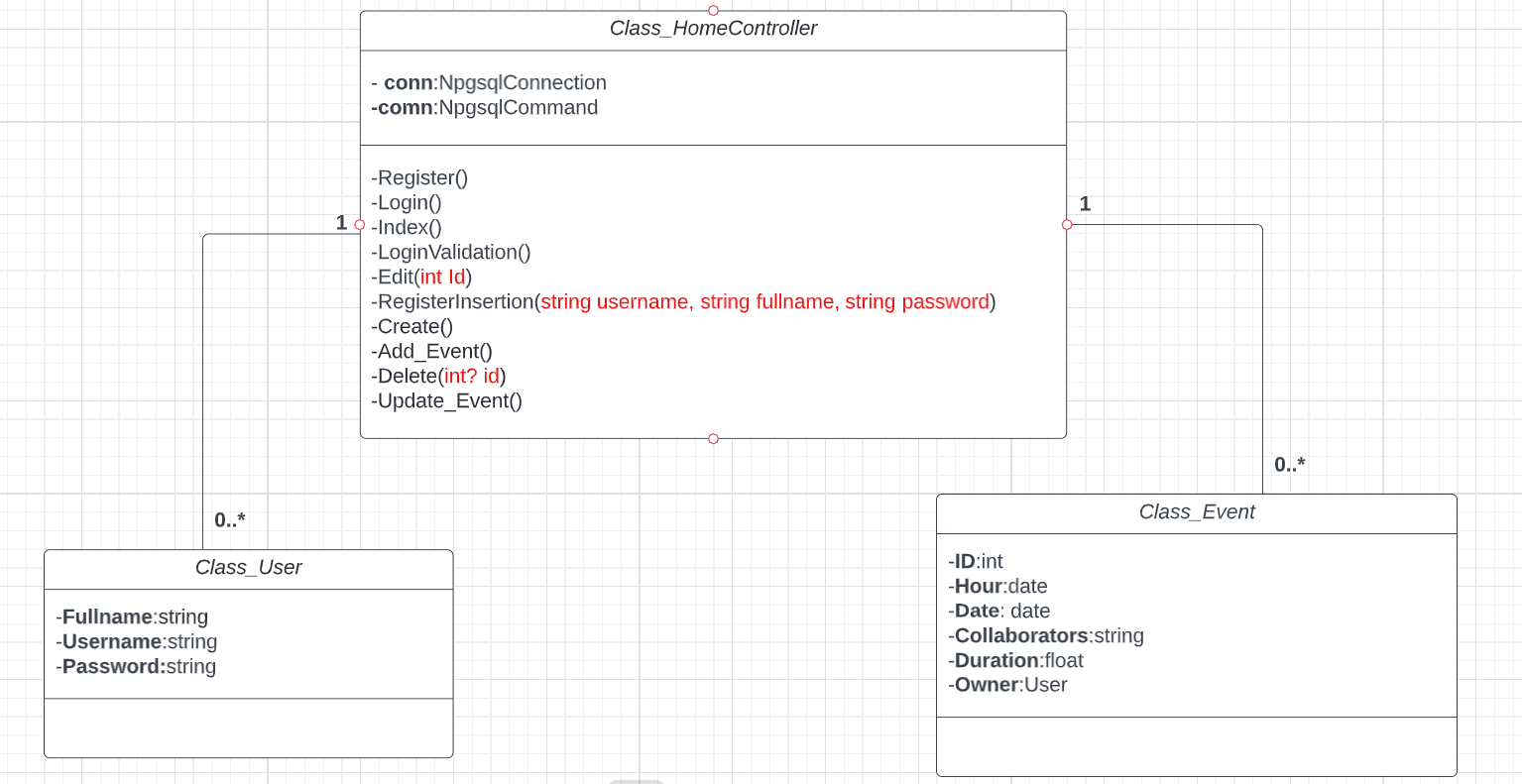
* Διαγράψουμε το Event not found μιας και η διαγραφή σαν λειτουργία εμφανίζεται μονάχα στα υπάρχοντα ραντεβού.
* Διαγράψαμε το Unauthorised Action στο delete Event μια και δεν υπάρχει περίπτωση η διαγραφή να μην είναι έγκυρη, αφού εμφανίζεται μόνο στα ήδη υπάρχοντα ραντεβού.
* Φτιάξαμε το update Event όπως ακολουθώντας τα πρότυπα της εφαρμογής, δηλαδή πρώτα διαλέγει ο χρήστης να κάνει Update και μετά κάνει αλλαγή σε συγκεκριμένο πεδίο του ραντεβού, άρα δεν είναι λειτουργίες που του δίνονται απευθείας.

****

* + 1. Διαγράμματα Τάξεων (2η έκδοση)

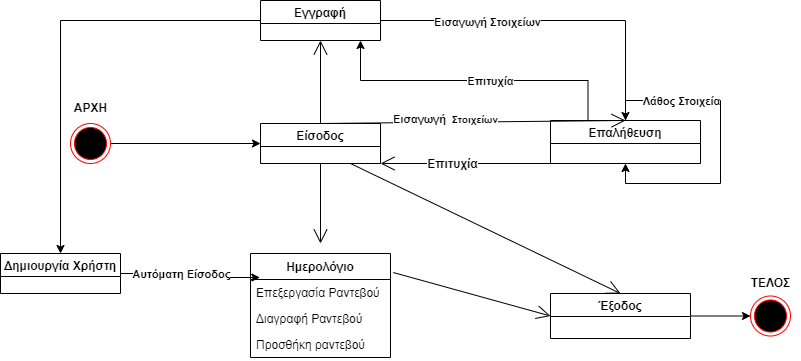
Στην 2η έκδοση του διαγράμματος τάξεων χρήστη χρειάστηκε να:

* Να βγάλουμε τις συναρτήσεις από τις κλάσεις User και Event καθώς πλέον αυτές εκτελούνται από τον HomeController, τον οποίο και προσθέσαμε μαζί με τις νέες συναρτήσεις
* Αλλάξαμε τον τύπο του πεδίου Collaborators σε String από Arraylist <String> καθώς o προηγούμενος τύπος δεν μας εξυπηρετούσε στην εφαρμογή



* + 1. **Διαγράμματα Αντικειμένων (1η έκδοση)**
    2. **Διαγράμματα Συνεργασίας (1η έκδοση)**
    3. **Διαγράμματα Σειράς (1η έκδοση)**
    4. **Διαγράμματα Δραστηριοτήτων (1η έκδοση)**
    5. Διαγράμματα Καταστάσεων (1η έκδοση)

Στο διάγραμμα αυτό αναπαρίσταται η ροή μεταξύ των καταστάσεων που περνάνε τα αιτήματα του χρήστη προς την εφαρμογή. Για παράδειγμα το αίτημα του χρήστη για είσοδο θα πρέπει πρώτα να περάσει από την επαλήθευση ώστε τα στοιχεία του να ελεγχθούν. Η προαναφερθείσα διαδικασία μπορεί να φανεί και από τα βέλη που συνδέουν τις καταστάσεις μεταξύ τους. Η χρονική σειρά φαίνεται από την φορά αυτών των βελών.



* + 1. **Διαγράμματα Εξαρτημάτων (1η έκδοση)**
    2. Διαγράμματα Διανομής (1η έκδοση)

Σε κανονικές συνθήκες αυτό το διάγραμμα δείχνει τη σύνδεση του υλικού σε μία εφαρμογή. Στην περίπτωσή μας όμως, καθότι η εφαρμογή μας δεν διαθέτει υλικό, θα υποθέσουμε πως αποτελείται από 3 κομμάτια το Front και Back End της σελίδας μας και την βάση δεδομένων μας. Τα κομμάτια αυτά συνδέονται μεταξύ τους είτε με την βοήθεια Connection String (DB🡪BACK END) είτε με την βοήθεια μιας κλάσης(HomeController) (BACK END 🡪 FRONT END). Οι φορές δείχνουν την αλληλεπίδραση καθώς είναι διπλής κατεύθυνσης και οι αριθμοί με πόσα επικοινωνούν. Για παράδειγμα το BACK END μπορεί ναεπικοινωνεί με πολλές από τις .cshtml σελίδες οπότε λέμε 0..\* (καμία ή άπειρες).



* 1. ***Υλοποίηση-Έλεγχος***
     1. **Υλοποίηση: 1η εκτελέσιμη έκδοση**
     2. **Αναφορά ελέγχου για την 1η εκτελέσιμη έκδοση**

**-------------*23 Μαΐου 2013---------Παρουσίαση στην τάξη*-----------------**

1. **Φάση: Κατασκευή (Construction)**
   1. ***Ανάλυση-Σχεδιασμός***
      1. **Διαγράμματα Περιπτώσεων Χρήσης (3η έκδοση)**
      2. **Διαγράμματα Τάξεων (3η έκδοση)**
      3. **Διαγράμματα Αντικειμένων (2η έκδοση)**
      4. **Διαγράμματα Συνεργασίας (2η έκδοση)**
      5. **Διαγράμματα Σειράς (2η έκδοση)**
      6. **Διαγράμματα Δραστηριοτήτων (2η έκδοση)**
      7. **Διαγράμματα Καταστάσεων (2η έκδοση)**
      8. **Διαγράμματα Εξαρτημάτων (2η έκδοση)**
      9. **Διαγράμματα Διανομής (2η έκδοση)**

***5.2 Υλοποίηση-Έλεγχος***

**5.2.1 Υλοποίηση: Τελική εκτελέσιμη έκδοση**

**5.2.2 Αναφορά ελέγχου για την τελική εκτελέσιμη έκδοση**

**6. Εγχειρίδιο Χρήστη**

* 1. ***Σύντομη παρουσίαση του προγράμματος***
  2. ***Παρουσίαση σεναρίων λειτουργίας***