**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**

**Όνομα Πρώτου Καθηγητή (Επιβλέπων/ουσα) [Calibri, Bold,** **14]**

**Βαθμίδα, Τμήμα, Ίδρυμα**

**Όνομα Δεύτερου Καθηγητή**

**Βαθμίδα, Τμήμα, Ίδρυμα**

**Όνομα Τρίτου Καθηγητή**

**Βαθμίδα, Τμήμα, Ίδρυμα**

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, [Όνομα] και [Όνομα], 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………………………………………………σ.7

Περίληψη στα Αγγλικά……………………………………………….σ.8

Κατάλογος Εικόνων……………………………………………………σ.9

Κατάλογος Πινάκων……………………………………………………σ.10

Κατάλογος Σχημάτων…………………………………………………σ.11

Συντομογραφίες…………………………………………………………σ.12

Εισαγωγή…………………………………………………………………σ.13

Κεφ.1…………………………………………………………………………….

1.1………………………………………………………………………………….

1.1.1……………………………………………………………………………….

1.2…………………………………………………………………………………..

1.2.1……………………………………………………………………………….

Κεφ.2……………………………………………………………………………..

2.1……………………………………………………………………………………

2.1.1………………………………………………………………………………..

2.2…………………………………………………………………………………..

2.2.1……………………………………………………………………………….

Βιβλιογραφία………………………………………………………………..

Γλωσσάρι………………………………………………………………………

Παράρτημα Α’……………………………………………………………….

Παράρτημα Β’………………………………………………………………..

Ευρετήριο………………………………………………………………………

**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 όπως στην Τεχνητή νοημοσύνη έως και διάφορα εργαλεία. Τα οποία διευκολύνουν την ζωή του, τόσο την προσωπική, όπως για παράδειγμα λογισμικό: για την παρακολούθηση δαπανών και σχεδιασμών εξόδων, επεξεργασίας φωτογραφιών, δημιουργίας μουσικής κλπ., όσο και την επαγγελματική, πλατφόρμες για επικοινωνία, διαχείρισης πελατών, διαχείρισης έργων κλπ. Για κάθε ένα από τα παραπάνω έχουν δημιουργηθεί και ίσως διαθέτει 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 στην χρήση χωρίς να αλλοιώνονται οποιαδήποτε από τις λειτουργίες. Η εκτέλεση χαρακτηριστικών 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 διαχειριστή και βεβαιώνουν την έγκυρη εισαγωγή έργων. Θέματα όπως η ασφάλεια και η εγκυρότητα των δεδομένων επίσης έπαιξε σημαντικό ρόλο στον σχεδιασμό και στα στάδια του. Η εφαρμογή διασφαλίζει ότι ο πληροφορίες για τον χρήστη και για το κάθε έργο είναι προστατευμένα από μη εξουσιοδοτημένα άτομα ή προσπάθειες για παραβίαση δεδομένων στην βάση. Οι παραπάνω προδιαγραφές είναι αναγκαίες για την εμπιστοσύνη του χρήστη και με την προσαρμοστικότητα με βάση του 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 βασίζεται στον σχεδιασμό που βασίζεται στους χρήστες, ισχυρή ενσωμάτωση, και αδιάκοπη εστίαση στην ασφάλεια και τη 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**

Εικ.1. Γεωλογική απεικόνιση Νήσου Σαντορίνης………………………………σ.10

Εικ.2. Οι χώρες της Ευρωπαϊκής Ένωσης………………………………………….σ.34

**ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ**

Πίν.1: Εργαλεία που επιλέχθηκαν…………………………..σ.24

**ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ**

Σχ.1: Ρυθμοί ανάπτυξης Ιταλίας 1960-2010…………………………………………..σ.10

Σχ.2: Ρυθμοί ανάπτυξης Γαλλίας 1970-2010………………………………………….σ.26

**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 |

**ΕΙΣΑΓΩΓΗ**

Η διαχείριση φύλλων χρόνου παίζει 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 παλιοί 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 την σύγχρονη τεχνολογία, η εφαρμογή στοχεύει να προσφέρει ένα ισχυρό, φιλικό προς τους χρήστες περιβάλλον. Ο τελικός σκοπός είναι να μειωθούν αν όχι εξαλειφθούν τα ανθρώπινα σφάλματα, να μειωθεί η ανάγκη διαχειριστή και να προσφέρει στους υπεύθυνους των έργων την δυνατότητα να λαμβάνουν τις αποφάσεις τους με βάση στοιχεία σε πραγματικό χρόνο.

O σκοπός αυτής της πτυχιακής είναι να σχεδιαστεί και να αναπτύξει μια εφαρμογή για την 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. Κάνοντας το ένα εργαλείο για τον υπεύθυνο να παρατηρεί και να διαχειρίζονται τα έργα.

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 την 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 χρόνου για ένα συγκεκριμένο μήνα από τους χρήστες σε μορφή csv. Η αναμονή για την λήψη του συγκεκριμένο αρχείου μπορεί να είναι έως και 5 δευτερόλεπτα ανάλογα τον όγκο των δεδομένων.
   * Για την καλύτερη κατανόηση και διαχείριση των έργων από τους χρήστες υπάρχει ένα GNANTT διάγραμμα το οποίο παρουσιάζει το όνομα του έργου, την ημερομηνία έναρξης έως και την ημερομηνία λήξης μαζί με έναν οπτικό πίνακα. Στον οπτικό πίνακα ο χρήστης μπορεί να δει όλες τις ημερομηνίες και με χρωματικές αναπαραστάσεις την αρχή και λήξη του έργου και με αιώρηση πάνω από αυτές τις αναπαραστάσεις ο χρήστης μπορεί να δει τις ημέρες που απομένουν για την λήξη του έργου. Αυτό το διάγραμμα μπορεί να πάρει έως και 10 δευτερόλεπτα να φορτώσει ανάλογα τον όγκο των δεδομένων.
   * Ο χρήστης έχει την δυνατότητα να επεξεργαστεί τα δεδομένα κάθε είδος τύπου χωρίς να υπάρχει κίνδυνος να χαθούν δεδομένα ή να μην αποθηκευτούν οι αλλαγές.
   * Ο χρήστης αντί για τις ημερομηνίες λήξης όπου χρειάζεται θα εισάγουν έναν αριθμό διάρκειας. Επίσης ο χρήστης θα εισάγει τον μήνα που αρχίζει μία δέσμη εργασιών με μορφή ΜΧΧ, η οποία χρησιμοποιείται ευρέως από τις ερευνητικές ομάδες. Αυτές οι πληροφορίες μεταφράζονται σωστά όταν τις λαμβάνει η εφαρμογή από την βάση και τις παρουσιάζει σε κανονική ημερολογιακή ημερομηνία.
   * Οι χρήστες έχουν την 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 έργων όταν ο χρήστης επισκέπτεται την 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 την δυνατότητα να συνδεθεί με άλλους χρήστες για την δημιουργία μίας ομάδας, παρέχοντας έτσι την δυνατότητα, εισαγωγής, επεξεργασίας, 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 λειτουργείας 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 δεν 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 μέσα στην χρονιά.
   * 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 μεγάλες 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. H 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.

Εξηγήσαμε πλήρως τις λειτουργικές και μη λειτουργικές απαιτήσεις. Στο κεφάλαιο 1, αναλύεται η τρέχουσα κατάσταση, αναφέροντας εργαλεία, συστήματα, αλγορίθμους, μοντέλα τα οποία μπορεί να υλοποιούν τους παραπάνω στόχους ή βοηθούν στην υλοποίηση τους. Θα αναφερθούν οι λόγοι για την επιλογή των εργαλείων που χρησιμοποιεί η εφαρμογή, 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 την ίδια έμφαση και στις λειτουργικές και στις 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 χρησιμοποίηση εξωτερικών συστημάτων που έχουν ήδη δημιουργηθεί για αυτό τον σκοπό. Πριν την ανάλυση των δύο επιλογών να αναφερθεί ότι από την #1 λειτουργική απαίτηση οι χρήστες θέλουν να έχουν την δυνατότητα εισόδου με τον λογαριασμό τους στην google.

Η δημιουργία ενός τέτοιου εσωτερικού συστήματος εμπεριέχει την διαχείριση των εγγραφών των χρηστών, των στοιχείων σύνδεσης και την γενική διαχείριση του συστήματος. Με το να επιλεχθεί αυτό το σύστημα, το σύστημα έχει τον πλήρη έλεγχο πάνω στον τρόπο εξουσιοδότησης , επιτρέποντας έτσι την δημιουργία προσαρμοσμένων χαρακτηριστικών ανάλογα με τις ανάγκες της εφαρμογής. Επίσης δεν 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 εσωτερικά χρειάζεται ένα μεγάλο ανθρώπινο δυναμικό, αρκετό χρόνο για την υλοποίηση του και ειδικές γνώσεις πάνω στο θέμα. Όμως πέρα από την 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 στην χρήση ενός εξωτερικού συστήματος για την είσοδο των χρηστών στην ιστοσελίδα.

Τα πλεονεκτήματα και τα σκεπτικά, που αναφέρθηκαν παραπάνω, στην 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 λειτουργείες και υπηρεσίες με μικρές διαφορές μεταξύ τους. Προσφέρουν την δυνατότητα ο χρήστης να συνδεθεί με διάφορους τρόπους όπως με διεύθυνση ηλεκτρονικoύ 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 την σωστή χρήση τους, ώστε οι εφαρμογές που τις χρησιμοποιούν να τηρούν όλους τους κανόνες ασφαλείας. Παρέχουν την 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 την 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. Αντίθετα το Okta και το Amazon Cognito τείνουν να εξυπηρετούν μεγάλους οργανισμούς που χρειάζονται περίπλοκα συστήματα. To Supabase είναι λογισμικό ανοιχτού κώδικα και προσφέρει την 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 την 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: Έχουν έτοιμη βιβλιοθήκη η οποία μπορεί να χρησιμοποιηθεί συνδυάζοντας το κομμάτι του σχεδιασμού και της υλοποίησης και παρέχουν επίσης παράδειγμα κώδικα για το πως να το υλοποιήσει κάποιος μόνος του. Αυτή ιστοσελίδα ακολουθεί την δεύτερη επιλογή για να δημιουργήσει αντικείμενα τα οποία ταιριάζουν 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 την δυνατότητα επέκτασης οποιαδήποτε στιγμή. Κρατά στην μνήμη τον συνδεδεμένο χρήστη ώστε ακόμη και να γίνει ανανέωση της σελίδας δεν θα χρειαστεί να ξανά συνδεθεί. Αυτή την πληροφορία χρησιμοποιείται για την ορθή εμφάνιση ορισμένων στοιχείων για τους χρήστες ανάλογα και με τον ρόλο τους. Δίνονται ορισμένοι συναρτήσεις ακροατών για τις διάφορες καταστάσεις που μπορεί να βρίσκεται ένας χρήστης όπως αν συνδέθηκε, αν αποσυνδέθηκε, αν το τεκμήριο ανανεώθηκε για οποιοδήποτε λόγο. Τέλος ο χρήστης έχει εξαρχής την δυνατότητα δημιουργίας και διαγραφής λογαριασμού στο περιβάλλον firebase, το οποίο τους δίνει τον πλήρη έλεγχο στον λογαριασμό του. Στην περίπτωση της πτυχιακής όπως αναφέρεται και στην 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 συναρτήσεις όπως αποθήκευση, 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 διεκπεραιωτικότητας 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 πέφτει η απόδοση της μη σχεσιακής βάσης, όμως, είναι ότι έχουν μειωμένη συνοχή και επιτρέπου την 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. Στην επόμενη έρευνα [7, 9] που μελετήθηκε δοκιμάστηκαν συγκεκριμένα εντολές όπως η εισαγωγή δεδομένων, επεξεργασίας τους, διαγραφή τους και η αναζήτηση τους. Τα αποτελέσματα που διαπιστώθηκαν είναι παρόμοια με αυτά του [6], τονίζοντας ότι είχε σημαντική διαφορά όταν δοκιμάστηκαν οι τρεις πρώτες λειτουργίες αλλά στην αναζήτηση η 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 και από την εφαρμογή και τον τύπο δεδομένων που έχει. Η τελευταία έρευνα [10] επικεντρώνεται κυρίως στο γεγονός ότι ένας από τους σκοπούς των μη σχεσιακές βάσεων είναι η δημιουργία κώδικα στην γλώσσα κατά την προτίμηση του καθενός, δεδομένου ότι οι περισσότερες εφαρμογές τωρινά αναπτύσσονται με αντικειμενοστραφή τρόπο.

Με βάση τα παραπάνω αποτελέσματα η επιλογή μίας βάσης δεδομένων εξαρτάται κυρίως από τις προτιμήσεις του υπευθύνου ανάπτυξης του συστήματος και το ίδιο το σύστημα. Σε αυτήν την πτυχιακή δεν υπάρχουν πολλές συνδέσεις 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. Επίσης όλα μπορούν να ανταπεξέλθουν σε μεγάλους φόρτους δουλειάς. 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. H 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 τα κατηγορίες τους. Όλα στοχεύουν στο να προσφέρουν υψηλή διαθεσιμότητα, με εξαίρεση το MongoDB να προσπαθεί να κρατά μια ισορροπία ανάμεσα σε όλα τα χαρακτηριστικά. Τέλος το Amazon DynamoDB είναι το μόνο το οποίο δεν μπορεί να αυτό-φιλοξενηθεί.

Με βάση τις παραπάνω δυνατότητες του καθενός 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 πέρα από αυτά που αναφέρθηκαν, εργαλεία όπως Mongo Compass το οποίο είναι ένα διαχειριστικό με γραφιστικό περιβάλλον για την επεξεργασία, 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 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 δημιουργημένο στο μοντέλο που χρεώνει ανάλογα τους πόρους οι οποίοι 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 δεδομένου ότι 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, 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. Η Java [25, 26] εφαρμογές μπορούν να εκτελεστούν σε οποιαδήποτε συσκευή έχει εγκατεστημένο JVM. Έχουν υψηλή αποδοτικότητα και μπορούν να διαχειριστούν ανάγκες μεγάλων εταιριών. Υπάρχουν κατανοητοί σκελετοί όπως Spring [22] και Hibernate [23] που βοηθούν στην ανάπτυξη του κώδικα. Στην Ruby μπορεί να χρησιμοποιηθεί ο σκελετός Rails [24] ο οποίος αυξάνει την ταχύτητα ανάπτυξης δίνοντας προτεραιότητα στην ευκολία και όχι στην παραμετροποίηση. Σενάρια χρήσης που μπορεί να εμφανιστεί η Ruby είναι εφαρμογές διαδικτύου, 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 οι βιβλιοθήκες για την εξομάλυνση 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 ένας σκελετός. 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 αυτούς τους σκελετούς επεξεργάζονται τα αιτήματα στην σειρά, επιτρέποντας την επαναχρησιμοποίηση κώδικα. Προωθούν την λογική του σπονδυλωτού κώδικα όταν δημιουργείται μια εφαρμογή, οργανώνοντας τον κώδικα σε μικρότερα, ευκολά διαχειρίσιμα κομμάτια. Κάθε ένα από αυτά προσφέρει την δυνατότητα της δρομολόγησης. To NestJS και Ts.ED μέσα από διακοσμητές, tο FoalTS μέσα από διαχειριστές, ενώ ta 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 επεκτάσεις όπου οι προγραμματιστές μπορούν να εισάγουν άλλες βιβλιοθήκες ή και δικό τους προσαρμοσμένο κώδικα. Παρέχουν αναλυτικό και λεπτομερή παραδείγματα μαζί με μεγάλη υποστήριξη από την ανάλογη κοινότητα.

Το NestJS δίνει έμφαση δημιουργία ενός καθαρού, δομημένου κώδικα και έχει επίσημή υποστήριξη για διάφορες 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 εργαλείων 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 παραπάνω θέματος 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 από το παραπάνω, ο προγραμματιστής έχει τον πλήρη έλεγχο του κώδικα και το πώς τρέχει. Μπορεί να είναι πολύ αποδοτική η παραπάνω χρήση αν υλοποιηθεί σωστά. Όμως από την άλλη πλευρά η υλοποίηση αυτή θα αύξηση των χρόνο ανάπτυξης για να μπορέσει να γίνει σωστά, διότι προστίθεται στον φόρτο και ο χειρισμός του DOM, δυσκολεύει η συντήρηση του προγράμματος γιατί αυξάνεται και 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 επαναχρησιμοποιήσιμων συνιστωσών, 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 μειώνονται 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. To Inferno όμως προσθέτει πολυπλοκότητα σε κέρδος την ταχύτητα. Τέλος το Svelte είναι περισσότερο ένας σκελετός παρά μια βιβλιοθήκη, ο οποίος 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. Δημιουργία χειριστών γεγονότων 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 μεταβάλλονται.

H 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, εύκολο 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, επιτρέποντας την χρησιμοποίηση των αντικείμενων που χρειάζονται και αυξημένη ανταπόκριση σε όλες τις πλατφόρμες. Λόγω της ελαφριάς του μορφή δεν έχει μεγάλο εύρος από έτοιμες λειτουργίες. Επόμενο στην λίστα επιλογών είναι το Chakra UI εύκολη 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 τονωθεί ότι σαν 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 σε συνδυασμό ένα προσεγμένο περιβάλλον αλληλεπίδρασης για τους χρήστες.

Επιλέχθηκε να χρησιμοποιηθεί ένας σκελετός, 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, διάφορα σενάρια, αυξάνοντας έτσι τον ολικό χρόνο ανταπόκρισης του Frontend. Υποστηρίζει επίσης την γραφή σε Typescript όπως χρησιμοποιείται και στο Backend αυξάνοντας την ενοποίηση ολόκληρου του συστήματος.

**1.5 Πίνακας Εργαλείων που επιλέχθηκαν**

|  |  |
| --- | --- |
| **Κατηγορίες** | **Εργαλεία** |
| Backend | Firebase |
|  | MongoDB |
|  | JavaScript/Typescript |
|  | NestJS |
| Frontend | React |
|  | Tailwind CSS |
|  | NextJS |

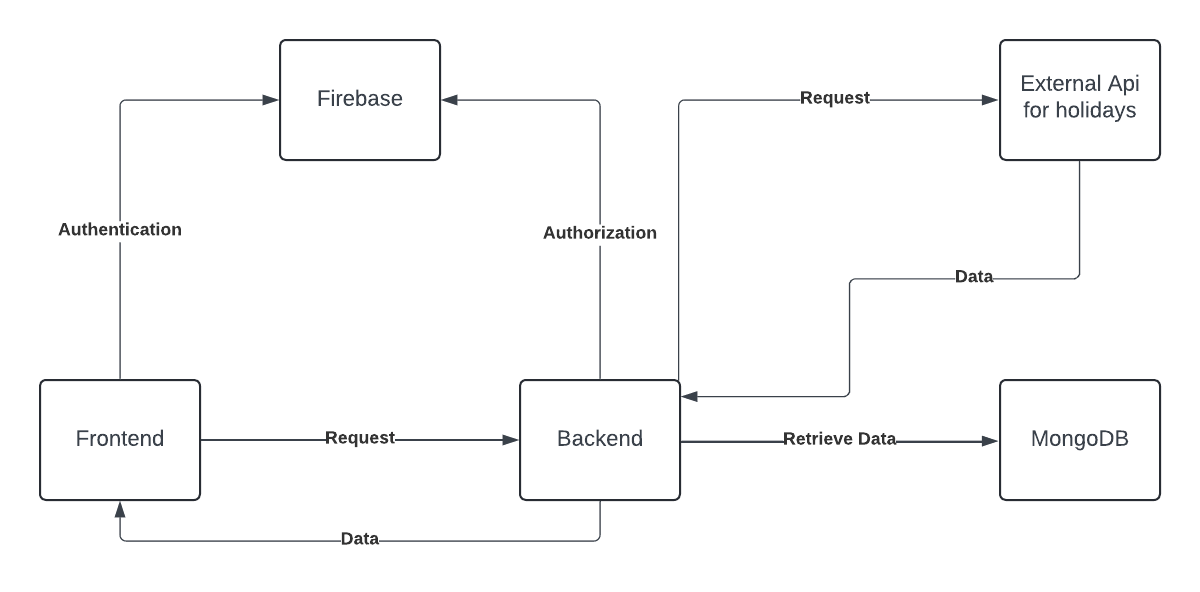
**ΚΕΦ.2: Σχεδίαση και υλοποίηση συστήματος/εφαρμογής**

**2.1 Αρχιτεκτονική συστήματος**

Η εξήγηση 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 μπορέσουν να υλοποιήσουν το τελικό αποτέλεσμα. Τα αντικείμενα αυτά μπορούν να διαχωριστούν σε τέσσερις μεγάλες 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 εμφανίζονται 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 ή δεδομένα που επιθυμεί, τότε επιστρέφεται το αποτελέσματα επιστρέφεται 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 ένα διάγραμμα το οποίο αναφέρεται στην αρχιτεκτονική του συστήματος, δείχνοντας την υψηλή λογική του.

****

(Σχήμα 1)

**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 GNAΝTΤ 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. 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 εργασίας. 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 οι διαδικασίες σε μορφή διαγραμμάτων.

**A screenshot of a project

Description automatically generated**

(Σχήμα 2)

A screenshot of a computer screen

Description automatically generated(Σχήμα 3)

**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 εμπεριέχεται 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 εργασίας, 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 ένα διάγραμμα με αναλυτικές πληροφορίες για τις οντότητες.

A diagram of a computer

Description automatically generated

(Σχήμα 4)

**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 τ 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 χρήστης ακολουθεί τα βήματα για πιστοποίηση. Η Google επιστρέφει στην εφαρμογή ένα μοναδικό σύμβολο το οποίο στέλνεται στην Firebase για να γίνει ταυτοποίηση και να παρθούν οι πληροφορίες του χρήστη. Αν αυτό το σύμβουλο είναι έγκυρο τότε από τις πληροφορίες, ελέγχεται ο ρόλος του χρήστη για την εμφανίσει του σωστού μενού. Να σημειωθεί ότι το μοναδικό σύμβουλο είναι 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 backend σύστημα. Παρακάτω παρατίθεται ένα διάγραμμα που επεξηγεί το παραπάνω σύστημα.

**A screenshot of a cell phone

Description automatically generated**

**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, 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 μικρότερα, 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. Για τυχόν σφάλματα τα οποία μπορεί να εμφανιστούν στην εφαρμογή, έχουν δημιουργηθεί συναρτήσεις για την σωστή διαχείριση τους και την αποφυγή αποτυχίας της εφαρμογής.

**ΚΕΦ.3: Αξιολόγηση συστήματος/εφαρμογής**

**3.1 Σκοποί της αξιολόγησης συστήματος**

Η αξιολόγηση της αποδοτικότητας, επίδοσης και την γενική επιρροή 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 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. Πόση πληροφορία μπορεί να διαχειριστεί ταυτόχρονα η εφαρμογή και πόσους πόρους καταναλώνει για την σωστή λειτουργία της. Δοκιμές σχετικά 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 απόδοσης του συστήματος**

**ΒΙΒΛΙΟΓΡΑΦΙΑ**

|  |  |
| --- | --- |
| **1** | Firebase Authentication Documentation <https://firebase.google.com/docs/auth> |
| **2** | Auth0 Documentation  <https://auth0.com/> |
| **3** | Okta Documentation  <https://www.okta.com/products/authentication/> |
| **4** | Amazon Cognito Documentation  <https://docs.aws.amazon.com/cognito/latest/developerguide/authentication.html> |
| **5** | Supabase Auth Documentation  <https://supabase.com/docs/guides/auth> |
| **6** | 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** | 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** | 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** | 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** | Kolonko, K. (2018). Performance comparison of the most popular relational and non-relational database management systems. |
| **11** | MongoDB Documentation  <https://www.mongodb.com/> |
| **12** | Cassandra Documentation  <https://cassandra.apache.org/_/index.html> |
| **13** | Redis Documentation  <https://redis.io/> |
| **14** | CouchDB  <https://couchdb.apache.org/> |
| **15** | Amazon DynamoDB  <https://aws.amazon.com/dynamodb/> |
| **16** | Uunonen, S. (2023). Backend as a service in web development. |
| **17** | Nodejs Documentation  <https://nodejs.org/en/learn/getting-started/introduction-to-nodejs> |  |
| **18** | Python Documentation  <https://www.python.org/doc/> |
| **19** | Java Documentation  <https://docs.oracle.com/en/java/> |
| **20** | Ruby Documentation  <https://www.ruby-lang.org/en/documentation/> |
| **21** | Go Lang Documentation  <https://go.dev/doc/> |
| **22** | Spring Framework  <https://spring.io/projects/spring-framework> |
| **23** | Hibernate Framework  <https://hibernate.org/> |
| **24** | Rails Framework  <https://rubyonrails.org/> |
| **25** | 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** | 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** | Cherny, B. (2019). Programming TypeScript: making your JavaScript applications scale. O'Reilly Media. |
| **28** | Jansen, R. H., Vane, V., & De Wolff, I. G. (2016). TypeScript: Modern JavaScript Development. Packt Publishing Ltd. |
| **29** | Goldberg, J. (2022). Learning TypeScript. " O'Reilly Media, Inc.". |
| **30** | Maharry, D. (2013). TypeScript revealed. Apress. |
| **31** | NestJS Documentation  <https://docs.nestjs.com/> |
| **32** | FoalTS Documentation  <https://foalts.org/docs/> |
| **33** | Ts.ED Documentation  <https://tsed.io/getting-started/> |
| **34** | Koa Documentation  <https://koajs.com/#application> |
| **35** | Hapi.dev Documentation  <https://hapi.dev/tutorials/?lang=en_US> |
| **36** | Pham, A. D. (2020). Developing back-end of a web application with NestJS framework: Case: Integrify Oy’s student management system. |
| **37** | React Documentation  <https://react.dev/learn> |
| **38** | Preact Documentation  <https://preactjs.com/guide/v10/getting-started/> |
| **39** | Inferno Documentation  <https://www.infernojs.org/docs/guides/installation> |
| **40** | Svelte Documentation  <https://svelte.dev/docs/introduction> |
| **41** | Tailwind CSS Documentation  <https://tailwindcss.com/docs/installation> |
| **42** | Bootstrap Documentation  <https://getbootstrap.com/docs/5.3/getting-started/introduction/> |
| **43** | Bulma CSS  <https://bulma.io/documentation/> |
| **44** | Chakra UI  <https://v2.chakra-ui.com/getting-started> |
| **45** | Material UI  <https://mui.com/material-ui/getting-started/> |
| **46** | NextJS  <https://nextjs.org/docs> |