ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

Σχεδιασμός και υλοποίηση Web εφαρμογής για τη διαχείριση φύλλων χρονοχρέωσης έργων

ΝΙΚΟΛΑΟΣ ΧΑΣΚΑΡΗΣ

Αθήνα, 2024

ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
SCHOOL OF DIGITAL TECHNOLOGY
DEPARTMENT OF INFORMATICS AND TELEMATICS

Design and implementation of a web-based application for the management of project timesheets

NIKOLAOS CHASKARIS

Athens, 2024

ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

Τριμελής Εξεταστική Επιτροπή

Τσερπές Κωνσταντίνος (Επιβλέπων/ούσα)
Αναπληρωτής Καθηγητής, Πληροφορική & Τηλεματική, Χαροκόπειο

Δημητρακόπουλος Γεώργιος
Αναπληρωτής Καθηγητής, Πληροφορική & Τηλεματική, Χαροκόπειο

Καμαλάκης Θωμάς
Καθηγητής, Πληροφορική & Τηλεματική, Χαροκόπειο

Ο ΝΙΚΟΛΑΟΣ ΧΑΣΚΑΡΗΣ

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 προσβάλει 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.

3) Όπου υφίστανται δικαιώματα άλλων δημιουργών έχουν διασφαλιστεί όλες οι αναγκαίες άδειες χρήσης ενώ το αντίστοιχο υλικό είναι ευδιάκριτο στην υποβληθείσα εργασία.

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να εκφράσω την ειλικρινή μου ευγνωμοσύνη στον σύμβουλο της διατριβής μου, καθηγητή Τσερπέ Κωνσταντίνο, για την πολύτιμη καθοδήγηση, την ενθάρρυνση και την υπομονή του καθ' όλη τη διάρκεια αυτού του έργου. Η τεχνογνωσία και οι γνώσεις του καθηγητή Τσερπέ συνέβαλαν καθοριστικά στη διαμόρφωση της κατεύθυνσης αυτής της έρευνας.

Θα ήθελα επίσης να ευχαριστήσω τα μέλη της επιτροπής της διατριβής μου, Δημητρακόπουλο Γεώργιο και Καμαλάκη Θωμά, για τα διορατικά σχόλια και τις υποδείξεις τους. Τα σχόλιά τους με βοήθησαν να βελτιώσω τη σαφήνεια και την εστίαση της εργασίας μου.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια και τους φίλους μου για την αμέριστη υποστήριξη και ενθάρρυνσή τους καθ' όλη τη διάρκεια των σπουδών μου. Η πίστη τους σε μένα με κράτησε παρακινημένο σε δύσκολες στιγμές.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Περίληψη στα Ελληνικά

Η τεχνολογία αναπτύσσεται συνεχώς και μαζί της και ο άνθρωπος είτε το θέλει είτε όχι. Αυτή η ανάπτυξη επεκτείνεται από σημαντικές προοδεύσεις στην Τεχνητή νοημοσύνη έως και διάφορα άλλα εργαλεία. Τα εργαλεία αυτά διευκολύνουν τη ζωή του, τόσο την προσωπική, όπως για παράδειγμα λογισμικό: για την παρακολούθηση δαπανών και σχεδιασμών εξόδων, διαμόρφωσης φωτογραφιών, δημιουργίας μουσικής κλπ., όσο και την επαγγελματική, πλατφόρμες για επικοινωνία, διαχείρισης πελατών, διαχείρισης έργων κλιπ. Για καθένα από τα παραπάνω έχουν δημιουργηθεί και ίσως διατεθεί στο κοινό εφαρμογές χωρίς όμως να καλύπτουν τις ανάγκες των υπηρεσιών. Έτσι το προσωπικό καταλήγει να χρησιμοποιεί Excel ή άλλες πλατφόρμες οι οποίες δεν είναι κατασκευασμένες για αυτό. Για τον λόγο αυτό μια Web εφαρμογή για τη διαχείριση φύλλων χρονοχρέωσης έργων αναπτύχθηκε. Κατά τον σχεδιασμό και ανάπτυξη της, το σχεδιάγραμμα επικεντρώθηκε κυρίως γύρω από τον χρήστη, βεβαιώνοντας ότι η εφαρμογή είναι εύκολη στη χρήση χωρίς να αλλοιώνονται οι βασικές λειτουργίες. Η εκτέλεση ενεργειών όπως η εισαγωγή δεδομένων σε πραγματικό χρόνο και η λεπτομερής εμφάνιση ειδοποιήσεων για σφάλματα ή προειδοποιήσεις αναδεικνύονται κύριες ανάγκες. Αυτά τα χαρακτηριστικά συνολικά ελαχιστοποιούν τα προβλήματα, την ανάγκη για επίβλεψη της εφαρμογής από διαχειριστή και βεβαιώνουν την έγκυρη εισαγωγή δεδομένων. Θέματα όπως η ασφάλεια και η εγκυρότητα των δεδομένων αποτέλεσε, επίσης σημαντικό ρόλο στον σχεδιασμό και στα στάδια του σχεδιαγράμματος. Η εφαρμογή διασφαλίζει ότι ο πληροφορίες για τον χρήστη και το κάθε έργο είναι προστατευμένα από μη εξουσιοδοτημένα άτομα ή προσπάθειες από αυτά για την παραβίαση δεδομένων. Οι παραπάνω προδιαγραφές είναι αναγκαίες για την εμπιστοσύνη του χρήστη και η εφαρμογή συμμορφώνεται στους κανονισμούς για την προστασία δεδομένων. Συμπερασματικά, ο σχεδιασμός και η υλοποίηση της Web εφαρμογής για τη διαχείριση φύλλων χρονοχρέωσης έργων οδηγούν προς την αποτελεσματική και αποδοτική διαχείριση έργων. Η επιτυχία της εφαρμογής εξαρτάται από τις αλληλεπιδράσεις των χρηστών, την ισχυρή ενσωμάτωση αναγκαίων λειτουργιών, και αδιάκοπη εστίαση στην ασφάλεια και στη χρηστικότητα. Η πλατφόρμα όχι μόνο καλύπτει τις ανάγκες των χρηστών αλλά και θέτει θεμέλια για την περαιτέρω ανάπτυξη της.

Λέξεις κλειδιά: Διαχείριση Φύλλου Χρόνου, Διαχείριση Έργου, Εφαρμογή μέσω Διαδικτύου

Abstract ή Περίληψη στα Αγγλικά

Technology is constantly evolving and with it, so is man, whether he likes it or not. This growth ranges from major advances such as in Artificial Intelligence to various tools. Which facilitates his life, both personal, such as for example software: for tracking expenses and expense planning, photo editing, music creation, etc., and professional, platforms for communication, customer management, project management, etc. For each of the above, applications have been created and may be made available to the public, but they do not cover the needs of services. So, staff end up using excel or other platforms which are not built for it. For this reason, a web application for managing project timesheets was developed. In its design and development, the layout was mainly centered around the user, making sure that the application is easy to use without altering any of the functions. Performing features such as real-time data entry and detailed display of error or warning notifications emerged as main needs. These features altogether minimize problems, the need for administrative supervision and ensure valid project import. Issues such as security and data validity also played an important role in the design and its stages. The application ensures that user and project information is protected from unauthorized individuals or attempts to compromise data in the database. The above specifications are necessary for user confidence and with adaptability based on data protection regulations. In conclusion, the design and implementation of the Web application for project timesheet management leads to effective and efficient project management. The success of the application is based on user-based design, robust integration, and relentless focus on security and usability. The platform not only meets user needs but also lays a foundation for further development.

Keywords: Timesheet Management, Project Management, Web-based Application

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

GNANTT	Graphical Representation of Activity Against Time
SDK	Software Development Kit
API	Application Programming Interface
BaaS	Backend as a Service
JVM	Java Virtual Machine
IDE	Integrated Development Environment
HTTP	Hypertext Transfer Protocol
DOM	Document Object Mode
UI	User Interface
CSS	Cascading Style Sheets
HTML	Hypertext Markup Language
CSV	Comma-separated values file

ΕΙΣΑΓΩΓΗ

Η διαχείριση φύλλων χρονοχρέωσης αποτελεί σημαντικό παράγοντα στην επιτυχή διεκπεραίωση ενός έργου και αποδοτική χρήση του ανθρώπινου δυναμικού. Καθώς τα έργα μεγαλώνουν και γίνονται ακόμη πιο περίπλοκα, οι οργανισμοί χρειάζονται εργαλεία για ακριβή και εύκολη καταγραφή των δεδομένων. Οι παραδοσιακές μέθοδοι είναι επιρρεπής σε λάθη, κατανάλωση υπερβολικού χρόνου και πιθανόν χάσιμο κεφαλαίων.

Το κίνητρο πίσω από αυτή την εφαρμογή πηγάζει από τις παραπάνω δυσκολίες. Χρησιμοποιώντας τη σύγχρονη τεχνολογία, η εφαρμογή στοχεύει να προσφέρει ένα ισχυρό, φιλικό προς τους χρήστες περιβάλλον. Ο τελικός σκοπός είναι να μειωθούν, αν όχι να εξαλειφθούν, τα ανθρώπινα σφάλματα, να μειωθεί η ανάγκη διαχείρισης από κάποιο φυσικό πρόσωπο και να προσφέρει στους υπεύθυνους των έργων τη δυνατότητα να λαμβάνουν τις αποφάσεις τους με βάση στοιχεία σε πραγματικό χρόνο.

Ο σκοπός αυτής της πτυχιακής είναι να σχεδιαστεί και να αναπτυχθεί μια εφαρμογή για τη διαχείριση φύλλων χρονοχρέωσης έργων. Η εφαρμογή θα επιτρέπει στους υπεύθυνους να εισάγουν τα μηνιαία φύλλα χρονοχρέωσης, να παρακολουθούν τα έργα και να εξάγουν αναφορές σχετικά με αυτά. Θα προσφέρει σε πραγματικό χρόνο την πρόσβαση και δυνατότητα αλλαγής οποιασδήποτε πληροφορίας, βοηθώντας τον υπεύθυνο στην παρατήρηση και διαχείριση έργων με ευκολία.

Για να είναι αποδεκτή η εφαρμογή πρέπει να τηρεί κάποιες απαιτήσεις οι οποίες χωρίζονται σε λειτουργικές και μη λειτουργικές. Παρακάτω είναι οι περιγραφές και οι μετρικές που χρησιμοποιήθηκαν για την κάθε απαίτηση.

1. Λειτουργικές Απαιτήσεις

    - Οι χρήστες έχουν τη δυνατότητα να συνδεθούν με τον λογαριασμό Google τους εφόσον έχει δημιουργηθεί λογαριασμός για αυτούς από κάποιον διαχειριστή. Τα ποσοστά επιτυχίας σύνδεσης είναι 100 τις εκατό όσο το σύστημα της Google δεν έχει διακοπές.

- Υπάρχει η δυνατότητα εξαγωγής φύλλων χρονοχρέωσης για ένα συγκεκριμένο μήνα από τους χρήστες σε μορφή CSV. Η αναμονή για τη λήψη του συγκεκριμένο αρχείου μπορεί να είναι έως και 5 δευτερόλεπτα ανάλογα τον όγκο των δεδομένων.

- Για την καλύτερη κατανόηση και διαχείριση των έργων, υπάρχει ένα GNANTT διάγραμμα το οποίο παρουσιάζει το όνομα του έργου, την ημερομηνία έναρξης έως και την ημερομηνία λήξης. Υπάρχει επίσης ένας οπτικό πίνακα που περιλαμβάνει όλες τις ημερομηνίες, χρωματικές αναπαραστάσεις των ημερομηνιών έναρξης και λήξης. Ο χρήστης μπορεί να αιωρηθεί πάνω από το κάθε έργο για να παρατηρήσει της ημέρες που απομένουν πριν την λήξη του.

- Ο χρήστης έχει τη δυνατότητα να επεξεργαστεί τα δεδομένα, έργα, συμβάσεις, φύλλα χρονοχρέωσης, χωρίς να υπάρχει κίνδυνος να χαθούν δεδομένα ή να μην αποθηκευτούν οι αλλαγές.

- Ο χρήστης αντί για τις ημερομηνίες λήξης εισάγει έναν αριθμό διάρκειας. Επίσης, ο χρήστης θα εισάγει τον μήνα που αρχίζει μία δέσμη εργασιών με μορφή MXX, η οποία χρησιμοποιείται ευρέως από τις ερευνητικές ομάδες. Αυτές οι πληροφορίες μεταφράζονται σωστά όταν τις λαμβάνει η εφαρμογή και τις παρουσιάζει σε κανονική ημερολογιακή ημερομηνία.

- Οι χρήστες έχουν τη δυνατότητα αναζήτηση συγκεκριμένου δεδομένου με ένα μοναδικό αναγνωριστικό πεδίο ανάλογα τον τύπο. Αυτή η αναζήτηση έχει μια καθυστέρηση έως και 3 δευτερόλεπτα σε μεγάλες ομάδες από πληροφορίες.

- Στα φύλλα χρονοχρέωσης όταν ο χρήστης επισκέπτεται τη σελίδα για εισαγωγή δεδομένων εμφανίζεται μια μορφή ημερολογίου για την πιο εύκολη κατανόηση. Στο ημερολόγιο αυτό τα σαββατοκύριακα και οι αργίες είναι απενεργοποιημένες εξαρχής και μόνο με παρέμβαση του χρήστη μπορούν να εισαχθούν δεδομένα σε αυτές. Αυτό το ημερολόγιο εμφανίζεται στον χρήστη στιγμιαία. Οι αργίες

ανανεώνονται καθημερινά κάθε μεσάνυχτα ώστε να μειωθεί η πιθανότητα αντιπαράθεσης με τους χρήστες.

- Υπάρχουν ειδοποιήσεις που ενημερώνουν τον χρήστη για οποιαδήποτε λάθος ενέργεια ή πληροφορία. Αυτή η διαδικασία έχει ποσοστό επιτυχίας 95 τις εκατό των φορών.

- Ο χρήστης έχει τη δυνατότητα να συνδεθεί με άλλους χρήστες για τη δημιουργία μίας ομάδας, παρέχοντας έτσι την δυνατότητα, εισαγωγής, επεξεργασίας και διαγραφής κοινού περιεχομένου.

2. Μη Λειτουργικές Απαιτήσεις

- Τα δεδομένα των χρηστών αποθηκεύονται σε μια βάση η οποία ακολουθεί όλους τους τυπικούς κανόνες ασφαλείας και περνάει 100 τις εκατό των τεστ ασφαλείας.

- Η εφαρμογή πρέπει να είναι πάντα διαθέσιμη στους χρήστες με ποσοστό χρόνου εκτός λειτουργίας 5 τις εκατό.

- Οι αλληλεπιδράσεις των χρηστών με την ιστοσελίδα πρέπει να είναι γρήγορες, κρατώντας υψηλή επίδοση ακόμη και σε ώρες αιχμής. Η μέση τιμή απόκρισης δε θα πρέπει να υπερβαίνει τα 3 δευτερόλεπτα.

- Η ιστοσελίδα θα πρέπει να λειτουργεί αξιόπιστα χωρίς απρόοπτες αποτυχίες. 99.99 τις εκατό η εφαρμογή θα είναι αξιόπιστη, με το υπόλοιπο ποσοστό να οφείλεται σε τυχαίες αποτυχίες καθ’ όλη τη διάρκεια του έτους.

- Η εφαρμογή θα πρέπει να είναι εύκολο να συντηρηθεί και να ενημερωθεί. Η αλλαγές σε κώδικα και ενημερώσεις θα πρέπει να γίνονται μέσα σε 2 ώρες για μικρές ενημερώσεις και μέσα σε 24 ώρες για σημαντικές χωρίς ιδιαίτερα προβλήματα.

- Θα πρέπει να υπάρχει κατανοητή τεκμηρίωση και για τους χρήστες αλλά και για τους προγραμματιστές. Θα πρέπει η τεκμηρίωση να καλύπτει 100 τις εκατό της εφαρμογής και να έχει οριστεί βοηθητική τουλάχιστον από το 80 τις εκατό των χρηστών.

- Πρέπει η χρήση της εφαρμογής να είναι εύκολη και να υπάρχει άνετη περιήγηση της. Η ικανοποίηση των χρηστών θα πρέπει να μετριέται από αξιολογήσεις με τουλάχιστον 80 τις εκατό θετική κριτική.

Με το να ακολουθεί αυτές τις προϋποθέσεις, η εφαρμογή θα αξιολογηθεί στην ικανότητα της, με ακρίβεια, να καταγράφει εργατικές ώρες, να παρέχει γρήγορα και ακριβή δεδομένα, να κρατά υψηλή ασφάλεια και διαθεσιμότητα και να υποστηρίζει έναν μεγάλο αριθμό χρηστών.

Στο κεφάλαιο 1, αναλύεται η τρέχουσα κατάσταση, αναφέροντας εργαλεία, συστήματα, αλγορίθμους και μοντέλα τα οποία μπορούν να υλοποιήσουν τους παραπάνω στόχους ή να βοηθήσουν στην υλοποίηση τους. Θα αναφερθούν οι λόγοι για την επιλογή των εν λόγω εργαλείων, ορίζοντας έτσι τα δομικά στοιχεία του συστήματος. Στο κεφάλαιο 2, περιγράφεται η σχεδίαση και η υλοποίηση του συστήματος μαζί με τις τεχνικές λεπτομέρειες. Το κεφάλαιο 3 ασχολείται με την αξιολόγηση του συστήματος και αν έχουν επιτευχθεί όλοι οι στόχοι με έμφαση στις λειτουργικές και μη λειτουργικές απαιτήσεις. Τέλος, στο κεφάλαιο 4 παρουσιάζονται τα συμπεράσματα και οι προτάσεις για μελλοντική εργασία.

ΚΕΦ.1: Ανάλυση τρέχουσας κατάστασης

1.1 Αποθήκευση διαπιστευτηρίων χρηστών για την είσοδο στην ιστοσελίδα

Ένα από τα σημαντικά αν όχι το πιο σημαντικό σημείο είναι η ασφάλεια και η εγκυρότητα των διαπιστευτηρίων των χρηστών. Η ταυτοποίηση είναι η πρώτη άμυνα ενάντια σε μη εξουσιοδοτημένη πρόσβαση, που οδηγεί έτσι στην ανάγκη για σωστή και ισχυρή δημιουργία μεθόδων για αποθήκευση αυτών των δεδομένων. Για τον παραπάνω σκοπό μπορούν να χρησιμοποιηθούν δύο τρόποι: η δημιουργία ενός συστήματος πιστοποίησης μαζί με τις άλλες υπηρεσίες ή η χρήση εξωτερικών συστημάτων που έχουν ήδη δημιουργηθεί για αυτό τον σκοπό. Πριν την ανάλυση των δύο επιλογών να αναφερθεί ότι από τη #1 λειτουργική απαίτηση οι χρήστες θέλουν να έχουν τη δυνατότητα εισόδου με τον λογαριασμό τους στην Google.

Η δημιουργία ενός τέτοιου εσωτερικού συστήματος εμπεριέχει τη διαχείριση των εγγραφών των χρηστών, των στοιχείων σύνδεσης και τη γενική διαχείριση του συστήματος. Με το να επιλεχθεί εσωτερικό σύστημα, επιτρέπει τη δημιουργία προσαρμοσμένων χαρακτηριστικών ανάλογα με τις ανάγκες της εφαρμογής. Επίσης, δε βασίζεται σε τρίτες υπηρεσίες με αποτέλεσμα να μειώνονται τα σημεία όπου η εφαρμογή μπορεί να αποτύχει χωρίς να μπορεί να κάνει κάτι για αυτό. Ταυτόχρονα μειώνονται και τα σημεία από τα οποία μπορούν τα διαπιστευτήρια των χρηστών να διαρρεύσουν. Για να δημιουργηθεί όμως όλη αυτή η υπηρεσία χρειάζεται ένα μεγάλο ανθρώπινο δυναμικό, αρκετό χρόνο για την υλοποίηση του και ειδικές γνώσεις πάνω στο θέμα. Όμως πέρα από τη δημιουργία του, υπάρχει και συνεχής ανάγκη για αναβαθμίσεις ασφαλείας και συντήρησης. Πρέπει επίσης το παραπάνω σύστημα να τηρεί και τους γενικούς κανόνες και κανονισμούς ασφαλείας και να ανανεώνεται όταν αλλάζουν αυτοί. Τα παραπάνω σκεπτικά οδήγησαν λοιπόν στη χρήση ενός εξωτερικού συστήματος για την είσοδο των χρηστών στην ιστοσελίδα.

Τα πλεονεκτήματα και τα σκεπτικά, που αναφέρθηκαν, στη χρησιμοποίηση εξωτερικής υπηρεσίας αντιστρέφονται, με αποτέλεσμα η υλοποίηση εισόδου να είναι εύκολη, με ευκολία στην επέκταση του προγράμματος, όταν ανεβεί ο αριθμός των χρηστών. Πρέπει να σημειωθεί

ότι προστίθεται στα αρνητικά το κόστος το οποίο θα αυξάνεται όσο αυξάνονται οι χρήστες. Μερικά εργαλεία που χρησιμοποιούνται ευρέως για την ταυτοποίηση των χρηστών κατά την είσοδο τους είναι το Firebase Authentication [1], Auth0 [2], Okta [3], Amazon Cognito [4], Supabase [5]. Τα αναφερόμενα εργαλεία έχουν όλες παρόμοιες λειτουργίες και υπηρεσίες με μικρές διαφορές μεταξύ τους. Προσφέρουν τη δυνατότητα ο χρήστης να συνδεθεί με διάφορους τρόπους όπως με διεύθυνση ηλεκτρονικού ταχυδρομείου/κωδικό, διάφορες πλατφόρμες (Google, Facebook, Twitter και άλλα) και με πολύ-επίπεδη πιστοποίηση γνησιότητας. Είναι σχεδιασμένα με σκοπό την εύκολη και γρήγορη επέκταση ανάλογα με την ανάπτυξη της εφαρμογής. Δίνουν προτεραιότητα στην ασφάλεια και προσφέρουν οδηγίες για τη σωστή χρήση τους, ώστε οι εφαρμογές που τις χρησιμοποιούν να τηρούν όλους τους κανόνες ασφαλείας. Παρέχουν τη δυνατότητα γενικής διαχείρισης των λογαριασμών των χρηστών, συμπεριλαμβανομένου, επαναφορά κωδικού και αλλαγής ρόλου χρήστη. Συνεισφέρουν στην ευκολία της εισαγωγής της πιστοποίησης, μέσω SDK και API σε πολλαπλές γλώσσες προγραμματισμού και περιβάλλοντα. Δίνει τη δυνατότητα να επεξεργαστεί τη ροή της προσθέτοντας αναγνωσιμότητα παρόμοια με αυτή της ιστοσελίδας. Τέλος, προσφέρουν την υποστήριξη σε διάφορες πλατφόρμες από εφαρμογές στα κινητά, ιστοσελίδες κλπ. Για να γίνει η σωστή επιλογή εργαλείου πρέπει να μελετηθούν και οι διαφορές τους. Το Firebase, Auth0 και το Supabase προορίζονται από μικρές, start-up, εταιρίες έως και μεγάλες προσφέροντας δωρεάν βαθμίδα για τις υπηρεσίες τους. Αντίθετα, το Okta και το Amazon Cognito τείνουν να εξυπηρετούν μεγάλους οργανισμούς που χρειάζονται περίπλοκα συστήματα. Το Supabase είναι λογισμικό ανοιχτού κώδικα και προσφέρει τη δυνατότητα για αυτό-φιλοξενία (self-host). Φυσικά, το κάθε εργαλείο έχει διαφορά στον τρόπο υλοποίησης του σε κώδικα στην ιστοσελίδα.

Με βάση τις παραπάνω δυνατότητες και διαφορές που έχουν οι υπηρεσίες, αυτές οι οποίες είναι χρήσιμες για την εφαρμογή μας είναι οι εξής: Firebase, Auth0 και Supabase, δεδομένου ότι προσφέρουν τη δυνατότητα δωρεάν βαθμίδας. Η τελική επιλογή βασίζεται κυρίως στις προτιμήσεις του προγραμματιστή και στην περίπτωση της πτυχιακής αυτής επιλέχθηκε το Firebase authentication.

Σε λεπτομερή αναφορά οι λειτουργίες που μπορεί να παρέχει η firebase είναι οι εξής: Προσφέρουν έτοιμη προς χρήση που περιέχει τόσο κώδικα για λόγους σχεδιασμού όσο και μόνο τη λογική. Παρέχονται παραδείγματα κώδικα για το πώς μπορεί κάποιος να εφαρμόσει την εν λόγω βιβλιοθήκη. Στην περίπτωση της πτυχιακής δημιουργούνται αντικείμενα τα οποία

εφαρμόζουν στον ήδη υπάρχοντα σχεδιασμό της ιστοσελίδας. Για τις μικρές εταιρίες, περιβάλλοντα δοκιμών ή και σε περιπτώσεις νεοφυών επιχειρήσεων δίνεται δωρεάν μια βαθμίδα που επιτρέπει έως και 3000 χρήστες ταυτόχρονα ενεργούς στην ιστοσελίδα, με τη δυνατότητα επέκτασης οποιαδήποτε στιγμή. Αποθηκεύει στη μνήμη τον συνδεδεμένο χρήστη ώστε ακόμη και να γίνει ανανέωση της σελίδας δε θα χρειαστεί να ξανά συνδεθεί. Αυτή η πληροφορία χρησιμοποιείται για την ορθή εμφάνιση ορισμένων στοιχείων για τους χρήστες ανάλογα και με τον ρόλο τους. Δίνονται ορισμένες συναρτήσεις ακροατών για τις διάφορες καταστάσεις που μπορεί να βρίσκεται ένας χρήστης όπως το αν συνδέθηκε, αν αποσυνδέθηκε ή αν το τεκμήριο ανανεώθηκε για οποιοδήποτε λόγο. Τέλος, ο χρήστης έχει εξαρχής τη δυνατότητα δημιουργίας και διαγραφής λογαριασμού στο περιβάλλον firebase, το οποίο του δίνει τον πλήρη έλεγχο στον λογαριασμό του. Στην περίπτωση της πτυχιακής όπως αναφέρεται και στη λειτουργική απαίτηση #1, μόνο ένας διαχειριστής μπορεί να δημιουργήσει χρήστες για την ιστοσελίδα.

1.2 Αποθήκευση δεδομένων στην βάση δεδομένων

Μια βάση δεδομένων είναι μία αποθήκη από δεδομένα, όπου αυτά μπορούν να υποστούν διάφορες ενέργειες και να αποθηκευτούν ξανά αυτά ή και καινούργια δεδομένα. Τα δύο είδη που χρησιμοποιούνται ευρέως είναι οι σχεσιακές και αντιστοίχως η μη σχεσιακές βάσεις. Το υπό κεφάλαιο αυτό καλύπτει κυρίως την λειτουργική απαίτηση #4 και την μη λειτουργική απαίτηση #1.

Στις σχεσιακές βάσεις δεδομένων όπως αναφέρει και το όνομα τους, υπάρχουν σχέσεις μεταξύ των δεδομένων. Δημιουργούνται πίνακες και αυτοί οι πίνακες συνδέονται ανάλογα τις σχέσεις. Τα δεδομένα μέσα στους πίνακες είναι αποθηκευμένα σε γραμμές και στήλες. Κάθε γραμμή σε ένα πίνακα έχει ένα μοναδικό κλειδί το οποίο είναι μέρος της παραπάνω σύνδεσης.

Οι μη σχεσιακές βάσεις δεδομένων δεν χρησιμοποιούν τον παραπάνω τρόπο, αποθηκεύοντας σε γραμμές και στήλες, αλλά αντιθέτως χρησιμοποιούν ένα μοντέλο το οποίο είναι βελτιστοποιημένο για τις προϋποθέσεις του συγκεκριμένου τύπου δεδομένων που χρειάζεται

να αποθηκευτεί. Συνήθης τρόπος αποθήκευσης δεδομένων είναι σε μορφή εγγράφου. Αυτός ο τρόπος διαχειρίζεται το έγγραφο σε ένα σύνολο από πεδία σε μορφή γραμματοσειράς και την τιμή αυτών σε μορφή αντικειμένου. Ένα έγγραφο εμπεριέχει όλες τις πληροφορίες για μια οντότητα, οι οποίες πληροφορίες σχετίζονται με την κάθε εφαρμογή. Δεν χρειάζεται ίδια δομή για όλα τα έγγραφα.

Και οι δύο τύποι βάσεων προσφέρουν τις βασικές λειτουργίες όπως αποθήκευση και διαχείριση των δεδομένων, όμως η αρχιτεκτονική, τα πλεονεκτήματα και οι περιορισμοί τους τα καθιστούν κατάλληλα για διαφορετικές εφαρμογές και φόρτους εργασίας. Οι συγκεκριμένες έρευνες [6, 8] αναφέρουν ότι οι μη σχεσιακές βάσεις έχουν υψηλό βαθμό διαπερατότητάς και εύκολη επέκταση. Η απόδοση μπορεί να αυξηθεί με την χρήση της μνήμης του συστήματος στις μη σχεσιακές ενώ αντίθετα στις σχεσιακές πρέπει να χρησιμοποιηθούν εργαλεία τρίτων. Ταυτόχρονα τα δεδομένα στις μη σχεσιακές μπορούν να εισαχθούν χωρίς κάποιο συγκεκριμένο σχήμα. Εκεί όπου μειώνεται η απόδοση της μη σχεσιακής βάσης, όμως, είναι στην περιορισμένη συνοχή και στην αποδεκτή εισαγωγή διπλότυπων εγγράφων που μπορεί ανάλογα με την εφαρμογή να μην είναι επιθυμητό. Τέλος η αναζήτηση δυσχερεί σε σχέση με τις σχεσιακές βάσεις, ειδικά όταν γίνεται σε πολλαπλά ταυτόχρονα πεδία. Στις επόμενες έρευνες [7, 9] που μελετήθηκαν δοκιμάστηκαν συγκεκριμένα ενέργειες όπως η εισαγωγή δεδομένων, επεξεργασίας τους, διαγραφή τους και η αναζήτηση τους. Τα αποτελέσματα που διαπιστώθηκαν είναι παρόμοια με αυτά της έρευνας [6], τονίζοντας την σημαντική διαφορά όταν δοκιμάστηκαν οι τρεις πρώτες λειτουργίες με τις μη σχεσιακές βάσεις να υπερτερούν. Αντιθέτως, στην αναζήτηση οι σχεσιακές βάσεις είχαν πάλι τα ταχύτερα αποτελέσματα. Επίσης αναφέρουν ότι για μικρές έως και μεσαίες εταιρίες χρησιμοποιείται γενικά μια βάση δεδομένων σχεσιακή, ενώ αντίστοιχα για μεγαλύτερες εταιρίες θα ήταν προτιμότερο να χρησιμοποιηθεί μια μη σχεσιακή βάση. Τα κριτήρια επιλογής πάντα επηρεάζονται από την εφαρμογή και τον τύπο δεδομένων που χρησιμοποιεί. Η τελευταία έρευνα [10] επικεντρώνεται κυρίως στο γεγονός ότι ένας από τους σκοπούς των μη σχεσιακών βάσεων είναι η δημιουργία κώδικα στην προτιμώμενη γλώσσα του προγραμματιστή σε αντικειμενοστραφή περιβάλλον.

Με βάση τα παραπάνω αποτελέσματα η επιλογή μίας βάσης δεδομένων εξαρτάται κυρίως από τις προτιμήσεις του υπεύθυνου ανάπτυξης του συστήματος και το ίδιο το σύστημα. Σε αυτήν την πτυχιακή δεν υπάρχουν πολλές διασυνδέσεις μεταξύ των τύπων δεδομένων, μεριμνάτε η επεκτασιμότητα της εφαρμογής μαζί με την ανάγκη για γρήγορες συναλλαγές και η ευκολία στην

ενσωμάτωση μιας τέτοιας βάσης στο σύστημα. Έτσι λοιπόν επιλέχθηκε μία μη σχεσιακή βάση για την αποθήκευση των δεδομένων. Μερικές από αυτές που χρησιμοποιούνται είναι η MongoDB [11], Cassandra [12], Redis [13], CouchDB [14], Amazon DynamoDB [15]. Οι αναφερόμενες βάσεις έχουν παρόμοια λογική και χρήση, με μικρές διαφορές. Σαν μη σχεσιακές βάσεις προσφέρουν δυναμικά, χωρίς σχήμα, μοντέλα για τα δεδομένα και μπορούν να ανταπεξέλθουν σε μεγάλους φόρτους εργασίας. Πέρα από το Amazon DynamoDB, τα υπόλοιπα εργαλεία, έχουν ισχυρή βάση στην κοινότητα ανοιχτού λογισμικού. Εμπεριέχουν μηχανισμούς για την εύκολη αναπαραγωγή των δεδομένων, ελαχιστοποιώντας την πιθανότητα να χαθεί κάποια πληροφορία. Οι διαφορές τους είναι κυρίως στην δομή αποθήκευσης των δεδομένων και την χρήση τους. Η MongoDB, CouchDB και η Amazon DynamoDB αποθηκεύουν σε μορφή εγγράφου. Η Redis και η Amazon DynamoDB μπορούν να αποθηκεύσουν και σε μορφή κλειδιού-τιμής. Τέλος η Cassandra αποθηκεύει σε μορφή στηλών, όπου οι στήλες αυτές είναι ομαδοποιημένες ανάλογα με τις κατηγορίες τους. Όλα στοχεύουν στο να προσφέρουν υψηλή διαθεσιμότητα, με εξαίρεση την MongoDB που προσπαθεί να κρατά μια ισορροπία ανάμεσα σε όλα τα χαρακτηριστικά. Τέλος η Amazon DynamoDB είναι η μόνη η οποία δεν μπορεί να αυτό-φιλοξενηθεί.

Με βάση τις παραπάνω δυνατότητες του κάθε εργαλείου, αυτές που είναι πιο χρήσιμες για την εφαρμογή, είναι η MongoDB, CouchDB και η Amazon DynamoDB η οποία εξαιρείται διότι δεν δίνεται η δυνατότητα για αυτό-φιλοξενία κάτι το οποίο δεν είναι επιθυμητό. Η τελική επιλογή βασίζεται κυρίως σε ποια χαρακτηριστικά στοχεύει το κάθε εργαλείο και την προτίμηση του προγραμματιστή. Για την καλύτερη υλοποίηση των απαιτήσεων που αναφέρθηκαν, η επιλογή της MongoDB είναι η καλύτερη, δεδομένου ότι ισορροπεί την διαθεσιμότητα με την αποδοτικότητα. Η MongoDB μας προσφέρει επίσης, εργαλεία όπως Mongo Compass το οποίο είναι ένα διαχειριστικό, με γραφιστικό περιβάλλον, για την επεξεργασία και εισαγωγή δεδομένων στην βάση. Επίσης προσφέρεται μια δωρεάν βαθμίδα αν κάποιος χρήστης θελήσει να δοκιμάσει την βάση στο υπολογιστικό τους νέφος. Τέλος έχει παραδείγματα σε ποικίλες γλώσσες προγραμματισμού.

1.3 Γλώσσα προγραμματισμού και εργαλεία υλοποίησης της backend υπηρεσίας

Για την υλοποίηση αυτής της υπηρεσίας, backend, υπάρχει η επιλογή της δημιουργίας της από την αρχή ή η επιλογή ενός BaaS συστήματος [16]. Ένα BaaS σύστημα παρέχει υπηρεσίες όπως βάση δεδομένων, εξουσιοδότησή, ειδοποιήσεις ανάλογα την πλατφόρμα, λογική που χρειάζεται για να δουλέψει η εφαρμογή. Όλες αυτές οι λειτουργίες δημιουργούνται και παρέχονται από τρίτους. Το BaaS είναι ιδανικό για μικρές εταιρίες, αυξάνοντας τον γενικό ρυθμό ανάπτυξης της ιστοσελίδας. Προσφέρει αυτόματη επέκταση, μειώνει την συντήρηση που χρειάζεται η εφαρμογή και είναι βασισμένο στο μοντέλο, που χρεώνει ανάλογα με τους πόρους που χρησιμοποιούνται. Σε αντίθεση η υλοποίηση ενός συστήματος από την αρχή προσφέρει ολικό έλεγχο στην δομή της εφαρμογής, ευκολία στην δημιουργία προσαρμοσμένων χαρακτηριστικών, μεγαλύτερη ασφάλεια στα δεδομένα, εφόσον δεν τα διαχειρίζεται κάποια άλλη οντότητα. Με βάση τα παραπάνω και τις τωρινές αλλά και μελλοντικές ανάγκες της εφαρμογής η πτυχιακή υλοποιεί την backend υπηρεσία από την αρχή.

Το επόμενο βήμα είναι η επιλογή της προγραμματιστικής γλώσσας που θα χρησιμοποιηθεί για την υλοποίηση του backend. Αυτές οι οποίες συμπεριλήφθηκαν στην απόφαση ήταν η Nodejs(JavaScript) [17], Python [18, 26], Java [19,], Ruby [20], Go Lang [21]. Συγκεκριμένα η JavaScript είναι ιδανική για να αναλαμβάνει πολλαπλές ενέργειές με υψηλή απόδοση. Η κύρια γλώσσα, για τις εφαρμογές frontend, που χρησιμοποιείται είναι η ίδια, οπότε προσφέρει ένα ενοποιημένο σύστημα βοηθώντας στην ανάπτυξη του. Χρησιμοποιείται κυρίως σε εφαρμογές που βασίζονται σε αλλαγές σε πραγματικό χρόνο όπως συνομιλίες ή APIs. Η Python [26] είναι εύκολη στην κατανόηση και στην χρησιμοποίηση της, κάνοντας την καλή επιλογή για νέους και παλαιούς προγραμματιστές. Είναι εύχρηστη σε διάφορα σύνολα σεναρίων χρήσης όπως ανάπτυξη διαδικτυακών εφαρμογών, ανάλυση δεδομένων, μηχανική μάθηση. Οι Java [25, 26] εφαρμογές μπορούν να εκτελεστούν σε οποιαδήποτε συσκευή έχει εγκατεστημένο JVM. Έχουν υψηλή αποδοτικότητα και μπορούν να διαχειριστούν ανάγκες μεγάλων εταιριών. Υπάρχουν κατανοητά frameworks όπως Spring [22] και Hibernate [23] που βοηθούν στην ανάπτυξη του κώδικα. Στην Ruby μπορεί να χρησιμοποιηθεί το framework Rails [24] το οποίο αυξάνει την ταχύτητα ανάπτυξης δίνοντας προτεραιότητα στην ευκολία και όχι στην παραμετροποίηση. Σενάρια χρήσης που μπορεί να εμφανιστεί η Ruby είναι εφαρμογές διαδικτύου ή ηλεκτρονικό

εμπόριο. Τέλος η GO προσφέρει υψηλή απόδοση, με αποτελεσματική διαχείριση ταυτόχρονων ενεργειών. Είναι δημιουργημένη για υψηλά φόρτους ενεργειών, κατανοητή στην χρήση μειώνοντας τον χρόνο ανάπτυξης και δημιουργία απρόσμενων προβλημάτων. Όλες οι παραπάνω γλώσσες έχουν εκτενή βιβλιοθήκες και κοινότητες για τυχόν ερωτήσεις.

Η επιλογή γλώσσας προγραμματισμού για την backend υπηρεσία επηρεάζεται από τις δυνατότητες τις οποίες προσφέρουν. Η εφαρμογή χρησιμοποιεί Nodejs για να χρησιμοποιήσει την ομαδοποίηση που προσφέρει σε σχέση με το frontend, ψηλή απόδοση σε πολλαπλές ενέργειες και βιβλιοθήκες για την εξομάλυνση και διευκόλυνση της ανάπτυξης κώδικα. Δεδομένου ότι στον χώρο του προγραμματισμού κυριαρχεί κυρίως ο αντικειμενοστραφής προγραμματισμός, επιλέχθηκε συγκεκριμένα να χρησιμοποιηθεί η Typescript [27, 28, 29, 30]. Η Typescript μεταφράζεται σε JavaScript για να εκτελεστεί. Κατά την ανάπτυξη του κώδικα, επιτρέπει την εισαγωγή κλάσεων, τύπων, διεπαφών. Έχει αυτομάτους ελέγχους για συνηθισμένα λάθη τα οποία τα εμφανίζει σε οποιαδήποτε IDE χρησιμοποιείται. Διευκολύνει την επέκταση και διατήρηση του προγράμματος καθ΄ όλη την διάρκεια ζωής ενός προγραμματικού έργου.

Για την διευκόλυνση της ανάπτυξης της υπηρεσίας backend, χρησιμοποιήθηκε ένα framework. Οι επιλογές που ήταν διαθέσιμες ήταν: NestJS [31, 36], FoalTS [32], Ts.Ed [33], Koa [34], Hapi.dev [35], οι οποίες επικεντρώθηκαν κυρίως σε εργαλεία που θα βοηθούσαν σε περιβάλλον Typescript. Όλα τα εργαλεία μπορούν να χρησιμοποιηθούν για JavaScript, αλλά επίσης υποστηρίζουν εις βάθος την Typescript. Βασίζονται κυρίως σε ενδιάμεσο λογισμικό για να διαχειριστούν οποιοδήποτε HTTP αίτημα. Οι ενδιάμεσες συναρτήσεις σε αυτά τα frameworks επεξεργάζονται τα αιτήματα στην σειρά, επιτρέποντας την επαναχρησιμοποίηση κώδικα. Προωθούν την λογική του modular κώδικα όταν δημιουργείται μια εφαρμογή, οργανώνοντας τον κώδικα σε μικρότερα, εύκολα διαχειρίσιμα κομμάτια. Κάθε ένα από αυτά προσφέρει την δυνατότητα του routing. Το NestJS και Ts.ED μέσα από διακοσμητές, το FoalTS μέσα από διαχειριστές, ενώ τα Koa και Hapi.dev τους δημιουργούν και τους διαχειρίζονται κατευθείαν χωρίς ενδιάμεσο μέσο. Είναι όλα δημιουργημένα με την λογική της επεκτασιμότητάς προσφέροντας τρόπους σύνδεσης και επέκτασης, όπου οι προγραμματιστές μπορούν να εισάγουν άλλες βιβλιοθήκες ή και δικό τους προσαρμοσμένο κώδικα. Παρέχουν αναλυτικά και λεπτομερή παραδείγματα μαζί με μεγάλη υποστήριξη από την ανάλογη κοινότητα.

Το NestJS δίνει έμφαση στη δημιουργία ενός καθαρού, δομημένου κώδικα και έχει επίσημή υποστήριξη για διάφορα γνωστά εργαλεία σαν το GraphQL, WebSocket, Microservices και άλλα. Το FoalTS προωθεί την απλότητα και αποδοτικότητα, μειώνοντας την διαμόρφωση που πρέπει κάποιος να κάνει. Το Ts.ED έχει ως στόχο την κλιμάκωση και συντήρηση της εφαρμογής. Το Koa δίνει έμφαση στην απλότητα και ευκαμψία του κώδικα. Το κεντρικό σημείο του Hapi.dev είναι η ασφάλεια και η διαθεσιμότητα που θα παρέχει η εφαρμογή. Να σημειωθεί ότι και τα άλλα εργαλεία πέρα από το NestJS έχουν υποστήριξη για τις υπηρεσίες που αναφέρθηκαν χωρίς κάποια επίσημη αναφορά. Από τις παραπάνω επιλογές προτιμήθηκε το NestJS διότι προσφέρει ένα δομημένο περιβάλλον το οποίο θα βοηθήσει, καθώς επεκτείνεται η εφαρμογή και κατά την διάρκεια της υλοποίησης της. Τέλος σημαντικό επίσης χαρακτηριστικό είναι η υποστήριξη εσωτερικά, για βιβλιοθήκες που θα χρησιμοποιηθούν στην υλοποίηση όπως το mongoose, το πακέτο για την σύνδεση με την MongoDB βάση.

1.4 Υλοποίηση του Frontend πλαισίου της εφαρμογής

Για να μπορέσουν να εφαρμοστούν όλες οι λειτουργικές απαιτήσεις και οι χρήστες να μπορούν να ολοκληρώσουν τις διεργασίες τους πρέπει να επιλεχθούν τα σωστά εργαλεία. Η γλώσσα επιλογής για αυτή την διαδικασία παραμένει η JavaScript, διότι κυριαρχεί σε αυτόν τομέα και γιατί κυρίως μας προσφέρει μια ομαδοποίηση της διαδικασίας μεταξύ των δύο πλαισίων Backend και Frontend.

Για την ανάπτυξη του frontend πλαισίου της εφαρμογής πρέπει επίσης να επιλεχθεί και ο τρόπος. Υπάρχει η επιλογή της χρήσης JavaScript χωρίς την προσθήκη τρίτου λογισμικού αλλά και η χρήση κάποιας βιβλιοθήκης. Με την χρήση JavaScript δεν χρειάζεται η χρήση εξωτερικής βιβλιοθήκης, μειώνοντας τον συνολικό αριθμό κώδικα που χρησιμοποιείται. Επίσης ο προγραμματιστής έχει τον πλήρη έλεγχο του κώδικα και το πώς εκτελείται. Μπορεί να είναι πολύ αποδοτική η παραπάνω χρήση αν υλοποιηθεί σωστά. Όμως από την άλλη πλευρά η υλοποίηση αυτή θα αυξήσει τον χρόνο ανάπτυξης για να μπορέσει να γίνει σωστά, διότι προστίθεται στον φόρτο και ο χειρισμός του DOM, με αποτέλεσμα να δυσκολεύει η συντήρηση του προγράμματος, γιατί αυξάνεται το μέγεθος του κώδικα. Οι βιβλιοθήκες που θα μελετηθούν είναι οι React [37], Preact [38], Inferno [40], Svelte [41]. Η React είναι μια βιβλιοθήκη που έχει ως σκοπό την δημιουργία διεπαφών για τους χρήστες. Προωθεί την χρήση modular συναρτήσεων,

προσφέροντας εύκολη συντήρηση και δομοστοιχείωση. Δημιουργεί ένα εικονικό DOM μειώνοντας τις αλληλεπιδράσεις με το πραγματικό, αυξάνοντας έτσι την απόδοση. Μαζί όμως αυξάνεται η πολυπλοκότητα, η δυσκολία μάθησης ενός τέτοιου εργαλείου και το μέγεθος του τελικού αρχείου. Υπάρχει και η Preact η οποία στην ουσία είναι μια πιο γρήγορη και ελαφριά εναλλακτική της react. Είναι δημιουργημένη για αποδοτικότητα με σημαντικά μικρότερο μέγεθος του τελικού προϊόντος, κατάλληλο για περιβάλλοντα με χαμηλούς πόρους. Χωρίς να μειώνεται σημαντικά η συμβατότητα που προσφέρει η react με άλλες βιβλιοθήκες. Το Inferno είναι μια βιβλιοθήκη παρόμοιου περιεχόμενου με της React, που είναι σχεδιασμένη για αποδοτικότητα και ταχύτητα, ειδικότερα στην φόρτωση μεγάλων συνόλων δεδομένων και περίπλοκα UI. Έχει επίσης συμβατότητα με τις βιβλιοθήκες τις React, προσφέροντας ένα μικρότερο σε μέγεθος αρχείου. Το Inferno όμως προσθέτει πολυπλοκότητα σε κέρδος της ταχύτητα. Τέλος το Svelte είναι περισσότερο ένα framework παρά μια βιβλιοθήκη, το οποίο διαχειρίζεται τα αντικείμενα τα οποία υλοποιούνται και τα μετατρέπει σε καθαρή JavaScript για μεγαλύτερη απόδοση. Είναι εύκολο στην εκμάθηση με καθαρές οδηγίες και ξεκάθαρη δομή. Στην αρνητική πλευρά είναι νέο εργαλείο και δεν υπάρχει εκτενής υποστήριξη από άλλες βιβλιοθήκες και εργαλεία. Για την δημιουργία ενός αντικειμενοστραφή συστήματος, με αυξημένη απόδοση και την πλήρη υποστήριξη από διαφορετικά άλλα εργαλεία και βιβλιοθήκες, επιλέχθηκε η React.

Αναλυτικά η React: επιτρέπει την εισαγωγή δεδομένων από JavaScript κατευθείαν μέσα στο περιεχόμενο. Διαθέτει εξαρτώμενη απεικόνιση με χρήση απλής JavaScript και εύκολη εμφάνιση λιστών μέσω της συνάρτησης map. Επιτρέπει την δημιουργία handlers για την εύκολη λειτουργία αντικειμένων όπως για παράδειγμα το πάτημα ενός κουμπιού. Παρέχει την δυναμική παρουσίαση δεδομένων είτε σε ένα αντικείμενο μιας σελίδας, είτε και σε πολλά τα οποία επιτρέπει την σύνδεση μεταξύ των τιμών που μπορεί μεταβάλλονται.

Η react θα προσφέρει την παρουσίαση της λειτουργικότητας στον χρήστη. Για την σχεδίαση και τρόπο με την οποία μεταφέρεται αυτή η πληροφορία θα χρησιμοποιηθεί βιβλιοθήκη για το UI το οποίο θα προσφέρει εύκολη CSS χωρίς ιδιαίτερη δυσκολία στον προγραμματιστή. Οι βιβλιοθήκες οι οποίες είναι προτιμότερες και ευρέως χρησιμοποιημένες είναι: Tailwind CSS, Bootstrap, Bulma, Chakra UI, and Material-UI. Η Tailwind προσφέρει ευελιξία, με τις χαμηλές επιπέδου κλάσεις, είναι εύκολη στην παραμετροποίηση χωρίς την ανάγκη για αλλαγή της συνηθισμένης δομής. Αναπαράγει μόνο την CSS την οποία χρειάζεται, για να λειτουργήσει,

μειώνοντας έτσι το συνολικό μέγεθος. Έχει σημαντικές διαφορές από την απλή υλοποίηση της CSS οπότε έχει αυξημένη δυσκολία και σε ένα αρχείο HTML αυξάνεται η πολυπλοκότητα του κώδικα. Η Bootstrap προσφέρει ένα μεγάλος πλήθος από αντικείμενα και λειτουργίες, δική της διάταξη και συνοχή από εφαρμογή σε εφαρμογή. Οι παραπάνω λειτουργίες χρειάζονται αρκετή παραμετροποίηση αυξάνοντας την δυσκολία και ταυτόχρονα το μέγεθος της εφαρμογής με την προσθήκη των καινούργιων λειτουργιών. Το Bulma είναι εύκολο στην χρήση, ευέλικτο στην διάταξη του περιεχομένου, ελαφρύ και αυξημένη ανταπόκριση σε όλες τις πλατφόρμες. Λόγω της ελαφριάς του μορφής δεν έχει μεγάλο εύρος από έτοιμες λειτουργίες. Επόμενο στην λίστα επιλογών είναι το Chakra UI που εύκολο στην παραμετροποίηση, δίνει προτεραιότητα στην προσβασιμότητα από την αρχή της υλοποίησης. Προσφέρει ένα πλούσιο περιβάλλον από αντικείμενα δημιουργημένα με React. Δεδομένου αυτού η χρήση της React σε όλο το πρόγραμμα είναι αναγκαία για την ομαλή λειτουργία του. Τέλος το Material UI παρέχει ένα μεγάλο εύρος από ήδη δημιουργημένα αντικείμενα υψηλής ποιότητας, με την επιλογή πάντα για την παραμετροποίηση τους. Βασίζεται στο Google’s Material Design προσφέροντας σύγχρονο αίσθημα. Να επισημανθεί ότι το Chakra UI βασίζεται στην React οπότε η χρήση της πρέπει να συνοδεύεται από παρόμοιο περιβάλλον.

Επιλέχθηκε η χρήση της Tailwind CSS για υψηλή και εύκολη παραμετροποίηση του σχεδιασμού της εφαρμογής, με την χρήση CSS περιβάλλοντος. Επίσης χρησιμοποιηθήκαν αντικείμενα από την βιβλιοθήκη Material UI για να προσφέρουν ένα προσεγμένο περιβάλλον αλληλεπίδρασης για τους χρήστες.

Επιλέχθηκε να χρησιμοποιηθεί το framework, NextJS[46], σε συνδυασμό με την React όπου τα αντικείμενα δημιουργούνται από αυτή και τη λογική την αναλαμβάνει η NextJS. Αναλυτικά η NextJS: διαχωρίζει τις καταστάσεις σε δύο κατηγορίες, των χρηστών και του προγράμματος. Οτιδήποτε υπάγεται στην δεύτερη κατηγορία μπορεί να δημιουργηθεί ένα στατικό περιβάλλον για αυτό, διότι δεν αλλάζει ποτέ, αυξάνοντας έτσι την ταχύτητα φόρτωσης περιεχομένου. Για την παραλαβή δεδομένων παρουσιάζονται επιπλέον λειτουργίες όπως η δυνατότητα αποθήκευσης των αποτελεσμάτων στην μνήμη για γρήγορή επαναχρησιμοποίηση, πιο γρήγορα αιτήματα και την επιλογή επικύρωσης τον ήδη αποθηκευμένων. Προσφέρει εύκολη εισαγωγή Tailwind CSS και άλλων βιβλιοθηκών, για την χρήση στον σχεδιασμό της εφαρμογής. Βελτιστοποιεί την φόρτωση εικόνων, γραμματοσειρών, και άλλων διάφορων σεναρίων, αυξάνοντας έτσι τον ολικό χρόνο ανταπόκρισης του Frontend. Υποστηρίζει επίσης τη γραφή σε

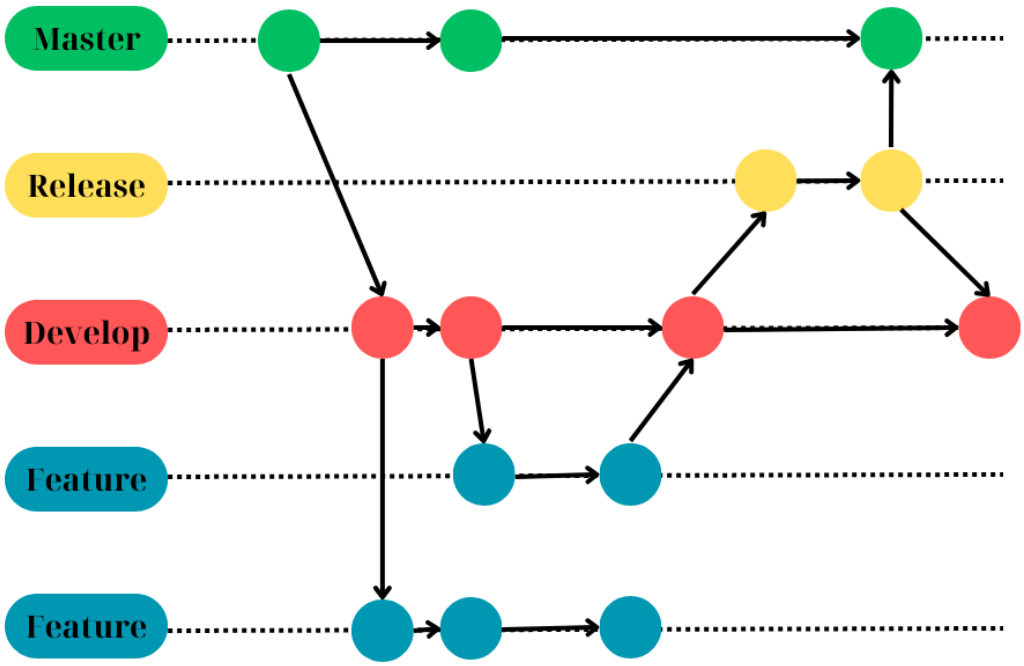
Typescript, όπως χρησιμοποιείται και στο Backend αυξάνοντας την ενοποίηση ολόκληρου του συστήματος.

1.5 Εργαλεία Ανάπτυξης

1.5.1. Git

Το Git είναι ένα σύστημα ελέγχου εκδόσεων για την παρακολούθηση αλλαγών σε αρχεία του υπολογιστή. Δημιουργήθηκε το 2005 από τον Linus Torvalds ο οποίος είναι επίσης και ο δημιουργός των Linux. Δεν είναι απαραίτητη κάποια συγκεκριμένη γλώσσα προγραμματισμού ή framework και μπορεί να χρησιμοποιηθεί ακόμη και σε απλές στατικές HTML ιστοσελίδες. Το σύστημα ελέγχου εκδόσεων αποθηκεύει συνεχώς παλαιές εκδόσεις του έργου και μπορεί κάποιος να επαναφέρει κάποια συγκεκριμένη έκδοση. Περαιτέρω, ο σκοπός του Git είναι η συνεργασία, δεδομένου ότι προσφέρει στους προγραμματιστές ένα κοινό περιβάλλον, και συνεχώς ενημερώνει ποιος έκανε την κάθε αλλαγή και τι ακριβώς άλλαξε. Για αυτό τον λόγο, το έργο θα αναπτύσσεται σαν ένα σύνολο από την αρχή και θα υπάρξει εξοικονόμηση χρόνου λόγω της μειωμένης ύπαρξης συγκρούσεων. Η συλλογή των εκδόσεων μπορεί να είναι πολύ κρίσιμη για τους προγραμματιστές, αφού η αποθήκευση εκδόσεων ενός έργου μετά την επεξεργασία του είναι πολύ ουσιώδης. Επομένως, το Git επιτρέπει στους προγραμματιστές να δημιουργήσουν σημεία αποθήκευσης (commits) κατά την διάρκεια ενός έργου. Αυτά τα σημεία αποθήκευσης επιτρέπουν στους προγραμματιστές να επιστρέψουν σε κάποιο σημείο στο οποίο κάθε συνάρτηση στο έργο λειτουργεί κανονικά, μετά από κάποια λανθασμένη αλλαγή που μπορεί να έγινε. Στο Git οι διακλαδώσεις (branches) διασυνδέουν τα σημεία αποθήκευσης και όταν ένα Git αποθετήριο (repository) δημιουργείται για πρώτη φόρα το Git αρχικοποιεί μία διακλάδωση η οποία ονομάζεται master, και όσο οι προγραμματιστές δημιουργούν νέα σημεία αποθήκευσης, τόσο η διακλάδωση master αναπτύσσεται. Επίσης προσφέρει την δυνατότητα της δημιουργίας διακλάδωσης συγκεκριμένου χαρακτηριστικού (feature branch), ο οποίος είναι ένας όρος που χρησιμοποιείται κατά την ανάπτυξη της εφαρμογής για την εισαγωγή ενός νέου χαρακτηριστικού χωρίς να αλλάξει η ροή της κύριας διακλάδωσης. Με αποτέλεσμα, ένας προγραμματιστής να μπορεί να δουλέψει στην διακλάδωση του χαρακτηριστικού και οι άλλοι να συνέχιζαν την επεξεργασία της κεντρικής διακλάδωσης. Όταν το χαρακτηριστικό είναι έτοιμο

και λειτουργικό, μπορεί να συγχωνευτεί με το κεντρικό και δημιουργείται ένα καινούργιο σημείο αποθήκευσης.



Εικ. 1. Git διακλαδώσεις

1.5.2. Visual Studio Code

Το Visual Studio Code (VsCode) προσφέρει στους προγραμματιστές μία νέα επιλογή, η οποία συνδυάζει την απλότητα και διευκολύνει την εμπειρία της χρήσης ενός επεξεργαστή κώδικα, παρέχοντας ότι οι προγραμματιστές χρειάζονται για τον βασικό κύκλο επεξεργασίας κώδικα και αποσφαλμάτωσης. Είναι ο πρώτος επεξεργαστής κώδικα στην οικογένεια με τα εργαλεία του visual studio, με την δυνατότητα εκτέλεσης σε περιβάλλοντα Windows, OS X και Linux. Το VsCode διαθέτει ένα ισχυρό και γρήγορο επεξεργαστή κώδικα, ιδανικό για καθημερινή χρήση, συμπεριλαμβανομένου εύκολη πλοήγηση στον κώδικα, προσαρμόσιμες συνδέσεις πλήκτρων, υπογράμμιση λάθους σύνταξης, αυτόματη αντιστοίχιση παρενθέσεων και ενσωματωμένη υποστήριξη για δεκάδες γλώσσες. Οι προγραμματιστές συνήθως χρειάζονται να δουλέψουν με κώδικα περισσότερο από ότι με απλό κείμενο. Το VsCode υποστηρίζει την αυτόματη συμπλήρωση κώδικα και πλούσια σημασιολογική πλοήγηση. Ενσωματώνεται με υπάρχοντα

εργαλεία και μπορεί να εκτελεί δοκιμές και να εκτελεί σχεδόν οποιαδήποτε εντολή μέσα από το ίδιο. Τα σφάλματα και οι προειδοποιήσεις είναι ορατά και καθιστούν τη ροή εργασιών πολύ εύκολη και ταχύτερη.

1.6 Πίνακας Εργαλείων που επιλέχθηκαν

Κατηγορίες	**Εργαλεία**
Backend	Firebase
	MongoDB
	JavaScript/Typescript
	NestJS
Frontend	React
	Tailwind CSS
	NextJS
Third Party Tools	Git
	Visual Studio Code

Πίν.1: Εργαλεία που επιλέχθηκαν

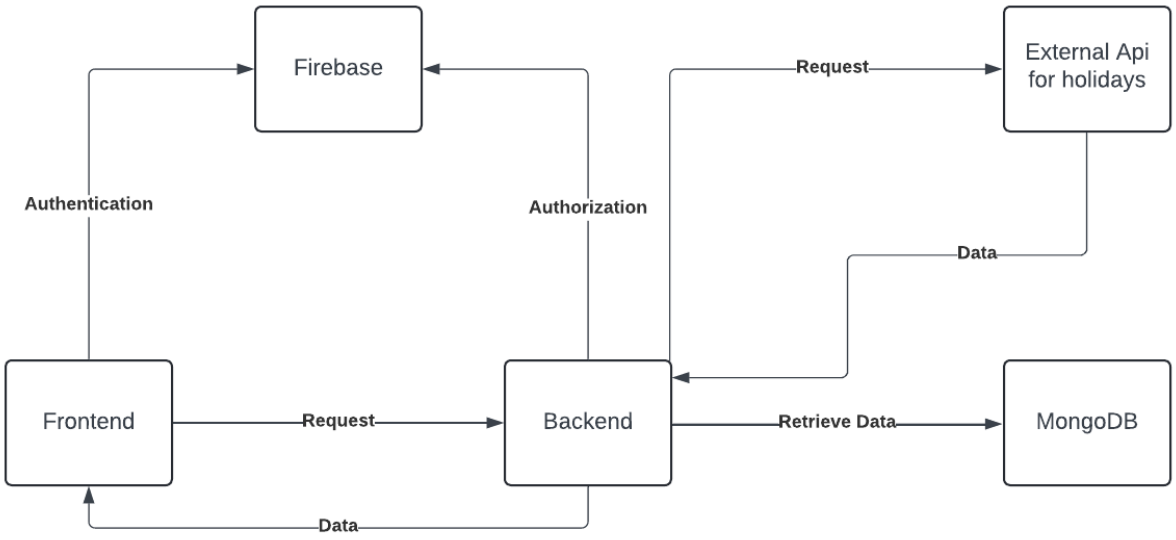
ΚΕΦ.2: Σχεδίαση και υλοποίηση συστήματος/εφαρμογής

2.1 Αρχιτεκτονική συστήματος

Η ανάλυση της αρχιτεκτονικής ενός συστήματος παρέχει μια όψη της δομής της εφαρμογής από άποψης υψηλού επιπέδου, δηλαδή δίνει έμφαση στα κύρια αντικείμενα και πως αυτά αλληλοεπιδρούν μεταξύ τους, για να μπορέσει να υλοποιηθεί το τελικό αποτέλεσμα. Τα αντικείμενα αυτά μπορούν να διαχωριστούν σε τέσσερις κατηγορίες. Το UI, το στρώμα δεδομένων, οποιαδήποτε APIs και εξωτερικές υπηρεσίες που χρησιμοποιούνται και τέλος το στρώμα ασφαλείας στην λογική της σύνδεσης των χρηστών. Το UI ουσιαστικά είναι το frontend κομμάτι της εφαρμογής, όπου ο χρήστης αλληλοεπιδρά με είσοδούς, όπως φόρμες, κουμπιά και οτιδήποτε δια δραστικό στοιχείο εμπεριέχεται στην ιστοσελίδα. Τα αποτελέσματα από τις πράξεις των χρηστών προβάλονται με εμφανίσιμο και αποδοτικό τρόπο, βεβαιώνοντας την εύκολη περιήγηση και χρήση από τους χρήστες. Το στρώμα δεδομένων είναι υπεύθυνο για την αποθήκευση, ανάκτηση και γενική διαχείριση των δεδομένων της εφαρμογής. Εμπεριέχει την βάση δεδομένων και την λογική της χρήσης της με το πρόγραμμα, διασφαλίζοντας διαθεσιμότητα, ακεραιότητα και συνέπεια. Το αντικείμενο που διαχειρίζεται τις εξωτερικές υπηρεσίες είναι υπεύθυνο για την επικοινωνία του συστήματος με κάποια υπηρεσία η οποία προσφέρει δεδομένα ή και κάποιο εργαλείο. Τέλος είναι το στρώμα ασφαλείας για την σύνδεση των χρηστών στην εφαρμογή με τον λογαριασμό Google, ο οποίος έχει εξουσιοδοτηθεί από διαχειριστή του συστήματος. Διαβεβαιώνεται μέσα από την χρησιμοποίηση ρόλων ότι διαφορετικοί τύποι χρηστών θα έχουν πρόσβαση σε διαφορετικές υπηρεσίες της εφαρμογής.

Η αναλυτική περιγραφή του συστήματος είναι η εξής. Οι χρήστες επισκέπτονται την ιστοσελίδα και συνδέονται με τον λογαριασμό Google τους. Αν οι λογαριασμοί τους έχουν εξουσιοδοτηθεί στην firebase, τότε εισέρχονται κανονικά στην εφαρμογή. Έπειτα σε κάθε ενέργεια στην οποία ο χρήστης ενεργεί στέλνεται ένα αίτημα στην backend υπηρεσία της εφαρμογής που εμπεριέχει το μοναδικό πεδίο με το οποίο γίνεται επικύρωση. Αν είναι σωστό και ο χρήστης έχει τον απαραίτητο ρόλο για την ενέργεια, τότε επιστρέφεται το αποτέλεσμα και παρουσιάζεται με εμφανίσιμο και απλό στην κατανόηση τρόπο στον χρήστη. Αυτή η ακολουθία γίνεται για οποιαδήποτε ενέργεια ο χρήστης πράττει η οποία χρειάζεται αντικείμενα από το backend. Να σημειωθεί ότι το σύστημα καθημερινά στις δώδεκα το βράδυ στέλνει αίτημα σε μια εξωτερική

υπηρεσία η οποία μας επιστρέφει ποιες μέρες είναι αργία μέσα στον χρόνο. Παρακάτω παρατίθεται το Σχήμα 1 Διάγραμμα αρχιτεκτονικής συστήματος το οποίο αναφέρεται στην αρχιτεκτονική του συστήματος, δείχνοντας την υψηλή λογική του.

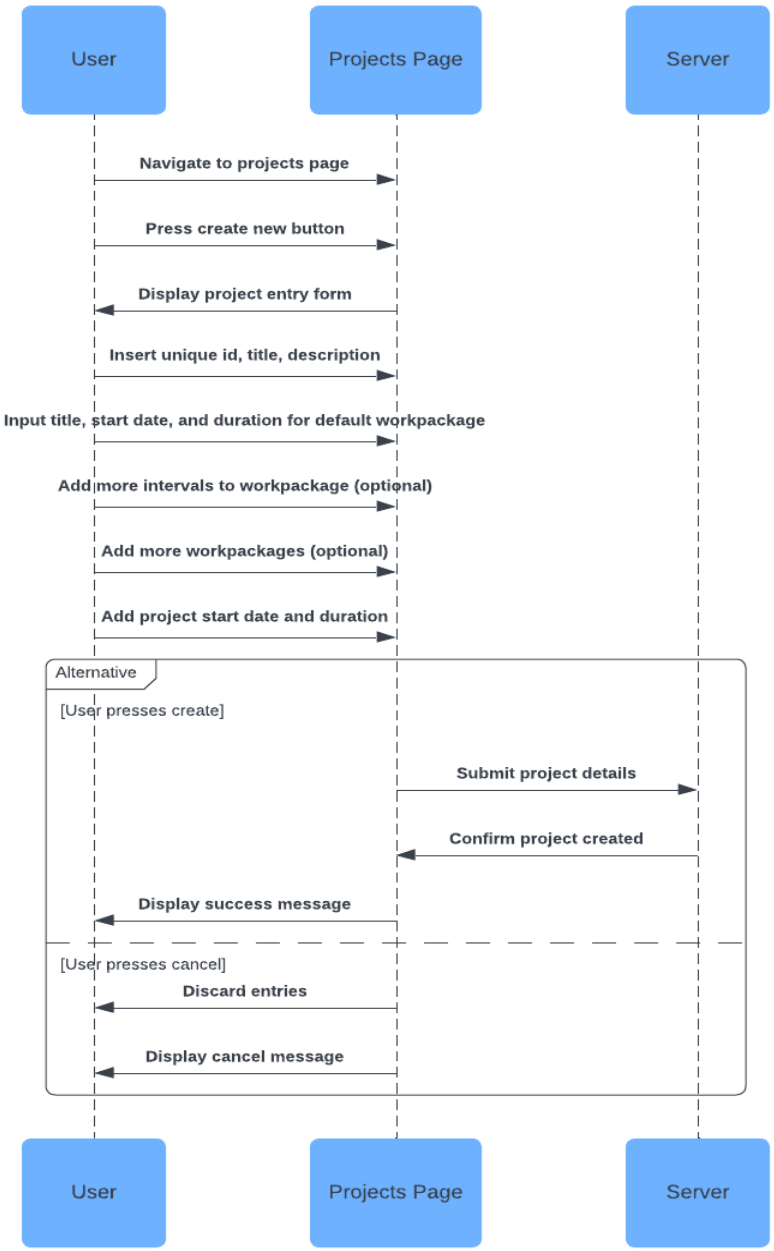


Σχήμα 2 Διάγραμμα αρχιτεκτονικής συστήματος

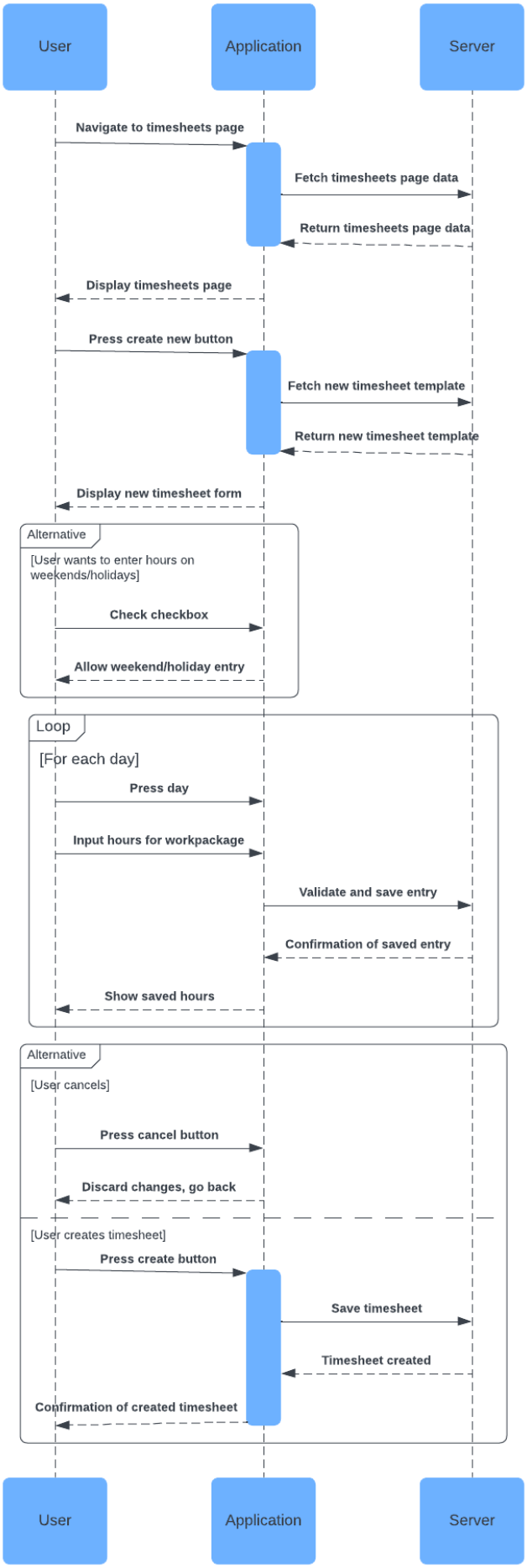
2.2 User interface αρχιτεκτονική

Για την επίδειξη του UI θα αναφερθούν δυο διαδικασίες που μπορούν οι χρήστες να προβούν κατά την επίσκεψή τους στην ιστοσελίδα. Η πρώτη είναι η δημιουργία ενός καινούργιου έργου. Ο χρήστης θα προσέλθει στην σελίδα με τα έργα. Στο συγκεκριμένο σημείο εμφανίζονται όλα τα δημιουργημένα έργα και ένα GNANTT διάγραμμα για την πιο εύκολη παρουσίαση τους. Επίσης διατίθεται τρόπος για την αναζήτηση συγκεκριμένου έργου μέσου του μοναδικού χαρακτηριστικού. Έπειτα ο χρήστης μέσω ενός κουμπιού μπορεί να εισέλθει για την δημιουργία ενός έργου και στην καινούργια σελίδα ο χρήστης εισάγει όλα τα δεδομένα ενός έργου, σε μορφή φόρμας. Τα δεδομένα είναι, το μοναδικό χαρακτηριστικό τους, τίτλος, επεξήγηση, τουλάχιστον ένα πακέτο εργασίας και τα στοιχεία του. Τέλος δίνεται η επιλογή για δημιουργία ή απόρριψη του έργου, μαζί με την εμφάνιση ενός μηνύματος ανάλογα με την κατάσταση. Αντίστοιχα η δεύτερη διαδικασία είναι η δημιουργία ενός φύλλου χρονοχρέωσης. Ακολουθούνται τα ίδια βήματα όπως και για το έργο με διαφορετική φόρμα για την είσοδο των δεδομένων των χρηστών. Εμφανίζεται στον χρήστη μία φόρμα σε μορφή ημερολογίου. Ο

χρήστης επιλέγει πάνω σε κάθε μέρα και εισάγει τις ώρες για κάθε πακέτο εργασίας. Αν θελήσει ο χρήστης να εισάγει ώρες σε ημέρα η οποία είναι Σάββατο η Κυριακή τότε πρέπει να επιλέξει μια ρύθμιση για να του δοθεί πρόσβαση. Τέλος ακολουθεί την ίδια διαδικασία με τα έργα και υποβάλλει το φύλλο. Παρακάτω παρουσιάζονται το *Σχήμα 3 Διάγραμμα ροής για την δημιουργία έργου* και το *Σχήμα 4 Διάγραμμα ροής για την δημιουργία φύλλου εργασίας* για την οπτική παρουσίαση των δύο διαδικασιών.



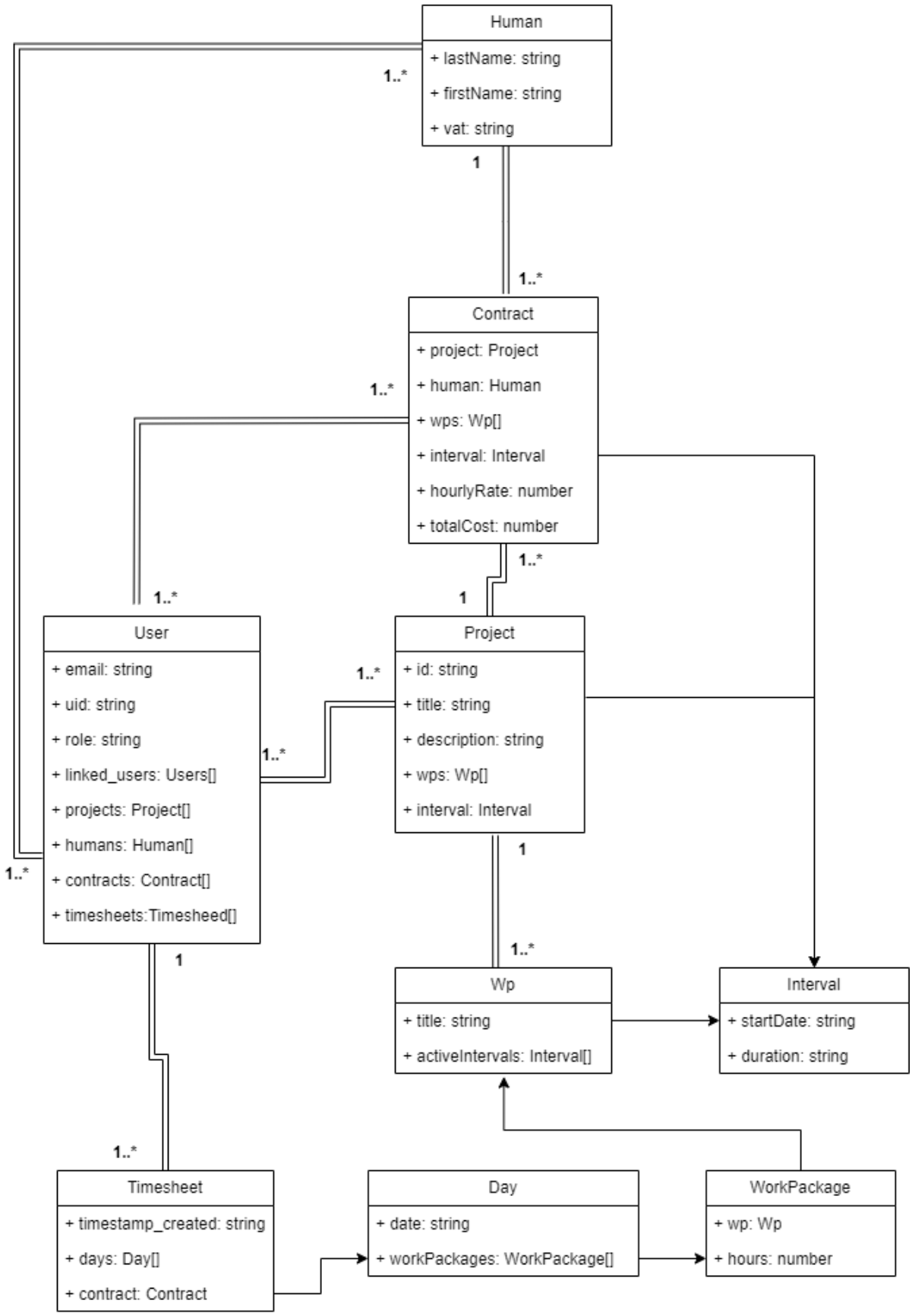
Σχήμα 5 Διάγραμμα ροής για την δημιουργία έργου


User
Application
Server
Navigate to timesheets page
Fetch timesheets page data
Return timesheets page data
Display timesheets page
Press create new button
Fetch new timesheet template
Return new timesheet template
Display new timesheet form
Alternative
[User wants to enter hours on weekends/holidays]
Check checkbox
Allow weekend/holiday entry
Loop
[For each day]
Press day
Input hours for workpackage
Validate and save entry
Confirmation of saved entry
Show saved hours
Alternative
[User cancels]
Press cancel button
Discard changes, go back
[User creates timesheet]
Press create button
Save timesheet
Timesheet created
Confirmation of created timesheet
User
Application
Server

Σχήμα 6 Διάγραμμα ροής για την δημιουργία φύλλου εργασίας

2.3 Αρχιτεκτονική Βάσης Δεδομένων

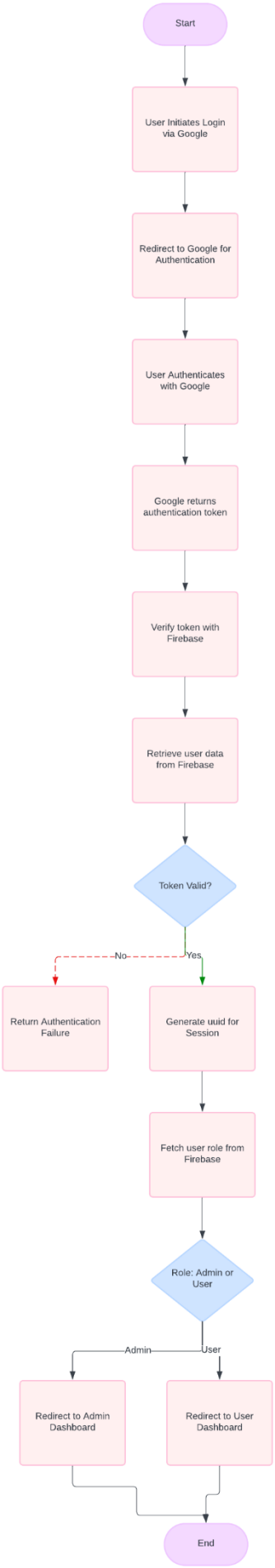
Στο κεφάλαιο αυτό θα σχολιασθεί ο σχεδιασμός της βάσης δεδομένων, οι οντότητες που δημιουργούνται και πως αυτές συνδέονται μεταξύ τους. Οι βασικές οντότητες που δημιουργήθηκαν είναι Human, Contract, Project, User, Wp, Timesheet. Δημιουργήθηκαν επίσης διεπαφές για την επαναχρησιμοποίηση του κώδικα, Interval, Day, WorkPackage. Τα βασικά χαρακτηριστικά του καθενός είναι τα εξής: Για την οντότητα Human είναι το ονοματεπώνυμο των εργαζομένων και ο μοναδικός αριθμός που τους αντιστοιχεί. Στην Contract εμπεριέχονται τα Project, Human, Wps στα οποία απευθύνεται το συγκεκριμένο συμβόλαιο. Επίσης έχει το ολικό κόστος, ωρομίσθιο και την διάρκεια του συμβολαίου. Τα Projects έχουν τον μοναδικό τους αναγνωριστικό, τίτλο, επεξήγηση, τα work packages τα οποία περιέχει και την διάρκεια του. Κάθε User έχει μια ηλεκτρονική διεύθυνση, ένα αναγνωστικό για την σύνδεση μέσω της firebase υπηρεσίας, ρόλο, άλλους Users με τους οποίους είναι linked ο λογαριασμός για τον διαμερισμό όλων των στοιχείων, και τα Projects, Humans, Contracts, Timesheets τα οποία έχει δημιουργήσει ο χρήστης ή κάποιος άλλος συνδεδεμένος χρήστης. Το κάθε Wp έχει ένα τίτλο και έναν πίνακα από ημερομηνίες για το πότε είναι ενεργό. Τέλος η οντότητα Timesheet εμπεριέχει την ημερομηνία δημιουργίας του φύλλου χρονοχρέωσης, το συμβόλαιο το οποίο αφορά και οι ημέρες που έχει κάποιος δουλέψει. Οι διαπαφές έχουν τα εξής χαρακτηριστικά: η Interval έχει την ημερομηνία έναρξης και την διάρκεια, η Day την ημερομηνία και τα WorkPackages που αφορούν εκείνη την ημέρα και WorkPackage είναι το wp το οποίο αφορά μαζί με τις ώρες εργασίας. Παρακάτω παρουσιάζεται το *Σχήμα 7 Διάγραμμα κλάσεων* με αναλυτικές πληροφορίες για τις οντότητες.



Σχήμα 8 Διάγραμμα κλάσεων

2.4 Αρχιτεκτονική Ασφαλείας

Η αρχιτεκτονική ασφαλείας σχεδιάζει, εφαρμόζει και διαχειρίζεται τους μηχανισμούς ασφαλείας μίας εφαρμογής. Εμπεριέχει τα εργαλεία και τις μεθοδολογίες που είναι απαραίτητες για την προστασία του συστήματος από πιθανούς κινδύνους, διασφαλίζοντας την ακεραιότητα και διαθεσιμότητα των δεδομένων. Ο κύριος σκοπός της συγκεκριμένης αρχιτεκτονικής είναι να προταθεί ένα σύστημα το οποίο μπορεί να ανταπεξέλθει από εξωτερικές επιθέσεις αλλά και από εσωτερικές. Για την διασφάλισή των παραπάνω έχει δημιουργηθεί μια σειρά βημάτων για την εξουσιοδότηση των χρηστών. Ο χρήστης ξεκίνα την διαδικασία σύνδεσης μέσω Google, όπου ο ακολουθεί τα βήματα για πιστοποίηση. Η Google επιστρέφει στην εφαρμογή ένα μοναδικό token το οποίο στέλνεται στην Firebase για να γίνει ταυτοποίηση και να παρθούν οι πληροφορίες του χρήστη. Αν αυτό το token είναι έγκυρο τότε από τις πληροφορίες, ελέγχεται ο ρόλος του χρήστη για την εμφανίσει του σωστού μενού. Να σημειωθεί ότι το μοναδικό token είναι ενεργό για μία συγκεκριμένη περίοδο για κάθε σύνδεση για να μεγιστοποιηθεί η ασφάλεια. Επίσης ο χρήστης πρέπει να βρίσκεται αποθηκευμένος και στην βάση της εφαρμογής, οπότε ακόμη και αν κάποιος καταφέρει να εισέλθει στην Firebase ένα λογαριασμό ο οποίος δεν έχει εξουσιοδοτηθεί δεν θα μπορέσει να επικοινωνήσει με το backend σύστημα. Παρακάτω παρατίθεται το *Σχήμα 9 Διάγραμμα διεπαφής χρήστη* που επεξηγεί το παραπάνω σύστημα.


Start
User Initiates Login via Google
Redirect to Google for Authentication
User Authenticates with Google
Google returns authentication token
Verify token with Firebase
Retrieve user data from Firebase
Token Valid?
No
Yes
Return Authentication Failure
Generate uuid for Session
Fetch user role from Firebase
Role: Admin or User
Admin
User
Redirect to Admin Dashboard
Redirect to User Dashboard
End

Σχήμα 10 Διάγραμμα διεπαφής χρήστη

2.5 Πρακτικές υλοποίησης κώδικα

Στον κόσμο του προγραμματισμού, το να ακολουθεί κάποιος τις καλύτερες πρακτικές υλοποίησης κώδικα είναι απαραίτητο για να εγγυηθεί η ποιότητα, συντήρηση και αποτελεσματικότητα. Οι παραπάνω μέθοδοι βοηθούν στην δημιουργία κώδικα ο οποίος όχι μόνο λειτουργεί αλλά είναι ευανάγνωστος και εύκολος στην συντήρηση. Στην περίπτωση της ανάπτυξης αυτής της εφαρμογής χρησιμοποιήθηκαν οι εξής. Για τις ονομασίες των μεταβλητών χρησιμοποιήθηκαν ονόματα τα οποία είναι εύκολα στην κατανόηση του περιεχόμενου. Ο κώδικας είναι διαχωρισμένος σε μικρότερες, επαναχρησιμοποιήσιμες συναρτήσεις ή τμήματα. Υπάρχει ξεκάθαρή αναφορά για την χρήση του προγράμματος και ανάπτυξης της. Επίσης σχολιασμός είναι διαθέσιμος στον κώδικα για εύκολο προγραμματισμό. Έχουν πραγματοποιηθεί αρκετές δοκιμές ώστε να εγγυηθεί η σωστή χρήση του συστήματος και για τυχόν σφάλματα τα οποία μπορεί να εμφανιστούν στην εφαρμογή, έχουν δημιουργηθεί συναρτήσεις για την σωστή διαχείριση τους και την αποφυγή αποτυχίας της.

ΚΕΦ.3: Αξιολόγηση συστήματος/εφαρμογής

3.1 Σκοποί της αξιολόγησης συστήματος

Η αξιολόγηση της αποδοτικότητας, επίδοσης και της γενικής επιρροής της εφαρμογής είναι σημαντική για να αναγνωριστούν οι δυνατότητες και αδυναμίες τις οποίες εμπεριέχει το σύστημα, μέρη τα οποία μπορούν να βελτιστοποιηθούν και να επικυρωθεί ότι η εφαρμογή καλύπτει επαρκώς τους στόχους και τις απαιτήσεις. Για να επιτευχθούν οι παραπάνω σκοποί θα χρησιμοποιηθούν οι εξής μεθοδολογίες. Δοκιμές σχετικά με την απόδοση του συστήματος, όπως το χρονικό διάστημα το οποίο χρειάζεται η εφαρμογή να απαντήσει σε κάποιο ερώτημα. Το μέγεθος της πληροφορίας που μπορεί να διαχειριστεί ταυτόχρονα η εφαρμογή και πόσους πόρους καταναλώνει για την σωστή λειτουργία της. Δοκιμές σχετικά με το πόσο χρησιμοποιήσιμη είναι η εφαρμογή από τους χρήστες μέσω διάφορων συνεντεύξεων και ερευνών. Τέλος θα συγκριθεί με ήδη υπάρχοντες εφαρμογές οι οποίες παρέχουν παρόμοιες υπηρεσίες για το ίδιο πρόβλημα.

3.2 Αξιολόγηση ευχρηστίας του συστήματος

Το πόσο εύκολη είναι η χρήση του προγράμματος είναι ένα απαραίτητο κομμάτι για την συνολική αξιολόγηση του συστήματος. Δίνει έμφαση στο πως οι χρήστες μπορούν να αλληλοεπιδράσουν με την εφαρμογή και τα ποσοστά ευχαρίστησης τους. Οι κύριοι σκοποί της συγκεκριμένης υπό-ενότητας είναι να ελεγχθούν η χρήση του συστήματος, τα ποσοστά ευχαρίστησης και τυχόν προβλήματα ή εμπόδια που ο χρήστης μπορεί να αντιμετωπίσει. Για να μπορέσει να πραγματοποιηθεί το παραπάνω, σε πρώτο στάδιο δόθηκε πρόσβαση στην εφαρμογή σε άτομα που διαχειρίζονται τέτοιες ερευνητικές ομάδες για την συλλογή σχολίων. Τα σχόλια τα οποία παραλήφθηκαν ήταν τα εξής: οι ειδοποιήσεις οι οποίες εμφανίζονται στους χρήστες όταν προβαίνουν σε οποιαδήποτε ενέργεια, πρέπει να διαρκούν περισσότερη περίοδο πριν εξαφανιστούν αυτόματα. Στις ειδοποιήσεις επίσης σχολίασαν ότι θα προτιμούσαν μια εκτενής περιγραφή του προβλήματος. Επίσης άτομα χωρίς άμεση γνώση της διάταξης της εφαρμογής αντιμετωπίζουν πρόβλημα ως προς την σειρά ενεργειών που πρέπει να προβεί ένας χρήστης, δεδομένου ότι για την δημιουργία ενός συμβολαίου είναι αναγκαίο ο χρήστης να έχει

δημιουργήσει τουλάχιστον ένα άτομο στο ανθρώπινο δυναμικό και ένα έργο. Αντίστοιχα και για την δημιουργία ενός φύλλου εργασίας πρέπει να έχει δημιουργηθεί τουλάχιστον ένα συμβόλαιο. Πέρα από τις προσθήκες και αλλαγές αυτές οι χρήστες ενημέρωσαν ότι στην δημιουργία της κάθε οντότητας, η διαδικασία ήταν ξεκάθαρη και δεν υπήρχαν προβλήματα στην οποιαδήποτε ενέργεια. Για την σωστή σχεδίαση της εφαρμογής μελετήθηκαν επίσης οι ευρετικές αξιολογήσεις του Nielsen. Συγκεκριμένα ήταν οι εξής:

1. Ορατότητα του συστήματος.

    a. Το σύστημα πρέπει να ενημερώνει τους χρήστες για οποιαδήποτε ενέργεια γίνεται στο υπόβαθρο, σε ένα ευνοϊκό σύντομο χρόνο από την έναρξη της ενέργειας αυτής.
2. Ομοιότητες μεταξύ του πραγματικού κόσμου και αυτουνού της εφαρμογής.
    a. Το σύστημα πρέπει να βρίσκεται σε γλώσσα και διάλεκτο την οποία καταλαβαίνει ο χρήστης. Χρήση λέξεων, προτάσεων και ιδεολογιών που ο χρήστης βρίσκει οικία χωρίς την προσθήκη προγραμματιστικών ορολογιών. Η εμφάνιση των λέξεων, προτάσεων και ιδεολογιών πρέπει να γίνεται με λογική σειρά ώστε να είναι εύκολη στην κατανόηση.
3. Ελευθερία στον χειρισμό της εφαρμογής από τον χρήστη.
    a. Ο χρήστης θα πρέπει να μπορεί με ευκολία να απομακρυνθεί από μία ενέργεια που έγινε κατά λάθος χωρίς να χρειάζεται να παρέμβει σε μία εκτενής και κουραστική διαδικασία.
4. Συνοχή και Προδιαγραφές.

    a. Ο χρήστης πέρα από την εφαρμογή αυτή, αξιοποιεί τον χρόνο του και σε πολλές άλλες. Οι εμπειρίες με αυτές προδιαθέτουν τον χρηστη στο να έχει κάποιες προσδοκίες, οπότε η εφαρμογή θα πρέπει να συμβαδίζει με την γενική σχεδιαστική λογική στην αγορά.

5. Αναγνώριση αντί για Ανάκληση.

a. Πληροφορίες οι οποίες χρησιμοποιούνται ξανά από τον χρήστη σε διαφορετικά σημεία πρέπει να εμφανίζονται αυτόματα χωρίς τον χρήστη να πρέπει να τα θυμάται, δεδομένου ότι οι άνθρωποι έχουν περιορισμένη βραχυπρόθεσμη μνήμη.

6. Ευελιξία και αποτελεσματικότητα χρήσης.

    a. Παρακάμψεις κρυμμένες, από τους αρχάριους χρήστες, μπορούν να βοηθήσουν αυτούς με περισσότερη εμπειρία επιταχύνοντας την οποιαδήποτε διαδικασία τους. Η επιλογή για την παραμετροποίηση διάφορων ενεργειών από τους χρήστες για την ατομική διευκόλυνση τους είναι επίσης ευνοϊκή.

7. Αισθητικός και Μινιμαλιστικός Σχεδιασμός.

    a. Οι διεπαφές δεν πρέπει να περιέχουν άχρηστες πληροφορίες ή πληροφορίες που χρησιμοποιούνται/χρειάζονται σπάνια. Κάθε έξτρα κομμάτι τέτοιας πληροφορίας ανταγωνίζεται τις σχετικές πληροφορίες με αποτέλεσμα να χάνουν την σημαντικότητα τους.

8. Βοήθεια στους χρήστες να αναγνωρίσουν, να διαγνώσουν και να ανακτήσουν από σφάλματα.

    a. Τα μηνύματα σφαλμάτων θα πρέπει να είναι αποτυπωμένα με κανονική διάλεκτο χωρίς την ύπαρξη μοναδικών κωδικών προβλημάτων. Θα πρέπει να αποτυπώνουν το πρόβλημα με σαφήνεια και να δίνουν ξεκάθαρες οδηγίες στους χρήστες για τα επόμενα βήματα.

9. Τεκμηρίωση.

    a. Είναι προτιμότερο αν το σύστημα είναι σχεδιασμένο με τέτοιο τρόπο που δεν χρειάζεται επιπλέον τεκμηρίωση. Στις περιπτώσεις που αυτό όμως δεν είναι

εφικτό, είναι πολύ σημαντικό να υπάρχει τεκμηρίωση αν όχι για όλη την εφαρμογή, τουλάχιστον για κριτικά σημεία της.

3.3 Συγκριτική Ανάλυση με Παρόμοιες Εφαρμογές

Ένας επίσης χρήσιμος τρόπος για την αξιολόγηση ενός συστήματος είναι η σύγκριση του προγράμματος που δημιουργήθηκε μαζί με υπάρχοντα συστήματα που προσφέρουν παρόμοιες ή και ακριβώς ίδιες λειτουργίες. Η σύγκριση αυτή θα τονίσει τα πλεονεκτήματα της τωρινής εφαρμογής και τι λειτουργίες μπορούν να υλοποιηθούν στο μέλλον. Για την επιλογή των εξωτερικών παρόμοιων εφαρμογών χρησιμοποιήθηκαν τα εξής κριτήρια. Οι εφαρμογές πρέπει να προσφέρουν σχετικά παρόμοιες λειτουργίες, να στοχεύουν να εξυπηρετήσουν παρόμοιο κοινό χρηστών, να έχουν αναπτυχθεί με παρόμοια ή ίδια εργαλεία και τεχνολογίες και τα εξωτερικά συστήματα να είναι αναγνωρίσιμα στην αγορά. Τα κριτήρια αυτά οδήγησαν στην επιλογή των τριών εφαρμογών: Asana, Trello, Microsoft Project. Η σύγκριση αυτών των εργαλείων θα επικεντρωθεί κυρίως στις λειτουργίες που παρέχουν, τον σχεδιασμό της διεπαφής που χρησιμοποιούν οι χρήστες, την επίδοση του συστήματος, δυνατότητα κλιμάκωσης, ποσοστό ευχαρίστησης χρηστών και τέλος την προσβασιμότητα. Αναλυτικά η Asana παρέχει προχωρημένες λειτουργίες για τα έργα, προβολή δεδομένων σε χρονική σειρά με εμφανίσιμο τρόπο, υψηλή προσαρμοστικότητα καθιστώντας το ιδανικό για ομάδες. Η σχεδίαση της εφαρμογή είναι προσεγμένη και κάθε αντικείμενο έχει μελετηθεί για το ρόλο του. Οι νέοι χρήστες έχουν μικρή δυσκολία στην εξοικείωση με την εφαρμογή, αλλά έπειτα από αυτό το στάδιο είναι εύκολο στην περιήγηση και χρήση. Προσφέρει μεγάλη απόδοση και αξιοπιστία, με καινούργιες λειτουργίες να εισάγονται σε συχνό ρυθμό, δεδομένου του μεγάλου αριθμού υποστήριξης. Σχεδιασμένο για μεγάλες ομάδες και σενάρια που συμπεριλαμβάνουν υψηλούς ρυθμούς ροής δεδομένων, με υψηλή θετική αντίληψη των χρηστών για την εφαρμογή, με μόνο αρνητικό την δυσκολία εισαγωγής στο περιβάλλον. Το Trello είναι απλό στην χρήση με την διαχείριση να γίνεται κυρίως μέσω καρτών, πινάκων και λιστών. Έχει λιγότερες προχωρημένες λειτουργίες αλλά παρέχει ευκολία στην χρήση μέσω απλής και λειτουργικής διεπαφής, ακόμη και για νέα μέλη. Είναι γνωστό για την σταθερότητα και συνέπεια που παρέχει κατά την διάρκεια με σπάνιες διακοπές στην λειτουργία, στοχεύοντας τις μικρές και μεσαίες ομάδες. Οι χρήστες είναι ευχαριστημένοι με το προϊόν, με εξαίρεση την έλλειψη προχωρημένων ενεργειών. Τέλος το Microsoft Project είναι σχεδιασμένο για μεγάλες επιχειρήσεις και οργανισμούς

προσφέροντας λειτουργίες όπως Gantt διαγράμματα, επιλογή για διαχείριση πόρων και προχωρημένους τρόπους προγραμματισμού. Η διεπαφή που οι χρήστες χρησιμοποιούν είναι κατανοήσιμη αν κάποιος αφιερώσει χρόνο, δεδομένου τον αριθμό των λειτουργιών που παρέχονται. Παρόλο τον μεγάλο αριθμό αυτών των λειτουργιών η ταχύτητα του προγράμματος δεν μειώνεται και καινούργιές προστίθεται σε συχνό ρυθμό. Όπως και το Asana οι νέοι χρήστες έχουν δυσκολία να εξοικειωθούν με το σύστημα. Η εφαρμογή που δημιουργήθηκε τείνει προς την Asana, διότι παρέχει ένα μεγάλο αριθμό από χαρακτηριστικά χωρίς ταυτόχρονα να αυξάνεται δραματικά η δυσκολία χρήσης. Ο σχεδιασμός του συστήματος είναι έτσι ώστε να παρέχει μεγάλα ποσοστά επιτυχίας στις ενέργειες των χρηστών χωρίς να μειώνεται η ταχύτητα.

3.4 Αξιολόγηση απόδοσης του συστήματος

Η αξιολόγηση της απόδοσης της συστήματος γίνεται μέσω μία σειρά δοκιμών σχεδιασμένες για την μετρικοποίηση της ανταπόκρισης, σταθερότητας και αποτελεσματικότητας υπό διαφορετικές συνθήκες. Για την συγκεκριμένη μέτρηση οι βασικές ρυθμίσεις που χρησιμοποιήθηκαν ήταν η κάθε δοκιμή να είναι δέκα λεπτά ακριβώς με τους συνολικούς χρήστες να ανέρχονται σε είκοσι. Δοκιμάστηκαν οι λειτουργίες Fixed, Ramp Up, Spike Up, Peak. Τα αποτελέσματα τα οποία μελετήθηκαν είναι ο χρόνος ανταπόκρισης του συστήματος, πόσα αιτήματα το σύστημα μπορούσε να εξυπηρετήσει κάθε δευτερόλεπτο και τον συνολικό αριθμό των αιτημάτων. Στην λειτουργία Fixed οι 20 χρήστες στέλνουν σταθερά αιτήματα κατά την διάρκεια της δοκιμής για να μετρηθεί η απόδοση του συστήματος υπό σταθερό φόρτο. Στην Ramp Up οι χρήστες αυξάνονται από έναν έως είκοσι κατά την διάρκεια των πρώτων πέντε λεπτών και για τα επόμενα πέντε λεπτά παραμένει στους είκοσι. Στην Spike Up υπάρχουν ξαφνικές αυξήσεις των χρηστών από μηδέν έως και είκοσι σε ένα μικρό διάστημα των τριάντα δευτερολέπτων. Τέλος στην Peak μεταβάλλεται ο φόρτος των χρηστών από υψηλές και μικρές ποσότητες. Για παράδειγμα είκοσι χρήστες στέλνουν αιτήματα για ένα λεπτό και έπειτα πέντε χρήστες στέλνουν αιτήματα για ένα επίσης λεπτό, με την διαδικασία αυτή να επαναλαμβάνεται μέχρι την λήξη της δοκιμής. Παρακάτω παρατίθεται ο *Πιν.2: Αποτελέσματα δοκιμών απόδοσης* που περιέχει τα αποτελέσματα των παραπάνω δοκιμών.

	Συνολικές Δοκιμές	Αριθμός εξυπηρέτησης ανά δευτερόλεπτο	Χρόνος ανταπόκρισης
Fixed	18000 req	31.13 req/sec	50ms
Ramp Up	13703 req	22.68 req/sec	45ms
Spike Up	2800 req	4.63 req/sec	35ms
Peak	10052 req	16.53 req/sec	40ms

Πιν.2: Αποτελέσματα δοκιμών απόδοσης

ΚΕΦ.4: Συμπεράσματα και μελλοντική εργασία

Ο σκοπός της πτυχιακής αυτής είναι ο σχεδιασμός και υλοποίηση Web εφαρμογής για τη διαχείριση φύλλων χρονοχρέωσης έργων. Μέσω της συνολικής διαδικασίας που περιεγράφηκε προηγουμένως, σημαντικά συμπεράσματα έχουν συνταχθεί. Η αξιολόγηση ευχρηστίας κατέδειξε ότι το σύστημα είναι φιλικό προς τον χρήστη και ο χρήστης είναι αρκετά ικανοποιημένος με τις λειτουργίες που παρέχονται. Το σύστημα επέδειξε ισχυρή απόδοση υπό διάφορες δοκιμές σε διαφορετικές μετρήσεις όπως fixed, ramp-up, spike-up και peak. Τα αποτελέσματα από αυτές τις δοκιμές επιβεβαίωσαν την ικανότητα του συστήματος να χειρίζεται αποτελεσματικά τα περισσότερα σενάρια χρηστών σε πραγματικό κόσμο. Ταυτόχρονα μελετήθηκαν και άλλες παρόμοιες εφαρμογές με τις οποίες έγινε σύγκριση. Η πτυχιακή παρατηρήθηκε ότι μπορούσε να ανταπεξέλθει στα ίδια περιβάλλοντα χρήσης όπως και οι άλλες εφαρμογές, χωρίς να μειώνονται σημαντικά οι λειτουργίες που προσφέρει. Το σύστημα μπορεί να έχει πετύχει τους βασικούς σκοπούς και λειτουργίες που απαιτήθηκαν αλλά υπάρχουν μελλοντικές ευκαιρίες για περαιτέρω προσθήκες. Στην δημιουργία ομάδας χρηστών για να μπορούν να διαχειρίζονται τα ίδια δεδομένα θα μπορούσε να προστεθεί ένας μηχανισμός ειδοποιήσεων για την αποδοχή ή όχι του αιτήματος αυτού. Οι ειδοποιήσεις που παρέχονται στους χρήστες θα μπορούσαν να είναι πιο αναλυτικές για την καλύτερη κατανόηση και καθοδήγηση τους. Τέλος θα μπορούσε να προστεθεί στην εφαρμογή η χρησιμοποίησης της τεχνητής νοημοσύνης για την αυτοματοποίηση των συνήθων καθηκόντων των χρηστών. Επίσης θα δοθεί και η δυνατότητα ενσωμάτωσης αλγορίθμων μηχανικής μάθησης για την ενεργοποίηση της προγνωστικής ανάλυσης, προσφέροντας στους χρήστες προληπτικές γνώσεις και συστάσεις.

ΒΙΒΛΙΟΓΡΑΦΙΑ

a/a	Είδος πηγής	Βιβλιογραφική αναφορά
1	Δικτυακός τόπος (Web site)	https://firebase.google.com/docs/auth
2	Δικτυακός τόπος (Web site)	https://auth0.com/
3		https://www.okta.com/products/authentication/
4	Δικτυακός τόπος (Web site)	https://docs.aws.amazon.com/cognito/latest/developerguide/authentication.html
5	Δικτυακός τόπος (Web site)	https://supabase.com/docs/guides/auth
6	Άρθρο	Jatana, N., Puri, S., Ahuja, M., Kathuria, I., & Gosain, D. (2012). A survey and comparison of relational and non-relational database. International Journal of Engineering Research & Technology, 1(6), 1-5.
7	Άρθρο	Gyorödi, C., Gyorödi, R., & Sotoc, R. (2015). A comparative study of relational and non-relational database models in a web-based application. International Journal of Advanced Computer Science and Applications, 6(11), 78-83.
8	Άρθρο	Kashyap, N. K., Pandey, B. K., Mandoria, H. L., & Kumar, A. (2016). A review of leading databases: Relational & non-relational database. i-Manager's Journal on Information Technology, 5(2), 34.
9	Άρθρο	Thakur, N., & Gupta, N. (2021). Relational and Non Relational Databases: A Review. Journal of University of Shanghai for Science and Technology, 23(8), 117-121.
10	Άρθρο	Kolonko, K. (2018). Performance comparison of the most popular relational and non-relational database management systems.
11	Δικτυακός τόπος (Web site)	https://www.mongodb.com/
12	Δικτυακός τόπος (Web site)	https://cassandra.apache.org/_/index.html
13	Δικτυακός τόπος (Web site)	https://redis.io/
14	Δικτυακός τόπος (Web site)	https://couchdb.apache.org/
15	Δικτυακός τόπος (Web site)	https://aws.amazon.com/dynamodb/
16	Άρθρο	Uunonen, S. (2023). Backend as a service in web development.

17	Δικτυακός τόπος (Web site)	https://nodejs.org/en/learn/getting-started/introduction-to-nodejs
18	Δικτυακός τόπος (Web site)	https://www.python.org/doc/
19	Δικτυακός τόπος (Web site)	https://docs.oracle.com/en/java/
20	Δικτυακός τόπος (Web site)	https://www.ruby-lang.org/en/documentation/
21	Δικτυακός τόπος (Web site)	https://go.dev/doc/
22	Δικτυακός τόπος (Web site)	https://spring.io/projects/spring-framework
23	Δικτυακός τόπος (Web site)	https://hibernate.org/
24	Δικτυακός τόπος (Web site)	https://rubyonrails.org/
25	Άρθρο	Bujnowski, G., & Smołka, J. (2020). Java and Kotlin code performance in selected web frameworks. Journal of Computer Sciences Institute, 16, 219–226. https://doi.org/10.35784/jcsi.2025
26	Άρθρο	Jelikj, I., & Gramatikov, S. (2023). Performance Evaluation of Back-end Web Application Programming Languages. Ss Cyril and Methodius University in Skopje, Faculty of Computer Science and Engineering, Republic of North Macedonia.
27	Άρθρο	Cherny, B. (2019). Programming TypeScript: making your JavaScript applications scale. O'Reilly Media.
28	Άρθρο	Jansen, R. H., Vane, V., & De Wolff, I. G. (2016). TypeScript: Modern JavaScript Development. Packt Publishing Ltd.
29	Άρθρο	Goldberg, J. (2022). Learning TypeScript. " O'Reilly Media, Inc.".
30	Άρθρο	Maharry, D. (2013). TypeScript revealed. Apress.
31	Δικτυακός τόπος (Web site)	https://docs.nestjs.com/
32	Δικτυακός τόπος (Web site)	https://foalts.org/docs/
33	Δικτυακός τόπος (Web site)	https://tsed.io/getting-started/
34	Δικτυακός τόπος (Web site)	https://koajs.com/#application
35	Δικτυακός τόπος (Web site)	https://hapi.dev/tutorials/?lang=en_US
36	Άρθρο	Pham, A. D. (2020). Developing back-end of a web application with NestJS framework: Case:

		Integrify Oy's student management system.
37	Δικτυακός τόπος (Web site)	https://react.dev/learn
38	Δικτυακός τόπος (Web site)	https://preactjs.com/guide/v10/getting-started/
39	Δικτυακός τόπος (Web site)	https://www.infernojs.org/docs/guides/installation
40	Δικτυακός τόπος (Web site)	https://svelte.dev/docs/introduction
41	Δικτυακός τόπος (Web site)	https://tailwindcss.com/docs/installation
42	Δικτυακός τόπος (Web site)	https://getbootstrap.com/docs/5.3/getting-started/introduction/
43	Δικτυακός τόπος (Web site)	https://bulma.io/documentation/
44	Δικτυακός τόπος (Web site)	https://v2.chakra-ui.com/getting-started
45	Δικτυακός τόπος (Web site)	https://mui.com/material-ui/getting-started/
46	Δικτυακός τόπος (Web site)	https://nextjs.org/docs
47	Άρθρο	Nielsen, J. (1992, June). Finding usability problems through heuristic evaluation. In Proceedings of the SIGCHI conference on Human factors in computing systems (pp. 373-380).