ΠΑΝΕΠΙΣΤΗΜΙΟ
ΙΩΑΝΝΙΝΩΝ

ΣΧΟΛΗ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΜΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΔΙΚΤΥΩΝ

ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΓΡΑΦΙΚΗ ΔΙΕΠΑΦΗ ΠΑΡΟΥΣΙΑΣΗΣ ΚΑΙ ΤΡΟΠΟΠΟΙΗΣΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΜΑΘΗΜΑΤΩΝ ΠΑΝΕΠΙΣΤΗΜΙΑΚΟΥ ΤΜΗΜΑΤΟΣ

Χαράλαμπος Κρανάς

Επιβλέπων: Χρήστος Γκόγκος

Άρτα, Σεπτέμβριος, 2021

GRAPHICAL USER INTERFACE FOR PRESENTING AND MODIFYING UNIVERSITY COURSE TIMETABLES

Εγκρίθηκε από τριμελή εξεταστική επιτροπή

Τόπος, Ημερομηνία

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Επιβλέπων καθηγητής

   Όνομα Επίθετο,

   τίτλος, βαθμίδα

2. Μέλος επιτροπής

   Όνομα Επίθετο,

   τίτλος, βαθμίδα

3. Μέλος επιτροπής

   Όνομα Επίθετο,

   τίτλος, βαθμίδα

Δήλωση μη λογοκλοπής

Δηλώνω υπεύθυνα και γνωρίζοντας τις κυρώσεις του Ν. 2121/1993 περί Πνευματικής Ιδιοκτησίας, ότι η παρούσα μεταπτυχιακή εργασία είναι εκ ολοκλήρου αποτέλεσμα δικής μου ερευνητικής εργασίας, δεν αποτελεί προϊόν αντιγραφής ούτε προέρχεται από ανάθεση σε τρίτους. Όλες οι πηγές που χρησιμοποιήθηκαν (κάθε είδους, μορφής και προέλευσης) για τη συγγραφή της περιλαμβάνονται στη βιβλιογραφία.

Επίθετο, Όνομα

Υπογραφή

ΕΥΧΑΡΙΣΤΙΕΣ

Η συγγραφή της παρούσας διπλωματικής εργασίας σημαίνει και το τέλος ενός μεγάλου ταξιδιού αναζήτησης γνώσης για την επαγγελματική μου πορεία και εφοδίων για την προσωπική μου ζωή. Είναι όμως και η αρχή για ένα βήμα που ήδη έχει γίνει. Θα ήθελα να ευχαριστήσω όλους του καθηγητές του μεταπτυχιακού προγράμματος και ιδιαίτερα θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή της διπλωματικής Δρ. Γκόγκο Χρήστο για την υπομονή και την ηθική υποστήριξη που υπέδειξε στο πρόσωπο μου όλο αυτό το διάστημα. Θα ήθελα επίσης να ευχαριστήσω την οικογένεια μου για όλη την κατανόηση που υπέδειξαν και που χωρίς εκείνους δεν θα ήταν δυνατή η ολοκλήρωση των μεταπτυχιακών μου σπουδών. Τέλος θα ήθελα να συμπεριλάβω και έναν άνθρωπο που προστέθηκε πρόσφατα στη ζωή μου αλλά ήταν εκείνη που πίστεψε στον εαυτό μου και μου έδωσε την δύναμη να μην τα παρατήσω και να το φέρω εις πέρας.

God is in the details.

Ludwig Mies van der Rohe, 1886 - 1969

Γερμανός Αρχιτέκτονας

ΠΕΡΙΛΗΨΗ

Κάθε χρόνο πολλά νέα παιδιά ξεκινάνε τις σπουδές τους στην τριτοβάθμια εκπαίδευση με στόχο να πραγματοποιήσουν τα όνειρα τους και να αποκτήσουν γνώση και πείρα πάνω στον τομέα που έχουν επιλέξει. Κάθε χρόνο τα πανεπιστημιακά ιδρύματα έρχονται αντιμέτωπα με το πρόβλημα το εβδομαδιαίου προγράμματος. Ένα πρόβλημα που σε μια πιο αφηρημένη εκδοχή του υπάρχει από απαρχής του κόσμου. Πολλές και διαφορετικές οντότητες πρέπει να τοποθετηθούν με ένα βέλτιστο τρόπο ώστε να δημιουργήσουν ένα πρόγραμμα το οποίο θα εξυπηρετεί όλους όσους το χρησιμοποιούν. Η παρούσα διπλωματική εργασία μέσα από μια ολοκληρωμένη εφαρμογή ανάπτυξης λογισμικού παρουσιάζει το πρόβλημα του χρονοπρογραμματισμού καθώς επίσης και τις τεχνολογίες και την επικοινωνία που χρειάζονται για την υλοποίηση της.

Λέξεις-κλειδιά: Χρονοπρογραμματισμός , ανάπτυξη λογισμικού, Spring Boot

ABSTRACT

Every year a lot of new students are going to enroll on a university to achieve personal goals of knowledge and experience in a field which have been chosen by them. Every year universities faces the same scheduling problem. A problem which in an abstract version exists from the beginning of the world. Lots of different entities should places in an optimal way which deserved every person who use it. This thesis through a full stack application presents the problem of university course timetabling along with the technologies and communication among them.

Keywords: University course timetabling, Software Development, Spring Boot

ΕΙΣΑΓΩΓΗ

Η ανάπτυξη λογισμικού είναι ο τομέας εκείνος της επιστήμης των υπολογιστών όπου η θεωρία συναντάει την πράξη. Μέσα από μια σειρά βημάτων ο προγραμματιστής καθορίζει το πρόβλημα καθώς επίσης τα εργαλεία και τις μεθοδολογίες που θα ακολουθήσει για να το λύσει. Εφαρμόζει όλες τις γνώσεις που έχει αποκτήσει και λαμβάνει υπόψιν του την πολυπλοκότητα του εκάστοτε προβλήματος. Σκοπός της παρούσας διπλωματικής εργασίας είναι να παρουσιάσει ένα κλασικό πρόβλημα της επιστήμης των υπολογιστών, το πρόβλημα του χρονοπρογραμματισμού των μαθημάτων ενός πανεπιστημιακού ιδρύματος και πιο συγκεκριμένα το προγράμμα μαθημάτων τέτοιο ώστε οι φοιτητές να έχουν τη δυνατότητα να δηλώσουν ποια μαθήματα θα παρακολουθήσουν και μέσα από συνδυασμό μοντέρνων τεχνολογιών όπως το Spring Boot και το Angular να δημιουργηθεί μια εφαρμογή που να το αποτυπώνει. Να γίνει αναφορά σε λύσεις που έχουν προταθεί μέχρι στιγμής όπως ο χρωματισμός γραφήματος, η tabu search κ.α. Να αναλυθούν εργαλεία όπως ένα ολοκληρωμένο περιβάλλον ανάπτυξης λογισμικού και ένα σύστημα που παρακολουθεί την πρόοδο του εκάστοτε έργου, τεχνολογίες που επιτρέπουν την αποθήκευση των δεδομένων και πως γίνεται η επικοινωνία με την εφαρμογή, το πως μπορεί ο προγραμματιστής να ελαχιστοποιήσει τον κώδικα που θα γράψει σε μια βάση δεδομένων και πως να εγκαταστήσει τις βιβλιοθήκες που χρειάζεται για την διεκπεραίωση του έργου του.

Στόχος είναι να μελετηθούν και να περιγράφουνε οι παραπάνω τεχνολογίες σε βάθος ώστε στο τέλος της ο αναγνώστης να είναι εξοικειωμένος μαζί τους και να μπορεί να τις χρησιμοποιήσει.

Πίνακας περιεχομένων

1.Περιγραφή του προβλήματος

Το πρόβλημα του χρονοδιαγράμματος (Nurlida Basir, 2013) (Prabodanie, 2016) γνωστό και ως πρόβλημα χρονοπρογραμματισμού είναι προβλήματα που έχουν εφαρμογή σε ένα ευρύ φάσμα πεδίων όπως στην εκπαίδευση (πρόγραμμα μαθημάτων πανεπιστημιακών ιδρυμάτων - σχολείων), στον τομέα της υγείας (βάρδιες νοσοκόμων), στα Μέσα Μαζικής Μεταφοράς (δρομολόγια λεωφορείων) καθώς και στον αθλητισμό (πρόγραμμα παιχνιδιών). Τα παραπάνω προβλήματα ανήκουν στην κατηγορία των NP-Hard προβλημάτων και είναι δύσκολα να υπάρξει μια καθολική λύση διότι οι περιορισμοί που θέτονται ποικίλουν ανάλογα με το πρόβλημα.

Ένα πρόβλημα χρονοδιαγράμματος μπορεί να οριστεί ως ένα πρόβλημα ανάθεσης ενεργειών σε δραστηριότητες, όπως είναι οι χρονικές περίοδοι, κάτω από συγκεκριμένους περιορισμούς που συνήθως αναφέρονται ως σκληροί (hard), οι οποίοι διαφέρουν ανάλογα με το πρόβλημα. Στόχος είναι να βρεθεί μια εφικτή λύση, που να αναθέτει όλες τις ενέργειες στις δραστηριότητες, ενώ ελαχιστοποιεί το άθροισμα του βάρους ή της ποινής ανταποκρινόμενος στην παραβίαση των εύκολων περιορισμών.

1.1 Οι τρεις κατηγορίες στο πρόβλημα του προγράμματος πανεπιστημίων

1.1.1 Examination Timetabling — EX TT

- Πρόγραμμα εξεταστικής Examination timetabling (Ex-TT) (Toth, 2015): Ο σχεδιασμός της εξεταστικής μέσα σε μια συγκεκριμένη χρονική περίοδο ενώ ικανοποιούνται οι σκληρή περιορισμοί

1.1.2 Post-Enrolment-based—Course Timetabling

- Post-enrollment course timetabling (PE - CTT) (Toth, 2015): Η παραγωγή προγράμματος τέτοιο ώστε όλοι οι σπουδαστές να μπορούν να παρακολουθήσουν μαθήματα τα οποία έχουν ήδη δηλώσει.

1.1.3 Curriculum-Based—Course Timetabling

- Curriculum-based course timetabling (CB - CTT) (Toth, 2015): Η κατασκευή προκαθορισμένου προγράμματος μαθημάτων τέτοιο ώστε οι φοιτητές να έχουν τη δυνατότητα να δηλώσουν ποια μαθήματα θα παρακολουθήσουν.

1.2 Κανόνες - περιορισμοί του προβλήματος

Ο χρονοπρογραμματισμός είναι η ανάθεση χρονικών περιόδων σε μια σειρά γεγονότων.

Για ένα πανεπιστήμιο με Np καθηγητές, Nq μαθήματα, Nx αίθουσες και Ns φοιτητές, είναι απαραίτητο να προγραμματιστούν ζευγάρια Nl καθηγητών-μαθημάτων μέσα σε χρονικά όρια Nt και να παραχθεί ένα έγκυρο εβδομαδιαίο πρόγραμμα. Για να θεωρηθεί ένα πρόγραμμα έγκυρο πρέπει ο καθηγητής, ο φοιτητής και το μάθημα να μην είναι σε περισσότερα από ένα μέρη για μια χρονική στιγμή καθώς επίσης μια αίθουσα να περιμένει να φιλοξενήσει περισσότερους φοιτητές από ότι μπορεί να εξυπηρετήσει.

Οι περιορισμοί (Coston, 1992) για το συγκεκριμένο πρόβλημα μπορεί να είναι σκληροί, μέτριοι ή εύκολοι. Οι μέτριοι και οι εύκολοι περιορισμοί είναι συνδεδεμένοι με ένα κόστος ή μια ποινή και αν δεν γίνεται να ικανοποιηθούν τότε στόχος είναι να ελαχιστοποιηθεί αυτό το κόστος. Οι εύκολοι περιορισμοί έχουν χαμηλότερη προτεραιότητα άρα και χαμηλότερο κόστος σε σχέση με τους μέτριους. Οι σκληροί περιορισμοί πρέπει να ικανοποιηθούν άρα πρέπει να έχουν μηδενικό κόστος. Ένα εφικτό εβδομαδιαίο πρόγραμμα είναι αυτό που έχει ικανοποιήσει όλους τους σκληρούς περιορισμούς.

1.2.1 Σκληροί περιορισμοί

Οι σκληροί (Nurlida Basir, 2013) (M. A. Saleh Elmohamed, 1998) περιορισμοί είναι συνήθως περιορισμοί που από την φύση τους δεν μπορούν να παραβιαστούν.
Τέτοιοι περιορισμοί είναι αυτοί που δεν μπορούν να πραγματοποιηθούν την ίδια χρονική στιγμή:

- Διαλέξεις που γίνονται από τον ίδιο καθηγητή
- Διαλέξεις που γίνονται στην ίδια αίθουσα
- Διαλέξεις και τα εργαστήρια τους

Επίσης σκληροί περιορισμοί είναι και αυτοί που έχουν να κάνουν με τις αίθουσες και την χωρητικότητα τους

- Μια αίθουσα δεν πρέπει να ανατεθεί σε ένα συγκεκριμένο μάθημα εκτός και αν η χωρητικότητα της είναι μεγαλύτερη ή ίση με τους φοιτητές που έχουν δηλώσει το μάθημα
- Κάποια εργαστήρια γίνονται μόνο σε συγκεκριμένες αίθουσες

1.2.2 Μέτριοι περιορισμοί

Οι μέτριοι (Nurlida Basir, 2013) (M. A. Saleh Elmohamed, 1998) περιορισμοί είναι συνήθως αυτοί που ακροβατούν ανάμεσα στους σκληρούς και τους εύκολους. Στα περισσότερα προβλήματα χρονοπρογραμματισμού οι μέτριοι περιορισμοί ορίζονται οι συγκρούσεις χρόνου και χώρου. Ωστόσο θεωρούνται μέτριοι αυτοί οι περιορισμοί διότι μπορούν να αποφευχθούν κάνοντας μικρές αλλαγές στις προδιαγραφές του προβλήματος. Ένα παράδειγμα είναι οι προτιμήσεις των φοιτητών. Δεν γίνεται να τους ικανοποιήσεις όλους στο ποια μαθήματα θέλουν να παρακολουθήσουν ή τις ώρες που θέλουν.

Οι μέτριοι περιορισμοί είναι συσχετισμένοι με μεγάλες ποινές και στο τελικό πρόγραμμα αυτές οι ποινές πρέπει να προσεγγίζουν το μηδέν.
Μερικά παραδείγματα:

- Απέφυγε χρονικές συγκρούσεις σε μαθήματα που έχουν τους ίδιους φοιτητές.
- Προϋποθέσεις και κριτήρια για τις συναντήσεις

1.2.3 Εύκολοι περιορισμοί

Οι εύκολοι (Nurlida Basir, 2013) (M. A. Saleh Elmohamed, 1998) περιορισμοί είναι συνήθως που δεν έχουν να κάνουν με χρονικές συγκρούσεις και για αυτό και έχουν χαμηλό κόστος. Στόχος είναι να ελαχιστοποιηθεί το κόστος παρόλο που δεν περιμένει κανένας να γίνει μηδέν.

Μερικά παραδείγματα εύκολων περιορισμών:

- Για κάθε φοιτητή να ισορρόπησε το πρόγραμμα στις μέρες Δευτέρα-Τετάρτη-Παρασκευή με τις μέρες Τρίτη-Πέμπτη
- Ισορρόπησε ή άπλωσε τις διαλέξεις σε όλη την εβδομάδα
- Τα μαθήματα μπορεί να γίνονται σε συνεχόμενες χρονικές περιόδους
- Το διάλλειμα για φαγητό πρέπει να λαμβάνεται υπόψιν
- Οι καθηγητές ίσως έχουν προτιμήσεις σε συγκεκριμένες αίθουσες
- Οι καθηγητές ίσως έχουν προτιμήσεις στην ώρα του μαθήματος

Μερικοί εύκολοι περιορισμοί ίσως έχουν μεγαλύτεροι προτεραιότητα και συνεπώς μεγαλύτερο κόστος σε σχέση με άλλους. Αυτοί συνήθως έχουν να κάνουν με τις προτιμήσεις των καθηγητών σε σχέση με αυτές των φοιτητών.

1.3 Χρωματισμός γραφήματος.

Το 1736 η θεωρία των γράφων γεννιέται από τις γέφυρες του Konigsberg και το πρόβλημα που ανέδειξε ο μαθηματικός Euler όπου αργότερα ονομάστηκε Eulerian graph. Την ίδια δεκαετία, ο Gustav Kirchhoff καθιερώνει την ιδέα του δέντρου, έναν συνδεδεμένο Γράφο χωρίς κύκλους που χρησιμοποιήθηκε για τον υπολογισμό των ηλεκτρικών δικτύων και των κυκλωμάτων και αργότερα για να μετρήσουν τα χημικά μόρια. Το 1840 ο A.F Mobius εμφάνισε την ιδέα του ολοκληρωμένου γράφου και του διμερή γράφου. Το 1852, ο Thomas Gutherie βρήκε το διάσημο πρόβλημα των τεσσάρων χρωμάτων. Τα πρώτα αποτελέσματα για τον χρωματισμό ενός γραφήματος αφορούσαν αποκλειστικά επίπεδους (planar) γράφους υπό την μορφή του χρωματισμού χαρτών. Παρόλο που το διάσημο αυτό πρόβλημα είχε έρθει στην επιφάνια, λύθηκε μετά από σχεδόν έναν αιώνα από τους Kenneth Appel και Wolfgang Haken. Το 1890, ο Heawood απέδειξε ότι το θεώρημα των πέντε

χρωμάτων, υποστηρίζοντας ότι κάθε επίπεδος χάρτης μπορεί να χρωματιστεί με όχι περισσότερα από πέντε χρώματα. Ο χρωματισμός γραφήματος έχει πολλές εφαρμογές σε πραγματικά προβλήματα όπως ο χρωματισμός ενός χάρτη, στον χρονοπρογραμματισμό, τον παράλληλο υπολογισμό, την σχεδίαση δικτύων. Επίσης έχει ικανοποιητική εφαρμογή σε πολλά σύνθετα προβλήματα όπως που έχουν να κάνουν με την βελτιστοποίηση, όπως είναι η επίλυση συγκρούσεων ή η εύρεση της καλύτερης θέσης σε μια σειρά γεγονότων, τέτοια προβλήματα είναι το εβδομαδιαίο πρόγραμμα για ένα πανεπιστήμιο ή το πρόγραμμα της εξεταστικής περιόδου

Σε κάθε πανεπιστημιακό ίδρυμα, τα δύο πιο συνηθισμένα προβλήματα που αντιμετωπίζει είναι το πρόβλημα του εβδομαδιαίου προγράμματος και αυτό του προγράμματος της εξεταστικής περιόδου. Το πρόβλημα του χρονοπρογραμματισμού είναι αποδεδειγμένα ένα NP Complete πρόβλημα αλλά η αντίστοιχη βελτιστοποίηση του είναι NP Hard. Ως εκ τούτου μια ευριστική προσέγγιση είναι επιθυμητή για να βρεθεί η κοντινότερη λύση που θα είναι υπολογίσιμη σε ένα λογικό χρόνο. Όταν κατασκευάζεται το πρόγραμμα ενός πανεπιστημίου είναι προφανές ότι τα μαθήματα που διδάσκονται από τον ίδιο καθηγητή καθώς και αυτά που χρησιμοποιούν συγκεκριμένες αίθουσες δεν μπορούν να τοποθετηθούν στην ίδια χρονική περίοδο. Συνεπώς το πρόβλημα του να καθοριστούν οι ελάχιστες ή ένα αποδεκτός αριθμός χρονικώς περιόδων ώστε να μπορεί να φτιαχτεί ένα πρόγραμμα είναι ένα τυπικό πρόβλημα χρωματισμού γραφήματος (Leighton, 1979) (Leighton, 1979) (E.K.Burke) όπου πρέπει να ελαχιστοποιηθούν οι συγκρούσεις μεταξύ των μαθημάτων. Έτσι η βέλτιστη λύση σε ένα τέτοιο πρόβλημα είναι βρεθούν τα ελάχιστα χρώματα για τον αντίστοιχο γράφο. Ο χρωματισμός ενός γραφήματος είναι αποδεδειγμένα ένα NP Complete πρόβλημα και δεν υπάρχει κάποιος αλγόριθμος τέτοιος, για κάθε γράφο, που να βρίσκει την βέλτιστη τιμή χρωμάτων για τους κόμβους του γραφήματος. Μερικοί από τους πιο διαδεδομένους είναι ο αλγόριθμος Saturation , ο αλγόριθμος Recursive Largest First, Simulated Annealing και ο αλγόριθμος Greedy.

Η επίλυση του προβλήματος του χρονοπρογραμματισμού με τη βοήθεια των υπολογιστών έχει μακρά και ποικίλη ιστορία. Το 1967, το πρόβλημα του εβδομαδιαίου προγράμματος ενός πανεπιστήμιου εκφράστηκε μέσω του χρωματισμού ενός γραφήματος. Ο Welsh και Powell ικανοποίησαν τη σχέση μεταξύ του ανάμεσα στο πρόβλημα και στο χρωματισμό γραφήματος και ανέπτυξαν ένα γενικό αλγόριθμο για τον χρωματισμό για την εύρεση του ελάχιστου αριθμού χρωμάτων (ή σχεδόν την επίλυση). Το 1969, ο αλγόριθμος γράφων του Wood's χειριζόταν δύο n X n πίνακες, όπου το n ήταν ο αριθμός των κορυφών στο γράφο, ένας πίνακας C με τις συγκρούσεις χρησιμοποιούταν για να παρουσιάσει πια

ζευγάρια κορυφών έπρεπε να έχουν διαφορετικό χρώμα σύμφωνα με τους περιορισμούς και ένας παρόμοιος πίνακας S χρησιμοποιούνταν για να καθορίσει πια ζευγάρια πρέπει να έχουν το ίδιο χρώμα. Ο Dutton και Bingham το 1981 εισήγαγαν δύο από τους πλέον διάσημους αλγόριθμους για τον χρωματισμό γραφήματος. Λαμβάνοντας υπόψιν ένα προς ένα τα χρώματα και μέσω ενός μηχανισμού όπου συνεχόμενα ενοποιούσε τις δύο κορυφές με τις περισσότερες κοινές κορυφές. Κατά την ολοκλήρωση του αλγόριθμου ίδια χρώματα τοποθετούνταν στις κορυφές εκείνες που είχαν ενοποιηθεί. Το 1991, οι Johnson, Aragon, McGeoch και Schevon εφάρμοσαν και έκαναν τεστ σε τρεις διαφορετικές προοπτικές για τον χρωματισμό γραφήματος σύμφωνα με την προσομοιωμένη ανόπτηση και να καταλήγουν ότι η προσομοιωμένη ανόπτηση μπορεί να επιτύχει καλά αποτελέσματα μόνο αφήνοντας την εφαρμογή να έχει μεγάλο χρόνο εκτέλεσης. Το 1992, οι Kriaer και Yellen στην εργασία τους περιέγραψαν έναν ευριστικό αλγόριθμο μέσω του χρωματισμού γραφήματος για να προσεγγίσουν τη λύση στο πρόβλημα του εβδομαδιαίου προγράμματος των πανεπιστημίων. Ο αλγόριθμος χρησιμοποιούσε έναν γράφο με βάρη για την μοντελοποιήση του προβλήματος με στόχο να βρει τουλάχιστον το κόστος κ-χρωμάτων του γράφου (όπου κ ο αριθμός των διαθέσιμων χρονικών περιόδων) για να ελαχιστοποιήσει τις συγκρούσεις. Το 1994, οι Burke , Elliman και Weare εισήγαν ένα πρόγραμμα για τα πανεπιστήμια βασισμένο στο χρωματισμό γραφήματος και στον χειρισμό περιορισμών. Ο χρωματισμός γραφήματος και ο ευριστικός αλγόριθμος για την κατανομή των αιθουσών παρουσιάστηκαν για το πως ο συνδυασμός των δύο θα μπορούσε να αποτελέσει την βάση και κάθε πρόγραμμα πανεπιστημίου αλλά κυρίως αυτό της εξεταστικής περιόδου. Το 2007, για το πρόβλημα του εβδομαδιαίου προγράμματος μια εναλλακτική μέθοδος χρωματισμού γραφήματος παρουσιάστηκε όπου ενσωμάτωνε την ανάθεση των αιθουσών κατά την διαδικασία του χρωματισμού. Το 2008, αναπτύχθηκε η βιβλιοθήκη για τον χρωματισμό γραφήματος Koala (Tomasz Dobrowolski) όπου είχε εφαρμογή σε πολλά πρακτικά προβλήματα. Το 2009, οι αλγόριθμοι αυτόματων συστημάτων προτάθηκαν για να λύσουν το πρόβλημα της εύρεσης του ελάχιστου στον χρωματισμό των γραφημάτων. Τέλος το 2013 οι Akbulut και G. Yilmaz (Akhan Akbulut) πρότειναν ένα νέο σύστημα για τον προγραμματισμό της εξεταστικής περιόδου των πανεπιστημίων χρησιμοποιώντας τον χρωματισμό γράφου βασισμένο στην τεχνολογία RFID.

1.4 Ακέραιος και Γραμμικός προγραμματισμός

Ο ακέραιος προγραμματισμός (Prabodanie, 2016) έχει να αντιμετωπίσει προβλήτα στα οποία μερικές ή όλες οι μεταβλητές του είναι ακέραιοι αριθμοί. Ένα γραμμικό πρόγραμμα είναι ένα μαθηματικό μοντέλο το οποίο έχει σχεδιαστεί για να βρίσκει ένα πλήθος μη αρνητικών αριθμών ή να ελαχιστοποιεί ή να μεγιστοποιεί τις τιμές των μεταβλητών σε μια γραμμική εξίσωση ή σε μια αντικειμενική συνάρτηση καθώς ικανοποιεί ένα σύστημα γραμμικών περιορισμών. Το πρόβλημα του εβδομαδιαίου προγράμματος ενός πανεπιστημίου το καθιστά έναν τέλειο υποψήφιο διότι όλα τα στοιχεία του που είναι προς εξέταση δεν γίνεται να έχουν δεκαδικές τιμές π.χ. 143.5 φοιτητές, ή 8.34 τμήματα μαθημάτων ή 2.8 καθηγητές. Οι μεταβλητές στο παραπάνω πρόβλημα είναι ακέραιοι αριθμοί.

1.5 Monte Carlo simulated annealing προσομοιωμένη ανόπτηση

1.5.1 Monte Carlo

Το όνομα Μόντε Κάρλο (Coston P. J., 1992) δόθηκε πρώτη φορά σε ένα σύνολο μαθηματικών μεθόδων από μια ομάδα επιστημόνων που εργαζόντουσαν πάνω σε πυρηνικά όπλα στο Los Alamos. Το όνομα ταίριαξε επειδή το πήρε από την γνωστή πόλη του Μονακό όπου είναι γνωστή για καζίνο της και γενικότερα για τα παιχνίδια τύχης. Η ουσία αυτών των μεθόδων είναι ότι περιέχουν το παράγοντα της τύχης. Αυτά τα παιχνίδια μπορούν να μελετηθούν και να δώσουν απαντήσεις σε πολλά προβλήματα.

Οι πρώτες επίσημες αναφορές είναι το 1777 από τον Comte de Buffon, που δημιούργησε εξισώσεις πιθανοτήτων βασισμένες σε πειράματα που έκανε στην ρήξη κερμάτων. Με αυτόν τον τρόπο μπορούσε να προβλέψει πως αυτά τα κέρματα θα έρθουν.

Πολλοί άλλοι συνέφεραν στην ανάπτυξη μαθηματικών μεθόδων που μπορούσαν να θεωρηθούν μέθοδοι Μόντε Κάρλο. Το Μόντε Κάρλο είναι μια μέθοδος στατιστικής φύσης επομένως υπόκειται στους νόμους της τύχης. Αρκετοί επιστήμονες πιστεύουν ότι οι μέθοδοι Μόντε Κάρλο δεν είναι στιβαρή διότι είναι καλή μόνο για εκτιμήσεις που αφορούν αριθμητικές ποσότητες. Τις θεωρούν ως μια τσάντα με διάφορες συσκευές για πολλές χρήσεις. Μερικά από τα πεδία όπου έχουν εφαρμογή είναι:

- Επιχειρησιακή έρευνα
- Μεταφορά ραδιενέργειας
- Οικονομικά μοντέλα
- Γεωλογικά μοντέλα
- Επεξεργασία δεδομένων
- Προσομοιώσεις των υγρών και των στερεών
- Φυσικά μοντέλα
- Χημικά μοντέλα

Το 1966, οι Macon και Walker εφάρμοσαν την ιδέα Μόντε Κάρλο στον αλγόριθμο του χρονοπρογραμματισμού. Το κύριο πλεονέκτημα της μεθόδου Μόντε Κάρλο ήταν ότι μπορούσε αποτελεσματικά να υπολογιστεί σε σύγκριση με άλλες ντετερμινιστικές μεθόδους που χρησιμοποιούσαν για να λύσουν πολυδιάστατα προβλήματα. Η μέθοδος Μόντε Κάρλο στο πρόβλημα του χρονοπρογραμματισμού στοχεύει μέσα από τυχαίες επιλογές να μειώσει τον αριθμό των φοιτητών που θα μείνουν εκτός από το μάθημα που επιθυμούσαν. Αυτή η τυχαιότητα προσφέρει μια δίκαιη προσέγγιση σε σχέση με το «ο πρώτος που έρθει θα είναι και αυτός που θα το κατοχυρώσει».

1.5.2 Simulated annealing

Ένας αλγόριθμος προσομοιωμένης ανόπτησης (Abramson, 1991) (Nurlida Basir, 2013)είναι μία μέθοδος Μόντε Κάρλο που χρησιμοποιείται να για να βρεθεί λύση σε ένα πρόβλημα βελτιστοποίησης. Ανόπτηση σημαίνει «να κάνεις κάτι λιγότερο εύθραυστο προκειμένου στη συνέχεια υποβαλλόμενο σε ψύξη να βελτιωθεί η ευκαμψία του». Προσομοιωμένη ανόπτηση είναι δηλαδή ένας αλγόριθμος που προσπαθεί να προσομοιώσει φυσικές ιδιότητες σε κάτι που δέχεται ανόπτηση.

Στην κατασκευή ενός προγράμματος για ένα πανεπιστήμιο η προσομοιωμένη ανόπτηση προγραμματίζει ένα στοιχείο σε μια χρονική περίοδο. Ως στοιχείο μπορεί να οριστεί ένα σύνολο φοιτητών, καθηγητών, μαθημάτων και αιθουσών. Ως χρονική περίοδος ορίζεται μια μέρα της εβδομάδας που έχει συγκεκριμένη ώρα έναρξης και λήξης. Πολυάριθμα στοιχεία μπορούν να προγραμματιστούν για μια συγκεκριμένη χρονική περίοδο. Ένα στοιχείο μπορεί να χωρέσει στο πρόγραμμα από μια έως πέντε φορές στο

εβδομαδιαίο πρόγραμμα. Τα στοιχεία που βρίσκονται παραπάνω από μία φορές στο πρόγραμμα πρέπει να είναι σε διαφορετική χρονική περίοδο αλλιώς θα υπάρχει σύγκρουση. Ένας φοιτητής ή ένας καθηγητής δεν μπορεί να βρίσκεται σε δύο μέρη ταυτόχρονα. Το αποτέλεσμα από αυτό τον αλγόριθμο προσομοιωμένης ανόπτησης είναι ή ένα έγκυρο εβδομαδιαίο πρόγραμμα χωρίς συγκρούσεις ή ένα που να είναι όσο πιο κοντά γίνεται στο έγκυρο με όσο το δυνατόν λιγότερες συγκρούσεις.

Σε κάθε στοιχείο αντιστοιχεί ένα κόστος, σύμφωνα με το πόσο σημαντικό είναι αυτό το στοιχείο. Γνωρίζοντας το κόστος κάθε στοιχείου, ο αλγόριθμος οδηγείται σε μία λύση όπου κάποιες συγκρούσεις είναι πιο σημαντικές από τις άλλες. Το συνολικό κόστος του συστήματος είναι το άθροισμα όλων των στοιχείων που συγκρούστηκαν. Στόχος είναι ένα σύστημα με χαμηλό κόστος. Έγκυρο θεωρείται ένα πρόγραμμα με μηδέν κόστος.

Όταν τα άτομα του συστήματος είναι «ζεστά» είναι ελεύθερα να κινηθούν τυχαία στο χώρο. Ωστόσο , όταν το σύστημα αρχίζει να «κρυώνει», τα εσωτερικά όρια αναγκάζουν τα άτομα να έρθουν κοντά. Όταν τέλος το σύστημα έχει «κρυώσει», καμία κίνηση δεν επιτρέπεται και η παραμετροποίηση των ατόμων «παγώνει». Αν το σύστημα «κρυώσει» απότομα τότε η πιθανότητα να αποκτήσει μια σταθερή θέση είναι μικρότερη σε σχέση με την σταδιακή μείωση της θερμοκρασίας. Τα άτομα δέχονται μια νέα παραμετροποίηση κάθε φορά που η θερμοκρασία μειώνεται. Ωστόσο, αν η ενέργεια του συστήματος παραμένει υψηλή αλλά η πιθανότητα να αυξηθεί και άλλο είναι μικρή τότε μια νέα παραμετροποίηση είναι αποδεκτή.

Στην περίπτωση του χρονοπρογραμματισμού του προγράμματος ενός πανεπιστημίου, τα στοιχεία αντιστοιχούν στα άτομα. Κατά την αρχικοποίηση του προγράμματος, τα στοιχεία τοποθετούνται με τυχαία σειρά σε χρονικές περιόδους και υπολογίζεται το κόστος του προγράμματος. Σε κάθε επανάληψη μια χρονική περίοδος (P1) επιλέγεται τυχαία και ένα στοιχείο επιλέγεται και αυτό τυχαία για αυτή την περίοδο. Μετά μια άλλη περίοδος (P2) επιλέγεται τυχαία. Αν η μετακίνηση του στοιχείου από την περίοδο (P1) στην (P2) μειώνει το κόστος του προγράμματος, τότε το στοιχείο μετακινείται. Αν το κόστος του συστήματος αρχικά θα αυξηθεί με την μετακίνηση αλλά η πιθανότητα να μην αυξηθεί και άλλο είναι μικρή τότε το στοιχείο πάλι μετακινείται ειδάλλως το αφήνει όπως είναι. Όπως και στον φυσικό κόσμο όπου μια πιο αργή ανόπτηση επιτυγχάνει πιο σταθερή κατάσταση, έτσι και η ανόπτηση στο προγραμματισμό του εβδομαδιαίου προγράμματος πετυχαίνει ένα μικρότερο σε κόστος πρόγραμμα. Μια πιθανή εξήγηση είναι ότι με την σταδιακή μείωση της «θερμοκρασίας» ένας αποτελεσματικός άπληστος αλγόριθμος

επιτρέπει την μείωση του κόστους του προγράμματος και έτσι είναι πιο πιθανό να μην παράγονται πολλές και διαφορετικές λύσεις στο τέλος κάθε εκτέλεσης.

Το κύριο πλεονέκτημα της προσομοιωμένης ανόπτησης σε σχέση με άλλους αλγορίθμους (αναζήτηση αναρρίχησης) είναι ότι είναι λιγότερο πιθανό να πέσει σε τοπικό minima, και αυτό διότι επιτρέπεται τόσο η αύξηση του κόστους όσο και η μείωση.

Εφόσον η προσομοιωμένη ανόπτηση βασίζεται τον παράγοντα της τύχης δεν μπορεί να εξασφαλιστεί ότι θα βρεθεί πραγματικά το μικρότερο κόστος για το πρόγραμμα ή ότι δύο διαφορετικές εκτελέσεις θα δώσουν το ίδιο αποτέλεσμα. Συνήθως, πρέπει να εκτελεστεί αρκετές φορές ο αλγόριθμος μέχρι να φτάσει σε αποδεκτή λύση. Αποδεκτή είναι η λύση όπου το κόστος είναι μηδέν (δεν υπάρχουν συγκρούσεις) ή σχεδόν μηδέν (υπάρχουν λίγες).

1.6 Tabu search

Η μέθοδος Tabu search (Izundu Kingsley, 2018) (Toth, 2015) (M. A. Saleh Elmohamed, 1998) είναι μια γενική ευριστική διαδικασία που οδηγεί στην απόκτηση μιας καλής λύσης στο χώρο ενός πολύπλοκου προβλήματος. Η μέθοδος χρησιμοποιείται και σε άλλες ευριστικές διαδικασίες. Ένα από τα κύρια συνθετικά της αναζήτησης Tabu είναι η ελαστική (ευπροσάρμοστή) μνήμη, όπου παίζει ένα βασικό ρόλο στη διαδικασία της αναζήτησης. Οι ρίζες της ξεκινάνε στα τέλη της δεκαετίας του 1960 και στις αρχές του 1970 και προτάθηκε στην σημερινή του μορφή μερικά χρόνια αργότερα το 1986 στην εργασία (GLOVER, 1986). Πλέον έχει καθιερωθεί ως η βέλτιστη προοπτική για την αναζήτηση και εξαπλώνεται σε πολλά πεδία, έχει βοηθήσει να βρεθεί λύση σε προβλήματα όπως

- ο προγραμματισμός πόρων
- οι τηλεπικοινωνίες
- η σχεδίαση VLSI
- η οικονομική ανάλυση
- ο χρονοπρογραμματισμός
- η σχεδίαση χώρου
- η κατανομή ενέργειας
- η μοριακή μηχανική
- η διαχείριση αποβλήτων
- η βίο-ιατρική ανάλυση

και πολλά άλλα.

Το κίνητρο για την ανάπτυξη της αναζήτηση Tabu προήλθε κυρίως από την παρατήρηση της ανθρώπινης συμπεριφοράς που εμφανίζεται κατά την διάρκεια μιας διαδικασίας όπου ένα τυχαίο στοιχείο οδηγεί σε αντιφατικές συμπεριφορές που όμως οδηγούν σε παρόμοιες καταστάσεις. Η μέθοδος Tabu search λειτουργεί με αυτόν τον τρόπο με την διαφορά ότι ένα νέο αντικείμενο δεν επιλέγεται τυχαία, αντ' αυτού η μέθοδος εκδηλώνεται σύμφωνα με την εικασία ότι δεν υπάρχει λόγος να αποδεχτεί μια νέα λύση εκτός και αν υπάρχει μονοπάτι το οποίο δεν έχει ήδη εξερευνηθεί. Με αυτό τον τρόπο εξασφαλίζεται ότι νέες περιοχές στο χώρο λύσης του προβλήματος θα εξερευνηθούν με στόχο να αποφευχθούν τα τοπικά ελάχιστα και να οδηγηθούν στην επιθυμητή λύση.

Η αναζήτηση Tabu ξεκινάει αναζητώντας το τοπικό ελάχιστο. Για την αποφυγή να ξαναγυρίζει σε προηγούμενα βήματα μία μέθοδος καταγράφει τις πρόσφατες κινήσεις σε ή περισσότερες Tabu λίστες. Ο αρχικός στόχος των λιστών δεν ήταν να αποφεύγουν τις προηγούμενες κινήσεις ώστε να μην επαναληφθούν αλλά να εξασφαλίσουν ότι δεν είναι ανεστραμμένες. Οι λίστες Tabu είναι ο πυρήνας της μνήμης στην αναζήτηση Tabu. Ο ρόλος της μνήμης μπορεί να αλλάξει κατά την εκτέλεση του αλγόριθμου. Κατά την αρχικοποίηση ο στόχος είναι να κάνει μια πλατιά αναζήτηση στο χώρο λύσης του προβλήματος γνωστή και ως «diversification» επεκτείνοντας την αναζήτηση πάνω στην κοντινότερη γειτονική λύση αλλά καθώς υποψήφιες τοποθεσίες έρχονται στην επιφάνεια η αναζήτηση επικεντρώνεται στο να παραχθεί μια τοπική βέλτιστη λύση γνωστή και ως «intensification». Σε πολλές περιπτώσεις οι διαφορές ανάμεσα στις διάφορες υλοποιήσεις της μεθόδου Tabu έχουν να κάνουν με το μέγεθος , την μεταβλητότητα και την προσαρμοστικότητα της μνήμης Tabu σύμφωνα με το πεδίο του προβλήματος.

Ένας γενικός αλγόριθμος αναζήτησης Tabu είναι

```
Initialisation
        S: = initial solution in X
        nbiter:= 0              {current iteration}
        besiter := 0            {iteration when the best solution
                                has been found}
        bestsol:= s             {best solution}
        T: = 0
        Initialise the aspiration function A
        While (f(s) > f*) and (nbiter-bestiter<nbmax) do
                nbiter:= nbiter +1
                generate a set V* of solutions s in n(s)
                which are either not tabu or such that
                A(f(s))>=f(s)
                choose a solution s* minimising f over V*
                update the aspiration function A and the
                tabu list T
                if f(s*) <f(bestsol) then
                        bestsol:=s*
                        bestiter:=nbiter
                        s:=s*
```

Εικόνα 1, (Izundu Kingsley, 2018)

Για κάθε λύση (s) η μέθοδος απαιτεί τον προσδιορισμό του γείτονα V(s), που να είναι προσβάσιμος σε ένα βήμα από το (s). Το βασικό βήμα είναι η κίνηση από την τρέχουσα λύση (s) στην καλύτερη λύση s*V(s). Μια λίστα Tabu (T) χρησιμοποιείται για την αποφυγή κύκλων καθώς αποθηκεύει την περιγραφή της τελευταίας λύσης ή κίνησης ενώ βρίσκει (s*) στο V(s).

Σε ένα πρόβλημα βελτιστοποίησης η αναζήτηση χώρου (S) ελαχιστοποιείται από την αντικειμενική συνάρτηση (f). Μια συνάρτηση N που εξαρτάται από τη δομή του συγκεκριμένου προβλήματος συσχετίζεται σε κάθε πιθανή λύση (s) που ανήκει στο (S) και στο γείτονα του. [N(s) υποσύνολο του S] και κάθε λύση του s ανήκει στο N(s) και λέγεται γείτονας του S.

Η αναζήτηση Tabu βασίστηκε σε επιλεγμένες ιδέες βελτιστοποίησης και της τεχνητής νοημοσύνης και είναι από τις τάσεις όπου θα απασχολήσουν πολύ τις επόμενες δεκαετίες και θα έχει πρακτική εφαρμογή.

1.7 Γενετικοί αλγόριθμοι

Οι γενετικοί αλγόριθμοι (Bambrick, 1997) είναι μετά-ευριστικές μέθοδοι που χρησιμοποιούνται για να βρουν λύση σε υπολογιστικά προβλήματα τα οποία έχουν μεγάλους χώρους αναζήτησης για πιθανές λύσεις. Πολύ συχνά βασίζονται σε προσαρμοστικά συστήματα για να έχουν απόδοση σε περιβάλλοντα τα οποία αλλάζουν.

Ένας γενετικός αλγόριθμος είναι μια πολύ ισχυρή τεχνική επίλυσης προβλημάτων. Ανήκει στην κατηγορία των εξελικτικών αλγορίθμων που είναι υποσύνολο της εξελικτικής υπολογιστικής στην τεχνητή νοημοσύνη. Αναπτύχθηκε το 1960 από τον καθηγητή John Holland του πανεπιστημίου του Μίσιγκαν και το βιβλίο του *Adaptation in Natural And Artificial Systems* ενέπνευσε την έρευνα για τους γενετικούς αλγορίθμους στην δεκαετία του 1970. Αυτή η τεχνική είναι εμπνευσμένη από την δαρβινική θεωρία της εξέλιξης της φύσης, όπου κάθε οργανισμός στον κόσμο πολλαπλασιάζεται με γεωμετρική αναλογία οδηγώντας τον σε πάλη για την ύπαρξη του εξαιτίας του περιορισμένου χώρου και τροφής. Σε αυτή την πάλη ο πιο ικανός θα επιβιώσει. Ο πιο ικανός είναι εκείνος που με ευνοϊκές παραλλαγές θα οδηγήσει στην εξέλιξη του είδους. Οι πιθανότητες επιβίωσης των οργανισμών με επιζήμιες παραλλαγές είναι ελάχιστες, γι’ αυτό και η εξέλιξη είναι μια διαδικασία φυσικής επιλογής.

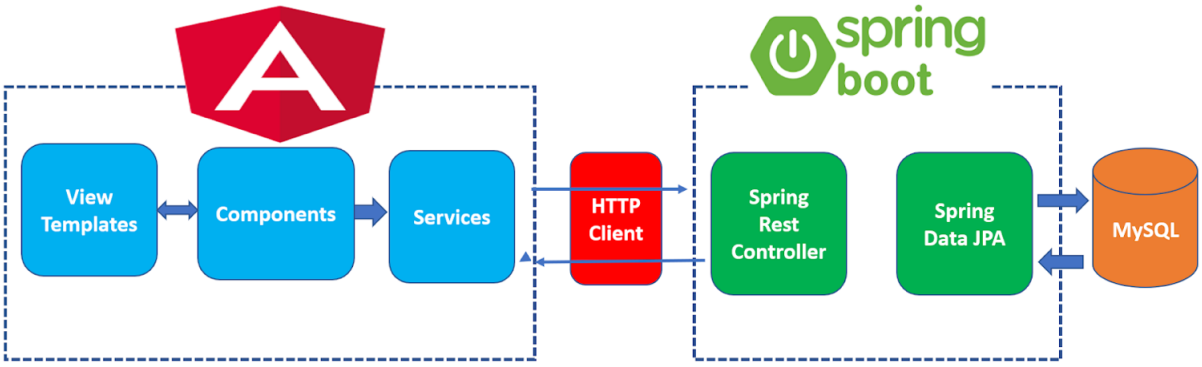
Κάθε κύτταρο ενός οργανισμού έχει ένα συγκεκριμένο αριθμό από γενετικές δομές που ονομάζεται χρωμόσωμα

Οι γενετικοί αλγόριθμοι μιμούνται την βιολογική συμπεριφορά των χρωμοσωμάτων και γονιδίων κάθε χρωμόσωμα να αναπαριστά έναν μεμονωμένο οργανισμό και κάθε γονιδίωμα ένα συστατικό μια λύσης που χρησιμοποιείται με έναν γενετικό αλγόριθμο, επιπλέον, ένα χρωμόσωμα αναπαριστά μια δομή δεδομένων. Όπως στη φύση, ένα χρωμόσωμα παράγεται από την μίξη γενετικού υλικού.

2.Περιγραφή εφαρμογής

Η ανάπτυξη μιας ολοκληρωμένης εφαρμογής γίνεται συνήθως από μια ομάδα ανθρώπων οι οποίοι χωρίζουν την εφαρμογή σε τρεις ομάδες.

- Τον σχεδιασμό της βάσης δεδομένων
- Την μεριά του Server (back end)
- Την μεριά του Client (front end)



Full stack App

Με τον όρο βάση δεδομένων εννοούμε ένα σύνολο δεδομένων που χαρακτηρίζονται από κάποια λογική οργάνωση και ομαδοποίηση έτσι ώστε να είναι εύκολη και αποτελεσματική η διαχείριση τους. Οι βασικές εντολές είναι τέσσερις: Create , Read, Update και Delete - CRUD. Μερικές από τις πιο γνωστές βάσεις δεδομένων είναι η Postgres, η Oracle και MongoDB η οποία είναι NoSQL.

Ο όρος Back end (Server side) αναφέρεται στο κομμάτι εκείνο της εφαρμογής που είναι υπεύθυνο για την λειτουργία της και κρύβει όλη την επιχειρησιακή λογική. Είναι επίσης υπεύθυνο για την διαχείριση της βάσης δεδομένων μέσω ερωτημάτων. Μερικές από τις πιο ευρέως διαδεδομένες γλώσσες προγραμματισμού για back end είναι η PHP, Java, Ruby με τα αντίστοιχα frameworks Lavarel, Spring, Ruby on Rails.

Τέλος το κομμάτι εκείνο που είναι υπεύθυνο για την παρουσίαση της εφαρμογής είναι το front end (Client side) στο οποίο ο χρήστης αλληλοεπιδρά με την εφαρμογή. Μερικές από τις πιο γνωστές τεχνολογίες για την ανάπτυξη στην μεριά του Client είναι η HTML, το CSS και η γλώσσα προγραμματισμού JavaScript η οποία τα τελευταία χρόνια χρησιμοποιείται εξίσου και στην μεριά του Server. Τα frameworks που χρησιμοποιούνται περισσότερο αυτή τη στιγμή είναι το Angular το οποίο αναλύεται παρακάτω, το React.js, το Bootstrap και το jQuery.

2.1PostgreSQL

Η PostreSQL είναι μια ισχυρή, ανοιχτού λογισμικού αντικείμενο-σχεσιακή βάση δεδομένων που επεκτείνει την γλώσσα SQL και σε συνδυασμό με πολλά πρόσθετα χαρακτηριστικά μπορεί να αποθηκεύει τα πιο πολύπλοκα σχήματα δεδομένων. Οι ρίζες της PostreSQL κρατάνε από το 1986 όταν και εντάχθηκε στο πρότζεκτ POSTGRES του πανεπιστημίου της Καλιφόρνια Μπέρκλεϊ και έχει περισσότερα από 30 χρόνια συνεχής ανάπτυξης. Η αρχιτεκτονική της σε συνδυασμό με την αξιοπιστία της, την διασφάλιση των δεδομένων , τα δυνατά χαρακτηριστικά της, την επεκτασιμότητα της και την αφοσίωση στην κοινότητα του ανοιχτού λογισμικού την έκανα πολύ γρήγορα γνωστή εκτοξεύοντας την φήμη της. Η PostreSQL μπορεί να εγκατασταθεί σε όλα τα λειτουργικά συστήματα, είναι ACID-compliant από το 2001 και έχει κάποια από τα καλύτερα πρόσθετα όπως είναι το PostGIS που αφορά γεωχωρικά δεδομένα.

2.1.1 Γιατί PostgreSQL?

Η PostgreSQL έρχεται με πλειάδα χαρακτηριστικών που στόχο έχουν να βοηθήσουν τον προγραμματιστή στην ανάπτυξη των εφαρμογών του, τους διαχειριστές να εξασφαλίσουν την ακεραιότητα των δεδομένων τους, να φτιάξουν περιβάλλοντα με ανοχή σφαλμάτων και να προσφέρει ευκολία στην διαχείριση των δεδομένων όσο μεγάλος ή μικρός είναι ο όγκος τους. Επιπλέον, επειδή είναι ανοιχτού κώδικα είναι αρκετά επεκτάσιμη. Αυτό πρακτικά σημαίνει ότι ένας προγραμματιστής μπορεί να καθορίσει τους δικούς του τύπους δεδομένων, να φτιάξει τις δικές του συναρτήσεις ακόμα και να γράφει κώδικα με διαφορετικές γλώσσες.

Μερικά από τα χαρακτηριστικά της PostgreSQL:

Τύπο δεδομένων:

- Πρωταρχικοί : Ακέραιοι, Αριθμοί, Αλφαριθμητικά, Boolean
- Δομημένοι: Date/Time , Array, Range, UUID
- Έγγραφα: JSON/JSONB, XML, Key-Value(Hstore)
- Γεωμετρία: Σημείο, Γραμμή, Κύκλος, Πολύγωνα.

Ακεραιότητα:

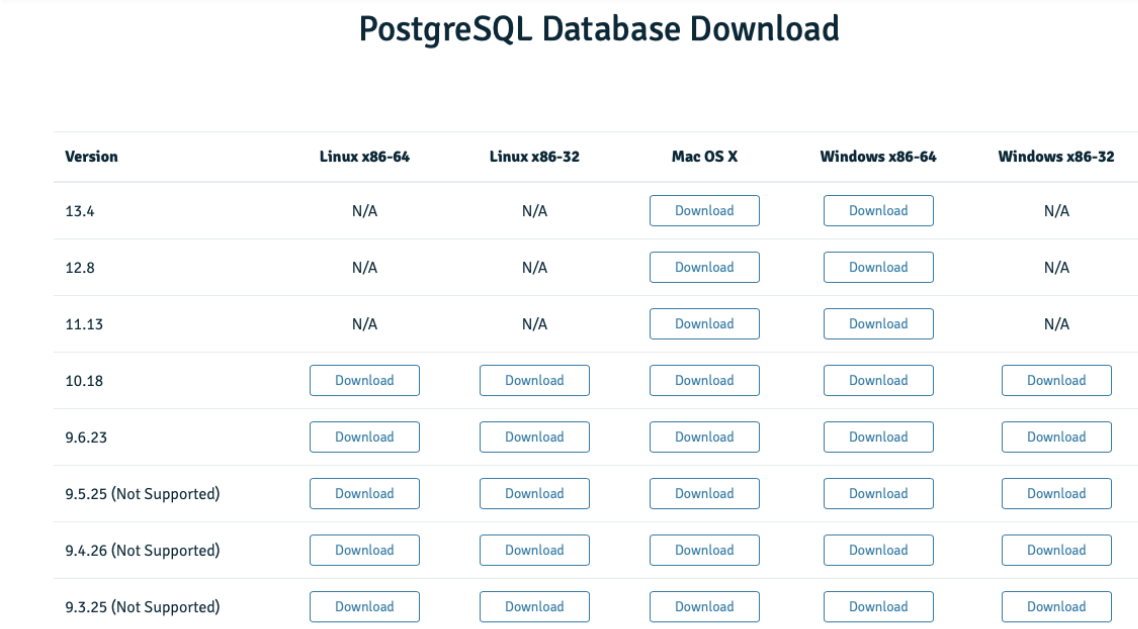
- Unique, Not null
- Πρωτεύον κλειδί, ξένο κλειδί

Ταυτοχρονισμός / Απόδοση

- Indexing: B-Tree, Multicolumn, Expressions
- Advanced Indexing: GiST, SP-Gist, KNN Gist, GIN, BRIN, Covering indexes, Bloom filters
- Multi-version concurrency Control (MVCC)

2.1.2 Εγκατάσταση PostgreSQL

Η PostgreSQL (postgresql, 2021) είναι διαθέσιμη για όλα τα περιβάλλοντα (Windows, Linux, Mac OS X) και η εγκατάσταση της είναι αρκετά εύκολη. Το πρώτο που πρέπει να κάνει ο προγραμματιστής είναι να το επίσημο site της PostgreSQL και από ένα αναλυτικό μενού να κατεβάσει το αρχείο που του ταιριάζει.



Εικόνα 2,επίσημη σελίδα της PostgreSQL

Από την έκδοση 8.0 και μετά, η PostgreSQL προσφέρει έναν εγκαταστάτη (installer) και η όλη διαδικασία είναι εξαιρετικά απλή. Ο προγραμματιστής θα πρέπει να εκτελέσει την εφαρμογή με δικαιώματα διαχειριστή. Τα μόνα σημεία που θα πρέπει να προσέξει είναι όταν του ζητηθεί ο κωδικός, ο οποίος είναι αυτός που θα έχει ο superuser και όταν θα του ζητηθεί να επιλέξει την πόρτα στην οποία θα ακούει η βάση δεδομένων, προεπιλογή είναι 5432. Τέλος για την επαλήθευση της εγκατάστασης θα πρέπει να ανοίξει το τερματικό και πληκτρολογώντας psql να συνδεθεί στο περιβάλλον της PostgreSQL.

2.2 Version Control System

Το σύστημα παρακολούθησης εκδόσεων Version Control System (VCS) (Rochkind, 1975) είναι ένα λογισμικό που επιτρέπει στους προγραμματιστές να δουλεύουν μαζί ταυτόχρονα , να μην μπορεί να υπερκαλύψει ο ένας τον κώδικα του άλλου και να διατηρούν ένα πλήρες ιστορικό της δουλείας τους.

Τα VCS χωρίζονται σε δύο κατηγορίες:

- Κεντροποιημένο σύστημα παρακολούθησης – Centralized version control system (CVCS)

- Αποκεντροποιημένο / Κατανεμημένο σύστημα παρακολούθησης – Decentralized version control system (DVCS), όπως είναι και το Git.

Ένα κεντροποιημένο σύστημα εκδόσεων χρησιμοποιεί ένα κεντρικό εξυπηρετητή (server) για να αποθηκεύει τα αρχεία και να επιτρέπει στους προγραμματιστές να συνεργάζονται. Το βασικό μειονέκτημα είναι ότι σε περίπτωση που ο εξυπηρετητής είναι εκτός λειτουργίας τότε κανένας δεν μπορεί να έχει πρόσβαση. Ακόμα χειρότερο είναι το σενάριο όπου ο σκληρός δίσκος του εξυπηρετητή αλλοιωθεί και δεν υπάρχει αντίγραφο ασφαλείας τότε όλο το ιστορικό του πρότζεκτ χάνεται.

2.2.1 Git

Το Git (Chacon, 2021) είναι ένα κατανεμημένο σύστημα παρακολούθησης και διόρθωσης του κώδικα με έμφαση στην ταχύτητα. Το Git αρχικά σχεδιάστηκε και αναπτύχθηκε από τον Linus Torvalds για την ανάπτυξη του πυρήνα του Linux. και είναι ανοιχτής άδειας υπό την αιγίδα του GNU General Public License έκδοση 2.

Τα πλεονεκτήματα του Git είναι :

- Δωρεάν και ανοιχτού κώδικα: Το Git διανέμεται υπό την αιγίδα του GNU General Public License και είναι διαθέσιμο από οποιονδήποτε έχει πρόσβαση στο διαδίκτυο. Μπορεί να χρησιμοποιηθεί τόσο για εμπορικά όσο και σε ερασιτεχνικά πρότζεκτ.

- Γρήγορο και μικρό: Καθώς οι περισσότερες διεργασίες γίνονται τοπικά, δίνει ένα μεγάλο πλεονέκτημα στο θέμα της ταχύτητας. Το Git δεν βασίζεται σε ένα κεντρικό εξυπηρετητή και για αυτό το λόγο δεν χρειάζεται η αλληλεπίδραση για κάθε ενέργεια που γίνεται. Ο πηγαίος κώδικας του Git είναι γραμμένος σε C και έτσι αποφεύγεται η έμμεση κατανάλωση πόρων που προκύπτει από γλώσσες υψηλού επιπέδου.

- Έμμεσο αντίγραφο: Οι περιπτώσεις που έχουν χαθεί δεδομένα είναι σπάνιες καθώς υπάρχουν πολλαπλά αντίγραφα. Η πληροφορία είναι διαθέσιμη σε κάθε τερματικό που έχει κατεβάσει το πρότζεκτ.

- Ασφάλεια: Το Git χρησιμοποιεί μια κοινή κρυπτογράφηση που ονομάζεται συνάρτηση ασφαλή κατακερματισμού (Secure Hash Function – SHA1) και με βάση αυτή ταυτοποιεί τα αντικείμενα στην βάση του.

- Δεν χρειάζεται ακριβό υλικό (hardware): Στην περίπτωση του κεντροποιημένου συστήματος παρακολούθησης ο εξυπηρετητής πρέπει να είναι σε θέση να εξυπηρετεί όλα τα αιτήματα που θα έχει μια ομάδα. Στην περίπτωση του κατανεμημένου συστήματος παρακολούθησης ο προγραμματιστής δεν αλληλοεπιδρά με τον εξυπηρετητή εκτός των περιπτώσεων του push και του pull.

- Εύκολη διακλάδωση: Η διακλάδωση με το Git είναι πολύ απλή και δεν χρειάζεται παρά μόνο λίγα δευτερόλεπτα σε αντίθεση με ένα κεντροποιημένο σύστημα όπου θα πρέπει να γίνει ένα αντίγραφο και να ανέβει στον εξυπηρετητή.

2.2.2 Βασικές έννοιες και ροή εργασιών

2.2.2.1 Τοπικό αποθετήριο Local Repository

Κάθε κετρνοποιήμενο σύστημα παρακολούθησης παρέχει ένα ιδιωτικό χώρο εργασίας ως σταθμό εργασίας. Οι προγραμματιστές κάνουν αλλαγές σε αυτό τον ιδιωτικό χώρο και μετά τις δεσμεύουν, οι αλλαγές αυτές γίνονται κομμάτι του αποθετηρίου. Το Git το πήγε ένα βήμα πιο κάτω παρέχοντας ένα ιδιωτικό αντίγραφο όλου του αποθετηρίου. Οι χρήστες μπορούν να κάνουν πολλές διεργασίες τοπικά όπως να προσθέσουν ένα αρχείο, να το μετακινήσουν, να το μετονομάσουν, να το διαγράψουν, να δεσμεύσουν τις αλλαγές τους και όλα αυτά χωρίς να επηρεάσουν το αποθετήριο.

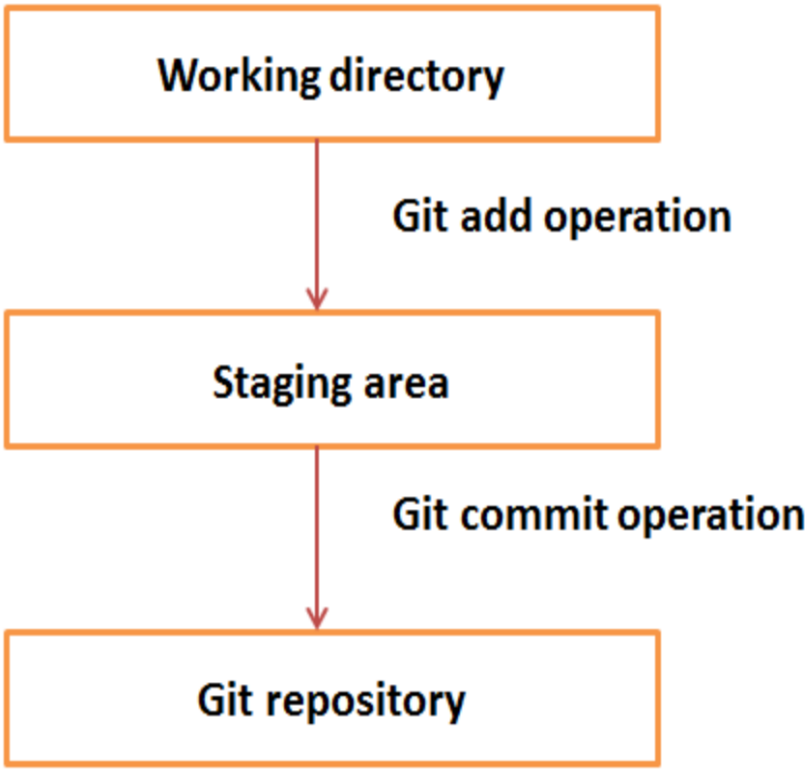
2.2.2.2 Αρχείο εργασίας Working Directory

Το αρχείο εργασίας είναι εκείνος ο χώρος όπου τα αρχεία ελέγχονται και εξετάζονται. Στο κεντροποιημένο σύστημα παρακολούθησης, οι προγραμματιστές κάνουν αλλαγές και τις δεσμεύουν κατευθείαν στο αποθετήριο όπως είδαμε και παραπάνω. Το Git ακολουθεί μια διαφορετική στρατηγική. Το Git δεν παρακολουθεί κάθε τροποποιημένο αρχείο. Κάθε φορά που διενεργείται μια δέσμευση από τον προγραμματιστή, το Git κοιτάει

για τα αρχεία που είναι στον χώρο του ανεβάσματος, μόνο αυτά τα αρχεία θεωρούνται πως όντως έχουν αλλαγές και είναι αυτά που πρέπει να ανέβουν.

Η βασική ροή εργασιών στο Git:

1. Τροποποιείται ένα αρχείο στο αρχείο εργασίας
2. Προστίθεται στο χώρο ανεβάσματος
3. Διενέργεια δέσμευσης όπου μετακινείται το αρχεία στο χώρο ανεβάσματος. Μετά την διενέργεια ώθησης, οι αλλαγές γίνονται μόνιμες στο αποθετήριο του Git



Εικόνα 3, στάδια επεξεργασίας

2.2.2.3 Blobs

Το Blob είναι ακρώνυμο για το Δυαδικό μεγάλο αντικείμενο Binary Large Object . Κάθε έκδοση αρχείου αναπαρίσταται ως ένα blob. Ένα blob κρατάει τα δεδομένα του αρχείου αλλά δεν περιέχει κανένα μετά-δεδομένο για το αρχείο. Είναι ένα δυαδικό αρχείο και στην βάση του Git έχει ονομαστεί σύμφωνα με μια συνάρτηση κατακερματισμού SHA1.

2.2.2.4 Δέντρα Trees

Τα δέντρα είναι αντικείμενα, και απεικονίζουν φακέλους. Κρατάνε τα blobs καθώς και άλλους υπό φακέλους. Τα δέντρα είναι επίσης δυαδικά αρχεία τα οποία αποθηκεύουν αναφορές για τα blobs.

2.2.2.5 Δέσμευση Commit

Η δέσμευση κρατάει την τωρινή κατάσταση του τοπικού αποθετηρίου. Ένα αντικείμενο δέσμευσης μπορεί να θεωρηθεί ως ένας κόμβος σε μια συνδεδεμένη λίστα. Κάθε δέσμευση έχει ένα δείκτη στην γονική της δέσμευση. Από μια δέσμευση μπορείς να γυρίσεις στην προηγούμενη κοιτώντας απλά τον γονέα της στο ιστορικό των δεσμεύσεων. Αν μια δέσμευση έχεις πολλούς γονείς τότε η συγκεκριμένη δέσμευση έχει δημιουργηθεί από την συγχώνευση δύο κλαδιών.

2.2.2.6 Κλαδιά Branches

Τα κλαδιά χρησιμοποιούνται ώστε να δημιουργηθεί μια νέα γραμμή παραγωγής κώδικα. Εκ των προτέρων το Git έχει ένα master κλαδί. Συνήθως ένα νέο κλαδί δημιουργείται όταν ο προγραμματιστής θέλει να ένα καινούργιο χαρακτηριστικό. Όταν τελειώσει με αυτό το χαρακτηριστικό τότε συγχωνεύει το κλαδί με το master. Κάθε κλαδί έχει και ένα HEAD όπου δείχνει πια είναι η τελευταία δέσμευση που έχει γίνει.

2.2.2.7 Tags

Στα Tags ανατίθεται ένα σημαντικό όνομα με μια συγκεκριμένη έκδοση του αποθετηρίου. Τα tags είναι πολύ κοντά στην λειτουργικότητα των κλαδιών με την διαφορά να είναι ότι τα tags είναι μη τροποποιήσιμα. Αυτό σημαίνει ότι ένα tag είναι ένα κλαδί το οποίο κανένας δεν μπορεί να πειράξει και συνήθως οι προγραμματιστές τα χρησιμοποιούν για να δείξουν μια σταθερή έκδοση του προγράμματος.

2.2.2.8 Clone

Η διεργασία κλωνοποίησης clone είναι αυτή όπου δημιουργεί ένα στιγμιότυπο του αποθετηρίου και δίνει στον προγραμματιστής να μπορεί να κάνει όποια ενέργεια θέλει τοπικά.

2.2.2.9 Pull

Η διεργασία pull αντιγράφει τις αλλαγές από το απομακρυσμένο αποθετήριο στο τοπικό και χρησιμοποιείται για τον συγχρονισμό δύο στιγμιότυπων.

2.2.2.10 Push

Η διεργασία push αντιγράφει τις αλλαγές από το τοπικό στιγμιότυπο του αποθετηρίου και τις «σπρώχνει» στο απομακρυσμένο για να τις κάνει μόνιμες.

2.2.2.11 HEAD

Δουλεύοντας με το Git κάθε φορά μπορεί να εξετάζει μόνο ένα κλαδί (branch) και αυτό είναι που αποκαλείται κεφάλι του κλαδιού (Head branch).

2.2.2.12 Revision

Μέχρι στιγμής αναφέρθηκαν εντολές του Git που λίγο πολύ η χρήση τους είναι καθημερινή. Μια εντολή που η χρήση της δεν είναι τόσο προφανής είναι το revision που αναφέρεται σε ένα ή περισσότερα commits σε κλαδιά που είναι στο τοπικό σου αποθετήριο.

2.3 Ολοκληρωμένα περιβάλλοντα ανάπτυξης

Ένα ολοκληρωμένο περιβάλλον ανάπτυξης είναι ένας συνδυασμός εργαλείων που στόχο έχει να κάνει τη δουλειά του προγραμματιστή ευκολότερη, πιο παραγωγική και με λιγότερα λάθη. Σε σύγκριση με έναν απλό επεξεργαστή κειμένου υπερτερεί σε τέσσερις βασικές κατηγορίες

- Προ εγκατεστημένο μεταγλωττιστή, debugger, παρακολούθηση εκδόσεων (VCS), διάφορα frameworks ανάλογα τον IDE κ.α.
- Υποστηρίζει την πλοήγηση στον κώδικα μεταξύ πολύπλοκων και πολλών κλάσεων , αυτόματη συμπλήρωση ,τροποποίηση (refactoring) και παραγωγή κώδικα.
- Υποστηρίζει το τεστ σε μονάδες κώδικα.
- Παρέχει μια ευρεία γκάμα από πρόσθετα τα οποία μπορούν εύκολα να εγκατασταθούν.

2.3.1 IntelliJ IDEA

IntelliJ IDEA (JetBrains, n.d.) είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης λογισμικού γραμμένο σε Java για την ανάπτυξη λογισμικού. Αναπτύχθηκε από την JetBrains και διατίθεται σε 2 βασικές εκδόσεις την Community και την Ultimate , αμφότερες μπορούν να χρησιμοποιηθούν για εμπορική ανάπτυξη. Εμφανίστηκε πρώτη φορά τον Ιανουάριο του 2001 και ήταν ένα από τα πρώτα ολοκληρωμένα περιβάλλοντα ανάπτυξης Java που υποστήριζε προηγμένη πλοήγηση στον κώδικα καθώς επίσης και ενσωματωμένες λειτουργίες code refactoring.

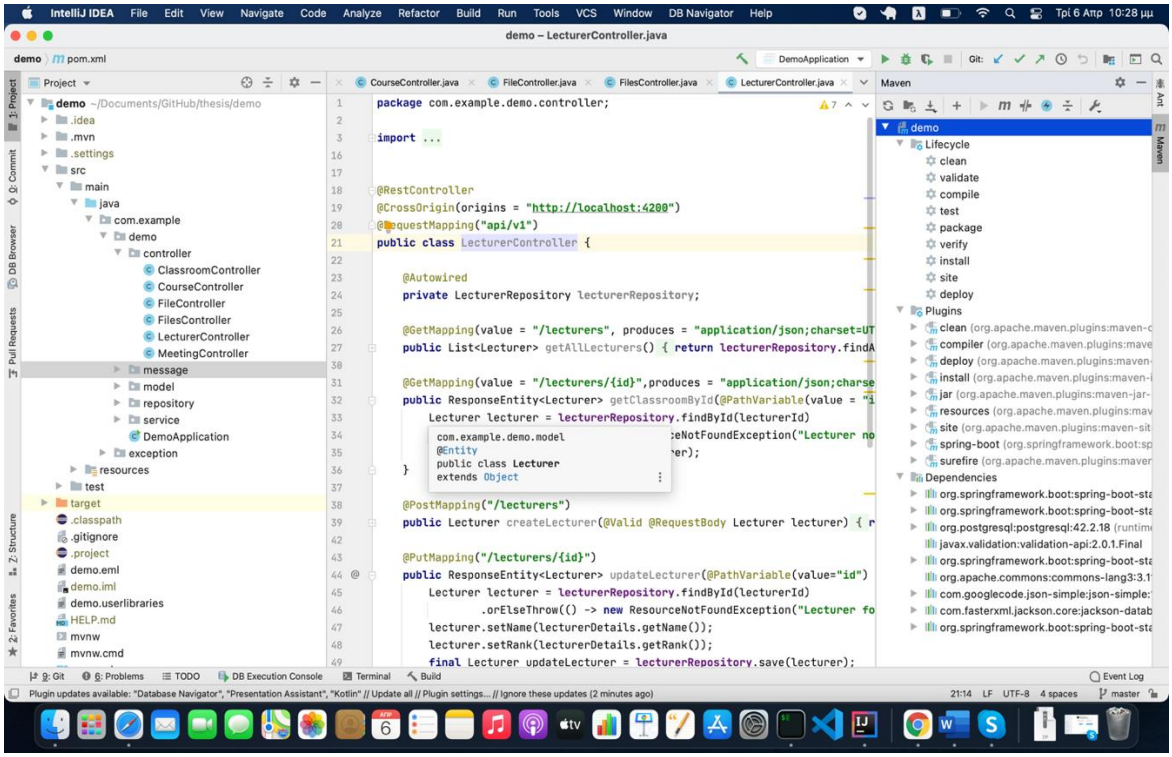
Μερικά από τα καλύτερα χαρακτηριστικά του IntelliJ που συμβάλουν στην ανάπτυξη του λογισμικού είναι οι αλγόριθμοι πρόβλεψης που με ακρίβεια συμπεραίνουν τι θα πληκτρολογήσει ο προγραμματιστής και το συμπληρώνουν ακόμα και αν δεν ξέρει πιο είναι το ακριβές όνομα μιας κλάσης ή μιας μεταβλητής ή όποιου άλλου πόρου. Πιο συγκεκριμένα

- Έξυπνη συμπλήρωση κώδικα

- Chain code completion
- Στατικές μεταβλητές
- Εντοπισμός διπλών εγγραφών
- Επιθεώρηση και γρήγορη αντικατάσταση: Όταν το IntelliJ εντοπίζει ότι πρόκειται ο προγραμματιστής να πέσει σε λάθος, τότε ένα διακριτικό παραθυράκι πετάγεται στην ίδια γραμμή και παρέχει προτεινόμενες λύσεις.

Παρέχει επίσης και πολλά χαρακτηριστικά που κάνουν την εμπειρία του προγραμματιστή καλύτερη όπως:

- Editor-centric περιβάλλον: Γρήγορα αναδυόμενα παραθυρα παρέχουν επιπρόσθετες πληροφορίες χωρίς να χρειαστεί να γίνει αλλαγή σε κάποιο άλλο παράθυρο
- Συντομεύσεις σχεδόν για όλα: Ο IntelliJ είναι ο μοναδικός IDE που παρέχει μια συντόμευση και κάθε ενέργεια συμπεριλαμβανομένου της γρήγορης επιλογής κειμένου και την εναλλαγή παραθύρων και τμημάτων.



Εικόνα 4, IntelliJ IDEA

2.3.2 Σύγκριση μεταξύ των δύο εκδόσεων

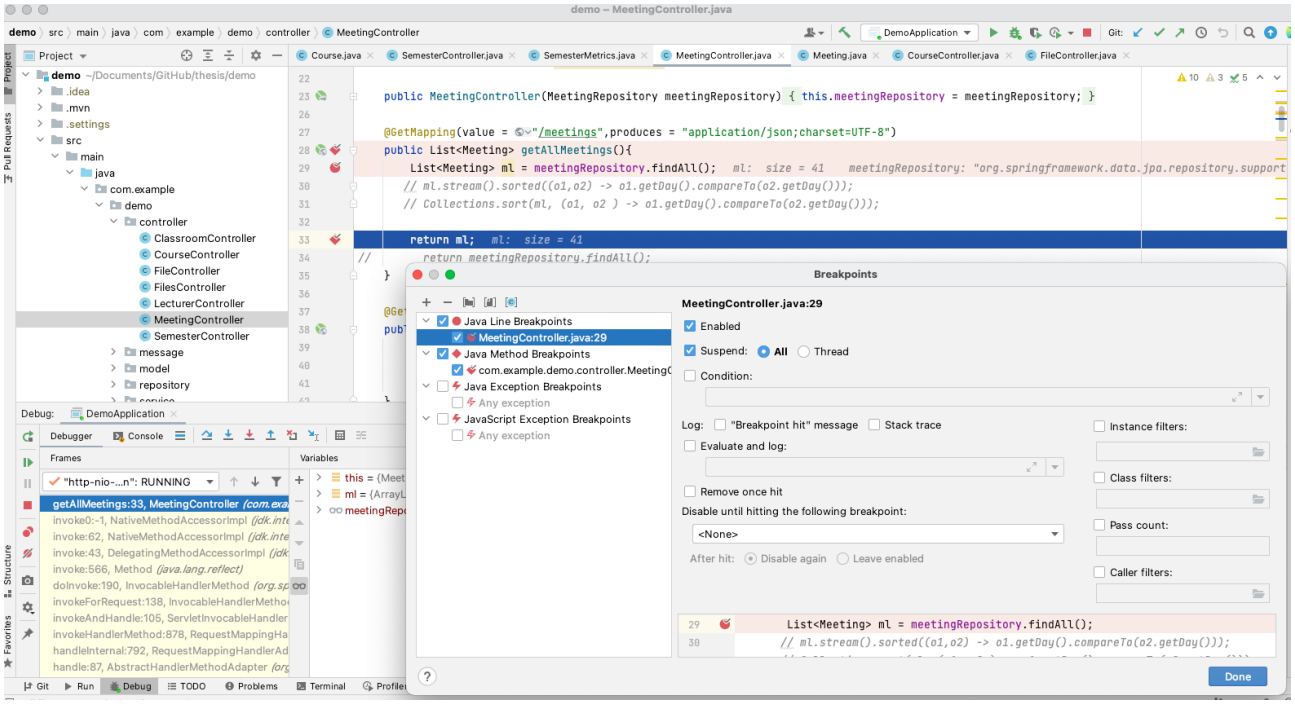
Η έκδοση Ultimate έχει σχεδιαστή για να παρέχει περισσότερα χαρακτηριστικά για την ανάπτυξη εφαρμογών ιστού και επιχειρήσεων ενώ η έκδοση Community που είναι δωρεάν έχει στόχο την εξοικείωση με τις τεχνολογίες Java Virtual Machine και Android αλλά μπορεί επίσης και αυτή να χρησιμοποιηθεί για την ανάπτυξη εμπορικών εφαρμογών.

Feature	Ultimate Edition	Community Edition
License	Commercial	Open-source, Apache 2.0. for commercial development.
Java, Kotlin, Groovy, Scala	Supported	Supported
Android development	Supported	Supported
Maven, Gradle, SBT	Supported	Supported
Git, SVN, Mercurial, CVS	Supported	Supported
Detecting Duplicates	Supported	Not supported
Perforce, TFS	Supported	Not supported
JavaScript, TypeScript	Supported	Not supported
Java EE, Spring, GWT, Vaadin, Play, Grails, Other Frameworks	Supported	Not supported
Database Tools, SQL	Supported	Not supported

Εικόνα 5, σύγκριση εκδόσεων

2.3.3 Debugger

Ένα από τα καλύτερα χαρακτηριστικά του IntelliJ είναι ο debugger που έχει. Τα καθιερωμένα breakpoints πλέον έχουν την λειτουργικότητα να δεσμεύουν όλη την μέθοδο με ένα σύμβολο ρόμβου και να αρχίζει το debugging στην κλήση της μεθόδου και να σταματάει πριν την επιστροφή της τιμής.



Εικόνα 6, στιγμιότυπο από debugging

Σε όλη αυτή τη διαδικασία ο IntelliJ προσφέρει την παρακολούθηση των μεταβλητών "inline", δηλαδή κατά την εκτέλεση των βημάτων στο debugging οι μεταβλητές παίρνουν τιμές δυναμικά πάνω στον κώδικα. Τέλος ένα ακόμα χαρακτηριστικό είναι ο υπολογισμός εκφράσεων στον αέρα "on-fly". Με αυτό τον τρόπο η διαδικασία του debugging έχει ακόμα ένα όπλο στην εύρεση λογικών λαθών.

2.4 Apache Maven

Το Maven (Foundation, 2002-2021) είναι ένα εργαλείο αυτοματισμού κατασκευής και συμπίεσης που χρησιμοποιείται κυρίως σε έργα Java. Βασίζεται στην ιδέα του Project

Object Model (POM). Το Maven επίσης μπορεί να χρησιμοποιεί χρησιμοποιηθεί για την κατασκευή και διαχείριση έργων γραμμένων σε C#, Ruby, Scala και άλλες γλώσσες.

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.3.5.RELEASE</version>
        <relativePath/> <!-- lookup parent from com.example.demo.repository -->
    </parent>
    <groupId>com.example</groupId>
    <artifactId>demo</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>demo</name>
    <description>Demo project for Spring Boot</description>

    <properties>
        <java.version>11</java.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
```

Παράδειγμα

Στοχεύει σε δύο “φάσεις” της ανάπτυξης λογισμικού, στο πως χτίζεται και στις εξαρτήσεις. Σε σχέση με αντίστοιχα εργαλεία όπως το Ant, που χρησιμοποιεί περιορισμούς κατά την ανάπτυξη του λογισμικού και μόνο εξαιρέσεις που πρέπει να γραφτούν. Ένα αρχείο XML περιγράφει όλη τη δομή καθώς και τις εξαρτήσεις που θα έχει το έργο, τόσο σε βιβλιοθήκες αλλά και σε συγκοινωνούντα δομοστοιχεία λογισμικού. Έρχεται με προκαθορισμένο στόχο για απόδοση συγκεκριμένων καλά ορισμένων εργασιών όπως είναι η μεταγλώττιση και το πακετάρισμα. Το Maven δυναμικά κατεβάζει Java βιβλιοθήκες και plug-ins από ένα ή περισσότερα αποθετήρια όπως είναι το Maven2 central repository και τα αποθηκεύει τοπικά στην κρυφή μνήμη. Αυτή η κρυφή μνήμη με τα κατεβασμένα artifacts μπορεί να αναβαθμιστεί.

2.4.1 Εγκατάσταση Maven

Για την επιτυχή εγκατάσταση του maven ο προγραμματιστής θα πρέπει να έχει ήδη εγκαταστήσει στον υπολογιστή του το Java JDK.

2.4.1.1 Windows

Για την εγκατάσταση σε περιβάλλον Windows θα πρέπει να κατεβάσει την τελευταία έκδοση από το επίσημο Apache Maven site και να επιλέξει το Maven zip αρχείο, π.χ. apache-maven-3.3.9-bin.zip. Έπειτα ενημερώνει τις μεταβλητές συστήματος (Environment Variables) με δύο καινούργιες την M2_HOME και την MAVEN_HOME και τις ενημερώνει να έχουν ως φάκελο εκεί όπου έχουμε κατεβάσει το maven zip. Στη συνέχεια ενημερώνει το Path Variable ώστε να βλέπει μέσα στο φάκελο του Maven %M2_HOME%\bin ώστε να μπορεί να χρησιμοποιεί τις εντολές του maven οπουδήποτε. Τέλος για να επιβεβαιώσει ότι η εγκατάσταση έχει γίνει μπορεί να ανοίξει το τερματικό να γράψει mvn -version και θα του επιτρέψει την έκδοση του maven.

2.4.1.2 Linux

Για την εγκατάσταση σε *nix ο προγραμματιστής θα πρέπει να μπει στο επίσημο Apache Maven site και αυτή τη φορά να κατεβάσει το Maven binary tar.gz αρχείο . πχ apache-maven-3.3.9-bin.tar.gz.

Με γνώμονα ότι οι περισσότερες διανομές Linux(Redhat, Ubuntu) χρησιμοποιούν το κέλυφος (Bash) ως προεπιλεγμένο οι παρακάτω διαδικασία είναι με εντολές bash.
Πρώτα θα πρέπει να αποσυμπιέσει το αρχείο που κατέβασε στον φάκελο που επιθυμεί και μετά να προσθέσει το maven στο μονοπάτι του συστήματος. Τέλος να τρέξει την εντολή του maven για να σιγουρευτεί ότι η εγκατάσταση ολοκληρώθηκε.

- $ tar -xvf apache-maven-3.3.9-bin.tar.gz -C /usr/local/apache-maven/apache-maven-3.3.9
- $ nano ~/.bashrc
    - export M2_HOME=/usr/local/apache-maven/apache-maven-3.3.9
    - export M2=$M2_HOME/bin

    - export MAVEN_OPTS=-Xms256m -Xmx512m
    - export PATH=$M2:$PATH
- $ mvn -version

2.4.1.3 Mac OS X

Για την εγκατάσταση σε Mac OS X ο προγραμματιστής θα πρέπει να μπει στο επίσημο Apache Maven site και αυτή τη φορά να κατεβάσει το Maven binary tar.gz αρχείο . πχ apache-maven-3.3.9-bin.tar.gz. (ίδια διαδικασία με το Linux)

Πρώρα πρέπει να ανοίξει το τερματικό και να συνδεθεί ως Super – User,
μετά θα πρέπει να δώσει τα κατάλληλα δικαιώματα στο αρχείο που κατέβασε και μετά να το αποσυμπιέσει στον φάκελο που επιθυμεί. Από εκεί και πέρα θα πρέπει να προσθέσει το δυαδικά αρχεία του maven στο μονοπάτι του συστήματος. Τέλος να τρέξει την εντολή του maven για να σιγουρευτεί ότι η εγκατάσταση ολοκληρώθηκε.

- rm Downloads/apache-maven*bin.tar.gz
- chown -R root:wheel Downloads/apache-maven*
- mv Downloads/apache-maven* /opt/apache-maven
- nano $HOME/.profile
    - export PATH=$PATH:/opt/apache-maven/bin
- $ mvn -version

2.4.2 Κύκλος ζωής

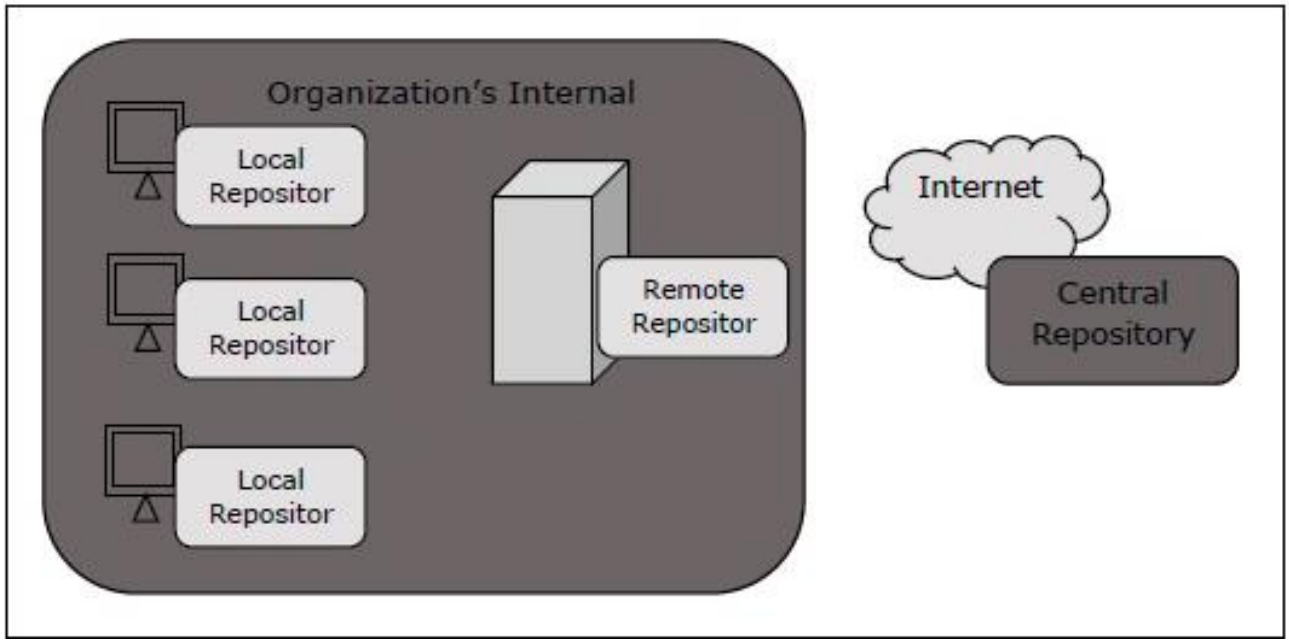
Ο κύκλος ζωής στην ανάπτυξη λογισμικού και πιο συγκεκριμένα στο Apache Maven είναι μια πολύ καθορισμένη σειρά φάσεων που στόχο έχει την σειρά με την οποία θα εκτελεστούν οι προκαθορισμένοι στόχοι. Μερικές από τις πιο βασικές φάσεις

- Validate: Επικυρώνει ότι το project είναι ορθά γραμμένο και δεν χρειάζονται επιπλέον πληροφορίες

- Compile: Μεταγλώττιση του πηγαίου κώδικα
- Test: Έλεγχος του μεταγλωττισμένου κώδικα σύμφωνα με το framework που έχει χρησιμοποιηθεί
- Package: Σε αυτή τη φάση δημιουργείται το JAR/WAR πακέτο ανάλογα με την επιλογή που έχει γίνει στο αρχικό POM.xml
- Install: Εγκατάσταση του πακέτου είτε τοπικά ή σε απομακρυσμένο αποθετήριο του maven
- Deploy: Αντιγράφει το τελικό πακέτο σε απομακρυσμένο αποθετήριο.

2.4.3 Αποθετήρια Maven

Το αποθετήρια του Maven είναι τριών τύπων, τοπικά , κεντρικά και απομακρυσμένα



Εικόνα 7, οπτική απεικόνιση του Maven

Τοπικό αποθετήριο είναι ένας φάκελος που βρίσκεται στο μηχάνημα του προγραμματιστή και δημιουργείται όταν τρέχει μία εντολή του maven. Στόχος του είναι να κρατάει συγκεντρωμένες όλες τις εξαρτήσεις (βιβλιοθήκες, πρόσθετα). Κατά την εκτέλεση της εντολής Maven build, αυτόματα κατεβαίνουν όλες οι εξαρτήσεις και αποθηκεύονται στο

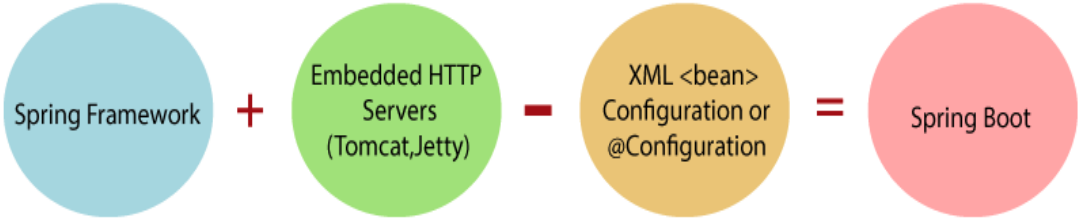
τοπικό αποθετήριο και βοηθάει στο να αποφεύγονται να ξανακατεβαίνουν εξαρτήσεις που είναι αποθηκευμένες σε απομακρυσμένο αποθετήριο.

Κεντρικό αποθετήριο είναι το αποθετήριο που παρέχεται από την κοινότητα του Maven και περιέχει ένα μεγάλο αριθμό από βιβλιοθήκες που χρησιμοποιούνται ευρέος. Όταν το maven δε βρίσκει μια εξάρτηση τοπικά τότε ξεκινάει και ψάχνει στο κεντρικό αποθετήριο : https://repo1.maven.org/maven2/

Μερικές φορές, το Maven δεν μπορεί να βρει την εξάρτηση που ψάχνει στο κεντρικό αποθετήριο , τότε σταματάει την διαδικασία του χτισίματος της εφαρμογής και εμφανίζει μήνυμα λάθους. Για να αποφύγουν τέτοιες συμπεριφορές το Maven παρέχει την ιδέα του απομακρυσμένου αποθετηρίου όπου ο προγραμματιστής δίνει την πληροφορία του δικού προσωπικού αποθετηρίου όπου το Maven θα βρει τις βιβλιοθήκες που ψάχνει καθώς και άλλα jars.

2.5 Spring Boot

Το Spring Boot (Pivotal, 2012-2021) είναι ένα ανοιχτού κώδικα λογισμικό βασισμένο στην γλώσσα προγραμματισμού Java και είναι ένα πρότζεκτ που έχει χτιστεί πάνω στο αρχικό Spring Framework. Αναπτύχθηκε από την Pivotal Team ώστε να παρέχει ένα πιο γρήγορο και εύκολο τρόπο για την ανάπτυξη εφαρμογών που μπορούν να σταθούν μόνες τους καθώς επίσης και για εφαρμογές παραγωγής. Επίσης εφοδιάζει τον προγραμματιστή με την αρχή σχεδίασης RAD (Rapid Application Development) που του δίνει την δυνατότητα να ξεκινήσει μια εφαρμογή με την ελάχιστη παραμετροποίηση του Spring Framework.



Εικόνα 8, Spring boot framework

Εν συντομία το Spring Boot είναι ένας συνδυασμός του Spring Framework και των Embedded Servers.

Micro Service είναι η αρχιτεκτονική που επιτρέπει στους προγραμματιστές να αναπτύξουν ανεξάρτητα services. Κάθε service τρέχει στη δικιά του διεργασία και έτσι επιτυγχάνεται ένα ελαφρύ μοντέλο που μπορεί να υποστηρίξει μεγάλες επαγγελματικές εφαρμογές.

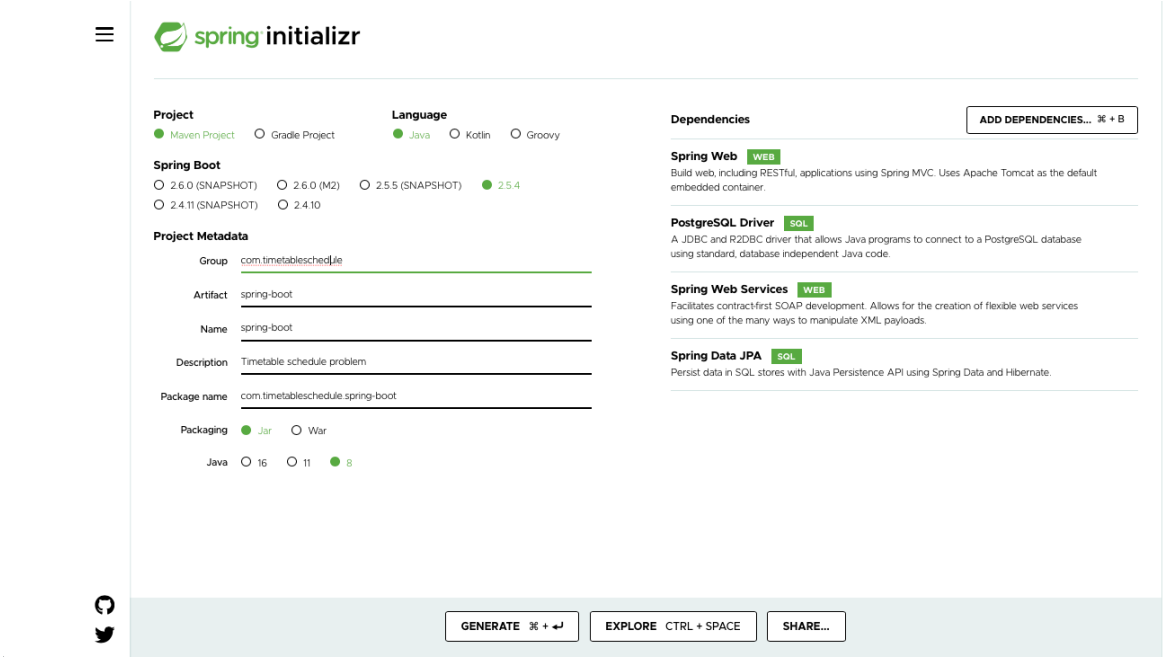
2.5.1 Πλεονεκτήματα

- Ευκολία στην ανάπτυξη
- Δυνατότητα επέκτασης
- Συμβατότητα με Containers
- Μικρή παραμετροποίηση
- Λιγότερος χρόνος μέχρι την παραγωγή

2.5.2 Εγκατάσταση και αρχικοποίηση

Για να ξεκινήσει κάποιος να φτιάξει ένα πρότζεκτ με το Spring Boot θα χρειαστεί να έχει κάνει μια προεργασία. Θα πρέπει να έχει εγκαταστήσει το JDK 1.8 ή νεότερη έκδοση, να έχει το Gradle ή το Maven και τέλος να έχει εγκατεστημένο έναν IDE, πχ Intellij IDEA ή Spring Tool Suite (STS). Μόλις έχει τελειώσει με όλα τα παραπάνω μπορεί να μεταβεί στην σελίδα του spring https://start.spring.io και να επιλέξει από εκεί την έκδοση

της Java , το τρόπο που θέλει να εισάγει το πρότζεκτ, τις εξαρτήσεις που θα έχει, το όνομα και το artifact του πρότζεκτ του και τέλος να πατήσει Generate και να κατεβάσει ένα συμπιεσμένο αρχείο το οποίο θα εισάγει στον IDE του. Για τους χρήστες που έχουν επιλέξει ως IDE τον STS ή τον Intellij IDEA professional η παραπάνω διαδικασία μπορεί να γίνει απευθείας στον IDE.



Εικόνα 9, αρχικοποίηση έργου

2.5.3 Spring Annotations

Τα Spring Boot Annotations είναι μια ειδική κατηγορία μετά-δεδομένων που παρέχουν δεδομένα για το πρόγραμμα, με άλλα λόγια, οι επισημάνσεις (annotations) χρησιμοποιούνται ως επιπρόσθετη πληροφορία για την εφαρμογή και ξεκινούν με το χαρακτηριστικό @. Δεν είναι μέρος της εφαρμογής η οποία αναπτύσσεται, δεν επηρεάζουν άμεσα την λειτουργία του κώδικα στον οποίο έχουν επισημανθεί και δεν έχουν καμία συμμετοχή στην εκτέλεση της εφαρμογής.

Υπάρχουν τέσσερις κατηγορίες επισημάνσεων, οι βασικές επισημάνσεις ή πυρήνα, οι Stereotype επισημάνσεις, οι ειδικές επισημάνσεις για το Spring Boot και οι επισημάνσεις που απευθύνονται στις υπηρεσίες μέσου ιστού(Web Services).

2.5.4 Βασικές επισημάνσεις:

- @Required: Εφαρμόζεται στη μέθοδο στο setter του bean δηλώνοντας ότι πρέπει να παραχθεί κατά την στιγμή της παραμετροποίησης αλλιώς θα πέσει σε εξαίρεση BeanInitilizationException.

- @Autowired: Το Spring παρέχει μια αυτόματη σύνδεση μέσω της επισήμανσης @Autowired και μπορεί να χρησιμοποιηθεί για να συνδέσει ένα bean σε μία μέθοδο setter ή σε ένα πεδίο ή σε μία κλάση κατασκευαστή (Constructor) ώστε το Spring container να το ταιριάξει με βάσει τον τύπο του.

- @Configuration: Είναι μια επισήμανση επιπέδου κλάσης και χρησιμοποιείται από τα Spring Containers ως πηγή για τη παραμετροποίηση των beans

- @ComponentScan: Χρησιμοποιείται όταν χρειάζεται να γίνει σάρωση σε ένα πακέτο για να βρεθούν τα beans, επίσης μπορεί να χρησιμοποιηθεί για την εύρεση components σε ένα πακέτο.

- @Bean: Είναι επισήμανση στο επίπεδο μεθόδου και είναι η εναλλακτική στο XML <bean>. Χρησιμοποιείται για να παρέχει πληροφορία ποια μέθοδο πρέπει να διαχειριστή το Spring Container.

2.5.5 Stereotype Επισημάνσεις:

- @Component: Είναι επισήμανση στο επίπεδο της κλάσης και χρησιμοποιείται για να δηλώσει μια κλάση Java ως bean. Μια κλάση Java που επισημαίνεται με το @Component βρίσκεται στο classpath, το spring framework τη παίρνει από εκεί και την καταχωρεί στην εφαρμογή ως περιεχόμενο Spring Bean.

- @Controller: Είναι επισήμανση στο επίπεδο της κλάσης και μια επιπλέον εξειδίκευση στη επισήμανση @component που καταχωρεί μια κλάση ως χειριστή αιτημάτων που σημειώνονται μέσω ιστού (web request handler). Συχνά χρησιμοποιείται για να παρέχει ιστοσελίδες. Εξ ορισμού επιστρέφει

ένα αλφαριθμητικό (string) που είναι δηλώνει την δρομολογημένη σελίδα. Κυρίως χρησιμοποιείται σε συνδυασμό με την επισήμανση @RequestMapping

- @Service: Είναι επισήμανση επιπέδου κλάσης και δηλώνει στο Spring ότι η κλάση περιέχει την επιχειρησιακή λογική.

- @Repository: Είναι επισήμανση επιπέδου κλάσης. Μέσω του αποθετηρίου παρέχεται πρόσβαση στην βάση δεδομένων και είναι υπεύθυνο για όλες της συναλλαγές με αυτήν.

2.5.6 Spring Boot Επισημάνσεις:

- @EnableAutoConfiguration: είναι μια αυτοματοποιημένη διαδικασία παραμετροποίησης όπου τα beans που βρίσκονται στο classpath παρεμετροποιούνται εκ νέου ώστε να τρέξουν τις μεθόδους τους. Η χρήση του έχει ελαττωθεί από την έκδοση Spring Boot 1.2.0 διότι οι προγραμματιστές βρήκαν μια ενναλακτική επισήμανση την @SpringBootApplication.

- @SpringBootApplication: είναι συνδυασμός τριών επισημάνσεων, του @EnableAutoConfiguration, @ComponentScan και του @Configuration.

2.5.7 REST επισημάνσεις:

- @RequestMapping: Χρησιμοποιείται για την αντιστοίχιση των αιτημάτων μέσω ιστού. Έχει πολλές προεραιτηκές επιλογές όπως κατανάλωση (consume), επικεφαλίδα (header), μέθοδος (method), όνομα (name), παράμετροι (params), μονοπάτι (path), παραγόμενο (produces) και αξία (value). Είναι σε επίπεδο κλάσης καθώς και μεθόδου.

- @PostMapping: Αντιστοιχεί τα αιτήματα HTTP POST σε ένα συγκεκριμένο χειριστή μεθόδου και χρησιμοποιείται ώστε να δημιουργηθεί ένα τελικό σημείο με μια υπηρεσία που στέλνει αντικείμενα.

- @GetMapping: Αντιστοιχεί τα αιτήματα HTTP GET σε ένα συγκεκριμένο χειριστή μεθόδου και χρησιμοποιείται ώστε να δημιουργηθεί ένα τελικό σημείο που φέρνει τα αποτελέσματα. Αντίστοιχο του @RequestMapping (method = RequestMethod.GET)

- @PutMapping: Αντιστοιχεί τα αιτήματα HTTP PUT σε ένα συγκεκριμένο χειριστή μεθόδου και χρησιμοποιείται ώστε να δημιουργηθεί ένα τελικό σημείο με μια υπηρεσία που στέλνει αντικείμενα ή αναβαθμίζει τα είδη υπάρχων.

- @DeleteMapping: Αντιστοιχεί τα αιτήματα HTTP DELETE σε ένα συγκεκριμένο χειριστή μεθόδου και χρησιμοποιείται ώστε να δημιουργηθεί ένα τελικό σημείο με μια υπηρεσία που διαγράφει έναν πόρο.

- @PatchMapping: Αντιστοιχεί τα αιτήματα HTTP PATCH σε ένα συγκεκριμένο χειριστή μεθόδου και χρησιμοποιείται ώστε να δημιουργηθεί ένα τελικό σημείο με μια υπηρεσία που

- @RequestBoddy: Χρησιμοποιείται για να αντιστοιχήσει (bind) ένα αίτημα HTTP με ένα αντικείμενο ως παράμετρο σε μία μέθοδο. Εσωτερικά χρησιμοποιεί τον μηχανισμό HTTP MessageConverters ώστε να μετατρέψει το σώμα του αιτήματος. Όταν επισημανθεί μια μέθοδος με το @Requestbody , το Spring Framework αντιστοιχεί το εισερχόμενο αίτημα HTTP στο αντικείμενο της μεθόδου που βρίσκεται στην υπογραφή της.

- @ResponseBody: Αντιστοιχεί τη επιστρεφόμενη τιμή της μεθόδου στο σώμα του αιτήματος. Το spring framework χρησιμοποιεί την συγκεκριμένη επισήμανση για να δημιουργήσει ένα JSON ή XML αντικείμενο.

- @PathVariable: Χρησιμοποιείται ώστε να εξαχθεί η πληροφορία που έρχεται από το URI και ταιριάζει περισσότερο με υπηρεσίες RESTfull όπου το URL

περιέχει μονοπάτι (path) μεταβλητή. Η μέθδος μπορεί να έχει περισσότερες από μία pathvariables.

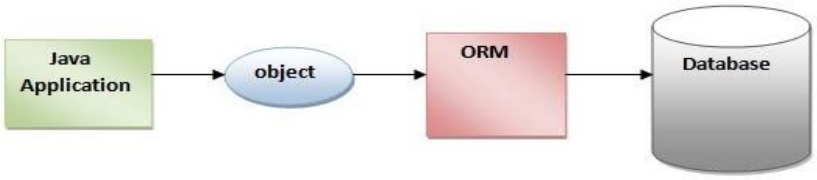
- @RequestParam: Χρησιμοποιείται για να εξάγει πληροφορία που είναι σε ένα ερώτημα (query) από ένα URL.
- @RequestHeader: Δίνει τη πληροφορία μέσω επικεφαλίδων σχετικά με το αίτημα HTTP
- @RestController : Μπορεί να θεωρηθεί ως συνδυασμός των @Controller και @ResponseBody.
- @RequestAttribute

2.6 Hibernate

Το Hibernate (hat, n.d.) είναι ένα Java framework το οποίο απλοποιεί την αλληλεπίδραση μιας Java εφαρμογής πχ Spring-boot με την βάση δεδομένων. Ξεκίνησε το 2001 από τον Gavin King ως εναλλακτική στο EJB2. Είναι ένα ανοιχτού κώδικα, ελαφρύ εργαλείο αντιστοίχισης αντικειμένων (ORM). Το Hibernate εφαρμόζει τις προδιαγραφές του JPA για την διατήρηση των δεδομένων.

2.6.1 Object Relational Mapping Εργαλεία

Τα Object Relational Mapping (ORM) εργαλεία που διευκολύνουν την δημιουργία, το χειρισμό και την προσβασιμότητα στα δεδομένα μιας βάσης. Είναι μια τεχνική προγραμματισμού που χαρτογραφεί τα αντικείμενα μιας κλάσης και τα αντιστοιχεί σε πληροφορία που θα αποθηκευτεί στην βάση δεδομένων. Τα ORM εργαλεία εσωτερικά χρησιμοποιούν το Java Data Base Connectivity Application Programming Interface - JDBC API για να αλληλεπιδρά με την βάση δεδομένων.



Εικόνα 10, δομή του Hibernate

2.6.2 Java Persistence API

Το Java Persistence API είναι μια προδιαγραφή της Java που παρέχει συγκεκριμένη λειτουργικότητα. Το πακέτο -βιβλιοθήκη- javax.persistence περιέχει όλες τις απαραίτητες κλάσης καθώς και τα interfaces.

2.6.3 Πλεονεκτήματα του Hibernate Framework

Μερικά από τα πλεονεκτήματα του Hibernate είναι:

- Λογισμικό ανοιχτού κώδικα υπό την αιγίδα της LGPL και ελαφρύ
- Γρήγορη απόδοση: η απόδοσή του είναι γρήγορη διότι η μνήμη cache χρησιμοποιείται εσωτερικά. Υπάρχουν 2 τύπο cache στο Hibernate το πρώτο επίπεδο και το δεύτερο. Το πρώτο επίπεδο είναι προκαθορισμένο να χρησιμοποιείται με την εγκατάσταση του.
- Ανεξάρτητα ερωτήματα στην βάση δεδομένων: HQL (Hibernate Query Language) είναι η αντικειμενοστραφής έκδοση της SQL και παράγει τα ανεξάρτητα ερωτήματα προς την βάση δεδομένων. Με αυτόν τον τρόπο δεν χρειάζεται ο προγραμματιστής να γράψει μόνος του τα ερωτήματα. Πριν το Hibernate , όταν γινόταν η βάση του πρότζεκτ άλλαζε έπρεπε από ο προγραμματιστής να γράψει από την αρχή τα ερωτήματα προς την βάση και αυτό οδηγούσε σε προβλήματα συντήρησης.
- Αυτόματη παραγωγή πινάκων: Το Hibernate παρέχει την λειτουργία της παραγωγής πινάκων στην βάση δεδομένων αυτόματα με αποτέλεσμα ο προγραμματιστής να μην χρειάζεται να γράψει μόνος του τους πίνακες και τις σχέσεις μεταξύ τους.

- Απλοποίηση σύνθετων συνδέσεων (join): Η φόρτωση δεδομένων από πολλούς πίνακες στη βάση με τη χρήση του Hibernate είναι πλέον απλή.

- Παροχή στατιστικών για τα ερωτήματα και για την βάση γενικότερα: Το Hibernate υποστηρίζει Query cache και παρέχει στατιστικά για αυτά.

2.6.4 Αρχιτεκτονική Hibernate

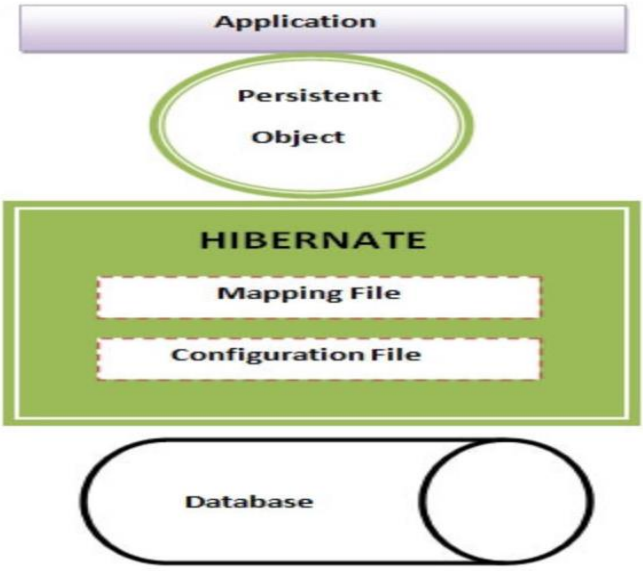
Η αρχιτεκτονική του Hibernate περιλαμβάνει πολλά αντικείμενα όπως το persistent object, session factory, transaction factory, connection factory, session, transaction κλπ., μαζί με είδη υπάρχοντα αντικείμενα και βιβλιοθήκες του Java API όπως το JTA(java transaction API) και το JNDI (Java Naming Directory Interface). Η αρχιτεκτονική του μπορεί να κατηγοριοποιηθεί σε τέσσερα επίπεδα:

- Επίπεδο εφαρμογής Java.

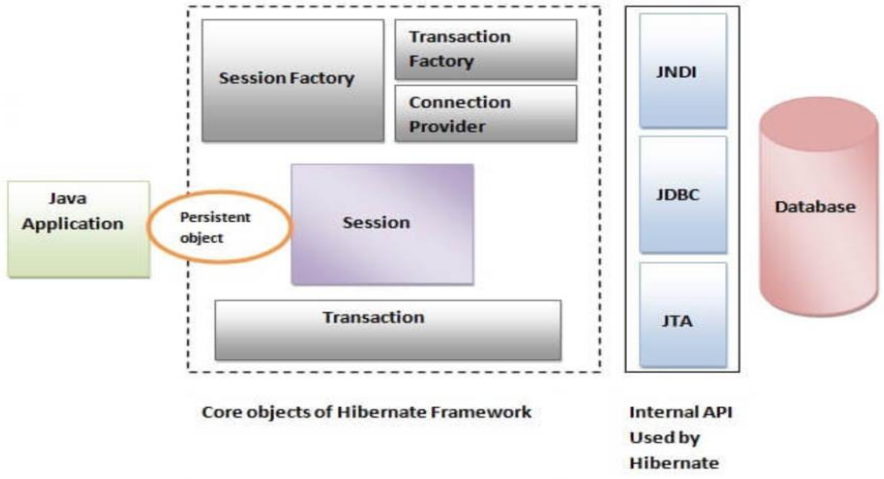
- Επίπεδο Hibernate framework.

- Επίπεδο Backhand API.

- Επίπεδο βάσης δεδομένων.



Εικόνα 11, Αρχιτεκτονική του Hibernate



Εικόνα 12, Αρχιτεκτονική του Hibernate

2.6.5 Στοιχεία της αρχιτεκτονικής του Hibernate

Για να δημιουργήσει ο προγραμματιστής μια εφαρμογή με το hibernate θα πρέπει να ξέρει πως λειτουργούν τα βασικά στοιχεία του.

2.6.5.1 SessionFactory

Το SessionFactory είναι ένα εργοστάσιο συνεδριών και πελάτης του ConnectionProvider. Κρατάει δεδομένα στο δεύτερο επίπεδο της cache. Το org. hibernate. SessionFactory είναι μια διεπαφή(interface) που παρέχει ένα εργοστάσιο μεθόδων ώστε να μπορεί ο προγραμματιστής να πάρει το αντικείμενο της συνεδρίας.

2.6.5.2 Session

Το αντικείμενο session παρέχει μια διεπαφή ανάμεσα στην εφαρμογή και στα δεδομένα που είναι αποθηκευμένα στην βάση δεδομένων. Είναι ένα χαμηλού χρόνου ζωής αντικείμενο που περιτυλίγει (wraps) την JDBC συνδεσιμότητα. Είναι επίσης ένα εργοστάσιο συναλλαγών(Transaction), ερωτημάτων(Query) και κριτηρίων(Criteria). Επίσης κρατάει το πρώτο επίπεδο της cache (υποχρεωτικό) των δεδομένων. Τέλος η διεπαφή org.hibernate.Session παρέχει μεθόδους για την εισαγωγή – insert -, αναβάθμιση – update – και διαγραφή – delete – ενός αντικειμένου.

2.6.6 Εγκατάσταση και εφαρμογή

Η εγκατάσταση του Hibernate σε ένα πρότζεκτ είναι πολύ εύκολη, αρκεί να προσθέσουμε την εξάρτηση στο pox.xml αρχείο και τότε ο IDE θα κατεβάσει όλες τις απαραίτητες βιβλιοθήκες.

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

Εικόνα 13, Hibernate dependency

2.7 Angular

Το Angular (©2010-2021., n.d.) είναι μια ανοιχτού κώδικα πλατφόρμα ανάπτυξης λογισμικού για την μεριά του εξυπηρετητή τόσο σε κινητά, εφαρμογές γραφείου ή μιας σελίδας εφαρμογή (Single Page Application – SPA). Το Angular είναι γραμμένο σε TypeScript που μεταγλωττίζει τον κώδικα σε JavaScript και χρησιμοποιείται Κυρίος για δυναμικές εφαρμογές ιστού. Οι δυναμικές εφαρμογές είναι ιστοσελίδες όπως το Gmail, το Facebook ή το Yahoo όπου έχουν την τάση να αλλάζουν τα δεδομένα, την πληροφορία που παρέχουν σύμφωνα με τρεις παραμέτρους, το χρόνο (ενημερωτικές σελίδες), τη τοποθεσία (ιστοσελίδες προβλέψεις καιρού) και τον χρήστη που τις επισκέπτεται.

2.7.1 Αρχιτεκτονική του Angular

Η αρχιτεκτονική του Angular βασίζεται σε 4 κύρια σημεία

- Τα συνθετικά – Components
- Τα πρότυπα με δέσιμο δεδομένων και τις οδηγίες – Templates, data binding and directives
- Τα δομοστοιχεία – Modules

- Τις υπηρεσίες και την εξάρτηση αναμεσά τους -Services & dependency injection

Modules, components και services είναι κλάσεις που έχουν συγκεκριμένη σήμανση. Αυτή η σήμανση παρέχει μετά-δεδομένα στο Angular για να είναι σε θέση να γνωρίζει πως να χρησιμοποιήσει αυτές τις κλάσεις.

2.7.2 Εγκατάσταση – Αρχικοποίηση

Για να ξεκινήσει κάποιος να αναπτύσσει την δικιά του εφαρμογή με την Angular θα πρέπει πρώτα να έχει εγκαταστήσει το Node.js. Το Node.js είναι μια ανοιχτού λογισμικού πλατφόρμα για την πλευρά του εξυπηρετητή (server) και είναι διαθέσιμο για όλα τα περιβάλλοντα (Windows, Linux, Mac OS). Η εγκατάσταση του είναι αρκετά εύκολη, ο προγραμματιστής πρέπει να επισκεφθεί το επίσημο web site του node.js και να κατεβάσει από εκεί το αντίστοιχο αρχείο για το περιβάλλον ανάπτυξης το οποίο χρησιμοποιεί.

Εικόνα 14, εκδόσεις του Angular για όλα τα λειτουργικά συστήματα

Για την επαλήθευση της εγκατάστασης θα πρέπει να ανοίξει το τερματικό (prompt) και να τρέξει την εντολή node -v και την node -v και να δει τις αντίστοιχες εκδόσεις που έχει εγκαταστήσει. Οι εφαρμογές του angular εξαρτώνται από τον διαχειριστή πακέτων npm (npm packages) για επιπλέον λειτουργικότητα. Ο npm client εγκαθίσταται μαζί με το node.js. Μόλις ολοκληρωθούν τα παραπάνω βήματα η εγκατάσταση του angular και η αρχικοποίηση του πρώτου πρότζεκτ είναι απλή. Πρώτα θα πρέπει να εγκαταστήσει το Angular CLI το οποίο χρησιμοποιείται για να δημιουργήσει το πρότζεκτ

npm install -g @angular/cli

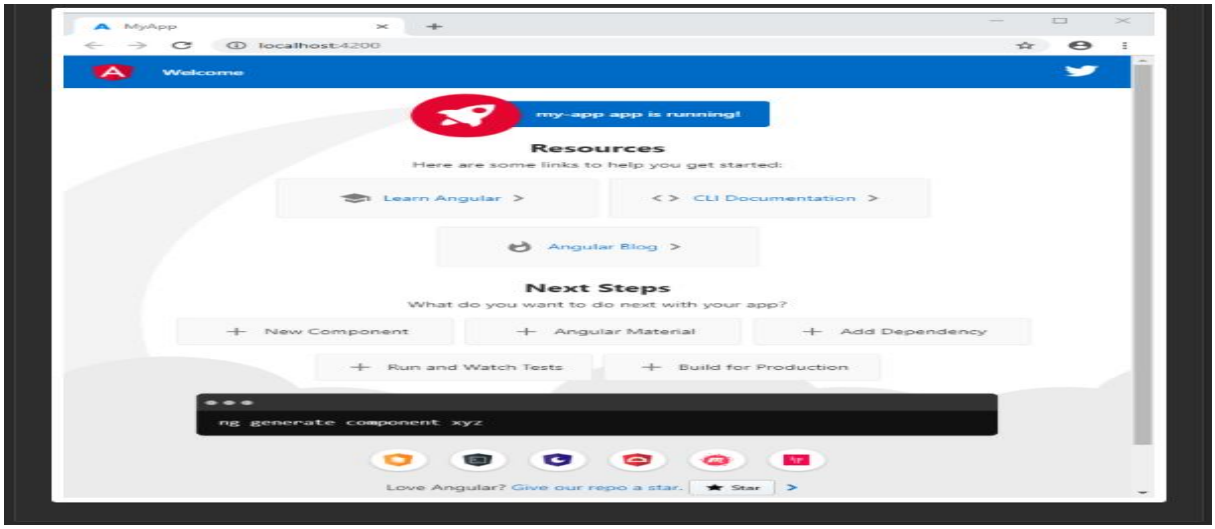
Ύστερα να τρέξει την εντολή ng new οι οποία δημιουργεί το πρότζεκτ ακολουθούμενη από το όνομα

ng new spring-front-end

Τέλος για να τρέξει το πρότζεκτ θα πρέπει να μετακινηθεί στον φάκελο του πρότζεκτ και να τρέξει την εντολή και να ανοίξει το browser της επιλογής του στη διεύθυνση localhost:4200 και να δει την αρχική σελίδα του angular όπου μπορεί να βρει χρήσιμες πληροφορίες καθώς και οδηγούς για την επιπλέον ανάπτυξη της εφαρμογής του.

cd spring-front-end

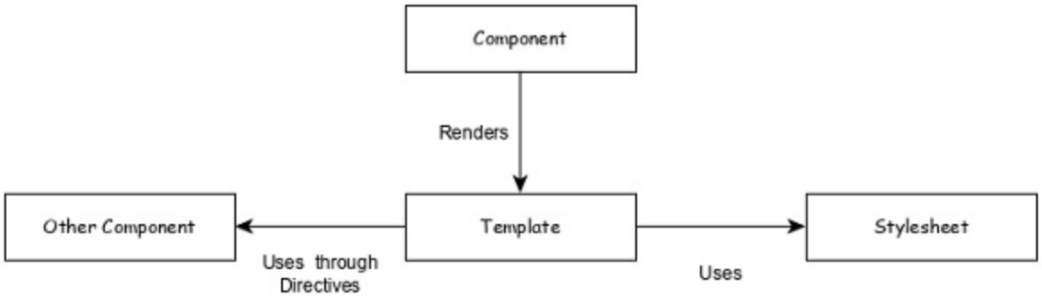
ng serve –open



Εικόνα 15, homepage κατά την πρώτη εκκίνηση του Angular

2.7.3 Component

Ο πυρήνας του του Angular είναι τα Angular components, πρόκειται για τμήματα κώδικα που συνθέτουν κάθε εφαρμογή που έχει γραφτεί σε Angular. Επί της ουσίας είναι απλές κλάσης JavaScript / Typescript μαζί με HTML πρότυπα σε συνάρτηση του ονόματός τους και η κύρια δουλεία τους είναι να παράγουν τμήματα που θα είναι ορατά σε μια σελίδα γνωστά και ως view. @Component είναι η σήμανση που μετατρέπει μια απλή κλάση Typescript σε Angular Component.



Εικόνα 16, δομή ενός δομοστοιχείου

2.7.4 Templates

Πρότυπο είναι ένα υπέρ-σύνολο HTML. Περιέχει όλα τα χαρακτηριστικά της HTML και επιπλέον δίνει περισσότερη λειτουργικότητα ώστε να έχει μεγαλύτερη αλληλεπίδραση με τα δεδομένα που παράγονται δυναμικά σε μια σελίδα.

Τα πρότυπα μπορούν να χωριστούν σε δύο κατηγορίες:

- Δέσιμο δεδομένων – Data binding : {{ title }}
- Οδηγίες – Directives: χρησιμοποιούνται για να δημιουργηθούν πολύπλοκα στοιχεία HTML

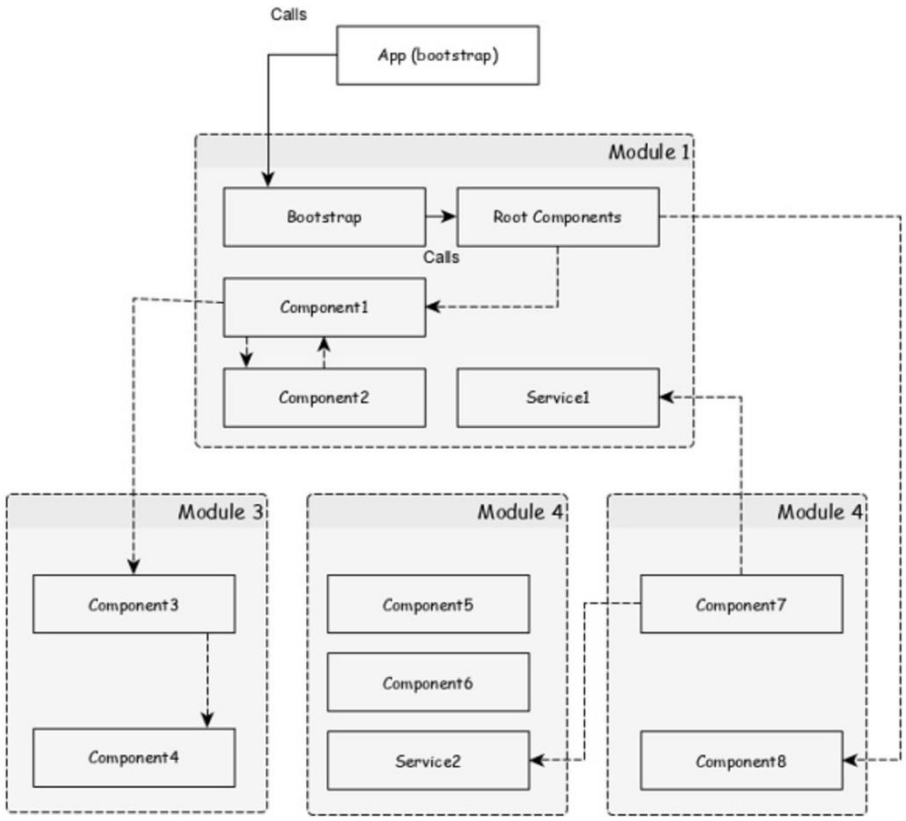
```
<p *ngIf="canShow">
   This sectiom will be shown only when the *canShow* propery's value
<p [showToolTip]='tips' />
```

Εικόνα 17, κομμάτι από ένα template

2.7.5 Modules

Τα δομοστοιχεία στην Angular είναι μια συλλογή από χαρακτηριστικά και λειτουργικότητα, πιο συγκεκριμένα ένα δομοστοιχείο στην Angular ομαδοποιεί πολλά συνθετικά (components) και υπηρεσίες (services) κάτω από ένα μόνο δομοστοιχείο (module).

Μία εφαρμογή Angular μπορεί να έχει πολλά δομοστοιχεία αλλά μόνο ένα μόνο μπορεί να θεωρηθεί ως το κεντρικό (root) στο οποίο είναι υπεύθυνο για τα υπόλοιπα. Ένα δομοστοιχείο μπορεί ο προγραμματιστής να το παραμετροποίηση ώστε να παρέχει την απαραίτητη λειτουργικότητα ανάμεσα στα διάφορα δομοστοιχεία της εφαρμογής. Εν ολίγης συνθετικά από δομοστοιχεία μπορούν να έχουν πρόσβαση σε υπηρεσίες από άλλα συνθετικά.



Εικόνα 18, δομή ενός module

Τμήμα από το κεντρικό δομοστοιχείο της εφαρμογής φαίνεται στην παρακάτω εικόνα

- NgModule - είναι σήμανση που χρησιμοποιείται για την μετατροπή της απλής Typescript / JavaScript σε κλάση του δομοστοιχείου του Angular

- Declarations – επιλογή που χρησιμοποιείται για να συμπεριληφθεί ένα συνθετικό στο κεντρικό δομοστοιχείο

- Bootstrap – επιλογή που χρησιμοποιείται για να θέσει πιο είναι το κεντρικό δομοστοιχείο

- Providers - επιλογή που χρησιμοποιείται για να συμπεριληφθεί μια υπηρεσία στο κεντρικό δομοστοιχείο

- Imports – επιλογή που χρησιμοποιείται για να συμπεριληφθούν τα υπόλοιπα δομοστοιχεία στο κεντρικό δομοστοιχείο

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component'; @NgModule({
   declarations: [
      AppComponent
   ],
   imports: [
      BrowserModule
   ],
   providers: [],
   bootstrap: [AppComponent]
})
export class AppModule { }
```

Εικόνα 19, τμήμα κώδικα ενός module

2.7.6 Services

Οι υπηρεσίες είναι απλές Typescript / JavaScript κλάσεις που παρέχουν μια συγκεκριμένη λειτουργικότητα και να την παρέχουν σωστά. Ο στόχος τους είναι η επαναχρησιμοποιήσει τους.

Ένα service μπορεί να χρησιμοποιηθεί από ένα component ως εξάρτηση, κάνοντας έτσι τον κώδικα που εύχρηστό και πιο αποτελεσματικό.

Πολλές φορές θα χρειαστεί να ένα κομμάτι κώδικα να είναι διαθέσιμο σε όλη την εφαρμογή π.χ. δεδομένα ανάμεσα στα components. Για να το πετύχουν αυτό το Angular παρέχει μια συγκεκριμένη κλάση το Service. Με το service μπορεί ο προγραμματιστής να μοιραστεί μεθόδους και μεταβλητές

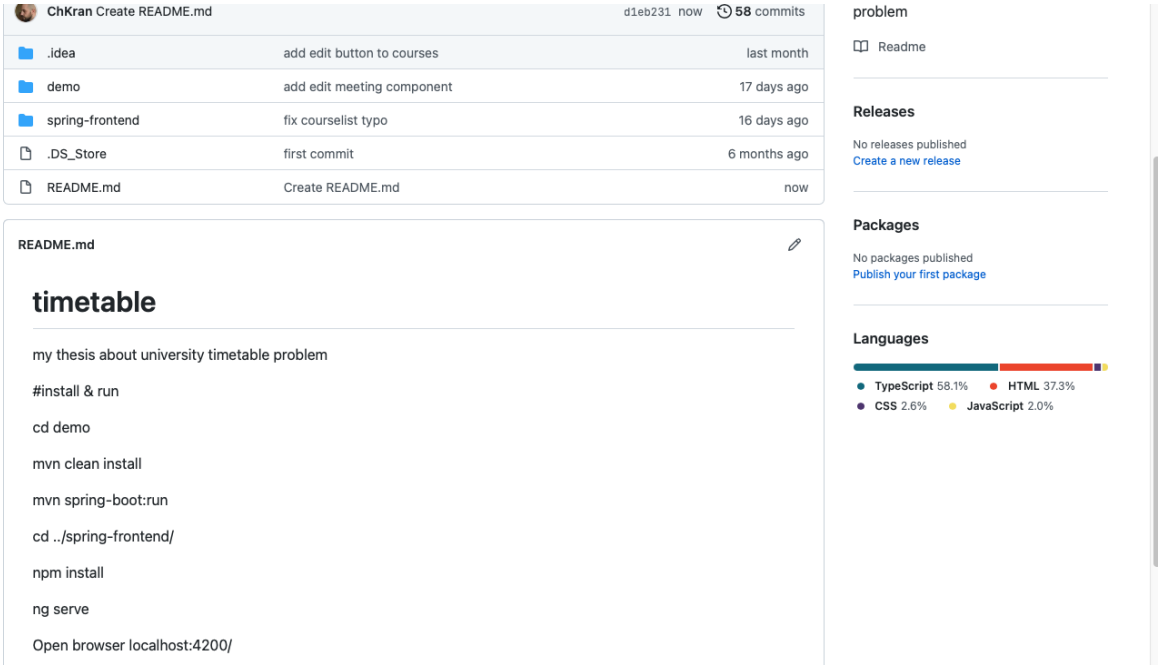
2.7.7 Http Client

Ο εξυπηρετητής http είναι ένα απαραίτητο χαρακτηριστικό για κάθε μοντέρνα εφαρμογή. Στις σημερινές ημέρες πολλές εφαρμογές εκθέτουν τα δεδομένα τους μέσω των REST APIs, για αυτό το λόγο η ομάδα του Angular παρέχει ένα δομοστοιχείο module το

HttpClientModule και μια υπηρεσία την HttpClient ώστε ο προγραμματιστής να έχει πρόσβαση σε όλες τις λειτουργίες που γίνονται με το πρωτόκολλο http.

3 Υλοποίηση εφαρμογής

Η παρούσα διπλωματική εργασία ήθελε να αναδείξει το πρόβλημα του χρονοπρογραμματισμού μέσω της ανάπτυξης μιας ολοκληρωμένης εφαρμογής, βάση δεδομένων, μεριά εξυπηρετητή, μεριά πελάτη. Έχοντας περιγράψει τις τεχνολογίες που έχουν χρησιμοποιηθεί είναι πλέον εύκολο να περάσουμε από την επιφάνια στον πυρήνα της εφαρμογής και στο πως όλα αυτά «δένουν» μεταξύ τους. Ολόκληρο το project είναι διαθέσιμο στο GitHub μαζί με οδηγίες εγκατάστασης και εκκίνησης της εφαρμογής: *https://github.com/ChKran/timetable*.

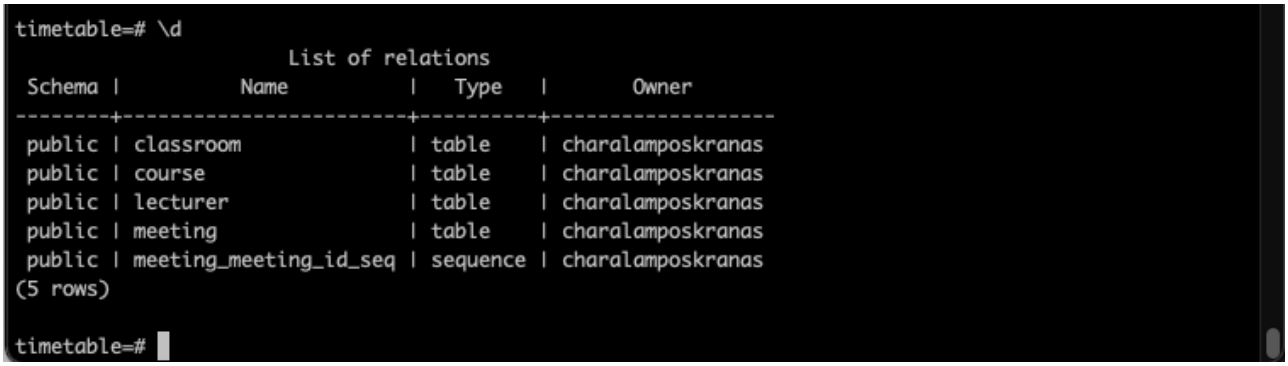


Εικόνα 20, GitHub αποθετήριο

3.1 Βάση δεδομένων

Η βάση δεδομένων είναι το κομμάτι τις εφαρμογής όπου ο προγραμματιστής έχει την μικρότερη επαφή. Πριν μερικά χρόνια για την ανάπτυξη μιας εφαρμογής το πρώτο πράγμα που είχε να κάνει η ομάδα των προγραμματιστών ήταν να καθορίσουν τις

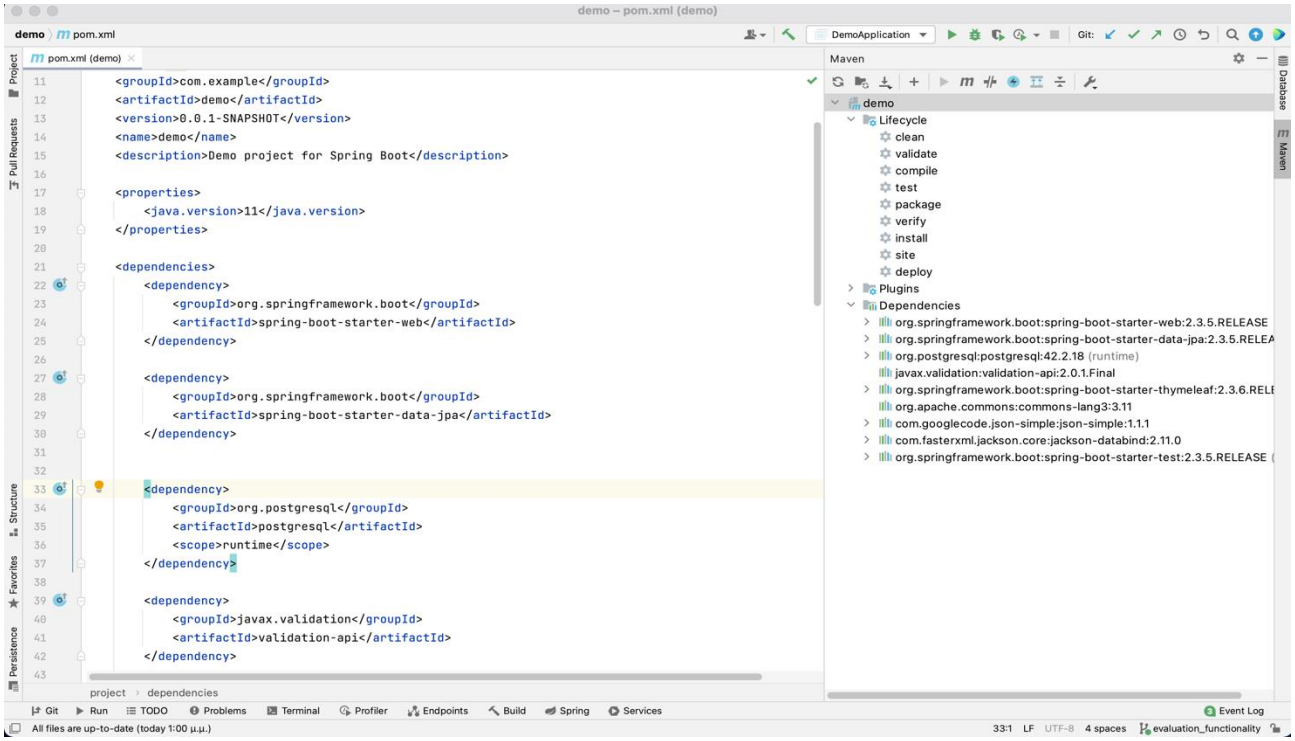
προδιαγραφές, να βρουν τις σχέσεις μεταξύ των οντοτήτων και να τα μετατρέψουν σε μορφή κατανοητή στη βάση δεδομένων (SQL). Με το πέρασμα των χρόνων και την αυξανόμενη ζήτηση για εφαρμογές έπρεπε να βρεθούν λύσεις που να μειώνουν το χρόνο υλοποίησής τους. Μια από αυτές είναι τα ORM εργαλεία που περιεγράφηκαν παραπάνω. Η εικόνα της βάσης είναι όπως έχει δημιουργηθεί μέσω του Hibernate φαίνεται στην εικόνα παρακάτω.



Εικόνα 21, στιγμιότυπο της βάσης δεδομένων

3.2 Μεριά εξυπηρετητή - Back end

Το back end της εφαρμογής είναι το κόμματί εκείνο όπου κρύβεται όλη η ουσία της. Έχοντας αρχικοποίησή το spring-boot project όπως έχει περιγραφή σε παραπάνω ενότητα το πρώτο πράγμα που θα κάνει ο προγραμματιστής είναι να το εισάγει και να κατεβάσει τις απαραίτητες βιβλιοθήκες στο τοπικό του αποθετήριο. Για να προστεθεί μια νέα εξάρτηση το μόνο που πρέπει να κάνει ο προγραμματιστής είναι να την βρει το επίσημο site του maven, να την αντιγράψει και να την επικολλήσει στο pom.xml και να κάνει clean install.



Εικόνα 22, αρχείο εξαρτήσεων

Στην παραπάνω εικόνα φαίνονται οι εξαρτήσεις που έχει η εργασία και που είναι απαραίτητες για την διεκπεραίωση της. Μερικές από αυτές είναι το spring-boot-starter-web που είναι η βασική βιβλιοθήκη του spring για να αναπτυχθεί μια εφαρμογή ιστού, η PostgreSQL που περιέχει όλες τι απαραίτητές κλάσεις ώστε να γίνετε η επικοινωνία ανάμεσα στην βάση δεδομένων και την εφαρμογή (αν κάποιος επιλέξει μια άλλη βάση δεδομένων π.χ. την MySQL τότε πολύ εύκολα μπορεί να αντικαταστήσει την PostgreSQL με την εξάρτηση της MySQL και κάνοντας clean install να έχει αλλάξει βάση δεδομένων), η starter-data-jpa που είναι η υλοποίηση του hibernate και το jpa, απαραίτητα για την αυτοματοποίηση των CRUD λειτουργιών στην εφαρμογή. Μετά την εισαγωγή των εξαρτήσεων θα πρέπει να διαμορφώσει κατάλληλα το αρχείο application.properties που είναι ο οδηγός του spring boot ώστε να μπορέσει να γίνει η σύνδεση με την βάση δεδομένων. Πιο αναλυτικά στις πρώτες γραμμές του αρχείου δηλώνεται η βάση δεδομένων και τα στοιχεία του χρήστη (schema). Οι επόμενες γραμμές δίνουν στον χρήστη μια πληθώρα επιλογών για ενεργοποίηση του μηχανισμού βέλτιστων ερωτήσεων (queries) και την επιλογή της αυτόματης αναβάθμισης της βάσης δεδομένων σε περίπτωση που προστεθεί ή αφαιρεθεί κάτι από μία κλάση. Για μικρά έργα είναι αρκετά βολικό, σε μεγάλα όμως αυξάνει αρκετά το χρόνο μεταγλώττισης και υπάρχει ο κίνδυνος να «σκάσει» η εφαρμογή.

Εικόνα 23, αρχείο παρεμετροποίησης application.properties

Η υλοποίηση της εφαρμογής έχει βασιστεί στο πρότυπο σχεδίασης Model View Controller MVC, το συγκεκριμένο πρότυπο βοηθά τον προγραμματιστή να κρατήσει χωριστά τα κομμάτια της εφαρμογής στην μεριά του server. Πιο αναλυτικά:

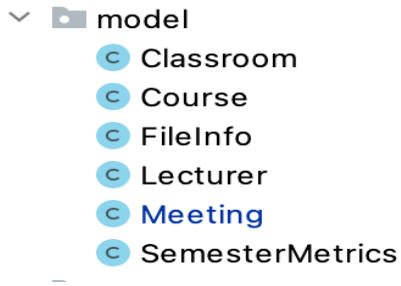
- Model: αναπαριστά ένα αντικείμενο ή ένα JAVA POJO (Plain Old Java Object) το οποίο μεταφέρει πληροφορία και μπορεί επίσης να την ανανεώσει όταν πραγματοποιηθεί μια ενέργεια από τον controller.

- View: αναπαριστά την απεικόνιση των δεδομένων που έχει το model, στην συγκεκριμένη εφαρμογή τον ρόλο αυτό παίζει το front end (Angular).

- Controller: ενεργεί ως συνδυασμός των model και view. Ελέγχει την ροή των δεδομένων στο αντικείμενο model και ανανεώνει το view όταν αυτό χρειάζεται. Επίσης είναι αυτό που κουβαλάει την επιχειρησιακή λογική καθώς και τους ελέγχους που γίνονται πριν μία ενέργεια φτάσει στην βάση δεδομένων.

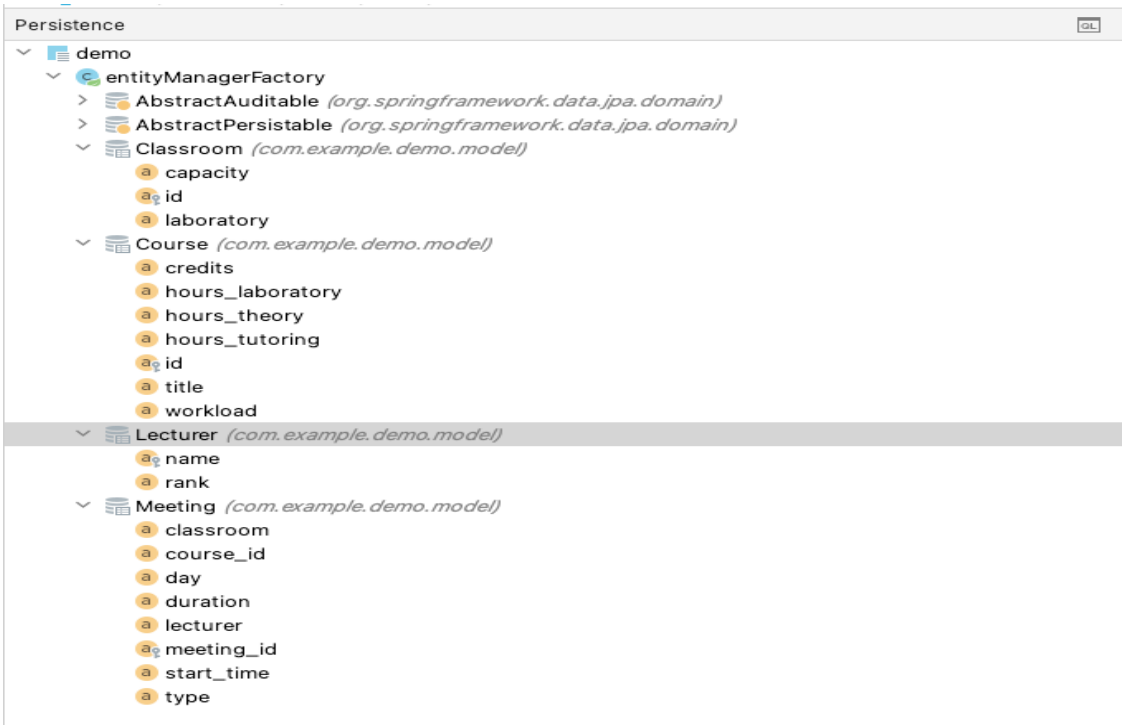
3.2.1 Model

Τα models σε μια εφαρμογή είναι οι κλάσεις που είναι υποψήφιες να γίνουν πίνακες στη βάση δεδομένων μέσο του hibernate και τον κατάλληλων επισημάνσεων. Είναι ομαδοποιημένες σε ένα δικό τους πακέτο που συνήθως φέρει το όνομα model όπως φαίνεται και στην παρακάτω φωτογραφία.



Εικόνα 24, κλάσεις java που μετατρέπονται σε οντότητες

Στην συγκεκριμένη εφαρμογή οι κλάσεις που μεταγλωττίζονται και γίνονται πίνακες στην βάση δεδομένων φέρουν την επισήμανση στην αρχή τους @Entity και @Table και είναι οι Classroom, Course, Lecturer και Meeting και στην βάση έχουν την παρακάτω μορφή. Οι υπόλοιπες είναι βοηθητικές κλάσεις που στόχο έχουν να εξομαλύνουν την ανάπτυξη του λογισμικού.



Εικόνα 25, μετάφραση κλάσεων σε sql

Ένα πολύ βασικό κομμάτι στα models παίζουν οι σχέσεις ανάμεσα τους. Σε μια σχεσιακή βάση δεδομένων πρέπει να δηλωθεί το πρωτεύων κλειδί και το ξένο κλειδί που δείχνει πως συνδέονται δύο ή περισσότεροι πίνακες. Στο spring-boot και με την βοήθεια του hibernate δηλώνονται οι σχέσεις ανάμεσα στις κλάσεις με τα κατάλληλα annotations και

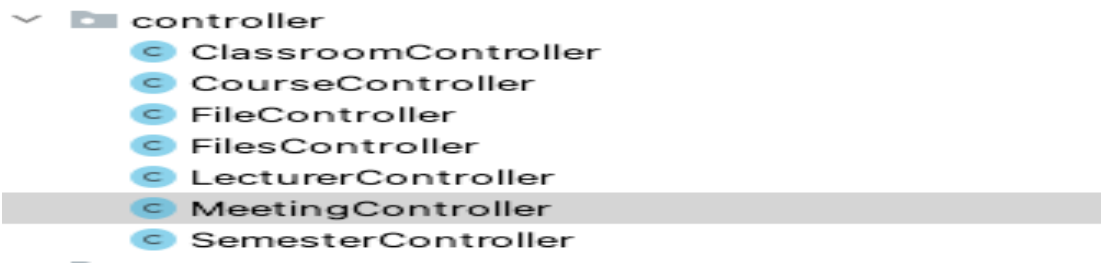
φαίνεται η πληθικότητα π.χ. (@ManyToOne, @JoinColumn). Τέλος ένα σημείο που αξίζει να προσέξει κάποιος είναι η μεταβλητές που έχουν την επισήμανση @Transient οι οποίες παρόλο που ανήκουν στην κλάση που μεταγλωττίζεται σε πίνακα στην βάση δεδομένων αυτές δεν γίνονται πεδίο. Είναι εσωτερικές βοηθητικές μεταβλητές, στην συγκεκριμένη εφαρμογή ο αρχικός σχεδιασμός της κλάσης meetings δεν είχε τον τίτλο του μαθήματος παρά μόνο το κωδικό του, στην πορεία όμως χρειάστηκε, για να αποφευχθεί ο επανασχεδιασμός της κλάσης αλλά και για να δειχθούν οι δυνατότητες του spring-boot προστέθηκε μια transient μεταβλητή με τον τίτλο του μαθήματος.

```
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private long meeting_id;
@ManyToOne(fetch = FetchType.EAGER)
@JoinColumn(name = "course_id" , nullable = false, referencedColumnName = "id")
private Course course_id;
@Column(name = "type")
private String type;
@Column(name = "day")
private String day;
@Column(name = "duration")
private int start_time;
@Column(name = "start_time")
private int duration;
@Column(name = "lecturer")
@ManyToOne(fetch = FetchType.EAGER)
@JoinColumn(name = "lecturer" , nullable = false, referencedColumnName = "id")
private Lecturer lecturer;
@ManyToOne(fetch = FetchType.EAGER)
@JoinColumn(name = "classroom" , nullable = false, referencedColumnName = "id")
private Classroom classroom;
@Transient
private String title;
```

Εικόνα 26, επισήμανση πεδίων μιας κλάσης java

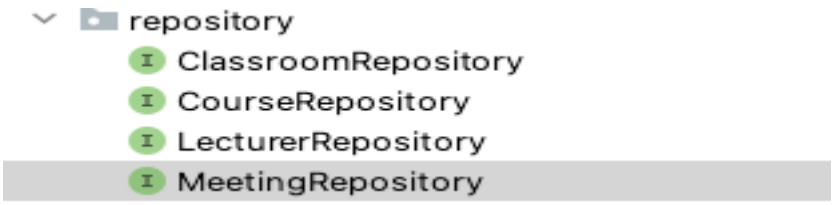
3.2.2 Controller

Οι controllers είναι οι κλάσεις που είναι αρμόδιες για την επιχειρησιακή λογική της εφαρμογής. Όπως και τα model έτσι και αυτοί είναι ομαδοποιημένη σε ένα δικό τους πακέτο. Μια καλή πρακτική είναι να διαχωρίζονται οι Web Controllers (get requests) από τους Rest Controllers (post, put, delete requests).



Εικόνα 27, οι controllers της εφαρμογής

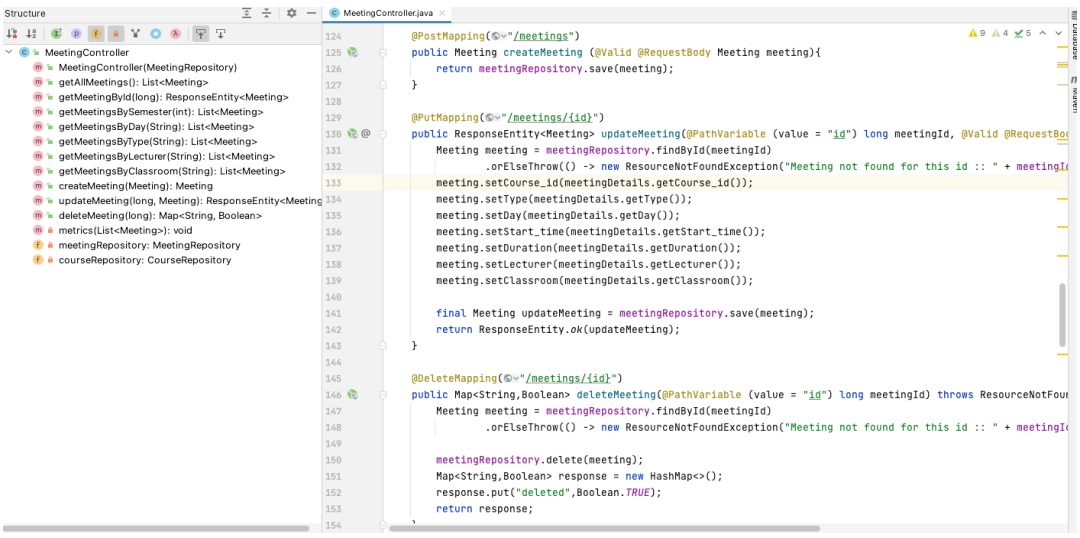
Πέρα όμως από την επιχειρησιακή λογική της εφαρμογής είναι αρμόδιοι και για την διεκπεραίωση ενέργειων όπως η δημιουργία νέων αντικειμένων, η ανανέωση των ήδη υπάρχοντος και η διαγραφή, όλες δηλαδή οι ενέργειες που γίνονται σε μια βάση δεδομένων. Οι Controllers μέσω του dependency injection που είναι το χαρακτηριστικό που έκανε το spring ευρέως γνωστό, χρησιμοποιούν τα interfaces τα οποία με την βοήθεια του hibernate καλύπτουν τις παραπάνω ανάγκες. Είναι τα γνωστά repositories και όπως και οι controllers και τα models έτσι και αυτοί ομαδοποιούνται και ανήκουν σε ένα δικό τους πακέτο. Η κύρια εργασία του repository είναι να παρέχει στον προγραμματιστή τους βασικούς μηχανισμούς επικοινωνίας με την βάση δεδομένων αλλά να του δίνει και την δυνατότητα να γράψει τα δικά του πιο σύνθετα ερωτήματα προς την βάση.




```
@Repository
public interface MeetingRepository extends JpaRepository<Meeting,Long> {
}
```

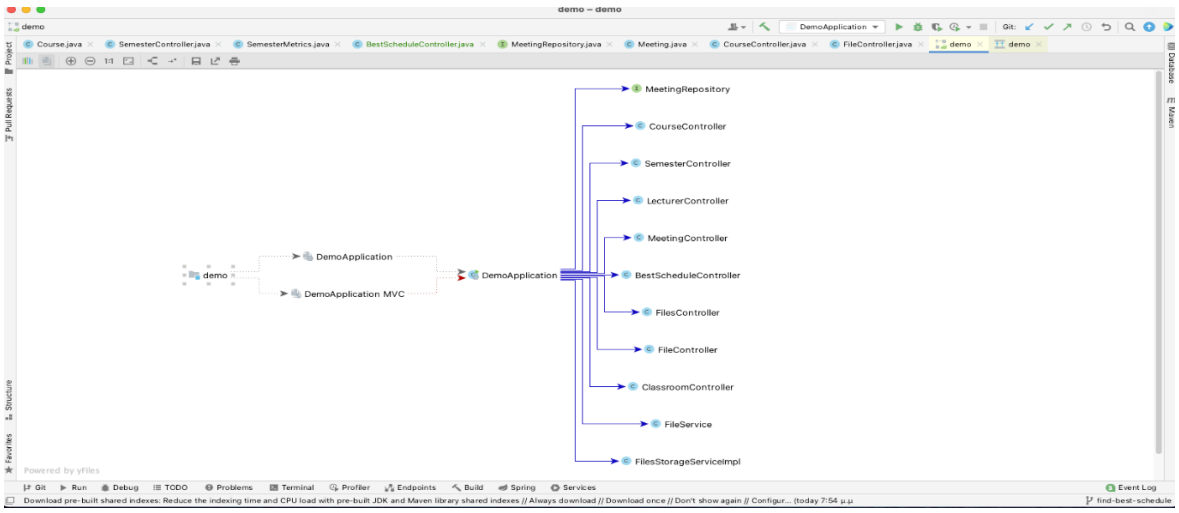
Εικόνα 28,interfaces της εφαρμογής

Πιο αναλυτικά ένας Controller είναι όπως φαίνεται στην παρακάτω εικόνα. Οι επισημάνσεις στο πάνω μέρος κάθε μεθόδου καθορίζουν την αρμοδιότητα την οποία έχει.

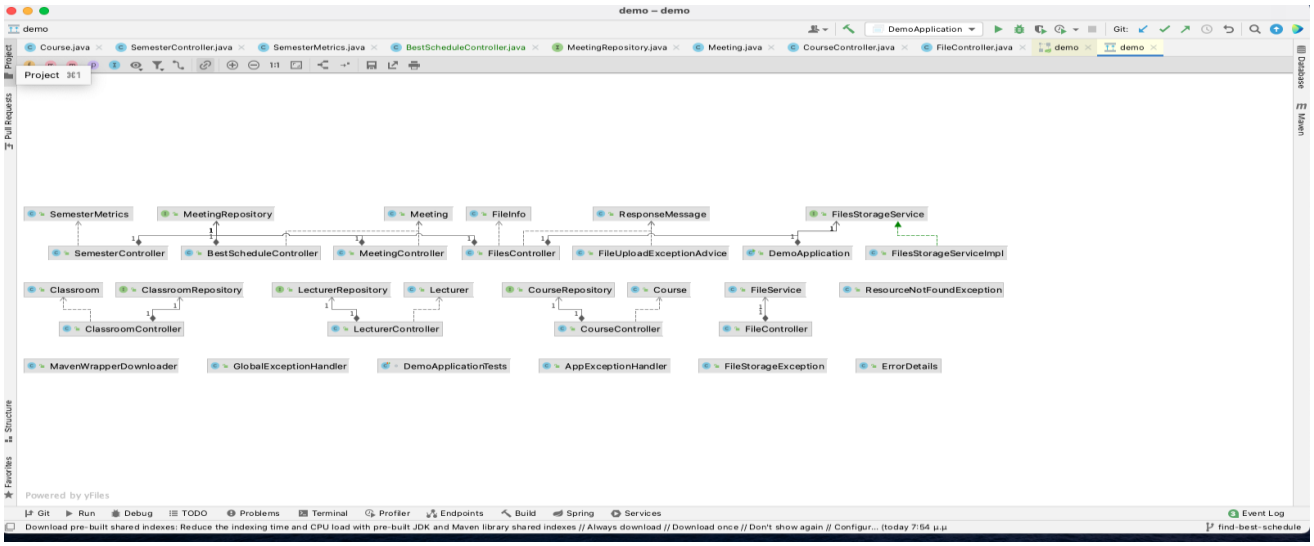


Εικόνα 29, τμήμα κώδικα ενός Controller

Συνοψίζοντας όλο το back end της εφαρμογής φαίνεται στα παρακάτω UML διαγράμματα.



Εικόνα 30, φαδσφα

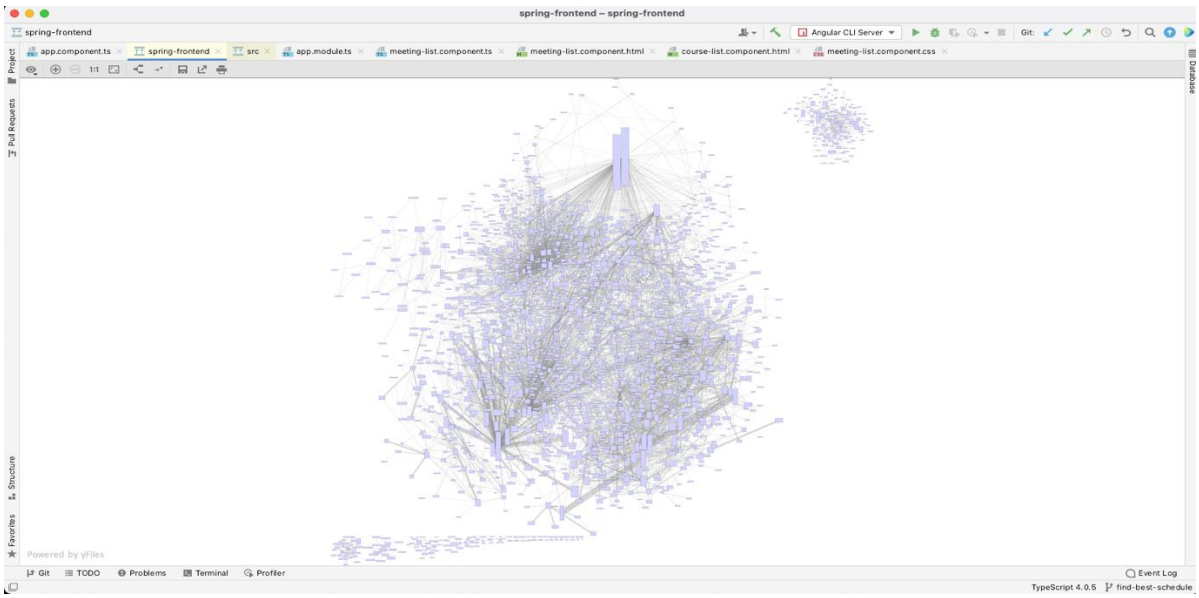


Εικόνα 31, UML class diagramm

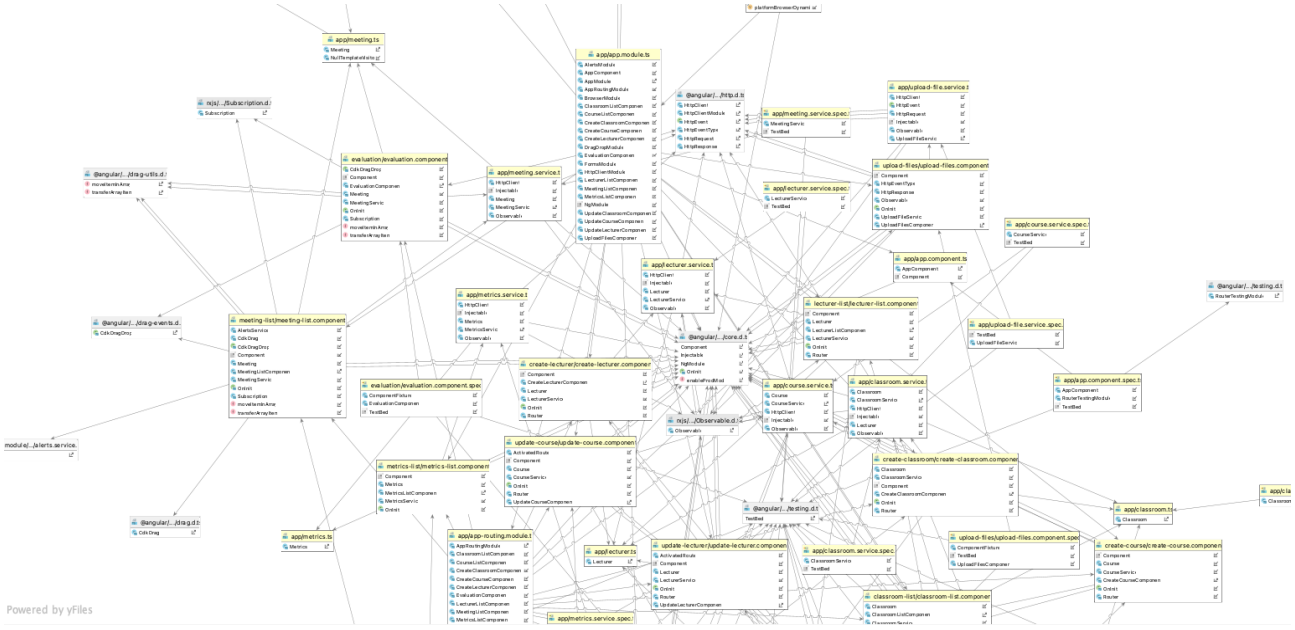
3.3 Μεριά Πελάτη Front end

3.3.1 View

Το τελευταίο κομμάτι της εφαρμογής, η μεριά του πελάτη (front end) είναι το σημείο εκείνο όπου τα δεδομένα μορφοποιούνται ώστε να είναι ευανάγνωστα στον τελικό χρήστη. Το σύνολο των εξαρτήσεων και των βιβλιοθηκών που χρειάστηκαν για την υλοποίηση του front end φαίνονται στα παρακάτω UML διαγράμματα. Στο πρώτο μπορεί κάποιος να διακρίνει την πολυπλοκότητα που έχει ένα framework στην προκειμένη περίπτωση το Angular για να φτάσει ο προγραμματιστής να εκθέσει τα δεδομένα του σε μια σελίδα στο διαδίκτυο, απεικονίζει το σύνολο των εξαρτήσεων που χρειάστηκαν μέσω του node.js για την υλοποίηση της εφαρμογής και το δεύτερο διάγραμμά δείχνει το σύνολο των δομοστοιχείων που δημιουργήθηκαν για την εφαρμογή.



Εικόνα 32, Angular library dependencies



Εικόνα 33, application modules dependencies

Ένας γενικός κανόνας είναι ότι για κάθε λειτουργία που έχει η εφαρμογή θα πρέπει να υπάρχει και το αντίστοιχο δομοστοιχείο που περιέχει την σελίδα html όπου είναι αυτό που βλέπει ο χρήστης, το αρχείο CSS που είναι για την μορφοποίηση των στοιχείων (χρώμα, μέγεθος γραμμάτων, θέση στην σελίδα) και το αρχείο .ts που είναι ο κώδικας ο οποίος εκτελείται στον πρόγραμμα περιήγησης (browser) του χρήστη. Επίσης στο UML διάγραμμα διακρίνονται και οι κλάσεις που είναι παρόμοιες με αυτές που έχουν δημιουργηθεί στην μεριά του server και είναι απαραίτητες για την κατανάλωση της πληροφορίας που έρχεται σε μορφή json, καθώς και οι υπηρεσίες services. Τα services μοιάζουν με τους controllers και είναι υπεύθυνα για την επικοινωνία μεταξύ του back end και του front end, επιπλέον είναι υπεύθυνα για τον καταμερισμό της πληροφορίας στα δομοστοιχεία της εφαρμογής.

```
private baseURL = 'http://localhost:8080/api/v1/lecturers';
constructor(private httpClient: HttpClient) { }

getLecturerList(): Observable<Lecturer[]>{
  return this.httpClient.get<Lecturer[]>( url: `${this.baseURL}`);
}
getLecturer(name: string): Observable<any>{
  return this.httpClient.get( url: `${this.baseURL}/${name}`);
}

createLecturer(lecturer: object): Observable<object>{
  return this.httpClient.post( url: `${this.baseURL}`, lecturer);
}

deleteLecturer(name: string): Observable<any>{
  return this.httpClient.delete( url: `${this.baseURL}/${name}`,  options: { responseType: 'text'});
}

updateLecturer(name: string, lecturer: Lecturer): Observable<Object>{
  return this.httpClient.put( url: `${this.baseURL}/${name}`, lecturer);
}
```

Εικόνα 34, τμήμα κώδικα ενός service

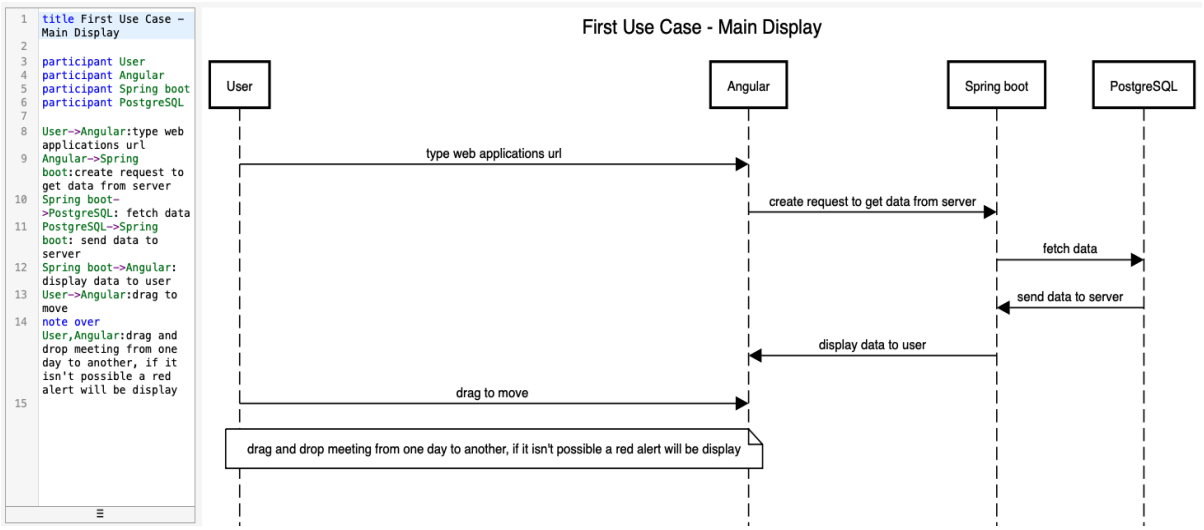
Ένα service έχει περίπου τις ίδιες λειτουργίες που έχει και ο controller και είναι το σημείο εκείνο όπου δηλώνεται πιο είναι το σημείο (inner URL) όπου η εφαρμογή επικοινωνεί με την μεριά του server. Ένα από τα σημεία που χρήζουν προσοχής είναι να μην εκτεθεί η συγκεκριμένη πληροφορία και αποκτήσουν πρόσβαση τρίτοι.

4.Περίπτωση χρήσης – Use Case

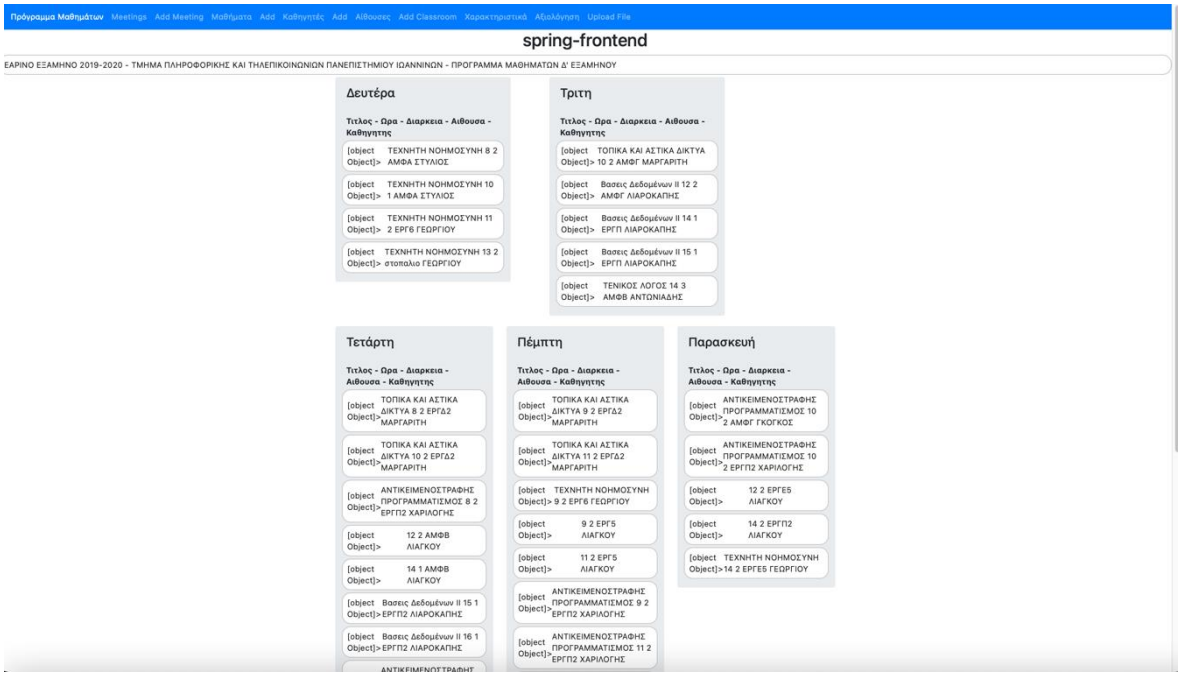
Τα sequence diagrams είναι ένα απαραίτητο εργαλείο για έναν προγραμματιστή. Είναι ο τρόπος με τον οποίο οπτικοποιείται ο κύκλος ζωής ενός αντικειμένου, από το αρχικό αίτημα του χρήστη, στην μεταφορά του στον server, στην άντληση πληροφορίας από την βάση δεδομένων και την επιστροφή του στο χρήστη. Τα παρακάτω διαγράμματα έχουν γίνει με την διαδικτυακή εφαρμογή *https://sequencediagram.org*

4.1 Πρώτο σενάριο

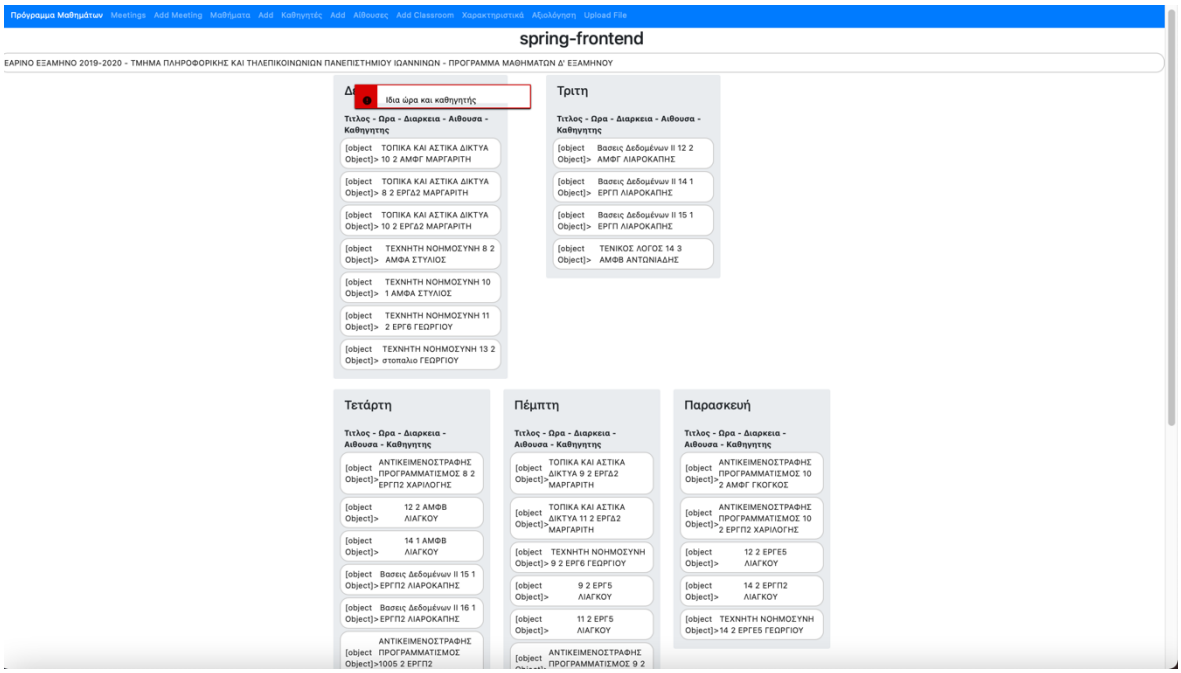
Ο χρήστης στέλνει ένα αίτημα στο URL της εφαρμογής. Το front end με τη σειρά του στέλνει ένα αίτημα get στο spring boot και αυτό με την σειρά του αντλεί τα δεδομένα από την βάση. Επιστρέφει τα δεδομένα στο angular και αυτό τα εμφανίζει στον χρήστη. Από εκεί και πέρα ο χρήστης έχει την δυνατότητα να πάρει τα δεδομένα από την μία ημέρα και με την χρήση drag and drop να τα μεταφέρει σε μία άλλη. Αν η επιλογή του παραβιάζει κάποιον περιορισμό τότε εμφανίζεται ένα μήνυμα που ενημερώνει τον χρήστη ότι αυτή η ενέργεια δεν γίνετε να πραγματοποιηθεί.



Εικόνα 35, sequence diagram σενάριο 1



Εικόνα 36, οθόνη πριν τη χρήση του drag and drop

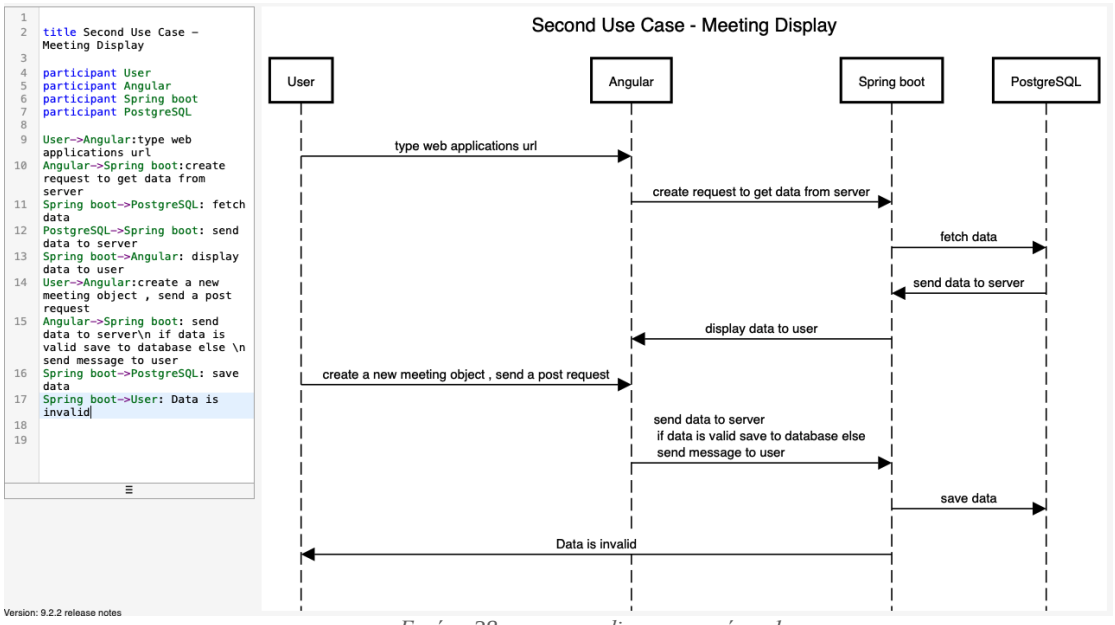


Εικόνα 37, οθόνη μετά τη χρήση του drag and drop

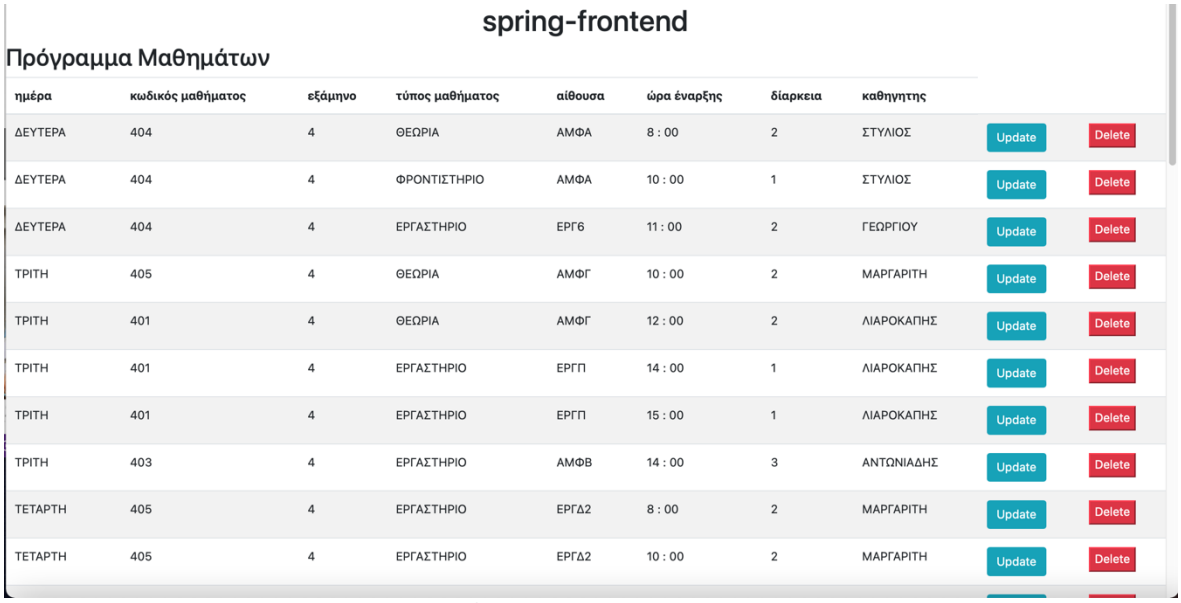
4.2 Σενάριο δεύτερο

Στο δεύτερο σενάριο ο χρήστης στέλνει ένα αίτημα για να εμφανίσει όλο το εβδομαδιαίο πρόγραμμα σε μία άλλη οθόνη, στο τέλος κάθε γραμμής υπάρχουν δύο κουμπιά , το ένα είναι για να γίνει τροποποίηση και το άλλο για να γίνει διαγραφεί. Μόλις τελειώσει με την ενέργεια που θέλει στέλνει τα δεδομένα στον server και αν είναι σωστά

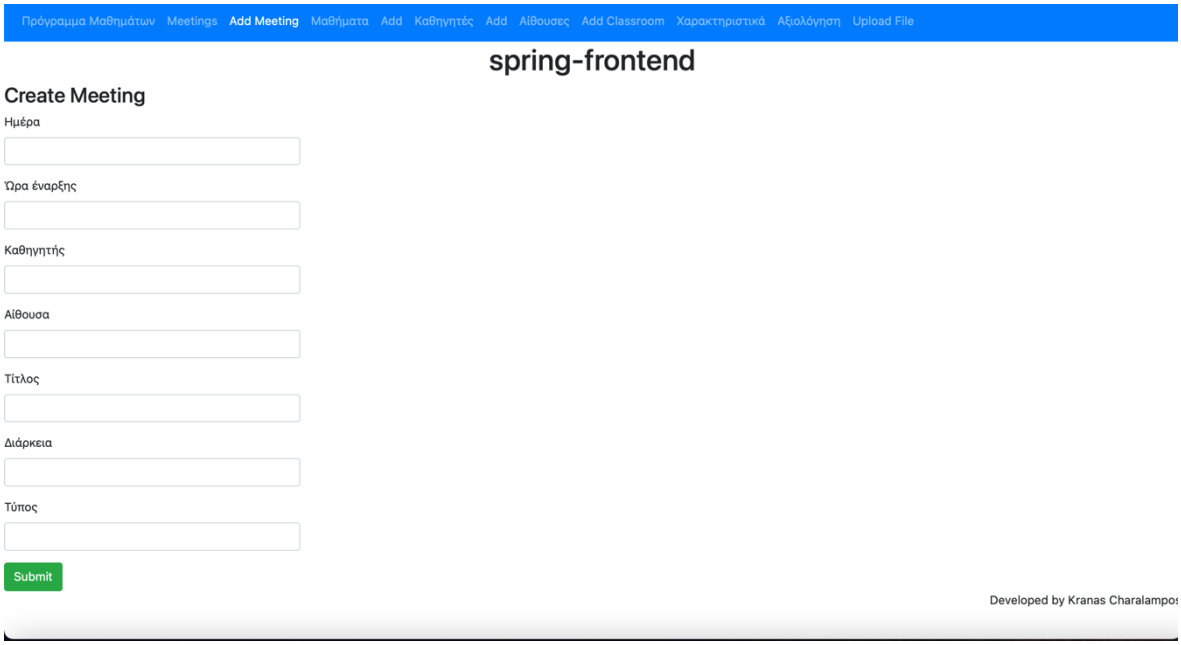
αποθηκεύονται στην βάση, αλλιώς ο χρήστης λαμβάνει ένα μήνυμα σφάλματος. Παρόμοιες ενέργειες υπάρχουν και στις υπόλοιπες οθόνες (καθηγητές, αίθουσες, μαθήματα).



Εικόνα 38, sequence diagram σενάριο 1



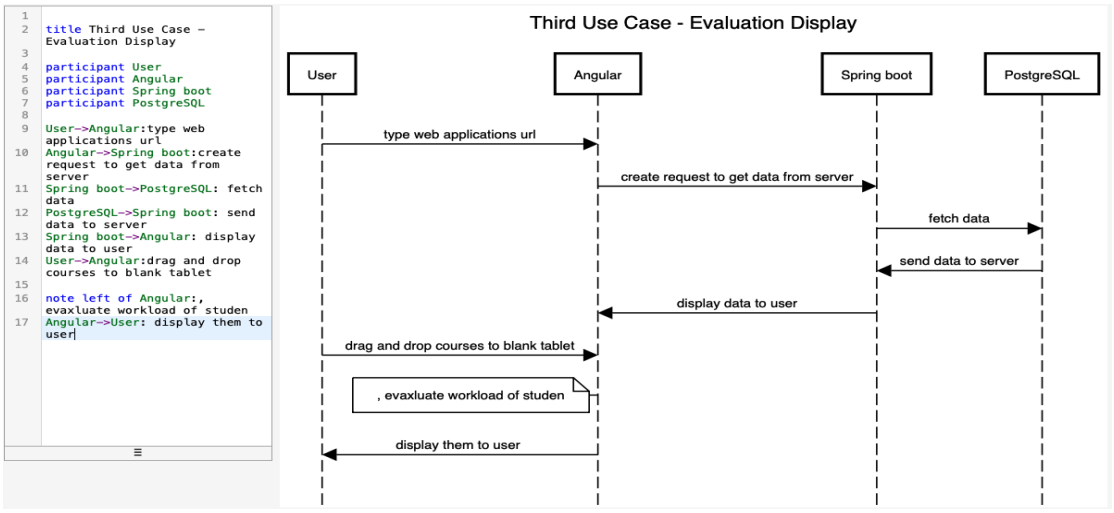
Εικόνα 39, οθόνη για την τροποποίηση συνατήσεων



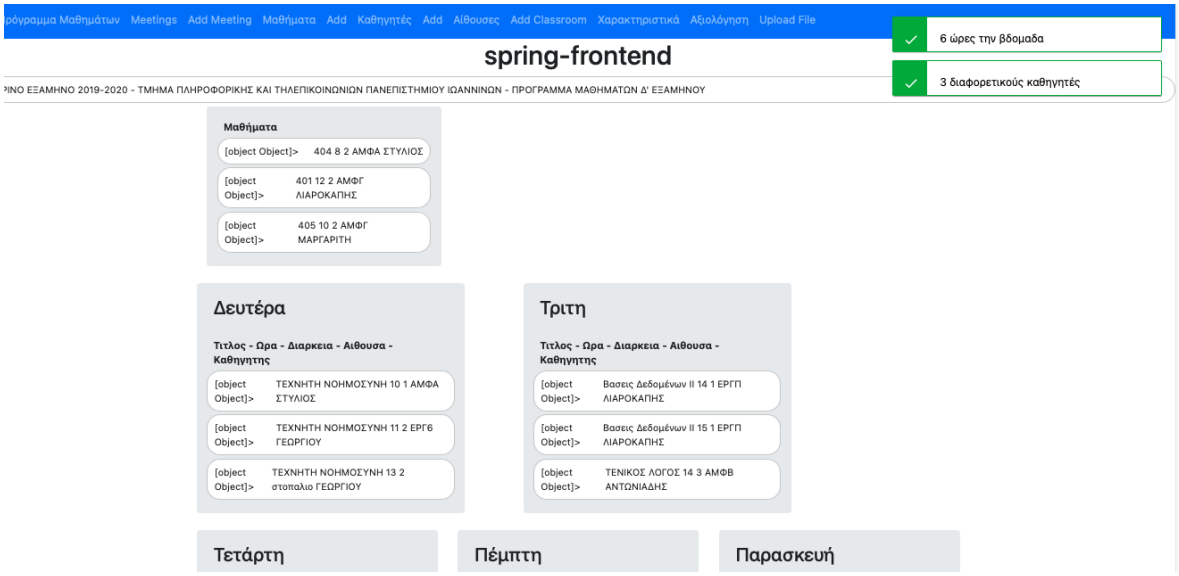
Εικόνα 40, εισαγωγή καινούργιας συνάντησης

4.3 Σενάριο τρίτο

Στο τρίτο σενάριο ο χρήστης στέλνει ένα αίτημα για την οθόνη της αξιολόγησης. Όπως και στα προηγούμενα σενάρια ακολουθείται μια σειρά βημάτων μέχρι να εμφανιστούν τα δεδομένα στην οθόνη του χρήστη που είναι παρόμοια με την πρώτη οθόνη. Η διαφορά είναι ότι σε αυτή την οθόνη υπάρχει μια έξτρα θέση για τα μαθήματα όπου ο χρήστης με drag and drop μπορεί να τα μετακινήσει εκεί και στο τέλος να υπολογίσει το φόρτο εργασίας που έχει ένας φοιτητής ανά βδομάδα για το τρέχον εξάμηνο.



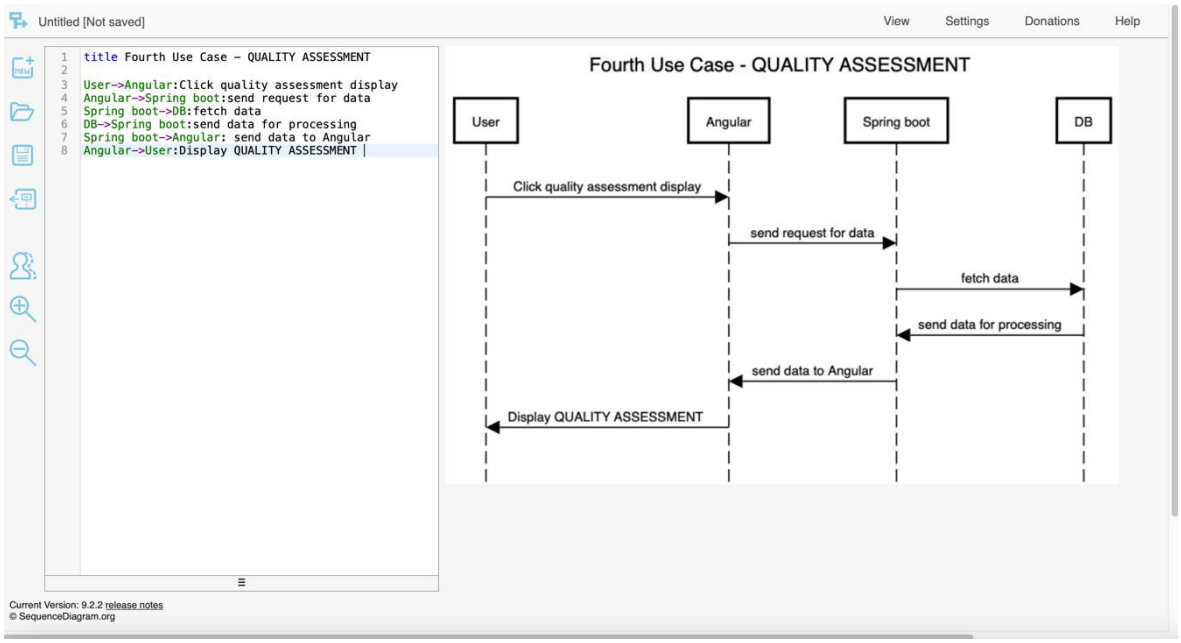
Εικόνα 41, sequence diagram σενάριο 3



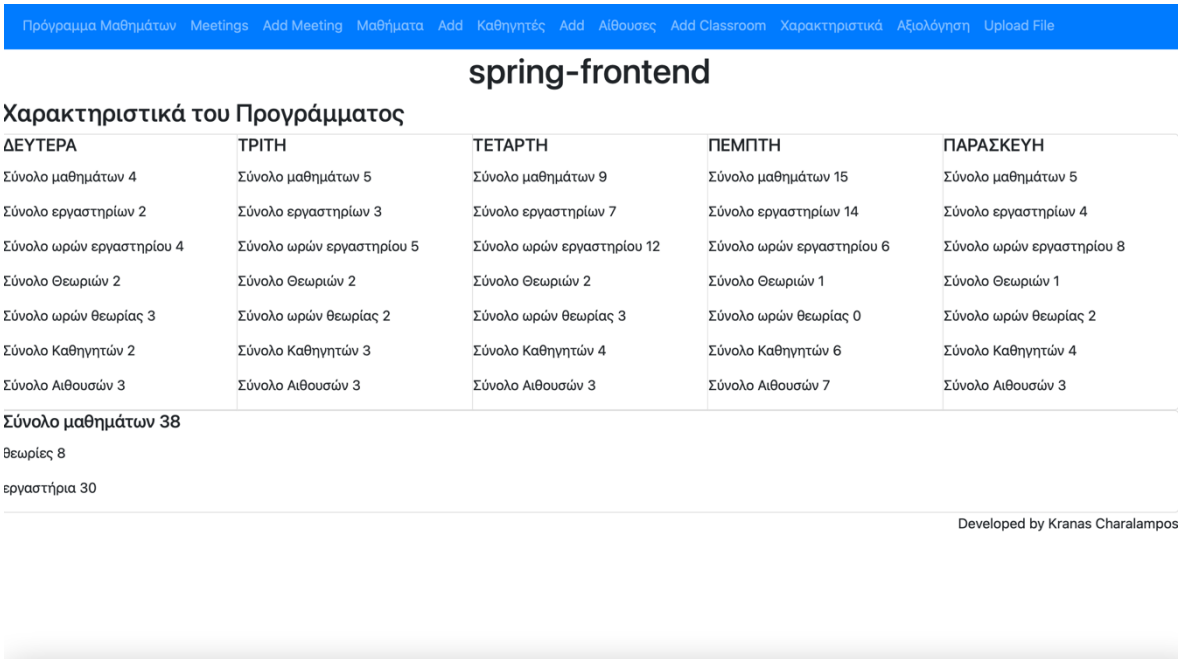
Εικόνα 42, οθόνη αξιολόγησης και υπολογισμού

4.4 Σενάριο τέταρτο

Στο τέταρτο σενάριο ο χρήστης στέλνει ένα αίτημα ώστε να δει τα χαρακτηριστικά του εβδομαδιαίου προγράμματος.



Εικόνα 43,sequence diagram σενάριο 4



Εικόνα 44, οθόνη υπολογισμού χαρακτηριστικών του προγράμματος

5.Συμπεράσματα

Στη παρούσα διπλωματική εργασία παρουσιάστηκε το πρόβλημα του χρονοπρογραμματισμού του εβδομαδιαίου προγράμματος ενός πανεπιστημιακού ιδρύματος διαχρονικά, οι κατηγορίες στις οποίες χωρίζεται και μερικές από τις προτάσεις που έχουν γίνει μέχρι σήμερα για την λύση του, όπως ο χρωματισμός γραφήματος, η αναζήτηση tabu, ο ακέραιος και γραμμικός προγραμματισμός, η περιορισμοί που μπορεί να έχει το πρόβλημα. Παρουσιάστηκαν επίσης και αναλυθήκαν όλα τα εργαλεία που θα χρησιμοποιήσει ένας προγραμματιστής για την διεκπεραίωση της εργασίας του, όπως, το IntelliJ IDEA που είναι ολοκληρωμένο περιβάλλον ανάπτυξης λογισμικού, τις ευκολίες που προσφέρει σε έναν προγραμματιστή και πως να το χρησιμοποιήσει ώστε να είναι αποδοτικός. Έγινε εκτενή ανάλυση για το τι είναι ένα σύστημα ελέγχου εκδόσεων όπως είναι το Git και ποιος είναι ο ρόλος του στην διαδικασία ανάπτυξης ενός πρότζεκτ καθώς επίσης, ποιες είναι οι βασικές εντολές και ποια η χρήση τους. Παρουσιάστηκε η βάση δεδομένων PostgreSQL και τα πλεονεκτήματα της έναντι άλλων σχεσιακών βάσεων δεδομένων. Το framework Spring Boot που χρησιμοποιείται ευρέως για την συγγραφή κώδικα στην μεριά του εξυπηρετητή και το πόσο εύκολο είναι στην χρήση του και την δυνατότητα που δίνει στον προγραμματιστή μέσω επισημάνσεων να επικοινωνήσει τόσο με την βάση δεδομένων για να εκτελέσει τις βασικές λειτουργίες εισαγωγή, διάβασμα, επεξεργασία και διαγραφή δεδομένων, όσο και με την μεριά του front end για να εκθέσει τα δεδομένα αυτά. Το εργαλείο maven που χρησιμοποιείται για την εγκατάσταση βιβλιοθηκών στο περιβάλλον του προγραμματιστή και το πόσο γρήγορα και εύκολα γίνεται. Το Hibernate που είναι ένα ORM και μετατρέπει τις κλάσεις java σε οντότητες στην βάση δεδομένων και πόσο εύκολη γίνετε η συγγραφή ερωτώ-απαντήσεων. Τέλος αναλύθηκε το front end framework Angular και πως ένα ολόκληρο πρότζεκτ χωρίζεται σε components, το πως η επικοινωνία με το back end της εφαρμογής πλέον έχει γίνει εύκολη απλά με την προσθήκη μιας βιβλιοθήκης όπως είναι ο http client και πως η σελίδα παρουσίασης παίρνει ζωή με τη χρήση βιβλιοθηκών σαν αυτή της drag and drop.

5.1 Προτάσεις για μελλοντική έρευνα

Οι προτάσεις για μελλοντική έρευνα μετά την ολοκλήρωση διπλωματικής εργασίας συνοψίζονται στις εξής:

1. Ολοκλήρωση της εφαρμογής ώστε να είναι έτοιμη για περιβάλλον παραγωγής.
2. Διαμερισμός σε containers.

   Τι είναι το docker και πως μπορεί να χρησιμοποιηθεί?
3. Ενσωμάτωση Testing Frameworks.
   a. Back end : JUnit, JBehave
   b. Front end: Siesta, Jasmine
4. Λύσεις που θα κάλυπταν όσο το δυνατόν περισσότερα πανεπιστημιακά ιδρύματα.
5. Αυτόματη εισαγωγή δεδομένων μετά από το ανέβασμα του αρχείου.

Βιβλιογραφία

©2010-2021., S.-p. b. (χ.χ.). *https://angular.io/guide/*. Ανάκτηση από https://angular.io/guide/what-is-angular

Abramson, D. (1991). Constructing School Timetables using Simulated Annealing: Sequential and Parallel Algorithms . Στο D. Abramson. Melbourne .

Akhan Akbulut, G. Y. (χ.χ.). *University Exam Scheduling System Using Graph Coloring Algorithm and RFID Technology*.

Bambrick, L. (1997). *Lecture Timetabling Using Genetic Algorithms.* St Lucia: Undergraduate Thesis Bachelor of Computer Systems Engineering.

Chacon, S. (2021). *http://git-scm.com/*. Ανάκτηση από http://git-scm.com/docs

Coston, P. J. (1992). Applying constraints to the timetable problem.

E.K.Burke, D. R. (χ.χ.). A University Timetabling System based on Graph Colouring and Constraint Manipulation.

Foundation, T. A. (2002-2021). *https://maven.apache.org*. Ανάκτηση από https://maven.apache.org/what-is-maven.html

GLOVER, F. (1986). «Future Paths for Integer Programming and Links to Artificial Intelligence (1986)».

hat, R. (χ.χ.). *https://hibernate.org/*. Ανάκτηση από https://hibernate.org/orm/documentation/5.5/

Izundu Kingsley, I. K. (2018). A Tabu Search-Based University Lectures Timetable Scheduling Model. *International Journal of Computer Applications*.

JetBrains. (χ.χ.). *https://www.jetbrains.com*. Ανάκτηση από /help/idea/discover-intellij-idea.html

Leighton, F. T. (1979). A Graph Coloring Algorithm for Large Scheduling Problems.

M. A. Saleh Elmohamed, P. C. (1998). A Comparison of Annealing Techniques for Academic Course Scheduling.

Nurlida Basir, W. I. (2013). A Simulated Annealing for Tahmidi Course Timetabling.

Pivotal. (2012-2021). *https://docs.spring.io/*. Ανάκτηση από https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/

postgresql. (2021). *postgresql*. Ανάκτηση από https://www.postgresql.org/about/

Prabodanie, R. A. (2016). An Integer Programming Model for a Complex University Timetabling Problem: A Case Study. Kuliyapitiya, Sri Lanka: Department of Industrial Management, Faculty of Applied Sciences, Wayamba University of Sri Lanka,.

Rochkind, M. J. (1975). *The source code control system.* IEEE TRANSACTIONS ON SOFTWARE ENGINEERING.

Roy, R. G. (2017). Study on Course Timetable Scheduling using Graph Coloring Approach . *International Journal of Computational and Applied Mathematics.*, 469-485.

Tomasz Dobrowolski, D. D. (χ.χ.). Koala graph coloring library: An open graph coloring library for real-world applications. *2008 1st International Conference on Information Technology*. IEEE.

Toth, A. B. (2015). An overview of curriculum-based course timetabling. Sociedad de Estadística e Investigación Operativa 2015.