



ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΜΑΤΙΚΗΣ

Τίτλος Εργασίας

**ΥΛΟΠΟΙΗΣΗ ΔΙΑΔΙΚΤΥΑΚΗΣ ΕΦΑΡΜΟΓΗΣ ΗΜΕΡΟΛΟΓΙΟΥ ΜΕ
ΔΥΝΑΤΟΤΗΤΑ ΑΥΤΟΜΑΤΟΥ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ**

Πτυχιακή εργασία

Όνομα φοιτητή

ΧΑΡΑ ΜΠΟΥΛΟΥΓΑΡΗ

Αθήνα, Ιούνιος 2019



ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΜΑΤΙΚΗΣ

Τριμελής Εξεταστική Επιτροπή

**Τσερπές Κωνσταντίνος, Επίκουρος Καθηγητής,
Τμήμα Πληροφορικής & Τηλεματικής,
Χαροκόπειο Πανεπιστήμιο**

**Βασιλοπούλου Κωνσταντίνα, Λέκτορας,
Τμήμα Πληροφορικής & Τηλεματικής,
Χαροκόπειο Πανεπιστήμιο**

**Σοφianoπούλου Χρύσα, Επίκουρη Καθηγήτρια,
Τμήμα Πληροφορικής & Τηλεματικής,
Χαροκόπειο Πανεπιστήμιο**

Η Μπουλούγαρη Χαρά,

δηλώνω υπεύθυνα ότι:

- 1) Είμαι ο κάτοχος των πνευματικών δικαιωμάτων της πρωτότυπης αυτής εργασίας και από όσο γνωρίζω η εργασία μου δε συκοφαντεί πρόσωπα, ούτε προσβάλλει τα πνευματικά δικαιώματα τρίτων.
- 2) Αποδέχομαι ότι η ΒΚΠ μπορεί, χωρίς να αλλάξει το περιεχόμενο της εργασίας μου, να τη διαθέσει σε ηλεκτρονική μορφή μέσα από τη ψηφιακή Βιβλιοθήκη της, να την αντιγράψει σε οποιοδήποτε μέσο ή/και σε οποιοδήποτε μορφότυπο καθώς και να κρατά περισσότερα από ένα αντίγραφα για λόγους συντήρησης και ασφάλειας.

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα πτυχιακή εργασία υλοποιήθηκε στα πλαίσια του προγράμματος προπτυχιακών σπουδών του τμήματος Πληροφορικής και Τηλεματικής στο Χαροκόπειο Πανεπιστήμιο Αθηνών από τον Νοέμβριο του 2018 έως και τον Απρίλη του 2019.

Θα ήθελα να ευχαριστήσω όλους όσους με υποστήριξαν καθ' όλη τη διάρκεια αυτής της εργασίας και κατ' επέκταση και των σπουδών μου.

Πιο συγκεκριμένα θα ήθελα να ευχαριστήσω τον κ. Κωνσταντίνο Τσερπέ, επίκουρο καθηγητή στο Χαροκόπειο Πανεπιστήμιο, που υπήρξε ο επιβλέπων καθηγητής της παρούσας πτυχιακής για όλη την καθοδήγηση, βοήθεια και υποστήριξη που μου παρείχε.

Επίσης, την οικογένειά μου, τον παππού μου και τη γιαγιά μου για όλη τη βοήθεια που μου έδωσαν για όλα τα χρόνια σπουδών μου, μαθητικών και ακαδημαϊκών...

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Περίληψη στα Ελληνικά.....	7
Περίληψη στα Αγγλικά.....	8
Κατάλογος Εικόνων.....	9
Κατάλογος Αποσπασμάτων Κώδικα.....	10
Συντομογραφίες.....	11
1 ΕΙΣΑΓΩΓΗ.....	12
1.1 Η τεχνολογία στην καθημερινή ζωή του ανθρώπου.....	12
1.2 Αντικείμενο πτυχιακής.....	12
1.2.1 Ορισμός προβλήματος.....	12
1.2.2 Τρόπος επίλυσης προβλήματος.....	13
1.2.3 Σύντομη περιγραφή λογικής.....	14
1.3 Οργάνωση κειμένου.....	15
2 ΑΝΑΣΚΟΠΗΣΗ ΤΡΕΧΟΥΣΑΣ ΚΑΤΑΣΤΑΣΗΣ (State of the Art).....	16
2.1 Παρόμοιες εφαρμογές στην αγορά.....	16
2.1.1 Εισαγωγή.....	16
2.1.2 Μια σύντομη γνωριμία με τις εφαρμογές της αγοράς.....	16
2.1.2.1 Τρόποι επίλυσης του προβλήματος της διαχείρισης χρόνου και γεγονότων από ήδη υπάρχουσες εφαρμογές στην αγορά.....	17
2.1.2.1.1 Τεχνικές με τις οποίες το Google Calendar επιτυγχάνει διαχείριση χρόνου και γεγονότων.....	17
2.1.2.1.2 Τεχνικές με τις οποίες το Outlook Calendar επιτυγχάνει διαχείριση χρόνου και γεγονότων.....	17
2.1.2.2 Παρατηρήσεις που προκύπτουν από τον τρόπο προσέγγισης του αυτόματου προγραμματισμού γεγονότων από τις υπάρχουσες εφαρμογές.....	18
2.1.3 Τρόπος επίλυσης με την προσθήκη επιπλέον χαρακτηριστικών λειτουργικότητας.....	18
2.2 Τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση της διαδικτυακής εφαρμογής και λογισμικά.....	20
2.2.1 Node.....	20
2.2.1.1 Παραδοσιακοί εξυπηρετητές που χρησιμοποιούν μια διεργασία ανά request.....	21
2.2.1.2 Παραδοσιακοί εξυπηρετητές που χρησιμοποιούν Thread Pool.....	21
2.2.1.3 Γιατί NodeJS ?.....	22
2.2.2 Npm.....	24
2.2.3 Angular.....	24
2.2.3.1 Γιατί Angular ?.....	24
2.2.3.2 Αρχιτεκτονική της Angular.....	25
2.2.4 Express.....	26
2.2.5 MySQL.....	28
2.2.6 Xampp.....	29
2.2.7 Visual Studio Code.....	29
2.2.8 Postman.....	30
2.2.9 Google Chrome.....	30
3 ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΥΣΤΗΜΑΤΟΣ.....	31
3.1 Εισαγωγή.....	31
3.2 Συνδεσμολογία.....	31
3.2.1 UML Component model Diagram.....	31
3.2.2 Επεξήγηση διαγράμματος.....	33
3.2.3 Συσχέτιση συστατικών συνδεσμολογίας με απαιτήσεις συστήματος.....	34
3.2.3.1 Τι είναι ένα use case diagram με λίγα λόγια ?.....	34
3.2.3.2 Συσχέτιση των δύο παραπάνω διαγραμμάτων (απαιτήσεων-συστατικών).....	35

ΥΛΟΠΟΙΗΣΗ ΔΙΑΔΙΚΤΥΑΚΗΣ ΕΦΑΡΜΟΓΗΣ ΗΜΕΡΟΛΟΓΙΟΥ ΜΕ ΔΥΝΑΤΟΤΗΤΑ ΑΥΤΟΜΑΤΟΥ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

3.2.3.2.1	Λίγα λόγια για τις απαιτήσεις που δεν σχετίζονται με κάποιο συστατικό της εφαρμογής.....	36
3.3	Λειτουργικότητα κάθε συστατικού και η επικοινωνία του με τα υπόλοιπα.....	38
3.3.1	Μια σύντομη γνωριμία με τα Observables και τη βιβλιοθήκη Rxjs.....	42
4	ΛΕΠΤΟΜΕΡΕΙΕΣ ΣΥΣΤΗΜΑΤΟΣ.....	44
4.1	Εισαγωγή.....	44
4.2	Εγκατάσταση απαραίτητων τεχνολογιών και εργαλείων.....	44
4.2.1	Εγκατάσταση NodeJS	44
4.2.2	Εγκατάσταση Visual Studio.....	45
4.2.2.1	Εξοικείωση και γνωριμία με τον editor.....	45
4.2.3	Εγκατάσταση XAMPP.....	48
4.2.4	Εγκατάσταση Postman.....	48
4.3	Δημιουργία νέου Project.....	49
4.4	Περιγραφή Αλγορίθμων που χρησιμοποιήθηκαν και προγραμματιστικών μοντέλων που ακολουθήθηκαν.....	52
4.4.1	Ο αλγόριθμος της φυσαλίδας.....	52
4.4.2	Το μοντέλο MVC.....	54
4.4.2.1	Μια γνωριμία με το μοντέλο MVC.....	54
4.4.2.2	Η εφαρμογή του μοντέλου MVC στην εφαρμογή που υλοποιήθηκε.....	56
4.4.3	Αρχιτεκτονική 3-Tier.....	57
4.4.3.1	Η εφαρμογή της αρχιτεκτονικής αυτής στην εφαρμογή ημερολογίου.....	58
4.5	Σημεία κώδικα και λογικής που αξίζουν να αναφερθούν.....	58
4.5.1	Η λογική της ταξινόμησης (με βήματα).....	59
4.5.2	Κομμάτια κώδικα ταξινόμησης με εικόνες και σχόλια.....	60
4.5.3	Σερβίροντας το frontend κάνοντας κλήση στο backend.....	64
5	ΑΞΙΟΛΟΓΗΣΗ ΣΥΣΤΗΜΑΤΟΣ.....	65
5.1	Εισαγωγή.....	65
5.2	Σενάρια Εκτέλεσης.....	65
5.2.1	Αρχικές συνθήκες και στόχοι σεναρίων εκτέλεσης.....	65
5.2.2	Ικανοποίηση απαιτήσεων και αντικειμενικών στόχων από το σύστημα.....	68
5.2.3	Σύνοψη ικανοποίησης αντικειμενικών στόχων (checklist).....	69
6	ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΗΣ ΣΕ ΥΠΟΔΟΜΕΣ ΥΠΟΛΟΓΙΣΤΙΚΟΥ ΝΕΦΟΥΣ.....	70
6.1	Εισαγωγή.....	70
6.2	Τεχνολογίες που θα χρειαστούν και εργαλεία.....	70
6.2.1	Εξοικείωση με τις έννοιες docker container και docker image.....	74
6.2.2	Τι είναι το docker hub.....	74
6.2.3	Τι είναι το azure.....	75
6.2.4	Η λογική της αρχιτεκτονικής της φιλοξενίας της εφαρμογής στο cloud.....	76
6.3	Εγκατάσταση επεκτάσεων (azure & docker) στο vs code και δημιουργία repository στο docker hub.....	78
6.4	Δημιουργία dockerFile και docker-Compose.yml και ανέβασμα στο azure web app for containers.....	79
7	ΣΥΜΠΕΡΑΣΜΑΤΑ.....	85
7.1	Σύνοψη της όλης υλοποίησης και αποτελέσματα επιτυχίας.....	85
7.2	Μελλοντική αναβάθμιση της εφαρμογής (επιπλέον λειτουργικότητα).....	86
7.3	Συνεισφορά στο ευρύ κοινό	86
	ΒΙΒΛΙΟΓΡΑΦΙΑ.....	88

ΠΕΡΙΛΗΨΗ

Αντικείμενο της παρούσας εργασίας είναι η ανάπτυξη μιας διαδικτυακής NodeJS Single Page Application (SPA) εφαρμογής ημερολογίου με σκοπό την οργάνωση όλων των καθημερινών δραστηριοτήτων ενός ατόμου με βάση την ώρα τη διάρκεια και τη σπουδαιότητα τους, ώστε να διεκπεραιωθούν με τη συνιστώμενη από την εφαρμογή σειρά. Η υλοποίηση της εφαρμογής είχε δύο στάδια τα οποία έγιναν με χρήση των τεχνολογιών Angular (ένα framework ανάπτυξης web εφαρμογών που τηρεί το μοντέλο M.V.C.) για το πρώτο στάδιο του frontend και Express (ένα ελαφρύ framework διαδικτυακών εφαρμογών για την οργάνωση της εφαρμογής αρχιτεκτονικής τύπου M.V.C. στη μεριά του εξυπηρετητή) για το στάδιο του backend. Επίσης, χρησιμοποιήθηκε μια βάση δεδομένων MySQL για την αποθήκευση των δεδομένων.

Η υλοποίηση μιας τέτοιας εφαρμογής στοχεύει στην εξοικονόμηση χρόνου και στην οργάνωση όλων των υποχρεωτικών αλλά και μη δραστηριοτήτων κάθε απλού χρήστη για την πιο αποτελεσματική χρονική διεκπεραίωση τους κατά τη διάρκεια της μέρας και των γρήγορων και απαιτητικών ρυθμών ζωής. Πιο αναλυτικά, η εφαρμογή δέχεται μέσω μιας φόρμας συμπλήρωσης δεδομένα που χαρακτηρίζουν μια δραστηριότητα και στη συνέχεια γίνεται η καταχώρησή τους σε μια βάση δεδομένων. Για την ίδια μέρα ο χρήστης είναι ελεύθερος να καταχωρήσει όσες δραστηριότητες έχει η μέρα του και οι οποίες θα εμφανίζονται με οργανωμένη σειρά μετά σε έναν πίνακα αφού ο χρήστης επιλέξει να δει τις όποιες δραστηριότητες έχει καταχωρήσει για αυτήν την μέρα. Η σειρά ταξινόμησης έχει οριστεί να γίνει με βάση την ώρα αλλά και με βάση τη διάρκεια, για παράδειγμα μια δραστηριότητα που έχει προγραμματιστεί να γίνει σε έναν συγκεκριμένο χρόνο και κρατάει τόση διάρκεια ώστε να ξεπερνά το χρόνο έναρξης της επόμενης δραστηριότητας ταξινομείται με βάση τη λογική του αλγόριθμου της φουσαλίδας μία θέση πιο πάνω έως ότου δεν ξεπερνά η ώρα έναρξης της σε άθροισμα με τη διάρκειά της την ώρα έναρξης της επόμενης δραστηριότητας. Ο χρήστης είναι σε θέση να τροποποιήσει τα δεδομένα που καταχώρησε όποτε το θελήσει ή ακόμα και να τα διαγράψει όλα ή κάποια από αυτά. Στη δεξιά μεριά της εφαρμογής υπάρχει ένα ημερολόγιο το οποίο σημειώνει με μια κουκίδα τις ημερομηνίες στις οποίες υπάρχουν καταχωρημένες δραστηριότητες για να τις υπενθυμίζει στον χρήστη.

Λέξεις Κλειδιά: Angular Client Side Framework, MVC, Calendar NodeJS SPA Web Application, Express Server Side Framework, MySQL DataBase

ABSTRACT

The subject of this dissertation is the development of a NodeJS Single Page Application (SPA) Calendar Web Application which aims to organize the daily activities of an individual based on their scheduled time, duration and priority so they are fulfilled in the recommended by the application order. The development of the application had two stages which were developed by using the following technologies: Angular (a web application framework based on the M.V.C. model) for the first stage of the frontend development and Express (*a light-weight web application framework to help organize your web application into an M.V.C. architecture on the server side*) for the backend stage. Furthermore, a MySQL database was used for data storage.

The development of such an application focuses not only on time-saving in favor of an individual but also on the most effective way to put in order all the necessary and non-necessary activities waiting for fulfillment in today's fast pace of life. In detail, the application accepts some data which describe an activity through a form and then the form submits them in a database. The user is free to submit more than one activity for the same day. All of them will be displayed inside a table in a priority order after the user selects a day to display all the submitted activities for that day. The priority order is determined to be done according to time and duration time, for instance an activity scheduled for a specific time with duration time enough to surpass the beginning time of the next activity is sorted according to the logic of the bubble algorithm a place above until its beginning time plus its duration time doesn't surpass the beginning time of the next activity. The user is allowed to amend or delete all or some of any of the data that he has submitted at any time. On the right side of the application's screen there is a calendar view which prints a dot next to a day if there are activities submitted for that day. This dot acts as a reminder to the user about his submissions.

Keywords: Angular Client Side Framework, MVC, Calendar NodeJS SPA Web Application, Express Server Side Framework, MySQL DataBase

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

Εικ.1. πρώτος παραδοσιακός τρόπος λειτουργίας ενός εξυπηρετητή.....	σ.21
Εικ.2. δεύτερος παραδοσιακός τρόπος λειτουργίας ενός εξυπηρετητή.....	σ.22
Εικ.3. Η λογική του JavaScript.....	σ.23
Εικ.4. Component Diagram.....	σ.31
Εικ.5. Επεξήγηση συμβόλων Component διαγράμματος.....	σ.32
Εικ.6. Use Case Diagram.....	σ.34
Εικ.7. Στυλ Αλληλεπίδρασης με το χρήστη.....	σ.38
Εικ.8. Γνωριμία με την εργαλειοθήκη του editor.....	σ.45
Εικ.9. Αρχείο app.module.ts.....	σ.47
Εικ.10. Οι αλληλεπιδράσεις σε ένα αρχιτεκτονικό μοντέλο MVC.....	σ.54
Εικ.11. 3-Tier Αρχιτεκτονική.....	σ.58
Εικ.12. Διόρθωση κενού του κώδικα που βρέθηκε μετά από το 1 ^ο σενάριο εκτέλεσης.....	σ.66
Εικ.13. Έλεγχος 1 σενάριο εκτέλεσης 2 ^ο	σ.66
Εικ.14. Έλεγχος 2 σενάριο εκτέλεσης 2 ^ο	σ.67
Εικ.15. Έλεγχος 3 σενάριο εκτέλεσης 2 ^ο	σ.67
Εικ.16. Έλεγχος 4 σενάριο εκτέλεσης 2 ^ο	σ.67
Εικ.17. Έλεγχος 5 σενάριο εκτέλεσης 2 ^ο	σ.67
Εικ.18. Έλεγχος 6 σενάριο εκτέλεσης 2 ^ο	σ.67
Εικ.19. Έλεγχος 7 σενάριο εκτέλεσης 2 ^ο	σ.67
Εικ.20. Έλεγχος 8 σενάριο εκτέλεσης 2 ^ο	σ.68
Εικ.21. Root project folder.....	σ.72
Εικ.22. Αρχιτεκτονική φιλοξενίας της εφαρμογής στο cloud.....	σ.76
Εικ.23. Azure Portal.....	σ.78
Εικ.24. Docker-Compose.yml με κώδικα σύνθεσης πολλών containers.....	σ.81

ΚΑΤΑΛΟΓΟΣ ΑΠΟΣΠΑΣΜΑΤΩΝ ΚΩΔΙΚΑ

Απόσπασμα 1: Τιμή επιλογής confirmation dialog	σ.41
Απόσπασμα 2: Ταξινόμηση μέρος1.....	σ.60
Απόσπασμα 3: Ταξινόμηση μέρος2.....	σ.61
Απόσπασμα 4: Ταξινόμηση μέρος3.....	σ.61
Απόσπασμα 5: Ταξινόμηση μέρος4.....	σ.61
Απόσπασμα 6: Ταξινόμηση μέρος5.....	σ.62
Απόσπασμα 7: Ταξινόμηση μέρος6.....	σ.62
Απόσπασμα 8: Ταξινόμηση μέρος7.....	σ.62
Απόσπασμα 9: Ταξινόμηση μέρος8.....	σ.63
Απόσπασμα 10: Ταξινόμηση μέρος9.....	σ.63
Απόσπασμα 11: Ταξινόμηση μέρος10.....	σ.63
Απόσπασμα 12: Εκκίνηση frontend μέσω backend μέρος 1.....	σ.64
Απόσπασμα 13: Εκκίνηση frontend μέσω backend μέρος 2.....	σ.64
Απόσπασμα 14: Εκκίνηση frontend μέσω backend μέρος 3.....	σ.64
Απόσπασμα 15: DockerFile.....	σ.80
Απόσπασμα 16: Docker-Compose.yml.....	σ.81

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

SPA	Single Page Application
MVC	Model View Controller
HTTP	Hypertext Transfer Protocol
APP	Application
iOS	iPhone Operating System
CPU	Central Processing Unit
RAM	Random Access Memory
OS	Operating System
I/O	Input/Output
CLI	Command Line Interface
NPM	Node Package Manager
DOM	Document Object Model
HTML5	HyperText Markup Language
IBM	International Business Machines Corporation
UML	Unified Modeling Language
API	Application Programming Interface
OBJ	Object
CRUD	Create, Read, Update, and Delete operations
CSS	Cascading Style Sheets
RXJS	Reactive Extensions for JavaScript
.ts	typescript file
URL	Uniform Resource Locator
JSON	JavaScript Object Notation
AJAX	Asynchronous JavaScript And XML
IP	Internet Protocol address
UI	User interface
PaaS	Platform as a Service
SSL	Secure Sockets Layer
TLS	Transport Layer Security
DNS	Domain Name System

1

ΕΙΣΑΓΩΓΗ



1.1 Η τεχνολογία στην καθημερινή ζωή του ανθρώπου

Η τεχνολογία στις μέρες μας αποτελεί ένα αναπόσπαστο κομμάτι της καθημερινής μας ζωής αλλά και του εαυτού μας. Τα επιτεύγματά της είναι φανερά και πληθώρα σε πάρα πολλούς τομείς της καθημερινής, επαγγελματικής αλλά και κοινωνικής ζωής του ανθρώπου, διευκολύνοντάς τον έτσι σε μεγάλο βαθμό και βρίσκοντας λύση σε χιλιάδες προβλήματα σε ελάχιστο χρόνο. Οι τομείς στους οποίους αυτή βρίσκει εφαρμογή μπορεί να είναι τόσο σε εξειδικευμένο επίπεδο όπως στην ιατρική, στην φυσική, στην χημεία, στην μετεωρολογία, στην ρομποτική, στην αστροφυσική, στα μαθηματικά, και σε όλες τις υπόλοιπες φυσικές επιστήμες αλλά και επιστήμες υγείας όσο και σε ποιο βιομηχανικό επίπεδο όπως σε εργοστάσια παραγωγής πρώτων υλών, αγαθών και τροφίμων, στον αγροτικό, κτηνοτροφικό τομέα, στις μεταφορές αεροπορικές και αστικές και τέλος ακόμα και σε πιο χαμηλό επίπεδο όπως στην καθημερινή ζωή του ανθρώπου μέσω της επικοινωνίας με τα μέσα κοινωνικής δικτύωσης, της εκπαίδευσης μέσω του e-learning, της διασκέδασης και ψυχαγωγίας μέσω των πολυμέσων, αλλά και των οικιακών δραστηριοτήτων μέσω των διαφόρων συσκευών που κάνουν τη ζωή όλο και πιο εύκολη. Όλα δείχνουν πως η τεχνολογία είναι όντως τελικά συνεχώς γύρω μας από τις πιο απλές έως τις πιο εξειδικευμένες πτυχές της ζωής μας λύνοντας τα χέρια και εξοικονομώντας έτσι χρόνο χωρίς κόπο σε κοινωνικό, περιβαλλοντολογικό και οικονομικό επίπεδο.

1.2 Αντικείμενο πτυχιακής

Αντικείμενο της παρούσας εργασίας είναι η ανάπτυξη μιας εφαρμογής και η αξιοποίησή της σε καθημερινή βάση ως ένα μέσο αποθήκευσης και ταξινόμησης των εργασιών με βάση το χρόνο και τη διάρκεια.

1.2.1 Ορισμός προβλήματος

Στις μέρες μας το πρόβλημα των απαιτητικών χρονοβόρων και ταυτόχρονα απαραίτητων αλλά και μη δραστηριοτήτων φέρει ως αποτέλεσμα την αύξηση του άγχους στην καθημερινότητά μας. Οι εργασίες που αναλογούν σε καθένα από εμάς μέσα στη μέρα και οι οποίες πρέπει να διεκπεραιωθούν είναι πολλές που ο καθένας

ΥΛΟΠΟΙΗΣΗ ΔΙΑΔΙΚΤΥΑΚΗΣ ΕΦΑΡΜΟΓΗΣ ΗΜΕΡΟΛΟΓΙΟΥ ΜΕ ΔΥΝΑΤΟΤΗΤΑ ΑΥΤΟΜΑΤΟΥ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

θέλει με την ευκολία ίσως ενός απλού κλικ να τις βάλει σε σειρά χωρίς να χρειαστεί να σκεφτεί ο ίδιος τι πρέπει να κάνει πρώτο και τι δεύτερο και παράλληλα να καταφέρει να οργανωθεί με τον πιο αποτελεσματικό τρόπο ώστε όλα να είναι έτοιμα στην ώρα τους ή και νωρίτερα. Δυστυχώς στις μέρες μας ο χρόνος είναι πολύτιμος και η παραμικρή σπατάλη μπορεί να προκαλέσει προβλήματα. Η εξοικονόμησή του ακόμα και από τα πιο απλά πράγματα χρήζει σπουδαίας σημασίας και η αξιοποίηση της τεχνολογίας μπορεί να πάρει καθοριστικό ρόλο για το σκοπό αυτό.

Στην αγορά υπάρχουν αρκετές εφαρμογές που παρέχουν ένα ψηφιακό ημερολόγιο καθώς και σημειώσεις πάνω σε αυτό, κοινοποίηση ημερολογίου ακόμα και διαδικτυακές βιντεοδιασκέψεις αλλά δεν παρέχουν αυτόματο προγραμματισμό των καταχωρημένων δραστηριοτήτων, ώστε να βοηθούν ένα βήμα παραπέρα τον χρήστη με τις δραστηριότητές του.

1.2.2 Τρόπος επίλυσης του προβλήματος

Τα κινητά και οι υπολογιστές είναι γεγονός ότι αποτελούν κομμάτι του εαυτού μας για πολλούς από εμάς. Οι γρήγορες και αποδοτικές λύσεις που μας παρέχουν διευκολύνουν ολοένα και περισσότερο τον τρόπο ζωής μας αντιμετωπίζοντας κάθε πρόβλημα. Στην προκειμένη περίπτωση η αντιμετώπιση του προβλήματος έγινε με την υλοποίηση ενός προγράμματος το οποίο έχει τη δυνατότητα να αποθηκεύει όσα δεδομένα θέλει και χρειάζεται ο χρήστης για μια μέρα και να τα εμφανίζει στην οθόνη με απλό, κατανοητό και δομημένο τρόπο. Ο χρήστης μπορεί να αποθηκεύσει για όσες μέρες θέλει δεδομένα/εργασίες και η εφαρμογή είναι υπεύθυνη να του εμφανίζει επάνω σε ημερολόγιο υπενθύμιση σχετικά με το αν έχει προγραμματίσει εργασίες ή όχι για κάποια μέρα. Έτσι ο χρήστης δεν χρειάζεται να ανησυχεί πια για το πρόγραμμα της μέρας του ή να κρατάει παντού σημειώσεις σε χαρτιά!

Ο αυτόματος προγραμματισμός των δραστηριοτήτων γίνεται με βάση το εάν μια ώρα έναρξης μιας εργασίας έχει καταχωριστεί μεταξύ των ωρών έναρξης και ολοκλήρωσης μιας άλλης πιο σημαντικής ή μη όπου σε τέτοια περίπτωση η πρώτη (μη σημαντική) ταξινομείται προς τα πάνω έως ότου δεν συγκρούεται η ώρα της με άλλης. Επίσης έχουμε την ίδια ταξινόμηση εάν διαρκεί τόσο ώστε να τελειώνει μετά την έναρξη μιας άλλης σημαντικότερης. Μόλις γίνει η ταξινόμηση οι δραστηριότητες που ταξινομήθηκαν ξαναταξινομούνται ώστε να γίνει πρώτη εκείνη με την νωρίτερη ώρα έναρξης ώστε όταν έρθει η ώρα της να έχει προλάβει να ολοκληρωθεί. Σε περίπτωση τέτοιας ταξινόμησης η εφαρμογή ορίζει νέα, ελάχιστη, προτεινόμενη ώρα έναρξης της μέρας την ώρα την οποία υπολογίζει από την ώρα

ΥΛΟΠΟΙΗΣΗ ΔΙΑΔΙΚΤΥΑΚΗΣ ΕΦΑΡΜΟΓΗΣ ΗΜΕΡΟΛΟΓΙΟΥ ΜΕ ΔΥΝΑΤΟΤΗΤΑ ΑΥΤΟΜΑΤΟΥ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

έναρξης της πρώτης σημαντικής δραστηριότητας αφαιρώντας το άθροισμα των διαρκειών που κρατούν συνολικά οι ταξινομημένες πριν από αυτή δραστηριότητες. Ο χρυσός κανόνας γενικά ταξινόμησης είναι είτε μια δραστηριότητα να ξεκινήσει στην ώρα της είτε νωρίτερα και αν νωρίτερα όσο πιο κοντά στην πραγματική της ώρα γίνεται!

1.2.3 Σύντομη περιγραφή λογικής

Η λογική που ακολουθήθηκε για την υλοποίηση και ανάπτυξη της εφαρμογής αποδίδεται με τα παρακάτω βήματα.

- Αρχικά υλοποιήθηκε το ημερολόγιο και μια φόρμα συμπλήρωσης με έλεγχο πριν την τελική και οριστική καταχώρηση στη βάση των δεδομένων συμπλήρωσης και της εγκυρότητάς τους.
- Στη συνέχεια γίνεται μέσω HTTP η καταχώρησή τους στη βάση
- Υπάρχει επιλογή όπου δίνεται η δυνατότητα στο χρήστη να επιλέξει για ποια μέρα επιθυμεί να δει όλες τις καταχωρήσεις. Τα αποτελέσματα του εμφανίζονται οργανωμένα με βάση την ώρα, τη σπουδαιότητα και τη διάρκεια μιας εργασίας. Στον χρήστη εμφανίζεται και προτεινόμενη ώρα έναρξης των εργασιών του για τη μέρα.
- Αφού εμφανιστούν είναι ελεύθερος να τα τροποποιήσει ή και να τα διαγράψει εάν το επιθυμεί
- Σε κάθε περίπτωση όπως τροποποίηση ή διαγραφή εμφανίζονται σχετικά προειδοποιητικά μηνύματα πριν την οριστικοποίηση κάθε κίνησης, καθώς και μετά το πέρας αυτών που ενημερώνουν τον χρήστη για την επιτυχή ή όχι καταχώρηση στη βάση. Τροποποίηση δεν μπορεί να γίνει εάν δεν έχουν συμπληρωθεί σωστά τα σχετικά πεδία.

1.3 Οργάνωση κειμένου

Η συνέχεια της παρούσας εργασίας ολοκληρώνεται με βάση την ακόλουθη δομή. Στο κεφάλαιο 2 αναλύεται η τρέχουσα κατάσταση της εφαρμογής σε σύγκριση με άλλες παρόμοιου ενδιαφέροντος που κυκλοφορούν ήδη στην αγορά. Στο κεφάλαιο 3 παρουσιάζεται η αρχιτεκτονική της εφαρμογής με ένα σχεδιάγραμμα και η λειτουργικότητα των επιμέρους συστατικών της. Στο κεφάλαιο 4 οι λεπτομέρειες της εφαρμογής όπως οι τεχνολογίες που χρησιμοποιήθηκαν, οι αλγόριθμοι όπως και κομμάτια κώδικα που αξίζει να αναφερθούν. Στο κεφάλαιο 5 γίνεται μια σύντομη αξιολόγηση της εφαρμογής. Στο κεφάλαιο 6 παραθέτονται τα συμπεράσματα και στο κεφάλαιο 7 αναγράφονται όλες οι αναφορές στη βιβλιογραφία και σε άλλες χρήσιμες πηγές στο διαδίκτυο.

2

ΑΝΑΣΚΟΠΗΣΗ



ΤΡΕΧΟΥΣΑΣ ΚΑΤΑΣΤΑΣΗΣ

2.1 Παρόμοιες εφαρμογές στην αγορά

2.1.1 Εισαγωγή

Στην αγορά υπάρχουν ήδη πολλές εφαρμογές διαχείρισης γεγονότων που παρέχουν αρκετές δυνατότητες όπως καταγραφή εργασιών, εξατομίκευση, εκτύπωση ημερολογίου, επιλογή περιοχής και ώρας για πιθανές ειδοποιήσεις, διοργάνωση τηλεδιασκέψεων με συγκεκριμένη χρονική περίοδο στις οποίες άλλοι χρήστες μπορούν να συμμετάσχουν μετά από συγκεκριμένη πρόσκληση καθώς και εισαγωγή ήδη υπάρχοντος ημερολογίου από άλλες παρόμοιες υπηρεσίες. Δύο πολύ αξιοσημείωτες και πολύ ευρέως γνωστές τέτοιου είδους εφαρμογές είναι το Google Calendar και το Outlook Calendar.

2.1.2 Μια σύντομη γνωριμία με τις εφαρμογές της αγοράς

Το Google Calendar είναι μια υπηρεσία ημερολογίου που σχεδιάστηκε από την ίδια την Google αρχικά σε μια δοκιμαστική έκδοση τον Απρίλιο του 2006. Εξυπηρετεί στη διαχείριση του χρόνου και διαφόρων γεγονότων επιτρέποντας τη δημιουργία τους και την τροποποίησή τους σε δεύτερο χρόνο. Το Google Calendar ήταν αρχικά μια διαδικτυακή εφαρμογή καθώς υπήρχε αρχικά και για το λειτουργικό σύστημα Android και στη συνέχεια τον Μάρτιο του 2015 έγινε συμβατή και για συστήματα iOS. (“Google Calendar,” 2019)

Ως μια εναλλακτική επίσης γνωστή και εύχρηστη εφαρμογή αποτελεί το Outlook Calendar της Microsoft. Οι δυνατότητες που παρέχει κι αυτό είναι παρόμοιες με αυτές που παρέχει και της Google.

2.1.2.1 Τρόποι επίλυσης του προβλήματος της διαχείρισης χρόνου και γεγονότων από ήδη υπάρχουσες εφαρμογές στην αγορά

Στη συνέχεια θα γνωρίσουμε πιο συγκεκριμένα με ποιον τρόπο αυτές οι δύο παραπάνω γνωστές εφαρμογές προσεγγίζουν και επιλύουν το πρόβλημα της οργάνωσης γεγονότων μέσα στην ημέρα.

2.1.2.1.1 Τεχνικές με τις οποίες το Google Calendar επιτυγχάνει διαχείριση χρόνου και γεγονότων

- Επιτρέπει την υπενθύμιση γεγονότων καθώς πλησιάζει η ώρα έναρξής τους.
- Προσφέρει ως δυνατότητα την διοργάνωση μιας τηλεδιάσκεψης σε συγκεκριμένο τόπο και χρόνο για όλους τους προσκεκλημένους χρήστες.
- Με την πάροδο του χρόνου η Google πρόσθεσε τη δυνατότητα με χρήση machine learning να μπορούν οι χρήστες να βλέπουν δραστηριότητες που πρέπει να διεκπεραιωθούν και προήλθαν από αυτόματη εύρεση από το ηλεκτρονικό τους ταχυδρομείο της Google, γνωστό και ως Gmail.
- Κάποιες άλλες καινούριες δυνατότητες είναι οι «έξυπνες προτάσεις» που προτείνουν τίτλους, τοποθεσίες και σχετικές επαφές για κάθε νέο γεγονός πριν την οριστική υποβολή.
- Και τέλος τους «στόχους» οι οποίοι είναι γεγονότα/δραστηριότητες τις οποίες το ημερολόγιο οργανώνει με βάση την ώρα. (“Google Calendar,” 2019)

2.1.2.1.2 Τεχνικές με τις οποίες το Outlook Calendar επιτυγχάνει διαχείριση χρόνου και γεγονότων

- Προσφέρει δημιουργία γεγονότων και τηλεδιασκέψεις οι οποίες μπορούν να διοργανωθούν μέσω ενός επιλεγμένου mail το οποίο ορίζεται ως “appointment” με το αντίστοιχο mail ως συνημμένο και στη συνέχεια μπορεί να μετατραπεί σε τηλεδιάσκεψη με συμμετέχοντες.
- Δίνεται η δυνατότητα να προβάλλονται οι δραστηριότητες όλων των ατόμων μέσα σε μια ομάδα.

- Τα διάφορα επιπλέον ημερολόγια που προβάλλονται μπορούν να τοποθετηθούν στην οθόνη δίπλα-δίπλα ή σε μορφή “overlay” για τη διευκόλυνση του χρήστη. (Microsoft, n.d.-b, n.d.-a)

2.1.2.2 Παρατηρήσεις που προκύπτουν από τον τρόπο προσέγγισης του αυτόματου προγραμματισμού γεγονότων από τις υπάρχουσες εφαρμογές

Οι εφαρμογές που υπάρχουν ήδη στην αγορά είναι αρκετά προσεγγμένες και παρέχουν πολλές και ποικίλες δυνατότητες και χαρακτηριστικά που κατά ένα μεγάλο ποσοστό οι λειτουργίες του ενός θυμίζουν του άλλου μεταξύ δυο διαφορετικών υπηρεσιών ημερολογίου. Το Google Calendar συγκεκριμένα διαθέτει τη δυνατότητα του προγραμματισμού γεγονότων όπως έχει ήδη προαναφερθεί παραπάνω με τα επονομαζόμενα «Goals» στόχους. Οι στόχοι είναι λοιπόν μια παρόμοια δυνατότητα αυτόματου προγραμματισμού εργασιών που έχει προσεγγιστεί και χρησιμοποιηθεί και ως αντικείμενο της παρούσας εργασίας. Αν λοιπόν σε αυτούς τους «στόχους» εντοπιστεί μια δραστηριότητα η οποία τελικά δεν μπορεί να γίνει την προγραμματισμένη ημέρα για λόγους προσθήκης έστω εκτάκτων νέων δραστηριοτήτων τότε η δραστηριότητα που προκαλεί “confliction” με τη νέα τοποθετείται στην επόμενη μέρα στην ίδια ώρα, χωρίς την άδεια του χρήστη. Το γεγονός αυτό λύνεται με τους τρόπους που παρουσιάζονται παρακάτω και φέρει ως αποτέλεσμα ο χρήστης να προλαβαίνει όλες τις δραστηριότητες που έχει ορίσει για μια μέρα σημαντικές αλλά και μη χωρίς να υπάρχει μετατόπιση μιας εργασίας του για άλλη μέρα σε περίπτωση που υπάρχει σύγκρουση ωρών.

2.1.3 Τρόπος επίλυσης με την προσθήκη επιπλέον χαρακτηριστικών λειτουργικότητας

Παρακάτω παρατίθενται κάποιοι καινούριοι τρόποι προσέγγισης του ζητήματος για την υλοποίηση μιας εφαρμογής διαχείρισης χρόνου και γεγονότων. Οι τρόποι αυτοί που εφαρμόζονται προσθέτουν μια επιπλέον δυνατότητα ευελιξίας στον προγραμματισμό του προγράμματός του, χωρίς να παραληφθεί κάποια εργασία του ή μετατοπιστεί σε άλλη μέρα χωρίς δική του απόφαση και επιλογή.

1. Προσθήκη δυνατότητας προτεινόμενης ώρας έναρξης

Με αυτό το χαρακτηριστικό στον χρήστη επιτρέπεται να ορίσει κατά την προσθήκη και δημιουργία νέων εργασιών μια επιλογή “Priority” δίπλα σε κάποια εργασία που πιστεύει ότι είναι υποχρεωτική να διεκπεραιωθεί στην ώρα της και μόνο... Εάν θέλει μπορεί να καταχωρήσει και άλλες σε διάφορες ώρες εκτός ήδη υπαρχόντων ακόμα και αν εντοπίζεται σύγκρουση

ωρών... Σε αυτήν την περίπτωση βέβαια η μία θα πρέπει να είναι τύπου “Priority” ενώ η άλλη όχι, τότε η εφαρμογή είναι υπεύθυνη να ταξινομήσει τις εργασίες και να προτείνει στον χρήστη την υπολογισμένη σειρά. Επειδή τις ώρες που δημιουργούν σύγκρουση η εφαρμογή τις μετατοπίζει προς τα πάνω ώστε να γίνουν στην αρχή της μέρας υπολογίζει και νέα ώρα έναρξης της μέρας. Εάν ο χρήστης ακολουθήσει την νέα ώρα θα προλάβει στην ώρα του τις σημαντικές υποχρεώσεις του καθώς θα έχει ήδη έτοιμες μέσα στη μέρα που τις όρισε και τις όχι τόσο σημαντικές υποχρεώσεις του.

2. Προσθήκη δυνατότητας ένδειξης περισσευούμενων μέγιστων λεπτών καταχώρησης κατά την διαδικασία της φόρμας συμπλήρωσης πριν την καταχώρηση

Για την αποφυγή λάθους της προτεινόμενης ώρας η εφαρμογή επιτρέπει στον χρήστη να καταχωρήσει τόσες εργασίες ώστε όλες μαζί να μην ξεπερνάνε σε διάρκεια τα συνολικά λεπτά που περιλαμβάνει μια ημέρα 24 ωρών. Κάθε φορά που ο χρήστης κάνει μια καταχώρηση τα συνολικά αυτά λεπτά μειώνονται, αφαιρείται η διάρκεια της νέας προς καταχώρηση δραστηριότητας. Σε αντίθετη περίπτωση η υπολογιζόμενη προτεινόμενη ώρα θα μπορούσε να αναφέρεται σε προηγούμενη ή επόμενη ή ακόμα και μεθεπόμενη μέρα κ.ο.κ. γεγονός που οδηγεί σε λάθος προγραμματισμό και άρα και λάθος διεκπεραίωση των γεγονότων από τον χρήστη.

3. Προσθήκη δυνατότητας ορισμού σημαντικών (priority) και μη σημαντικών δραστηριοτήτων για την καλύτερη οργάνωση των εργασιών

Αυτή η ρύθμιση δίνει τη δυνατότητα ο χρήστης να ξεχωρίζει σε σημαντικές και μη σημαντικές τις δραστηριότητές του ώστε να μπορεί να τις προγραμματίσει όλες στο διάστημα της ημέρας που τις επέλεξε χωρίς να χρειαστεί να μετακινήσει κάποια για άλλη μέρα επειδή απλώς οι ώρες συγκρούονται. Σε αυτή την περίπτωση αυτές που ταξινομούνται σε άλλη θέση στην εφαρμογή είναι οι μη σημαντικές και τοποθετούνται να γίνουν πριν την ώρα τους και με αύξουσα σειρά ώστε να έχουν ολοκληρωθεί μέχρι την ώρα τους. Στη συνέχεια η εφαρμογή προτείνει ώρα έναρξης τέτοια ώστε να έχουν προλάβει όλες οι άλλες που άλλαξαν θέση να ολοκληρωθούν μέχρι την ώρα έναρξης της πρώτης σημαντικής. Οι δε σημαντικές δεν αλλάζουν θέση και απαγορεύεται να υπάρχουν ώρες σύγκρουσης μεταξύ δύο σημαντικών κάτι που επιτρέπεται στις μη σημαντικές αφού ταξινομούνται έτσι ώστε να μην συγκρούονται και προτείνεται νέα ώρα έναρξης.

4. Έλεγχοι για την πρόληψη πιθανών συγκρούσεων ωρών

Οι έλεγχοι που έχουν συμπεριληφθεί στην εφαρμογή κατά τη συμπλήρωση φόρμας ή κάποιας τροποποίησης γενικά είναι οι εξής για την επιτυχή προσθήκη γεγονότων και μετέπειτα ταξινόμησή τους έτσι ώστε να μην αγνοηθεί για δεύτερο χρόνο καμία:

- Έλεγχος μην υπάρχει η προς καταχώρηση ώρα ήδη στο σύστημα. Δύο δραστηριότητες δεν μπορούν να ξεκινούν την ίδια ώρα είτε είναι είτε δεν είναι τύπου “Priority”.
- Έλεγχος να μην καταχωρείται νέο γεγονός τύπου “Priority” εάν υπάρχει ήδη άλλο πάλι ίδιου τύπου και για τις ώρες και των δύο ισχύει το εξής:

Ώρα Ολοκλήρωσης X = Ώρα Έναρξης X + Διάρκεια X

Ώρα Ολοκλήρωσης X+1 = Ώρα Έναρξης X+1 + Διάρκεια X+1

Ώρα Ολοκλήρωσης X > Ώρα Έναρξης X+1 **ΚΑΙ** Ώρα Έναρξης X < Ώρα Έναρξης X+1

ή

Ώρα Έναρξης X > Ώρα Έναρξης X+1 **ΚΑΙ** Ώρα Έναρξης X < **Ώρα Ολοκλήρωσης X+1**

Σε κάθε περίπτωση η εφαρμογή προτείνει λύσεις διαμόρφωσης των προς καταχώρηση στοιχείων στη φόρμα ώστε να μην υπάρχει άλλο “confliction”.

2.2 Τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση της διαδικτυακής εφαρμογής και λογισμικά



2.2.1 Node

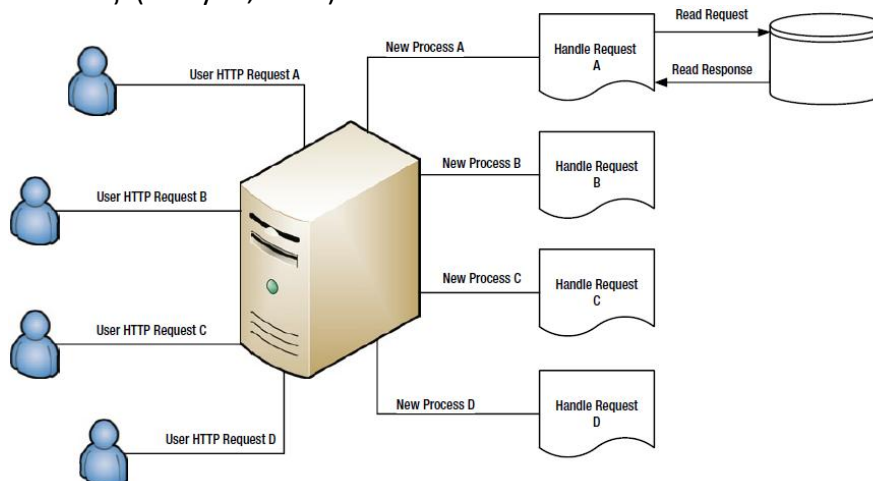
Το NodeJS είναι μια πλατφόρμα ανάπτυξης λογισμικού (κυρίως για διακομιστές) χτισμένη σε περιβάλλον JavaScript. Σε αντίθεση από τα περισσότερα σύγχρονα περιβάλλοντα ανάπτυξης εφαρμογών δικτύων μία διεργασία node δεν στηρίζεται στην πολυνηματικότητα αλλά σε ένα μοντέλο ασύγχρονης επικοινωνίας εισόδου/εξόδου. (“NodeJS,” 2019; Tilkov & Vinoski, n.d.)

Αρχικά θα πούμε δυο λόγια για τον αρχικό τρόπο λειτουργίας των διακομιστών και τα ελαττώματα που αυτοί οι τρόποι παρουσίαζαν σχετικά με την ΥΛΟΠΟΙΗΣΗ ΔΙΑΔΙΚΤΥΑΚΗΣ ΕΦΑΡΜΟΓΗΣ ΗΜΕΡΟΛΟΓΙΟΥ ΜΕ ΔΥΝΑΤΟΤΗΤΑ ΑΥΤΟΜΑΤΟΥ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

κατανάλωση CPU και RAM. Στη συνέχεια θα δώσουμε έμφαση στον τρόπο λειτουργίας του NodeJS επισημαίνοντας έτσι την επικράτειά του έναντι άλλων μεθόδων ανάπτυξης λογισμικού για εξυπηρετητές.

2.2.1.1 Παραδοσιακοί εξυπηρετητές που χρησιμοποιούν μια διεργασία ανά request

Οι παραδοσιακοί εξυπηρετητές συνηθίζουν να δημιουργούν μία καινούρια διεργασία κάθε φορά για να αναλάβει κάθε ένα ξεχωριστό request. Κάτι τέτοιο αποτελεί βαριά εργασία και υπερβολική κατανάλωση πόρων όσον αφορά CPU και RAM. Στην παρακάτω εικόνα φαίνεται η παραπάνω διαδικασία. Πιο συγκεκριμένα η διαδικασία ενός read request μπορεί να διαρκέσει κάποια λεπτά. Αυτό θα δεσμεύσει τον επεξεργαστή και τη μνήμη για άγνωστο χρόνο καταναλώνοντας πόρους που θα μπορούσαμε να εκμεταλλευτούμε όσο περιμένουμε την απάντηση από τη βάση δεδομένων. Επίσης οι διεργασίες είναι χρονοβόρες στο να ξεκινήσουν και δεσμεύουν πολύ χώρο και στη μνήμη. Γι' αυτό το λόγο πολλές σύγχρονες διαδικτυακές εφαρμογές χρησιμοποιούν το λεγόμενο thread pool για την καταπολέμηση της παραπάνω δυσκολίας. (Ali Syed, 2014)



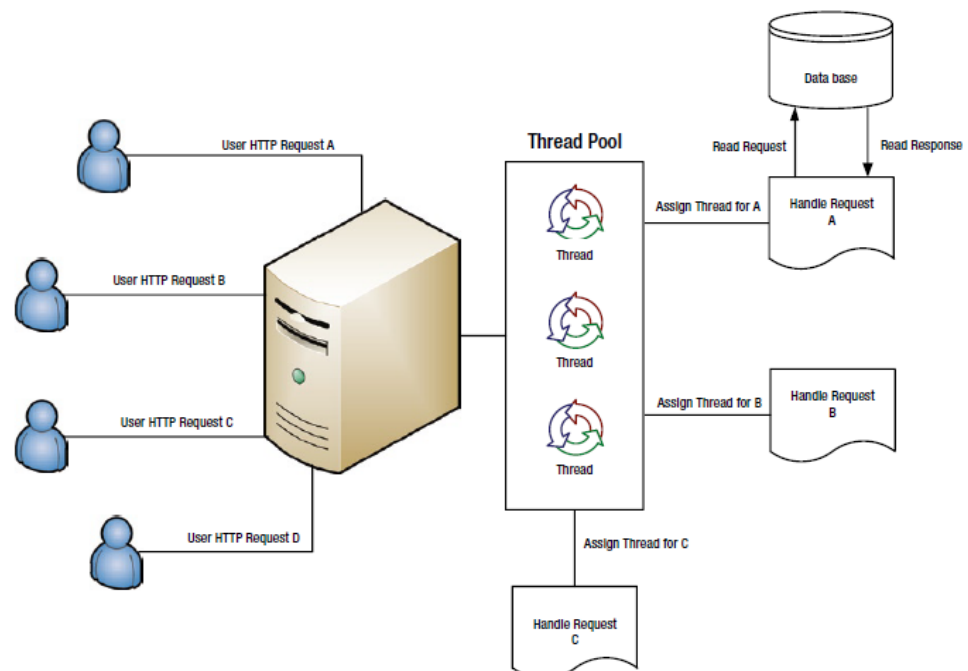
Εικ.1. πρώτος παραδοσιακός τρόπος λειτουργίας ενός εξυπηρετητή (Ali Syed, 2014)

2.2.1.2 Παραδοσιακοί εξυπηρετητές που χρησιμοποιούν Thread Pool

Οι σύγχρονοι εξυπηρετητές χρησιμοποιούν ακόμα ένα μοντέλο λειτουργίας το οποίο βασίζεται στη λογική του thread pool. Χρησιμοποιώντας λοιπόν ένα thread από ένα thread pool εξυπηρετούν κάθε request. Επειδή έχουμε ήδη κάποια λειτουργικά συστήματα που λειτουργούν με αυτή τη λογική δεν ανησυχούμε για

κάποιο πιθανό ενδεχόμενο να καθυστερούσαμε στη διαδικασία ξεκινήματος ή σταματήματος διεργασιών του λειτουργικού συστήματος, οι οποίες για να δημιουργηθούν θα καταναλώναν και θα δέσμευαν τεράστιες ποσότητες πόρων από ένα απλό thread. Πλέον όταν ένα request δημιουργηθεί δεσμεύουμε για όσο χρονικό διάστημα χρειαστεί το request ένα thread να το επεξεργαστεί. Η μέθοδος αυτή είναι πολύ προτιμότερη από την προηγούμενη και πολλοί εξυπηρετητές την εφαρμόζουν ακόμα και σήμερα. Παρόλα αυτά το μειονέκτημα και σε αυτήν είναι ότι καταναλώνει πολύ RAM μεταξύ των threads και για να μεταφέρει δεδομένα από το ένα thread στο άλλο όπως απαιτεί το λειτουργικό σύστημα OS καταναλώνει πολλούς πόρους CPU.

Στην παρακάτω εικόνα απεικονίζεται ακριβώς η παραπάνω λογική. (Ali Syed, 2014)



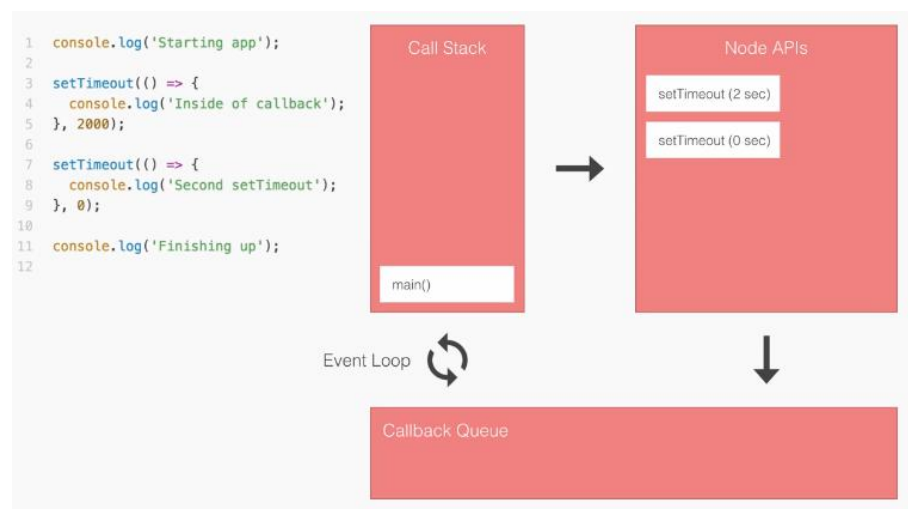
Εικ.2. δεύτερος παραδοσιακός τρόπος λειτουργίας ενός εξυπηρετητή (Ali Syed, 2014)

2.2.1.3 Γιατί NodeJS ?

Το NodeJS δουλεύει με τη λογική του να υπάρχει ένα thread που θα χειρίζεται requests. Η ιδέα δεν είναι καινούρια καθώς το Nginx έχει χτιστεί με βάση αυτόν τον κανόνα. Το Nginx είναι ένας single-threaded web server και μπορεί να χειριστεί έναν τεράστιο αριθμό ταυτόχρονων (concurrent) requests. (Ali Syed, 2014)

Το NodeJS λοιπόν θυμίζει στη λογική που προαναφέρθηκε τη λογική που χρησιμοποιεί το JavaScript το οποίο είναι single-threaded

όχι ακριβώς αλλά θεωρείται έτσι επειδή έχει ένα single-threaded event loop. Γενικά αυτό το event loop λειτουργεί βάζοντας την main μέσα σε μια call stack και τρέχει όσες είναι για εκτύπωση στην κονσόλα απευθείας. Στη συνέχεια μπαίνουν και οι επόμενες μία- μία στο stack και όσες μεθόδους καλούμε χτυπώντας το node api ενεργοποιείται η διαδικασία event-callback το event μέσα στο node api όπως δείχνει η εικόνα περιμένει 2' και μετά γυρνάει στη callback μέσα στη callback queue αφού την έχει διαγράψει από τη stack. Μόλις διαγραφούν τα πάντα από τη stack το event loop το βλέπει και παίρνει την πρώτη μέθοδο από την callback queue για εκτέλεση. Η εικόνα απεικονίζει ακριβώς αυτή τη λογική. (Patel, 2018)



Εικ.3. Η λογική του JavaScript(Patel, 2018)

Το NodeJS λειτουργεί και αυτό σαν την παραπάνω λογική αφού είναι βασισμένο στο JavaScript χρησιμοποιεί δηλαδή ένα event-driven, non-blocking I/O μοντέλο που το καθιστά ελαφρύ και αποδοτικό ταυτόχρονα. Μια διαδικασία I/O είναι απαιτητική και χρειάζεται χρόνο με αποτέλεσμα να μπλοκάρει άλλες ενέργειες που πρέπει να γίνουν. Εδώ έρχεται να βοηθήσει ο non-blocking I/O τρόπος που ελαχιστοποιεί την ανάγκη multi-threading μιας που ο εξυπηρετητής πλέον μπορεί να διαχειριστεί πολλά requests την ίδια στιγμή. (Patel, 2018)

Αυτή ακριβώς η παραπάνω λογική καθιστά το NodeJS ένα πολύ χρήσιμο εργαλείο για ανάπτυξη λογισμικού από μεριάς εξυπηρετητών κάνοντάς το παράλληλα την καλύτερη επιλογή.



2.2.2 Npm

Το npm είναι ο αναγκαίος package manager των πακέτων του Node. Το Node έρχεται με διάφορα πακέτα όπως το cli της angular, την express και άλλα πολλά. Γενικά ένα πακέτο δεν είναι τίποτα άλλο από βιβλιοθήκες μαζεμένες όλες σε ένα πακέτο.

Ένας package manager έχει δύο βασικές ευθύνες οι οποίες είναι να εγκαθιστά πακέτα και να διαχειρίζεται “dependencies”. Ο npm είναι ένας γρήγορος, ικανός και εύκολος package manager. Ο npm εγκαθίσταται μαζί με το Node. (Brown, 2014)



2.2.3 Angular

Η Angular είναι το πιο διαδεδομένο και κυρίαρχο framework για να αναπτύξει κανείς απαιτητικές JavaScript εφαρμογές. Χρησιμοποιείται συγκεκριμένα για την ανάπτυξη SPA εφαρμογών, οι οποίες είναι εφαρμογές όπου δεν απαιτούν την επαναφόρτιση ολόκληρης της σελίδας παρά μόνο εκείνου του στοιχείου που ζητείται. Επίσης κρατάνε ιστορικό δεδομένων ώστε μόλις γίνει επιστροφή στην προηγούμενη μορφή της να έχουν επαναφερθεί τα δεδομένα τα οποία είχαν συμπληρωθεί στο προηγούμενό τους στάδιο. (Lim, 2017)

2.2.3.1 Γιατί Angular ?

Η Angular από την πρώτη της έκδοση έχει γνωρίσει πολύ μεγάλη άνθιση καθώς ενημερώνεται 2 φορές το χρόνο. Ακολουθεί το μοντέλο MVC.

Τα πλεονεκτήματά των νέων εκδόσεών της είναι:

- Είναι σύγχρονη, ικανή και εύκολη ακόμα και για νέους προγραμματιστές. Τα Views και οι Controllers έχουν πλέον αντικατασταθεί από τα components
- Είναι γραμμένη σε typescript
- Οι επόμενες εκδόσεις της θυμίζουν τις προηγούμενες. Π.χ. οι νέες εκδόσεις εξακολουθούν να χρησιμοποιούν

“dependency injection” και όσοι προγραμματιστές είναι ήδη εξοικειωμένοι με αυτή τη λογική μπορούν να την επωφεληθούν και στις νέες εκδόσεις. Είναι γενικά ποιο εύκολη η αλλαγή από μια προηγούμενη έκδοση της σε επόμενη της παρά η αλλαγή από κάποια άλλη βιβλιοθήκη όπως η React ή framework όπως το Ember. (*Rangle’s Angular 2 Training Book*, n.d.)

2.2.3.2 Αρχιτεκτονική της Angular

Τα τέσσερα βασικά συστατικά που την αποτελούν είναι τα components, τα services, τα routers και τα directives.

Τα components περικλείουν μέσα τους τα δεδομένα, το template και τη συμπεριφορά του. Κάθε component έχει τη δική του html στο template του καθώς και τις δικές του μεταβλητές και λογική. Είναι απλά αρχεία TypeScript με μια κλάση γραμμένη σε TypeScript, που έχει μεθόδους και μεταβλητές. Οι μεταβλητές κρατούν τα δεδομένα για το view και οι μέθοδοι υλοποιούν τη συμπεριφορά του.

Services Τα components μερικές φορές χρειάζονται να επικοινωνούν με κάποιους backend servers (π.χ. Node, ASP.NET, Ruby on Rails) για να πάρουν ή να καταχωρήσουν δεδομένα. Για να έχουμε τα πράγματα οργανωμένα στον κώδικα μεταφέρουμε όση λογική δεν έχει σχέση με το view σε ένα service το οποίο είναι μία απλή κλάση που στο σώμα της υλοποιεί ασυσχέτιστη με το view λογική όπως η κλήση μέσω http, logging στην κονσόλα, κλπ.

Router Ένας router είναι υπεύθυνος για την πλοήγηση από σελίδα σε σελίδα. Καθώς πλοηγούμαστε λοιπόν ο router θα εντοπίσει ποιο component θα εμφανίσει στον χρήστη με βάση την αλλαγή που θα έχει γίνει στο όνομα του router

Directives Παρόμοια με τα components λοιπόν χρησιμοποιούμε και directives για να τροποποιήσουμε το DOM, όπως το να προσθέσουμε επιπλέον συμπεριφορά σε ήδη υπάρχοντα στοιχεία του DOM. Για παράδειγμα χρησιμοποιούμε την autoGrow directive για να κάνουμε ένα απλό textbox να μεγαλώνεται αυτόματα όταν ο χρήστης επικεντρώνεται πάνω του.

```
<input type="text" autoGrow />
```

Η Angular έχει χιλιάδες directives για κοινές απαιτήσεις όπως προσθήκη ή διαγραφή διαφόρων DOM στοιχείων, προσθήκη κλάσεων ή στυλ σε αυτά τα στοιχεία ή ακόμα και για την επανάληψη

τους. Μας επιτρέπει να φτιάξουμε μέχρι και δικά μας directives. (Lim, 2017)

MODULES

Η Angular έχει κάτι μηχανισμούς για να ομαδοποιεί τα components, τα directives, τα pipes, και τα services που συσχετίζονται μεταξύ τους με τέτοιο τρόπο που μπορούν να συνδυαστούν με άλλους τέτοιους μηχανισμούς ώστε να δημιουργήσουν την τελική εφαρμογή. Αυτοί οι μηχανισμοί ονομάζονται modules. Μια λοιπόν Angular εφαρμογή μπορεί να γίνει αντιληπτή σαν ένα παζλ όπου κάθε κομμάτι είναι ένα module που είναι απαραίτητο για να δει κανείς όλη την εικόνα. (*Rangle's Angular 2 Training Book*, n.d.)

Το Gmail για παράδειγμα είναι μια τεράστια εφαρμογή που μέσα της συναντάμε διάφορες λειτουργικότητες όπως τα Εισερχόμενα, τις Επαφές και τη δημιουργία νέου mail. Κάθε λειτουργία έχει υλοποιηθεί με το δικό της σετ από υψηλού επιπέδου συσχετιζόμενες κλάσεις, όπου εμείς ομαδοποιούμε σε ένα module. (Lim, 2017)



2.2.4 Express

Η ιστοσελίδα της Express περιγράφει την Express ως «ένα μινιμαλιστικό και ευέλικτο NodeJS framework διαδικτυακών εφαρμογών, που παρέχει ένα δυνατό σετ χαρακτηριστικών για ανάπτυξη single, multiple καθώς και υβριδικών web εφαρμογών»

Ας δούμε μια ανάλυση του παραπάνω ορισμού...

Μινιμαλιστικό

Αποτελεί ίσως ένα από τα πιο ενδιαφέροντα χαρακτηριστικά της Express. Πολλοί προγραμματιστές frameworks αδιαφορούν για το ότι πολλές φορές «το απλό είναι και ικανό». Η φιλοσοφία της Express είναι να προσφέρει το

πιο μινιμαλιστικό στρώμα επικοινωνίας μεταξύ του τρόπου σκέψης του ανθρώπινου εγκεφάλου και ενός εξυπηρετητή, χωρίς αυτό να σημαίνει ότι δεν είναι ικανή για πολλά. Αντίθετα σημαίνει ότι δεν σε εμποδίζει από το να εκφράσεις όλες σου τις ιδέες, προσφέροντάς σου παράλληλα κάτι χρήσιμο.

Ευέλικτο

Μια άλλη πλευρά της φιλοσοφίας της Express είναι ότι είναι επεκτάσιμη. Πολλά frameworks έχουν ως πρώτη προϋπόθεση την διαγραφή μη χρήσιμων λειτουργιών ή την αντικατάστασή τους σε περίπτωση που οι λειτουργίες αυτές δεν απαιτούνται. Αυτό οδηγεί σε χάσιμο χρόνου. Η Express εγγυάται ακριβώς το αντίθετο αφού προσφέρει μινιμαλισμό επιτρέποντάς σου να προσθέσεις διαφορετικά κομμάτια των λειτουργιών της όπως και όταν τα χρειάζεσαι και τα οποία αντικαθιστούν ότι δεν ικανοποιεί τις ανάγκες της εφαρμογής.

Web application framework

Τι είναι μια web εφαρμογή? Είναι εφικτό να φτιάξουμε ένα website με την Express ή όχι? Ναι, ένα website είναι μια web εφαρμογή, αλλά μια web εφαρμογή μπορεί να είναι κάτι παραπάνω. Μπορεί να προσφέρει λειτουργικότητα σε άλλες web εφαρμογές. Γενικά, μια εφαρμογή δεν είναι κάτι στατικό (αν και ακόμα και κάτι τέτοιο δεν παύει να αποτελεί ένα απλό παράδειγμα web εφαρμογών). Ενώ προς το παρόν υπάρχει διαφοροποίηση μεταξύ μιας εφαρμογής που τρέχει τοπικά στη συσκευή ενός χρήστη με μια web εφαρμογή που τρέχει απομακρυσμένα σε άλλο μηχάνημα και παρέχεται στη συσκευή του χρήστη μέσα από το δίκτυο, αυτή η διαφοροποίηση γίνεται ολοένα και πιο θολή χάρις τις προσπάθειες της Microsoft να επιτρέψει σε εφαρμογές που χρησιμοποιούν HTML5 να τρέχουν στην επιφάνεια εργασίας σαν να ήταν τοπικές. Εύκολα κανείς μπορεί να φανταστεί ότι σε λίγα χρόνια δεν θα υπάρχει πια αυτή η διαφοροποίηση μεταξύ μιας web εφαρμογής και μιας τοπικής.

Single-page web applications

Single-page web application ή αλλιώς SPA είναι σχετικά μια καινούρια ιδέα που σκοπό έχει να μην απαιτείται κάθε φορά που ο χρήστης ζητά μια νέα σελίδα απ' την ιστοσελίδα να γίνεται κάποιο request στον server παρά να κατεβάζει ολόκληρη τη σελίδα στον φυλλομετρητή του. Με αυτή τη λογική η περιήγηση στη σελίδα θα είναι πιο γρήγορη μιας που δεν θα υπάρχει ή θα υπάρχει σχετικά μικρή επικοινωνία με τον εξυπηρετητή. Η Express λοιπόν μπορεί να συνεργαστεί με τεχνολογίες όπως η Angular ή η Ember.

Multipage and hybrid web applications

Multipage web application είναι μια πιο παραδοσιακή προσέγγιση για ιστοσελίδες. Είναι η περίπτωση όπου για κάθε περιήγηση σε διαφορετική σελίδα μέσα σε μια ιστοσελίδα γίνεται κλήση στον διακομιστή. Αυτό δεν την κάνει μη επιτρεπτή προσέγγιση όμως. Υπάρχουν περιπτώσεις όπου μπορεί

να γίνει επιλογή ποιο κομμάτι της σελίδας θα ανήκει σε αυτού του είδους την προσέγγιση και ποιο θα εφαρμόζει τη λογική του Single-page web application. Αυτή η τελευταία λογική συνδυασμού των δύο παραπάνω προσεγγίσεων είναι αυτή που αποκαλούμε σήμερα υβριδική. (Brown, 2014)

2.2.5 MySQL



Η MySQL είναι μια σχετικά πρόσφατη προσθήκη στον τέλεια διαμορφωμένο κόσμο των συστημάτων διαχείρισης σχετικών βάσεων δεδομένων (*relational database management systems*) (RDBMs), και είναι μια ιδέα που εφευρέθηκε από έναν ερευνητή της IBM.

Προσφέρει:

Ταχύτητα

Μπορεί να τρέχει σε σχετικά περιορισμένο hardware και καταναλώνει πολύ λίγους πόρους από το σύστημα. Η ταχύτητα με την οποία λαμβάνει δεδομένα την κάνει την αγαπημένη των διαχειριστών δικτύου (web administrators).

Ευκολία Εγκατάστασης

Μπορεί να εγκατασταθεί χωρίς δύσκολες και φιλοσοφημένες παραμετροποιήσεις (configurations). Ειδικά που τώρα πολλά κατανεμημένα Linux συστήματα συμπεριλαμβάνουν τη εγκατάσταση της MySQL η διαδικασία μπορεί να γίνει ακόμα πιο αυτόματη, χωρίς αυτό όμως να σημαίνει ότι η MySQL είναι ανεξαρτημένη από διαχειριστικές διεργασίες.

Τήρηση προτύπων

Πολλά πρότυπα υπάρχουν στον κόσμο των σχετικών βάσεων δεδομένων και είναι αδύνατο να υπάρξει ακριβής συμμόρφωση με όλα αυτά. Η MySQL όμως προετοιμάζει για το ενδεχόμενο μεταφοράς σε άλλες μηχανές βάσεων. Η μεταφορά κώδικα από τη μία βάση στην άλλη δεν ήταν ποτέ κάτι το ασήμαντο, αλλά η MySQL κάνει μια πολύ λογική προσπάθεια στο να παρέχει ένα σίγουρο περιβάλλον (standard environment), καθώς γίνεται ολοένα και καλύτερη αναπτύσσοντας κι άλλα χαρακτηριστικά.

Ανταπόκριση από την κοινότητα

Υπάρχουν τοπικοί MySQL ομάδες χρηστών σχεδόν σε κάθε μεγάλη πόλη και η ανταπόκριση που έχει ενισχύεται από το γεγονός ότι είναι ανοικτού κώδικα και δωρεάν. Έτσι κάθε προγραμματιστής με επαρκή ικανότητες μπορεί να βοηθήσει στην διόρθωση σφαλμάτων.

Εύκολη διεπαφή σε άλλα λογισμικά

Είναι εύκολο να τη χρησιμοποιήσεις ως κομμάτι ενός μεγαλύτερου λογισμικού συστήματος. Για παράδειγμα είναι εφικτή η συγγραφή προγράμματος που θα επικοινωνεί απευθείας με μία MySQL βάση δεδομένων. Οι περισσότερες δημοφιλέστερες γλώσσες προγραμματισμού διαθέτουν ειδικές βιβλιοθήκες για χρήση με την MySQL. Κάποια παραδείγματα τέτοιων γλωσσών είναι οι:

C, PHP, Perl, Python, Ruby, and the Microsoft .NET languages. Η MySQL επίσης υποστηρίζει και το πρότυπο Open Database Connectivity (ODBC), για να είναι προσβάσιμη ακόμα και όταν η συγκεκριμένη λειτουργικότητα που απαιτείται για τη MySQL δεν είναι διαθέσιμη. (Tabaghoghi & Williams, 2007)

2.2.6 Xampp



Το XAMPP είναι ένα πακέτο προγραμμάτων ελεύθερου λογισμικού ανοικτού κώδικα και ανεξαρτήτου πλατφόρμας το οποίο περιέχει τον εξυπηρετητή ιστοσελίδων http Apache, την βάση δεδομένων MySQL και ένα διερμηνέα για κώδικα γραμμένο σε γλώσσες προγραμματισμού PHP και Perl. ("XAMPP," 2017)

2.2.7 Visual Studio Code



Για την ανάπτυξη οποιασδήποτε εφαρμογής, λογισμικού ή συστήματος βασικό ρόλο έχει και η επιλογή ενός κατάλληλου editor. Για την ανάπτυξη λοιπόν της παρούσας εφαρμογής ως editor επιλέχτηκε το visual studio code της Microsoft.

Το visual studio code είναι ένας editor ανοικτού λογισμικού που μπορεί να χρησιμοποιηθεί με μια μεγάλη ποικιλία γλωσσών προγραμματισμού και έχει δημιουργηθεί από τη Microsoft για πλατφόρμες Windows, Linux και macOS και είναι γραμμένο σε typescript, javascript και css. ("Visual Studio Code," 2019) Υποστηρίζει δικό του debugger με breakpoints, call stacks και διαδραστική κονσόλα. Υποστηρίζει επίσης snippets, code refactoring, syntax highlighting, autocomplete και με χρήση IntelliSense πηγαίνει ένα βήμα παραπάνω στο να παρέχει έξυπνες συμπληρώσεις κώδικα βασισμένες σε τύπους μεταβλητών, ορισμούς μεθόδων και προστιθέμενων modules. (Microsoft, 2019) Επιπλέον μπορεί να προσθέσει κι άλλη

ΥΛΟΠΟΙΗΣΗ ΔΙΑΔΙΚΤΥΑΚΗΣ ΕΦΑΡΜΟΓΗΣ ΗΜΕΡΟΛΟΓΙΟΥ ΜΕ ΔΥΝΑΤΟΤΗΤΑ ΑΥΤΟΜΑΤΟΥ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

λειτουργικότητα με την εγκατάσταση διαφόρων extensions. (“Visual Studio Code,” 2019) Στα πλαίσια της εφαρμογής αυτής έχει γίνει εγκατάσταση ενός extension και συγκεκριμένα του tslint ενός extension που βοηθάει για τη διόρθωση του συντακτικού, προτείνοντας λύσεις. Τέλος παρέχει και ενσωματωμένο terminal για άμεση εκτέλεση εντολών του node το οποίο χρησιμοποιήθηκε για την εφαρμογή αυτή. Σε έρευνα που διεξάγει στο StackOverflow.com το 2019 το visual studio code ψηφίστηκε ως το πιο δημοφιλές εργαλείο περιβάλλον ανάπτυξης κώδικα με 50.7% στους 87.317 χρήστες να ισχυρίζονται ότι το χρησιμοποιούν. (“Visual Studio Code,” 2019)



2.2.8 Postman

Για τη δημιουργία του rest api της εφαρμογής χρησιμοποιήθηκε ένα ξεχωριστό εργαλείο ώστε αυτό να υλοποιηθεί πιο γρήγορα και χωρίς σφάλματα σε πρώτο χρόνο ακόμα και αν η εφαρμογή δεν έχει ολοκληρωθεί ακόμα. Το εργαλείο που χρησιμοποιήθηκε λέγεται postman.

Ο postman είναι ένα εργαλείο από μεριάς του client για άμεσες κλήσεις σε api, το οποίο βοηθάει στον έλεγχο των api's. Μας επιτρέπει να ελέγχουμε το ίδιο request έναντι διαφορετικών περιβαλλόντων με συγκεκριμένες environment μεταβλητές. (*Testing with Postman*, n.d.)



2.2.9 Google Chrome

Το frontend της εφαρμογής έτρεχε σε έναν browser και γι' αυτό το σκοπό χρησιμοποιήθηκε το google chrome. Επίσης το debugging του κώδικα από μεριάς frontend έγινε στον εξαιρετικά γραμμένο και ικανό debugger του chrome.

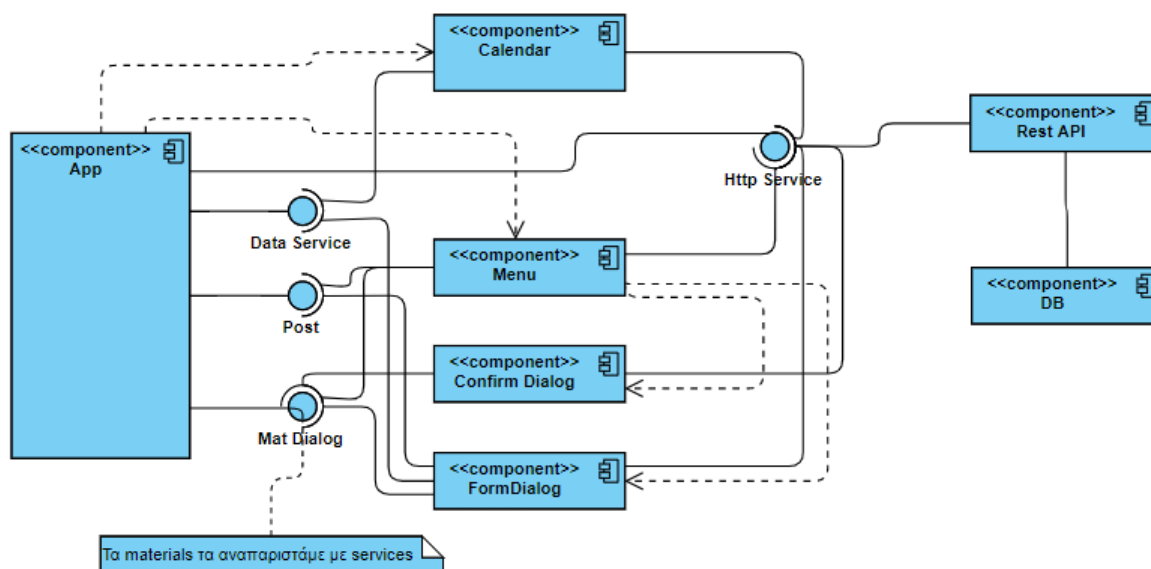
3

ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΥΣΤΗΜΑΤΟΣ

3.1 Εισαγωγή

Σκοπός αυτού του κεφαλαίου είναι να αναλυθεί και να περιγραφεί η αρχιτεκτονική την οποία ακολουθεί η εφαρμογή, ο τρόπος συσχέτισης των επιμέρους συστατικών της με τις αρχικές απαιτήσεις και αντικειμενικούς στόχους που είχαν τεθεί πριν την υλοποίηση της καθώς επίσης και η λειτουργικότητα καθενός συστατικού από αυτά.

3.2 Συνδεσμολογία



Εικ.4. Component Diagram

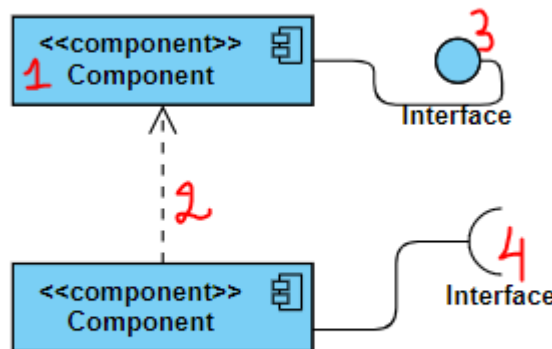
3.2.1 UML Component model Diagram ?

Για την αναπαράσταση της αρχιτεκτονικής καθώς και την επεξήγηση αυτής χρησιμοποιήθηκε component diagram.

Ένα component diagram είναι ένα αναπόσπαστο κομμάτι για την ανάπτυξη ενός λογισμικού συστήματος. Σχεδιάζεται με τη βοήθεια ενός ΥΛΟΠΟΙΗΣΗ ΔΙΑΔΙΚΤΥΑΚΗΣ ΕΦΑΡΜΟΓΗΣ ΗΜΕΡΟΛΟΓΙΟΥ ΜΕ ΔΥΝΑΤΟΤΗΤΑ ΑΥΤΟΜΑΤΟΥ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

λογισμικού σχεδίασης σε UML και είναι οι αρωγοί στην κατανόηση της δομής των ήδη υπαρχόντων συστημάτων με σκοπό την επιτυχή δημιουργία νέων. Σκοπός τους είναι να τονίσουν τη σχέση μεταξύ διαφορετικών components του συστήματος. Στη UML κατατάσσεται στην κατηγορία των Structural Diagrams (“Component Diagram Tutorial,” n.d.)

Πριν εξηγήσουμε πως σχετίζονται τα components στο παραπάνω διάγραμμα, θα εξηγήσουμε πρώτα με λίγα λόγια τα σύμβολα τα οποία εμφανίζονται στο διάγραμμα. Η παρακάτω εικόνα έχει αριθμημένα όλα τα σύμβολα που χρησιμοποιήθηκαν και επεξηγούνται παρακάτω.



Εικ.5. Επεξήγηση συμβόλων Component διαγράμματος

1. Component	Μία οντότητα που απαιτείται για να εκτελεστεί μια λειτουργία. Προσφέρει και χρησιμοποιεί λειτουργικότητα από τα interfaces και άλλα components. (“Component Diagram Tutorial,” n.d.)
2. Dependency	Δείχνει ότι ένα κομμάτι του συστήματος εξαρτάται από κάποιο άλλο. (“Component Diagram Tutorial,” n.d.)
3. Provided Interface	Αναπαριστά όλα τα inputs και τα materials που παρέχει ή λαμβάνει ένα component. Αντιπροσωπεύει τα interfaces τα οποία ένα component παρέχει ως δεδομένα στο required interface ενός άλλου. (“Component Diagram Tutorial,” n.d.)

Αντιπροσωπεύει και τα services τα οποία το component χρειάζεται ώστε να φέρει εις πέρας τις υποχρεώσεις του (Visual Paradigm, n.d.)

4. Required Interface

Αναπαριστά όλα τα inputs και τα materials που παρέχει ή λαμβάνει ένα component. Αντιπροσωπεύει τα interfaces από τα οποία το component χρειάζεται δεδομένα ώστε να φέρει εις πέρας τις λειτουργίες του. ("Component Diagram Tutorial," n.d.)

Αντιπροσωπεύει και τα services τα οποία το component χρειάζεται ώστε να φέρει εις πέρας τις υποχρεώσεις του (Visual Paradigm, n.d.)

3.2.2 Επεξήγηση διαγράμματος

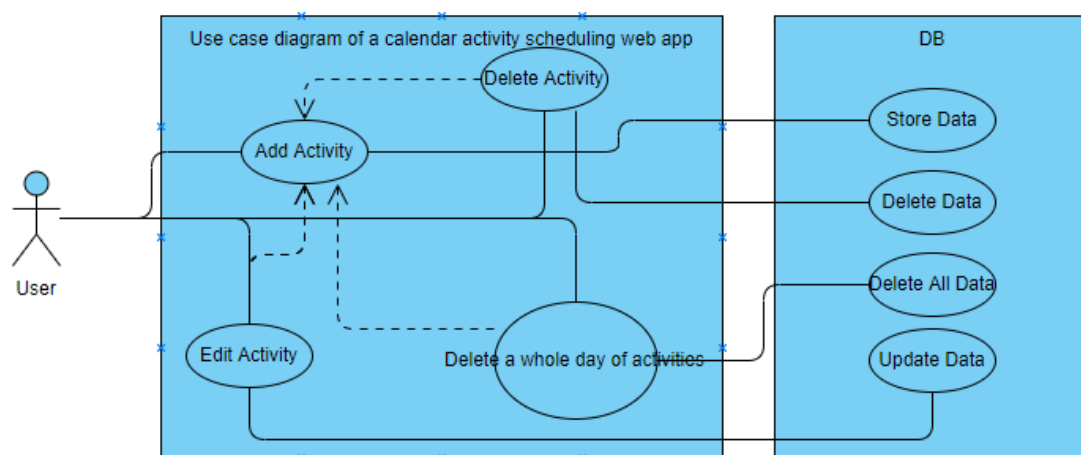
Η εφαρμογή αποτελείται από 5 συνολικά components από τα οποία το ένα λέγεται appComponent και εξαρτάται από τα templates των calendar και menu components για να μπορέσει να εμφανίσει το template του στην οθόνη. Όλα μαζί ανήκουν σε ένα module το AppModule. Τα 5 components μεταξύ τους είναι ξένα και η επικοινωνία γίνεται μέσω services.

Τα services αυτά δεν είναι τίποτα άλλο παρά μια κλάση γραμμένη σε typescript και χρησιμοποιεί τον decorator @Injectable ώστε να είναι διαθέσιμη από άλλα components. Ο Injector φτιάχνει instances αυτού του service χρησιμοποιώντας τη 'new' και τα κάνει inject στις κλάσεις των components από τα οποία καλείται το service. (Angular, n.d.-a)

Το service DataService είναι υπεύθυνο για την επικοινωνία μεταξύ του component calendar και του component MatDialog. Το πρώτο περιλαμβάνει το view του ημερολογίου και το δεύτερο τη φόρμα συμπλήρωσης στην οποία τα δεδομένα που θα πρέπει να μεταφερθούν είναι απλά η ημερομηνία που θα επιλέξει ο χρήστης και θα πρέπει να ενημερώσει τη φόρμα γι' αυτό. Το Post είναι απλά ένα interface που σκοπός του είναι να πακετάρει σε μορφή obj τα δεδομένα από π.χ. τη φόρμα συμπλήρωσης. Το MatDialog ανήκει στο module Material της Angular το

οποίο φέρει το component MatDialog από το οποίο χρησιμοποιούμε μια μέθοδο open() που ως παράμετρό της περνάμε το όνομα της κλάσης του component που θέλουμε να ανοίξουμε σε ένα pop up dialog. Το γεγονός ότι συμπεριλαμβάνεται στο διάγραμμα με τη χρήση του συμβόλου ενός interface είναι διότι εκτός των services μπορούμε να αναπαραστήσουμε και όσα materials έχουν χρησιμοποιηθεί, όπως έχει ήδη προαναφερθεί. Το http service είναι εκείνο που κάνει import στην κλάση του τον httpClient της Angular ο οποίος είναι ένα component του module http για ασύγχρονες κλήσεις σε api. Σκοπός αυτού του service είναι να κάνει κλήση στο rest api που έχει υλοποιηθεί σε express ώστε να γίνουν οι κατάλληλες CRUD ενέργειες μετέπειτα στη βάση. Κάθε φορά που η βάση επικυρώσει την επιτυχή καταχώρηση, τροποποίηση ή διαγραφή στέλνεται πίσω στον client μήνυμα επιτυχούς καταχώρησης, τροποποίησης ή διαγραφής μέσω http σαν response. Σε αντίθετη περίπτωση στέλνεται ανάλογο μήνυμα.

3.2.3 Συσχέτιση συστατικών συνδεσμολογίας με απαιτήσεις συστήματος



Εικ.6. Use Case Diagram

Στο παραπάνω use case διάγραμμα φαίνονται οι αρχικές βασικές απαιτήσεις του συστήματος που υλοποιήθηκαν και οι σχέσεις που έχουν μεταξύ τους.

3.2.3.1 Τι είναι ένα use case diagram με λίγα λόγια ?

Ένα UML use case διάγραμμα είναι ο κύριος και βασικός τρόπος απεικόνισης των απαιτήσεων ενός νέου υπό υλοποίηση συστήματος/λογισμικού. Επικεντρώνονται στην προσδοκώμενη

συμπεριφορά (στο τι θέλουμε) και όχι στον ακριβή τρόπο με τον οποίο αυτό θα συμβεί/υλοποιηθεί (όχι στο πως). Μας βοηθάει στο να κατασκευάσουμε ένα σύστημα από άποψη πλευράς χρήστη.

- Συνοψίζει κάποιες από τις σχέσεις μεταξύ των cases, των actors και του συστήματος.
- Δεν εστιάζει στη σειρά την οποία ακολουθούν τα διάφορα βήματα που εκτελούνται για να επιτύχουν τους στόχους του κάθε case.

Στη UML ανήκει στην κατηγορία των Behavioral Diagrams (Visual Paradigm, n.d.)

3.2.3.2 Συσχέτιση των δύο παραπάνω διαγραμμάτων (απαιτήσεων-συστατικών)

Οι βασικές απαιτήσεις του συστήματος είναι 4 και συγκεκριμένα η πρώτη απαίτηση είναι να μπορεί ο χρήστης να εκχωρεί δραστηριότητες στη βάση δεδομένων, έπειτα να μπορεί να τροποποιήσει κάποια αν το επιθυμεί και τέλος να μπορεί να διαγράψει όποια δεν χρειάζεται πια. Η τελευταία απαίτηση έχει δύο επιμέρους απαιτήσεις. Ο χρήστης πρέπει να μπορεί να διαγράψει μία συγκεκριμένη από μία μέρα δραστηριότητα εάν αυτός το επιλέξει ή να μπορεί να διαγράψει όλες τις δραστηριότητες για μια συγκεκριμένη μέρα μαζικά χωρίς να χρειάζεται να τις διαγράψει μία-μία χωριστά.

Οι επιπλέον απαιτήσεις του συστήματος που θα πρέπει να υλοποιηθούν και δεν παρουσιάζονται στο διάγραμμα είναι να μπορεί το σύστημα και να είναι σε θέση να ταξινομεί σε σωστή σειρά τις δραστηριότητες ώστε ο χρήστης να μπορεί να τις εκτελέσει έτσι ώστε να τις προλάβει όλες. Το σύστημα είναι υπεύθυνο για κάθε τροποποίηση να προτείνει νέα ώρα έναρξης της μέρας. Οι δραστηριότητες θα πρέπει σε περίπτωση αλλαγής σειράς να μην συγκρούονται με ώρες έναρξης άλλων ειδικά σημαντικών δραστηριοτήτων και αν είναι να ταξινομηθούν διαφορετικά να ταξινομηθούν πριν την ώρα έναρξής τους και όσο το δυνατόν να έχουν ολοκληρωθεί καθώς η ώρα πλησιάζει την αρχική ώρα έναρξής τους. Επίσης κάποιες επιπλέον απαιτήσεις της εφαρμογής είναι και θα πρέπει να προσφέρει στον τελικό χρήστη επίπεδα αποφυγής λάθους εκτελώντας κάποιου ελέγχους πριν την εκτέλεση εντολών.

Τέλος, εξίσου σημαντικές είναι και τα στυλ αλληλεπίδρασης που πρέπει να υλοποιηθούν για να υπάρχει η σχετική αρμονία μεταξύ του χρήστη και της μηχανής.

Για να πραγματοποιηθεί η πρώτη λοιπόν απαίτηση της καταχώρησης η εφαρμογή στην ουσία αυτό που κάνει είναι να απευθύνεται στο FormDialog component του Component διαγράμματος και αυτό με τη σειρά του να φέρει εις πέρας και να εκτελέσει όλες τις λειτουργίες που του έχουν ανατεθεί. Η αρμοδιότητα του τελειώνει όταν πακετάρει μέσω του post interface τα δεδομένα της φόρμας σε μορφή αρχικά obj κι έπειτα μέσω του http service σε JSON η οποία είναι μια εύκολα διαβιβάσιμη μορφή μέσω http ώστε να καταφέρει να επικοινωνήσει με τη βάση. Μόλις όλα εγκριθούν επιστρέφει τον έλεγχο πίσω στο menu component του διαγράμματος.

Η δεύτερη απαίτηση της τροποποίησης στοιχείου βρίσκεται στο menu component που και αυτό με τη σειρά του εκμεταλλεύεται το http service με ανάλογο τρόπο όπως και το προαναφερθέν component για να κάνει την ενημέρωσή του στη βάση. Σε αυτό το component λοιπόν έχει υλοποιηθεί μια μέθοδος που λέγεται update().

Η τρίτη απαίτηση της διαγραφής μιας συγκεκριμένης δραστηριότητας σχετίζεται με το menu component κι αυτή (το οποίο έχει πάρει το όνομά του παρεμπιπτόντως από το γεγονός ότι όλες οι απαιτήσεις υλοποιούνται σε αυτό άμεσα ή έμμεσα). Για να ολοκληρωθεί όμως αυτή η απαίτηση το component αυτό χρειάζεται ένα άλλο component και συγκεκριμένα το confirmDialog component ώστε να επιβεβαιωθεί η ενέργεια.

Η τέταρτη απαίτηση που είναι η μαζική διαγραφή δραστηριοτήτων για μια συγκεκριμένη μέρα ακολουθεί την ίδια λογική και συσχετίζεται με τα ίδια components που συσχετίζεται και η τρίτη.

3.2.3.2.1 Λίγα λόγια για τις απαιτήσεις που δεν σχετίζονται με κάποιο συστατικό της εφαρμογής.

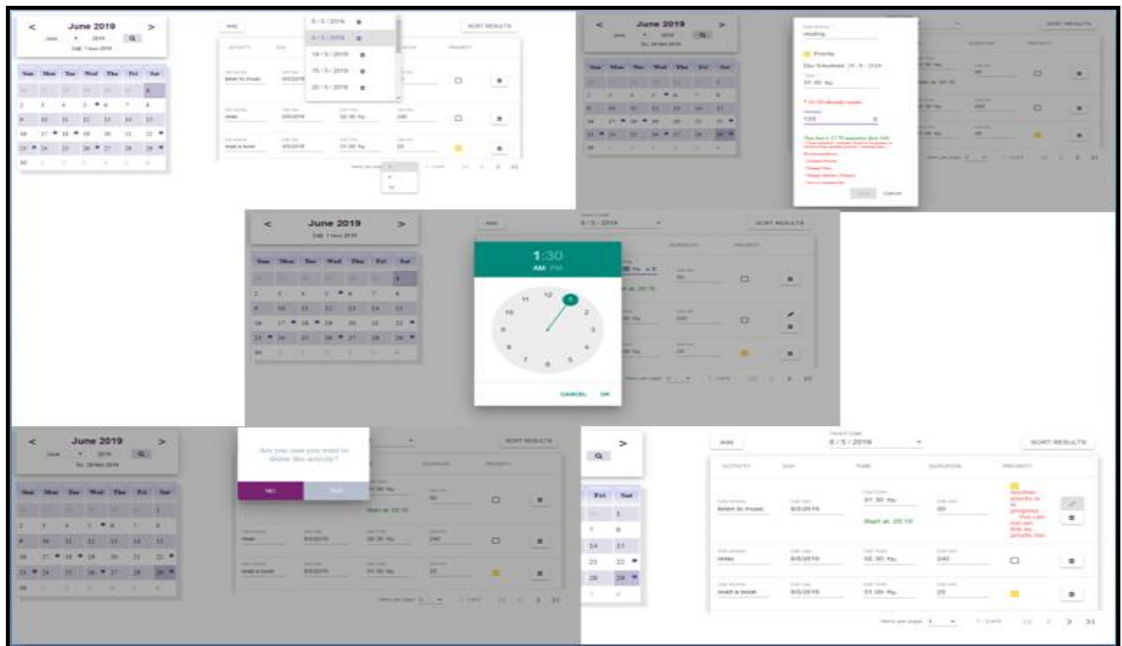
Τα επιπλέον χαρακτηριστικά σε ένα λογισμικό, σύστημα, εφαρμογή ή πρόγραμμα τα προσθέτουν αυτές οι επιπλέον απαιτήσεις που θα πρέπει να ικανοποιούνται πριν το τελικό προϊόν φτάσει στον τελικό χρήστη.

Στην εφαρμογή που υλοποιήθηκε οι επιπρόσθετες αυτές απαιτήσεις είναι οι εξής:

Στυλ Αλληλεπίδρασης

- Φόρμα συμπλήρωσης με κινούμενα και χρωματιστά labels πάνω από κάθε input field της φόρμας.
- Pop Up Dialog Boxes για την φόρμα συμπλήρωσης καθώς και την επιλογή ώρας μέσα από τη φόρμα ή τον πίνακα για τροποποίηση της ώρας σε δεύτερο χρόνο.
- Κρυφό μενού στο view του ημερολογίου για επιλογή μήνα και έτους, καθώς και εμφάνιση ενός σημαδιού στο κελί της ημέρα στο view του ημερολογίου η οποία έχει καταχωρημένες δραστηριότητες προς εκτέλεση.
- View ημερολογίου με bold γραφή στον αριθμό της τρέχουσας ημερομηνίας και με διαδραστικότητα με το πέρασμα του κέρσορα από πάνω του.
- Έλεγχοι που εμφανίζονται πριν κάθε καταχώρηση με προτεινόμενες λύσεις ώστε να επιτευχθεί η εντολή.
- Ειδικό select button στο πάνω μέρος του πίνακα για την επιλογή ημερομηνίας από τη βάση.
- Buttons για την εμφάνιση της φόρμας συμπλήρωσης, την ταξινόμηση του πίνακα, και την τροποποίηση και διαγραφή δραστηριοτήτων.
- Snackbar με την απάντηση επιτυχίας ή αποτυχίας εκτέλεσης κάποιας εντολής στη βάση.
- Δυνατότητα επιλογής αποτελεσμάτων ανά όσα στοιχεία ανά σελίδα επιθυμεί ο χρήστης και ειδική σελιδοποίηση.
- Δυναμική Υπενθύμιση υπολειπόμενων διαθέσιμων λεπτών για μια μέρα μέσα στη φόρμα συμπλήρωσης.
- Ρολόι προτεινόμενης ώρας έναρξης μέσα στον πίνακα σε περίπτωση ταξινόμησης και μετάθεσης των δραστηριοτήτων σε νέες ώρες.
- Σύγχρονη εμφάνιση και ήπια χρώματα.

Στην παρακάτω εικόνα φαίνονται τα στυλ αλληλεπίδρασης με το χρήστη.



Εικ.7. Στυλ Αλληλεπίδρασης με το χρήστη και έλεγχοι

3.3 Λειτουργικότητα κάθε συστατικού και η επικοινωνία του με τα υπόλοιπα

Σε αυτή την παράγραφο θα αναλύσουμε με λεπτομέρεια τη λειτουργικότητα την οποία έχει αναλάβει να εκτελέσει κάθε συστατικό της εφαρμογής αναφερόμενοι ακόμα και σε λογική η οποία έχει χρησιμοποιηθεί μέσα σε μεθόδους όπως επίσης και τον τρόπο με τον οποίο ανταλλάσσουν μηνύματα μεταξύ τους και με τη βάση δεδομένων.

Το calendar component αναλαμβάνει να εμφανίσει το view του ημερολογίου στην οθόνη όπως έχει ήδη προαναφερθεί. Για να το επιτύχει αυτό, αυτό που στην ουσία υλοποιείται στη λογική του είναι πέρα από το να δημιουργεί τρεις πίνακες έναν για τον προηγούμενο μήνα, έναν για τον τρέχοντα και έναν για τον επόμενο οι οποίοι διαβάζονται και οι τρεις από το view είναι λοιπόν η λογική μιας μεθόδου η οποία μόλις καλείται η μέθοδος της φόρτωσης των ημερών (από πάτημα ανάλογου button στο ημερολόγιο πάνω) να καλείται αμέσως μετά εκείνη και να κάνει κλήση στη βάση για να 'δεί' ποιες νέες ημερομηνίες προστέθηκαν στη βάση ώστε να τις εντοπίσει στο ημερολόγιο και να βάλει δίπλα μια μικρή κουκκίδα υπενθύμισης. Το

ημερολόγιο υλοποιεί στη λογική του και μεθόδους που εκτελούνται κάθε φορά που ο χρήστης θέλει να αλλάξει μήνα ή να εμφανίσει το κρυφό μενού επιλογής μήνα και έτους. Επίσης υλοποιεί μια μέθοδο η οποία πάντα παίρνει την τιμή της ημέρας την οποία επέλεξε ο χρήστης και ενημερώνει το data service με τις νέες τιμές ώστε από αυτό να τις διαβάσει όποιο άλλο component χρειαστεί τις τιμές αυτές.

To data service είναι ένα αρχείο .ts το οποίο υλοποιεί μια κλάση σε typescript και εισάγει την βιβλιοθήκη rxjs, από την οποία εμείς μετά έχουμε τη δυνατότητα να χρησιμοποιήσουμε τα observables για να επιτύχουμε την επικοινωνία μεταξύ ξένων components. Συγκεκριμένα δημιουργείται ένα behavior subject το οποίο θέτεται ως observable και πιο κάτω μια μέθοδος που παίρνει μια παράμετρο για να ενημερώνει το observable αυτό από όποιο σημείο του κώδικα καλείται. Μετά από ένα άλλο σημείο του κώδικα μπορούμε να κάνουμε subscribe στο observable αυτό και να πάρουμε τη νέα ενημέρωση κάθε φορά που καλούμε τη subscribe. Με αυτή τη λογική το calendar component ενημερώνει το observable με το που ο χρήστης επιλέξει ημερομηνία και το dialog component που υλοποιεί τη φόρμα κάνει subscribe και καθώς ενεργοποιείται η subscribe του δίνεται πρόσβαση στην νέα τιμή κάθε φορά του observable και έτσι κάθε φορά που ο χρήστης επιλέγει μέρα από το ημερολόγιο και μετά πατάει το add button στην φόρμα του εμφανίζεται η επιλογή μέρας που έκανε. Στο τέλος αυτής της παραγράφου γίνεται μια μικρή σύντομη αναφορά και γνωριμία με αυτή τη βιβλιοθήκη και τα observables.

Στο dialog component υπάρχει και άλλη λογική που υλοποιείται όπως αυτή των ελέγχων για πιθανή σύγκρουση ωρών. Μέθοδοι που πυροδοτούνται καθώς ο χρήστης αλλάξει την τιμή σε κάποιο πεδίο αναλαμβάνουν να ελέγξουν αν η τιμή ώρας που καταχωρεί ο χρήστης υπάρχει ήδη ή αν η ώρα που καταχωρεί μαζί με τη διάρκεια που διαρκεί η νέα δραστηριότητα παρεμποδίζει τη διεκπεραίωση κάποιας άλλης, οπότε και ενημερώνει το χρήστη. Προτείνοντάς του διορθώσεις. Που μόλις ο χρήστης τηρήσει αμέσως του ενεργοποιεί το save button. Επίσης από τη στιγμή που κάνει κλήση στη βάση για να έχει πρόσβαση στα δεδομένα της μπορεί να αθροίσει όλες τις ήδη καταχωρημένες διάρκειες και να βρει ποια είναι η νέα μέγιστη υπολειπόμενη και επιτρεπόμενη τιμή διαθέσιμων λεπτών που μπορεί να χρησιμοποιήσει ο χρήστης για τις επόμενες του καταχωρήσεις για τη συγκεκριμένη μέρα.

Στο menu component γράφεται κώδικας για τις λειτουργίες της τροποποίησης και ταξινόμησης. Η Μέθοδος update είναι αυτή που χρησιμοποιώντας το interface Post φτιάχνει το obj με τις νέες τιμές των μεταβλητών που τροποποιήθηκαν και καλεί την update του httpservice στην οποία στέλνει το obj και το id της δραστηριότητας που θέλει να ενημερώσει ο χρήστης, ώστε αυτή

ΥΛΟΠΟΙΗΣΗ ΔΙΑΔΙΚΤΥΑΚΗΣ ΕΦΑΡΜΟΓΗΣ ΗΜΕΡΟΛΟΓΙΟΥ ΜΕ ΔΥΝΑΤΟΤΗΤΑ ΑΥΤΟΜΑΤΟΥ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

μέσω του `httpClient` να στείλει τις αλλαγές κάνοντας κλήση στο `api` και να ενημερωθούν στη βάση. Υλοποιεί μια ακόμα μέθοδο η οποία με την ίδια διαδικασία περίπου όπως και η `update` καλεί το `api` για να πάρει τις ημερομηνίες που έχουν ήδη καταχωρηθεί ώστε να διαβαστούν από το `template` στο `element select` και να μπορούν να εμφανιστούν. Μια άλλη μέθοδος αναλαμβάνει να φέρει από τη βάση όλες τις δραστηριότητες που αντιστοιχούν στη μέρα που της δόθηκε ως παράμετρος κάνοντας κι αυτή ανάλογη κλήση στο `api`. Έτσι οι δραστηριότητες εμφανίζονται αφού ταξινομηθούν πρώτα από μια ξεχωριστή μέθοδο ανάλογα με τις απαιτήσεις για ταξινόμηση που απαιτούνται. Αυτή η ταξινόμηση γίνεται μέσα στο σώμα μιας μεθόδου που λέγεται `sortTable` και η οποία αναλαμβάνει να συμπεριλάβει ελέγχους για κάθε πιθανό είδος ταξινόμησης. Αρχικά για λόγους ευκολίας και για να μπορεί να κάνει τους σωστούς ελέγχους καθώς διατρέχει τον πίνακα με τα αποτελέσματα τα οποία θα πρέπει να ελέγξει, ταξινομεί σε αύξουσα σειρά ώρας τις δραστηριότητες. Υπάρχουν κι άλλες δύο μέθοδοι που αναλαμβάνουν να διαγράψουν κάποια συγκεκριμένη ή όλες τις δραστηριότητες για μια επιλεγμένη ώρα, η πρώτη απλά έχοντας το `id` της προς διαγραφή δραστηριότητας με τα γνωστά βήματα μέσω του `httpService` και η δεύτερη έχοντας απλά την ημέρα προς διαγραφή με την ίδια τεχνική. Μία ακόμα λειτουργία εκτελείται και είναι αυτή του ανοίγματος του `popup confirm dialog` ή και του απλού `dialog` που περιλαμβάνει τη φόρμα συμπλήρωσης. Η λειτουργία αυτή γίνεται καλώντας την `open()` σε ένα instance του component `MatDialog` του module `material` που έγινε inject στον constructor της κλάσης του τρέχοντος component για να είναι διαθέσιμο σε αυτό και να μπορούμε μετά να έχουμε πρόσβαση στην `open()` όπως κι έγινε. Η μέθοδος `open` παίρνει ως παράμετρο το όνομα της κλάσης του component εκείνου που θέλουμε να ανοίξουμε μέσα σε ένα `popup dialog`.

To confirm dialog στη συνέχεια αναλαμβάνει να εκτελέσει δυο λειτουργίες ανάλογα με το ποια επέλεξε ο χρήστης. Υλοποιεί τις `deleteActivity` και `deleteAllActivitiesForThatDay` που δεν κάνουν κάτι ιδιαίτερο από το να καλούν τις αντίστοιχες μεθόδους του `httpService` από το οποίο θα γίνει και η κανονική κλήση στο `api` για τη διαγραφή. Για λόγους εξοικονόμησης κώδικα το component αυτό υλοποιεί δύο ξεχωριστές φάσεις για `delete confirmation`, μία για τη διαγραφή μιας δραστηριότητας και μια για όλων. Η διαφοροποίηση ανάμεσα στο ποιο κομμάτι κώδικα θα πρέπει να εκτελεστεί από τη λογική γίνεται στο `template` το οποίο σπάει σε δύο μέρη και αποφασίζεται ποιο θα γίνει `render` στο τέλος από ένα `*ngIf directive` το οποίο κάνει έλεγχο με χρήση μιας τιμής που έρχεται από την ανάλογη κάθε φορά μέθοδο που καλέστηκε στο `menu component`. Η τιμή αυτή μπαίνει ως παράμετρο στην `open` μαζί με το όνομα της κλάσης του component που θέλουμε να ανοίξουμε σε `popup` σαν ένα `obj` σε ένα `data property` του όλου `obj` που παίρνεται ως παράμετρος. Έτσι ανοίγει το σωστό `render` στην οθόνη και ανάλογα με το ποιο

κουμπί θα πατηθεί καλείται στη συνέχεια η σωστή μέθοδος από τη λογική που υλοποιείται στο confirm dialog component. Στην παρακάτω εικόνα φαίνεται ένα μέρος από το κομμάτι κώδικα που βρίσκεται στο menu component και στέλνει την τιμή διαφοροποίησης στο confirm όταν αυτό ανοίξει. Ο κώδικας αυτός είναι μια μέθοδος που καλείται από μια άλλη την deleteAllActivities η οποία έχει καλεστεί από τον χρήστη ο οποίος έχει πατήσει στο ανάλογο button στο view που αντιστοιχεί σε αυτήν την μέθοδο.

```
openSecondConfirmDialog(dd: any, mm: any, yy: any) {  
  this.dialog.open(ConfirmDialogComponent, {  
    width: '300px',  
    panelClass: 'confirm-dialog-container',  
    disableClose: true,  
    position: { top: '10px' },  
    data: {  
      message: [dd, mm, yy, 2]  
    }  
  });  
}
```

Απόσπασμα 1: Τιμή επιλογής confirmation dialog

Και το calendar component και το menu και το dialog και το confirm χρησιμοποιούν τη βάση οπότε και για να το επιτύχουν αυτό επικοινωνούν μέσω του http service. Το http service υλοποιεί μεθόδους όπως η deleteActivity() ή η addActivity() ή η updateActivity() ή η getActivities() οι οποίες στο σώμα τους καλούν την delete ή post αντίστοιχα κλπ. του httpClient και παίρνουν ως παραμέτρους όλες εκτός της get 3 στοιχεία. Ένα είναι το ROOT_URL το οποίο έχει την ip και το port του server που υλοποιήθηκε συγκεκριμένα σε express και είναι το rest api της εφαρμογής, το δεύτερο στοιχείο είναι το JSON που θα σταλεί στο backend κι ύστερα θα καταχωρηθεί στη βάση και το τρίτο είναι κάποια http header options τα οποία υποδηλώνουν το περιεχόμενο που στέλνεται.

Το Post είναι απλά ένα αρχείο typescript στο οποίο μέσα υλοποιείται ένα μόνο interface το οποίο ορίζει ένα obj με τις μεταβλητές που θα πρέπει να σταλούν σε τελική μορφή JSON στη βάση. Το interface αυτό παίρνει τιμές μέσα σε μεθόδους των component τα οποία θέλουν να κάνουν post στη βάση get, update ή delete. Έτσι μόλις πάρουν τις τιμές προς καταχώρηση οι μεταβλητές του περνιέται ως παράμετρος στην αντίστοιχη μέθοδο του httpService που καλούμε και αυτό με τη σειρά του που χρησιμοποιεί τον httpClient όπως έχει προαναφερθεί χρησιμοποιεί το XMLHttpRequest browser API για να εκτελέσει το HTTP request.

Στο app component δεν έχει υλοποιηθεί κάτι παρά μόνο έχουν συμπεριληφθεί τα element tags των calendar και menu components ώστε να ΥΛΟΠΟΙΗΣΗ ΔΙΑΔΙΚΤΥΑΚΗΣ ΕΦΑΡΜΟΓΗΣ ΗΜΕΡΟΛΟΓΙΟΥ ΜΕ ΔΥΝΑΤΟΤΗΤΑ ΑΥΤΟΜΑΤΟΥ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

εμφανιστούν τα views τους στον browser μέσα από το template του app component. Κάθε component ορίζει με τον @Component decorator πάνω και έξω από την υπογραφή της κλάσης του έναν selector. Με αυτόν τον selector μπορούμε να εμφανίσουμε ένα view ενός component μέσα από το view ενός άλλου αρκεί μέσα στα tags στην html να βάλουμε το όνομα του selector του αντίστοιχου component. Αυτό ακριβώς έχει γίνει και στην περίπτωση αυτή.

Τέλος, έχουμε το αρχείο με το rest api το οποίο όπως γνωρίζουμε έχει υλοποιηθεί σε express. Αυτό αναλαμβάνει να εγκαθιδρύσει πρώτα μια σύνδεση με τη βάση και μετά έχει όλες τις απαραίτητες μεθόδους για routing για όλες τις CRUD διαδικασίες όπως η PUT, GET, POST, DELETE για mysql queries στη βάση. Καθεμία από αυτές αναλαμβάνει να στείλει στο response του http μήνυμα επιβεβαίωσης επιτυχίας ή αποτυχίας μιας προσπάθειας εκτέλεσης κάποιας εντολής.

3.3.1 Μια σύντομη γνωριμία με τα Observables και τη βιβλιοθήκη Rxjs.

Observables (με λίγα λόγια)

Τα Observables παρέχουν υποστήριξη για τη μεταφορά μηνυμάτων μεταξύ των publishers και των subscribers μέσα στην εφαρμογή. Ο κώδικας που χρησιμοποιείται ως publisher δημιουργεί ένα instance ενός observable που ορίζει μια subscriber μέθοδο. Αυτή είναι η μέθοδος που θα εκτελεστεί όταν ένας consumer καλέσει τη subscribe() η οποία παίρνει έναν observer ως παράμετρο. Αυτός είναι ένα javascript object που ορίζει τους handlers οι οποίοι υλοποιούν το Observer Interface για τη λήψη των ενημερώσεων του observable. Ο handler είναι ένα object που με τη σειρά του ορίζει callback μεθόδους για να διαχειριστούν τα τρία είδη ειδοποιήσεων/ενημερώσεων που ένα observable μπορεί να στείλει τα οποία είναι:

- η next()
- η error()
- η complete()

Η κλήση της subscribe() επιστρέφει ένα subscription object το οποίο έχει την unsubscribe(), μια μέθοδο που μπορούμε να καλέσουμε για να διακόψουμε τη λήψη ενημερώσεων. Το observable δεν εκτελείται μέχρις ότου κάνουμε subscribe σε αυτό. (Angular, n.d.-b)

Η Angular λοιπόν κάνει χρήση των observables ως ένα interface για να διαχειριστεί μια ποικιλία ασύγχρονων διαδικασιών. Για παράδειγμα:

- Η κλάση του Event Emitter κάνει extend την Observable
- Το HTTP Module χρησιμοποιεί επίσης τα observables για να διαχειριστεί AJAX requests και responses.
- Το Router και Forms Module τα χρησιμοποιούν για να ακούνε και να ανταποκρίνονται σε input events που προκαλεί ο χρήστης. (Angular, n.d.-c)

Rxjs (με λίγα λόγια)

Reactive Extensions for JavaScript (RxJS) είναι μια βιβλιοθήκη reactive streams που επιτρέπει την επικοινωνία με ασύγχρονα data streams. Η RxJS μπορεί να χρησιμοποιηθεί στον φυλλομετρητή μας καθώς και στη μεριά του εξυπηρετητή χρησιμοποιώντας το NodeJS.

Ανάλυση Ορισμού...

- **Ασύγχρονα**, στο javascript σημαίνει ότι μπορούμε να καλέσουμε μια μέθοδο και να κάνουμε σε αυτήν register μια callback για να μένουμε ενημερωμένοι για το πότε τα αποτελέσματα μας είναι διαθέσιμα, ώστε εμείς μετά να μπορούμε να συνεχίσουμε με την εκτέλεση του υπόλοιπου προγράμματος και να αποφύγουμε από το να μην είναι ανταποκρίσιμη η εφαρμογή μας. Αυτό είναι χρήσιμο για AJAX κλήσεις, DOM-events, Promises, WebWorkers και WebSockets.
- **Data**, είναι raw πληροφορίες σε μορφή javascript data types όπως: Number, String, Objects (Arrays, Sets, Maps).
- **Streams**, είναι ακολουθίες από data που γίνονται διαθέσιμα με την πάροδο του χρόνου. Ως παράδειγμα μπορούμε να αναφέρουμε το γεγονός ότι δεν είναι υποχρεωτική η παραλαβή όλων των δεδομένων ενός array για να ξεκινήσουμε να τον χρησιμοποιούμε. Μπορούμε να ξεκινήσουμε να χρησιμοποιούμε το πρώτο δεδομένο ενός πίνακα περιμένοντας το επόμενο κ.ο.κ. (Sans, 2015)

4

ΛΕΠΤΟΜΕΡΕΙΕΣ ΣΥΣΤΗΜΑΤΟΣ

4.1 Εισαγωγή

Στο προηγούμενο κεφάλαιο αναφερθήκαμε γενικά αλλά και λίγο αναλυτικά για τη λειτουργία και πιο συγκεκριμένα τη λογική που αναπτύσσει μέσα του το κάθε component καθώς επίσης και για την επικοινωνία που επιτυγχάνεται μεταξύ αυτών και πως. Σε αυτό το κεφάλαιο τώρα θα εισχωρήσουμε λίγο πιο βαθιά στη λογική των components και θα ξεκινήσουμε την ανάλυση αυτή από το πρώτο στάδιο της υλοποίησης που ήταν η εγκατάσταση των τεχνολογιών και πως μπορούμε να ξεκινήσουμε μετά την εγκατάσταση ένα τέτοιο project. Θα μιλήσουμε και για τον editor που χρησιμοποιήθηκε κατά την υλοποίηση της εφαρμογής και θα εξηγήσουμε εν συντομία και με απλά λόγια δυο τρία πράγματα που θα πρέπει να γνωρίζει κανείς για το interface αυτού του editor για λόγους εξοικείωσης με αυτόν. Επίσης θα δείξουμε με ολόκληρα ειδικά επιλεγμένα κομμάτια κώδικα τη λογική που περιγράφεται και στο προηγούμενο κεφάλαιο αλλά και σε αυτό έτσι ώστε όλα αυτά να γίνουν πιο κατανοητά.

Θα γίνει περιγραφή αλγορίθμων που χρησιμοποιήθηκαν, αρχιτεκτονικών μοντέλων που ακολουθήθηκαν και όλα αυτά πάντα με αναφορές σε κομμάτια κώδικα τα οποία θα εξηγούνται γραμμή-γραμμή για ακόμα καλύτερη κατανόηση από τον αναγνώστη.

4.2 Εγκατάσταση απαραίτητων τεχνολογιών και εργαλείων

4.2.1 Εγκατάσταση NodeJS

Για να εγκαταστήσουμε το NodeJS επισκεπτόμαστε την ιστοσελίδα του <https://nodejs.org/en/download/> και κατεβάζουμε την έκδοση “Recommended For Most Users”. Έπειτα ανοίγουμε το κατεβασμένο αρχείο και ακολουθούμε τα βήματα στην οθόνη για την εγκατάσταση. Μόλις ο installer τελειώσει την εγκατάσταση ανοίγουμε το Node.js command prompt που θα βρούμε στον φάκελο εγκατάστασης του NodeJS. Σε αυτόν τον command prompt μπορούμε γράφοντας και τρέχοντας την εντολή

```
node -v
```

να δούμε την τρέχουσα εγκατεστημένη έκδοση που μόλις εγκαταστήσαμε.

Σε περίπτωση που έχουμε το NodeJS αυτό που πρέπει να κάνουμε είναι να ελέγξουμε την εγκατεστημένη έκδοση με την εντολή που προαναφέρθηκε και αν θέλουμε τα εγκαταστήσουμε μια νεότερη θα πρέπει να διαγράψουμε πλήρως την ήδη εγκατεστημένη. Για χρήστες των Windows η διαγραφή θα πρέπει να γίνει πέρα από τον πίνακα ελέγχου και από τα μονοπάτια:

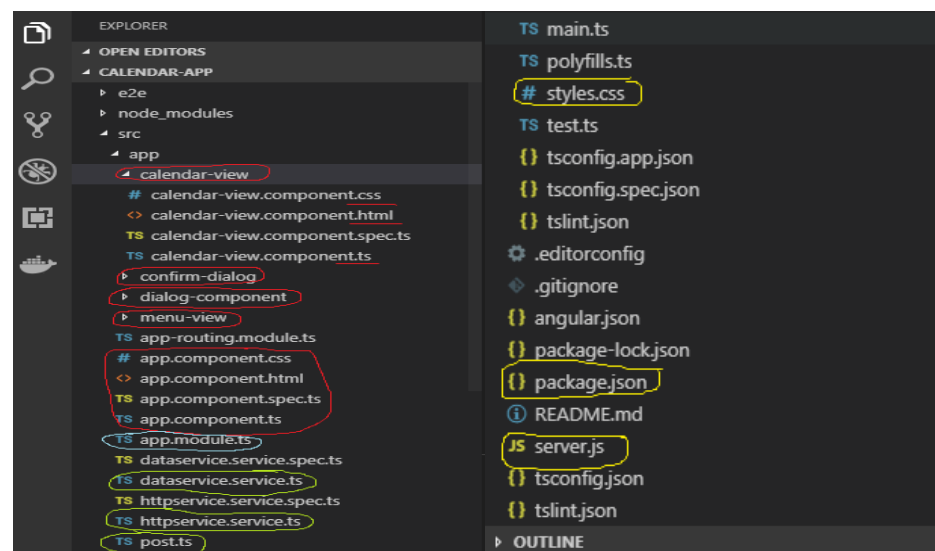
```
C:\Users\User_Name\AppData\Roaming\npm  
C:\Users\User_Name\AppData\Roaming\npm-cache
```

Επόμενο βήμα η εγκατάσταση του editor...

4.2.2 Εγκατάσταση Visual Studio

Για την εγκατάσταση του Visual Studio Code πηγαίνουμε στην ιστοσελίδα της Microsoft <https://code.visualstudio.com/download> και από εκεί κατεβάζουμε την έκδοση που μας ενδιαφέρει. Στη συνέχεια εγκαθιστούμε από το εκτελέσιμο αρχείο που κατεβάσαμε τον editor ακολουθώντας τα βήματα στην οθόνη. Μόλις η εγκατάσταση ολοκληρωθεί μπορούμε να ανοίξουμε τον editor μας και να αρχίσουμε να εξοικειωνόμαστε μαζί του.

4.2.2.1 Εξοικείωση και γνωριμία με τον editor



Εικ.8. Γνωριμία με την εργαλειοθήκη του editor

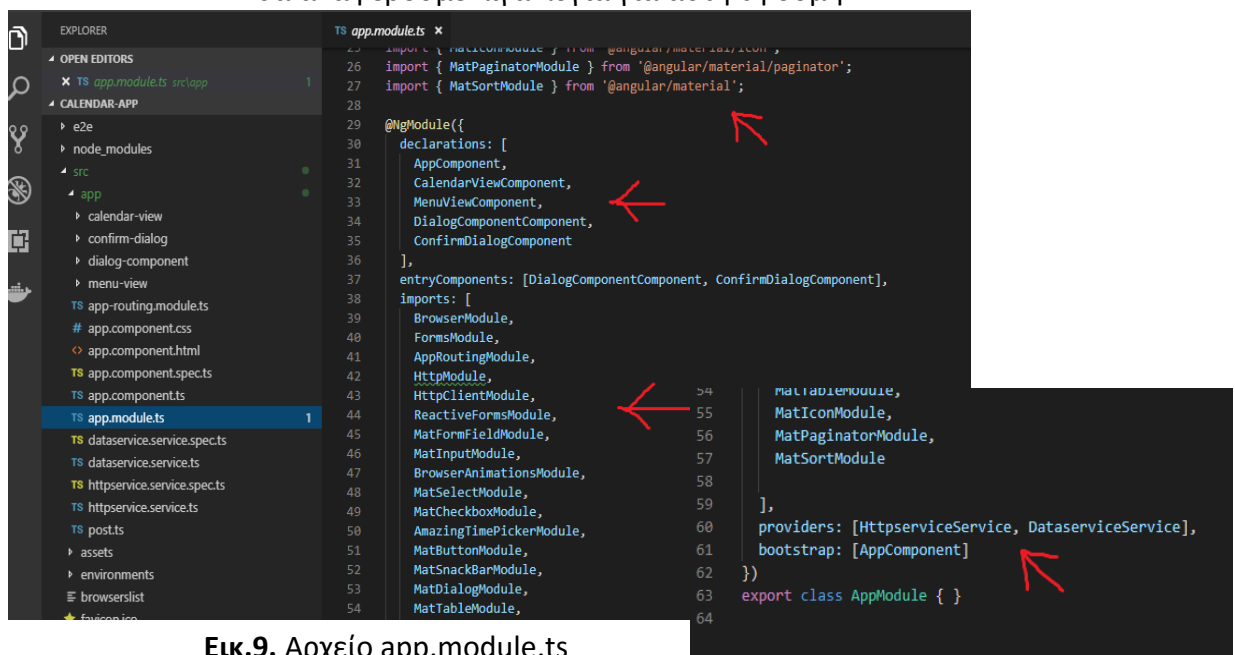
Στην αριστερή εικόνα φαίνεται η εργαλειοθήκη του editor. Εκεί μπορούμε να εντοπίσουμε και τη δομή του project μας ανά φάκελο μέσα στο navigation panel της πρώτης επιλογής

από το μενού/εργαλειοθήκη. Συγκεκριμένα φαίνονται τα 5 components μας με τα τρία βασικά αρχεία από τα οποία αποτελούνται (με κόκκινο), τα 2 services μαζί με το post interface (με ανοιχτό πράσινο) και το κεντρικό app module στο οποίο κάνουμε declare όσα νέα component ή services δημιουργούμε και όλα όσα νέα modules δημιουργούμε ή κάνουμε import. (με ανοιχτό μπλε) Στην παράγραφο 4.2.2 γίνεται αναφορά για τη δομή των components που φαίνονται σε αυτήν την εικόνα και εξηγείται παράλληλα και το μοντέλο το οποίο ακολουθείται. Τέρμα αριστερά στην ίδια εικόνα βλέπουμε ένα μενού. Από εκεί μπορούμε να μεταφερθούμε στην καρτέλα του ενσωματωμένου debugger που έχει ο editor και που προτιμάται για debugging του backend καθώς το frontend μπορεί άνετα να γίνει debug από τον debugger του chrome. Η ακριβώς επόμενη επιλογή του μενού μας μετά τον debugger με το ανάλογο εικονίδιο είναι η επιλογή των extensions. Από εκεί μπορούμε να εγκαταστήσουμε διάφορες επεκτάσεις για τον editor ώστε να του προσθέσουμε επιπλέον λειτουργικότητα. Για την εφαρμογή αυτή έχουν ήδη εγκατασταθεί δυο extensions. Το ένα είναι η επέκταση docker που προσθέτει τη λειτουργία βοήθειας στη συγγραφή containers για το πακετάρισμα του κώδικα και την εύκολη μεταφορά του σε απομακρυσμένο μηχάνημα, για τη δημιουργία images του κώδικα που είναι κλάσεις που φτιάχνουν τα containers καθώς και auto-generate των βασικών αρχείων που απαιτούνται για μια τέτοια ενέργεια containerizing της εφαρμογής και είναι το ακριβώς από κάτω εικονίδιο από το εικονίδιο των επεκτάσεων. Το δεύτερο είναι το tslint που βοηθάει στη σωστή γραφή του συντακτικού του κώδικα. Και τα δύο παρέχονται από τη Microsoft.

Στη δεξιά εικόνα φαίνεται η συνέχεια του navigation panel της εργαλειοθήκης και μπορούμε να εντοπίσουμε κι άλλα σημαντικά αρχεία για την εφαρμογή μας. Αυτά είναι κυκλωμένα με κίτρινο και ας δούμε εν συντομία σε τι βοηθάνε. Το πρώτο αρχείο με όνομα styles.css αποθηκεύει css κώδικα για styles με γενικό scope που θέλουμε να βλέπουμε από κάθε component, ώστε να είναι διαθέσιμο το στυλ σε όλα τα components. Το δεύτερο package.json είναι ένα αρχείο configuration που λέει στο node με ποιες εντολές θα ξεκινήσει την εφαρμογή μας και θα την κάνει compile όπως επίσης το ενημερώνει για όλα τα modules τα οποία έχουμε χρησιμοποιήσει στον κώδικα ή εγκαταστήσει. Εκεί γράφονται πληροφορίες τέτοιου είδους που ενδιαφέρουν και το backend και το frontend. Εμείς δεν χρειάζεται να ανησυχούμε γιατί αυτό το αρχείο δημιουργείται μόνο του από το ίδιο το node

με το που τρέχουμε την εντολή η οποία θα μας δημιουργήσει το project μας και ενημερώνει τα dependencies του κάθε φορά που εγκαθιστούμε κάτι καινούριο μέσω npm με την αντίστοιχη πάντα εντολή και μπροστά το χαρακτηριστικό --save την εντολή που δημιουργεί το project θα αναφέρουμε στη συνέχεια σε αυτό το κεφάλαιο. Χρησιμοποιώντας αυτό το αρχείο το node μπορεί σε δεύτερο χρόνο να το διαβάσει και να εγκαταστήσει εκ νέου όλα τα dependencies τα οποία απαιτούνται για να τρέξει η εφαρμογή μας αν την μεταφέρουμε σε άλλο μηχάνημα εκτελώντας την εντολή npm install. Το επόμενο σημαντικό που πρέπει να αναφέρουμε είναι φυσικά το αρχείο στο οποίο έχουμε γράψει κώδικα σε express για τη δημιουργία του rest api της εφαρμογής. Αυτό λοιπόν είναι ένα αρχείο το οποίο τοποθετείται έξω από τον κεντρικό φάκελο της εφαρμογής "src folder" και δεν είναι τίποτα άλλο παρά ένα αρχείο .js (javascript). Αυτό το αρχείο θα χρειαστεί με συγκεκριμένη εντολή του node να το τρέξουμε στο command prompt του node διότι αποτελεί το backend της εφαρμογής και πρέπει να εκτελεστεί ώστε να δημιουργηθεί και το connection με τη βάση. Στην παράγραφο 4.3 αναφέρεται η εντολή και η διαδικασία εκτέλεσης αυτού του αρχείου.

Τελευταίο αφήσαμε το αρχείο app.module.ts που φαίνεται στην αριστερή εικόνα με ανοιχτό μπλε χρώμα. Σε αυτό το αρχείο θα πρέπει κάθε φορά που κάνουμε import ένα νέο module στην angular εφαρμογή μας να το προσθέτουμε και στους κατάλληλους declaration arrays μέσα σε αυτό το αρχείο. Στην εικόνα παρακάτω απεικονίζεται η δομή του και θα αναφέρουμε λίγα λόγια για αυτή τη δομή.



Εικ.9. Αρχείο app.module.ts

ΥΛΟΠΟΙΗΣΗ ΔΙΑΔΙΚΤΥΑΚΗΣ ΕΦΑΡΜΟΓΗΣ ΗΜΕΡΟΛΟΓΙΟΥ ΜΕ ΔΥΝΑΤΟΤΗΤΑ ΑΥΤΟΜΑΤΟΥ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Υπάρχουν 5 σημεία στα οποία θα πρέπει να προσθέσουμε εμείς ένα dependency σε περίπτωση που το κάνουμε απλά import και δεν χρησιμοποιήσουμε τον npm, όπως στην περίπτωση εισαγωγής σε κάποιο component το module MatPaginatorModule που μπορούμε να κάνουμε απλά import στον κώδικά μας αφού έχουμε εγκαταστήσει το module material. Έτσι στο αρχείο αυτό δηλώνουμε το ότι θέλουμε να χρησιμοποιήσουμε αυτό το module κάνοντάς το πρώτα import και σε αυτό το αρχείο και μετά δηλώνοντας το όνομά του και στον imports array. Στον declarations array γράφουμε όλα τα components που έχουμε δημιουργήσει εμείς βάζοντας το όνομα της κλάσης τους και στον entry components όλα τα components τα οποία έχουμε κάνει import από modules της angular, όπως φαίνεται και στην εικόνα. Επίσης υπάρχει και ένας ακόμα array στον οποίο δηλώνουμε όλα τα services που φτιάχνουμε και αυτός είναι ο providers array που φαίνεται κάτω δεξιά στην εικόνα.

4.2.3 Εγκατάσταση XAMPP

Από την ιστοσελίδα του XAMPP

<https://www.apachefriends.org/index.html>

κατεβάζουμε τον installer του XAMPP και ακολουθώντας τα βήματα στην οθόνη τον εγκαθιστούμε. Μόλις η εγκατάσταση ολοκληρωθεί ανοίγουμε τον control panel του και από εκεί μπορούμε να πατήσουμε start στον apache και στη mysql, μόλις είμαστε έτοιμοι με την εφαρμογή μας για να συνδεθούμε στη βάση. Δίπλα στο start της mysql υπάρχει η επιλογή Admin που εάν πατηθεί θα μας συνδέσει με το phpmyadmin περιβάλλον όπου από εκεί θα μπορούμε να χειριστούμε τη βάση πριν χτίσουμε το api.

Τα credentials για τη βάση του xampp είναι τα εξής:

```
host   : 'localhost',
user   : 'root',
password : '',
database : 'mydb'
```

4.2.4 Εγκατάσταση Postman

Για την εγκατάσταση του Postman επισκεπτόμαστε τη σελίδα <https://www.getpostman.com/> και από εκεί κατεβάζουμε τον installer. Μόλις κατέβει τον εγκαθιστούμε και τον ανοίγουμε. Το

περιβάλλον του είναι αρκετά απλό και κατανοητό. Πάνω αριστερά από το κουμπί επιλογής http ρήματος μπορούμε να επιλέξουμε ποιο request θέλουμε να στείλουμε και δίπλα στο url bar να δώσουμε το ip και port που θέλουμε να χτυπήσουμε ώστε να εκτελεστεί η εντολή αφού πατήσουμε το send button και να πάρουμε το response. Αν είναι 200 τότε η εντολή εκτελέστηκε με επιτυχία και θα πρέπει να έχουμε τα αποτελέσματα ακριβώς στο χώρο που δίνεται από κάτω εάν όχι τότε γνωρίζουμε πριν φτιάξουμε την εφαρμογή μας ότι το api θέλει διόρθωση.

4.3 Δημιουργία νέου Project

Αφού βεβαιωθούμε ότι η εγκατάσταση του NodeJS ήταν επιτυχής και τρέξουμε και την εντολή

```
npm -v
```

η οποία θα μας εμφανίσει την τρέχουσα έκδοση του npm που έχει εγκατασταθεί και βεβαιωθούμε και πως όλα είναι εντάξει και με τον npm, μπορούμε να προχωρήσουμε στην εγκατάσταση της Angular από τον npm ο οποίος εγκαθίσταται μαζί με το NodeJS. Πηγαίνουμε λοιπόν στον ίδιο command prompt και εκτελούμε την παρακάτω εντολή

```
npm install -g @angular/cli
```

Μόλις η εγκατάσταση ολοκληρωθεί θα εμφανιστεί ανάλογο μήνυμα και θα είμαστε έτοιμοι να δημιουργήσουμε το πρώτο μας project.

Για τη δημιουργία νέου project στον command prompt του NodeJS εκτελούμε την εξής εντολή η οποία θα δημιουργήσει εκ νέου και το αρχείο package.json. Το χαρακτηριστικό ng στην αρχή κάθε τέτοιας εντολής δηλώνει ότι θα χρησιμοποιήσουμε της angular το cli για να την τρέξουμε:

```
ng new NAME_OF _THE_PROJECT
```

Είτε εκτελούμε όλες τις παραπάνω εντολές από τον command prompt του Node είτε τις εκτελούμε από τον ενσωματωμένο terminal του visual studio αφού τις εκτελέσουμε όλες μέχρι και αυτό το σημείο μετά θα πρέπει να τρέξουμε την παρακάτω εντολή για να μας μεταφέρει μέσα στον φάκελο του project μας που δημιουργήθηκε για εμάς.

```
cd NAME_OF _THE_PROJECT (με relative path)
```

ή την ίδια εντολή με absolute path στον φάκελο που εμπεριέχει το project μας. Για χρήστες των windows αυτό είναι:

```
C:\Users\PC-NAME\calendar-app
```

Η εντολή `ng new` όμως θα μας δημιουργήσει ένα μόνο βασικό component με τα τρία βασικά αρχεία που θα χρησιμοποιήσουμε για να γράψουμε κώδικα και στα νέα components που θα φτιάξουμε

- `app.component.ts`
- `app.component.html` και
- `app.component.css`

Εμείς όμως για την εφαρμογή χρειαστήκαμε παραπάνω components τα οποία δημιουργήσαμε εκτελώντας την επόμενη εντολή τόσες φορές όσες και τα components που θέλαμε να δημιουργήσουμε

```
ng g c componenet_name
```

- όπου το `ng` σημαίνει `angular`
- το `g` `generate`
- και το `c` `component`

Η εφαρμογή όμως περιλαμβάνει και κάποια `services` τα οποία και αυτά θα πρέπει με κάποια εντολή να δημιουργήσουμε. Η εντολή είναι η επόμενη και θυμίζει την προηγούμενη.

```
ng g s service_name
```

- όπου `s` σημαίνει `service`

Ένα ακόμα απαραίτητο στοιχείο που πρέπει να εγκατασταθεί είναι το `material module` της `angular`. Αυτό είναι υπεύθυνο για όλο το σύγχρονο `design` της εφαρμογής. Με την παρακάτω εντολή μπορούμε να το εγκαταστήσουμε από τον `npm` και με το `--save` να ενημερώσουμε το `package.json` για την εγκατάστασή του και μετέπειτα χρήση του.

```
npm install --save @angular/material @angular/cdk @angular/animations
```

Στη συνέχεια έγινε μια ακόμα εγκατάσταση που είναι εκείνη του `amazing time picker (atp) module` που μας προσφέρει το `pop up dialog` με το ρολόι στο `view` της τελικής εφαρμογής, η εγκατάσταση του οποίου έγινε με την παρακάτω εντολή.

`npm i atp`

Για το backend χρειάστηκε η εγκατάσταση της express η οποία έγινε κι αυτή με την ακόλουθη εντολή

`npm install express`

Και επειδή η εντολή `node server.js` που στην ουσία είναι αυτή που λέει στο node να διαβάσει το αρχείο `server.js` γραμμένο σε `express` και να το εκτελέσει έχει ένα μειονέκτημα το οποίο είναι ότι κάθε φορά που γίνεται μια αλλαγή στο αρχείο αυτό δεν την εντοπίζει με αποτέλεσμα να μην εκτελείται σωστά ο κώδικας στο backend αν δεν τερματίσουμε και ξανά εκτελέσουμε αυτήν την εντολή έτσι λοιπόν ήταν απαραίτητη η εγκατάσταση του `nodemon` που βλέπει τις αλλαγές κάθε φορά που κάνουμε `save` το αρχείο και επανεκινεί την εντολή για εμάς. Η εντολή εγκατάστασής του είναι η ακόλουθη

`npm install -g nodemon`

το `-g` δηλώνει ότι θέλουμε αυτή η εντολή να εκτελεστεί έτσι ώστε τα αποτελέσματά της να είναι διαθέσιμα `globally` στον κώδικα σε όλο το `project`.

Τέλος μετά από όλες αυτές τις εγκαταστάσεις και αφού έχουμε κάποιο κώδικα γραμμένο και για το backend και για το frontend μπορούμε να εκτελέσουμε τις δύο παρακάτω εντολές για να γίνει `compile` η εφαρμογή μας και να δούμε τα αποτελέσματα στον browser. Πριν εκτελέσουμε όμως την πρώτη θα πρέπει να θυμηθούμε να ξεκινήσουμε τη βάση δεδομένων μας από τον `control panel` του `xampp` καθώς και τον `apache`.

`nodemon server.js` (εκκινεί τον server μας και κάνει τη σύνδεση στη βάση)

`ng serve` (σερβίρει την εφαρμογή μας στον browser)

Οι δύο παραπάνω εντολές τρέχουν οι καθεμία σε ξεχωριστό `nodeJS command prompt` ή `terminal` μέσα στο `VS Code`.

Μετά είμαστε έτοιμοι να ανοίξουμε το φυλλομετρητή μας στο `localhost:4200` και να δούμε το frontend μας να λειτουργεί και να επικοινωνεί με το backend.

Στην παράγραφο 4.5 αυτού του κεφαλαίου όπου παρατίθενται κομμάτια κώδικα αποτυπώνεται άλλη μία προσέγγιση εκκίνησης της εφαρμογής η οποία παραλείπει το τελευταίο βήμα της παραπάνω προσέγγισης εκκίνησης της εφαρμογής με χρήση δύο εντολών των `nodemon server.js` και `ng serve` σε μόνο χρήση της `nodemon server.js` και ανατρέχοντας στο σύνδεσμο `localhost:3000` αντί του `localhost:4200`. Η λογική της προσέγγισης αυτής έχει ως εξής:

Τρέχουμε την εντολή

```
ng build --prod
```

Αυτή θα μας ετοιμάσει τον κώδικα για το frontend μας γραμμένο σε `angular` ώστε να είναι έτοιμος για `deploy`. Το flag `--prod` τροποποιεί τον κώδικα από το `development` στάδιο στο οποίο βρίσκεται για το `production`. Κατά αυτή τη διαδικασία αφαιρεί τα σχόλια και κάνει `'uglify'` τον κώδικα, δηλαδή δυσανάγνωστο για τον άνθρωπο. Πιο συγκεκριμένα μια μέθοδος όπως η `UpdateRecord()` θα γίνει ίσως `UR()` κατά τη διαδικασία αυτή.

Μόλις ολοκληρωθεί θα δημιουργήσει έναν φάκελο `dist` μέσα στον φάκελο του όλου `project` μας. Εκεί θα εμπεριέχεται το frontend μας έτοιμο για `distribution` σε κάποιο `cloud Paas provider` που θα τρέχει το `node` όπως το `Azure` της `Microsoft` ή το `Heroku`. Αυτός ο φάκελος περιέχει και ένα `index.html` αρχείο το οποίο θα εκκινούμε μέσα από το `server.js` αρχείο για να εκκινείται η εφαρμογή μας ανατρέχοντας στον σύνδεσμο `localhost:3000`. Πλέον θα χρειάζεται να εκτελούμε στο `command prompt` του `node` μόνο την εντολή `nodemon server.js` πριν ανατρέξουμε στον φυλλομετρητή μας. Ο κώδικας δίνεται στην παράγραφο 4.5.

4.4 Περιγραφή Αλγορίθμων που χρησιμοποιήθηκαν και προγραμματιστικών μοντέλων που ακολουθήθηκαν

4.4.1 Ο αλγόριθμος της φυσαλίδας

Ο αλγόριθμος της φυσαλίδας είναι μια κλασική περίπτωση αλγορίθμου για ταξινομήσεις στοιχείων μέσα σε έναν πίνακα. Στην εφαρμογή υλοποιήθηκε ως `nested bubble algorithm` για διπλή ταξινόμηση στοιχείων.

Ο αλγόριθμος αυτός έχει τη λογική του αν ισχύει μια συνθήκη ελέγχου τότε παίρνει το `i` στοιχείο το βάζει σε έναν προσωρινό `temp array` και στη θέση του `i` βάζει έστω το επόμενο στοιχείο

$i + 1$ ή όποιο άλλο συγκρίνει με το i και στη συνέχεια βάζει στη θέση του $i + 1$ αυτό που έβαλε στον προσωρινό πίνακα.

Πιο συγκεκριμένα με κώδικα typescript έχουμε για τη γενική μορφή του αλγορίθμου τα εξής...

```
for(let i = 0; i < this.array.length; i++) {  
    if(this.addMinutes(this.array[i].Scheduled_at,  
this.array[i].duration) > this.array[i + 1].Scheduled_at) {  
        this.temp[0] = this.array[i].Scheduled_at;  
        this.array[i] = this.array[i + 1].Scheduled_at;  
        this.array[i + 1] = this.temp[0];  
    }  
}
```

Ο παραπάνω αλγόριθμος ταξινομεί μία θέση πάνω τη δραστηριότητα με την ώρα που δημιουργεί time confliction... Όμως όταν έχουμε να κάνουμε με ώρες δεν είμαστε βέβαιοι ότι από τι στιγμή που έγινε ένα πέρασμα του πίνακα και βρέθηκε ένα το confliction time και γίνει μια μετατόπιση ότι η θέση στην οποία θα μετατοπιστεί δεν θα προκαλεί άλλη σύγκρουση ωρών με κάποια άλλη. Σίγουρα δεν θα προκαλεί ξανά με αυτήν που προκαλούσε στην αρχή αφού μετατοπίστηκε όμως μπορεί να προκαλεί με την ώρα της νέας δραστηριότητας που βρέθηκε εκ νέου από κάτω της μετά τη μετατόπιση, άρα θα πρέπει να ξανά συμπεριλάβουμε τον ίδιο έλεγχο όμως αυτή τη φορά θα γίνουν τόσες επαναλήψεις όσες ο αριθμός της θέσης που βρίσκεται η μετατοπισμένη δραστηριότητα και έως την κορυφή του πίνακα...

Άρα μια τροποποίηση του παραπάνω αλγορίθμου ώστε να θυμίζει μια περίπτωση ενός nested bubble algorithm και που υλοποιείται παρόμοια και στην εφαρμογή είναι η παρακάτω.

```
for(let i = 0; i < this.array.length; i++) {  
    if(this.addMinutes(this.array[i].Scheduled_at,  
this.array[i].duration) > this.array[i + 1].Scheduled_at) {  
        this.temp[0] = this.array[i].Scheduled_at;  
        this.array[i] = this.array[i + 1].Scheduled_at;  
        this.array[i + 1] = this.temp[0];  
    }  
    for(let y = i + 1; y < this.array.length; y++) {  
        if(this.addMinutes(this.array[y].Scheduled_at,  
this.array[y].duration) > this.array[y + 1].Scheduled_at){  
            this.temp[0] = this.array[y].Scheduled_at;  
            this.array[y] = this.array[y + 1].Scheduled_at;  
            this.array[y + 1] = this.temp[0];  
        }  
    }  
}
```

```

    }
}
}

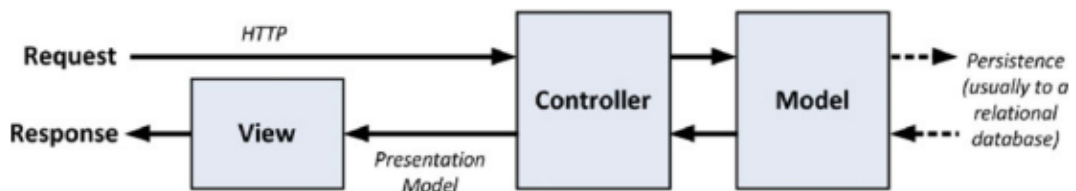
```

Η υλοποίηση του nested bubble algorithm τονίζεται με πράσινο χρώμα στον παραπάνω κώδικα...

Στην παράγραφο 4.5 παρατίθεται όλος ο κώδικας της ταξινόμησης με αναλυτικά σχόλια και επεξήγηση της λογικής.

4.4.2 Το μοντέλο MVC

4.4.2.1 Μια γνωριμία με το μοντέλο MVC



Εικ.10. Οι αλληλεπιδράσεις σε ένα αρχιτεκτονικό μοντέλο MVC (Freeman, 2016)

Κατανοώντας το μοντέλο MVC

Ως ένας υψηλού επιπέδου ορισμός για το μοντέλο αυτό θα μπορούσε να είναι ότι μία mvc εφαρμογή είναι αυτή που θα χωριστεί σε τουλάχιστον 3 μέρη.

- Το Model που εμπεριέχει ή αντιπροσωπεύει τα δεδομένα που οι χρήστες χρησιμοποιούν
- Το View που χρειάζεται για να εμφανιστούν κάποια μέρη του μοντέλου ως ui
- Ο Controller που επεξεργάζεται τα εισερχόμενα requests, εκτελεί τις λειτουργίες πάνω στο μοντέλο και επιλέγει τα views που θα εμφανιστούν στην οθόνη στο ui.

Κάθε μέρος που αποτελεί το μοντέλο αυτό είναι πλήρως καθορισμένο και αυτοεμπεριέχει ό,τι χρειάζεται, το οποίο αποτυπώνεται και ως ο "διαχωρισμός των αρμοδιοτήτων". Με άλλα λόγια η λογική που χειρίζεται τα δεδομένα στο μοντέλο συμπεριλαμβάνεται μόνο στο μοντέλο, η λογική που εμφανίζει τα δεδομένα συμπεριλαμβάνεται μόνο στο

view και η λογική που χειρίζεται τα όσα ο χρήστης καταχωρεί συμπεριλαμβάνεται μόνο στον controller.

Κατανοώντας το Model

Το Model (το M από το MVC) χωρίζεται σε δύο κύριες κατηγορίες και έτσι έχουμε το view-model που αντιπροσωπεύει τα δεδομένα που περνάμε από τον controller στον view και το domain-model που περιέχει τα δεδομένα σε ένα business domain (δηλαδή σε ένα πεδίο ορισμού της λογικής που υλοποιείται) μαζί με τις λειτουργίες και κανόνες για τη δημιουργία, αποθήκευση και διαχείριση των δεδομένων που όλα αυτά μαζί αποκαλούνται γενικά ως model logic.

το model μιας εφαρμογής που ακολουθεί αυτό το αρχιτεκτονικό μοντέλο του mvc θα πρέπει να :

- Συμπεριλαμβάνει τα domain data
- Συμπεριλαμβάνει τη λογική για τη δημιουργία, διαχείριση και τροποποίηση των domain data
- παρέχει ένα καθαρό api που θα κάνει διαθέσιμο τα data του μοντέλου και τις λειτουργίες του σε αυτό

δεν θα πρέπει να συμπεριλαμβάνει

- Τη διαθεσιμότητα στον controller ή στο view λεπτομερειών για τον τρόπο με τον οποίο αποκτήθηκαν τα δεδομένα του model
- Λογική που τροποποιεί το μοντέλο με βάση την αλληλεπίδραση του χρήστη διότι αυτό είναι δουλειά του controller
- Λογική για εμφάνιση δεδομένων στο χρήστη διότι αυτό είναι δουλειά του view

Κατανοώντας τον controller

Ο controller είναι ο συνδετικός κρίκος σε ένα μοντέλο mvc δρα ως αγωγός μεταξύ των δεδομένων του Model και του view. Ορίζει μεθόδους που παρέχουν τη λογική που θα εφαρμοστεί πάνω στα δεδομένα του model και αυτό με τη σειρά του θα επιτρέψει την παροχή δεδομένων στο view ώστε μετά αυτά να εμφανιστούν στον χρήστη

Ο controller θα πρέπει να περιέχει

- τις λειτουργίες για να ενημερώνει το model βασιζόμενος στις αλληλεπιδράσεις του χρήστη

δεν θα πρέπει να περιέχει

- τη λογική που διαχειρίζεται την εμφάνιση των δεδομένων. Αυτό το αναλαμβάνει το view
- τη λογική που διαχειρίζεται τη διατήρηση των δεδομένων. Αυτό το αναλαμβάνει το model

Κατανοώντας το view

Περιέχει τη λογική που απαιτείται για να εμφανίσει στον χρήστη ή για να παραλάβει δεδομένα από το χρήστη ώστε να επεξεργαστούν από μια μέθοδο του controller.

Το view θα πρέπει να περιλαμβάνει

- τη λογική και το markup code που απαιτείται για να παρουσιάσει τα δεδομένα στο χρήστη

δεν θα πρέπει να

- περιλαμβάνει πολύπλοκη λογική. Αυτό καλύτερα να γίνεται στον controller
- περιλαμβάνει λογική που δημιουργεί, αποθηκεύει ή διαχειρίζεται το domain model (Freeman, 2016)

4.4.2.2 Η εφαρμογή του μοντέλου MVC στην εφαρμογή που υλοποιήθηκε

Κάθε component αποτελείται από 3 αρχεία, όπως φαίνεται και στην παραπάνω εικόνα. (Εικ.8.) Ένα αρχείο απευθύνεται στο view της εφαρμογής και εκεί γράφεται κώδικας σε html με κάποιες επεκτάσεις που προσφέρει η angular όπως επιπλέον προσαρμοσμένα (custom) html elements που μας παρέχει μέσω του material module, directives όπως το *ngFor που παρέχουν δυναμικότητα στην html και τρόπους για one-way ή two-way data και property binding ή event binding για επικοινωνία του view με το model του component το οποίο είναι οι μεταβλητές που έχουμε στο δεύτερο αρχείο όπου υλοποιείται η λογική του component το model δηλαδή μαζί με τον controller τον οποίον αποτελούν οι μέθοδοι που έχουν υλοποιηθεί σε αυτό. Το

τρίτο αρχείο ενός component είναι ένα αρχείο για να γραφεί ο css κώδικας που θα καθορίσει την εμφάνιση των elements της html.

Με ποιο απλά λόγια η λογική ενός component αποτελείται από τρία χαρακτηριστικά που αποτελούν το μοντέλο MVC.

- το **Model** (μεταβλητές)
- και τον **Controller** (μέθοδοι) που υλοποιούνται σε ένα αρχείο .ts γραμμένο σε typescript
και μετά έχουμε πέρα από τη λογική του component και την εμφάνισή του στην οθόνη. Αυτό επιτυγχάνεται με...
- το **View** που είναι το template του component και είναι γραμμένο σε html και css και είναι δύο αρχεία ξεχωριστά ένα .html και ένα .css

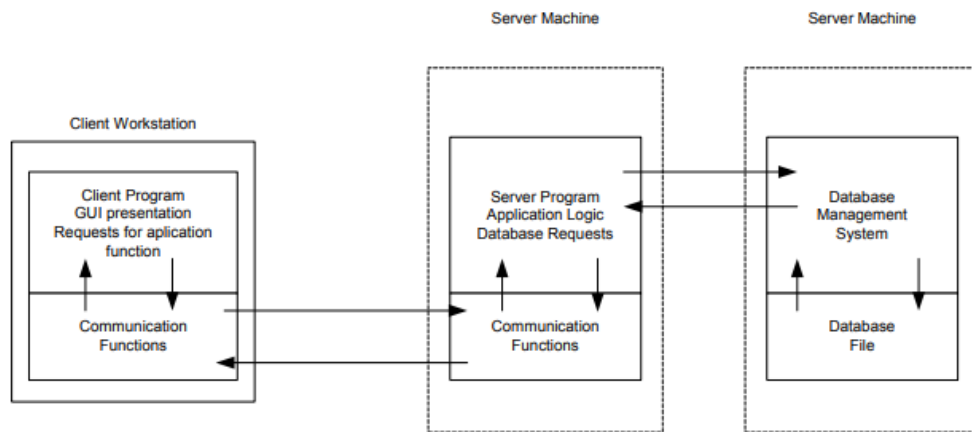
4.4.3 Αρχιτεκτονική 3-Tier

Οι αρχιτεκτονικές 3-tier client-server έχουν 3 βασικά συστατικά:

- Ένα client pc
- Έναν application server
- Έναν database server

Η λογική της αρχιτεκτονικής αυτής έχει ως εξής:

- Ο κώδικας για τον client θα περιέχει μόνο τη λογική για το presentation που σημαίνει ότι
 - ✓ Χρειάζονται λιγότεροι πόροι για το workspace του client.
 - ✓ Δεν θα επηρεαστεί ο client εάν η βάση δεδομένων αλλάξει τοποθεσία
 - ✓ Λιγότερος κώδικας για κατανομή σε client workstations
- Ένας μόνο εξυπηρετητής χειρίζεται πολλά requests από clients που σημαίνει ότι
 - ✓ Περισσότεροι πόροι διαθέσιμοι για το πρόγραμμα του server
 - ✓ Μείωση της κίνησης δεδομένων (data traffic) στο δίκτυο (Kambalyal, n.d.)



Εικ.11. 3-Tier Αρχιτεκτονική (Kambalyal, n.d.)

4.4.3.1 Η εφαρμογή της αρχιτεκτονικής αυτής στην εφαρμογή ημερολογίου

Στην εφαρμογή που υλοποιήθηκε η παραπάνω αρχιτεκτονική έχει την ακόλουθη εφαρμογή σχετικά και με την εικόνα 12.

Στη μεριά του client αρχικά από το view πυροδοτούνται οι αντίστοιχες μέθοδοι με την ενέργεια που θα εκτελέσει κάθε φορά στο view της εφαρμογής ο χρήστης. Αυτές οι μέθοδοι είναι τα communication function που βλέπουμε στην εικόνα στο client workstation και βρίσκονται μέσα στη λογική του component με άλλα λόγια στον controller που υλοποιείται και ο οποίος επεξεργάζεται δεδομένα τα οποία αποθηκεύονται στο model του component. Μετά η επικοινωνία συνεχίζει με το δεύτερο στάδιο που είναι εκείνο του backend. Από το frontend (client workstation στην εικόνα) λοιπόν πρέπει να περάσουν τα δεδομένα μας στο backend (server machine server program application logic στην εικόνα). Για να γίνει αυτό τα δεδομένα μεταφέρονται μέσω κάποιων άλλων μεθόδων που βρίσκονται σε ένα αρχείο httpService όπως έχει αναφερθεί ήδη και οι οποίες μέθοδοι αποτυπώνονται στην εικόνα ως communication function από μεριάς server αυτή τη φορά. Μετά αυτές οι μέθοδοι μεταφέρουν τα δεδομένα στη λογική του προγράμματος που αποτελεί τον server και από εκεί εισέρχονται στη βάση δεδομένων ως τελικό στάδιο του ταξιδιού τους μέσα από κάθε στρώμα της αρχιτεκτονικής.

4.5 Σημεία κώδικα και λογικής που αξίζουν να αναφερθούν

Παρακάτω παρατίθεται ένα πολύ σημαντικό κομμάτι της εφαρμογής το κομμάτι της ταξινόμησης που αποτελεί και σχεδόν όλη τη λογική πίσω από την εφαρμογή. Δίνεται αρχικά η βασική ιδέα και στη συνέχεια όλος ο κώδικας ταξινόμησης που υλοποιήθηκε σε εικόνες πλήρως σχολιασμένος για πιο ευκολία στην κατανόηση. Επίσης δίνονται και άλλα κομμάτια κώδικα που θεωρήθηκαν ενδιαφέροντα για επεξήγηση σε αυτό το κεφάλαιο.

4.5.1 Η λογική της ταξινόμησης (με βήματα)

- 1) Ταξινόμηση με αύξουσα σειρά ώρας για πιο εύκολους στη συνέχεια ελέγχους.
- 2) Εύρεση της 1^{ης} priority σε τι ώρα ξεκινάει μετά την αύξουσα σειρά ταξινόμησης με βάση την ώρα.
- 3) Ξεκίνα κάτω-πάνω έλεγχο (*βλ. Σημείωση) ταξινόμησης για όλα τα στοιχεία και αν βρεις κάποιο x να συγκρούεται με κάποιο $x-1$ στοιχείο κάνε μετατόπιση των δύο και διέκοψε τον μεγάλο έλεγχο να μπει σε έναν μικρότερο που ξεκινά από τη θέση του στοιχείου που μετατοπίστηκε στη νέα του θέση και μέχρι την κορυφή του πίνακα. Αυτός είναι ο nested bubble algorithm αν βρεις κάποιο πάλι confliction κάνε μετατόπιση αν όχι πήγαινε πίσω στον μεγάλο έλεγχο και συνέχισε την ίδια λογική για κάθε φορά που βρίσκεις confliction.

* Σημείωση: Έλεγε αν ο πίνακας έχει μέγεθος ίσο με 2 και αν ναι τότε αν υπάρχει και confliction δηλαδή η ώρα ολοκλήρωσης της δραστηριότητας x είναι $>$ της ώρα έναρξης της δραστηριότητας $x + 1$ τότε απλά βρες τη συνιστώμενη ώρα χωρίς να κάνεις κάποια ταξινόμηση. Είναι ήδη ταξινομημένα τα δύο αποτελέσματα με το 1^ο να δημιουργεί confliction με το 2^ο και απ' τη στιγμή που είναι ήδη 1^ο στον πίνακα το μόνο που μένει είναι να βρούμε νέα ώρα έναρξης να προτείνουμε σε τέτοια περίπτωση, χωρίς κάποια ταξινόμηση.

- 4) Συνέχισε με πάνω-κάτω έλεγχο τώρα, διότι ο προηγούμενος δεν βρίσκει τέτοιας φοράς conflictions, ακολούθησε την ίδια λογική με άλλη φορά όμως και πρόσθεσε τις διάρκειες όσων ταξινομούνται για να γίνει

μετά αφαίρεση απ' την ώρα έναρξης της 1^{ης} priority και να βρεθεί η νέα συνιστώμενη ώρα.

- 5) Μετά όσες ταξινόμησης είτε με την μία φορά ταξινόμησης είτε με την άλλη φορά του αλγόριθμου ξανά ταξινόμησές τες σε αύξουσα σειρά ώρας, αφού βρεις πρώτα τη θέση που πέφτει η 1^η priority, γιατί αν μετατοπιστούν θα πρέπει αυτές που είναι νωρίς να γίνουν πρώτες από όσες άλλες μετατοπίστηκαν συνολικά. Πρόσθεσε και τις διάρκειές τους.
- 6) Υπάρχει άλλη μια ειδική περίπτωση πίνακα που θέλει ιδιαίτερη ταξινόμηση. Είναι η ίδια περίπτωση με τη σημείωση στο βήμα 3 μόνο που αυτή τη φορά υπάρχει μια μικρή διαφοροποίηση. Το μέγεθος του πίνακα αν είναι όχι μόνο απλά ίσο με 2 αλλά ή και μεγαλύτερο του 2 και το 2^ο στοιχείο είναι priority ενώ το 1^ο στοιχείο ξεπερνά την ώρα έναρξης του priority τότε μην ταξινομήσεις κάτι παρά βρες την συνιστώμενη ώρα μόνο αφαιρώντας απ' την ώρα έναρξης του 2^{ου} τη διάρκεια του 1^{ου}
- 7) Στο τέλος αν παρατηρηθεί πίνακας του οποίου οι ώρες είναι εξ' αρχής σε αύξουσα σειρά χωρίς να έχει γίνει κάποιο confliction ωρών τότε ο πίνακας δεν έχει πειραχτεί με κάποια μέθοδο ταξινόμησης και άρα αν δεν βρει και κάποιο priority, τότε για να προτείνει ώρα θα πρέπει να βρει την μικρότερη ώρα και να προτείνει αυτήν για ώρα έναρξης.

4.5.2 Κομμάτια κώδικα ταξινόμησης με εικόνες και σχόλια

```
TS menu-view.component.ts x
508
509 // εδώ γίνεται sort ο πίνακας με όλα τα αποτελέσματα που βρέθηκαν καταχωρημένα για μια συγκεκριμένη μέρα
510 // με βάση την ώρα αρχικά ... στη συνέχεια θα γίνει sorting αν χρειαστεί βέβαια με βάση άλλους
511 // παράγοντες που αναφέρονται παρακάτω όπως πχ το αν μια δραστηριότητα ξεπερνάει σε χρόνο λόγω μεγάλης
512 // διάρκειας την ώρα έναρξης της επόμενης της κλπ
513 for (let i = 0; i < this.recordposts.length; i++) {
514   this.recordposts.sort((a, b) => (a.Scheduled_at > b.Scheduled_at) ? 1 : -1);
515 }
516
517
518 for (let i = 0; i < this.recordposts.length; i++) {
519
520   if (this.recordposts[i].priority === 1 && this.BikeMiaForaIdi === false) {
521     // η firstPriorityLinesScheduledAt αποθηκεύει την τιμή της ώρας της πρώτης priority
522     // δραστηριότητας που εντοπίζεται στον πίνακα αφού έχει ο πίνακας γίνει sort
523     this.firstPriorityLinesScheduledAt = this.recordposts[i].Scheduled_at;
524     this.BikeMiaForaIdi = true; // κάνει task ώστε μόλις εντοπιστεί η 1η μόνο priority
525     // δραστηριότητα στον πίνακα την οποία και μόνο θέλουμε να μην ξαναψάξει για άλλη...
526   }
527
528   this.currentActivityEndsAt = this.addMinutes(this.recordposts[i].Scheduled_at, this.recordposts[i].duration);
529   this.StartingHourOfNextActivity = this.recordposts[i + 1] && this.recordposts[i + 1].Scheduled_at;
530 }
531
```

Απόσπασμα 2: Ταξινόμηση μέρος1

```

TS menu-view.components.ts x
534 | | | if ((this.currentActivityEndsAt > this.StartingHourOfNextActivity)) {
535 // αν ισχύει αυτό τότε δύο φάσεις ταξινόμησης παίζονται ανάλογα με τη συνθήκη
536 // ένα πέρασμα του αλγόριθμου από πάνω προς τα κάτω ή από κάτω προς τα πάνω ανάλογα με το πως θα
537 // γίνει το time confliction...
538
539 // σε περίπτωση που το μέγεθος του πίνακα είναι 2 τότε για να αποφευχθούν
540 // σφάλματα τύπου undefined λόγω του ότι στο for loop εμφανίζεται δείκτης πίνακα -1
541 // που φέρει ως αποτέλεσμα το πάγωμα του κώδικα πάρθηκαν ειδικές περιπτώσεις ταξινόμησης για τέτοιες
542 // περιπτώσεις πίνακα...
543 // σε αυτήν την περίπτωση έχουμε έναν πίνακα 2 στοιχείων όπου το πρώτο ξεπερνά την ώρα έναρξης
544 // του δεύτερου τότε για να προτείνουμε την νέα ώρα έναρξης αφού το conflict στοιχείο είναι ήδη στην
545 // κορυφή του πίνακα δεν χρειάζεται να κάνουμε κάποια κατάταξη παρά μόνο μια αφαίρεση της ώρας που ξεκινά
546 // το 2ο με τα λεπτά που κρατά το 1ο
547 if ((this.currentActivityEndsAt > this.StartingHourOfNextActivity) && i === 0) {
548   if (this.recordposts.length === 2) {
549     this.recommendedHour = this.SubtractMinutes(this.recordposts[1].Scheduled_at, this.recordposts[0].duration);
550     break;
551   } else {
552
553
554 // αλλιώς ταξινόμηση ότι πρέπει από κάτω προς τα πάνω... Ο λόγος που γίνεται από πάνω προς τα κάτω
555 // και κάτω προς τα πάνω ταξινόμηση είναι γιατί μπορεί να μην να γίνεται confliction μια ώρα
556 // σε μια θέση στον πίνακα χ με μια άλλη ώρα μιας άλλης δραστηριότητας στη θέση χ+1
557 // αλλά υπάρχει και η περίπτωση η ώρα στη θέση χ να συγκρούεται με την ώρα της δραστηριότητας στη θέση

```

Απόσπασμα 3: Ταξινόμηση μέρος2

```

TS menu-view.components.ts x
558 χ-1, σε τέτοια περίπτωση αν δεν είχε συμπεριληφθεί η αναποή αξιολόγηση ο κωδικός δεν μπορούσε να συγκρίνει
559 τις τιμές και η εφαρμογή κράρσε... Το confliction στην ανάποδη περίπτωση περιγράφεται μέσα στη συνθήκη if
560 που ακολουθεί και είναι ο ίδιος έλεγχος με την κανονική περίπτωση ελέγχου την από πάνω προς τα κάτω...
561 | for (let f = this.recordposts.length - 1; f > 0; f--) {
562
563   this.currentActivityStartsAt = this.recordposts[f].Scheduled_at;
564   this.startingHourOfPreviousActivity = this.recordposts[f - 1].Scheduled_at;
565   this.endingHourOfPreviousActivity = this.addMinutes(this.recordposts[f - 1].Scheduled_at, this.recordposts[f - 1].duration);
566   // εδώ υλοποιείται ο αλγόριθμος της φυσικής για την από κάτω προς τα πάνω περίπτωση...
567   // tslint:disable-next-line:max-line-length
568   | if (this.currentActivityStartsAt > this.startingHourOfPreviousActivity && this.currentActivityStartsAt < this.endingHourOfPreviousActivity
569     this.temp5[0] = this.recordposts[f];
570     this.recordposts[f] = this.recordposts[f - 1];
571     this.recordposts[f - 1] = this.temp5[0];
572
573   } // εδώ ξανα υλοποιείται ο αλγόριθμος για να γίνει πάλι ο έλεγχος για όλα τα στοιχεία μέχρι την κορυφή του πίνακα
574   // ξεκινώντας από τη θέση πλέον όπου βρέθηκε ένα confliction και προς τα πάνω...
575   | for (let y = f - 1; y > 0; y--) {
576     this.currentActivityStartsAt2 = this.recordposts[y].Scheduled_at;
577     this.startingHourOfPreviousActivity2 = this.recordposts[y - 1].Scheduled_at;
578     this.endingHourOfPreviousActivity2 = this.addMinutes(this.recordposts[y - 1].Scheduled_at, this.recordposts[y - 1].duration);
579
580     // tslint:disable-next-line:max-line-length
581   | if (this.currentActivityStartsAt2 > this.startingHourOfPreviousActivity2 && this.currentActivityStartsAt < this.endingHourOfPreviousActivity2
582

```

Απόσπασμα 4: Ταξινόμηση μέρος3

```

TS menu-view.components.ts x
583 | | | this.temp5[0] = this.recordposts[y];
584 | | | this.recordposts[y] = this.recordposts[y - 1];
585 | | | this.recordposts[y - 1] = this.temp5[0];
586 | | | }
587 | | | }
588
589 | }
590 break; // σπάσε την από κάτω προς τα πάνω ταξινόμηση
591 | }
592 | }
593
594 // κι εδώ ξεκίνα τον από πάνω προς τα κάτω έλεγχο ταξινόμησης... ο οποίος θα εκτελεστεί αν βρεθεί confliction κι ας
595 // ταξινομήθηκαν με τον από κάτω προς τα πάνω προηγουμένως... αυτό γίνεται γιατί ο προηγούμενος έλεγχος δεν βρήκε
596 // τα confliction που θα βρει ο ακόλουθος το ίδιο ισχύει και για τον ακόλουθο όμως... Το for loop του παρακάτω
597 // βρίσκεται στην αρχή-αρχή πριν τον προηγούμενο αλγόριθμο, απλά ο προηγούμενος ήταν παρεβρισκόμενος ως προέλεγχος εντός
598 // του for loop του επόμενου παρακάτω...
599 | | | this.temp[0] = this.recordposts[i];
600 | | | this.recordposts[i] = this.recordposts[i - 1];
601 | | | this.recordposts[i - 1] = this.temp[0];
602 | | | } // από κάτω γίνεται η φυσική για να συγκριθεί η activity στη θέση που βρέθηκε ακριβώς από πάνω
603 | | | // με όσες άλλες βρίσκονται από κάτω της κάθε φορά που εκείνη μετατοπίζεται μια θέση
604 | | | // πιο πάνω γιατί μπορεί να ξαναχρησιαστεί να πάει κι άλλη θέση πιο πάνω αν εξακολουθεί να
605 | | | // ξεπερνά η διάρκεια της την ώρα έναρξης της επόμενης της γι' αυτό η αρίθμηση στο από κάτω
606 | | | // for ξεκινάει από y = i - 1...

```

Απόσπασμα 5: Ταξινόμηση μέρος4

ΥΛΟΠΟΙΗΣΗ ΔΙΑΔΙΚΤΥΑΚΗΣ ΕΦΑΡΜΟΓΗΣ ΗΜΕΡΟΛΟΓΙΟΥ ΜΕ ΔΥΝΑΤΟΤΗΤΑ ΑΥΤΟΜΑΤΟΥ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

```

TS menu-view.component.ts x
605 // ξεπερνά η διάρκεια της την ώρα έναρξης της επόμενης της γι αυτό η αρίθμηση στο αλλο κάτω
606 // for ξεκινάει από y = i - 1...
607 for (let y = i - 1; y > 0; y--) {
608   this.currentActivityOfSecondSortEndsAt = this.addMinutes(this.recordposts[y].Scheduled_at, this.recordposts[y].duration);
609   this.StartingHourOfNextActivityOfSecondSort = this.recordposts[y + 1].Scheduled_at;
610 // οι έλεγχοι έχουν την κλασική συνθήκη ελέγχου μέσα τους... Τα ονόματα των μεταβλητών βοηθούν στην κατανόηση της σύγκρισης...
611 if ((this.currentActivityOfSecondSortEndsAt > this.StartingHourOfNextActivityOfSecondSort)) {
612   this.temp2[0] = this.recordposts[y];
613   this.recordposts[y] = this.recordposts[y - 1];
614   this.recordposts[y - 1] = this.temp2[0];
615   this.addduration = this.addduration + this.recordposts[y].duration;
616   this.addduration = this.addduration + this.recordposts[y - 1].duration;
617
618   // στην recommendedHour αποθηκεύεται η τιμή που παράγεται από την αφαίρεση
619   // της ώρας έναρξης της πρώτη priority δραστηριότητας που εντοπίζεται στον πίνακα μείον
620   // το άθροισμα των διαρκειών όλων των από πάνω από την priority δραστηριοτήτων
621   // έτσι βρίσκουμε την ώρα που ο χρήστης πρέπει να ξεκινήσει την μέρα του μιας που δραστηριότητες
622   // προγραμματισμένες για άλλες ώρες έχουν τοποθετηθεί πάνω από ήδη πρώινές ίσως δραστηριότητες
623   this.recommendedHour = this.SubtractMinutes(this.firstPriorityLinesScheduledAt, this.addduration);
624
625   this.addduration = 0; // προσθέτει όλες τις διάρκειες των δραστηριοτήτων πριν το πρώτο
626   // priority του πίνακα...
627
628 }

```

Απόσπασμα 6: Ταξινόμηση μέρος5

```

TS menu-view.component.ts x
630 }
631 }
632 // *** εδώ ξεκινά άλλο ένα sorting μόνο στα στοιχεία του πίνακα πάνω από το 1ο priority
633 // ώστε να ταξινομηθούν με βάση την ώρα (αύξουσα σειρά) γιατί όλες μετατοπίστηκαν θέλω να
634 // γίνουν όσο πιο κοντά στην ώρα τους γίνεται γιατί θα πρέπει όλες να γίνουν ή στην πραγματική τους ώρα
635 // ή αν μετατοπιστούν θα πρέπει να γίνουν νωρίτερα αλλά και πάλι αυτές που είναι νωρίς πρώτες από άλλες...
636
637 // βρες τώρα και τι θέση του 1ου priority ΠΡΟΣΟΧΗ πριν βρίκαμε την ώρα έναρξης του 1ου priority τώρα βρίσκουμε τη θέση
638 for (let i = 0; i < this.recordposts.length; i++) {
639   if (this.recordposts[i].priority === 1 && this.BikeMiaFora === false) {
640     this.BikeMiaFora = true;
641     this.thesiProtouPriorityStoixeiou = i;
642     break;
643   }
644 }
645
646 // και υλοποίησε ότι λέει το σχόλιο πιο πάνω... το σχόλιο αυτό ξεκινά με ***
647 for (let i = 0; i < this.thesiProtouPriorityStoixeiou - 1; i++) {
648   if (this.recordposts[i].Scheduled_at > this.recordposts[i + 1].Scheduled_at) {
649     this.temp3[0] = this.recordposts[i];
650     this.recordposts[i] = this.recordposts[i + 1];
651     this.recordposts[i + 1] = this.temp3[0];
652 // εννοείται ότι πρέπει να προσθέτουμε κάθε φορά σε κάθε loop που βρισκόμαστε τη διάρκεια για να αφαιρέσουμε στο τέλος το άθροισμα
653 // που θα βρεθεί από την ώρα έναρξης της 1ης priority...

```

Απόσπασμα 7: Ταξινόμηση μέρος6

```

TS menu-view.component.ts x
652 // εννοείται ότι πρέπει να προσετούμε κάθε φορά σε κάθε loop που βρισκόμαστε τη διάρκεια για να αφαιρέσουμε στο τέλος το άθροισμα
653 // που θα βρεθεί από την ώρα έναρξης της 1ης priority...
654 if (this.addduration === 0) {
655   this.addduration = this.addduration + this.recordposts[i].duration;
656   this.addduration = this.addduration + this.recordposts[i + 1].duration;
657   this.recommendedHour = this.SubtractMinutes(this.firstPriorityLinesScheduledAt, this.addduration);
658
659   this.addduration = 0; // προσθέτει όλες τις διάρκειες των δραστηριοτήτων πριν το πρώτο
660   // priority του πίνακα...
661 }
662
663 this.idd = this.recordposts[0].id; // σε αυτή τη μεταβλητή βάζω το id του μηδενικού
664 // στοιχείου του πίνακα ώστε όταν στην html χρησιμοποιώ το directive ngIf για να
665 // συγκρίνω το id του κάθε row στο template με αυτό στον πίνακα που έρχεται από τη βάση
666 // να ταυτίζονται και άρα να μου εκτυπώνει την νέα προτεινόμενη ώρα και στο template
667 // μόνο στο πρώτο στοιχείο του edit box στο template...
668 // παίρνει την τιμή του id που έχει το πρώτο πάντα στη σειρά activity του πίνακα...
669
670 }
671 }
672 // είναι για όταν ο πίνακας διαμορφώνεται έτσι ώστε στη θέση 1 του πίνακα να έχουμε priority
673 // (και στη θέση 0 να μην έχουμε) τότε ο τρόπος που υπολογίζεται η συνιστώμενη ώρα αλλάζει διότι
674 // έχουμε να αφαιρέσουμε μόνο μία διάρκεια αυτή του μόνου και μοναδικού activity στη θέση 0 πάνω από το 1ο
675 // priority που βρίσκεται στη θέση 1 άρα αφαιρούμε τη διάρκεια του activity στη θέση 0 από την ώρα έναρξης του 1ου priority
676 // που βρίσκεται στη θέση 1 του πίνακα

```

Απόσπασμα 8: Ταξινόμηση μέρος7

ΥΛΟΠΟΙΗΣΗ ΔΙΑΔΙΚΤΥΑΚΗΣ ΕΦΑΡΜΟΓΗΣ ΗΜΕΡΟΛΟΓΙΟΥ ΜΕ ΔΥΝΑΤΟΤΗΤΑ ΑΥΤΟΜΑΤΟΥ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

```

TS menu-view.component.ts •
677
678 // αν δεν έβαζα τη συνθήκη if λοιπόν που ακολουθεί δεν θα ρένταρε τον πίνακα που θα είχε 1 μόνο αποτέλεσμα...
679
680 // tslint:disable-next-line:max-line-length
681 if (this.recordposts.length >= 2 && this.recordposts[1].priority === 1 && this.recordposts[0].priority === 0) {
682
683     // tslint:disable-next-line:max-line-length
684     if (this.addMinutes(this.recordposts[0].Scheduled_at, this.recordposts[0].duration) > this.recordposts[1].Scheduled_at
685         || this.recordposts[0].Scheduled_at > this.recordposts[1].Scheduled_at) {
686         // τότε πρόσθεσε τη διάρκεια της μιας και μοναδικής activity πάνω από το priority στη θέση 0 του πίνακα
687         // ώστε να βρεις τη νέα συνιστώμενη ώρα έναρξης...
688
689         this.addduration = this.addduration + this.recordposts[0].duration;
690         this.recommendedHour = this.SubtractMinutes(this.recordposts[1].Scheduled_at, this.addduration);
691         // εμφάνισε τη συνιστώμενη ώρα μόνο στο πρώτο row του πίνακα...
692         // το πρώτο στοιχείο του πίνακα έχει ένα id βρες το στον πίνακα μετά στην html
693         // που θα πρέπει να είναι και εκεί το πρώτο στοιχείο οπότε ταύτισέ τα ταίριαξέ τα
694         // και εκτίπωσε μόνο στη row που αντιστοιχεί στο κοινό id τη συνιστώμενη ώρα...
695
696         this.idd = this.recordposts[0].id;
697         this.addduration = 0; // προσθέτει όλες τις διάρκειες των δραστηριοτήτων πριν το πρώτο
698         // priority του πίνακα...
699     }
700

```

Απόσπασμα 9: Ταξινόμηση μέρος8

```

TS menu-view.component.ts •
700
701 }
702 for (let i = this.recordposts.length - 1; i > 0; i--) {
703     // αν οι ώρες μέσα στον πίνακα εντοπιστούν να είναι σε αύξουσα σειρά σημαίνει ότι ο πίνακας δεν πειράχτηκε από κάποιο
704     // είδος ταξινόμησης γιατί προφανώς δεν χρειάστηκε... και άρα αφού δεν έγινε κάποιο sorting μην
705     // εκτυπώσεις προτεινόμενη ώρα στον παρακάτω κώδικα, ο οποίος αναφέρεται στην περίπτωση πρότασης προτεινόμενης ώρας
706     // σε περίπτωση που δεν βρει κάποιο priority...
707
708     if (this.recordposts[i - 1].Scheduled_at > this.recordposts[i].Scheduled_at) { // έλεγχος αν όντως οι ώρες είναι σε αύξουσα σειρά...
709         this.HoursAreNotInAscendingOrder = true;
710         break;
711     }
712 }
713 // διατρήχει τον πίνακα και αν τον βρει χωρίς κανένα priority τότε για να να προτείνει ώρα
714 // θα πρέπει να βρει την μικρότερη ώρα και να προτείνει αυτήν για νέα ώρα έναρξης της μέρας...
715 for (let i = 0; i < this.recordposts.length; i++) {
716
717     if (this.recordposts[i].priority === 1) {
718         this.thereArePriorities = true;
719     }
720 }
721 if (this.recordposts[i].Scheduled_at < this.previousMinimumHour && this.recordposts.length > 1 && this.HoursAreNotInAscendingOrder) {
722     this.previousMinimumHour = this.recordposts[i].Scheduled_at;
723 }
724 }

```

Απόσπασμα 10: Ταξινόμηση μέρος9

```

TS menu-view.component.ts •
725 }
726 if (this.thereArePriorities !== true && this.recordposts.length > 1 && this.HoursAreNotInAscendingOrder) {
727     this.recommendedHour = this.previousMinimumHour;
728     this.idd = this.recordposts[0].id;
729 }
730
731

```

Απόσπασμα 11: Ταξινόμηση μέρος10

Ολόκληρος ο κώδικας βρίσκεται στο GitHub Repository στον ακόλουθο σύνδεσμο
<https://github.com/dithua/NodeJs-Angular-Express-Calendar-Scheduling-Web-App>

ΥΛΟΠΟΙΗΣΗ ΔΙΑΔΙΚΤΥΑΚΗΣ ΕΦΑΡΜΟΓΗΣ ΗΜΕΡΟΛΟΓΙΟΥ ΜΕ ΔΥΝΑΤΟΤΗΤΑ ΑΥΤΟΜΑΤΟΥ
ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

4.5.3 Σερβίροντας το frontend μέσω του backend

```
JS server.js x
JS server.js ▸ publicweb
1  const express = require('express'); // Εδώ κάνουμε import την express και
2  const app = express();              // από κάτω τη χρησιμοποιούμε καλώντας την
3
4  const port = process.env.PORT || 3000; // Παρέχουμε 2 επιλογές για port μία για το port που δίνει ο
5  const publicweb = process.env.PUBLICWEB || './dist'; // host provider (azure) σε περίπτωση deploy του κώδικα
6  // και σε εναλλακτική αν δεν βρει τέτοιο port δίνεται το
7  app.use(express.static(publicweb));    // 3000. Το ίδιο σκεπτικό και για τον φάκελο που επιλέγεται
8  console.log(`serving ${publicweb}`);  // ώστε να ψάξει για το αρχείο index.html. Αν τρέχει η
9  // εφαρμογή localhost τότε ο φάκελος στον οποίο θα το
10 var allowCrossDomain = function(req, res, next) { // βρει θα είναι ο dist (distribute)
11   res.header('Access-Control-Allow-Origin', '*');
12   res.header('Access-Control-Allow-Methods', 'GET,PUT,POST,DELETE');
13   res.header('Access-Control-Allow-Headers', 'Content-Type');
14   next(); // CORS-origin configuration δηλαδή για να επιτρέψει ο browser την επικοινωνία των
15   web resources μεταξύ διαφορετικών domain που γίνονται μέσω HTTP
16
17 app.use(allowCrossDomain);
18
19
20 app.use(express.json()); // χάρης αυτό μπορούμε άνετα να κάνουμε parse το εισερχόμενο json με
21 // τον εξής τρόπο req.body.Το όνομα της μεταβλητής της οποίας θέλουμε την τιμή και όλο αυτό να το αποθηκεύσουμε
22 // σε μια καινούρια μεταβλητή έτσι var Myvar = req.body.whatever; τόσο απλά !!!
23
24
```

Απόσπασμα 12: Εκκίνηση frontend μέσω backend μέρος 1

```
24
25 // Configure MySQL connection ... db connection credentials ...
26 var mysql = require('mysql')
27 var connection = mysql.createConnection({
28   host      : 'localhost',
29   user      : 'root',
30   password  : '',
31   database  : 'mydb'
32 });
33
34 //Establish MySQL connection
35 connection.connect(function(err) {
36   if (err)
37     throw err
38   else {
39     console.log('Connected to MySQL');
40     // Start the app when connection is ready
41     app.listen(port, () => console.log(`Server is listening on port ${port}`));
42   }
43 });
44
45 // get homepage ...
46
```

Απόσπασμα 13: Εκκίνηση frontend μέσω backend μέρος 2

```
46
47 app.get('/', (req, res) => {
48   res.sendFile(`index.html`, { root: publicweb });
49 });
50 // serve the static html file που θα βρεις στον ανάλογο φάκελο κάθε
51 φορά, το όνομα του οποίου αποθηκεύεται στην μεταβλητή publicweb
52 όπως έχει οριστεί παραπάνω στον κώδικα
```

Απόσπασμα 14: Εκκίνηση frontend μέσω backend μέρος 3

ΥΛΟΠΟΙΗΣΗ ΔΙΑΔΙΚΤΥΑΚΗΣ ΕΦΑΡΜΟΓΗΣ ΗΜΕΡΟΛΟΓΙΟΥ ΜΕ ΔΥΝΑΤΟΤΗΤΑ ΑΥΤΟΜΑΤΟΥ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

5

ΑΞΙΟΛΟΓΗΣΗ ΣΥΣΤΗΜΑΤΟΣ

5.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα αξιολογήσουμε την εφαρμογή αναφερόμενοι σε σενάρια εκτέλεσης. Θα αποτυπώσουμε την επιτυχία ικανοποίησης των απαιτήσεων καθώς και αντικειμενικών στόχων που είχαν τεθεί πριν την υλοποίηση και τέλος θα δώσουμε μία σύνοψη σε μορφή checklist για το ποιοι αντικειμενικοί στόχοι κατάφεραν να ικανοποιηθούν.

5.2 Σενάρια Εκτέλεσης

5.2.1 Αρχικές συνθήκες και στόχοι σεναρίων εκτέλεσης

Μόλις η εφαρμογή είχε φτάσει σε τελικό στάδιο υλοποιήθηκαν 2 σενάρια εκτέλεσης.

Οι στόχοι των σεναρίων είναι να ελεγχθεί η ανταπόκριση σε πραγματικά δεδομένα που είχε η εφαρμογή και κατά πόσο τα εξυπηρετεί. Επίσης να φανερωθούν πιθανά κενά στη λογική του κώδικα και να διορθωθούν. Επιπλέον να δούμε κατά πόσο αυτή η εφαρμογή ικανοποιεί τις απαιτήσεις που τέθηκαν στην αρχή λειτουργικές καθώς και μη λειτουργικές.

Στο πρώτο σενάριο εκτέλεσης παρατηρήθηκε ότι υπήρχε ένα κενό στον κώδικα ταξινόμησης που έφερνε σαν αποτέλεσμα το να μην γίνει έλεγχος σε περίπτωση που ο χρήστης καταχωρούσε μια δραστηριότητα τύπου priority και η ώρα ολοκλήρωσής της ξεπερνούσε την ώρα έναρξης ενός ήδη υπάρχοντος priority. Αυτή η περίπτωση αποτελεί λογικό σφάλμα της εφαρμογής και γι' αυτό διορθώθηκε αμέσως ώστε να αποφευχθεί. Επίσης προστέθηκαν και οι ανάλογοι έλεγχοι που θα απέτρεπαν τον χρήστη από κάποιου τέτοιου είδους καταχώρηση η οποία θα οδηγούσε σε μη ικανοποιητικά αποτελέσματα. Στην παρακάτω εικόνα φαίνεται ο έλεγχος που προστέθηκε με επιτυχία και διόρθωσε το κενό του κώδικα.

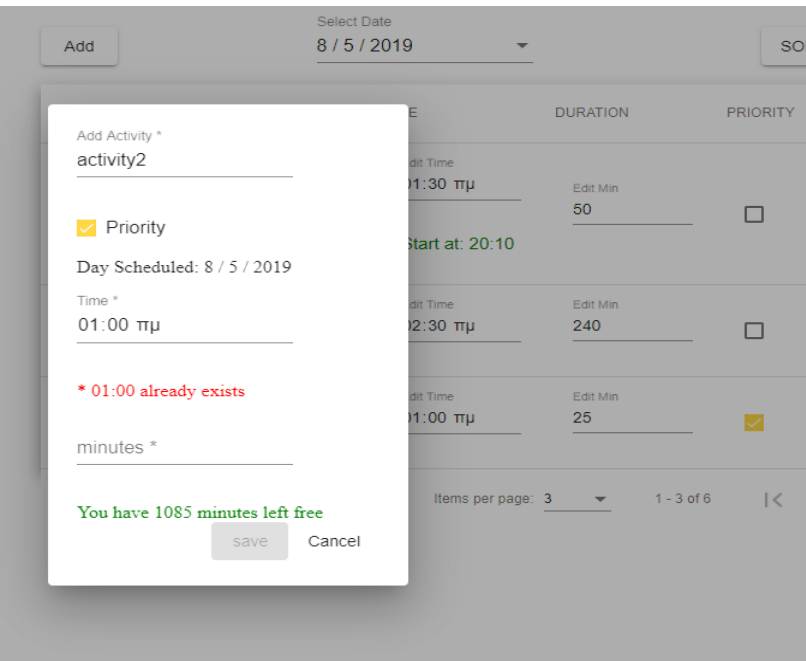
Στο δεύτερο σενάριο εκτέλεσης αποφασίστηκε ότι είναι βολικό να προστεθεί για διευκόλυνση του χρήστη η επισήμανση των υπολειπόμενων διαθέσιμων λεπτών της επιλεγμένης ημέρας μέσα στη φόρμα συμπλήρωσης.

Επίσης προστέθηκαν και τελειοποιήθηκαν όλοι οι προαπαιτούμενοι επιπλέον έλεγχοι που ίσως να μην είχαν υπολογιστεί ώστε να ενημερώνεται ο χρήστης πριν γίνει η οποιαδήποτε καταχώρηση ή τροποποίηση όπως δείχνουν οι παρακάτω εικόνες όλων των ενημερώσεων περί σφαλμάτων που γίνονται και έχουν συμπεριληφθεί στη λογική της εφαρμογής. Τέλος σε ένα συγκεκριμένο σφάλμα που ενημερώνει το χρήστη και προτείνει λύσεις πάνω στα δεδομένα τα οποία καταχώρησε και προκαλούν σύγχυση στην εφαρμογή προτείνει και αφαίρεση ελάχιστων λεπτών διάρκειας ώστε να διορθωθεί η φόρμα και να καταχωρηθεί. Τα ενδεχόμενα αυτά απεικονίζονται στις παρακάτω εικόνες.

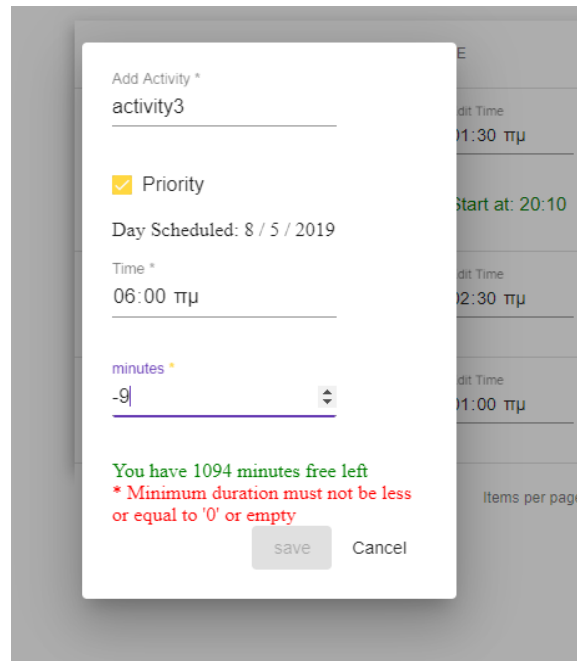
Εικ.12. Διόρθωση κενού του κώδικα που βρέθηκε μετά από το 1^ο σενάριο εκτέλεσης

Εικ.13. Έλεγχος 1 σενάριο εκτέλεσης 2^ο

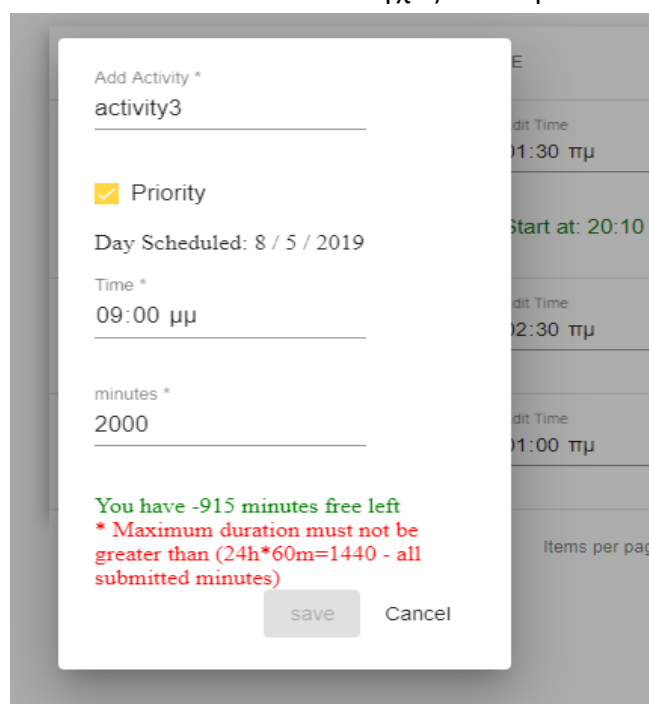
ΥΛΟΠΟΙΗΣΗ ΔΙΑΔΙΚΤΥΑΚΗΣ ΕΦΑΡΜΟΓΗΣ ΗΜΕΡΟΛΟΓΙΟΥ ΜΕ ΔΥΝΑΤΟΤΗΤΑ ΑΥΤΟΜΑΤΟΥ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ



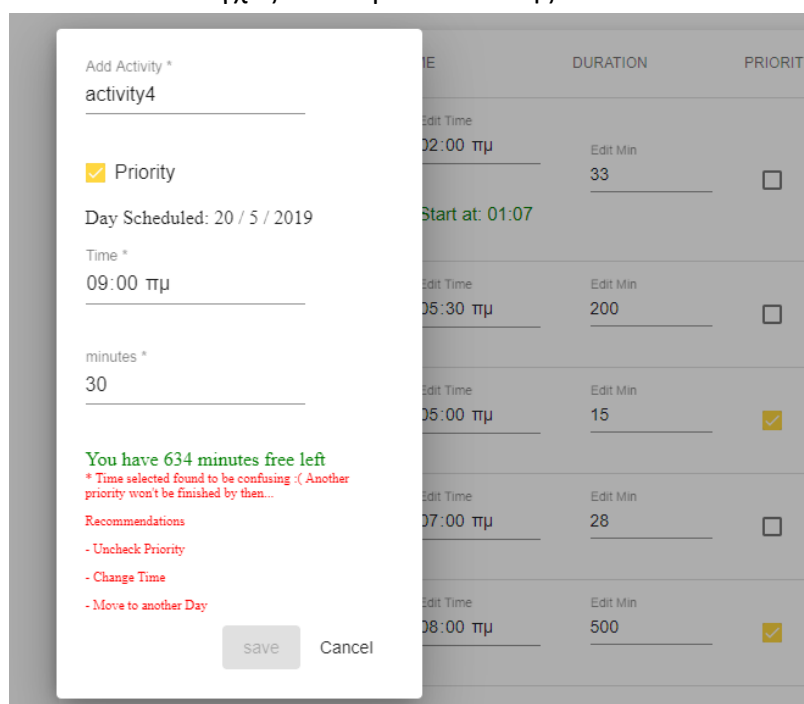
Εικ.14. Έλεγχος 2 σενάριο εκτέλεσης 2^ο



Εικ.15. Έλεγχος 3 σενάριο εκτέλεσης 2^ο



Εικ.16. Έλεγχος 4 σενάριο εκτέλεσης 2^ο



Εικ.17. Έλεγχος 5 σενάριο εκτέλεσης 2^ο

ACTIVITY	DAY	TIME	DURATION	PRIORITY
Edit Activity listen to music	Edit Day 8/5/2019	Edit Time 01:00 πμ Time already exists or there's a conflict Start at: 20:10	Edit Min 50	<input type="checkbox"/>
Edit Activity relax	Edit Day 8/5/2019	Edit Time 02:30 πμ	Edit Min 240	<input type="checkbox"/>
Edit Activity read a book	Edit Day 8/5/2019	Edit Time 01:00 πμ	Edit Min 25	<input checked="" type="checkbox"/>

Εικ.18. Έλεγχος 6 σενάριο εκτέλεσης 2^ο

ACTIVITY	DAY	TIME	DURATION	PRIORITY
Edit Activity listen to music	Edit Day 8/5/2019	Edit Time 01:00 πμ Start at: 20:10	Edit Min 50	<input type="checkbox"/>
Edit Activity relax	Edit Day 8/5/2019	Edit Time 02:30 πμ	Edit Min 240	<input type="checkbox"/>
Edit Activity read a book	Edit Day 8/5/2019	Edit Time 01:00 πμ	Edit Min 200	<input checked="" type="checkbox"/>
Edit Activity read a 2nd book	Edit Day 8/5/2019	Edit Time 02:00 πμ	Edit Min 12	<input checked="" type="checkbox"/>

Εικ.19. Έλεγχος 7 σενάριο εκτέλεσης 2^ο

ΥΛΟΠΟΙΗΣΗ ΔΙΑΔΙΚΤΥΑΚΗΣ ΕΦΑΡΜΟΓΗΣ ΗΜΕΡΟΛΟΓΙΟΥ ΜΕ ΔΥΝΑΤΟΤΗΤΑ ΑΥΤΟΜΑΤΟΥ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Add

Select Date
 8 / 5 / 2019

SORT RESULTS

ACTIVITY	DAY	TIME	DURATION	PRIORITY	
Edit Activity listen to music	Edit Day 8/5/2019	Edit Time 01:30 πμ Start at: 20:10	Edit Min 50	<input checked="" type="checkbox"/> <div style="color: red; font-size: 0.8em;"> Another priority is in progress ... You can not set this as priority too </div>	<div style="border: 1px solid #ccc; padding: 2px; text-align: center;"> Edit </div> <div style="border: 1px solid #ccc; padding: 2px; text-align: center;"> Delete </div>
Edit Activity relax	Edit Day 8/5/2019	Edit Time 02:30 πμ	Edit Min 240	<input type="checkbox"/>	<div style="border: 1px solid #ccc; padding: 2px; text-align: center;"> Edit </div> <div style="border: 1px solid #ccc; padding: 2px; text-align: center;"> Delete </div>
Edit Activity read a book	Edit Day 8/5/2019	Edit Time 01:00 πμ	Edit Min 25	<input checked="" type="checkbox"/>	<div style="border: 1px solid #ccc; padding: 2px; text-align: center;"> Edit </div> <div style="border: 1px solid #ccc; padding: 2px; text-align: center;"> Delete </div>

Items per page: 3
1 - 3 of 6

 < > << >>

Εικ.20. Έλεγχος 8 σενάριο εκτέλεσης 2^ο

5.2.2 Ικανοποίηση απαιτήσεων και αντικειμενικών στόχων από το σύστημα

Στις λειτουργικές απαιτήσεις συνοψίζοντας έχουμε:

- Καταχώρηση
- Τροποποίηση
- Διαγραφή μιας δραστηριότητας
- Διαγραφή όλων των δραστηριοτήτων μιας μέρας μαζικά
- Ταξινόμηση
- Προτεινόμενη ώρα
- Έλεγχοι αποφυγής λανθασμένων δεδομένων κατά την καταχώρηση
- Στυλ αλληλεπίδρασης

Στις μη λειτουργικές/δευτερεύουσες έχουμε:

- Σύγχρονο UI

Η εφαρμογή καταφέρνει να ικανοποιήσει όλες τις παραπάνω απαιτήσεις χρησιμοποιώντας σύγχρονες τεχνολογίες και εργαλεία ανάπτυξής τους που μας επιτρέπουν την εφαρμογή σύγχρονων αρχιτεκτονικών μοντέλων υλοποίησης για την πιο αποδοτική επικοινωνία των δεδομένων μεταξύ frontend και backend καθώς και την πιο σύγχρονη εμφάνισή τους στον τελικό χρήστη. Πιο συγκεκριμένα χρησιμοποιώντας όλες τις τεχνολογίες που έχουν ήδη προαναφερθεί η εφαρμογή καταφέρνει με χρήση του αλγορίθμου της φουσαλίδας καθώς και τον nested αλγόριθμο της φουσαλίδας να ταξινομήσει με χρήση διαφόρων συνθηκών ελέγχου τις δραστηριότητες με τέτοιο τρόπο όπου θα ωφελήσουν το χρήστη στη

οργάνωση του καθημερινού του προγράμματος. Επίσης αφού η ταξινόμηση ολοκληρωθεί γίνεται υπολογισμός νέας συνιστώμενης ώρας έναρξης με χρήση των αποτελεσμάτων ταξινόμησης. Με χρήση του block των reactive forms [FormControl] της angular επιτυγχάνεται το validation των υποχρεωτικών πεδίων στον πίνακα πριν γίνει η τροποποίηση στη βάση. Πιο συγκεκριμένα το FormControl είναι μια κλάση που χρησιμοποιείται για να πάρει ή να θέσει τιμές και validation στο στοιχείο της html όπως <input> και <select> tag και γίνεται import στον κώδικα από το module forms της angular. Οι reactive forms προσφέρουν τη δημιουργία και τον έλεγχο μιας φόρμας συμπλήρωσης από μέσα από τη λογική του component και όχι απευθείας στο view/template όπως κάνουν οι template driven forms, οι οποίες έχουν χρησιμοποιεί στη φόρμα συμπλήρωσης κατά την καταχώρηση για να γίνουν τα κατάλληλα validations κι εκεί και να ικανοποιηθούν οι απαιτήσεις των ελέγχων. Για τις απαιτήσεις αυτές η εφαρμογή μεταφέρει στη λογική την τιμή που καταχωρεί ο χρήστης μέσω του template και κάνει τις απαραίτητες συγκρίσεις για να ελέγξει με τα ήδη υπάρχοντα στοιχεία τη βάση εάν η κίνηση του χρήστη είναι επιτρεπτή ή όχι πριν την καταχώρηση. Η εφαρμογή επίσης χρησιμοποιεί pop up dialogs σε πολλά σημεία της χρησιμοποιώντας το MatDialog component του module material για την αλληλεπίδραση του χρήστη με τη μηχανή.

5.2.3 Σύνοψη ικανοποίησης αντικειμενικών στόχων (checklist)

Ανταπόκριση σε πραγματικά δεδομένα	ΝΑΙ
Σύγχρονη εμφάνιση	ΝΑΙ
Απλό design	ΝΑΙ
Αλληλεπίδραση με το χρήστη	ΝΑΙ

6

ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΗΣ ΣΕ ΥΠΟΔΟΜΕΣ

ΥΠΟΛΟΓΙΣΤΙΚΟΥ ΝΕΦΟΥΣ

6.1 Εισαγωγή

Μετά την υλοποίηση και ανάπτυξη της εφαρμογής και αφού την έχουμε δοκιμάσει και λειτουργεί στο τοπικό μηχάνημα μπορούμε να τη διαθέσουμε στο ευρύ κοινό κάνοντας την deploy σε κάποιον hosting cloud provider ώστε να μπορεί ο κάθε χρήστης να τη χρησιμοποιεί χωρίς να κάνει κάποια εγκατάσταση στο μηχάνημά του παρά να διατρέχει απλά στον σύνδεσμο της εφαρμογής μέσω ενός απλού προγράμματος περιήγησης και μιας σύνδεσης στο διαδίκτυο.

Σε αυτήν την παράγραφο θα αναφέρουμε ποιος τρόπος είναι ο προτιμότερος για να γίνει deploy σε κάποιον hosting cloud provider μια node διαδικτυακή εφαρμογή, θα αναλύσουμε τις τεχνολογίες που μπορούν να χρησιμοποιηθούν πως λειτουργούν καθώς και πως μπορούμε να τις εγκαταστήσουμε και θα παραθέσουμε τις εντολές που θα πρέπει να εκτελεστούν στο docker cli ώστε να δημιουργηθούν τα απαραίτητα αρχεία για το deploy.

6.2 Τεχνολογίες που θα χρειαστούν και εργαλεία

Για να κάνουμε deploy την εφαρμογή μας στο cloud θα πρέπει να επιλέξουμε έναν hosting cloud provider ο οποίος θα μας παρέχει υπηρεσίες PaaS. Ο λόγος είναι το ότι θέλουμε ο server που θα μας δώσουν να τρέχει το node και επομένως χρειαζόμαστε ο cloud provider μας να μας στήσει ένα app service resource το οποίο θα τρέχει το node ώστε να μπορούμε να κάνουμε και εμείς μετά σε δεύτερο χρόνο deploy σε αυτό το resource την εφαρμογή μας. Δύο τέτοιου είδους πολύ δημοφιλής hosting cloud providers είναι το azure της Microsoft και το Heroku. Συγκεκριμένα, εάν επιλέξουμε το azure θα πρέπει από το azure portal κατά την επιλογή subscription plan (μηνιαία συνδρομή για κάθε πόρο που χρησιμοποιούμε) να επιλέξουμε ότι θέλουμε να στήσουμε ένα web app for containers που να τρέχει το node.

Για να κάνουμε deploy την εφαρμογή μας λοιπόν στο cloud υπάρχουν δύο τρόποι που μπορούμε να επιλέξουμε. Ο ένας είναι να κάνουμε deploy όλον τον κώδικα και με την εντολή `npm install` που θα τρέξουμε στο command line της

υπηρεσίας web app που στήνουμε στον εξυπηρετητή που μας προσφέρεται από τον provider να εγκαταστήσουμε όλα τα απαραίτητα modules και dependencies που χρειάζεται η εφαρμογή μας για να τρέξει. Η εντολή αυτή διαβάσει ότι πρέπει να εγκαταστήσει από το αρχείο package.json. Μια τέτοια προσέγγιση θα μπορούσε να γίνει απευθείας από τον editor μας το vs code με τον εξής τρόπο. Αρχικά κατεβάζοντας το azure cli στον υπολογιστή μας από το σύνδεσμο

<https://docs.microsoft.com/en-us/cli/azure/install-azure-cli?view=azure-cli-latest>

και στη συνέχεια αφού θα έχουμε δημιουργήσει και έναν λογαριασμό με ένα subscription στο azure μπορούμε ανοίγοντας το azure cli να γράψουμε την εντολή αρχικά για login

az login

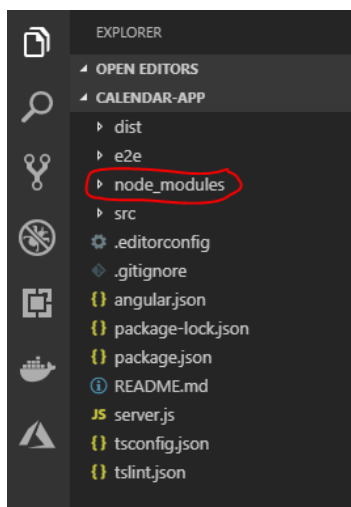
η οποία θα μας ανοίξει τον φυλλομετρητή μας για να γίνει η σύνδεση και έπειτα μπορούμε να εγκαταστήσουμε την επέκταση που δέχεται το azure cli για να κάνει push όλον τον κώδικα στο cloud. Εάν επιλέξουμε να κάνουμε deploy με την παρακάτω εντολή θα πρέπει να σημειωθεί ότι αυτό που κάνει είναι να ανεβάζει όλον τον κώδικα και αυτό σημαίνει ότι ανεβάζει και τον φάκελο node_modules του project μας με αποτέλεσμα να είναι άχρηστο να πάμε να εγκαταστήσουμε ξανά με την εντολή npm install που αναφέρθηκε προηγουμένως ξανά όλα τα modules του κώδικα. Το npm install χρησιμεύει στην περίπτωση που δεν ανεβάζουμε και αυτόν τον φάκελο ο οποίος υπάρχει στον root φάκελο του project μας από τη δημιουργία του project μας. Έτσι θα πρέπει εμείς μετά να επισκεφτούμε τη διαδρομή

`https://[dns name of the web app service].scm.azurewebsites.net/ > Debug console > CMD`

και να εκτελέσουμε εκεί την εντολή npm install. Αυτός ο φάκελος είναι αρκετά μεγάλος και πολλές φορές από τη στιγμή που οι εφαρμογή μας είναι κι αυτή μεγάλη δε συμφέρει να τον ανεβάσουμε όλον παρά μόνο τα modules που χρειαζόμαστε. Αν λοιπόν δεν ανεβάσουμε όλον τον source code μας και αποφασίσουμε πριν από αυτό να τρέξουμε την εντολή ng build –prod η οποία δημιουργεί τον φάκελο dist τον οποίο και κάνουμε serve μέσα από το server.js αρχείο τότε θα χρειαζούμε προφανώς τα modules που χρειάζεται η express την οποία χρησιμοποιεί ο server μας. Σε τέτοια περίπτωση δημιουργούμε έναν φάκελο ‘server’ στο project μας μέσα στον editor και τοποθετούμε το αρχείο server.js μετά με την εντολή cd στον terminal του editor μεταφερόμαστε στον νέο μας αυτό φάκελο βάζοντας το μονοπάτι του και στη συνέχεια μέσα σε αυτόν τον φάκελο τρέχουμε την εντολή npm init –y η οποία δημιουργεί ένα νέο κενό package.json αρχείο. Μετά τρέχουμε την εντολή npm I express –save για να εγκαταστήσει μέσα σε αυτό το package.json όλα τα απαραίτητα modules και dependencies που χρειάζεται ο server μας. Τέλος μπορούμε να ανεβάσουμε στον εξυπηρετητή του hosting provider μας τον νέο node_modules φάκελο που μόλις δημιουργήσαμε με την τελευταία εντολή ή να ανεβάσουμε μόνο το αρχείο package.json και από το cmd του cloud να τρέξουμε την

ΥΛΟΠΟΙΗΣΗ ΔΙΑΔΙΚΤΥΑΚΗΣ ΕΦΑΡΜΟΓΗΣ ΗΜΕΡΟΛΟΓΙΟΥ ΜΕ ΔΥΝΑΤΟΤΗΤΑ ΑΥΤΟΜΑΤΟΥ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

εντολή `npm install` που προαναφέρθηκε για να δημιουργήσει τον ίδιο φάκελο αν αποφασίσουμε αυτή τη προσέγγιση. Στην παρακάτω εικόνα φαίνονται όλα όσα περιέχονται στον root φάκελο του project μας. Η εντολή αυτή είναι η εξής:



```
az extension add --name webapp
```

Εικ.21. Root project folder

Αξίζει να σημειωθεί ότι το `azure cli` τρέχει μέσα από το `cmd` των windows ή το `powershell` και πριν την κάθε εντολή αρκεί να μην παραλείψω το χαρακτηριστικό `'az'`.

Οι παραπάνω εντολές καθώς εκτελούνται μέσα από `command line interfaces` καλό θα είναι πριν τις εκτελέσουμε να επιβεβαιώσουμε ότι έχουμε οδηγήσει το `command line` κάθε φορά που τον ανοίγουμε μέσα στο φάκελο του όλου project μας με την εντολή

```
cd C:\Users\PC_NAME\calendar-app
```

Η εντολή η οποία αναφέρεται παραπάνω και κάνει `deploy` όλον τον κώδικα στο `azure` δημιουργεί πρώτα ένα `resource group` και στη συνέχεια ένα `web app service`. Αυτό το `web app service` είναι ένα απλό `nodeJS web app service` που δημιουργείται για εμάς ώστε να μπορούμε να κάνουμε `deploy` την εφαρμογή μας μετά.

Σε αυτή την περίπτωση όπως ήδη γνωρίσαμε θα πρέπει να έχουμε δημιουργήσει ένα απλό `web app service` το οποίο έρχεται με μία βάση δεδομένων που τρέχει στο ίδιο `instance` και ονομάζεται `MySQL InApp`. Αυτή η βάση θέλει πριν οποιαδήποτε κίνηση ενεργοποίηση ακολουθώντας τη διαδρομή στο `azure portal`

```
All resources > your_resource_name > MySQL In App > On
```

και θα πρέπει να έχουμε τα `credentials` της για να μπορέσουμε να καταχωρήσουμε δεδομένα σε αυτήν. Αυτά μπορούμε να τα βρούμε εάν ακολουθήσουμε την εξής διαδρομή

Dashboard > All resources > your_app_service_name > Advanced Tools > Go

που θα μας βγάλει στην παρακάτω σελίδα κι εμείς θα ακολουθήσουμε την επόμενη διαδρομή από αυτή τη σελίδα

`https://[dns name of the web app service].scm.azurewebsites.net/ > Debug console
> CMD`

και μετά θα επιλέξουμε τη διαδρομή 'D:\home\data\mysql\' και βρίσκουμε το αρχείο 'MYSQLCONNSTR_localdb.ini' το οποίο έχει τα credentials για την MySQL InApp DB του Azure (Toroman, 2017)

Μπορούμε να έχουμε πρόσβαση μέσω του phpmyAdmin στη βάση στο azure ακολουθώντας τη διαδρομή

All Resources > choose_your_appservice_name > MySQL In App (από το μενού αριστερά) > Manage

Σε περίπτωση που μας εμφανίσει 'access denied' χωρίς να έχουμε κάνει κάτι εμείς και με το που πατάμε στο 'Manage' επανενεκινούμε το web app service μας και χτυπάμε το σύνδεσμο του site μας. Έπειτα ξαναπροσπαθούμε να συνδεθούμε στη βάση ακολουθώντας εκ νέου την παραπάνω διαδρομή.

Στη συνέχεια αναφέρεται ο δεύτερος τρόπος προσέγγισης ο οποίος απαιτεί νέο webapp και συγκεκριμένα απαιτεί ένα web app for containers service το οποίο θα πρέπει να δημιουργηθεί όπως αναλύεται στη συνέχεια της παραγράφου.

Κάτι τέτοιο όμως δεν προτείνεται γενικά διότι το να κάνουμε deploy όλων των κώδικα με όλα τα modules και dependencies που προφανώς θα έχει είναι κάτι το χρονοβόρο και ταυτόχρονα βαρύ για το μηχάνημα στο οποίο θα κάνουμε το deploy. Επίσης εκθέτουμε στο cloud το source code μας. Έτσι υπάρχει ένας άλλος τρόπος προσέγγισης που έχει τη λογική του να πακετάρουμε τον κώδικα σε containers και έτσι να τον μεταφέρουμε στο cloud χωρίς να μας ενδιαφέρει πολύ το τι πλατφόρμα χρησιμοποιούμε για το deploy. Με άλλα λόγια το πακετάρισμα του κώδικα σε containers είναι μια έξυπνη επιλογή καθώς κάνει συμβατό τον κώδικά μας σε μηχανήματα ανεξαρτήτου πλατφόρμας. Για αυτόν τον σκοπό χρειαζόμαστε να στήσουμε στο azure ένα web app for containers resource. Αυτή η προσέγγιση υπόσχεται και πιο lightweight εφαρμογές και ο κώδικάς μας δεν είναι εκτεθειμένος στο διαδίκτυο. Μια καλή επιλογή τεχνολογίας για αυτό το σκοπό είναι η χρήση του docker, του οποίου η λογική είναι απλή και η χρήση και λειτουργία του καθώς και όλα τα απαραίτητα συστατικά του αναλύονται στα παρακάτω βήματα.



6.2.1 Εξοικείωση με τις έννοιες docker container και docker image

Με πολύ απλά λόγια η τεχνολογία αυτή απαιτεί την ύπαρξη δυο αρχείων ενός dockerFile αρχείου και ενός docker-compose.yml αρχείου. Το πρώτο είναι εκείνο στο οποίο θα γράψουμε εντολές οι οποίες σκοπό θα έχουν να αντιγράψουν τον κώδικά μας μέσα στον container και να τον τρέξουν. Το δεύτερο είναι αυτό που θα πάρει μετά όλους τους containers που θα έχουμε δημιουργήσει και θα τους ενώσει ώστε να μπορούν να επικοινωνούν. Ας πούμε ότι έστω έχουμε μια εφαρμογή σαν την δική μας και έχουμε ένα frontend ένα backend και μία db. Για κάθε συστατικό από αυτά έχουμε δημιουργήσει και έναν φάκελο στο project μας όπου έχει μέσα τον αντίστοιχο κώδικα για το καθένα από αυτά. Μέσα σε κάθε λοιπόν φάκελο βάζουμε και από ένα dockerFile με τις ανάλογες εντολές κάθε φορά και έτσι έχουμε έναν container για το frontend έναν για το backend και έναν για τη βάση. Μετά βάζουμε το docker-compose.yml αρχείο στον root φάκελο του project μας ώστε όταν τρέξουμε τις κατάλληλες εντολές οι οποίες παρατίθενται σε επόμενη παράγραφο μαζί με την εικόνα της δομής των παραπάνω αρχείων να μπορούν αυτές να διαβάσουν το docker-compose.yml αρχείο και να φτιάξουν μια image των containers που συνθέσει το αρχείο αυτό. Πιο απλά μια image θα μπορούσε να συσχετιστεί με μια κλάση και οι containers να είναι τα αντικείμενα της κλάσης αυτής.

Η λογική των containers έχει βρει εφαρμογή προσφάτως σε αρχιτεκτονικές συστημάτων που βασίζονται στην υλοποίηση μικρών, επαναχρησιμοποιούμενων λειτουργικότητων που ενσωματώνονται σε δυναμικές ροές εργασίας.(Tserpes, 2019)

6.2.2 Τι είναι το docker hub

Το docker hub είναι ένα repository που κάποιος μπορεί να ανεβάσει τα images της πακεταρισμένης εφαρμογής του.

Ένα docker repository λοιπόν σαν το docker hub είναι ένα μέρος όπου μπορεί κάποιος να αποθηκεύσει μία ή περισσότερες εκδόσεις μιας συγκεκριμένης docker image της εφαρμογής του. Το docker hub είναι μία δημόσια αλλά και ιδιωτική υπηρεσία για hosting των docker αποθετηρίων (repositories). Υπάρχουν κι άλλα βέβαια τρίτα τέτοια μέρη σαν το docker hub. Επίσης τέτοιες υπηρεσίες φιλοξενίας αποθετηρίων ονομάζονται και registries. Ένα registry αποθηκεύει μια συλλογή από αποθετήρια. Μπορούμε να πούμε ότι ένα registry έχει πολλά αποθετήρια (repositories) και πως ένα

αποθετήριο έχει πολλές διαφορετικές εκδόσεις μιας συγκεκριμένης εικόνας (image) της εφαρμογής που διαφοροποιούνται χωριστά με την προσκόλληση ενός tag μιας ετικέτας που δηλώνει την έκδοση της εικόνας π.χ. calendar-app:latest όπου το latest δηλώνει την έκδοση και είναι σε αυτήν την περίπτωση η ετικέτα της docker image. (Janetakakis, 2018)



6.2.3 Τι είναι το azure

Η σελίδα του azure την οποία επισκεπτόμαστε για να δημιουργήσουμε λογαριασμό και να στήσουμε στη συνέχεια το web app for containers ακολουθώντας τα βήματα στην οθόνη είναι η

<https://azure.microsoft.com/en-in/overview/what-is-azure/>

Το Microsoft Azure είναι μία υπηρεσία cloud computing που δημιουργήθηκε από τη Microsoft για δημιουργία, έλεγχο, ανάπτυξη (deploy) και διαχείριση εφαρμογών και υπηρεσιών μέσω των data centers της Microsoft που χειρίζεται αυτή. Παρέχει software as a service (SaaS) υπηρεσία, platform as a service (PaaS) υπηρεσία και infrastructure as a service (IaaS) υπηρεσία και υποστηρίζει πολλές διαφορετικές γλώσσες προγραμματισμού, εργαλεία και frameworks συμπεριλαμβανομένων και Microsoft καθώς και τρίτων λογισμικά και συστήματα.

Το Azure ανακοινώθηκε τον Οκτώβρη του 2008 και κυκλοφόρησε τον Φεβρουάριο του 2010 με το όνομα “Windows Azure” που στη συνέχεια το Μάρτιο του 2014 μετονομάστηκε σε “Microsoft Azure”. (“Microsoft Azure,” 2019)

Εμείς χρησιμοποιούμε το azure ως PaaS υπηρεσία. Πιο συγκεκριμένα μια τέτοια υπηρεσία είναι μια κατηγορία cloud computing που παρέχει μια πλατφόρμα που επιτρέπει στους αγοραστές την ανάπτυξη, το τρέξιμο και τη διαχείριση εφαρμογών χωρίς την πολυπλοκότητα της υλοποίησης και συντήρησης της υποδομής (infrastructure) που συχνά σχετίζεται με την ανάπτυξη και εκκίνηση μιας εφαρμογής

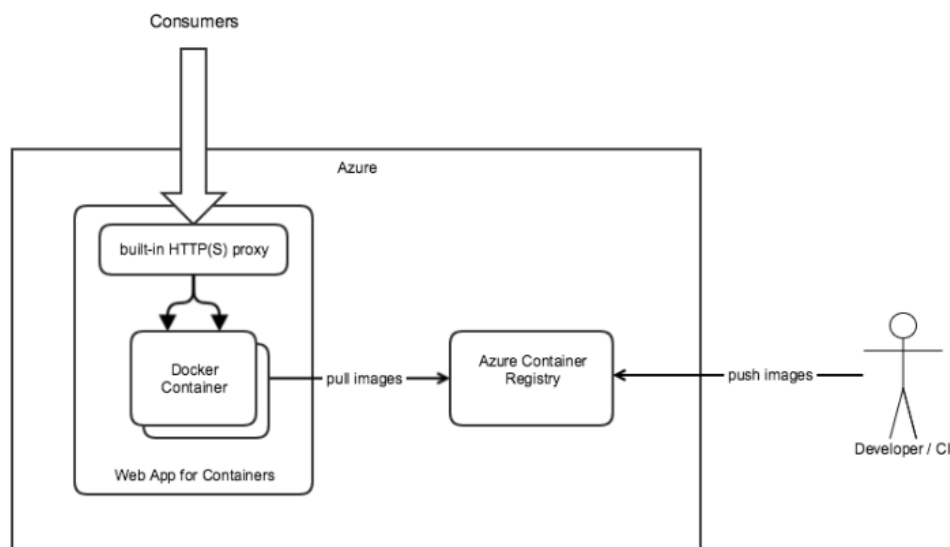
Μια PaaS υπηρεσία μπορεί να διανεμηθεί με τρεις τρόπους:

- Ως μία cloud υπηρεσία δημόσια που παρέχεται από έναν πάροχο, όπου ο αγοραστής μπορεί να έχει τον έλεγχο της ανάπτυξης της

εφαρμογής (software deployment) με ελάχιστες ρυθμίσεις (configuration options) και ο πάροχος να παρέχει το δίκτυο, τους εξυπηρετητές, τον αποθηκευτικό χώρο, το λειτουργικό σύστημα, το ενδιάμεσο λογισμικό (middleware) όπως για παράδειγμα το Java runtime, .NET runtime, integration κλπ., βάση δεδομένων και άλλες υπηρεσίες για να φιλοξενήσει την εφαρμογή του αγοραστή.

- Ως μία ιδιωτική υπηρεσία (λογισμικού) πίσω από ένα τείχος προστασίας (firewall).
- Ως ένα λογισμικό που αναπτύχθηκε που έγινε deploy με απλά λόγια, σε μία δημόσια υπηρεσία IaaS. ("Platform as a service," 2019)

6.2.4 Η λογική της αρχιτεκτονικής της φιλοξενίας της εφαρμογής στο cloud



Εικ.22. Αρχιτεκτονική φιλοξενίας της εφαρμογής στο cloud (Möller, 2018)

Η παραπάνω εικόνα δείχνει όλες τις ενέργειες που πρέπει να γίνουν για να πακετάρουμε την εφαρμογή μας και να την ανεβάσουμε στο cloud. Αφού δημιουργήσουμε τα αρχεία dockerFile και docker-compose.yml και τρέξουμε τις κατάλληλες εντολές στο docker cli ώστε να δημιουργήσουμε μέσω αυτών των αρχείων την εικόνα της πακεταρισμένης εφαρμογής μας (τα βήματα αυτά που πρέπει να γίνουν αναλύονται σε επόμενη παράγραφο) μέσω του docker cli ανεβάζουμε την εικόνα σε κάποιο registry παραδείγματα τέτοιων είναι το docker hub που επιτρέπει 1 δωρεάν repository και του azure το azure container registry το οποίο θέλει συνδρομή. Το azure όμως είναι συμβατό και με το docker hub για δική μας ευκολία. Αφού την ανεβάσουμε στο αποθετήριο της αρεσκείας μας επόμενο βήμα είναι να την κάνουμε pull από αυτό το αποθετήριο στο web app

for containers που δημιουργήσαμε προηγουμένως επισκεπτόμενοι την ιστοσελίδα του azure portal και ακολουθώντας τα βήματα στην οθόνη. Εάν η εικόνα μας δημιουργεί πολλούς containers σύμφωνα με το πώς έχει αυτή δημιουργηθεί από το docker-compose.yml αρχείο τότε θα πρέπει να έχουμε ορίσει μέσα σε αυτό το αρχείο και σε ποιο network αυτοί οι containers βρίσκονται ώστε να επικοινωνούν μέσω της ip διεύθυνσής τους ή εκτός και εάν έχουμε συμπεριλάβει σε έναν container και το frontend και το backend δηλαδή έχουμε δημιουργήσει ένα dockerfile αρχείο στο οποίο περνάμε τον κώδικα και των δύο αυτών συστατικών της εφαρμογής μας και στη συνέχεια το έχουμε δηλώσει και στο docker-compose.yml αρχείο που σε τέτοια περίπτωση δεν έχουμε πολλούς containers και άρα δεν είναι υποχρεωτικοί κάποια επικοινωνία μέσω δικτύου. Μετά επισκεπτόμαστε το σύνδεσμο της web app for containers εφαρμογής μας που βρίσκουμε στο dashboard του azure portal και έτσι βλέπουμε την εφαρμογή μας στον φυλλομετρητή διαθέσιμη μέσα από το διαδίκτυο. Οι παρακάτω εικόνες δείχνουν πως δημιουργούμε ένα web app for containers στο azure και που μπορούμε να βρούμε το σύνδεσμο. Για να αποκτήσουμε πρόσβαση στο azure portal αρκεί να επισκεφτούμε την ιστοσελίδα που αναφέρθηκε παραπάνω και να επιλέξουμε από το μενού πάνω δεξιά την επιλογή portal. Θα χρειαστεί να γίνει σύνδεση με κάποιο Microsoft e-mail. Γενικά το ui του azure είναι απλό και εύκολο στη χρήση.

Για τη δημιουργία web app for containers ακολουθούμε τα παρακάτω βήματα

Στο portal του azure

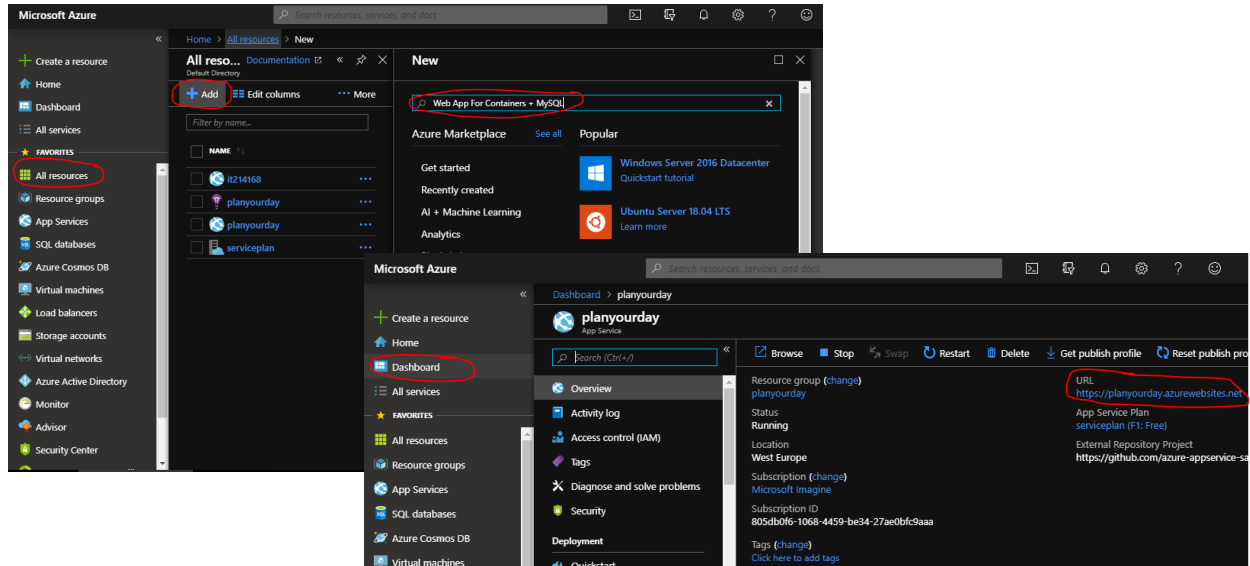
- All Resources > Add > Search 'Web App for Containers + mySQL' > Create
- Επιλέγουμε όνομα. Στην περίπτωσή μας 'planyourday'
- Κάνουμε configure τον container μας
 - Image Source: Azure Container Registry or Docker Hub
 - Registry: calendarappregistry
 - Image: calendarapp:latest (όπου latest είναι το tag της εικόνας μας που μπαίνει by default αν δεν ορίσουμε εμείς κάποιο σε εντολή που εκτελούμε για τη δημιουργία της εικόνας μέσω του docker cli που αναλύεται σε επόμενη παράγραφο)
- Ok > Create

Τέλος μπορούμε να επισκεφτούμε στον φυλλομετρητή μας τη νέα μας εγκατάσταση της web app for containers εφαρμογής είτε μέσω του dashboard του portal είτε (αφού βρούμε το δημόσιο DNS όνομα

της υπηρεσίας που μόλις δημιουργήσαμε μέσω του portal) μέσω της παρακάτω εντολής

```
curl https://planyourday.azurewebsites.net
```

(Möller, 2018)



Εικ.23. Azure Portal

6.3 Εγκατάσταση επεκτάσεων (azure & docker) στο vs code και δημιουργία repository στο docker hub

Για να μπορέσουμε να χρησιμοποιήσουμε την τεχνολογία αυτή θα πρέπει να την εγκαταστήσουμε κατεβάζοντάς την από την ιστοσελίδα

<https://www.docker.com/get-started>

Θα πρέπει να γνωρίζουμε ότι για χρήστες των windows μπορεί να εγκατασταθεί μόνο σε 64bit συστήματα. Επόμενο βήμα είναι να εγκαταστήσουμε και στον editor μας vs code το ανάλογο extension για το οποίο ανοίγουμε από το μενού αριστερά την επιλογή 'extensions' και μετά γράφουμε docker για να μας εμφανίσει τα αποτελέσματα από τα οποία επιλέγουμε αυτό το οποίο ανήκει στη Microsoft. Αφού προστεθεί στο μενού θα έχουμε πρόσβαση σε όλες μας τις εικόνες της εφαρμογής που δημιουργούμε καθώς και σε όλα τα containers και registries. Θα μπορούμε με δεξί κλικ πάνω σε μια εικόνα να την εκκινήσουμε, επανεκκινήσουμε, τερματίσουμε κλπ. αρκεί να έχουμε εκκινήσει το docker cli που εγκαταστήσαμε προηγουμένως γιατί θα χρειαστεί να τρέξουμε όλες τις εντολές για τη δημιουργία της εικόνας της εφαρμογής κλπ. τόσο απλά ! Με το που δημιουργήσουμε τους containers και αρχικά την image η επέκταση θα μας εμφανίσει όλα όσα δημιουργήθηκαν. Το ίδιο και μόλις δημιουργήσουμε ένα repository στο docker hub.

ΥΛΟΠΟΙΗΣΗ ΔΙΑΔΙΚΤΥΑΚΗΣ ΕΦΑΡΜΟΓΗΣ ΗΜΕΡΟΛΟΓΙΟΥ ΜΕ ΔΥΝΑΤΟΤΗΤΑ ΑΥΤΟΜΑΤΟΥ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Αυτό το repository θα εμφανιστεί κάτω από την κατηγορία Registries > Docker Hub. Μόλις κάνουμε upload την image στο docker hub θα εμφανιστεί στην επέκταση κάτω από την κατηγορία λυιπόν που προαναφέρθηκε και από εκεί θα πατήσουμε δεξί κλικ για να την κάνουμε deploy στο azure τραβώντας την από το docker hub.

Επίσης θα χρειαστούμε και άλλη μία επέκταση εκείνη του azure. Με παρόμοιες κινήσεις από το μενού πατάμε στις επεκτάσεις γράφουμε azure και εγκαθιστούμε εκείνο που φέρει την υπογραφή της Microsoft. Σε αυτήν την επέκταση για να μπορούμε να τη χρησιμοποιήσουμε θα πρέπει να κάνουμε σύνδεση μέσω αυτής στο λογαριασμό μας στο azure portal. Η διαδικασία είναι απλή και γίνεται μέσα από την επέκταση πατώντας στο κουμπί 'sing in' μέσα από το vs code. Μόλις γίνει η σύνδεση θα μας εμφανίσει όλα τα web app services που έχουμε στήσει. Πατώντας σε κάποιο θα μπορούμε να δούμε όλους τους φακέλους που έχουν δημιουργηθεί κατά το deploy του καθώς και όσα αρχεία κάνουμε κι εμείς deploy. Επίσης θα μας δώσει τη δυνατότητα να τροποποιούμε αρχεία μέσα από τον editor μας και μετά καθώς τα αποθηκεύουμε από τη στιγμή που τα έχουμε ανοίξει μέσα από την επέκταση θα ενημερώνονται απευθείας στο cloud. (Microsoft, 2019)

Μόλις ολοκληρώσουμε τα βήματα με τις επεκτάσεις συνεχίζουμε με τη δημιουργία repository στο docker hub. Αρχικά επισκεπτόμαστε την παρακάτω σελίδα όπου και δημιουργούμε λογαριασμό. Στη συνέχεια επιλέγουμε 'Create Repository' και είμαστε έτοιμοι να ανεβάσουμε την εικόνα σε αυτό αφού τη φτιάξουμε πρώτα μέσα στον editor.

<https://hub.docker.com/>

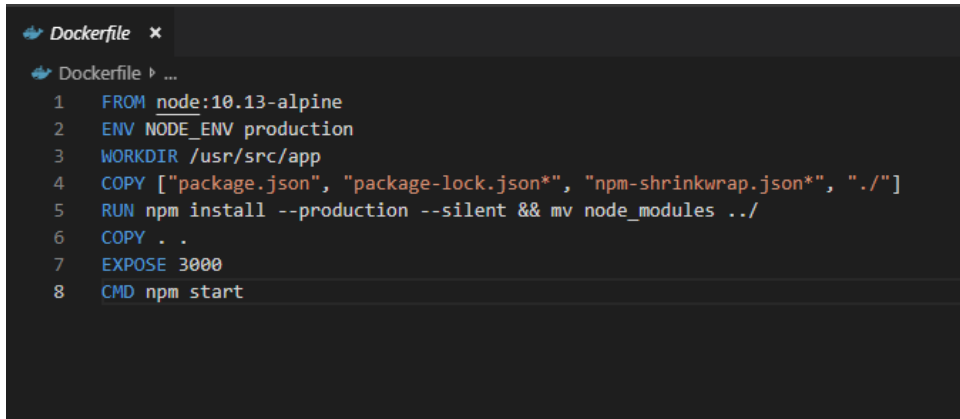
6.4 Δημιουργία dockerFile και docker-Compose.yml και ανέβασμα στο azure web app for containers

Για τη δημιουργία της εικόνας χρειάζονται οπωσδήποτε δύο αρχεία το dockerfile με τις εντολές που αντιγράφουν στον container τον κώδικα που θέλουμε να του περάσουμε και το docker-compose.yml που δημιουργεί την εικόνα μέσα από τις οδηγίες που περιλαμβάνει. Συγκεκριμένα οι οδηγίες αυτές λένε πόσους και ποιους containers έχουμε, σε ποια ports ακούνε, σε ποιο δίκτυο βρίσκεται ο καθένας από αυτούς κλπ. (το τελευταίο βοηθάει στην επικοινωνία τους η οποία επιτυγχάνεται μέσα από την ip διεύθυνσή του η οποία δεν είναι δημόσια παρά μέσω του port mapping που γίνεται στην ουσία έχουμε πρόσβαση στον container χτυπώντας την ip του host machine με το port mapping. Το port mapping αναφέρεται παρακάτω αλλά γενικά κάνει map το port στο οποίο ακούει ο container σε port του host machine (Nouredin Sadawi, 2016)).

Για να δημιουργήσουμε αυτά τα αρχεία πηγαίνουμε στο vs code και πατάμε Ctrl + Shift + P έτσι ανοίγει η command palette πάνω στην οθόνη στην οποία γράφουμε

add docker files

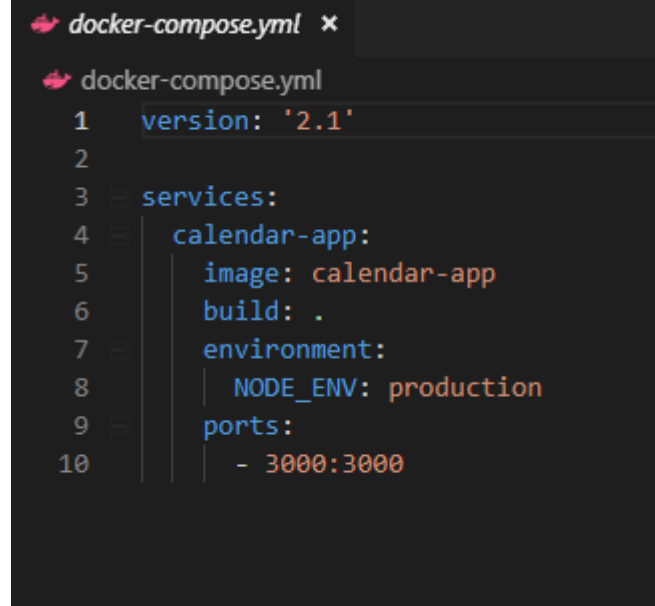
μετά πατάμε enter μετά επιλέγουμε NodeJs πατάμε πάλι enter και επιλέγουμε port 3000. Αυτό θα σημαίνει ότι ο container θα ακούει στο ίδιο port με την εφαρμογή. Άρα γίνεται port mapping κι εδώ κι αυτό είναι εμφανές μέσα στο αρχείο docker-compose.yml. Σε γενικές γραμμές το port mapping δεν είναι κάτι ιδιαίτερο παρά από μια απλή δήλωση τύπου 4000:3000 που μπορεί να δηλωθεί στην εντολή docker run κάποιου container ή μέσα στο αρχείο docker-compose.yml στο property ports. Οι εικόνες δείχνουν τα αρχεία dockerfile και docker-compose.yml που δημιουργήθηκαν για την εφαρμογή μας.



```
Dockerfile x
Dockerfile ▸ ...
1 FROM node:10.13-alpine
2 ENV NODE_ENV production
3 WORKDIR /usr/src/app
4 COPY ["package.json", "package-lock.json*", "npm-shrinkwrap.json*", "./"]
5 RUN npm install --production --silent && mv node_modules ../
6 COPY . .
7 EXPOSE 3000
8 CMD npm start
```

Απόσπασμα 15: DockerFile

Η γραμμή 1 λέει ποια έκδοση της image του node θα χρησιμοποιηθεί αφού η εφαρμογή μας τρέχει σε node περιβάλλον. Η γραμμή 2 θέτει το περιβάλλον του node σε production λειτουργία. Η γραμμή 3 μας μεταφέρει μέσα στον φάκελο /usr/src/app που δημιουργεί μέσα στον container και ο οποίος θα περιέχει την εφαρμογή μας. Σε αυτόν τον φάκελο μέσα θα πρέπει να τρέξουν και οι επόμενες εντολές και να εγκαταστήσουν τα απαραίτητα modules & dependencies. Η γραμμή 4 αντιγράφει από το project μας μέσα στον container το package.json από το οποίο θα γίνει η εγκατάσταση των modules & dependencies. Η γραμμή 5 τρέχει την εντολή run η οποία διαβάζει από το package.json και σε περιβάλλον production και μετακινεί τον φάκελο node_modules στον root φάκελο του container (αυτό υποδηλώνεται με την παράμετρο ../ στο τέλος της εντολής. Αυτό το μονοπάτι δηλώνει ότι θα πάει ένα φάκελο πίσω από τον τρέχον άρα θα πάει στον root φάκελο) Η γραμμή 6 αντιγράφει τα πάντα από τον τρέχον φάκελο του project μας στον τρέχον φάκελο του container παίρνει όπως βλέπουμε δύο τελείες ως παραμέτρους δηλαδή παίρνει δύο παραμέτρους όπου η πρώτη τελεία δηλώνει τον source folder και η δεύτερη τον destination folder εντός του container. Η γραμμή 7 λέει ότι ο container θα ακούει στο port 3000 και στο οποίο θα γίνει mapping με τον port που ακούει η εφαρμογή στη συνέχεια εντός του αρχείου docker-compose.yml. Τέλος η γραμμή 8 τρέχει την εντολή npm start όπου διαβάζει από το package.json το property 'start' του οποίου το value είναι το ng serve και εκκινεί τον container.



```

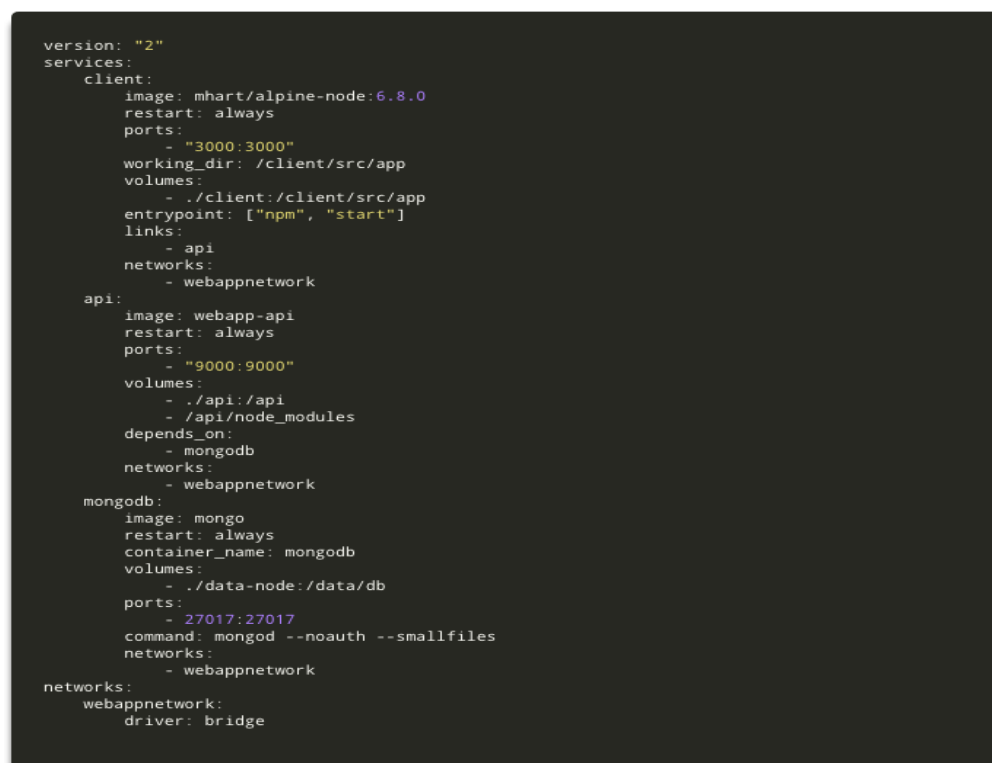
1  version: '2.1'
2
3  services:
4    calendar-app:
5      image: calendar-app
6      build: .
7      environment:
8        NODE_ENV: production
9      ports:
10     - 3000:3000

```

Απόσπασμα 16: Docker-Compose.yml

Τώρα αφού έχουμε τις ρυθμίσεις του container ήρθε η ώρα να δημιουργήσουμε την image η οποία θα δημιουργεί τους containers με τη δομή και λογική που αυτοί αποτυπώνονται μέσα σε αυτό το αρχείο του αποσπάσματος 16.

Το αρχείο όπως φαίνεται ορίζει έναν container εν ονόματι calendar-app ο οποίος θα περιέχει μόνο όσα αρχεία είναι απαραίτητα από όλη την εφαρμογή. Σε διαφορετική περίπτωση θα μπορούσε να όριζε έναν για το frontend έναν για το backend και έναν για τη βάση δεδομένων που θα χρησιμοποιεί. Ένα παράδειγμα τέτοιας δομής αυτού του αρχείου απεικονίζεται στην παρακάτω εικόνα, η οποία δεν είναι όμως σχετική με αυτή την εφαρμογή. Παρατίθεται απλά για ενημέρωση.



```

version: "2"
services:
  client:
    image: mhart/alpine-node:6.8.0
    restart: always
    ports:
      - "3000:3000"
    working_dir: /client/src/app
    volumes:
      - ./client:/client/src/app
    entrypoint: ["npm", "start"]
    links:
      - api
    networks:
      - webappnetwork
  api:
    image: webapp-api
    restart: always
    ports:
      - "9000:9000"
    volumes:
      - ./api:/api
      - /api/node_modules
    depends_on:
      - mongodb
    networks:
      - webappnetwork
  mongodb:
    image: mongo
    restart: always
    container_name: mongodb
    volumes:
      - ./data-node:/data/db
    ports:
      - 27017:27017
    command: mongod --noauth --smallfiles
    networks:
      - webappnetwork
networks:
  webappnetwork:
    driver: bridge

```

Εικ.24. Docker-Compose.yml με κώδικα σύνθεσης πολλών containers (João, 2018)

Εδώ βλέπουμε ότι έχουμε πολλά services το καθένα από αυτά αναπαριστά έναν container. Μόλις η εικόνα δημιουργηθεί από ένα τέτοιο αρχείο κάθε φορά που θα την τρέχουμε αυτή θα δημιουργεί τόσους containers όσους ορίζει αυτό το αρχείο από το οποίο η εικόνα δημιουργήθηκε. Στην εικόνα βλέπουμε επίσης στο κάτω μέρος το property 'networks' αυτό ορίζει μέσα σε ποιο δίκτυο θα βρίσκονται όλοι αυτοί οι containers για να επικοινωνούν. Το δίκτυο ονομάζεται webappnetwork στην προκειμένη περίπτωση της εικόνας. Σε αυτή την περίπτωση έχουν δημιουργηθεί σε κάθε φάκελο κάθε συστατικού ένα ξεχωριστό αρχείο dockerfile τα οποία και συνθέτει το παραπάνω αρχείο της εικόνας.

Αφού δημιουργήσουμε αυτά τα αρχεία ήρθε η ώρα να τα χρησιμοποιήσουμε με την παρακάτω εντολή για να δημιουργήσουμε την τελική image της πακεταρισμένης πλέον εφαρμογής μας. Εντός του editor πάμε στο command palette (Ctrl+Shift+P) και γράφουμε Docker: Build Image πατάμε 'enter' και μετά επιλέγουμε το dockerfile και δίνουμε όνομα στην εικόνα αυτής της μορφής

```
[registry_name or username or dockerhub_name]/[image_name]:[tag]latest
```

ή μπορούμε ανοίγοντας το docker cli να γράψουμε την εντολή

```
docker-compose build και μετά docker-compose up
```

για να ξεκινήσει να τρέχει η εφαρμογή μέσα από τον container. Θα πρέπει πρώτα να έχουμε μεταφερθεί στον φάκελο του project μας με την εντολή cd και το μονοπάτι που παραπέμπει στον φάκελο του project.

Μόλις ολοκληρωθεί θα μας ενημερώσει με το όνομα της δημιουργημένης εικόνας το οποίο θα έχει τη μορφή calendar-app:latest (αν δεν ορίσουμε version βάζει by default το tag latest). Με την εντολή docker images μπορούμε να δούμε όλες όσες έχουμε. Τώρα είμαστε έτοιμοι να την κάνουμε push στο repository με την εντολή

docker login (αρχικά για σύνδεση στο repository) και πατάω enter συμπληρώνω username & password όπως ζητάει το cli και στη συνέχεια πρέπει να βάλω μια ετικέτα στην εικόνα μου ώστε να τη συσχετίσω με το repository

```
docker images (και βρίσκω το tag της IMAGE_TAG δίπλα στο όνομά της)
```

```
docker tag IMAGE_TAG username_in_dockerhub/respository_in_dockerhub
```

Τέλος τη στέλνω στο repository

```
docker push username_in_dockerhub/respository_in_dockerhub
```

Εάν επισκεφτούμε το αποθετήριο μας στο dockerhub μέσω ενός φυλλομετρητή θα δούμε ότι πλέον έχει την πρώτη του καταχώρηση.

Επιπλέον εντολές που μπορεί να χρησιμεύσουν...

Με την παρακάτω εντολή μπορούμε να βρούμε το port του host machine στο οποίο γίνεται mapping το port του container μας. Ανοίγουμε το docker cli και γράφουμε την παρακάτω εντολή

```
docker port container_id port_of_the_container
```

το container_id μπορούμε να το μάθουμε από την εντολή docker ps και αφού το βρω συμπληρώνω στην παραπάνω εντολή τα δύο τρία πρώτα στοιχεία μετά πατάω tab για autocomplete και τέλος 'enter'. Το port_of_the_container το γνωρίζουμε από το docker-compose.yml όπου το έχουμε εμείς ορίσει.

Η παρακάτω εντολή μπορεί να διαγράψει τον container

```
docker kill container_id
```

Ο κάθε container έχει όπως έχουμε πει μια ip address η οποία δεν είναι δημόσια και βοηθάει στην επικοινωνία μεταξύ των containers που έχουν οριστεί να βρίσκονται στο ίδιο δίκτυο στο αρχείο docker-compose.yml. Επίσης είπαμε ότι μέσω του port mapping έχουμε τη δυνατότητα να φτάνουμε τον container μας μέσα από την ip του host machine και το port σε αυτό το host machine στο οποίο γίνεται mapping το port του container. Η εντολή η οποία μας εμφανίζει την ip του container είναι η παρακάτω.

```
docker inspect --format '{{.NetworkSettings.IPAddress}}' container_id
```

Για να τρέξουμε έναν container από το docker cli και όχι από τον editor με δεξί κλικ > run μπορούμε να χρησιμοποιήσουμε την παρακάτω εντολή

```
docker run --d --publish-all image_name:version
```

όπου --d σημαίνει ότι θα τρέξει στο φόντο σαν 'daemon' και --publish-all ότι θα μπορεί να φτάσουμε αυτόν τον container από άλλους υπολογιστές.

Τέλος έχουμε την παρακάτω εντολή

```
docker run -d --restart=always -p 8080:80 image_name:version
```

η οποία τρέχει τον container και κατά το τρέξιμο δηλώνουμε εμείς το mapping που θέλουμε να γίνει ως εξής -p 8080:80. Το 80 είναι το port στο οποίο ακούει ο container και το 8080 είναι το δημόσιο port του host machine και στο οποίο θέλουμε να γίνει το mapping. (Noureddin Sadawi, 2016)

7

ΣΥΜΠΕΡΑΣΜΑΤΑ

7.1 Σύνοψη της όλης υλοποίησης και αποτελέσματα επιτυχίας

Με λίγα λόγια φτάνοντας στο τέλος αυτής της πτυχιακής θα αναφερθούμε στο τι υλοποιήθηκε και αν κατάφερε να ικανοποιήσει τις απαιτήσεις με επιτυχία.

Οι σύγχρονοι ρυθμοί ζωής απαιτούν την ύπαρξη σωστής οργάνωσης για την επιτυχή διεκπεραίωση των καθημερινών δραστηριοτήτων ώστε να αποφευχθεί το άγχος και η κούραση. Στην αγορά για αυτόν ακριβώς το σκοπό υπάρχουν εφαρμογές ημερολογίων που υπόσχονται την οργάνωση του καθημερινού όγκου των υποχρεώσεων ενός ατόμου ώστε αυτό να μην ανησυχεί γι' αυτό. Οι εφαρμογές αυτές είναι εξαιρετικά και έξυπνα φτιαγμένες να επιτελούν αρκετές λειτουργίες και να παρέχουν πολλές δυνατότητες στον χρήστη. Βέβαια υπάρχουν δυνατότητες και λειτουργίες που θα μπορούσαν να προσθέσουν επιπλέον ισχύ στις εφαρμογές αυτές σχετικά με τον τρόπο επιλογής των δραστηριοτήτων και αυτές έχουν υλοποιηθεί στην παρούσα εφαρμογή αυτής της πτυχιακής εργασίας.

Υλοποιήθηκε λοιπόν μια εφαρμογή που έχει ως σκοπό να οργανώσει τη μέρα ενός ατόμου με την πιο αποδοτική πάντα σειρά προτεραιότητας των δραστηριοτήτων που έχουν καταχωρηθεί, αποφεύγοντας συγκρούσεις ωρών και παρέχοντας μια ταξινόμηση τέτοιου είδους που κάθε δραστηριότητα μπορεί να ολοκληρωθεί μέσα στη μέρα. Για την επίτευξη αυτού λοιπόν του σκοπού επιλέχθηκαν ειδικές τεχνολογίες και εργαλεία που θα βοηθούσαν στην πιο αποτελεσματική υλοποίησή της. Χρησιμοποιήθηκαν οι τεχνολογίες Angular για το frontend, Express για το backend τις οποίες εγκαταστήσαμε μέσα από τη τεχνολογία NodeJS καθώς και MySQL για τη βάση δεδομένων και τα εργαλεία VS Code ως ένας πολύ ικανός editor ανάπτυξης κώδικα, XAMPP ένας τοπικός server που διαθέτει και μια βάση δεδομένων τοπική, postman για τη δημιουργία του rest api και ένας φυλλομετρητής συγκεκριμένα ο chrome για το render του view της εφαρμογής στην οθόνη και το debugging του κώδικα. Οι τεχνολογίες αυτές τηρώντας τις αρχιτεκτονικές MVC και 3-Tier οργάνωσαν και δόμησαν την εφαρμογή σε διακριτά στάδια υλοποίησης που όλα επικοινωνούν μεταξύ τους.

Η εφαρμογή μετά κι από 2 σενάρια εκτέλεσης τελειοποιήθηκε και το τελικό αποτέλεσμα είναι να έχει δημιουργηθεί μια σύγχρονη εφαρμογή ημερολογίου με εύκολο και σύγχρονο UI και στυλ αλληλεπίδρασης με το χρήστη για καλύτερη εμπειρία χρήστη. Η λειτουργικότητα που παρέχει έχει διορθωθεί από σφάλματα καθιστώντας την εφαρμογή πλέον έτοιμη για χρήση.

ΥΛΟΠΟΙΗΣΗ ΔΙΑΔΙΚΤΥΑΚΗΣ ΕΦΑΡΜΟΓΗΣ ΗΜΕΡΟΛΟΓΙΟΥ ΜΕ ΔΥΝΑΤΟΤΗΤΑ ΑΥΤΟΜΑΤΟΥ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

7.2 Μελλοντική αναβάθμιση της εφαρμογής (επιπλέον λειτουργικότητα)

Όπως κάθε νέο λογισμικό έτοιμο προς διάθεση στην αγορά ποτέ δε σταματά να εξελίσσεται και να αναβαθμίζεται από τους δημιουργούς του έτσι και στην περίπτωση αυτής της εφαρμογής σκοπός είναι η εξέλιξή της με την προσθήκη μελλοντικών προσδοκώμενων χαρακτηριστικών περισσότερων επιλογών λειτουργικότητας.

Τα χαρακτηριστικά αυτά τα οποία προβλέπεται να προστεθούν στο μέλλον είναι δύο κατηγοριών. Αρχικά στην 1^η κατηγορία έχουμε την προσθήκη κάποιων μη λειτουργικών επιπλέον απαιτήσεων όπως η ασφάλεια των δεδομένων της εφαρμογής που καταχωρούνται αφού η εφαρμογή γίνει διαθέσιμη στο διαδίκτυο και για πολλούς χρήστες που ο καθένας θα μπορεί να έχει πρόσβαση στα δεδομένα του χρησιμοποιώντας τον προσωπικό του λογαριασμό. Κάτι ανάλογο με τις ήδη υπάρχουσες εφαρμογές. Η ασφάλεια αυτή θα λαμβάνεται με χρήση https (που είναι σε γενικές γραμμές το http που εφαρμόζεται πάνω σε SSL/TLS) αντί του απλού http πρωτοκόλλου.

Στη 2^η κατηγορία θα μπορούσε να υλοποιηθεί λογική τέτοια η οποία θα ειδοποιούσε με μήνυμα στο κινητό για το ότι πλησιάζει η ώρα να ξεκινήσει κάποια priority δραστηριότητα. Αυτό βέβαια για να επιτευχθεί θα πρέπει και ο χρήστης να έχει συνδεθεί στο λογαριασμό του από κάποια συσκευή. Επίσης αναγκαίο θα είναι να γίνονται συνεχώς έλεγχοι και να εκτελούνται σενάρια εκτέλεσης για πιθανό εντοπισμό παραπάνω σφαλμάτων και για άμεση διόρθωσή τους, με άλλα λόγια ενημέρωση της εφαρμογής αν αυτό χρειαστεί. Επιπλέον χρήσιμο θα ήταν να δίνεται η δυνατότητα στον χρήστη να καταχωρεί και κάποια σχόλια τα οποία θα σχετίζονται με την προς καταχώρηση δραστηριότητα εάν το επιθυμεί. Τέλος, μια επίσης εντυπωσιακή ιδέα είναι η προσθήκη επιλογής drag&drop για σε περίπτωση που κάποια δραστηριότητα με τον τρόπο που έχει ταξινομηθεί στην οποιαδήποτε προτεινόμενη θέση δεν αρέσει στον χρήστη. Με αυτήν την επιλογή θα μπορεί να αλλάξει θέση της επιλεγμένης δραστηριότητας μέσα στον πίνακα και τα νέα δεδομένα θα καταχωρηθούν στη βάση ενημερώνοντάς την.

7.3 Συνεισφορά στο ευρύ κοινό

Σε αυτήν την παράγραφο θα αναφερθούμε σε πόσα επίπεδα μια τέτοια εφαρμογή θα μπορούσε να φανεί χρήσιμη και πως. Αρχικά σε ατομικό επίπεδο και στη συνέχεια θα αναφερθούμε στις θετικές συνέπειες που η χρήση της θα μπορούσε να επιφέρει σε ένα ευρύτερο φάσμα όπως η κοινωνία.

Σε ατομικό λοιπόν αρχικά επίπεδο θα μπορούσε να χρησιμοποιηθεί τόσο από καθημερινούς ανθρώπους, φοιτητές, ιδιωτικούς ή και δημόσιους υπαλλήλους όσο και επαγγελματίες εταιριών, διευθυντές και γενικότερα κάθε είδους εργαζόμενο και μη.

Συγκεκριμένα για τους φοιτητές που εργάζονται παράλληλα με τις σπουδές τους θα ήταν τέλεια ιδέα οργάνωσης του προγράμματός τους και των υποχρεώσεών τους απέναντι στις εργασίες και τα μαθήματα της σχολής που πρέπει να γίνονται παράλληλα με τόσα άλλα πράγματα χωρίς να παραλειφθεί καμία υποχρέωση ή δραστηριότητα λόγω λάθους διαχείρισης της ημέρας τους από μεριάς χρόνου. Επίσης και μια απλή νοικοκυρά θα μπορούσε να επωφεληθεί οργανώνοντας τις υποχρεώσεις του σπιτιού χρονικά με αποτελεσματικό τρόπο που θα της επιφέρει να έχει ολοκληρώσει όλες τις δραστηριότητές της από της πιο απλές έως τις πιο απαιτητικές διότι όλοι πολλές φορές λίγο πολύ τυγχάνει να αναβάλλουμε δραστηριότητες από το πρόγραμμά μας διότι πιστεύουμε ότι δε μας φτάνει ο χρόνος αλλά αν σκεφτούμε μια σωστή διαχείριση αυτού θα καταφέρουμε να προσθέσουμε περισσότερα μέσα στη μέρα μας γεγονότα και θα έχουμε καταφέρει να τα ολοκληρώσουμε στην ώρα τους. Οι ιδιωτικοί και δημόσιοι υπάλληλοι πέρα από τη διαχείριση του καθημερινού τους εκτός δουλειάς προγράμματος θα μπορούσαν να τη χρησιμοποιήσουν και ως μέσο διαχείρισης των εργασιακών υποχρεώσεών τους κατά τη διάρκεια εργασίας τους. Για παράδειγμα αν ένας εργαζόμενος ιδιωτικός-δημόσιος υπάλληλος ή επαγγελματίας ελεύθερος ή μη, διευθυντής ή κάθε άλλου είδους εργαζόμενος έχει να αναλάβει κατά τη διάρκεια της εργασίας του υποχρεώσεις όπως διαχείριση εγγράφων, ενημέρωση προσωπικού, τηλεφωνική επικοινωνία με άλλους φορείς και άλλα παρόμοια γεγονότα θα μπορούσε να τα οργανώσει με χρήση μιας τέτοιας εφαρμογής και να αποφασίσει πότε και ποιος θα ήταν ο καλύτερος τρόπος διεκπεραίωσής τους.

Σε ένα ευρύτερο τώρα φάσμα οι συνέπειες που απορρέει η χρήση της εφαρμογής από κάθε είδους άνθρωπο εργαζόμενο ή και μη δε θα μπορούσαν να είναι τίποτα άλλο παρά μόνο θετικές και κάτι τέτοιο μπορεί να επιφέρει λόγω σωστής διαχείρισης του χρόνου από κάθε άνθρωπο καλύτερη λειτουργία της κοινωνίας καθώς θα υπάρχει σωστός προγραμματισμός στον καθένα ξεχωριστά, αρχικά σε ατομικό επίπεδο και στη συνέχεια σε ευρύτερο χωρίς άγχη και καθυστερήσεις γενικότερα.

ΒΙΒΛΙΟΓΡΑΦΙΑ

- Tserpes, K. (2019, April 29). stream-MSA: A microservices' methodology for the creation of short, fast-paced, stream processing pipelines. Retrieved from <https://www.sciencedirect.com/science/article/pii/S2405959519301092>
- Ali Syed, B. (2014). *Beginning Node.js UNLEASH THE POWER OF NODE.JS AND CREATE HIGHLY SCALABLE WEBSITES*. Apress.
- Angular, D. (n.d.-a). *Dependency Injection in Angular*. Retrieved from <https://angular.io/guide/dependency-injection>
- Angular, D. (n.d.-b). *Observables*. Retrieved from <https://angular.io/guide/observables>
- Angular, D. (n.d.-c). *Observables in Angular*. Retrieved from <https://angular.io/guide/observables-in-angular>
- Brown, E. (2014). *Web Development with Node & Express LEVERAGING THE JAVASCRIPT STACK* (First). O'REILLY.
- Component Diagram Tutorial [Computer Science]. (n.d.). Retrieved from <https://www.lucidchart.com/pages/uml-component-diagram>
- Freeman, A. (2016). *Pro ASP.NET Core MVC Develop cloud-ready web applications using Microsoft's latest framework, ASP.NET Core MVC* (Sixth). Retrieved from www.alitebooks.com
- Google Calendar. (2019). In *Wikipedia* (Online). Retrieved from https://en.wikipedia.org/wiki/Google_Calendar
- Janetakakis, N. (2018, April 5). Docker Tip #53: Difference between a Registry, Repository and Image [Computer Science]. Retrieved from <https://nickjanetakakis.com/blog/docker-tip-53-difference-between-a-registry-repository-and-image>
- João, H. (2018, October 15). How to create a full stack React/Express/MongoDB app using Docker [Computer Science]. Retrieved from <https://www.freecodecamp.org/news/create-a-fullstack-react-express-mongodb-app-using-docker-c3e3e21c4074/>
- Kambalyal, C. (n.d.). *3-Tier Architecture* (Online). Retrieved from <http://channukambalyal.tripod.com/NTierArchitecture.pdf>
- Lim, G. (2017). *Beginning Angular 2 with TypeScript* (First).
- Microsoft. (2019). Visual Studio Code [Computer Science]. Retrieved from <https://code.visualstudio.com/>
- Microsoft Azure. (2019). In *Wikipedia* (Online). Retrieved from https://en.wikipedia.org/wiki/Microsoft_Azure
- Microsoft, O. (n.d.-a). *Create or schedule an appointment*. Retrieved from <https://support.office.com/en-us/article/create-or-schedule-an-appointment-be84396a-0903-4e25-b31c-1c99ce0dacf2>
- Microsoft, O. (n.d.-b). *Introduction to the outlook calendar*. Retrieved from <https://support.office.com/en-us/article/introduction-to-the-outlook-calendar-d94c5203-77c7-48ec-90a5-2e2bc10bd6f8>
- Möller, M. (2018, June 13). HOW TO DEPLOY A DOCKERIZED APP TO MICROSOFT AZURE WEB APP FOR CONTAINERS [Computer Science]. Retrieved from <https://blog.akquinet.de/2018/06/13/how-to-deploy-a-dockerized-app-to-microsoft-azure-web-app-for-containers/>
- NodeJs. (2019). In *Wikipedia* (Online). Retrieved from https://el.wikipedia.org/wiki/Nodejs#cite_note-3
- Patel, P. (2018, April 18). What exactly is Node.js? [Computer Science Web Development]. Retrieved from <https://www.freecodecamp.org/news/what-exactly-is-node-js-ae36e97449f5/>
- Platform as a service. (2019). In *Wikipedia* (Online). Retrieved from https://en.wikipedia.org/wiki/Platform_as_a_service
- Rangle's Angular 2 Training Book*. (n.d.). GitBook.
- Sans, G. (2015, September 19). Angular — Introduction to Reactive Extensions (RxJS) [Computer Science]. Retrieved from <https://medium.com/google-developer-experts/angular-introduction-to-reactive-extensions-rxjs-a86a7430a61f>

Tabaghoghi, S. M. M., & Williams, H. E. (2007). *Learning MySQL Get a Handle on Your Data* (Online). O'REILLY.

Testing with Postman. (n.d.). Retrieved from <https://wiki.onap.org/display/DW/Testing+with+Postman>

Toroman, M. (2017, March 7). MySQL In-app [Computer Science]. Retrieved from <http://toroman.cloud/2017/07/03/mysql-in-app/>

Visual Paradigm. (n.d.). What is Use Case Diagram? [Computer Science]. Retrieved from <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/>

Visual Studio Code. (2019). In *Wikipedia* (Online). Retrieved from https://en.wikipedia.org/wiki/Visual_Studio_Code

XAMPP. (2017). In *Wikipedia* (Online). Retrieved from <https://el.wikipedia.org/wiki/XAMPP>

Visual Paradigm. (n.d.). What is Component Diagram? [Computer Science]. Retrieved from <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/?WWWSESSIONID=3D181BA45CE746E7A2DC246AB84B1A82.www1>

Αναφορές αναφορών

Tilkov, S., & Vinoski, S. (n.d.). *Node.js: Using JavaScript to Build High-Performance Network Programs*.

Αναφορές που βοήθησαν στη συγγραφή του κώδικα

Από stackoverflow.com

Για τον κώδικα

<https://stackoverflow.com/questions/34657821/ngif-and-ngfor-on-same-element-causing-error>

<https://stackoverflow.com/questions/37026219/angular2-ngfor-and-ngif-on-same-element-producing-error>

<https://stackoverflow.com/questions/40378562/how-to-highlight-a-selected-row-in-ngfor>

<https://stackoverflow.com/questions/50604093/how-to-highlight-a-column-in-the-material-data-table-based-on-a-certain-condition/50604280>

<https://stackoverflow.com/questions/29803045/how-to-clear-an-angular-array>

<https://stackoverflow.com/questions/32062051/angular2-forms-submit-button-disabled>

<https://stackoverflow.com/questions/47529327/angular-4-get-input-value>
<https://stackoverflow.com/questions/44205950/angular-passing-a-variable-from-a-component-to-a-service>

<https://stackoverflow.com/questions/45891998/posting-form-data-to-mysql-using-nodejs-w-express>

<https://stackoverflow.com/questions/35837338/get-key-value-from-the-list-of-json-in-nodejs>

<https://stackoverflow.com/questions/52132243/general-nodejs-express-error-when-declaring-routes-typeerror-cannot-read-pro/52132244>

<https://stackoverflow.com/questions/39484123/how-to-update-multiple-columns-in-mysql-using-nodejs>

<https://stackoverflow.com/questions/35588699/response-to-preflight-request-doesnt-pass-access-control-check>

<https://stackoverflow.com/questions/18642828/origin-origin-is-not-allowed-by-access-control-allow-origin>

<https://stackoverflow.com/questions/41500102/cleanest-way-to-reset-forms>

<https://stackoverflow.com/questions/46258449/cors-error-requests-are-only-supported-for-protocol-schemes-http-etc>

<https://stackoverflow.com/questions/9261713/regex-strip-out-text-between-first-and-second-forward-slashes>

<https://stackoverflow.com/questions/8519734/javascript-get-characters-between-slashes>

<https://stackoverflow.com/questions/34778608/warning-mysqli-connect-hy000-2002-an-attempt-was-made-to-access-a-socket>

<https://stackoverflow.com/questions/43549216/how-to-make-2-ways-data-binding-for-td-in-angular-2>

<https://stackoverflow.com/questions/47651718/can-we-assign-ng-model-to-span>

Για το deploy

<https://stackoverflow.com/questions/45078663/docker-4-containers-for-frontend-backend-database-push-server>

<https://stackoverflow.com/questions/17157721/how-to-get-a-docker-containers-ip-address-from-the-host>

<https://stackoverflow.com/questions/28349392/how-to-push-a-docker-image-to-a-private-repository>

<https://stackoverflow.com/questions/18497688/run-a-docker-image-as-a-container>

<https://stackoverflow.com/questions/53507378/create-docker-file-for-client-and-server>

Από documentations των τεχνολογιών

Για τον κώδικα

<https://angular.io/docs>

<https://material.angular.io/components/categories>

<https://expressjs.com/en/guide/routing.html>

Για το deploy

<https://docs.docker.com/docker-hub/>

<https://docs.microsoft.com/en-us/azure/container-instances/container-instances-quickstart>

<https://code.visualstudio.com/tutorials/docker-extension/deploy-container>

<https://code.visualstudio.com/docs/azure/docker>

Από διάφορα blogs και websites

Για τον κώδικα

<https://malcoded.com/posts/angular-ngfor>

https://www.w3schools.com/nodejs/nodejs_mysql.asp

<https://flaviocopes.com/node-mysql/>

<https://www.opentechguides.com/how-to/article/nodejs/124/express-mysql-json.html>

<https://www.concretepage.com/angular-2/angular-httpclient-get-example#get>

<https://www.telerik.com/blogs/updating-to-angular-httpclient-simpler-http-calls>

<https://www.w3schools.com/icons/tryit.asp?icon=fas fa-circle&unicon=f111>

https://www.tutorialspoint.com/typescript/typescript_arrays.htm

<https://ourcodeworld.com/articles/read/278/how-to-split-an-array-into-chunks-of-the-same-size-easily-in-javascript>

<https://coryryan.com/blog/angular-ng-for-syntax>

Χρήσιμα εκπαιδευτικά κανάλια στο **YouTube**

Που βοήθησαν στον κώδικα

- ***Programming with Mosh***

Angular 4 in 40 Minutes by Mosh →

https://www.youtube.com/watch?v=gK21_9QuthM

Directives in Angular Applications by Mosh →

<https://www.youtube.com/watch?v=LtT01ZCHRjk>

Building Forms in Angular Apps by Mosh →

https://www.youtube.com/watch?v=hAaoPOx_olw

Angular Material Tutorial by Mosh →

<https://www.youtube.com/watch?v=wPT3K3w6JtU>

Angular Tutorial for Beginners: Learn Angular from Scratch by Mosh →

<https://www.youtube.com/watch?v=k5E2AVpwsko>

Two-way Binding and ngModel in Angular 4 by Mosh →

<https://www.youtube.com/watch?v=Wjcl09xgo3o>

Working with Components in Angular by Mosh →

https://www.youtube.com/watch?v=SRn_E6juCVE

Angular 2 Tutorial for Beginners: Learn Angular 2 from Scratch by Mosh →

https://www.youtube.com/watch?v=-CD_5YhJTA

- **FreeCodeCamp.org**

Learn Angular - Full Tutorial Course by FreeCodeCamp.org →

https://www.youtube.com/watch?v=2OHbjep_WjQ

- **CodAffection**

Angular Material Confirm Dialog by CodAffection →

<https://www.youtube.com/watch?v=L7mrAYsh0-0>

Angular Material Popup Dialog & Model by CodAffection →

<https://www.youtube.com/watch?v=ZL0d3M3uoRQ>

Angular Material Data Table - Paging, Sorting and Filter Operation by CodAffection

→ <https://www.youtube.com/watch?v=7wilnsiotqM>

Node.js + MySQL CRUD - GET,POST,PUT and DELETE by CodAffection →

<https://www.youtube.com/watch?v=4fWWn2Pe2Mk>

- **Fireship**

Angular HTTP Client Quick Start Tutorial by Fireship →

<https://www.youtube.com/watch?v=05v0mrNLh0>

Sharing Data between Components in Angular by Fireship →

<https://www.youtube.com/watch?v=I317BbehZKM>

- **Codedamn**

NodeJS + MySQL Database Connection Tutorial by codedamn →
https://www.youtube.com/watch?v=hGZX_SA7IYg

- **Techie Ocean**

ANGULAR 7 TypeError Cannot read property of undefined by Techie Ocean →
<https://www.youtube.com/watch?v=OptW5UXTz6U>

Που βοήθησαν στην κατανόηση του deploy του κώδικα

- **John Papa**

Angular App from Scratch - 3/n - Deploying to Azure by John Papa →
<https://www.youtube.com/watch?v=waOlkiFITIQ>

- **Noureddin Sadawi**

Docker Training 20/29: Docker Container IP Address and Port Number by Noureddin Sadawi → <https://www.youtube.com/watch?v=MkRBZSCyN-8>

Docker Training 19/29: Container Networking Basics by Noureddin Sadawi →
<https://www.youtube.com/watch?v=m138loKiYo0>

- **Jeremy Likness**

Docker Containers with Azure App Service by Jeremy Likness →
<https://www.youtube.com/watch?v=nWfpwgHRfqk>

- **KodeKloud**

Understanding Docker by KodeKloud →

<https://www.youtube.com/watch?v=wxxigbHwDGM>

- ***Jonny Langefeld***

From Zero to Docker - Tutorial for Beginners by Jonny Langefeld →

<https://www.youtube.com/watch?v=JprTjTViaEA>