ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ

ΣΧΟΛΗ: ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ: ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Διεύθυνση: Μ. Αλεξάνδρου 1, Τηλ.:2610 - 369236, fax: 2610-369193

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΘΕΜΑ

‘’Έρευνα, σχεδιασμός, ανάπτυξη και λειτουργία μιας εφαρμογής διαχείρισης εργασιών με χρήση προοδευτικών εφαρμογών ιστού (PWA) με χρήση Vue js’’

ΟΝΟΜΑΤΑ ΦΟΙΤΗΤΩΝ

Ευάγγελος Δριβάκος

2983

ΟΝΟΜΑ ΕΠΙΒΛΕΠΟΝΤΑ ΚΑΘΗΓΗΤΗ

Γεώργιος Κων. Ασημακόπουλος

Περίληψη

Ο τίτλος της παρούσας πτυχιακής εργασίας είναι "Έρευνα, σχεδιασμός, ανάπτυξη και λειτουργία μιας εφαρμογής διαχείρισης εργασιών με χρήση προοδευτικών εφαρμογών ιστού (PWA) με χρήση Vue.js".

Η εργασία αυτή επικεντρώνεται στην ανάπτυξη μιας εφαρμογής διαχείρισης εργασιών που χρησιμοποιεί τεχνολογίες web, κυρίως το Vue.js, καθώς και την προοδευτική ιδεολογία των εφαρμογών ιστού (PWA).

Ο πρωταρχικός στόχος της εφαρμογής είναι να επιτρέπει στους χρήστες τη δημιουργία, επεξεργασία και διαγραφή εργασιών, με την δυνατότητα να παρακολουθούν το χρόνο που δαπανάται σε κάθε εργασία. Επιπλέον, οι χρήστες μπορούν να ενημερώνουν την κατάσταση των εργασιών τους, αναφέροντας εάν είναι ολοκληρωμένες ή όχι.

Η εφαρμογή έχει σχεδιαστεί ώστε να λειτουργεί σε διάφορες συσκευές και πλατφόρμες, χάρη στην υποστήριξη της PWA. Η εργασία αυτή περιλαμβάνει την πλήρη διαδικασία ανάπτυξης, από την έρευνα και σχεδιασμό, μέχρι την ανάπτυξη και λειτουργία της εφαρμογής.

Λέξεις κλειδιά:

Ανάπτυξη λογισμικού, Εφαρμογές διαχείρισης εργασιών, Vue.js, Προοδευτικές εφαρμογές ιστού (PWA), Σχεδιασμός εφαρμογών, Ανάπτυξη front-end, Ανάπτυξη back-end, JavaScript, Εξυπηρετητής Node.js, Αποθήκευση δεδομένων, Αυθεντικοποίηση χρηστών, offline υποστήριξη, Κατάσταση εργασιών, Επεξεργασία εργασιών, Καταγραφή χρόνου εργασίας.

Περιεχόμενα

Πίνακας Περιεχομένων :

Πίνακας εικόνων

Εισαγωγή

Στο σύγχρονο, γρήγορα εναλλασσόμενο επαγγελματικό περιβάλλον, η αποτελεσματική διαχείριση των εργασιών είναι ζωτικής σημασίας. Με την αυξανόμενη πολυπλοκότητα των έργων, οι επαγγελματίες αντιμετωπίζουν δυσκολίες στη διαχείριση των εργασιών τους, την παρακολούθηση της προόδου τους και την εκτίμηση του χρόνου που απαιτείται για την ολοκλήρωση των διαφόρων εργασιών.

Επιπλέον, υπάρχει μια αυξανόμενη ανάγκη για μετατρεψιμότητα και προσβασιμότητα, καθώς οι επαγγελματίες χρειάζονται να έχουν πρόσβαση στις εργασίες τους από διάφορες συσκευές και τοποθεσίες. Ωστόσο, πολλές από τις υφιστάμενες λύσεις για τη διαχείριση εργασιών είτε δεν προσφέρουν την απαραίτητη δυνατότητα προσαρμογής και ευελιξίας, είτε είναι υπερβολικά πολύπλοκες για τον μέσο χρήστη.

Γι' αυτό το λόγο, είναι απαραίτητη η ανάπτυξη μιας νέας εφαρμογής διαχείρισης εργασιών που θα προσφέρει ευκολία χρήσης, δυνατότητες προσαρμογής και μετατρεψιμότητας. Η εφαρμογή αυτή θα πρέπει να είναι σχεδιασμένη για να λειτουργεί ως Προοδευτική Εφαρμογή Ιστού (PWA), ώστε να είναι προσβάσιμη από οποιαδήποτε συσκευή που διαθέτει ίντερνετ, να προσφέρει υποστήριξη για πολλαπλές συσκευές και να διατηρεί την υψηλή απόδοση και την ευχρηστία.

1. Σκοπός της εργασίας

Ο κύριος σκοπός της παρούσας εργασίας είναι η σχεδίαση, η ανάπτυξη και η λειτουργία μιας εφαρμογής διαχείρισης εργασιών ως Προοδευτική Εφαρμογή Ιστού (PWA) χρησιμοποιώντας το JavaScript Framework, Vue.js. Η εφαρμογή θα πρέπει να παρέχει τη δυνατότητα διαχείρισης εργασιών, καταγραφής χρόνου, επεξεργασίας και ενημέρωσης κατάστασης εργασιών, εξασφαλίζοντας ταυτόχρονα υψηλό επίπεδο μεταφερσιμότητας και προσβασιμότητας.

Στόχοι της Εργασίας

Έρευνα: Να διερευνηθούν οι απαιτήσεις της διαχείρισης εργασιών και οι δυνατότητες που προσφέρει η τεχνολογία της Vue.js και το (PWA).

Σχεδιασμός: Να σχεδιαστεί η αρχιτεκτονική και η διεπαφή χρήστη της εφαρμογής, παρέχοντας ένα αποτελεσματικό και ευχάριστο περιβάλλον για τους χρήστες.

Ανάπτυξη: Να αναπτυχθεί η εφαρμογή χρησιμοποιώντας τις τεχνικές και τις βέλτιστες πρακτικές της Vue.js και των PWA.

Αξιολόγηση: Να αξιολογηθεί η εφαρμογή σε πραγματικές συνθήκες για να διαπιστωθεί η αποτελεσματικότητα και η αποδοτικότητα της στη διαχείριση εργασιών.

Διάδοση: Να προετοιμαστεί και να παρουσιαστεί το προϊόν για την διάδοση στην κοινότητα Vue.js και PWA.

1.1 Περιγραφή του υπόλοιπου εγγράφου

Κεφάλαιο 2: Επισκόπηση της σχετικής βιβλιογραφίας

Στο δεύτερο κεφάλαιο θα παρουσιάσουμε τη θεωρία και τις βασικές αρχές που σχετίζονται με τη διαχείριση εργασιών, τις Προοδευτικές Εφαρμογές Ιστού (PWA), και τη Vue.js. Θα παρουσιάσουμε επίσης τις βέλτιστες πρακτικές για την ανάπτυξη εφαρμογών PWA χρησιμοποιώντας τη Vue.js.

Κεφάλαιο 3: Μεθοδολογία

Το τρίτο κεφάλαιο θα περιγράψει τη μεθοδολογία που ακολουθήσαμε για την σχεδίαση, ανάπτυξη και manual testing της εφαρμογής. Θα περιλαμβάνει την ανάλυση απαιτήσεων, τη σχεδίαση, την ανάπτυξη και τις διαδικασίες testing.

Κεφάλαιο 4: Υλοποίηση

Στο τέταρτο κεφάλαιο, θα παρουσιάσουμε λεπτομερώς την υλοποίηση της εφαρμογής, περιλαμβάνοντας τον κώδικα, τη δομή των αρχείων, τις βάσεις δεδομένων και τα σημαντικά σημεία της διαδικασίας ανάπτυξης.

Κεφάλαιο 5: Αξιολόγηση

Στο πέμπτο κεφάλαιο, θα αναλύσουμε την απόδοση και την ευχρηστία της εφαρμογής, μέσα από διάφορες διαδικασίες δοκιμών και αξιολογήσεων. Επιπλέον, θα παρουσιάσουμε τα αποτελέσματα από τη χρήση της εφαρμογής σε πραγματικές συνθήκες.

Κεφάλαιο 6: Συμπεράσματα και Μελλοντική Έρευνα

Στο τελευταίο κεφάλαιο, θα συνοψίσουμε τα κύρια συμπεράσματα της εργασίας, θα αναφέρουμε τυχόν περιορισμούς και θα προτείνουμε μελλοντικές κατευθύνσεις για επιπλέον έρευνα και ανάπτυξη της εφαρμογής.

Επεξήγηση των βασικών έννοιών: PWA, Vue.js, Task

PWA (Progressive Web Applications)

Οι Προοδευτικές Εφαρμογές Ιστού, γνωστές ως PWA, είναι μια τεχνολογία που επιτρέπει στις εφαρμογές ιστού να λειτουργούν περισσότερο σαν εγκατεστημένες εφαρμογές. Αυτό σημαίνει ότι μπορούν να λειτουργήσουν χωρίς σύνδεση στο διαδίκτυο, να λάβουν ειδοποιήσεις push και να αποθηκεύσουν σημαντικά δεδομένα στη συσκευή του χρήστη. Η τεχνολογία PWA χρησιμοποιεί αρχεία manifest και service workers για να πετύχει αυτήν την απόδοση.

Vue.js

Το Vue.js είναι ένα πλαίσιο JavaScript για τη δημιουργία διαδραστικών εφαρμογών ιστού. Έχει σχεδιαστεί για να είναι ευέλικτο και εύκολο στη χρήση, με την ικανότητα να σχεδιάζει εξατομικευμένες λύσεις για κάθε πρόβλημα. Το Vue.js χρησιμοποιεί ένα σύστημα επαναλαμβανόμενων στοιχείων και επαναχρησιμοποιήσιμων συστατικών, επιτρέποντας στους προγραμματιστές να επικεντρώνονται στη δημιουργία υψηλής ποιότητας κώδικα.

Task (Εργασία)

Στο πλαίσιο της εφαρμογής διαχείρισης εργασιών, ένα task ή εργασία αναφέρεται σε μια μονάδα εργασίας που πρέπει να ολοκληρωθεί. Μια εργασία μπορεί να περιλαμβάνει πληροφορίες όπως ο τίτλος, η περιγραφή, η προθεσμία, ο υπεύθυνος της εργασίας και η κατάσταση (π.χ., εκκρεμεί, είναι σε εξέλιξη, ολοκληρώθηκε). Οι εφαρμογές διαχείρισης εργασιών επιτρέπουν στους χρήστες να δημιουργούν, επεξεργάζονται, διαγράφουν και να παρακολουθούν εργασίες.

2. Επισκόπηση της σχετικής βιβλιογραφίας **(PWA)** και επεξήγησή όρων

Η τεχνολογία των PWA έχει εξελιχθεί σημαντικά τα τελευταία χρόνια.
Σύμφωνα με την εργασία του Οργανισμού προοδευτικών Εφαρμογών Ιστού (2020), τα PWA παρέχουν μια καλή εναλλακτική λύση στις εφαρμογές για smartphones, λόγω της δυνατότητας λειτουργίας χωρίς σύνδεση και της αυτόνομης ενημέρωσης.

Η υποστήριξη Progressive Web Apps (PWA) είναι ενσωματωμένη στο Vue.js και μπορείτε εύκολα να δημιουργήσετε PWA χρησιμοποιώντας το Vue CLI, το βασικό εργαλείο για τη δημιουργία Vue.js εφαρμογών. Τα PWA είναι web εφαρμογές που προσφέρουν παρόμοιες εμπειρίες χρήστη με εφαρμογές για κινητά, συμπεριλαμβανομένης της δυνατότητας εκκίνησης από την αρχική οθόνη, offline λειτουργία, και ενημερώσεις.

Εδώ είναι μερικά βήματα για τη δημιουργία του PWA χρησιμοποιώντας το Vue.js:

1. Εγκατάσταση του Vue CLI: Αν δεν έχετε το Vue CLI εγκατεστημένο, μπορείτε να το εγκαταστήσετε ως εξής:

   npm install -g @vue/cli

2. Δημιουργία νέου Vue Project: Δημιουργήστε ένα νέο Vue project χρησιμοποιώντας το Vue CLI. Κατά τη δημιουργία, θα σας ρωτήσει εάν θέλετε να προσθέσετε υποστήριξη PWA. Επιλέξτε "Yes".

   vue create my-pwa-app

3. Διαμόρφωση του PWA: Μετά τη δημιουργία του project, μεταβείτε στον φάκελο του

project και διαμορφώστε το αρχείο vue.config.js για να προσαρμόσετε τις ρυθμίσεις του PWA.

Παράδειγμα:

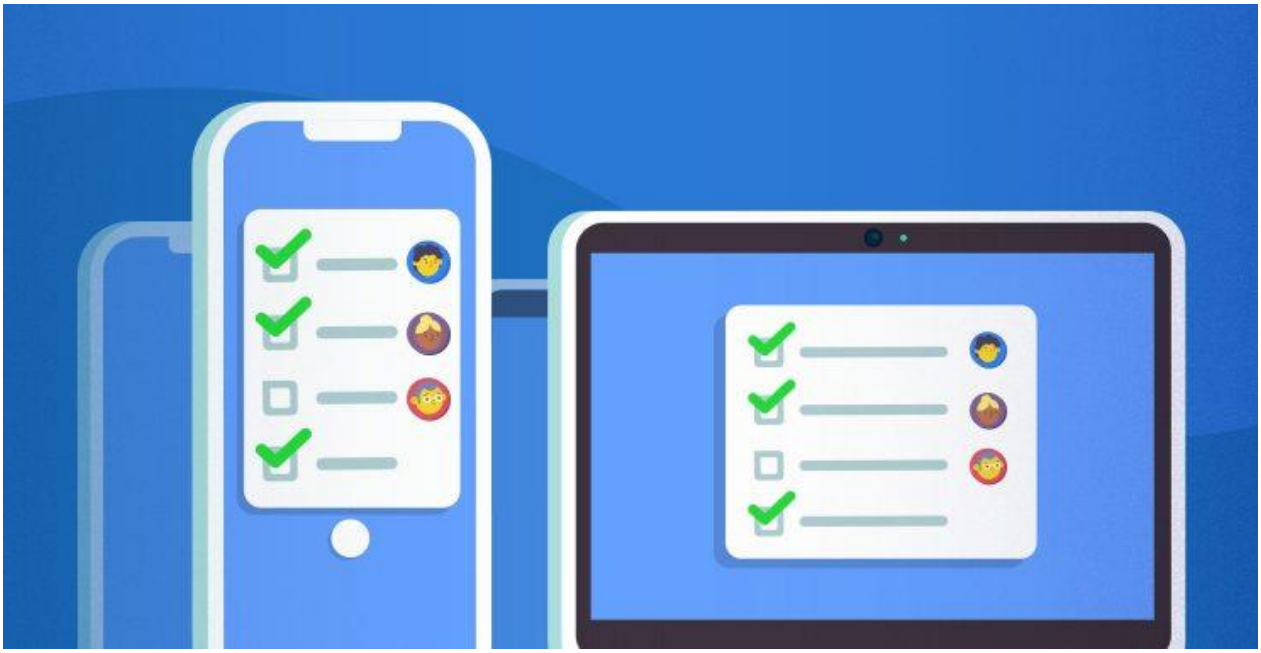
```
// vue.config.js

module.exports = {
  pwa: {
    name: 'My PWA App',
    themeColor: '#4DBA87',
    msTileColor: '#000000',
    appleMobileWebAppCapable: 'yes',
    appleMobileWebAppStatusBarStyle: 'default',
  },
};
```

4. Δημιουργία Service Worker: Ο Vue CLI θα δημιουργήσει αυτόματα ένα Service Worker για την εφαρμογή σας, το οποίο θα μπορούσε να χρησιμοποιηθεί για την διαχείριση της offline λειτουργίας και την ενημέρωση της εφαρμογής.

5. Σχεδιασμός της Εφαρμογής: Σχεδίαση της εφαρμογής ώστε να είναι φιλική προς PWA, συμπεριλαμβάνοντας την δυνατότητα εγκατάστασης από την αρχική οθόνη των κινητών.

Τα PWA προσφέρουν πολλά οφέλη, όπως η δυνατότητα λειτουργίας offline, η εγκατάσταση από την αρχική οθόνη, η αυξημένη απόδοση και η ευκολία στη διανομή. Ωστόσο, πρέπει να ληφθούν υπόψη κάποιες προκλήσεις, όπως ο διαχωρισμός λειτουργίας online και offline και η διαχείριση της ασφάλειας σε offline καταστάσεις.

Εφαρμογές Διαχείρισης Εργασιών



Όσον αφορά τις εφαρμογές διαχείρισης εργασιών, υπάρχει μια πληθώρα ερευνών που αναδεικνύουν την σημασία της οργάνωσης και της διαχείρισης των εργασιών.
Για παράδειγμα, η έρευνα του Robert Pellerin. (2013) " Project Management Software Utilization and Project Performance" επισημαίνει πόσο σημαντική είναι η αποτελεσματική διαχείριση των εργασιών για την επίτευξη των στόχων ενός έργου.
Ωστόσο αντλήθηκε έμπνευση και από υπάρχων εφαρμογές όπως asana τόσο στο σχεδιαστικό κομμάτι όσο και στο κομμάτι της υλοποίησης.

Τα εργαλεία διαχείρισης εργασιών όπως το Asana είναι απαραίτητα τόσο σε προσωπικό όσο και σε επαγγελματικό επίπεδο για πολλούς λόγους:

1. Οργάνωση: Σας βοηθούν να οργανώσετε τα καθημερινά καθήκοντά σας, τα έργα, τις εργασίες και τις προθεσμίες. Μπορείτε να δημιουργήσετε λίστες εργασιών και να τις διαχωρίσετε σε κατηγορίες, κάτι που καθιστά πολύ ευκολότερη τη διαχείριση του χρόνου σας.

2. Προτεραιότητες: Σας επιτρέπουν να ορίσετε προτεραιότητες στις εργασίες σας. Αυτό σας βοηθά να επικεντρωθείτε στα πιο σημαντικά και επείγοντα καθήκοντα.

3. Συνεργασία: Σε επαγγελματικό επίπεδο, τα εργαλεία διαχείρισης εργασιών σάς επιτρέπουν να συνεργάζεστε με άλλα άτομα. Μπορείτε να μοιραστείτε εργασίες, να αναθέσετε καθήκοντα, να παρακολουθείτε την πρόοδο των έργων και να επικοινωνείτε με τους συνεργάτες σας.

4. Παρακολούθηση προόδου: Σας δίνουν τη δυνατότητα να παρακολουθήσετε την πρόοδο

των έργων σας. Μπορείτε να δείτε πόσο έχετε προχωρήσει, ποιες εργασίες έχουν ολοκληρωθεί και ποιες παραμένουν ανολοκλήρωτες.

5. Διαχείριση Χρόνου: Σας βοηθούν να διαχειρίζεστε τον χρόνο σας αποτελεσματικά. Μπορείτε να προγραμματίσετε εργασίες, να ορίσετε προθεσμίες, και να παρακολουθείτε τα deadlines.

6. Αυξάνουν την Παραγωγικότητα: Σας βοηθούν να γίνετε πιο παραγωγικοί, καθώς σας επιτρέπουν να εστιάζετε στο να ολοκληρώνετε εργασίες χωρίς να αποσπάστε από τις συνεχείς διακοπές.

7. Ελαχιστοποίηση του Λάθους: Με τη χρήση των εργαλείων διαχείρισης εργασιών, μπορείτε να μειώσετε τον κίνδυνο λησμονιάς ή αμελητικού διαχειρισμού των εργασιών σας.

Στην ουσία, τα εργαλεία διαχείρισης εργασιών βοηθούν να διοικήσετε αποτελεσματικά τον χρόνο, να διαχειριστείτε τις εργασίες σας και να είστε πιο οργανωμένοι, είτε στην καθημερινή ζωή σας είτε στο επαγγελματικό περιβάλλον.

Όπως αναφέρει και η έρευνα "Project Management Software Utilization and Project Performance" που δημοσιεύτηκε στο Association for Computing Machinery, εξετάζει τη χρήση εργαλείων διαχείρισης εργασιών και τον τρόπο με τον οποίο αυτά επηρεάζουν την απόδοση σε έργα και εργασίες. Παρακάτω παρέχω μια αναλυτική περίληψη των βασικών σημείων της μελέτης:

Εργαλεία Διαχείρισης Εργασιών (Task Management Tools): Τα εργαλεία διαχείρισης εργασιών είναι φυσικά ή ψηφιακά εργαλεία που διευκολύνουν τη συλλογή, παρακολούθηση και διαχείριση πληροφοριών που σχετίζονται με εργασίες. Αυτά τα εργαλεία χρησιμοποιούνται τόσο στην επαγγελματική όσο και στην προσωπική ζωή για τη διαχείριση εργασιών. Υπάρχουν εφαρμογές για την ατομική χρήση (όπως Google Tasks, Todoist, Microsoft ToDo) και εφαρμογές για τη διαχείριση εργασιών σε ομάδες ή οργανισμούς (όπως Asana, Trello, Jira).

Ανάπτυξη της Αγοράς: Οι ερευνητές παρατηρούν ότι η αγορά των εργαλείων διαχείρισης εργασιών αυξάνεται και αναμένεται να υπερβεί τα 4 δισεκατομμύρια δολάρια μέχρι το 2023. Αυτό υποδεικνύει τη σημασία αυτής της αγοράς για την έρευνα, την καινοτομία και την εξέλιξη.

Χρήση ως Μέσοι Βοήθειας στη Μνήμη: Τα εργαλεία διαχείρισης εργασιών χρησιμοποιούνται για να καταγράφουν γρήγορα σκέψεις και εργασίες καθώς αυτές προκύπτουν. Για παράδειγμα, κάποιος μπορεί να σταματήσει προσωρινά το μαγείρεμα για να καταγράψει μια νέα ιδέα ή εργασία.

Αλλαγές στον Τρόπο Εργασίας λόγω της Πανδημίας: Η μελέτη επικεντρώνεται στον τρόπο με τον οποίο οι αλλαγές στον τρόπο εργασίας λόγω της πανδημίας έχουν επηρεάσει τις στρατηγικές διαχείρισης εργασιών.

Σημαντικά Ευρήματα: Οι εργαζόμενοι χρησιμοποιούν τα εργαλεία διαχείρισης εργασιών ανάλογα με τους στόχους και τις προτιμήσεις τους στον τομέα της ενοποίησης ή απομόνωσης

της εργασίας και της προσωπικής ζωής.

Προκλήσεις και Ευκαιρίες: Οι ερευνητές εξάγουν κύρια ερωτήματα για τον σχεδιασμό των μελλοντικών εργαλείων διαχείρισης εργασιών, λαμβάνοντας υπόψη τις αλλαγές στον τρόπο εργασίας. Συζητούν τις ευκαιρίες και τις προκλήσεις που υπάρχουν στον τομέα της διαχείρισης εργασιών καθώς η εργασία εξελίσσεται τόσο σε ατομικό όσο και σε οργανισμικό επίπεδο.

Αυτή η μελέτη εξετάζει τον τρόπο με τον οποίο οι εργαζόμενοι χρησιμοποιούν εργαλεία διαχείρισης εργασιών και πώς αυτά επηρεάζουν τη διαχείριση των εργασιών τους στον σύγχρονο κόσμο της εργασίας.

Ακόμα εξετάζει την πρακτική της διαχείρισης εργασιών μέσω των εργαλείων διαχείρισης εργασιών, όπου αυτά χρησιμοποιούνται ως μέσα υπενθύμισης και οργάνωσης της εργασίας. Η πρακτική της διαχείρισης εργασιών προσαρμόζεται στο προσωπικό πλαίσιο και είναι υψίστημα περιβαλλοντική.

Οι μελέτες επικεντρώνονται στη χρήση διαφόρων τύπων εργαλείων διαχείρισης εργασιών, όπως προσωπικές λίστες εργασιών, ηλεκτρονική αλληλογραφία (email) και εξυπνους βοηθούς. Αυτές οι μελέτες εξετάζουν πώς οι άνθρωποι διαχειρίζονται τις εργασίες τους και πώς η διαχείριση αυτή είναι συγκεκριμένη στο πλαίσιο τους. Τέλος, διερευνούν τον τρόπο με τον οποίο οι τεχνολογικές εξελίξεις επηρεάζουν τη διαχείριση εργασιών και την αντίληψη της παραγωγικότητας από τη σκοπιά του ατόμου.

Οι εφαρμογές διαχείρισης εργασιών (task management apps) αποτελούν σημαντικό εργαλείο για τη διαχείριση των εργασιών και των καθημερινών εργασιών τόσο σε προσωπικό όσο και επαγγελματικό επίπεδο. Κατά τη διάρκεια της πανδημίας COVID-19, αυτά τα εργαλεία έπαιξαν έναν τεράστιο ρόλο στην διατήρηση της αποτελεσματικότητας των εργαζομένων και στην διασφάλιση ότι οι εργασίες εκτελούνταν ομαλά, ανεξάρτητα από την απόσταση και την απομακρυσμένη εργασία.

Αυτά τα εργαλεία συνέβαλαν στη διατήρηση της αποτελεσματικότητας των εργαζομένων και στην διασφάλιση της ομαλής εκτέλεσης των εργασιών, ανεξάρτητα από τις νέες συνθήκες της απομακρυσμένης εργασίας. Κατά συνέπεια, τα task management apps αποδείχθηκαν απαραίτητα εργαλεία για τη διαχείριση της εργασίας κατά τη διάρκεια της πανδημίας, ενισχύοντας την προσωπική και επαγγελματική παραγωγικότητα.

Τα task management apps διαδραματίζουν έναν σημαντικό ρόλο όσο στην προσωπική ζωή όσο και στην οργάνωση των έργων μιας επιχείρησης και στην εκτέλεση ενός project, καθώς και στον έλεγχο της προόδου του και στην πρόβλεψη πιθανών λαθών. Ας αναλύσουμε πιο αναλυτικά:

Οργάνωση Έργου: Οι εφαρμογές διαχείρισης εργασιών επιτρέπουν τη δημιουργία λεπτομερών λιστών εργασιών και προγραμματισμό τους σε χρονικά πλαίσια. Οι χρήστες μπορούν να αναθέτουν εργασίες σε εαυτούς ή σε άλλους μέλη της ομάδας, ορίζοντας προθεσμίες και προτεραιότητες. Αυτό βοηθά στην αποτελεσματική διαχείριση των εργασιών και στην ευκολότερη παρακολούθησή τους.

Tracking της Προόδου: Οι εφαρμογές διαχείρισης εργασιών παρέχουν εργαλεία για τον έλεγχο

της προόδου των εργασιών. Οι χρήστες μπορούν να σημειώνουν πόσο έχουν προχωρήσει σε κάθε εργασία, να ενημερώνουν την κατάσταση της και να επαναπρογραμματίζουν τυχόν καθυστερήσεις ή αλλαγές στο πλάνο.

Πρόβλεψη Προβλημάτων: Οι εφαρμογές αυτές επιτρέπουν την πρόβλεψη τυχόν προβλημάτων και καθυστερήσεων στο έργο. Οι χρήστες μπορούν να αναγνωρίσουν προβληματικές περιοχές, να αναλύσουν την πρόοδο του έργου και να λάβουν τα απαραίτητα μέτρα για τη διόρθωση των προβλημάτων πριν επιδεινωθούν.

Συνεργασία και Επικοινωνία: Οι εφαρμογές διαχείρισης εργασιών επιτρέπουν την εύκολη συνεργασία μεταξύ των μελών της ομάδας. Οι χρήστες μπορούν να συζητούν τις εργασίες, να στέλνουν σημειώσεις και να ενημερώνουν για την πρόοδο. Αυτό βελτιώνει την επικοινωνία και συνεργασία μεταξύ των μελών της ομάδας, ακόμη και όταν εργάζονται από απόσταση.

Συνολικά, τα task management apps διευκολύνουν τη διαχείριση των έργων, τον έλεγχο της προόδου και την εξασφάλιση ότι οι εργασίες εκτελούνται αποτελεσματικά. Αυτό αποδεικνύεται ιδιαίτερα σημαντικό σε περιβάλλοντα εργασίας όπως αυτό της απομακρυσμένης εργασίας και των προκλήσεων που προέκυψαν λόγω της πανδημίας, καθώς επέτρεψαν στις οργανώσεις να διατηρήσουν την αποτελεσματικότητά τους ακόμη και κάτω από δυσμενείς συνθήκες.

GitHub

Για την δημιουργία του task management application μου χρησιμοποίησα το GitHub το οποίο είναι μια διαδικτυακή πλατφόρμα που χρησιμοποιείται για τη διαχείριση του κώδικα πηγαίου ανοιχτού λογισμικού (open-source) και τη συνεργασία μεταξύ προγραμματιστών και ανάπτυξη λογισμικού. Στην ουσία, το GitHub λειτουργεί ως αποθετήριο (repository) για τον κώδικα, διευκολύνοντας τη διαχείριση, την ενημέρωση και την κοινή χρήση του.

Ορισμένα βασικά χαρακτηριστικά του GitHub περιλαμβάνουν:

Αποθετήρια (Repositories): Τα αποθετήρια αποτελούν τους χώρους όπου αποθηκεύεται ο κώδικας ενός έργου. Τα αποθετήρια μπορούν να είναι δημόσια, οπότε ο κώδικας είναι προσβάσιμος από όλους, ή ιδιωτικά, όπου η πρόσβαση περιορίζεται.

Κλώνοι (Forks): Οι χρήστες μπορούν να δημιουργήσουν ανεξάρτητές αντίγραφα ενός αποθετηρίου, γνωστά ως "forks". Αυτό επιτρέπει την ανάπτυξη του κώδικα ανεξάρτητα και την υποβολή αιτημάτων συγχώνευσης (pull requests) για να ενσωματωθούν αλλαγές στον αρχικό κώδικα.

Συνεργασία (Collaboration): Οι προγραμματιστές μπορούν να συνεργάζονται σε ένα έργο, να σχολιάζουν τον κώδικα, να ανοίγουν ζητήματα (issues) για προβλήματα ή νέες λειτουργίες και να υποβάλλουν αιτήματα συγχώνευσης για να ενσωματώσουν τις αλλαγές τους στον κώδικα.

Εργαλεία Συνεργασίας (Collaboration Tools): Το GitHub παρέχει εργαλεία όπως τα issues, τον διαχωρισμό των εργασιών σε projects, τη διαχείριση των εκδόσεων (versions) του κώδικα, και άλλες χρήσιμες λειτουργίες για τη διαχείριση ενός έργου.

Συνεχής Ολοκλήρωση (Continuous Integration - CI) και Αυτοματοποιημένες Δοκιμές (Automated Testing): Το GitHub επιτρέπει την ενσωμάτωση της διαδικασίας CI και των αυτοματοποιημένων δοκιμών στον κώδικα, προκειμένου να εξασφαλίσει την ποιότητα και τη σταθερότητα του.

Το GitHub το διάλεξα σαν πλατφόρμα καθώς μέσα από αυτό μπορούσα να έχω τη διαχείριση Κώδικα καθώς είναι μια πλατφόρμα που επιτρέπει να διαχειρίζεστε τον κώδικά. Μπορείτε να δημιουργείτε repositories για τα έργα σας και να ανεβάζετε τον κώδικά σας, κάνοντας τη

διαχείριση του κώδικα πιο εύκολη.

Επίσης βοηθάει με το version control ή αλλιώς τις εκδόσεις της εφαρμογής καθώς μπορείτε να δημιουργείτε εκδόσεις του λογισμικού σας, να διαχειρίζεστε τις αλλαγές και να παρακολουθείτε την εξέλιξη του έργου σας με τον χρόνο. Πέρα από την εξέλιξη του έργου η πλατφόρμα αυτή επιτρέπει την παρακολούθηση των αλλαγών στο έργο.
Ώπως αναφέραμε και ποιο πριν για μεγαλύτερα project υποστηρίζει και ci / cd (Continuous Integration/Continuous Deployment) για την αυτόματη δοκιμή και ανανέωση του κώδικα.

Τέλος το GitHub παρέχει διαφάνεια όσον αφορά τις αλλαγές και την εξέλιξη του έργου. Αυτό βοηθά στην καλύτερη κατανόηση της προόδου του έργου από τους συνεργάτες και την κοινότητα.

Node Package Manager

Το Npm αντιπροσωπεύει το "Node Package Manager" και είναι ένα από τα πιο δημοφιλή εργαλεία για τη διαχείριση πακέτων (packages) και την διανομή JavaScript κώδικα. Τα κύρια σημεία που πρέπει να γνωρίζετε σχετικά με το Npm περιλαμβάνουν:

Τι είναι το Npm:

Διαχείριση Πακέτων: Το Npm διευκολύνει τη διαχείριση πακέτων JavaScript που χρησιμοποιούνται σε ένα έργο. Αυτά τα πακέτα μπορεί να είναι βιβλιοθήκες, πλαίσια εργασίας (frameworks), εργαλεία ανάπτυξης, και πολλά άλλα.

Κατανομή Κώδικα: Το Npm επιτρέπει την κατανομή του JavaScript κώδικα, συμπεριλαμβανομένων των πακετών και των εφαρμογών, προς άλλους προγραμματιστές σε όλο τον κόσμο.

Εξάρτηση Πακέτων: Το Npm αναλαμβάνει τη διαχείριση των εξαρτήσεων πακέτων, διασφαλίζοντας ότι τα απαιτούμενα πακέτα είναι διαθέσιμα και συμβατά με το έργο σας.

Θετικά του Npm:

Διαθεσιμότητα: Το Npm παρέχει έναν τεράστιο κατάλογο από χιλιάδες πακέτα που καλύπτουν σχεδόν κάθε ανάγκη ανάπτυξης.

Απλότητα: Είναι εύκολο στη χρήση και αποκτάτε πρόσβαση σε πακέτα με μερικές απλές εντολές.

Αυτοματοποίηση: Το Npm επιτρέπει την αυτοματοποίηση των εργασιών που αφορούν τη διαχείριση πακέτων μέσω του αρχείου package.json.

Κοινότητα: Υπάρχει μια μεγάλη και ενεργή κοινότητα πίσω από το Npm που συνεχώς συνεισφέρει στον κατάλογο πακέτων και παρέχει υποστήριξη.

Αρνητικά του Npm:

Προβλήματα συμβατότητας: Καθώς το Npm δίνει τη δυνατότητα σε πολλούς προγραμματιστές να δημιουργούν πακέτα, υπάρχει περίπτωση ασυμβατότητας μεταξύ πακετών.

Μέγεθος του Node Modules φακέλου: Ο Node Modules φάκελος μπορεί να γίνει πολύ μεγάλος, καθώς προστίθενται πολλά πακέτα, και αυτό μπορεί να αυξήσει τον χώρο αποθήκευσης που απαιτείται για το έργο σας.

Ασφάλεια: Η εγκατάσταση πακέτων από τρίτους μπορεί να δημιουργήσει προβλήματα ασφαλείας, καθώς δεν είναι πάντα δυνατή η ελέγχου της ποιότητας και της ασφάλειας των πακετών.

Συνολικά, το Npm είναι ένα ισχυρό εργαλείο για τους προγραμματιστές JavaScript που διευκολύνει τη διαχείριση των πακέτων και την ανάπτυξη εφαρμογών. Ωστόσο, πρέπει να είστε προσεκτικοί και να ελέγχετε τα πακέτα που χρησιμοποιείτε για ασφάλεια και συμβατότητα.

Vue.js

Υπάρχει πλήθος ερευνών και πηγών που αναφέρονται στη Vue.js. Συγκεκριμένα, χρησιμοποιήθηκε το documentation από το official website της vue js, το οποίο παρέχει μια εξαντλητική εισαγωγή στην χρήση της vue js, καθώς και πρακτικά παραδείγματα για την ανάπτυξη εφαρμογών.

Πιο συγκεκριμένα το Vue js σαν framework:

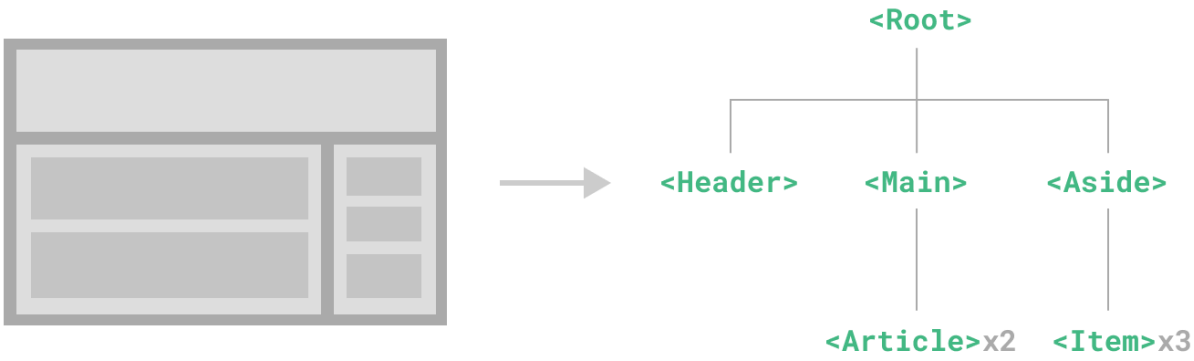
Το Vue.js είναι ένα πλαίσιο ανοικτού κώδικα για τη δημιουργία χρήστη-φιλικών ενιαίων εφαρμογών διαδικτύου (SPA) και διαχείρισης διεπαφών χρήστη. Στην έκδοση 3 του Vue.js, υπάρχουν ορισμένες αλλαγές και βελτιώσεις σε σχέση με τις προηγούμενες εκδόσεις. Ας δούμε πώς λειτουργεί το Vue.js 3 σε λεπτομέρεια:

1) Δήλωση Component: Η βασική μονάδα του Vue.js είναι το "component". Ένα component είναι μια ανεξάρτητη μονάδα που περιέχει HTML, CSS και JavaScript για τη συγκεκριμένη διεπαφή.

Για να δηλώσετε ένα component, μπορείτε να χρησιμοποιήσετε τη μέθοδο.

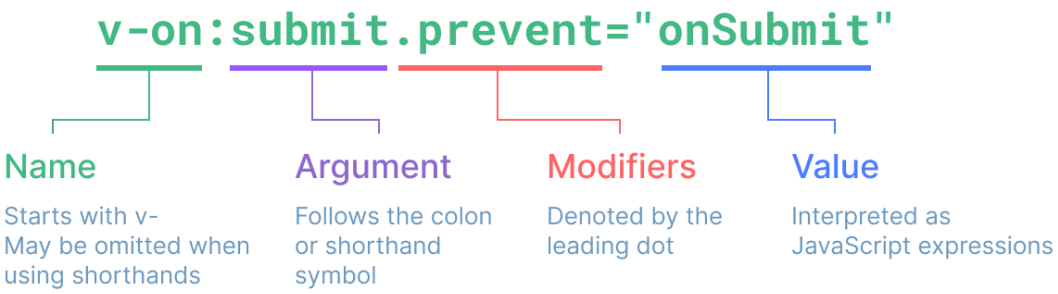
Παραδείγμα:

```
const app = Vue.createApp({
  data() {
    return {
      message: 'Καλημέρα, Vue!'
    }
  }
})
```



2) Δέσμευση Δεδομένων (Data Binding): Μπορείτε να δηλώσετε μεταβλητές δεδομένων στο component και να τις δεσμεύσετε στο HTML χρησιμοποιώντας διάφορες οδηγίες. Για παράδειγμα:

```
<div id="app">
  <p>{{ message }}</p>
</div>
```



3) Συναρτήσεις (Methods): Μπορείτε να προσθέσετε μεθόδους στο component για την επεξεργασία δεδομένων ή αντίδραση σε συμβάντα, όπως το κλικ του χρήστη.

Παραδείγμα:

```
const app = Vue.createApp({
  data() {
    return {
      count: 0
    }
  },
  methods: {
    increment() {
      this.count++
    }
  }
})
```

4) Οδηγίες (Directives): Οι οδηγίες είναι ειδικά χαρακτηριστικά που προστίθενται στο HTML για την επικοινωνία με το Vue.js. Παραδείγματα περιλαμβάνουν το v-bind για δέσμευση τιμών, το v-on για ακρόαση συμβάντων και το v-for για επανάληψη στοιχείων.

Παραδείγμα:

```
<button v-on:click="increment">Αύξηση</button>
<ul>
  <li v-for="item in items">{{ item }}</li>
</ul>
```

5) Στο Vue.js, τα computed properties (υπολογιζόμενες ιδιότητες) είναι μια ισχυρή λειτουργία που σας επιτρέπει να υπολογίσετε δεδομένα με βάση τις τιμές άλλων ιδιοτήτων, και να τα παρακολουθείτε για αλλαγές. Συνήθως χρησιμοποιούνται για να υπολογίσετε και να εμφανίσετε δεδομένα στο προτυπικό (template) του Vue component, και να διατηρήσετε την ανταπόκριση της διεπαφής σας στις αλλαγές.

Οι υπολογιζόμενες ιδιότητες δηλώνονται στο Vue component με τη βοήθεια του computed αντικειμένου.

Εδώ είναι ένα παράδειγμα:

```
<template>
  <div>
    <p>Το πλάτος είναι: {{ width }}</p>
    <p>Το ύψος είναι: {{ height }}</p>
    <p>Το εμβαδόν είναι: {{ area }}</p>
  </div>
</template>

<script>
export default {
  data() {
    return {
      length: 5,
      width: 3,
    };
```

```
  },
  computed: {
    height() {
      return this.length * 2;
    },
    area() {
      return this.length * this.width;
    },
  },
};
</script>
```

Στο παραπάνω παράδειγμα, έχουμε ένα Vue component που δηλώνει δύο βασικές ιδιότητες length και width στο data. Επίσης, χρησιμοποιούμε το computed για να υπολογίσουμε το height και το area ως υπολογιζόμενες ιδιότητες.

Το πλεονέκτημα της χρήσης υπολογιζόμενων ιδιοτήτων είναι ότι αυτές οι τιμές υπολογίζονται αυτόματα και ανανεώνονται μόνο όταν οι εξαρτώμενες ιδιότητες (σε αυτή την περίπτωση, length και width) αλλάζουν. Αυτό σημαίνει ότι δεν χρειάζεται να επανυπολογίσετε τα δεδομένα χειροκίνητα κάθε φορά που αλλάζετε τις εισόδους. Ο Vue.js αναλαμβάνει τον υπολογισμό και την ανανέωση τους αυτόματα.

6) Τα "Class Bindings" και "Style Bindings" στο Vue.js είναι τρόποι για να δηλώσετε δυναμικά κλάσεις και στυλ σε στοιχεία HTML με βάση την κατάσταση ή τα δεδομένα του Vue component. Αυτοί οι δυναμικοί δεσμευτικοί μηχανισμοί είναι χρήσιμοι όταν θέλετε να αλλάξετε την εμφάνιση των στοιχείων HTML ανάλογα με την κατάσταση ή τα δεδομένα της εφαρμογής.

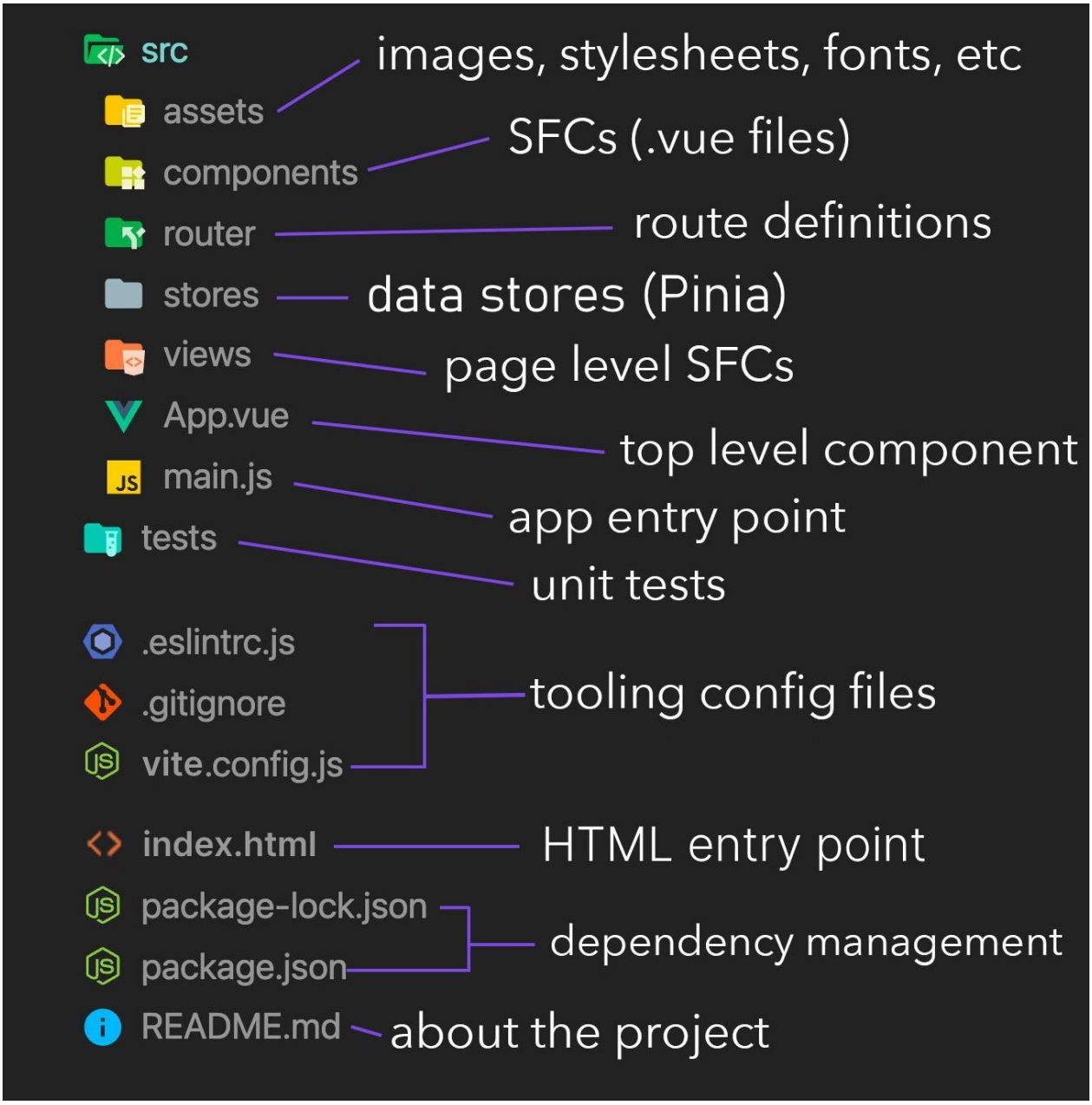
Για να δηλώσετε δυναμικές κλάσεις σε ένα στοιχείο HTML, μπορείτε να χρησιμοποιήσετε το v-bind:class ή τη συντόμευσή του :class. Αυτός ο μηχανισμός είναι χρήσιμος για να προσθέσετε ή να αφαιρέσετε κλάσεις ανάλογα με τις συνθήκες.

Παραδείγμα:

```
<template>
  <div>
    <!-- Χρησιμοποιείτε το :class για να δηλώσετε δυναμικές κλάσεις -->
    <div :class="{ active: isActive, error: hasError }">Κείμενο</div>
  </div>
</template>

<script>
export default {
  data() {
    return {
      isActive: true,
      hasError: false,
    };
```

```
  },
};
</script>
```

7) Δομή Εφαρμογής (): Συνήθως, μια εφαρμογή Vue.js αποτελείται από πολλά components που ενσωματώνονται μαζί για να δημιουργήσουν την τελική διεπαφή. Η δομή της εφαρμογής συνήθως περιλαμβάνει έναν ριζικό (root) component που αποτελεί το κέντρο της εφαρμογής.



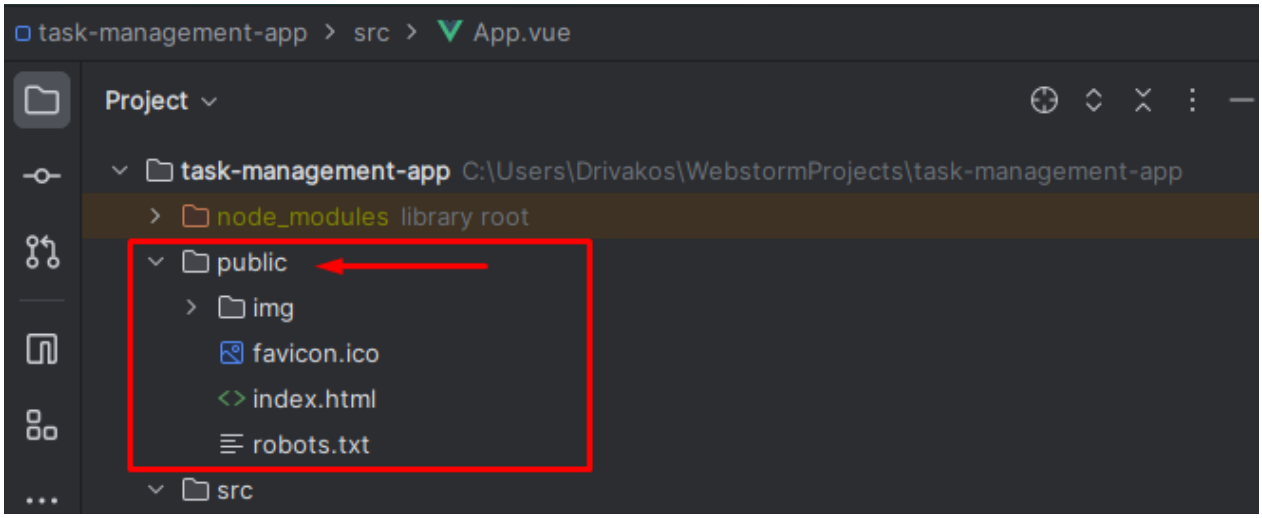
Η δομή μιας εφαρμογής γραμμένης με το Vue.js μπορεί να ποικίλλει ανάλογα με το μέγεθος και τις απαιτήσεις της εφαρμογής, αλλά υπάρχει μια τυπική δομή που συνιστάται για μεγαλύτερη οργάνωση και συντηρησιμότητα.

Εδώ είναι μια τυπική δομή μιας εφαρμογής Vue.js:

```
my-app/
├── public/
│   ├── index.html
│   └── ...
├── src/
│   ├── assets/
│   ├── components/
│   ├── views/
│   ├── App.vue
│   └── main.js
├── package.json
└── ...
```

Πιο αναλυτικά :

1) public/: Σε αυτόν τον κατάλογο βρίσκονται στατικά αρχεία όπως το index.html, εικόνες, και άλλα αρχεία που δεν χρειάζονται επεξεργασία από τον Webpack ή το Vue CLI.



Στο Vue.js, ο φάκελος "public" αποτελεί έναν από τους βασικούς φακέλους στη δομή του Vue.js project και χρησιμοποιείται για να φιλοξενεί στατικά αρχεία και πόρους που δεν χρειάζονται περαιτέρω επεξεργασία ή διαχείριση από τον Vue CLI ή το webpack.

Στο φάκελο "public", μπορείτε να τοποθετήσετε αρχεία όπως εικόνες, γραφικά, αρχεία CSS, αρχεία JavaScript, και άλλους στατικούς πόρους που χρειάζονται οι σελίδες της εφαρμογής σας.

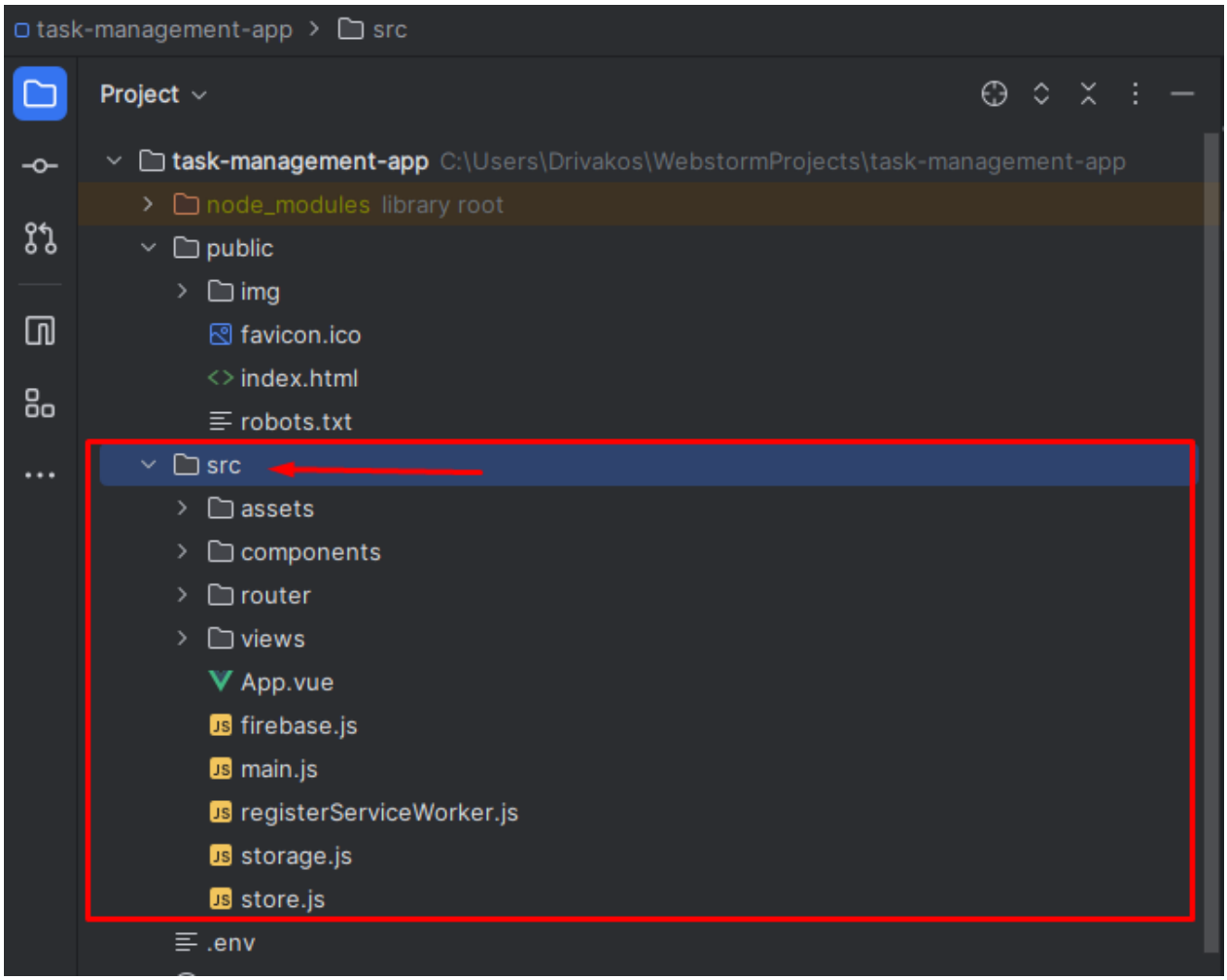
Οι στατικοί πόροι που τοποθετούνται στον φάκελο "public" δεν υπόκεινται σε διαχείριση μεταγλώττισης, συμπίεσης, ή μετατροπής από το Vue CLI. Απλά αντιγράφονται απευθείας στον κατάλογο εξόδου της εφαρμογής κατά τη διάρκεια της διαδικασίας μεταγλώττισης (build), και μπορούν να προσπελαστούν από τον πελάτη με τον τυπικό

τρόπο.

Ένα κοινό παράδειγμα χρήσης του φακέλου "public" είναι η τοποθέτηση του κύριου HTML αρχείου (όπως το index.html) εκεί. Επίσης, μπορείτε να τοποθετήσετε εξωτερικές βιβλιοθήκες (π.χ. Bootstrap, Font Awesome) ή προσαρμοσμένα στυλ CSS που δεν χρειάζονται περαιτέρω επεξεργασία όπου στην συγκεκριμένη περίπτωση δεν χρειαστήκαμε καθώς γράφτηκαν custom stylings για κάθε component.

Είναι σημαντικό να γνωρίζετε ότι τα αρχεία που τοποθετούνται στον φάκελο "public" δεν υπόκεινται στην πολυπλοκότητα του συστήματος μεταγλώττισης (build system) και είναι προσβάσιμα απευθείας από τον διακομιστή (server) χωρίς περαιτέρω επεξεργασία.

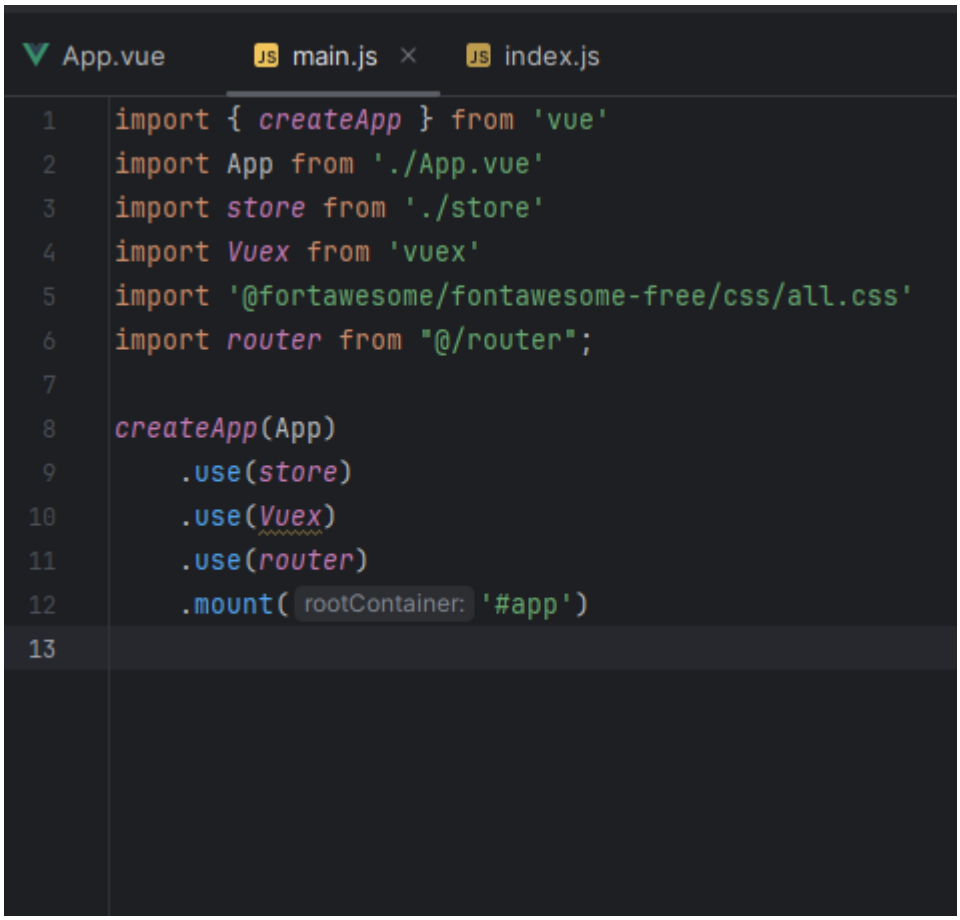
2) src/: Αυτός είναι ο βασικός κατάλογος της εφαρμογής. Ορίζει την JavaScript, τα components, και τις διεπαφές της εφαρμογής.



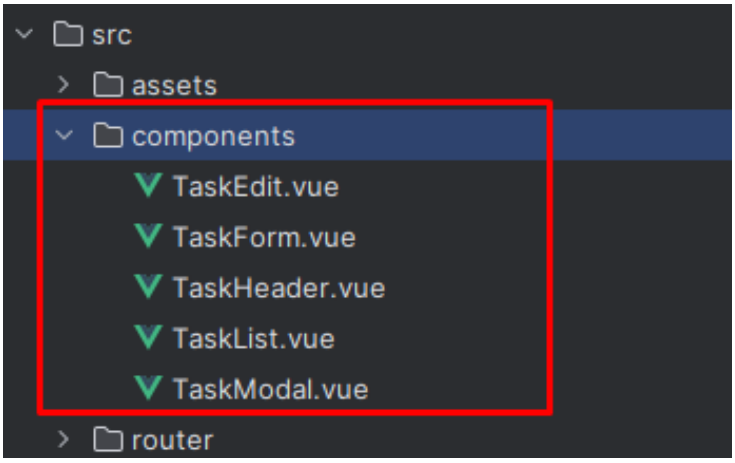
Στον φάκελο "src" (που είναι συντομογραφία του "source") σε ένα project Vue.js, τοποθετούνται όλα τα πηγαία αρχεία και τον κώδικα της εφαρμογής σας. Αυτός ο

φάκελος είναι το κέντρο του ανάπτυγματικού περιβάλλοντος για την εφαρμογή σας και περιλαμβάνει τα ακόλουθα στοιχεία:

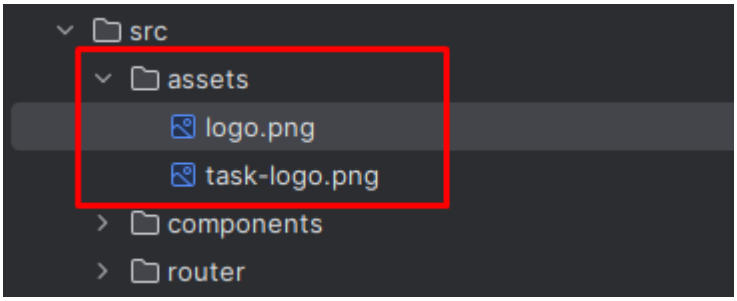
1. main.js: Αυτό το αρχείο είναι ο εισαγωγικός σημείο της εφαρμογής σας. Είναι εδώ που δημιουργείτε το Vue instance και συνδέετε το κεντρικό Vue component με το HTML αρχείο.



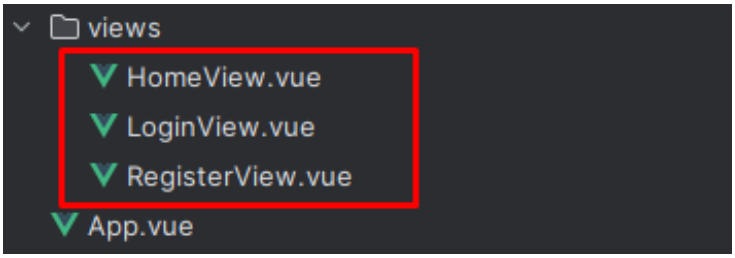
2. Υποφάκελοι Vue components: Ενδέχεται να έχετε πολλά Vue components στον υποφάκελο "components". Αυτά τα components αντιπροσωπεύουν διάφορα τμήματα της εφαρμογής σας και είναι υπεύθυνα για τον σχεδιασμό και τη λογική τους.



3. Υποφάκελος assets: Στον φάκελο "assets", μπορείτε να τοποθετήσετε εικόνες, γραφικά, στυλ CSS και άλλους πόρους που θέλετε να χρησιμοποιήσετε στην εφαρμογή σας.



4. Υποφάκελος views: Συνήθως, ο φάκελος "views" περιέχει τα Vue components που αντιπροσωπεύουν τις σελίδες της εφαρμογής σας. Κάθε σελίδα μπορεί να έχει το δικό της Vue component.

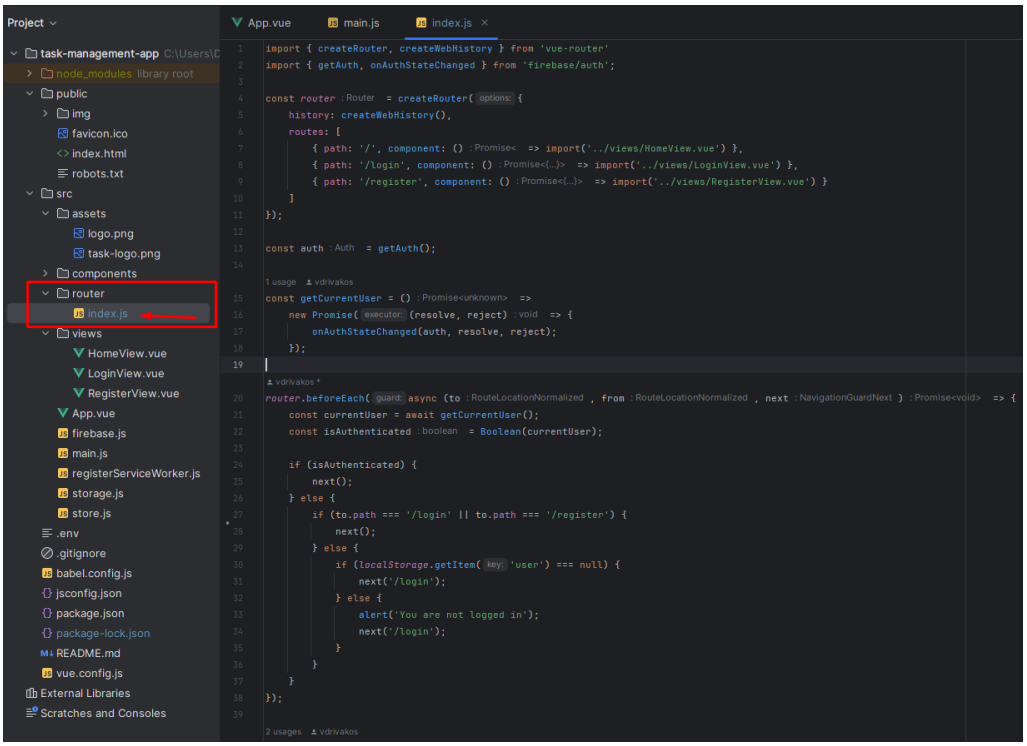


5. App.vue: Το αρχείο App.vue είναι το κύριο Vue component που περιβάλλει όλη την εφαρμογή και περιλαμβάνει τη δομή της σελίδας και τα κοινά στοιχεία όπως μπάρα πλοήγησης.

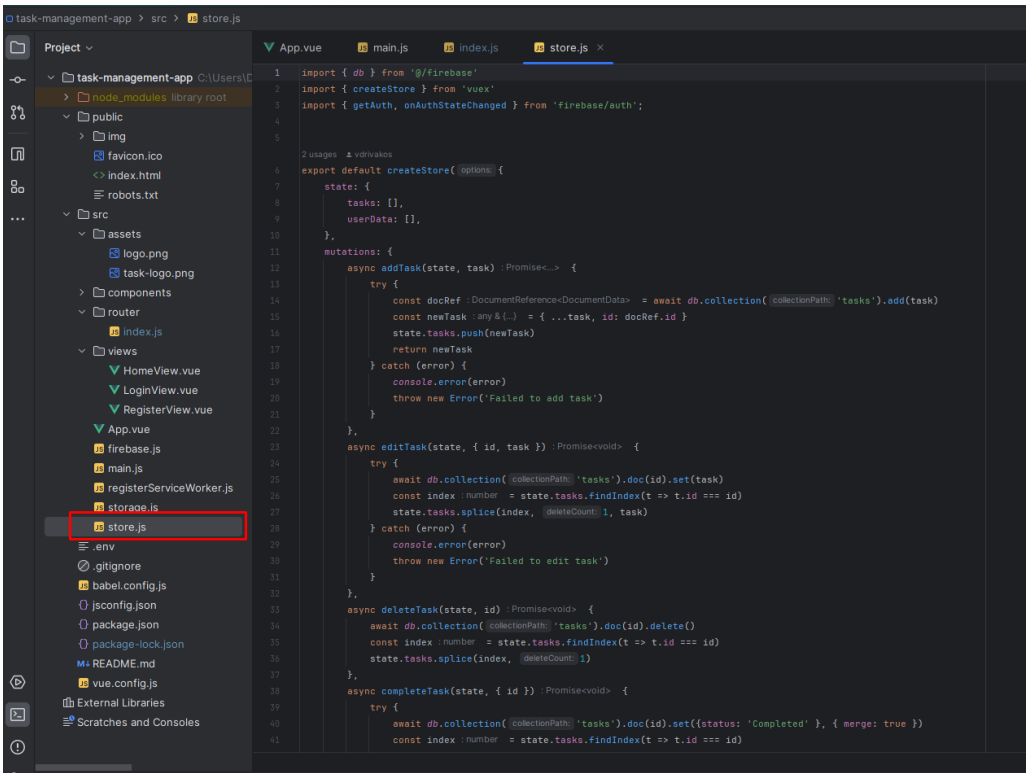
App.vue × main.js index.js

```
<template class="wrapper">
 <task-header></task-header>
 <router-view/>
</template>

<style scoped>
</style>
<script setup>
import TaskHeader from "./components/TaskHeader.vue";
</script>

<style scoped>
  .logo-wrapper img {
    max-width: 300px;
    display: block;
    margin: 0 auto;
  }
</style>

```

6. router/index.js: Αυτό το αρχείο αντιπροσωπεύει το router της εφαρμογής σας, όταν χρησιμοποιείτε το Vue Router για τον χειρισμό των διαδρομών της σελίδας.



7. store/index.js: Αν χρησιμοποιείτε το Vuex για τη διαχείριση της κατάστασης της εφαρμογής, αυτό το αρχείο αντιπροσωπεύει τον κεντρικό αποθηκευτικό χώρο (store).



Ο φάκελος "src" είναι ο πιο σημαντικός φάκελος στη δομή ενός Vue.js project, καθώς περιλαμβάνει τον κώδικα και τα αρχεία που αναπτύσσετε και επεξεργάζεστε κατά τη διάρκεια της ανάπτυξης της εφαρμογής σας.

3) assets/: Εδώ μπορείτε να αποθηκεύσετε εικόνες, στυλ CSS, ή άλλα αρχεία που χρησιμοποιούνται από την εφαρμογή.

Ο φάκελος "assets" σε ένα project Vue.js χρησιμοποιείται για την αποθήκευση στατικών αρχείων και πόρων όπως εικόνες, γραφικά, στυλ CSS, ή άλλα αρχεία που θα χρησιμοποιηθούν στην εφαρμογή σας. Αυτός ο φάκελος συνήθως περιλαμβάνει αρχεία που δεν απαιτούν περαιτέρω επεξεργασία από τον webpack ή το Vue CLI, αλλά απλά είναι πόροι που η εφαρμογή σας χρησιμοποιεί.

Συγκεκριμένα, μπορείτε να χρησιμοποιήσετε τον φάκελο "assets" για:

1. Εικόνες: Μπορείτε να τοποθετήσετε εικόνες που θα χρησιμοποιηθούν στον ιστότοπό σας. Αυτές μπορεί να είναι εικόνες προφίλ, λογότυπα, φόντο, ή οτιδήποτε άλλο χρειάζεται να εμφανιστεί στη σελίδα.

2. Στυλ CSS: Εάν έχετε προσαρμοσμένα στυλ CSS που θέλετε να εφαρμόσετε στην εφαρμογή σας, μπορείτε να τα αποθηκεύσετε στον φάκελο "assets" και

να τα φορτώσετε στην εφαρμογή σας.

3. Αρχεία γραφικών, ήχου κ.λπ.: Ο φάκελος "assets" μπορεί να χρησιμοποιηθεί για την αποθήκευση άλλων στατικών αρχείων όπως εικονίδια, ήχοι, γραφικά, βίντεο και οτιδήποτε άλλο θεωρείτε απαραίτητο για την εφαρμογή σας.

Όταν τοποθετείτε πόρους στον φάκελο "assets," μπορείτε να αναφερθείτε σε αυτούς από τον κώδικά σας χρησιμοποιώντας σχετικές διαδρομές, όπως @/assets/image.jpg για μια εικόνα που βρίσκεται στον φάκελο "assets."

Αυτός ο φάκελος βοηθά στον οργανισμό των στατικών πόρων της εφαρμογής σας και την ευκολότερη διαχείριση τους, ενώ παράλληλα διατηρεί τον κώδικα της εφαρμογής σας καθαρό και δομημένο.

4) components/: Σε αυτόν τον κατάλογο βάζετε τα Vue components που χρησιμοποιούνται στην εφαρμογή.

Ο υποφάκελος "components" στον φάκελο "src" ενός project Vue.js χρησιμοποιείται για την οργάνωση και την τοποθέτηση των Vue components που αναπτύσσετε ώστε να διαχειρίζεστε τη διεπαφή χρήστη (UI) και τη λογική της εφαρμογής σας. Οι Vue components αντιπροσωπεύουν διάφορα τμήματα της εφαρμογής σας και μπορούν να επαναχρησιμοποιηθούν ανάλογα με τις ανάγκες σας.

Εδώ είναι μερικά βασικά σημεία για τον υποφάκελο "components":

1. Οργάνωση: Ο φάκελος "components" χρησιμοποιείται για να οργανώσετε τα διάφορα Vue components που αναπτύσσετε στην εφαρμογή σας. Μπορείτε να δημιουργήσετε υποφακέλους μέσα σε αυτόν το φάκελο για να ομαδοποιήσετε components που σχετίζονται με την ίδια λειτουργικότητα.

2. Επαναχρησιμοποίηση: Οι Vue components είναι επαναχρησιμοποιήσιμοι. Μπορείτε να χρησιμοποιήσετε τα ίδια components σε διάφορα μέρη της εφαρμογής σας, βοηθώντας στη διατήρηση της συνοχής και της ευκολίας συντήρησης.

3. Διαχείριση Λογικής: Συνήθως, τα Vue components περιλαμβάνουν τόσο το HTML που καθορίζει τη δομή του UI όσο και το JavaScript που περιγράφει τη λογική που αναπτύσσει το component. Αυτή η διάσπαση του κώδικα σε components βοηθά στην ευκολία κατανόησης και συντήρησης.

4. Εξατομίκευση: Τα Vue components μπορούν να εξατομικευθούν με διάφορες παραμέτρους (props) που μπορείτε να περάσετε στα components, καθιστώντας τα ευέλικτα και δυνατά για διάφορες χρήσεις.

Συνολικά, ο φάκελος "components" είναι ένας βασικός μέρος της δομής του κώδικά σας σε ένα project Vue.js και σας βοηθά να οργανώσετε και να διαχειριστείτε τα διάφορα μέρη της εφαρμογής σας με αποτελεσματικό και επαναχρησιμοποιήσιμο τρόπο.

5) views/: Συνήθως, αυτός ο κατάλογος περιέχει τις διεπαφές των σελίδων της εφαρμογής, οι οποίες μπορεί να περιλαμβάνουν διάφορα components.

Ο φάκελος "views" σε ένα project Vue.js χρησιμοποιείται για την οργάνωση των Vue components που αντιπροσωπεύουν τις σελίδες της εφαρμογής σας. Συνήθως, κάθε αρχείο Vue component που βρίσκεται στον φάκελο "views" αντιπροσωπεύει μια σελίδα της εφαρμογής σας. Ο φάκελος "views" χρησιμοποιείται για να διαχωρίσετε τις διάφορες σελίδες της εφαρμογής σας και να οργανώσετε την δομή τους.

Εδώ είναι μερικά βασικά σημεία σχετικά με τον φάκελο "views":

1. Σελίδες: Κάθε αρχείο Vue component στον φάκελο "views" αντιπροσωπεύει μια σελίδα της εφαρμογής σας. Για παράδειγμα, μπορείτε να έχετε ένα αρχείο "Home.vue" που αντιπροσωπεύει την αρχική σελίδα της εφαρμογής, και ένα αρχείο "About.vue" που αντιπροσωπεύει τη σελίδα "Σχετικά με εμάς."

2. Διαχείριση Δρομολόγησης: Συνήθως, ο φάκελος "views" χρησιμοποιείται σε συνδυασμό με το Vue Router για τη διαχείριση της δρομολόγησης στην εφαρμογή σας. Κάθε αρχείο Vue component που αντιπροσωπεύει μια σελίδα μπορεί να συσχετιστεί με μια συγκεκριμένη διαδρομή URL.

3. Διαχείριση Κατάστασης: Αν η σελίδα σας απαιτεί τη διατήρηση κατάστασης, μπορείτε να χρησιμοποιήσετε το Vuex ή άλλη λύση διαχείρισης κατάστασης για να μοιράζεστε δεδομένα μεταξύ των σελίδων που βρίσκονται στον φάκελο "views."

4. Διαχωρισμός Ευθύνης: Ο χωρισμός των Vue components μεταξύ των φακέλων "components" και "views" βοηθά στον διαχωρισμό της ευθύνης. Τα components συνήθως αναπτύσσονται για επαναχρησιμοποίηση, ενώ τα views αντιπροσωπεύουν την εξωτερική διεπαφή και την διαχείριση της δρομολόγησης.

Ο φάκελος "views" είναι ένας σημαντικός στοιχείο για τη δομή της εφαρμογής Vue.js σας και βοηθά στον οργανισμό και τη διαχείριση των σελίδων της εφαρμογής σας.

6) App.vue: Το κύριο component που αντιπροσωπεύει την ολική διεπαφή της εφαρμογής. Περιέχει τη δομή της σελίδας, την κύρια διαχείριση της κατάστασης, και το δρομολόγιο.

Το αρχείο "App.vue" είναι το κύριο Vue component της εφαρμογής σας σε ένα project Vue.js. Αυτό το αρχείο περιλαμβάνει τη δομή και τη συμπεριφορά του κύριου πλαισίου (layout) της εφαρμογής σας. Συνήθως, περιέχει τα εξής στοιχεία:

Δομή (HTML): Το αρχείο "App.vue" περιλαμβάνει τη δομή της εφαρμογής σας, συμπεριλαμβανομένων των διάφορων στοιχείων HTML όπως επικεφαλίδα (header), μπάρα πλοήγησης, περιεχόμενο, και άλλα.

Στυλ (CSS): Συχνά περιλαμβάνει στυλ CSS που αφορούν το γενικό σχήμα και την

εμφάνιση της εφαρμογής σας.

Κεντρικό Vue Component: Το "App.vue" εισάγει ένα κύριο Vue component, το οποίο ενδέχεται να ονομάζεται "App" ή κάτι παρόμοιο. Αυτό το component αντιπροσωπεύει το κεντρικό πλαίσιο της εφαρμογής σας και συνήθως περιλαμβάνει τον υπόλοιπο δυναμικό περιεχόμενο της εφαρμογής.

Ένα βασικό παράδειγμα του πώς μπορεί να δομηθεί το αρχείο "App.vue" είναι το εξής:

```
 <template>
<div>
 <header>
  <!-- Εδώ μπορεί να είναι η μπάρα πλοήγησης -->
 </header>
 <main>
  <!-- Εδώ μπορεί να είναι το κεντρικό περιεχόμενο της εφαρμογής -->
  <router-view></router-view> <!-- Χρησιμοποιείται για τη δρομολόγηση
σελίδων -->
 </main>
 <footer>
  <!-- Εδώ μπορεί να είναι ο υποσέλιδος -->
 </footer>
</div>
</template>

<script>
export default {
 name: 'App',
 // Εδώ μπορείτε να προσθέσετε τη λογική του App.vue component
}
</script>

<style>
/* Εδώ μπορείτε να προσθέσετε στυλ CSS για το App.vue component */
</style>
```

Το αρχείο "App.vue" είναι η βάση της εφαρμογής σας και περιλαμβάνει τον σκελετό της διεπαφής χρήστη, τον κώδικα Vue.js που διαχειρίζεται τη γενική συμπεριφορά της εφαρμογής, και τυχόν γενικά στυλ για την εμφάνιση της εφαρμογής.

7) main.js: Αυτό το αρχείο είναι υπεύθυνο για την εκκίνηση της εφαρμογής. Εδώ εισάγονται τα απαραίτητα πακέτα και τα Vue components, και δημιουργείται η Vue εφαρμογή.

Το αρχείο "main.js" είναι το εισαγωγικό σημείο (entry point) της εφαρμογής σας σε ένα project Vue.js. Αυτό το αρχείο είναι υπεύθυνο για τη δημιουργία του Vue instance και τη

σύνδεση του κεντρικού Vue component με το HTML αρχείο της εφαρμογής σας. Συνήθως, το "main.js" περιλαμβάνει τις εξής λειτουργίες:

Δημιουργία Vue Instance: Στο "main.js," δημιουργείτε ένα νέο Vue instance χρησιμοποιώντας τον constructor του Vue. Αυτό το instance θα χρησιμοποιηθεί για τη δημιουργία και τη διαχείριση των Vue components της εφαρμογής σας.

Παράδειγμα :

```
 import Vue from 'vue';
const app = new Vue({
  // Εδώ μπορείτε να παραμετροποιήσετε το Vue instance
});
```

Δημιουργία Vue Router (εάν χρησιμοποιείτε Router):
Εάν χρησιμοποιείτε το Vue Router για τη διαχείριση της δρομολόγησης στην εφαρμογή σας, το "main.js" είναι το σημείο όπου δημιουργείτε τον router και τον συνδέετε με το Vue instance.
Javascript

Παράδειγμα :

```
import Vue from 'vue';
import VueRouter from 'vue-router';
import App from './App.vue';

Vue.use(VueRouter);

const router = new VueRouter({
  // Εδώ μπορείτε να δηλώσετε τις διαδρομές της εφαρμογής σας
});

new Vue({
  router,
  render: h => h(App)
}).$mount('#app');
```

Σύνδεση του Vue Component με το HTML: Το "main.js" συνήθως περιλαμβάνει κώδικα για τη σύνδεση του κύριου Vue component (συνήθως το "App.vue") με το HTML αρχείο της εφαρμογής σας. Αυτό γίνεται με τη χρήση της μεθόδου $mount, η οποία συνδέει το Vue instance με ένα στοιχείο HTML στο οποίο θα ρενταριστεί η εφαρμογή.

Παράδειγμα:

```
new Vue({
  // ... άλλες ρυθμίσεις
```

}).$mount('#app');

Το "main.js" είναι το πρώτο αρχείο που φορτώνεται κατά την εκκίνηση της εφαρμογής σας και παίζει κρίσιμο ρόλο στην αρχικοποίηση του Vue environment και τη σύνδεση του με την HTML σελίδα της εφαρμογής σας.

8) package.json: Αυτό το αρχείο περιέχει τις εξαρτήσεις της εφαρμογής και τις συντομογραφίες για τη διαχείριση του έργου.

Το αρχείο "package.json" είναι ένα αρχείο που χρησιμοποιείται σε πολλά JavaScript projects, συμπεριλαμβανομένων των project Vue.js, για τη διαχείριση των εξαρτήσεων (dependencies), των σεναρίων (scripts), των πληροφοριών έκδοσης και άλλων ρυθμίσεων του project. Το "package.json" περιέχει μεταδεδομένα για το πώς το project πρέπει να διαχειρίζεται, να εγκαθιστά εξαρτήσεις, να εκτελεί σενάρια, και άλλες σχετικές πληροφορίες.

Ορισμένα από τα κύρια στοιχεία που περιλαμβάνει το "package.json" αρχείο είναι:

1. Εξαρτήσεις (Dependencies): Συμπεριλαμβάνει τις εξαρτήσεις του project, που είναι βιβλιοθήκες και πακέτα που χρησιμοποιούνται στο project. Αυτές οι εξαρτήσεις μπορούν να είναι πλαίσια όπως το Vue.js, το Vue Router, το Vuex, ή άλλες βιβλιοθήκες που χρησιμοποιούνται στο project.

2. Σενάρια (Scripts): Περιλαμβάνει πληροφορίες σχετικά με τα σενάρια που μπορείτε να εκτελέσετε στο project. Για παράδειγμα, μπορεί να περιέχει σενάρια για την εκτέλεση της εφαρμογής σας σε έναν τοπικό διακομιστή ανάπτυξης, την παραγωγή της έκδοσης παραγωγής, τον έλεγχο του κώδικα και πολλά άλλα.

3. Πληροφορίες Έκδοσης (Version Information): Περιέχει πληροφορίες σχετικά με την έκδοση της εφαρμογής σας, συμπεριλαμβανομένων του ονόματος του project, της έκδοσης, της περιγραφής, και άλλων παραμέτρων που αφορούν την εφαρμογή.

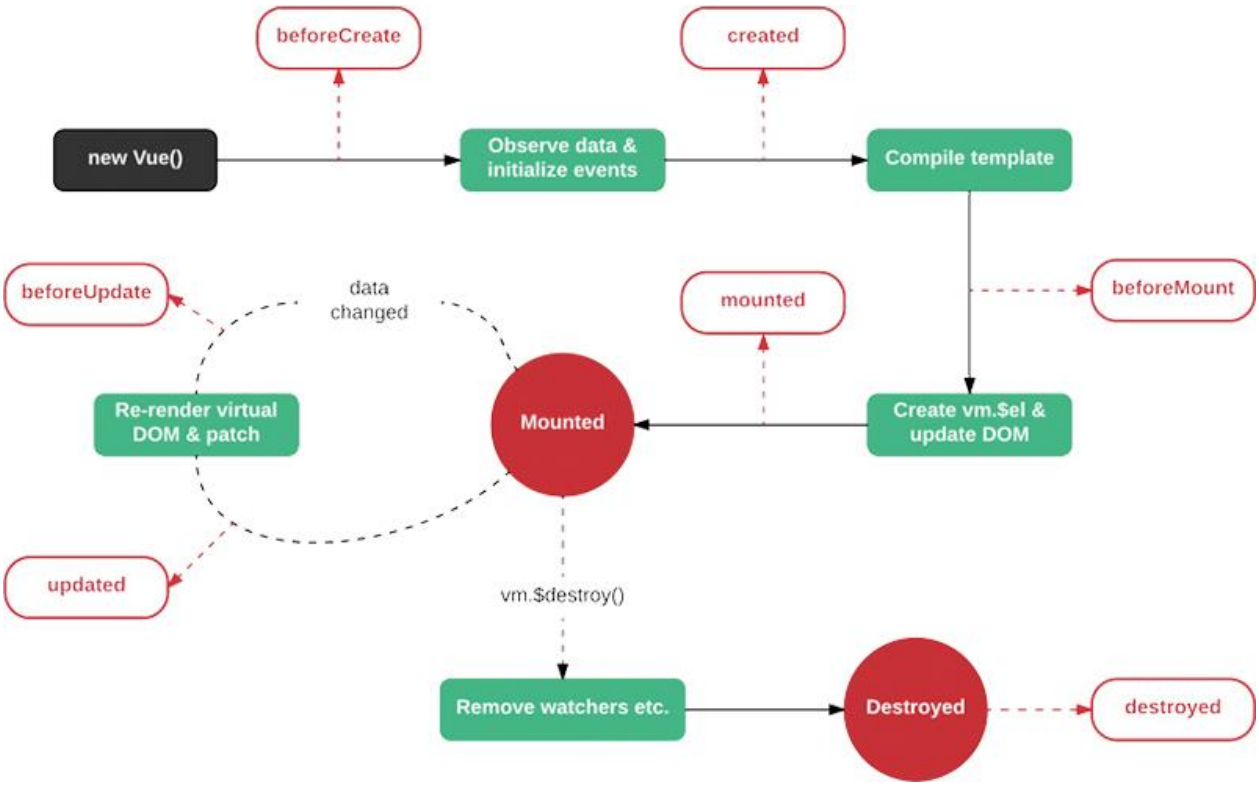
4. Στυλ Κώδικα (Code Style): Περιέχει ρυθμίσεις στυλ κώδικα, όπως το είδος (indentation), οι κανόνες του linter, και άλλες σχετικές παραμέτρους που βοηθούν στον έλεγχο της ποιότητας του κώδικα.

5. Εξαρτήσεις Ανάπτυξης (Development Dependencies): Περιλαμβάνει εξαρτήσεις που αφορούν τη διαδικασία ανάπτυξης, όπως εργαλεία για τον έλεγχο του κώδικα, τη συμπίεση, τη δοκιμή, και άλλα.

Το "package.json" είναι ένα σημαντικό αρχείο για τη διαχείριση του project σας, των εξαρτήσεών του και της διαδικασίας ανάπτυξης. Αν βέλετε να προσθέσετε νέες εξαρτήσεις, να εκτελέσετε σενάρια, ή να διαμορφώσετε τις ρυθμίσεις του project σας, συνήθως θα το κάνετε μέσω του "package.json" αρχείου.

8) Ζωτικό Κύκλο (Lifecycle): Το Vue.js παρέχει έναν ζωτικό κύκλο για τη διαχείριση του

κύκλου ζωής των components, όπως δημιουργία, ενημέρωση και καταστροφή. Αυτό επιτρέπει να εκτελέσετε κώδικα σε συγκεκριμένα σημεία του κύκλου ζωής.



Τα "lifecycle hooks" του Vue.js είναι στιγμές κατά τη διάρκεια του κύκλου ζωής των Vue component όπου μπορείτε να εφαρμόσετε λογική. Αυτά τα hooks επιτρέπουν στον προγραμματιστή να εκτελέσει κώδικα σε συγκεκριμένες φάσεις της διάρκειας ζωής του component, όπως δημιουργία, ενημέρωση και καταστροφή.

Εδώ είναι μια λίστα με τα κυριότερα lifecycle hooks του Vue.js και μια σύντομη περιγραφή της κάθε μίας:

1. beforeCreate: Καλείται πριν από τη δημιουργία του Vue component. Σε αυτό το σημείο, τα δεδομένα και τις μεθόδους δεν είναι ακόμη διαθέσιμα.

2. created: Καλείται μετά τη δημιουργία του Vue component. Τα δεδομένα είναι διαθέσιμα, αλλά το προτυπικό δεν έχει ακόμη ανακτηθεί.

3. beforeMount: Καλείται πριν από την ανακτηση (render) του προτυπικού. Σε αυτό το σημείο, το Vue component έχει δημιουργηθεί αλλά δεν έχει ακόμη ανακτηθεί στο DOM.

4. mounted: Καλείται μετά την ανακτηση (render) του προτυπικού και τοποθέτηση του Vue component στο DOM. Σε αυτό το σημείο, το component είναι ορατό στη σελίδα.

5. beforeUpdate: Καλείται πριν από την ανανέωση του προτυπικού λόγω αλλαγών στα δεδομένα του component.

6. updated: Καλείται μετά από την ανανέωση του προτυπικού λόγω αλλαγών στα δεδομένα του component.

7. beforeDestroy: Καλείται πριν από την καταστροφή του Vue component. Είναι μια καλή στιγμή για να καθαρίσετε τους πόρους που μπορεί να έχει καταχρησιμοποιήσει το component.

8. destroyed: Καλείται μετά από την καταστροφή του Vue component. Σε αυτό το σημείο, το component έχει αφαιρεθεί από το DOM.

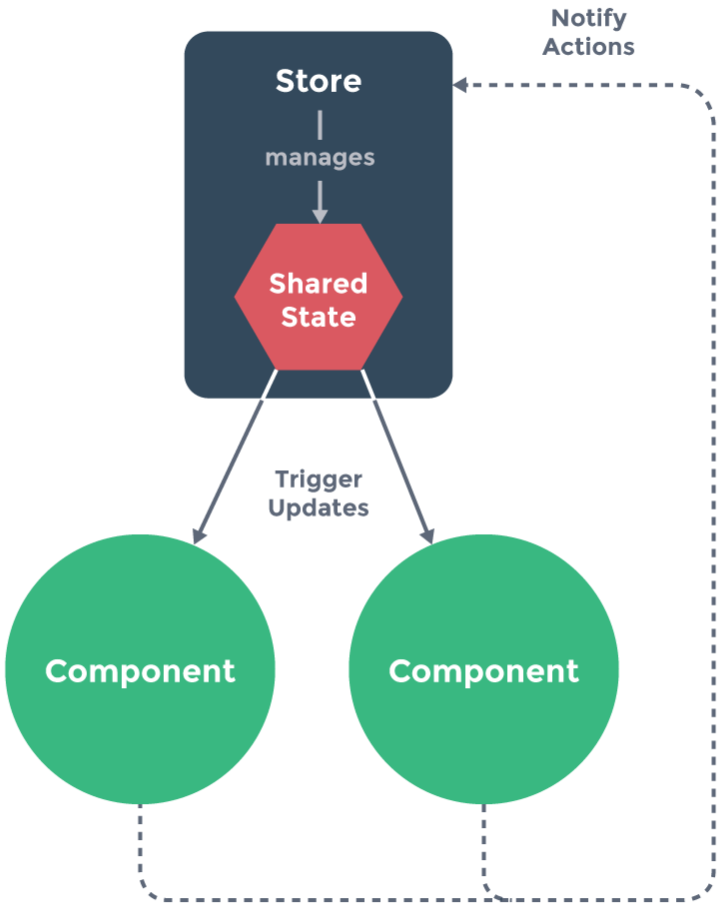
Κατά την διάρκεια της ανάπτυξης, μπορείτε να χρησιμοποιήσετε αυτά τα lifecycle hooks για να εκτελέσετε λογική όπως φόρτωση δεδομένων, αλλαγές στυλ, αποθήκευση δεδομένων, καθαρισμό και άλλες λειτουργίες που απαιτούν τον έλεγχο της κατάστασης του Vue component.

Το routing και το state management είναι δύο σημαντικά έννοιες στον κόσμο της ανάπτυξης εφαρμογών, και ιδιαίτερα όταν πρόκειται για εφαρμογές που χρησιμοποιούν το Vue.js.

Routing (Δρομολόγηση):
Η δρομολόγηση (routing) αναφέρεται στη διαχείριση της μετάβασης ανάμεσα σε διάφορες σελίδες ή προβολές μιας ενιαίας εφαρμογής. Αυτό είναι ιδιαίτερα χρήσιμο για τις ενιαίες εφαρμογές διαδικτύου (SPA), όπου η σελίδα δεν ανανεώνεται ολοκληρωτικά κατά την πλοήγηση.

Για το Vue.js, ένα από τα πιο δημοφιλή εργαλεία για τη διαχείριση της δρομολόγησης είναι το Vue Router. Το Vue Router μάς επιτρέπει να καθορίσετε διάφορους "δρόμους" (routes) στην εφαρμογή και να ορίσετε ποιες συνιστώσες (components) πρέπει να ανακατευθυνθούν κατά την πλοήγηση. Μπορείτε να διαμορφώσετε παραμέτρους και να εκτελέσετε πολλά άλλα δρομολόγησης συνδυάζοντας την Vue Router με το Vue.js.

State Management (Διαχείριση Κατάστασης):

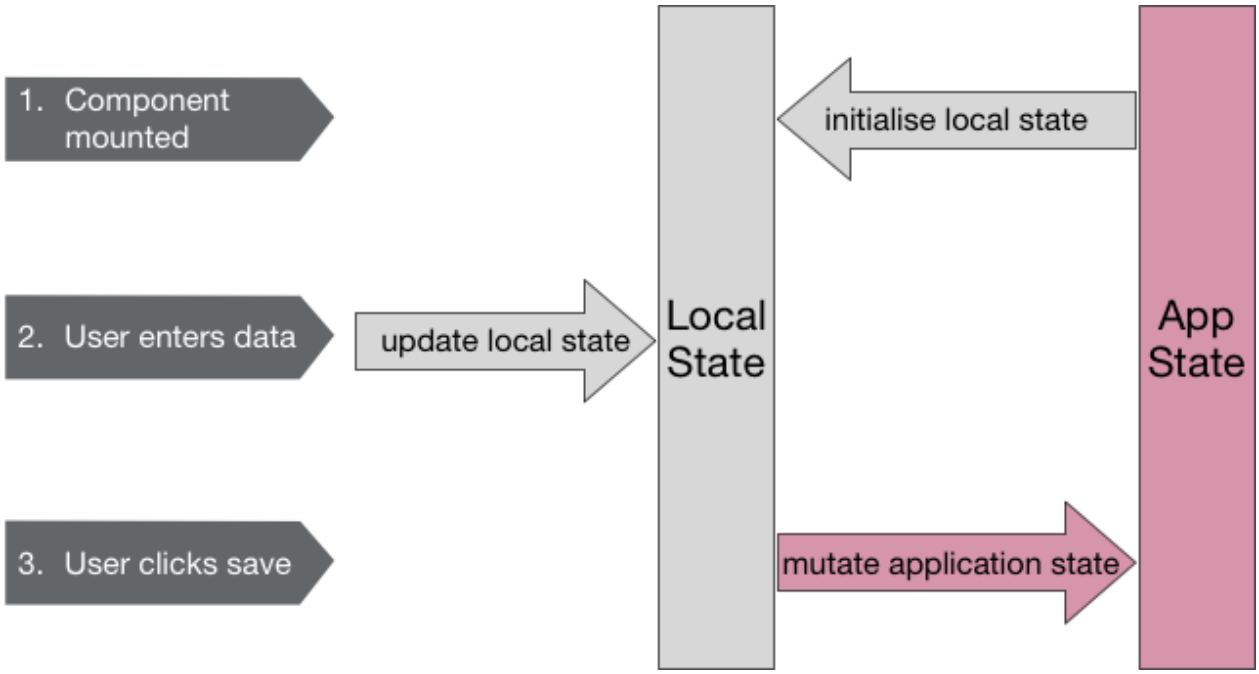


Η διαχείριση κατάστασης (state management) αναφέρεται στον τρόπο που διαχειρίζεστε την κατάσταση και τα δεδομένα που χρησιμοποιούνται στην εφαρμογή. Όσο η εφαρμογή γίνεται πιο πολύπλοκη, είναι σημαντικό να έχετε έναν καλό τρόπο για την κοινή χρήση και την ενημέρωση της κατάστασης ανάμεσα στα διάφορα components της εφαρμογής.

Στο Vue.js, κοινώς χρησιμοποιούνται δύο βασικά εργαλεία για τη διαχείριση της κατάστασης:

1) Vuex: Το Vuex είναι ένα κατάλληλο εργαλείο για τη διαχείριση της κατάστασης σε μεγάλες εφαρμογές. Παρέχει ένα κεντρικό αποθήκευσης (store) όπου μπορείτε να αποθηκεύσετε τα δεδομένα και να τα κοινοποιήσετε σε όλα τα components. Οποιαδήποτε αλλαγή στον αποθηκευτικό χώρο αυτόματα ενημερώνεται σε όλα τα μέρη της εφαρμογής που το χρησιμοποιούν.

2) Local State (Τοπική Κατάσταση): Για απλές εφαρμογές, μπορείτε να χρησιμοποιήσετε τοπική κατάσταση στα Vue components, δηλαδή να διαχειρίζεστε τα δεδομένα στο εσωτερικό κάθε component χωρίς να χρησιμοποιείτε εξωτερικά state management εργαλεία.



Η επιλογή μεταξύ Vuex ή τοπικής κατάστασης εξαρτάται από τις απαιτήσεις της εφαρμογής. Για απλές εφαρμογές, τοπική κατάσταση μπορεί να είναι αρκετή. Για πιο πολύπλοκες εφαρμογές, όπου χρειάζεστε διαχείριση δρομολόγησης και κοινή χρήση δεδομένων μεταξύ διαφόρων components, το Vue Router και το Vuex είναι πιο κατάλληλα όπως χρησιμοποιήθηκε και για αυτή την εφαρμογή.

Vue Router

The official router for Vue.js.

Ένα router (δρομολογητής) στον τομέα της ανάπτυξης εφαρμογών αναφέρεται σε ένα εργαλείο ή μηχανισμό που χρησιμοποιείται για τη διαχείριση της διάταξης των σελίδων ή των προβολών ενός ενιαίου ιστότοπου (συνήθως σε μια εφαρμογή διαδικτύου) καθώς και για τη διευθυνσιοδότηση των διάφορων URL και τη διαχείριση της πλοήγησης του χρήστη μεταξύ αυτών των σελίδων.

Συγκεκριμένα, στο πλαίσιο του Vue.js, το "Vue Router" είναι ένα πρόσθετο που χρησιμοποιείται για τη διαχείριση της δρομολόγησης (routing) σε εφαρμογές Vue.js. Το Vue Router επιτρέπει να διαχειρίζεστε τα δρομολόγια της εφαρμογής, προσθέτοντας ποιες προβολές (components) πρέπει να φορτωθούν όταν ο χρήστης αλλάζει URL.

Οι βασικές λειτουργίες του Vue Router περιλαμβάνουν:

1) Δήλωση Δρομολογίων: Ορίζετε τα δρομολόγια, που συσχετίζουν μια διαδρομή URL με μια συγκεκριμένη προβολή (component).

2) Διαχείριση Παραμέτρων: Τα δρομολόγια μπορούν να περιέχουν δυναμικές παραμέτρους που επιτρέπουν την περάτωση παραμέτρων στα URLs, όπως τα αναγνωριστικά των προϊόντων ή των χρηστών.

3) Ενσωμάτωση Προβολών: Καθορίζετε ποια προβολή (component) θα φορτώνεται για κάθε δρομολόγηση.

4) Πλοήγηση: Χρησιμοποιείτε διαφορετικές μεθόδους όπως router-link για την πλοήγηση ανάμεσα στις σελίδες και την αλλαγή του URL.

5) Επίβλεψη Αλλαγών στο URL: Το Vue Router παρακολουθεί τις αλλαγές στο URL και ανακατευθύνει τον χρήστη στην κατάλληλη προβολή αυτόματα. Είναι ιδιαίτερα χρήσιμο όταν θέλετε να δημιουργήσετε μια εφαρμογή με πολλές σελίδες ή προβολές, όπου ο χρήστης μπορεί να αλλάξει σελίδα χωρίς να φορτώσει εκ νέου την πλήρη σελίδα.

6) Το "conditional rendering" αναφέρεται στη δυνατότητα να ελέγχετε αν ένα στοιχείο (π.χ., ένα στοιχείο HTML) εμφανίζεται ή αποκρύπτεται δυναμικά στο προτυπικό (template) μιας εφαρμογής Vue.js με βάση μια συγκεκριμένη συνθήκη. Αυτό είναι πολύ χρήσιμο όταν θέλετε να διαχειριστείτε την προβολή στον χρήστη με βάση την κατάσταση της εφαρμογής.

Υπάρχουν διάφοροι τρόποι για το conditional rendering στο Vue.js:

1) v-if και v-else: Το v-if είναι ένας τρόπος για να ελέγξετε αν ένα στοιχείο πρέπει να εμφανίζεται ή όχι. Εάν η συνθήκη που δηλώνετε με το v-if είναι true, το στοιχείο εμφανίζεται. Σε αντίθετη περίπτωση, μπορείτε να χρησιμοποιήσετε το v-else για να ορίσετε τι θα εμφανίζεται όταν η συνθήκη είναι false.

Παραδείγμα:

```
<div>
  <p v-if="isVisible">Αυτό το κείμενο εμφανίζεται αν isVisible είναι true.</p>
  <p v-else>Αυτό το κείμενο εμφανίζεται αν isVisible είναι false.</p>
</div>
```

2) v-show: Το v-show είναι παρόμοιο με το v-if, αλλά αντί να αποκρύπτει εντελώς το στοιχείο, απλά το κρύβει με CSS display. Αυτό σημαίνει ότι το στοιχείο παραμένει στο DOM, αλλά μπορεί να είναι ορατό ή αόρατο ανάλογα με τη συνθήκη.

Παραδείγμα:

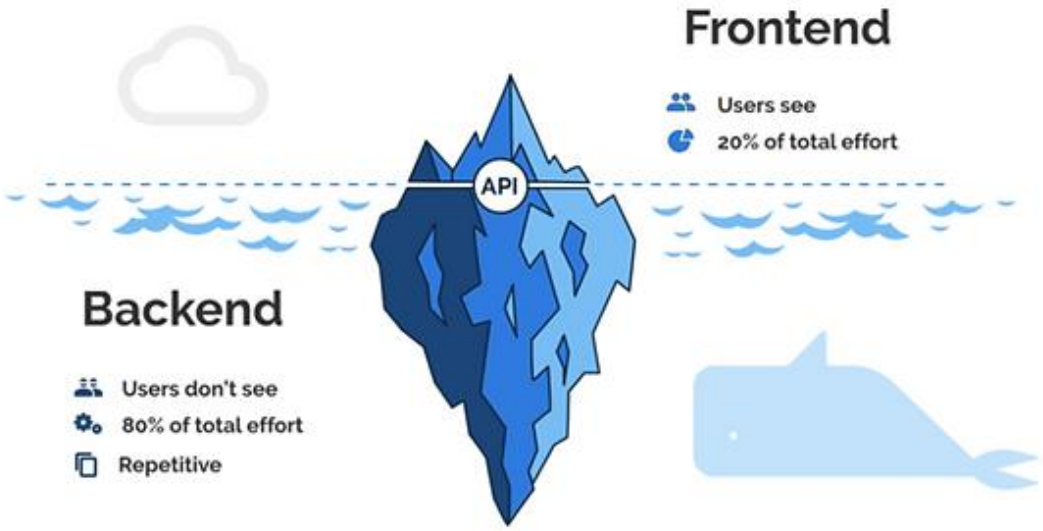
```
<div>
  <p v-show="isVisible">Αυτό το κείμενο εμφανίζεται αν isVisible είναι true.</p>
</div>
```

3) Συνθήκες μέσα στο Template: Μπορείτε να χρησιμοποιήσετε JavaScript συνθήκες μέσα στο προτυπικό του Vue component χρησιμοποιώντας τον v-if και τον v-else-if όπως θα κάνατε σε κανονικό JavaScript.

Παραδείγμα:

```
<div>
  <p v-if="condition === 'A'">Αυτό το κείμενο εμφανίζεται αν η συνθήκη είναι 'A'.</p>
  <p v-else-if="condition === 'B'">Αυτό το κείμενο εμφανίζεται αν η συνθήκη είναι
'B'.</p>
</div>
```

Σχετικά με το backend :



Το Vue.js είναι ένα JavaScript framework που συνήθως χρησιμοποιείται για την ανάπτυξη της κομμάτι του χρήστη (frontend) μιας εφαρμογής. Το Vue.js δεν ασχολείται με το backend, αλλά αντίθετα επικεντρώνεται στον χρήστη και τον τρόπο με τον οποίο αλληλεπιδρά με την εφαρμογή.

Για να λειτουργήσει μια πλήρης εφαρμογή, όπως μια εφαρμογή διαδικτύου (web application), χρειάζεστε τόσο ένα frontend με Vue.js όσο και ένα backend που θα παρέχει τις υπηρεσίες και τα δεδομένα που απαιτούνται από το frontend. Το backend είναι υπεύθυνο για την επεξεργασία των αιτημάτων του frontend, τη διαχείριση της βάσης δεδομένων, και την παροχή δεδομένων στο frontend.

Ο τρόπος που το Vue.js συνδέεται με το backend εξαρτάται από τις τεχνολογίες που χρησιμοποιείτε. Οι κοινοί τρόποι επικοινωνίας μεταξύ του frontend (Vue.js) και του backend περιλαμβάνουν:

1) HTTP requests: Το Vue.js μπορεί να κάνει αιτήσεις HTTP (π.χ., GET, POST, PUT, DELETE) προς το backend χρησιμοποιώντας το αντικείμενο axios, το fetch, ή άλλες βιβλιοθήκες. Το backend πρέπει να ακούει αυτές τις αιτήσεις, να τις επεξεργάζεται και να αποστέλλει τα απαραίτητα δεδομένα.

2) APIs: Συχνά, το backend παρέχει μια API (Προγραμματιστική Διεπαφή Εφαρμογής) που ο frontend χρησιμοποιεί για να ανταλλάξει δεδομένα. Ο frontend καλεί τις διάφορες διαδρομές της API για να ανακτήσει ή να αποστείλει δεδομένα.

3) Αυθεντικοποίηση (Authentication) και Εξουσιοδότηση (Authorization): Το backend χρησιμοποιεί τεχνικές αυθεντικοποίησης (π.χ., JWT, OAuth) για να ελέγξει τα δικαιώματα πρόσβασης του χρήστη. Το Vue.js μπορεί να συνεργαστεί με το backend για να ελέγξει και να διαχειριστεί τις διαπιστεύσεις και τις άδειες πρόσβασης των χρηστών.

Το Vue.js διευκολύνει την ανάπτυξη του frontend της εφαρμογής, αλλά είναι σημαντικό να κατανοήσετε ότι το backend είναι υπεύθυνο για την επικοινωνία με τη βάση δεδομένων, τη διαχείριση της λογικής εφαρμογής και την ασφάλεια. Ο συνδυασμός του Vue.js με ένα αξιόπιστο backend είναι ο τρόπος που δημιουργείτε μια πλήρη λειτουργική εφαρμογή.

3. Μεθοδολογία

Ανάλυση Απαιτήσεων:

Στην αρχική φάση του σχεδιασμού της εφαρμογής μας, προσδιορίσαμε μια σειρά απαιτήσεων που καλύπτουν τόσο τις λειτουργικές όσο και τις μη λειτουργικές πτυχές της. Αυτό μας βοήθησε να κατανοήσουμε καλύτερα τη λειτουργικότητα που πρέπει να προσφέρει η εφαρμογή και πώς θα σχεδιάσουμε την αρχιτεκτονική της.

Λειτουργικές Απαιτήσεις:

Δημιουργία Καταγραφής Εργασιών: Ο χρήστης πρέπει να μπορεί να δημιουργεί νέες εργασίες προσδιορίζοντας τίτλο, περιγραφή και ημερομηνία.
Λίστα Εργασιών: Οι εργασίες πρέπει να εμφανίζονται σε μια λίστα με τον τίτλο και την ημερομηνία τους.
Επεξεργασία Εργασιών: Ο χρήστης πρέπει να μπορεί να επεξεργάζεται τις υπάρχουσες εργασίες, αλλάζοντας τίτλο, περιγραφή και ημερομηνία.
Διαγραφή Εργασιών: Ο χρήστης πρέπει να μπορεί να διαγράφει εργασίες που δεν χρειάζεται πλέον.

Μη Λειτουργικές Απαιτήσεις:

Σχεδιασμός Χρήστη: Ο σχεδιασμός της εφαρμογής πρέπει να είναι φιλικός προς τον χρήστη και ευανάγνωστος.
Ασφάλεια Δεδομένων: Οι πληροφορίες των χρηστών πρέπει να αποθηκεύονται με ασφάλεια.
Απόκριση: Η εφαρμογή πρέπει να ανταποκρίνεται αυτόματα στις ενέργειες των χρηστών με ελάχιστο χρόνο αναμονής.

Περιπτώσεις Χρήσης:
Ο χρήστης μπορεί να :
Δημιουργεί μια νέα εργασία με τίτλο, περιγραφή και ημερομηνία.
Βλέπει τη λίστα με όλες τις εργασίες και τις λεπτομέρειές τους.
Επεξεργάζεται μια υπάρχουσα εργασία, αλλάζοντας τίτλο, περιγραφή ή ημερομηνία.
Διαγράφει μια εργασία που έχει ολοκληρωθεί ή δεν χρειάζεται πλέον.

Συνολικά, ο σχεδιασμός της εφαρμογής μας εστιάστηκε στην εκπλήρωση των παραπάνω απαιτήσεων. Η δημιουργία, προβολή, επεξεργασία και διαγραφή εργασιών αποτελούν τον πυρήνα της εφαρμογής μας.
Ενώ οι μη λειτουργικές απαιτήσεις όπως ο σχεδιασμός χρήστη και η ασφάλεια δεδομένων διασφαλίζουν μια θετική εμπειρία χρήστη και την προστασία των προσωπικών του δεδομένων.

4. Υλοποίηση

Αρχικά για να ξεκινήσει η υλοποίηση της εφαρμογής έπρεπε να γίνει εγκατάσταση της vue. Επέλεξα πάντα με την βοήθεια του vue js documentation να ξεκινήσω το project αυτό μέσω του vue cli τρέχωντας την εξής εφαρμογή.

npm install -g @vue/cli

Αυτή η εντολή θα εγκαταστήσει το Vue CLI στον υπολογιστή.

Βήμα 2: Δημιουργία ενός νέου project Vue Μόλις έχετε εγκαταστήσει το Vue CLI, μπορείτε να δημιουργήσετε ένα νέο project Vue.
Πληκτρολογήσα την παρακάτω εντολή στο terminal:

vue create my-pwa

Καθώς και :

vue add pwa

Αυτή η εντολή θα δημιουργήσει ένα νέο project Vue με το όνομα "my-pwa".
Κατά τη διάρκεια της διαδικασίας δημιουργίας, θα σας ζητηθεί να επιλέξετε μια προεπιλεγμένη επιλογή για τον τύπο εγκατάστασης.
Επιλέξτε την επιλογή "Manually select features" για να έχετε πλήρη έλεγχο επί των λειτουργιών που θέλετε να συμπεριλάβετε.

Βήμα 3: Επιλογή χαρακτηριστικών
Στη συνέχεια, θα εμφανιστεί μια λίστα με διαθέσιμα χαρακτηριστικά για το project Vue σας. Μπορείτε να επιλέξετε τα χαρακτηριστικά που θέλετε να συμπεριλάβετε χρησιμοποιώντας τα πλήκτρα "Space" για να επιλέξετε και "Enter" για να επιβεβαιώσετε την επιλογή σας.

Για το παράδειγμά μας, θα πρέπει να επιλέξετε τις εξής επιλογές:

Choose Vue version: 3.x
Babel: Y
TypeScript: N
Progressive Web App (PWA) Support: Y
Router: Y
CSS Pre-processors: None
Linter / Formatter: ESLint with error prevention only
Unit Testing: None
E2E Testing: None

Βήμα 4: Εγκατάσταση των dependencies
Μόλις έχετε επιλέξει τα χαρακτηριστικά, το Vue CLI θα δημιουργήσει το project σας και θα εγκαταστήσει τις απαραίτητες εξαρτήσεις. Αυτό μπορεί να διαρκέσει λίγα λεπτά, ανάλογα με την ταχύτητα της σύνδεσής σας.

Βήμα 5: Εκτέλεση του project

Μόλις ολοκληρωθεί η εγκατάσταση των εξαρτήσεων, μπορείτε να μεταβείτε στον φάκελο του project σας με την ακόλουθη εντολή:

cd my-pwa

Από εκεί, μπορείτε να εκκινήσετε τον ανάπτυξη του πρότυπου του Vue με την παρακάτω εντολή:

npm run serve

Αυτή η εντολή θα ξεκινήσει τον ενσωματωμένο διακομιστή ανάπτυξης του Vue CLI και θα σας παρέχει ένα URL όπου μπορείτε να δείτε την εφαρμογή σας στον περιηγητή σας.

Αφού κατάφερα και εγκατέστησα τόσο την vue js στο project μου το αμέσως επόμενο βήμα ήταν να διατηρήσω κάπου τον κώδικά μου.

Για την λύση αυτού του προβλήματος χρησιμοποιήθηκε το github.
Εφόσον υπήρχε ήδη το project τοπικά μέσω του terminal του ide (intellij) και με χρήση του git cli έγινε η αρχικοποίηση του project στο github.Πιο αναλυτικά :

Για την αρχικοιποίηση:

git init

Συνδέστε το τοπικό σας αποθετήριο με το αποθετήριο στο GitHub με την ακόλουθη εντολή:

git remote add origin <αντίγραφο-URL-του-project-σας-στο-GitHub>

Για να κάνουμε stage τις αλλαγές μας τρέχουμε την εντολή:

git add .

Για να κάνουμε commit τις αλλαγές μας τρέχουμε την εντολή:

git commit -m "Πρώτη δέσμευση"

Τέλος για να ανέβουν οι αλλαγές μας στο GitHub τρέχουμε την εντολή:

git push origin master

Για την διαχείριση των σελίδων της εφαρμογής χρησιμοποιήθηκε το εργαλείο vue router.

Μέσω του npm κάνουμε εγκατάσταση αυτό το εργαλείο τρέχοντας την εξής εντολή:

npm install vue-router

Έπειτα κάτω από τον φάκελο router θα υπάρχει το index js όπου θα ορίσουμε τις διαδρομές (routes) μας.

```
import { createRouter, createWebHistory } from 'vue-router'
import { getAuth, onAuthStateChanged } from 'firebase/auth';

const router = createRouter( options: {
    history: createWebHistory(),
    routes: [
        { path: '/', component: () => import('../views/HomeView.vue') },
        { path: '/login', component: () => import('../views/LoginView.vue') },
        { path: '/register', component: () => import('../views/RegisterView.vue') }
    ]
});
```

Επείτα πάμε στο αρχικό έγγραφο της εφαρμογής μας συνήθως το ‘main.js’ και εισάγουμε το router.js

main.js | index.js | HomeView.vue | LoginView.vue | .er

```
1  import { createApp } from 'vue'
2  import App from './App.vue'
3  import store from './store'
4  import Vuex from 'vuex'
5  import { register } from './registerServiceWorker'
6  import '@fortawesome/fontawesome-free/css/all.css'
7  import router from "@/router";
8
9  createApp(App)
10     .use(register)
11     .use(store)
12     .use(Vuex)
13     .use(router)
14     .mount( rootContainer: '#app')
15
```

Με αυτόν τον τρόπο, ο Vue Router θα είναι διαθέσιμος στην εφαρμογή σας και θα μπορείτε να χρησιμοποιήσετε τις διαδρομές (routes) για την πλοήγηση ανάμεσα στις σελίδες της εφαρμογής.

Αμέσως μετά έπρεπε να εμβαθύνουμε στο backend κομμάτι της εφαρμογής μας και πιο συγκεκριμένα στην βάση που επέλεξα το firebase.

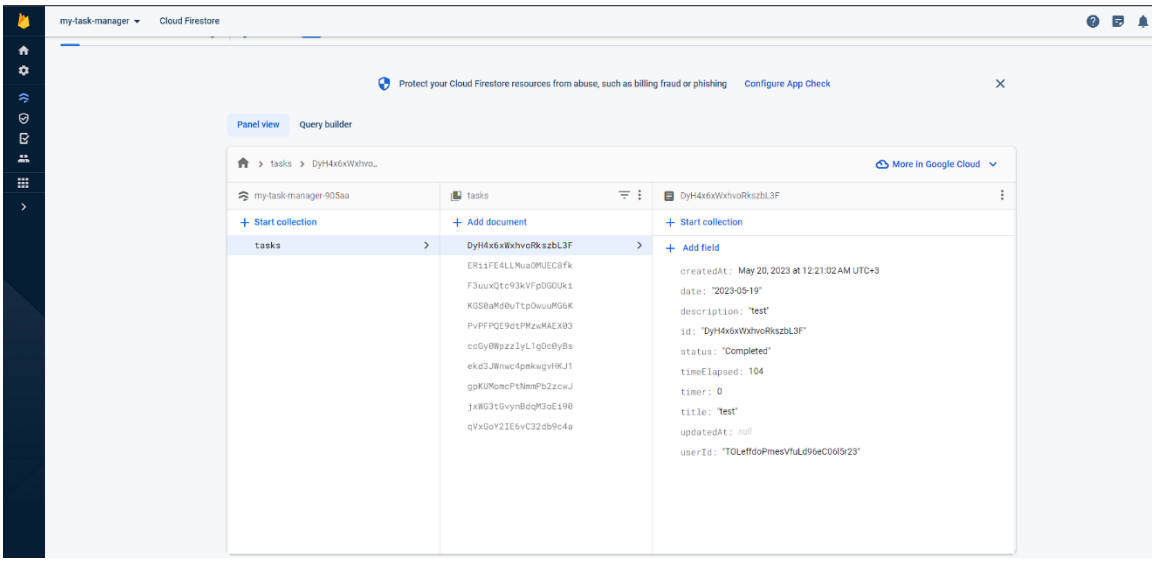
Firebase επιλέχθηκε ως η βάση δεδομένων για την εφαρμογή μου PWA για αρκετούς λόγους.

Καταρχήν, το Firebase προσφέρει μια ολοκληρωμένη πλατφόρμα ανάπτυξης που περιλαμβάνει όχι μόνο μια βάση δεδομένων, αλλά και πολλές άλλες υπηρεσίες όπως ελεγχόμενη πρόσβαση χρηστών, αποθήκευση αρχείων, ελεγκτές λειτουργιών και πολλά άλλα. Αυτό μου επέτρεψε να αναπτύξω μια πλήρη και ολοκληρωμένη εφαρμογή χρησιμοποιώντας μια μόνο πλατφόρμα.

Επιπλέον, το Firebase παρέχει μια πολύ φιλική προς τους προγραμματιστές αναπτυξιακή εμπειρία. Η ενσωματωμένη λειτουργικότητα του Firebase μειώνει τον αριθμό των απαιτούμενων κωδικοποιήσεων και διευκολύνει την ανάπτυξη και τη συντήρηση της εφαρμογής.

Επιπροσθέτως, το Firebase προσφέρει έναν υψηλό βαθμό ασφάλειας για την αποθήκευση των δεδομένων μου. Οι υπηρεσίες του Firebase προστατεύονται από ισχυρά μέτρα ασφαλείας και παρέχουν επίσης εργαλεία για τον έλεγχο της πρόσβασης των χρηστών στα δεδομένα.

Τέλος, το Firebase παρέχει έναν εύχρηστο πίνακα ελέγχου (dashboard) που επιτρέπει τη διαχείριση και την εποπτεία της εφαρμογής μου. Μπορώ να παρακολουθώ τη χρήση της εφαρμογής, να παρακολουθώ τα δεδομένα και να εκτελώ data debugging and management.



Συνολικά, η επιλογή της Firebase ως βάσης δεδομένων για το PWA μου ήταν μια λογική επιλογή, παρέχοντας μια ολοκληρωμένη και ασφαλή λύση για την ανάπτυξη και τη λειτουργία της εφαρμογής μου.

Για να δημιουργήσετε μια βάση δεδομένων στο Firebase και να τη συνδέσετε με την εφαρμογή Vue.js σας, ακολουθήστε τα παρακάτω βήματα:

Εγγραφή στο Firebase: Εάν δεν έχετε ήδη λογαριασμό Firebase, επισκεφθείτε την ιστοσελίδα του Firebase (https://firebase.google.com/) και κάντε εγγραφή με τον λογαριασμό Google σας.

Δημιουργία νέου έργου: Στο Firebase Console, δημιουργήστε ένα νέο έργο πατώντας το κουμπί "Add project" ή "Create a project".

Προσθήκη της εφαρμογής Vue.js σας: Μετά τη δημιουργία του έργου, προσθέστε την εφαρμογή Vue.js σας στο Firebase Console. Κάντε κλικ στο κουμπί "Add app" ή "Add Firebase to your

web app". Δώστε ένα όνομα στην εφαρμογή σας και αποθηκεύστε τις πληροφορίες της εφαρμογής (API Key, Auth Domain κλπ.) που θα χρειαστείτε αργότερα.

Δημιουργία βάσης δεδομένων Firestore: Στο Firebase Console, μεταβείτε στο μενού "Firestore Database" και δημιουργήστε μια νέα βάση δεδομένων Firestore. Ακολουθήστε τις οδηγίες για να δημιουργήσετε τη βάση δεδομένων.

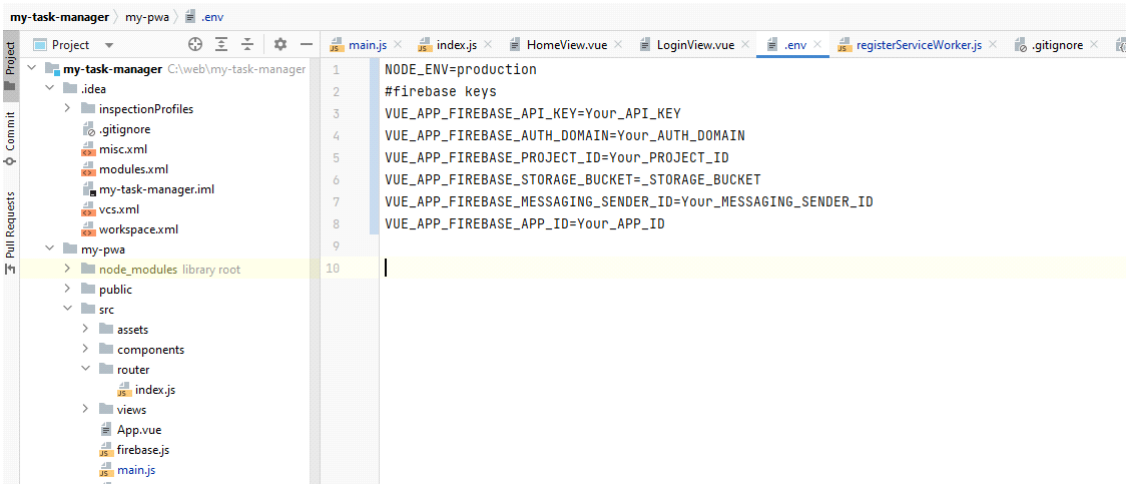
Σύνδεση της εφαρμογής Vue.js με το Firebase: Στον κώδικα της εφαρμογής Vue.js σας, εισάγετε το πακέτο Firebase και συνδεθείτε με το Firebase χρησιμοποιώντας τις πληροφορίες της εφαρμογής που αποθηκεύσατε στο βήμα 3. Αυτό μπορεί να γίνει συνήθως στο αρχείο main.js ή σε ένα αρχείο firebase.js που εισάγεται στην εφαρμογή σας.

Παράδειγμα κώδικα για τη σύνδεση με το Firebase:

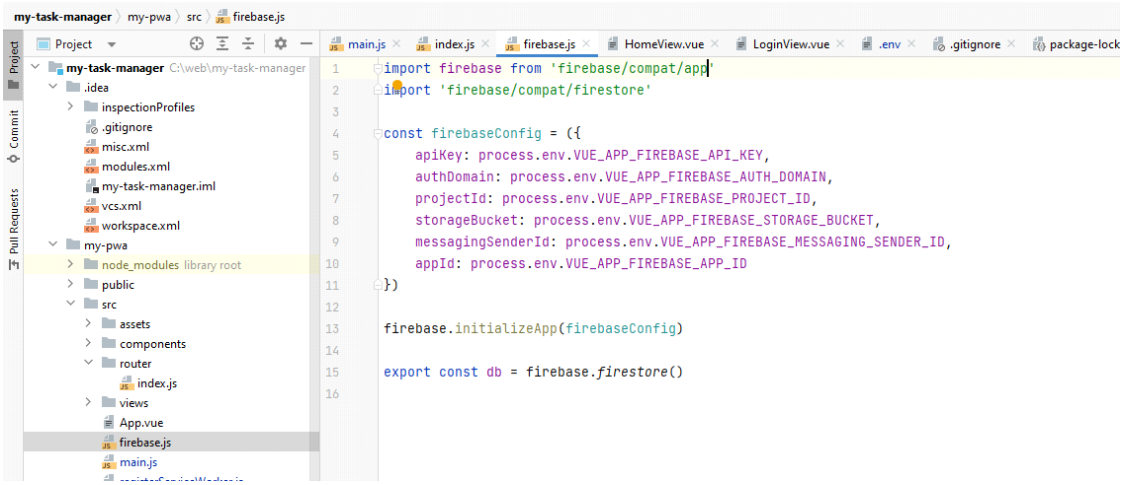
Ανοίγουμε ένα terminal στην τοποθεσία που βρίσκεται το project μας και τρέχουμε την εξής εντολή :

npm install firebase

Αμέσως μετά πρόσθεσα τα κλειδιά της βάσης μου σε ένα .env αρχείο όπου θα καλέσω δυναμικά αργότερα για να συνδεθώ στην βάση μου.



Αμέσως μετά δημιούργησα ένα αρχείο με όνομα firebase.js κάτω από τον φάκελο src όπου καλούσα τις μεταβλητές αυτές.



Μετά τη σύνδεση με το Firebase, μπορείτε να χρησιμοποιήσετε τη βάση δεδομένων Firestore για να αποθηκεύσετε και να ανακτήσετε δεδομένα από την εφαρμογή σας. Μπορείτε να δημιουργήσετε, να αναζητήσετε και να ενημερώσετε εγγραφές στη βάση δεδομένων χρησιμοποιώντας τις μεθόδους που παρέχονται από το Firestore SDK του Firebase.

Είναι σημαντικό να κατανοήσετε τη δομή της βάσης δεδομένων Firestore και τις δυνατότητες του Firebase SDK για τη διαχείριση των δεδομένων σας. Μπορείτε να ανατρέξετε στην τεκμηρίωση του Firebase για περισσότερες πληροφορίες και παραδείγματα.

Με τη σύνδεση της εφαρμογής Vue.js με το Firebase, μπορείτε να αποκτήσετε πρόσβαση σε πολλές δυνατότητες, όπως τη διαχείριση χρηστών, την αποθήκευση και την ανάκτηση δεδομένων, την εκτέλεση λειτουργιών (functions) στην πλευρά του διακομιστή, και πολλά άλλα.

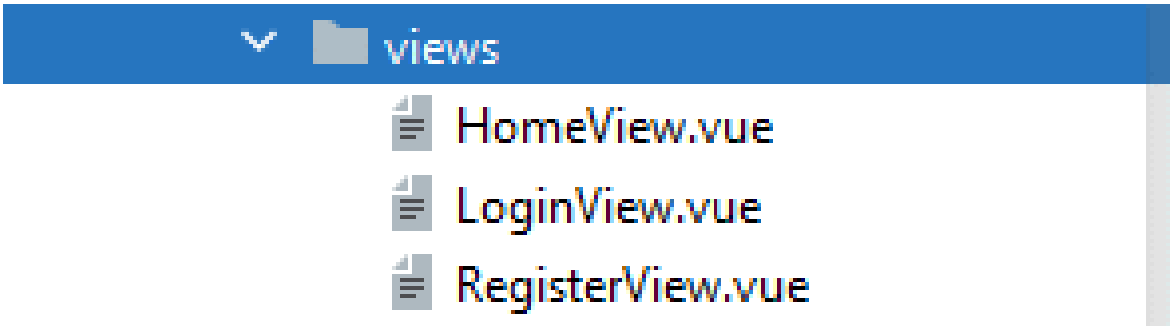
Αμέσως μετά την εγκατάσταση του Vue Router και την ανάθεση των διαδρομών (routes) στο πρότυπο μας, προχωρήσαμε στη δημιουργία των αντίστοιχων views. Κάθε view αντιπροσωπεύει μια σελίδα στην εφαρμογή μας και παρουσιάζει την αντίστοιχη λειτουργικότητα στον χρήστη.

Συγκεκριμένα, δημιουργήσαμε τρία views με τα ονόματα HomeView.vue, LoginView.vue και RegisterView.vue. Το HomeView.vue αναπαριστά την αρχική σελίδα της εφαρμογής μας, όπου ο χρήστης μπορεί να διαχειριστεί τα task του. Το LoginView.vue παρέχει τη λειτουργία σύνδεσης για τους χρήστες, ενώ το RegisterView.vue παρέχει τη λειτουργία εγγραφής για νέους χρήστες.

Για τη δημιουργία αυτών των views, δημιουργήσαμε αντίστοιχα αρχεία με τα παραπάνω ονόματα και προσθέσαμε τον απαιτούμενο κώδικα HTML και JavaScript για τη σωστή απεικόνιση και λειτουργία των σελίδων. Ο κώδικας αυτός περιλαμβάνει τη δομή της σελίδας, την επεξεργασία των δεδομένων, την επικοινωνία με τη βάση δεδομένων και άλλες απαραίτητες λειτουργίες.

Η διαδικασία αυτή μας επιτρέπει να οργανώσουμε την εφαρμογή μας σε διάφορες σελίδες και να

προσφέρουμε στον χρήστη μια πλοήγηση μεταξύ αυτών των σελίδων μέσω του Vue Router. Έτσι, ο χρήστης μπορεί να αλληλοεπιδράσει με την εφαρμογή μας μέσω διαφορετικών προβολών ανάλογα με τις ανάγκες και τις ενέργειές του.



Μετά από τη δημιουργία των views, δημιουργήθηκε το αρχείο store.js για τη διαχείριση της κατάστασης της εφαρμογής. Σε αυτό το αρχείο, περιέχονται οι διάφορες μεθόδοι (mutations, actions, getters) που επιτρέπουν την επεξεργασία και την ανάκτηση δεδομένων από τη βάση δεδομένων Firebase.

Οι βασικές λειτουργίες που υλοποιούνται στο store.js είναι η προσθήκη νέων task, η επεξεργασία και διαγραφή υπάρχοντων task, καθώς και η ανάκτηση των task από τη βάση δεδομένων. Επίσης, υπάρχουν μεθόδοι για την ενημέρωση της κατάστασης των task ως ολοκληρωμένα ή μη ολοκληρωμένα.

Επιπλέον, δημιουργήθηκε η μέθοδος fetchCurrentUser για την ανάκτηση των στοιχείων του τρέχοντος χρήστη, όπως το email, το login name και το user ID. Αυτά τα δεδομένα αποθηκεύονται στο state με τη βοήθεια της μεθόδου setCurrentUser.

Τέλος, μέσω των getters, μπορούμε να ανακτήσουμε τις λίστες των task και τα στοιχεία του τρέχοντος χρήστη από το state.

Αυτός ο κώδικας επιτρέπει την αλληλεπίδραση με τη βάση δεδομένων Firebase και την ανάκτηση και ενημέρωση των δεδομένων της εφαρμογής μας.

Στο αρχείο store.js, υπάρχουν διάφορες μέθοδοι (mutations, actions, getters) που επιτρέπουν την επεξεργασία και την ανάκτηση δεδομένων από τη βάση δεδομένων Firebase. Ας αναλύσουμε τη λειτουργία της κάθε μεθόδου αναλυτικά:

Mutations:

addTask: Αυτή η μέθοδος αναλαμβάνει την προσθήκη ενός νέου task στη βάση δεδομένων. Ελέγχει την επιτυχία της προσθήκης και επιστρέφει το νέο task.

editTask: Αυτή η μέθοδος αναλαμβάνει την επεξεργασία ενός υπάρχοντος task στη βάση δεδομένων. Ελέγχει την επιτυχία της επεξεργασίας και ενημερώνει τη λίστα tasks στο state.

deleteTask: Αυτή η μέθοδος αναλαμβάνει τη διαγραφή ενός υπάρχοντος task από τη βάση δεδομένων. Ελέγχει την επιτυχία της διαγραφής και αφαιρεί το task από τη λίστα tasks στο state.

completeTask: Αυτή η μέθοδος αναλαμβάνει την ενημέρωση του status ενός task ως "Completed" στη βάση δεδομένων. Ελέγχει την επιτυχία της ενημέρωσης και ενημερώνει το task στο state.

incompleteTask: Αυτή η μέθοδος αναλαμβάνει την ενημέρωση του status ενός task ως "Pending" στη βάση δεδομένων. Ελέγχει την επιτυχία της ενημέρωσης και ενημερώνει το task στο state.

setTasks: Αυτή η μέθοδος αναλαμβάνει την ενημέρωση της λίστας tasks στο state με τα νέα tasks που ανακτήθηκαν από τη βάση δεδομένων.

setCurrentUser: Αυτή η μέθοδος αναλαμβάνει την ενημέρωση των δεδομένων του τρέχοντος χρήστη στο state. Αυτά τα δεδομένα περιλαμβάνουν το email, το όνομα σύνδεσης, το ID χρήστη και τα δεδομένα του χρήστη.

Actions:

addTask: Αυτή η μέθοδος απλώς καλεί τη mutation addTask για να προσθέσει ένα νέο task.

editTask: Αυτή η μέθοδος απλώς καλεί τη mutation editTask για να επεξεργαστεί ένα υπάρχον task.

deleteTask: Αυτή η μέθοδος απλώς καλεί τη mutation deleteTask για να διαγράψει ένα υπάρχον task.

fetchTasks: Αυτή η μέθοδος ανακτά τα tasks από τη βάση δεδομένων και ενημερώνει τη λίστα tasks στο state.

completeTask: Αυτή η μέθοδος απλώς καλεί τη mutation completeTask για να σημειώσει ένα task ως "Completed".

incompleteTask: Αυτή η μέθοδος απλώς καλεί τη mutation incompleteTask για να σημειώσει ένα task ως "Pending".

fetchCurrentUser: Αυτή η μέθοδος ανακτά τα δεδομένα του τρέχοντος χρήστη από τη βάση δεδομένων και ενημερώνει τα σχετικά δεδομένα στο state.

Getters:

tasks: Αυτός ο getter επιστρέφει τη λίστα tasks από το state.

userData: Αυτός ο getter επιστρέφει τα δεδομένα του τρέχοντος χρήστη από το state.

Αυτές οι μέθοδοι επιτρέπουν τη διαχείριση και την ανάκτηση των tasks και των δεδομένων του χρήστη από τη βάση δεδομένων Firebase και την ενημέρωση του state του Vue.js εφαρμογής. Οι mutations χρησιμοποιούνται για την ενημέρωση του state, ενώ οι actions είναι υπεύθυνες για την κλήση των mutations και των απαιτούμενων ενεργειών. Οι getters χρησιμοποιούνται για την ανάκτηση των δεδομένων από το state.

Αφού δημιούργησα τον αρχείο store.js και διαμόρφωσα τις βασικές λειτουργίες για τη διαχείριση των tasks και των δεδομένων του χρήστη στο state, προχώρησα στη δημιουργία των αντίστοιχων components.

TaskForm.vue: Αυτό το component αντιπροσωπεύει ένα φόρμα για την προσθήκη νέων tasks. Ο χρήστης μπορεί να εισάγει τίτλο, περιγραφή και άλλα στοιχεία για το task και να πατήσει το κατάλληλο κουμπί για να προσθέσει το task στη λίστα.

TaskEdit.vue: Αυτό το component επιτρέπει στον χρήστη να επεξεργαστεί ορισμένα δεδομένα του task. Προσφέρει μια φόρμα όπου ο χρήστης μπορεί να τροποποιήσει τον τίτλο, την περιγραφή και άλλα στοιχεία του task και να αποθηκεύσει τις αλλαγές.

TaskList.vue: Αυτό το component εμφανίζει τη λίστα των tasks, χωρίζοντάς τα σε δύο κατηγορίες: "Completed" (ολοκληρωμένα) και "Pending" (εκκρεμή). Τα tasks παρουσιάζονται σε μια λίστα με τις αντίστοιχες πληροφορίες, όπως ο τίτλος, η περιγραφή και η κατάσταση.

TaskModal.vue: Αυτό το component αντιπροσωπεύει ένα Modal παράθυρο που εμφανίζει το TaskEdit.vue component για την επεξεργασία ενός task. Όταν ο χρήστης επιλέγει ένα task για επεξεργασία, αυτό το Modal εμφανίζεται και του επιτρέπει να τροποποιήσει τα απαραίτητα δεδομένα.

Αυτά τα components συμβάλλουν στη διευκόλυνση της διαχείρισης των tasks από τους χρήστες. Ο TaskForm.vue επιτρέπει την προσθήκη νέων tasks, ενώ ο TaskEdit.vue επιτρέπει την επεξεργασία των υπαρχόντων tasks. Ο TaskList.vue εμφανίζει τη λίστα των tasks με βάση την κατάστασή τους, και το TaskModal.vue προσφέρει ένα παράθυρο επεξεργασίας για ευκολότερη πρόσβαση στην επεξεργασία των tasks.

Πιο συγκεκριμένα :

Το component TaskForm αντιπροσωπεύει μια φόρμα που χρησιμοποιείται για τη δημιουργία νέων tasks στην εφαρμογή. Ας αναλύσουμε τον κώδικα του component για να κατανοήσουμε τη λειτουργία του:

Στην ενότητα template, δημιουργούμε μια φόρμα με διάφορα πεδία εισαγωγής όπως τίτλο, περιγραφή και ημερομηνία. Η φόρμα έχει ένα κουμπί "Save Task" που εκτελεί τη μέθοδο submitTask όταν πατηθεί.

Στην ενότητα script, ξεκινάμε με τη δήλωση των data properties του component. Έχουμε τις μεταβλητές taskTitle, taskDescription και taskDate που χρησιμοποιούνται για την αποθήκευση των εισόδων του χρήστη. Υπάρχουν επίσης οι μεταβλητές timer, timeElapsed και status που αρχικοποιούνται με καθορισμένες τιμές. Η μέθοδος data() επιστρέφει ένα αντικείμενο με αυτές τις μεταβλητές και τις αρχικές τιμές τους.

Στην ενότητα methods, έχουμε τη μέθοδο submitTask που εκτελείται όταν ο χρήστης υποβάλλει τη φόρμα. Μέσα σε αυτήν τη μέθοδο, δημιουργείται ένα νέο task από τις τιμές που έδωσε ο χρήστης και αποστέλλεται μέσω της δράσης 'addTask' στο store. Τέλος, οι μεταβλητές taskTitle, taskDescription και taskDate αδειάζουν για την επόμενη εισαγωγή του χρήστη.

Στην ενότητα computed, χρησιμοποιούμε τα mapGetters και mapState για να αποκτήσουμε πρόσβαση στις μεταβλητές userData και tasks από το store.

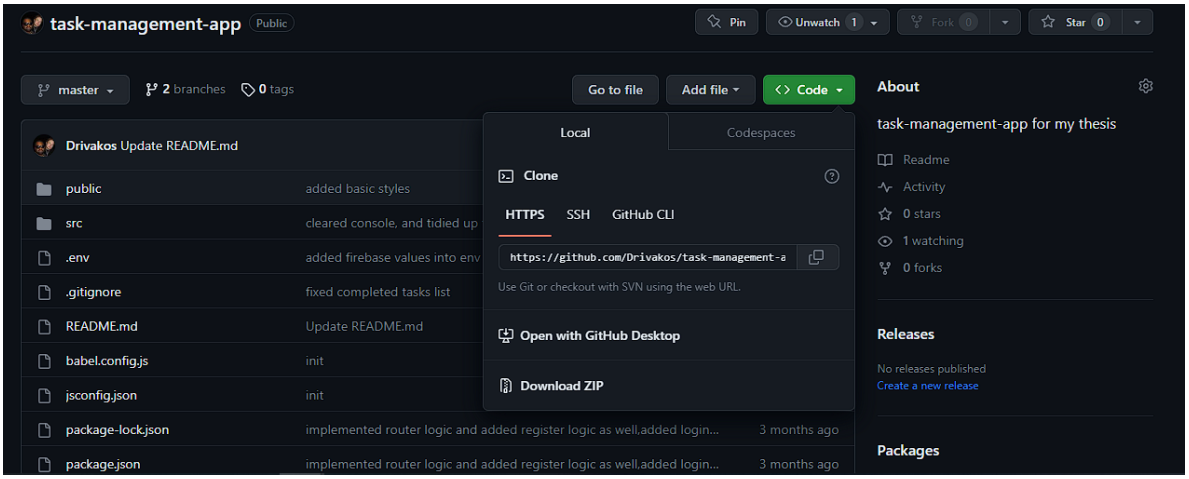
Στην ενότητα style, ορίζουμε την εμφάνιση του TaskForm μέσω CSS. Το component έχει ένα χαρακτηριστικό στυλ που το καθιστά ευδιάκριτο στην εφαρμογή.

Με αυτόν τον τρόπο, το TaskForm.vue επιτρέπει στον χρήστη να δημιουργεί νέα tasks προσθέτοντας τίτλο, περιγραφή και ημερομηνία. Τα δεδομένα αποστέλλονται στο store για να αποθηκευτούν και να γίνουν διαθέσιμα στην εφαρμογή.

5. Αξιολόγηση

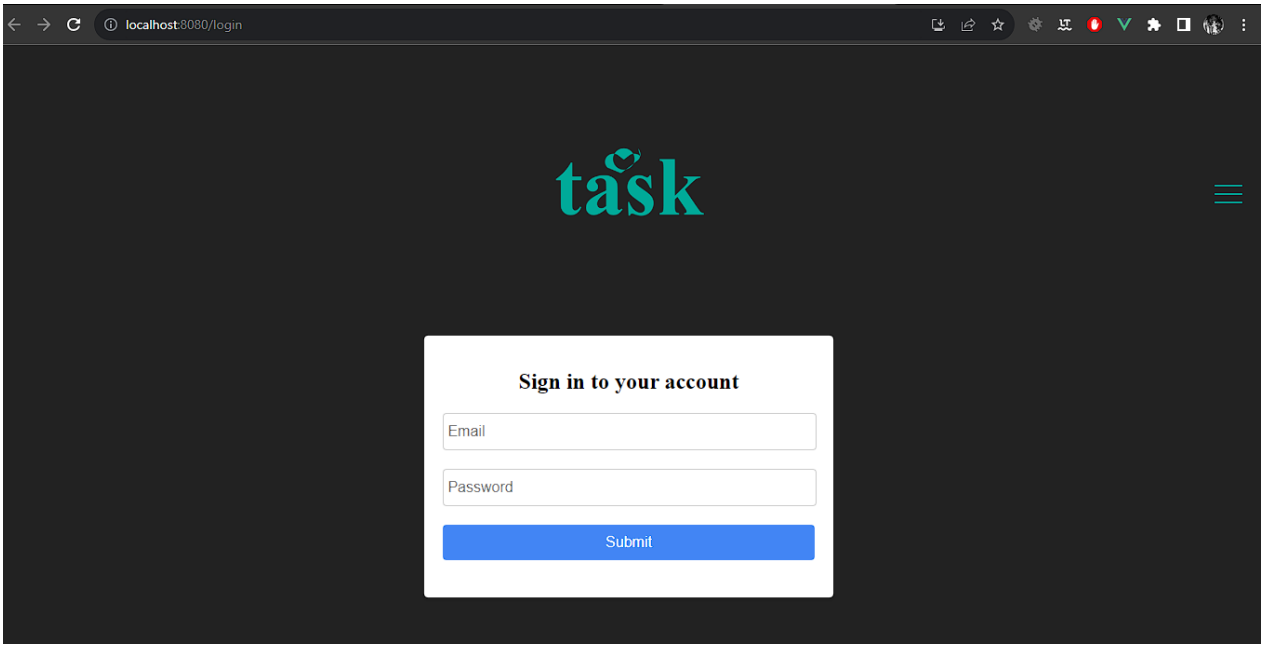
Για να αξιολογήσουμε την εφαρμογή μας θα την χρησιμοποιήσουμε για να μας βοηθήσει με την ανάπτυξη αυτού του ίδιου του εγγράφου.

Αυτή την στιγμή η εφαρμογή δεν βρίσκεται hosted σε κάποιο server οπότε θα πρέπει να την κατεβάσουμε από το github στο παρακάτω link. (Η εφαρμογή είναι public για ευνόητους λόγους)

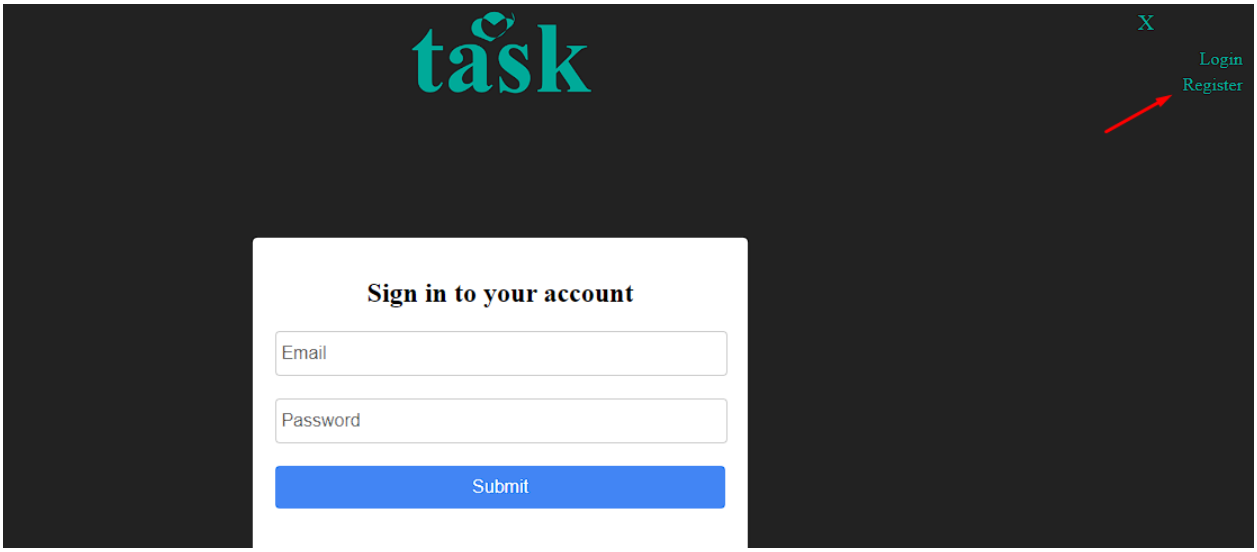


Από εκεί πηγαίνουμε και το κατεβάζουμε στο ide της επιλογής μας , διαβάζουμε το readme.md αρχείο και ακολουθούμε τα βήματα για να τρέξουμε την εφαρμογή μας.

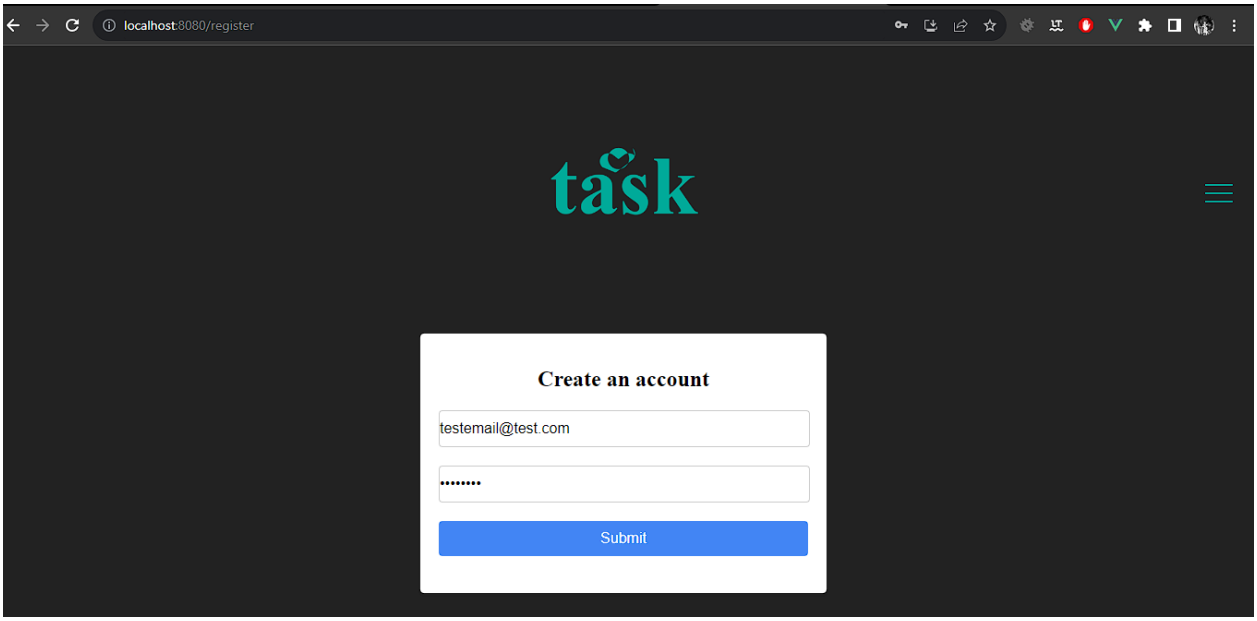
Μόλις ολοκληρώσουμε την διαδικασία αυτή πλέον θα έχουμε την εφαρμογή μας στο browser μας και θα δούμε την οθόνη του Login.



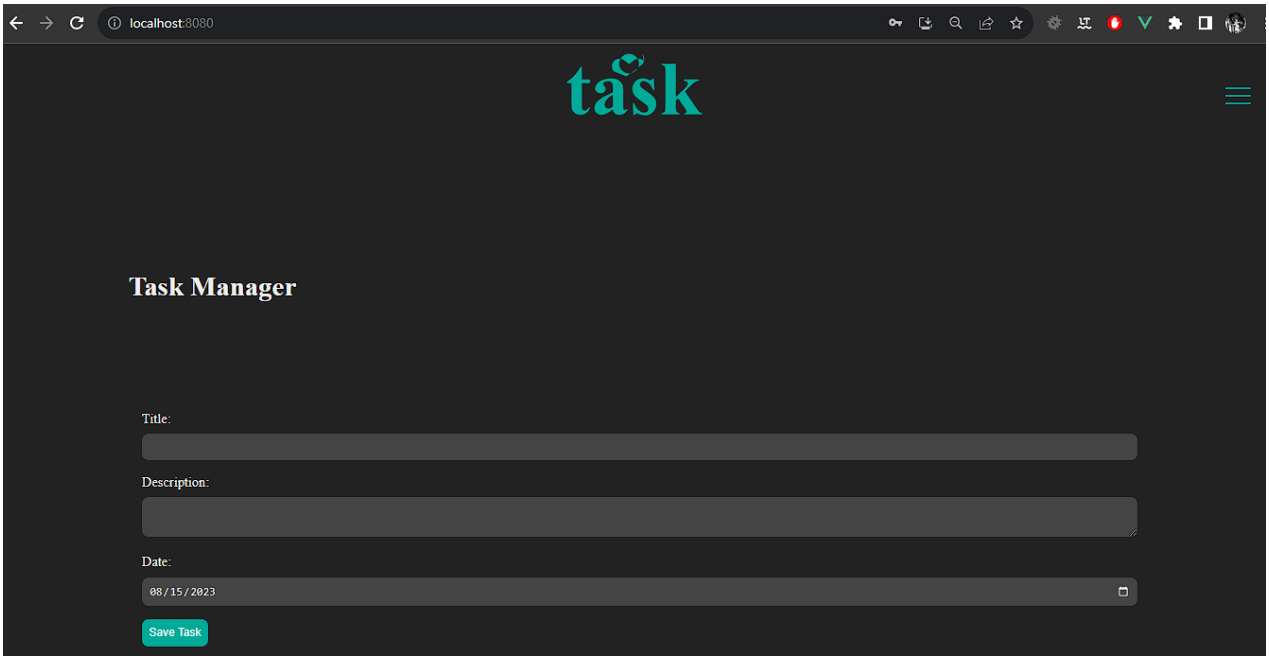
Σε περίπτωση που δεν έχουμε λογαριασμό μπορούμε να πατήσουμε το burger menu που βρίσκεται στα δεξία και να πάμε στο Register page ώστε να φτιάξουμε τον λογαριασμό μας.



Από εκεί πηγαίνουμε στο Register page το οποίο είναι αντίστοιχο οπτικά με το Login, πιο συγκεκριμένα :

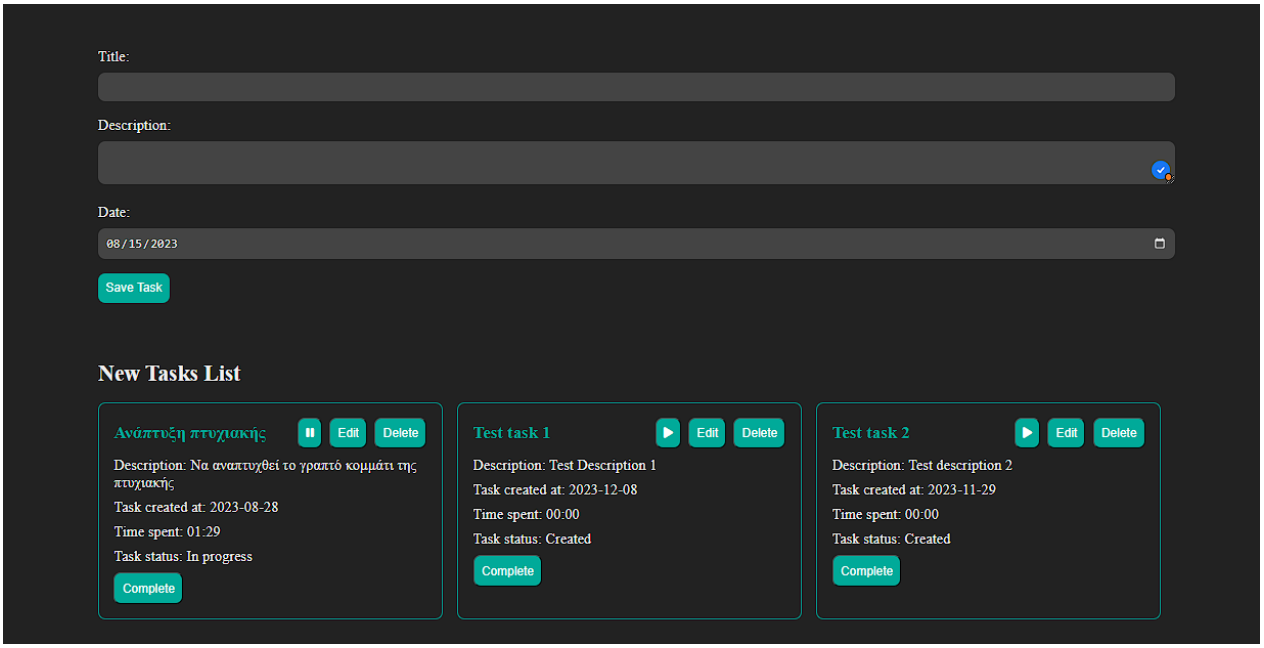


Πατάμε submit και προχωράμε στο dashboard της εφαρμογής.

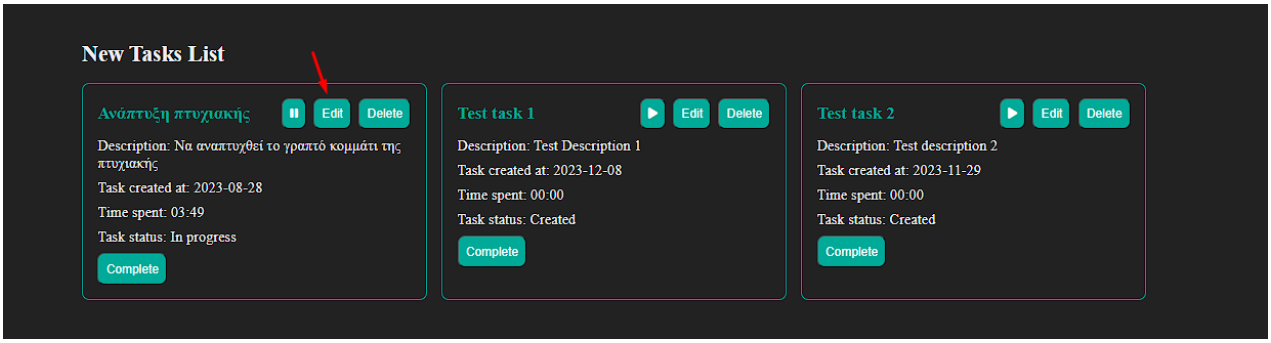


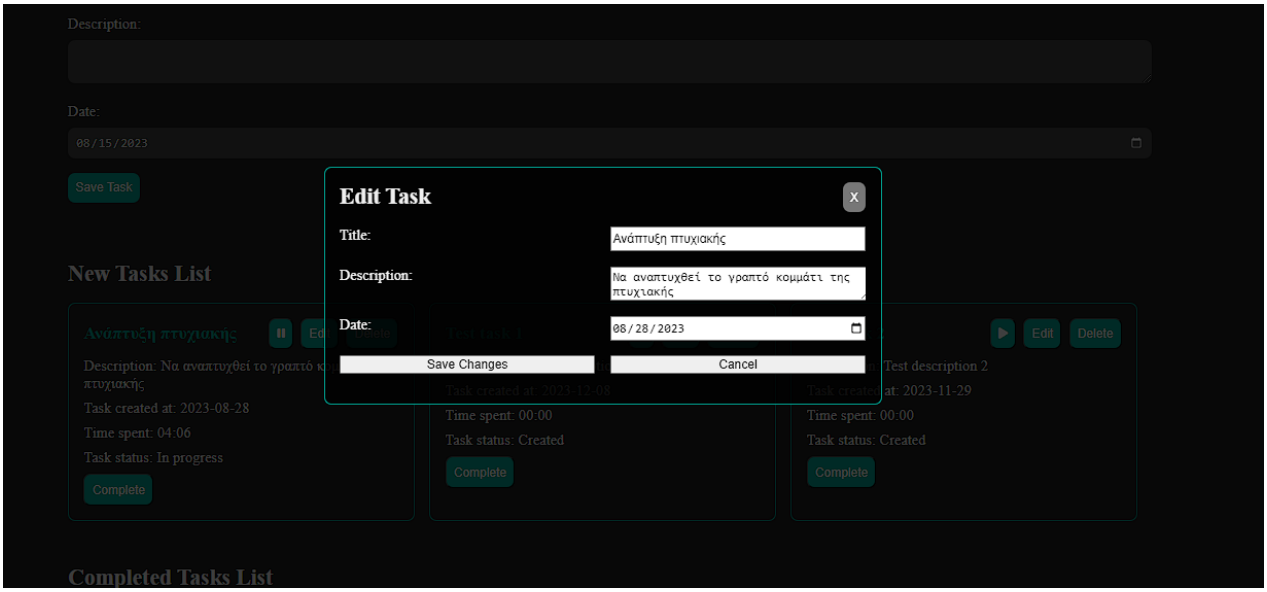
Εκεί θα δούμε την φόρμα για να μπορέσουμε να φτιάξουμε τα task που θέλουμε και μας δίνεται η δυνατότητα να δώσουμε ένα τίτλο ένα description και ένα deadline για το task μας.

Όταν συμπληρώσουμε τα στοιχεία που επιθυμούμε για το task πατάμε το Save Task button και αυτό μεταφέρει το task μας στο New tasks list που βρίσκεται ακριβώς από κάτω.



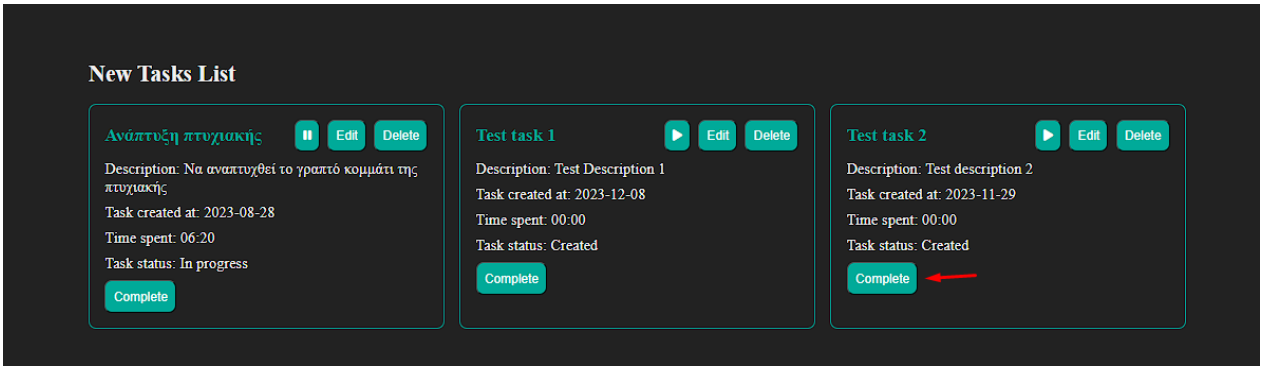
Από εκεί μας δίνεται η δυνατότητα να χρονομετρήσουμε το πόσο χρόνο μας πήρε ένα συγκεκριμένο task, επίσης μπορούμε να επεξεργαστούμε το συγκεκριμένο task και να αλλάξουμε κάποια στοιχεία όπως τίτλο, περιγραφή, και date.

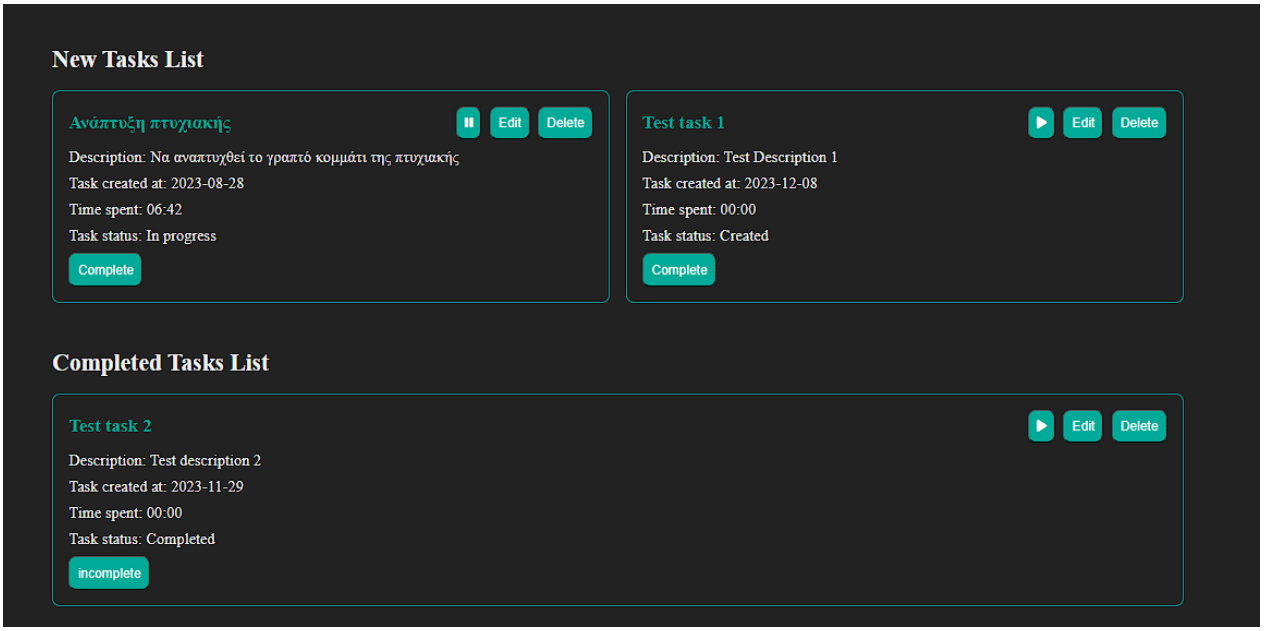




Από εκεί μπορούμε να ακυρώσουμε τις αλλαγές ή να τις αποθηκεύσουμε.

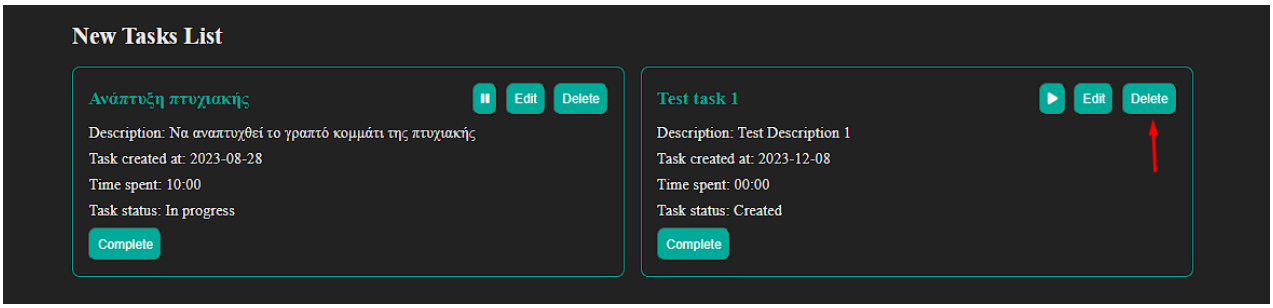
Ακόμα μπορούμε να ολοκληρώσουμε κάποιο task το οποίο μόλις πατήσουμε το complete button θα πάει στην λίστα με τα completed tasks.



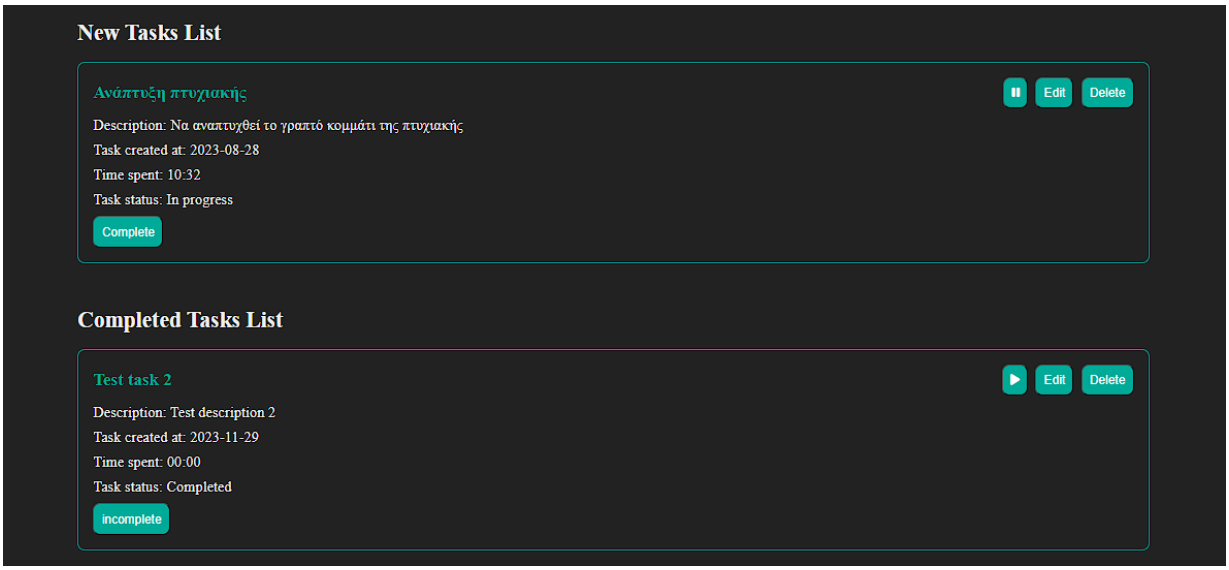


Από εκεί μπορούμε να βλέπουμε όλα τα completed tasks, να επεξεργαστούμε δεδομένα του task αλλά και να γυρίσουμε την κατάσταση του task από complete σε incomplete ώστε να συνεχίσουμε να δουλεύουμε σε αυτό σε περίπτωση που χρειαστεί.

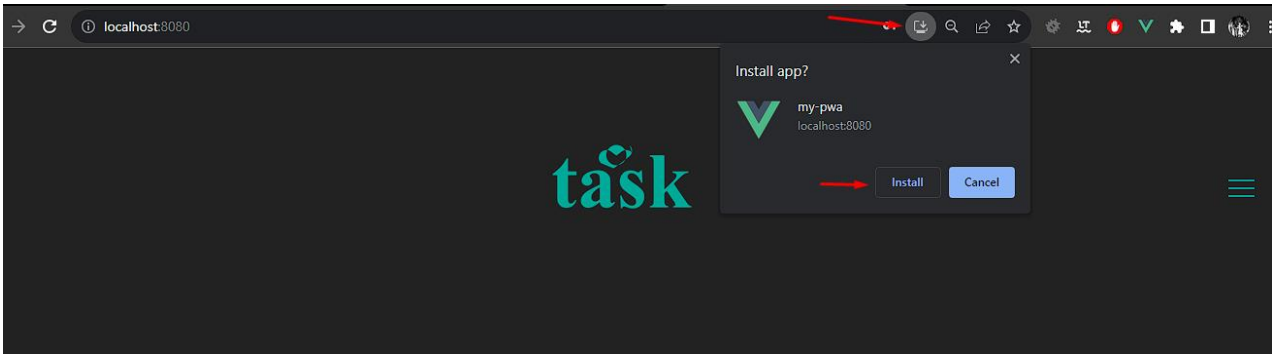
Τέλος μπορούμε να διαγράψουμε κάποιο task σε περίπτωση που δεν το χρειαζόμαστε πλέον.



Με αποτέλεσμα να διαγράφετε.



Αν έχουμε πολλαπλούς λογαριασμούς παραδείγμα έναν για την δουλεία και έναν για προσωπικούς λόγους μπορούμε πατώντας το burger menu να δούμε το λογαριασμό με τον οποίο είμαστε συνδεδεμένοι και να κάνουμε αποσύνδεση εφόσον το επιθυμούμε για να αλλάξουμε λογαριασμούς. Ακόμα αν θέλουμε να εγκαταστήσουμε την εφαρμογή στο κινητό μας ή στον υπολογιστή μας μπορούμε μέσα από τον browser της επιλογής μας να εγκαταστήσουμε σαν pwa application.



6. Συμπεράσματα και Μελλοντική Έρευνα

Συνοψίζοντας, τα task management applications αποτελούν ζωτικό εργαλείο για την αποτελεσματική οργάνωση και διαχείριση των καθηκόντων μας. Είτε πρόκειται για τη διαχείριση ενός επαγγελματικού έργου είτε για την οργάνωση των καθημερινών καθηκόντων μας, αυτές οι εφαρμογές προσφέρουν μια δομημένη προσέγγιση για την καταγραφή, την επισήμανση προτεραιοτήτων και την παρακολούθηση της προόδου.

Με τη δυνατότητα να οργανώνουμε τα καθήκοντα μας σε μία κεντρική πλατφόρμα, μπορούμε να επιτελούμε τις εργασίες μας με μεγαλύτερη αποτελεσματικότητα και αποφεύγουμε την αναποτελεσματική δαπάνη χρόνου και πόρων. Επιπλέον, η δυνατότητα παρακολούθησης της προόδου μας και η ευκολία στην ανακατανομή των καθηκόντων ανάλογα με τις προτεραιότητες συμβάλλει στη μείωση του στρες και τη βελτίωση της χρονοδιακίνησης.

Ειδικά στον επαγγελματικό τομέα, τα task management applications ενισχύουν τη συνεργασία, δίνοντας τη δυνατότητα να μοιράζονται εργασίες και να παρακολουθούνται οι πρόοδοι. Οι διαφορετικοί ρόλοι μπορούν να επικοινωνούν και να συνεργάζονται αποτελεσματικά με σαφήνεια σχετικά με το ποιος υπεύθυνος για κάθε καθήκον.

Σε κάθε πτυχή της ζωής μας, αυτές οι εφαρμογές συντελούν στη διαμόρφωση ενός οργανωμένου, αποτελεσματικού και λιγότερο στρεσογόνου περιβάλλοντος. Οι καθημερινές μας εργασίες, στόχοι και επιτεύγματα βελτιώνονται μέσα από μια δομημένη προσέγγιση που αυτές οι εφαρμογές παρέχουν.

Όσο για το τεχνικό της υλοποιήσης ενός task management pwa υπήρξαν αρκετές δυσκολίες τόσο στον σχεδιασμό όσο και στην αρχιτεκτονική της εφαρμογής.

Πιο συγκεκριμένα υπήρξαν αρκετά προβλήματα με τα tests και πιο συγκεκριμένα με το vue js

vite και το firebase. Υπήρξαν αρκετά error λόγο μη καλής συνδεσιμότητας μεταξύ των τεχνολογιών για αυτό και διάλεξα vanilla vue js για να αποφύγω όσο περισσότερο γίνεται τέτοιων ειδών conflicts.

Ακόμα μια δυσκολία που αντιμετώπισα ήταν η δυνατότητα να λειτουργήσει η εφαρμογή offline και να κάνει sync τα task με αυτά που υπήρχαν online αλλά λόγο χρονικών περιορισμών επέλεξα να μην το συμπεριλάβω.
Ένα αντίστοιχο πρόβλημα μη συμβατότητας υπήρξαν και με τα test της εφαρμογής που δεν τα πρόσθεσα λόγο των προβλημάτων με το vite.

Το Vite είναι ένα γρήγορο εργαλείο ανάπτυξης (build tool) που χρησιμοποιείται κυρίως για τη δημιουργία σύγχρονων εφαρμογών ιστού (web applications) χρησιμοποιώντας τεχνολογίες όπως JavaScript και TypeScript. Το Vite ξεχωρίζει για την ταχύτητά του στη δημιουργία και ανάπτυξη εφαρμογών, καθιστώντας τον κώδικα πιο αποδοτικό κατά τη διάρκεια της ανάπτυξης. Επιπλέον, το Vite είναι ειδικά σχεδιασμένο για να δουλεύει καλύτερα με πλαίσια εργασίας (frameworks) όπως το Vue.js, το React, και το Svelte.

Κατά την διάρκεια της ανάπτυξης της εφαρμογής, αντιμετώπισα ποικίλες τεχνικές προκλήσεις που απαιτούσαν την ανάπτυξη νέων δεξιοτήτων και την έρευνα σε βάθος τεχνολογικών κομματιών. Παρά την αρχική αίσθηση ανασφάλειας, η αντιμετώπιση αυτών των δυσκολιών με έκανε να εξελιχθώ ως προγραμματιστής και να κατανοήσω βαθύτερα τη λειτουργία των διαφόρων τεχνολογιών που χρησιμοποίησα.

Παράλληλα με τις δυσκολίες της ανάπτυξης μιας τέτοιας εφαρμογής αναγνώρισα και τις βελτιώσεις που θα προσέφεραν ακόμη καλύτερη λειτουργικότητα και επίδοση. Ας δούμε ορισμένες από αυτές τις βελτιώσεις :

1. Λειτουργία Offline και Sync: Όπως προανέφερες, η δυνατότητα λειτουργίας της εφαρμογής offline και συγχρονισμού των δεδομένων με αυτά που υπάρχουν online θα ήταν μια σημαντική βελτίωση. Αυτό θα επέτρεπε στους χρήστες να εργάζονται χωρίς σύνδεση στο διαδίκτυο και να αποθηκεύουν τις αλλαγές τους για μετέπειτα συγχρονισμό.

2. Βελτιωμένο Σύστημα Ειδοποιήσεων: Ένα προηγμένο σύστημα ειδοποιήσεων θα μπορούσε να παρέχει εξατομικευμένες υπενθυμίσεις και ειδοποιήσεις για τα task, βασιζόμενες στις προτεραιότητες και τα χαρακτηριστικά του χρήστη.

3. Ανάλυση Δεδομένων: Ένα προηγμένο σύστημα ανάλυσης δεδομένων μπορεί να προσφέρει γραφήματα και αναφορές για την παρακολούθηση της προόδου των έργων. Αυτό θα βοηθούσε τους χρήστες να λαμβάνουν πιο ενδελεχείς αποφάσεις για τη διαχείριση των εργασιών τους.

4. Ενσωμάτωση τεχνητής νοημοσύνης (AI): Η χρήση της τεχνητής νοημοσύνης θα μπορούσε να βελτιώσει την εφαρμογή σε πολλούς τομείς, όπως:

- Αυτόματη Ανάθεση Εργασιών: Ο AI θα μπορούσε να αναθέτει αυτόματα εργασίες σε βάση προτεραιότητας και δεξιοτήτων, βοηθώντας τους χρήστες να εργάζονται αποτελεσματικότερα.

- Πρόβλεψη Φορτίου Εργασίας: Ο AI θα μπορούσε να προβλέπει πότε θα υπάρξουν αυξημένες απαιτήσεις εργασίας και να βοηθά τους χρήστες να προγραμματίσουν κατάλληλα.

- Ανίχνευση Σφαλμάτων: Ο AI θα μπορούσε να ανιχνεύει ανομαλίες στη διαχείριση των εργασιών και να προτείνει βελτιώσεις.

Ένα ζωντανό παράδειγμα αυτού είναι το Asana που εφάρμοσε λειτουργίες τεχνητής νοημοσύνης (AI) για να βελτιώσει τη δυνατότητα διαχείρισης των εργασιών του. Αυτές οι λειτουργίες "έξυπνων" πεδίων, επεξεργαστών και περιλήψεων είναι προφανώς βασισμένες σε AI και φαίνεται ότι αναλαμβάνουν καθήκοντα που απαιτούν συνήθως τον ανθρώπινο παράγοντα. Για παράδειγμα, μπορεί να γράφουν κείμενο ακολουθώντας οδηγίες που τους δίνονται ή να εξάγουν σημαντικές λέξεις και ιδέες από μπλοκ κειμένου που τους ζητείται να διαβάσουν.

Αυτές οι εφαρμογές διαχείρισης εργασιών που βασίζονται σε AI μπορούν να δώσουν στους χρήστες τη δυνατότητα να αυτοματοποιούν τις διαδικασίες τους και να βελτιώσουν την αποτελεσματικότητά τους. Είναι ένα ζωντανό παράδειγμα του πώς η τεχνητή νοημοσύνη μπορεί να ενσωματωθεί στην καθημερινή διαχείριση εργασιών για να κάνει το έργο πιο αποτελεσματικό και ευκολότερο για τους χρήστες.

Αυτές οι βελτιώσεις θα καθιστούσαν την εφαρμογή πιο ισχυρή, αποδοτική και χρήσιμη για τους χρήστες της. Βέβαια κάτι τέτοιο θα απαιτούσε μια ολοκληρωμένη ομάδα τόσο προγραμματιστών όσο και επιστημόνων.

Ακόμα θα μπορούσε να γίνει ανασχεδιασμός UI/UX από έναν ειδικό σχεδιαστή θα μπορούσε να βελτιώσει σημαντικά την εμπειρία των χρηστών με την εφαρμογή διαχείρισης εργασιών. Ας εξετάσουμε πιθανούς τομείς βελτίωσης:

Σχεδιασμός Χρήστη (UI): Ένας σχεδιαστής μπορεί να αναθεωρήσει το UI της εφαρμογής για να το καταστήσει πιο ελκυστικό και εύχρηστο. Αυτό περιλαμβάνει τη βελτίωση του σχεδιασμού των κουμπιών, της γραφικής απεικόνισης και της γενικής ροής της εφαρμογής.

Βέλτιστη Εμπειρία Χρήστη (UX): Ο σχεδιαστής μπορεί να βελτιώσει την γενική εμπειρία χρήστη με τη βελτίωση της διάταξης των στοιχείων, της διανομής πληροφοριών και της αλληλεπίδρασης με την εφαρμογή.

Συμβατότητα με Συσκευές: Ο σχεδιαστής μπορεί να διασφαλίσει ότι η εφαρμογή είναι πλήρως συμβατή με διάφορες συσκευές, όπως φορητές συσκευές, tablet και υπολογιστές, για βέλτιστη απόδοση σε όλα τα μεγέθη οθονών.

Ανταπόκριση (Responsiveness): Η ανταπόκριση της εφαρμογής σε διάφορες αναλύσεις οθονών θα βελτίωνε την εμπειρία των χρηστών και θα επέτρεπε την χρήση της σε διάφορες συνθήκες.

Διαχείριση Χρωμάτων και Στυλ: Ένας σχεδιαστής μπορεί να δημιουργήσει ένα σύνολο στοιχείων σχεδιασμού, συμπεριλαμβανομένων χρωμάτων, γραμματοσειρών και στιλ, που θα δίνουν στην εφαρμογή μια ενιαία και αναγνωρίσιμη εμφάνιση.

Διευκρίνιση των Δεδομένων: Ο σχεδιαστής μπορεί να βελτιώσει την παρουσίαση των δεδομένων για να καταστήσει τις πληροφορίες πιο ευανάγνωστες και κατανοητές.

Ανάπτυξη Εικονικών Πρωτοτύπων (Prototyping): Η δημιουργία εικονικών πρωτοτύπων της εφαρμογής θα επιτρέπει στους χρήστες και τον ανάπτυκτη να εκτιμήσουν το νέο UI/UX πριν υλοποιηθεί πλήρως.

Ένας σχεδιαστής με εξειδίκευση σε UI/UX θα βοηθούσε στη δημιουργία μιας πιο ελκυστικής, λειτουργικής και χρήσιμης εφαρμογής διαχείρισης εργασιών που θα ικανοποιεί τους χρήστες και θα τους επιτρέπει να είναι πιο αποδοτικοί στη διαχείριση των καθηκόντων τους.

Τέλος η ανάγκη να δημιουργήσω από το μηδέν μια εφαρμογή που μέχρι τότε απλώς χρησιμοποιούσα, με ώθησε να εστιάσω στην ανάπτυξη κατάλληλων δομών και τη βέλτιστη αξιοποίηση των προτεινόμενων τεχνολογιών. Αυτό με οδήγησε σε μια βαθύτερη κατανόηση της αρχιτεκτονικής, του state management και της διαχείρισης της βάσης δεδομένων.

Σε γενικότερο επίπεδο, ανακάλυψα πώς τα task management applications διαδραματίζουν έναν ζωτικό ρόλο στην οργάνωση των εργασιών, είτε αυτές είναι προσωπικές είτε επαγγελματικές. Αντιστοιχούν σε μια δικτύωση σκέψης και προγραμματισμού, που διευκολύνει την καθημερινή εφαρμογή των στόχων μας. Η δυνατότητα να παρακολουθούμε, επεξεργαζόμαστε και οργανώνουμε τα καθήκοντά μας επιτρέπει την αύξηση της παραγωγικότητας, τη βελτίωση του χρόνου διαχείρισης και την ενίσχυση της επίτευξης των στόχων μας.

Πηγές

Vue js documentation - https://vuejs.org/guide/introduction.html

Firebase documentation - https://firebase.google.com/docs

Firebase guide - https://www.tutorialspoint.com/firebase/index.htm

Asana - https://asana.com/

Microsoft to do - https://to-do.office.com/tasks/

Building a Task Management Library with Vue 3 and Composition API - https://medium.com/@seancheongzhenxiong/building-a-task-management-library-with-vue-3-and-composition-api-9548aa96f14e

Vue.js 3 Firebase Firestore Setup Tutorial - https://www.koderhq.com/tutorial/vue/firebase-firestore/

Επιστημονικά άρθρα, studies και βιβλία :

" Project Management Software Utilization and Project Performance" - https://rb.gy/1xq75

“What a To-Do: Studies of Task Management Towards the Design of a Personal Task List Manager” - https://www.researchgate.net/publication/221518959_What_a_to-do_studies_of_task_management_towards_the_design_of_a_personal_task_list_manager

“ What a to-do: studies of task management towards the design of a personal task list manager” - https://www.semanticscholar.org/paper/What-a-to-do%3A-studies-of-task-management-towards-of-Bellotti-Dalal/8946a35c5ee027078d953f2cb34d1248346bd37b

“ Managing Tasks across the Work–Life Boundary: Opportunities, Challenges, and Directions ” https://dl.acm.org/doi/10.1145/3582429

Εργαλεία :

Intelij Webstorm (ide) - https://www.jetbrains.com/webstorm/

Github (Thesis application link) - https://github.com/Drivakos/task-management-app

Node js - Node.js (nodejs.org)