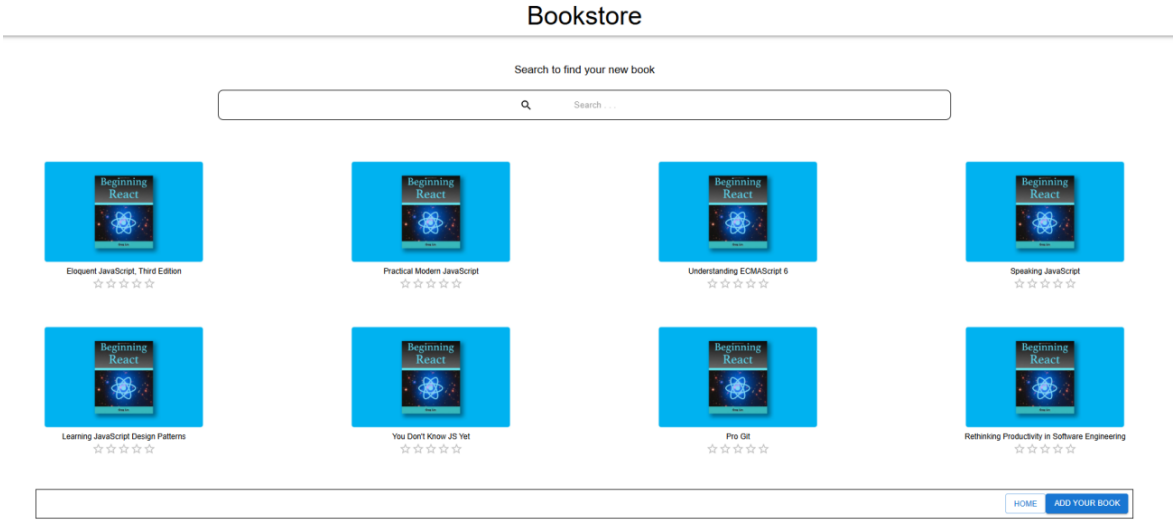
ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«BOOKSTORE- ΔΗΜΟΥΡΓΙΑ ΔΥΝΑΜΙΚΗΣ ΕΦΑΡΜΟΓΗΣ FRONT-END ΓΙΑ ΔΙΑΧΕΙΡΙΣΗ ΒΙΒΛΙΟΠΩΛΕΙΟΥ»



Του φοιτητή
Σταμάτιου Καλούδη
Αρ. Μητρώου: it164694

Επιβλέπων
Ονοματεπώνυμο: Χαράλαμπος Μπράτσας
Βαθμίδα: Επίκουρος Καθηγητής

Ημερομηνία 27/10/2024

Τίτλος Δ.Ε. Δυναμική Εφαρμογή Διαχείριση Βιβλιοπωλείου
Κωδικός Δ.Ε. 24286
Ονοματεπώνυμο φοιτητή: Σταμάτιος Καλούδης
Ονοματεπώνυμο εισηγητή Χαράλαμπος Μπράτσας
Ημερομηνία ανάληψης Δ.Ε. 27-10-2024
Ημερομηνία περάτωσης Δ.Ε. 27-10-2026

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως πτυχιακή εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

«Αφιέρωση»
Στους «δικούς μου» ανθρώπους

Πρόλογος

Από μικρός, με ενδιέφερε ο κόσμος του διαδικτύου και των τεχνολογιών που το στηρίζουν. Η πρώτη μου επαφή με την HTML και την JavaScript, κατά τη διάρκεια του 2ου εξαμήνου με τίτλο μαθήματος Γλώσσες και Τεχνολογίες Ιστού και στην συνέχεια κατά τη διάρκεια της δημιουργίας της προσωπικής μου ιστοσελίδας, αποτέλεσε το ερέθισμα για να ασχοληθώ με τον προγραμματισμό και τη σχεδίαση ιστοσελίδων. Χρησιμοποιώντας HTML, CSS και JavaScript, πειραματίστηκα με διάφορες τεχνολογίες, όπως το Parallax effect και το hover effect, προσπαθώντας να δημιουργήσω πιο διαδραστικά και ελκυστικά αποτελέσματα.

Η πρακτική μου άσκηση ως Front-End προγραμματιστής σε Ευρωπαϊκά έργα, κυρίως με το Ευρωπαϊκό Γραφείο Διανοητικής Ιδιοκτησίας (European Union Intellectual Property Office – EUIPO), με βοήθησε να εμβαθύνω στην ανάπτυξη web εφαρμογών και να κατανοήσω τη σημασία της χρηστικότητας και της αποδοτικότητας στον προγραμματισμό. Η συνεργασία μου με τον υπεύθυνο μου, ο οποίος ήταν και ο ηγέτης της ομάδας, με ενέπνευσε να δημιουργήσω μία εφαρμογή που θα ενσωματώνει ευρωπαϊκά πρότυπα και θα συμβάλλει στη βελτίωση της διαδικασίας εκμάθησης και διαχείρισης βιβλίων μέσω μιας δυναμικής front-end εφαρμογής.

Με την παρούσα πτυχιακή εργασία, σκοπός μου είναι να δημιουργήσω μια εφαρμογή βιβλιοπωλείου, η οποία θα επιτρέπει την καταχώρηση και ανάγνωση των λεπτομερειών των βιβλίων, ενώ ταυτόχρονα θα αξιοποιεί σύγχρονες τεχνολογίες όπως React, Vite, Material UI και άλλες προκειμένου να προσφέρει έναν μοντέρνο και εύχρηστο τρόπο αλληλεπίδρασης με τα δεδομένα.

Περίληψη

Ο στόχος της παρούσας πτυχιακής εργασίας είναι η ανάπτυξη μιας δυναμικής και αυτόνομης front-end web εφαρμογής, για τη διαχείριση βιβλίων, με δυνατότητες αναζήτησης, καταχώρησης και παρουσίασης βιβλίων. Η εφαρμογή “Bookstore” δημιουργήθηκε με σκοπό να παρέχει στους χρήστες μια εύχρηστη και σύγχρονη πλατφόρμα για την εύκολη καταχώρηση και εύρεση βιβλίων με βάση διάφορα κριτήρια, όπως είναι ο συγγραφέας, ο τίτλος και τα ISBN. Η εφαρμογή διαθέτει δυνατότητες για την αναζήτηση και την εμφάνιση σημαντικών πληροφοριών για το κάθε βιβλίο και την προσθήκη νέων από τους χρήστες, διασφαλίζοντας μια ευχάριστη εμπειρία για τον χρήστη.

Η ανάπτυξη της εφαρμογής πραγματοποιήθηκε χρησιμοποιώντας σύγχρονα εργαλεία και τεχνολογίες που επιτρέπουν την ανάπτυξη γρήγορων και αποδοτικών διαδικτυακών εφαρμογών. Ιδιαίτερη έμφαση δόθηκε στη χρήση της βιβλιοθήκης React για την ανάπτυξη των στοιχείων. Επιπλέον, χρησιμοποιήθηκε η βιβλιοθήκη Mock Service Worker, για την προσομοίωση API αιτημάτων και την δημιουργία ψεύτικων δεδομένων κατά την διάρκεια ανάπτυξης της, προσφέροντας απομόνωση από εξωτερικές και πραγματικές υπηρεσίες και αποθηκεύοντας τα δεδομένα προσωρινά σε ένα τοπικό αρχείο JSON. Η εφαρμογή παρέχει δυναμικό περιβάλλον αναζήτησης, ενώ τα δεδομένα επικοινωνούν με τα δομικά στοιχεία μέσω των properties και των states.

Η εφαρμογή υποστηρίζει τη λειτουργία Single Page Application, επιτρέποντας στους χρήστες να περιηγηθούν χωρίς να απαιτείται η ανανέωση σελίδας. Μέσα από την χρήση του React Router, οι χρήστες μπορούν να πλοηγηθούν εύκολα και αποτελεσματικά στις σελίδες της εφαρμογής για την καταχώρηση και την αναζήτηση των βιβλίων αλλά και για την προβολή των λεπτομερειών του κάθε βιβλίου.

Η μελλοντική εξέλιξη της εφαρμογής περιλαμβάνει τη δυνατότητα προσθήκης νέων λειτουργιών και την ενσωμάτωση πραγματικής βάσης δεδομένων για μόνιμη αποθήκευση των βιβλίων, καθώς και τη βελτίωση του περιβάλλοντος με νέα χαρακτηριστικά.

Η εφαρμογή επικεντρώνεται στην παρουσίαση των τεχνολογιών που χρησιμοποιούνται για την ανάπτυξη της εφαρμογής, καθώς και στην περιγραφή της αρχιτεκτονικής και των λειτουργικών χαρακτηριστικών που προσφέρει η εφαρμογή στους χρήστες.

«BOOKSTORE- Creation of a dynamic front-end web application for Bookstore management»

«Kaloudis Stamatios»

Abstract

The goal of this thesis is the development of a dynamic and autonomous front-end web application for book management, with functionalities about searching, registering and displaying books. The "Bookstore" which is the title of the application, was created to provide users with an easy-to-use and modern platform for the effortless registration and discovery of books based on various criteria, such as author, description, title, and ISBN. The application offers features such as dynamic search bar for the books, displaying essential information about each book individual and allowing the users to add new books, ensuring a pleasant and fast user experience.

The development of the application was developed by using modern tools and technologies that enable the creation of fast and efficient web applications. Special emphasis was placed on the use of library React for the development of components. Moreover, Vite is a build tool that aims to provide a faster and leaner development experience for modern web projects. Additionally, Mock Service Worker was used for simulating API requests and providing mock data during development, offering isolation from external and real services which the data are storing in a local JSON file. The application provides a dynamic search environment, while properties are used to pass data from a parent component to a child component, while the state is used to manage the data within a component.

The application supports Single Page Application functionality, allowing users to navigate without requiring page refreshes. Through the usage of React Router, users can easily and efficiently navigate between the application's pages for book registration, book search and viewing book details.

Furthermore, the application was developed with the aim of improving the book registration process, large-scale book search and user experience, offering a simpler and faster interaction with the system. Future developments of the application include the ability to add new features and integrate a real database for permanent book storage, as well as enhancing the environment with new characteristics.

With this application, the process of searching and registering books becomes more accessible, easy and quick. It focuses mainly on presenting the features that the application offers to its user. technologies used in the development of the "Bookstore" application, as well as describing the architecture and functionality.

Ευχαριστίες

Για την εκπόνηση της πτυχιακής μου εργασίας, θα ήθελα να ευχαριστήσω πρώτα από όλους, τον καθηγητή μου, κο Χαράλαμπο Μπράτσα, Επίκουρο Καθηγητή του Τμήματος Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙΠΑΕ, για το ενδιαφέρον και για την εμπιστοσύνη που έδειξε στην πρόταση του θέματος της πτυχιακής εργασίας.

Ιδιαίτερα, οφείλω να αποδώσω τη μέγιστη εκτίμηση μου στον μέντορα της εταιρείας στην οποία εργάζομαι Netcompany – Intrasoft, η οποία δραστηριοποιείται στον τομέα των Ευρωπαϊκών προγραμμάτων και ζητημάτων. Η εταιρεία και ειδικότερα η υποστήριξη που μου παρείχαν, με βοήθησαν να κατανοήσω σε βάθος τις σύγχρονες τεχνολογίες και να εμπνευστώ για την εκπόνηση και παρουσίαση αυτών στην παρούσα πτυχιακή εργασία. Η εμπειρία αυτή υπήρξε καθοριστική για την ανάπτυξη μου τόσο σε τεχνικό επίπεδο όσο και σε επαγγελματικό.

Τέλος, θα ήθελα να εκφράσω τις πιο θέρμες από καρδίας μου ευχαριστίες για τους φίλους, την ηθική τους συμπαράσταση καθ’ όλη την διάρκεια της πορείας μου, αλλά περισσότερο από όλους όμως, την οικογένεια μου, η οποία ήταν εκεί για εμένα και με στήριξε σε κάθε βήμα αυτής της διαδικασίας.

Περιεχόμενα

Κατάλογος Πινάκων

Κατάλογος Σχημάτων

Συντομογραφίες

HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
MSW	Mock Service Worker
SWR	Stale While Revalidate
API	Application Programming Interface
JSON	JavaScript Object Notation
JS	JavaScript
JSX	JavaScript XML
DOM	Document Object Model
JSDOM	JavaScript Document Object Model
SPA	Single Page Application
ISBN	International Standard Book Number
UI	User Interface
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
HMR	Hot Module Replacement
HTTPS	Hypertext Transfer Protocol Secure
Props	Properties

Κεφάλαιο 1ο: Εισαγωγή στο Βιβλιοπωλείο και τις Σχετικές Τεχνολογίες

1.1 Εισαγωγή

Στη σύγχρονη εποχή της τεχνολογίας, οι διαδικτυακές εφαρμογές γίνονται ολοένα και πιο σημαντικές για την καθημερινή ζωή του ανθρώπου, καθώς διευκολύνουν την αλληλεπίδραση των χρηστών με διάφορες υπηρεσίες και δεδομένα. Η εφαρμογή, δημιουργήθηκε για να καλύψει την ανάγκη διαχείρισης βιβλίων, όπου οι χρήστες μπορούν να αναζητούν, να καταχωρούν και να διατρέχουν στις λεπτομέρειες των βιβλίων.

Αυτή η πτυχιακή εργασία αφορά την ανάπτυξη μιας πλήρους και ολοκληρωμένης front-end εφαρμογής για τη διαχείριση ενός βιβλιοπωλείου, η οποία παρέχει μια πλήρη και ομαλή εμπειρία στον χρήστη με λειτουργίες αναζήτησης, προβολής βιβλίων, καταχώρησης νέων βιβλίων και παρουσίασης προτεινόμενων βιβλίων. Όλες οι λειτουργίες υλοποιούνται αποκλειστικά και μόνο με την χρήση JavaScript, React και των πιο σύγχρονων εργαλείων για την ανάπτυξη διαδικτυακών εφαρμογών.

1.2 Σκοπός και Στόχοι της Εφαρμογής

Η εφαρμογή στοχεύει στο να προσφέρει στους χρήστες, μια πλήρη εμπειρία διαχείρισης και ανασκόπησης βιβλίων που είναι καταχωρημένα στο σύστημα. Οι βασικές λειτουργίες της εφαρμογής περιλαμβάνουν:

- **Αναζήτηση Βιβλίων – Αρχική Σελίδα:** Ο χρήστης μπορεί να αναζητήσει βιβλία χρησιμοποιώντας διάφορα κριτήρια όπως είναι ο τίτλος, ο συγγραφέας και το ISBN του βιβλίου. Αυτό γίνεται μέσω μιας δυναμικής μπάρας αναζήτησης που επιτρέπει την γρήγορη αναζήτηση και με ακριβές φιλτράρισμα των αποτελεσμάτων. Επιπλέον, την παρακολούθηση όλων των καταχωρημένων βιβλίων και την δυνατότητα ο χρήστης να αλληλεπιδράσει με αυτά.

- **Καταχώρηση Βιβλίων:** Στη δεύτερη σελίδα της εφαρμογής, οι χρήστες μπορούν να καταχωρήσουν τα βιβλία τους με όλα τα απαραίτητα στοιχεία που χρειάζεται να έχει ένα βιβλίο, όπως είναι ο τίτλος, ο συγγραφέας, το ISBN10, το ISBN13, το περιεχόμενο κλπ.

- **Λεπτομέρειες Βιβλίου:** Όταν ο χρήστης επιλέξει και αλληλεπιδράσει με ένα βιβλίο από την αρχική σελίδα της εφαρμογής, θα πραγματοποιηθεί η μεταφορά του στην σελίδα του βιβλίου με βάση του μοναδικό κωδικό ISBN. Στην συνέχεια θα γίνει η εμφάνιση των στοιχείων του βιβλίου, όπως είναι η περιγραφή, ο συγγραφέας , τα ISBN, την αξιολόγηση του κλπ.

Η σελίδα περιλαμβάνει επίσης ένα Carousel που προτείνει άλλα βιβλία στον χρήστη και εξαιρεί από την λίστα αυτή το βιβλίο της σελίδας που ήδη προβάλλεται. Ενισχύοντας μέσα από αυτόν τον τρόπο, την συνολική εμπειρία του χρήστη, αλληλοεπιδρώντας με δυναμικό τρόπο με την εφαρμογή.

Η εφαρμογή υλοποιείται ως μια εξ' ολοκλήρου front-end λύση , χωρίς να εξαρτάται από εξωτερικούς παράγοντες για τη λειτουργία της, εστιάζοντας στην καλύτερη εμπειρία, αλληλεπίδραση και διεπαφή του χρήστη αλλά και στην ευχρηστία του περιβάλλοντος.

1.3 Τεχνολογίες που Χρησιμοποιούνται

Η εφαρμογή έχει αναπτυχθεί χρησιμοποιώντας σύγχρονες και καινοτόμες τεχνολογίες και εργαλεία, τα οποία συνδυαστικά παρέχουν υψηλές αποδόσεις και επαγγελματική ποιότητα στην ανάπτυξη της:

- **React:** Η βιβλιοθήκη React χρησιμοποιείται για την ανάπτυξη της εφαρμογής, παρέχοντας τη δυνατότητα δυναμικής ανανέωσης της διεπαφής του χρήστη χωρίς πλήρη ανανέωση της σελίδας. Με το React, επιτυγχάνεται η δημιουργία επαναχρησιμοποιημένων δομικών στοιχείων, καθιστώντας την εφαρμογή πιο ευέλικτη, σταθερή και επεκτάσιμη.

- **Vite:** Το Vite χρησιμοποιείται ως κύριο εργαλείο ανάπτυξης, προσφέροντας ταχύτητα και αποτελεσματικότητα κατά τη διάρκεια ανάπτυξης της εφαρμογής. Επιτρέπει τη γρήγορη ανανέωση της εφαρμογής κατά την τροποποίηση του κώδικα, διευκολύνοντας τη διαδικασία αποσφαλμάτωσης αλλιώς το γνωστό debugging.

- **Vitest:** Το Vitest χρησιμοποιείται προκειμένου να εξασφαλίσει τη ποιότητα του κώδικα και για τη δημιουργία και την υλοποίηση των δοκιμών της εφαρμογής. Αυτό επιτρέπει τη δημιουργία και εκτέλεση δοκιμών στοιχείων για τις λειτουργίες της εφαρμογής, βοηθώντας στην ανίχνευση και διόρθωση σφαλμάτων, εξασφαλίζοντας ότι η εφαρμογή λειτουργεί σωστά χωρίς σφάλματα.

- **Mock Service Worker:** Το Mock Service Worker(MSW) χρησιμοποιείται για την δημιουργία τεχνικών API κλήσεων, επιτρέποντας την άμεση δοκιμή της εφαρμογής με δεδομένα που προσομοιώνουν πραγματικές API απαντήσεις. Αποφεύγεται με αυτόν τον τρόπο η ανάγκη σύνδεσης της εφαρμογής με εξωτερικούς και απομακρυσμένους διακομιστές.

- **Stale While Revalidate:** Το Stale While Revalidate(SWR) χρησιμοποιείται για τη διαχείριση ανάκτησης δεδομένων από εξωτερικές, αλλά και τεχνικές υπηρεσίες στην περίπτωση της

εφαρμογής. Η εφαρμογή μπορεί να αντλεί τα δεδομένα με αποδοτικό και ασφαλή τρόπο, εξασφαλίζοντας ταχύτητα και αξιοπιστία στην επικοινωνία με τα APIs.

- **Material UI:** Η βιβλιοθήκη Material UI χρησιμοποιείται με σκοπό να παρέχει τα απαραίτητα στοιχεία διεπαφής χρήστη, προκειμένου να διασφαλιστεί η κομψότητα και η ευχρηστία της εφαρμογής. Χρησιμοποιείται για την υλοποίηση των κουμπιών, των κελιών της φόρμας, την δυναμική μπάρα αναζήτησης στην αρχική σελίδα, την διαμόρφωση των σελίδων και άλλων οπτικών στοιχείων της εφαρμογής.

1.4 Αρχιτεκτονική της Εφαρμογής

Η αρχιτεκτονική της εφαρμογής ακολουθεί τη δομής μίας μοντέρνας React εφαρμογής με δομικά στοιχεία που επαναχρησιμοποιούνται και επικοινωνούν μέσω props και states. Ο χρήστης αλληλεπιδρά με την εφαρμογή μέσω της αρχικής σελίδας, της σελίδας καταχώρησης βιβλίων και της σελίδας λεπτομερειών βιβλίου με την χρήση εμφανών κουμπιών.

Η εφαρμογή επικεντρώνεται στην χρήση React για τις ενέργειες του χρήστη, ενώ η διαχείριση της κατάστασης γίνεται με το built-in React state και hooks που προσφέρει η συγκεκριμένη JavaScript βιβλιοθήκη.

1.5 Διάγραμμα Ροής

Για την καλύτερη κατανόηση της λειτουργίας της εφαρμογής, στο διάγραμμα ροής, το οποίο παρουσιάζει την αλληλουχία των ενεργειών στην εφαρμογή, από την στιγμή που ο χρήστης αλληλοεπιδρά μέχρι και την εμφάνιση των αποτελεσμάτων. Το διάγραμμα ροής απεικονίζει τα εξής βήματα της εφαρμογής:

- **Εισαγωγή Δεδομένων**: Ο χρήστης εισάγει δεδομένα στην μπάρα αναζήτησης είτε στην καταχώρηση νέου βιβλίου μέσω της φόρμας καταχώρησης, προκειμένου να επιστραφούν τα δεδομένα με βάση την αναζήτηση ή την καταχώρηση του χρήστη.

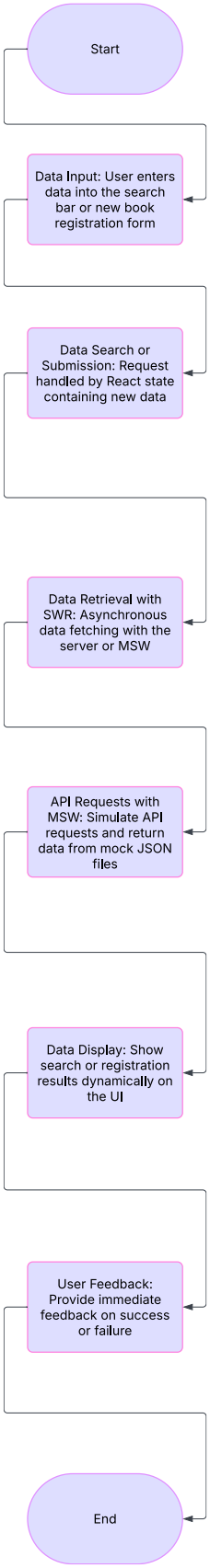
- **Αναζήτηση-Καταχώρηση Δεδομένων:** Τα αιτήματα του χρήστη για αναζήτηση είτε για καταχώρηση επεξεργάζονται μέσω του state που προσφέρει η React, το οποίο περιλαμβάνει τα νέα δεδομένα.

- **Ανάκτηση Δεδομένων με χρήση SWR**: Χρησιμοποιείται το SWR για την ασύγχρονη ανάκτηση των δεδομένων. Αναλαμβάνει την επικοινωνία με τον διακομιστή ή με το MSW όταν υπάρχουν αιτήματα και διαχειρίζεται τα δεδομένα που επιστρέφονται χωρίς να περιορίζει την εφαρμογή.

- **API αιτήματα με την χρήση του MSW**: Γίνεται η προσομοίωση των αιτημάτων API μέσω του MSW καθώς επιστρέφει τα δεδομένα από το ψεύτικο άρχει JSON.

- **Εμφάνιση Δεδομένων**: Όλα τα αποτελέσματα καταχώρησης ή αναζήτησης βιβλίων εμφανίζονται στην οθόνη του χρήστη απευθείας μέσα από ένα αρκετά γρήγορο αλλά και δυναμική διεπαφή χρήστη. Ανανεώνεται διαρκώς με χρήση των state και γίνεται η αποθήκευση των δεδομένων σε αυτών σε συνδυασμό με την βοήθεια που παρέχουν τα εργαλεία SWR και MSW.

- **Ανατροφοδότηση Χρήστη**: Η εφαρμογή παρέχει άμεση ανατροφοδότηση στον χρήστη σε περίπτωση αποτυχίας ή επιτυχίας των ενεργειών του είτε είναι για αναζήτηση βιβλίων είτε για καταχώρηση βιβλίων και γίνεται η εμφάνιση αυτών στην οθόνη του χρήστη μέσω σφαλμάτων.



Σχήμα 1.1: Διάγραμμα Ροής της ανάπτυξης της εφαρμογής.

Κεφάλαιο 2ο: Ανάλυση των Τεχνολογιών και Βιβλιογραφική Ανασκόπηση

2.1 Εισαγωγή στις Τεχνολογίες Ανάπτυξη Ιστού

Η ανάπτυξη σύγχρονων διαδικτυακών εφαρμογών απαιτεί την χρήση ποικίλων εργαλείων και τεχνολογιών, οι οποίες εξελίσσονται συνεχώς ανά τακτά χρονικά διαστήματα προκειμένου να καλύψουν τις αυξανόμενες απαιτήσεις και ανάγκες των χρηστών και των προγραμματιστών. Στη παρούσα πτυχιακή εργασία, η εφαρμογή αναπτύχθηκε με τη χρήση σύγχρονων τεχνολογιών για να παρέχει μια δυναμική και φιλική προς τον χρήστη εμπειρία. Στο παρόν κεφάλαιο, θα παρουσιαστεί μια ανασκόπηση της ιστορίας και της εξέλιξης των βασικών τεχνολογιών που χρησιμοποιήθηκαν για την ανάπτυξη της, όπως είναι η HyperText Markup Language(HTML), η JavaScript, το React και το Vite. Παράλληλα, θα γίνει ανάλυση της βιβλιογραφίας και των χαρακτηριστικών αυτών των τεχνολογιών, καθώς και τη συνεισφορά τους στην ανάπτυξη σύγχρονων διαδικτυακών εφαρμογών και των δυνατοτήτων που μπορούν να προσφέρουν σε έναν προγραμματιστή.

2.2 Τι είναι το Front-End;

Το Front-end αναφέρεται στη δυναμική διεπαφή χρήστη(User Interface) ή αλλιώς UI και στις τεχνολογίες που χρησιμοποιούνται για την ανάπτυξη του τμήματος της εφαρμογής, που η κύρια λειτουργία του είναι η αλληλεπίδραση με τον χρήστη. Σε αντίθεση με το back-end, το οποίο είναι αρμόδιο για τη διαχείριση των δεδομένων και την λειτουργία της εφαρμογής στο παρασκήνιο της. Το front-end εστιάζει στην εμφάνιση και αλληλεπιδράσεις της εφαρμογής.

Συνοπτικά το front-end περιλαμβάνει όλα όσα βλέπει ο χρήστης, όπως είναι οι φόρμες, τα κουμπιά, τα γραφικά κλπ. Οι κύριες τεχνολογίες που απαρτίζουν το front-end είναι η HTML, η CSS, η JavaScript και διάφορα άλλα συστήματα ανάπτυξη της.

Στοιχεία του Front-end:

1. **HyperText Markup Language(HTML):** Είναι η γλώσσα σήμανσης που καθορίζει τη δομή της ιστοσελίδας.

2. **Cascading Style Sheets(CSS):** Χρησιμοποιείται κατά κύριο λόγο για τον σχεδιασμό και την εμφάνιση της ιστοσελίδας, δίνοντας οπτικές ιδιότητες.

3. **JavaScript:** Είναι η γλώσσα προγραμματισμού που δίνει δυναμική συμπεριφορά στην ιστοσελίδα και επιτρέπει στον χρήστη να αλληλεπιδράσει με αυτήν.

2.2 Ιστορική Αναδρομή στην HTML

Η HTML είναι η θεμελιώδης γλώσσα σήμανσης που χρησιμοποιείται για την κατασκευή ιστοσελίδων και εφαρμογών στο διαδίκτυο. Δημιουργήθηκε το 1991 από τον Tim Berners-Lee, τον ιδρυτή του Παγκόσμιου Ιστού ή αλλιώς World Wide Web. Ο σκοπός της ήταν να επιτρέπει τη σύνδεση και παρουσίαση κειμένου μέσω υπερσυνδέσμων και να παρέχει τη βασική δομή για τη δημιουργία ιστοσελίδων.

Η HTML ήταν η πρώτη γλώσσα που έθεσε τα θεμέλια για τη διαχείριση και την οργάνωση πληροφοριών στο διαδίκτυο που στην αρχή ήταν περιορισμένη σε βασικές ετικέτες, όπως για παράδειγμα ήταν τα <p> , <a> , <h1> , αλλά και για την εμφάνιση κειμένου και την πλοήγηση μεταξύ σελίδων. [1]

Με την πάροδο των χρόνων, οι ανάγκες του διαδίκτύου αυξήθηκαν και η HTML εξελίχθηκε και αναπτύχθηκε για να υποστηρίξει τις πιο σύνθετες ανάγκες που δημιουργήθηκαν. Η HTML4, που είναι η πιο διαδεδομένη και γνωστή έκδοση της HTML, έδωσε τη δυνατότητα να ορίζουμε πιο σύνθετες και περίπλοκες δομές, όπως είναι οι πίνακες, οι φόρμες και τα πλαίσια.

Ωστόσο, η πραγματική επανάσταση ξεκίνησε το 2014 με την παρουσίαση της HTML5, η οποία προσέφερε νέα χαρακτηριστικά και καινοτομίες για τη δημιουργία πολυμέσων, διαδραστικών στοιχείων και καλύτερη υποστήριξη για κινητές συσκευές.

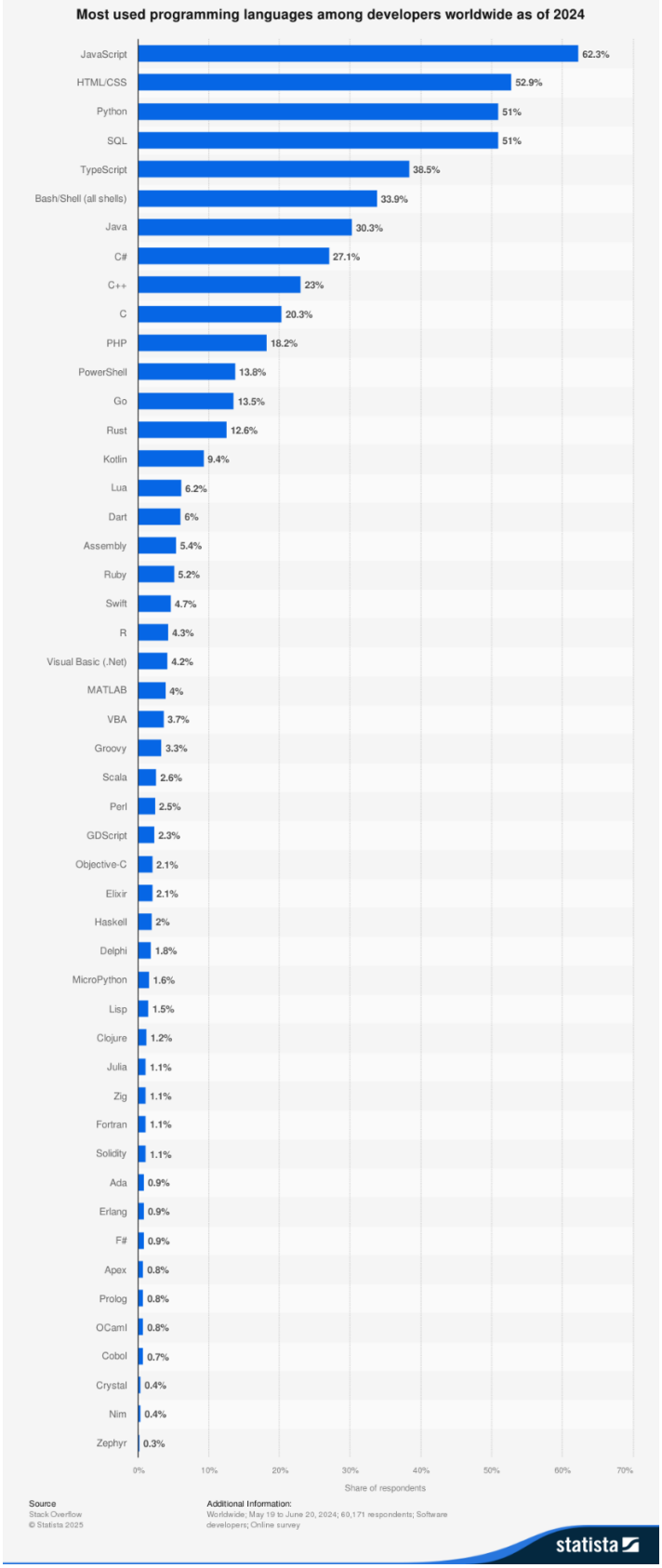
Η HTML5 έφερε στο φως νέα στοιχεία όπως είναι τα <video>, <radio> , <canvas> και πολλά άλλα, επιτρέποντας με αυτόν τον τρόπο την ενσωμάτωση πολυμέσων στις εφαρμογές αλλά και την ανάπτυξη δυναμικών εφαρμογών χωρίς την ανάγκη εξωτερικών πρόσθετων λειτουργιών όπως είναι το Flash. Η HTML5 είναι αυτή που χρησιμοποιείται στην εφαρμογή, προσφέροντας τη δυνατότητα για πλουσιότερη και δυναμικότερη παρουσίαση των βιβλίων και των πληροφοριών τους. [2]

2.3 Η Εξέλιξη της JavaScript

Η δυναμική γλώσσα προγραμματισμού JavaScript δημιουργήθηκε το 1995 από τον Brendan Eich και χρησιμοποιήθηκε αρχικά για να προσφέρει δυνατότητες αλληλεπίδρασης στον χρήστη, όπως η επαλήθευση φορμών και η δημιουργία διαδραστικών χαρακτηριστικών στις ιστοσελίδες. Αρχικά, η JavaScript ήταν περιορισμένη σε πολύ βασικές δυνατότητες, όμως με την συνεχή ανάπτυξη της και με την εισαγωγή του ECMAScript, εξελίχθηκε σε μια πλήρη και αξιόλογη γλώσσα προγραμματισμού. [3]

Η JavaScript είναι πλέον η πιο δημοφιλής γλώσσα προγραμματισμού για ανάπτυξη διαδικτυακών εφαρμογών και χρησιμοποιείται για αυτοματοποιημένες δοκιμές με την βοήθεια αρκετών βιβλιοθηκών όπως είναι η Vitest. Οι δυνατότητες χειρισμού των DOM στοιχείων μέσω του Document Object Model(DOM) και η συνεχής εξέλιξη της JavaScript, καθίσταται απαραίτητη για τη δημιουργία διαδραστικών και δυναμικών διαδικτυακών εφαρμογών.

Η κύρια γλώσσα προγραμματισμού που χρησιμοποιείται στην εφαρμογή είναι η JavaScript, για τη διαχείριση των δεδομένων των βιβλίων, την αλληλεπίδραση με τον χρήστη και την επικοινωνία με εσωτερικές υπηρεσίες με την βοήθεια API κλήσεων.[4]



Σχήμα 2.1: JavaScript η πιο χρησιμοποιημένη γλώσσα προγραμματισμού το 2024. [5]

2.4 Ανάπτυξη του React

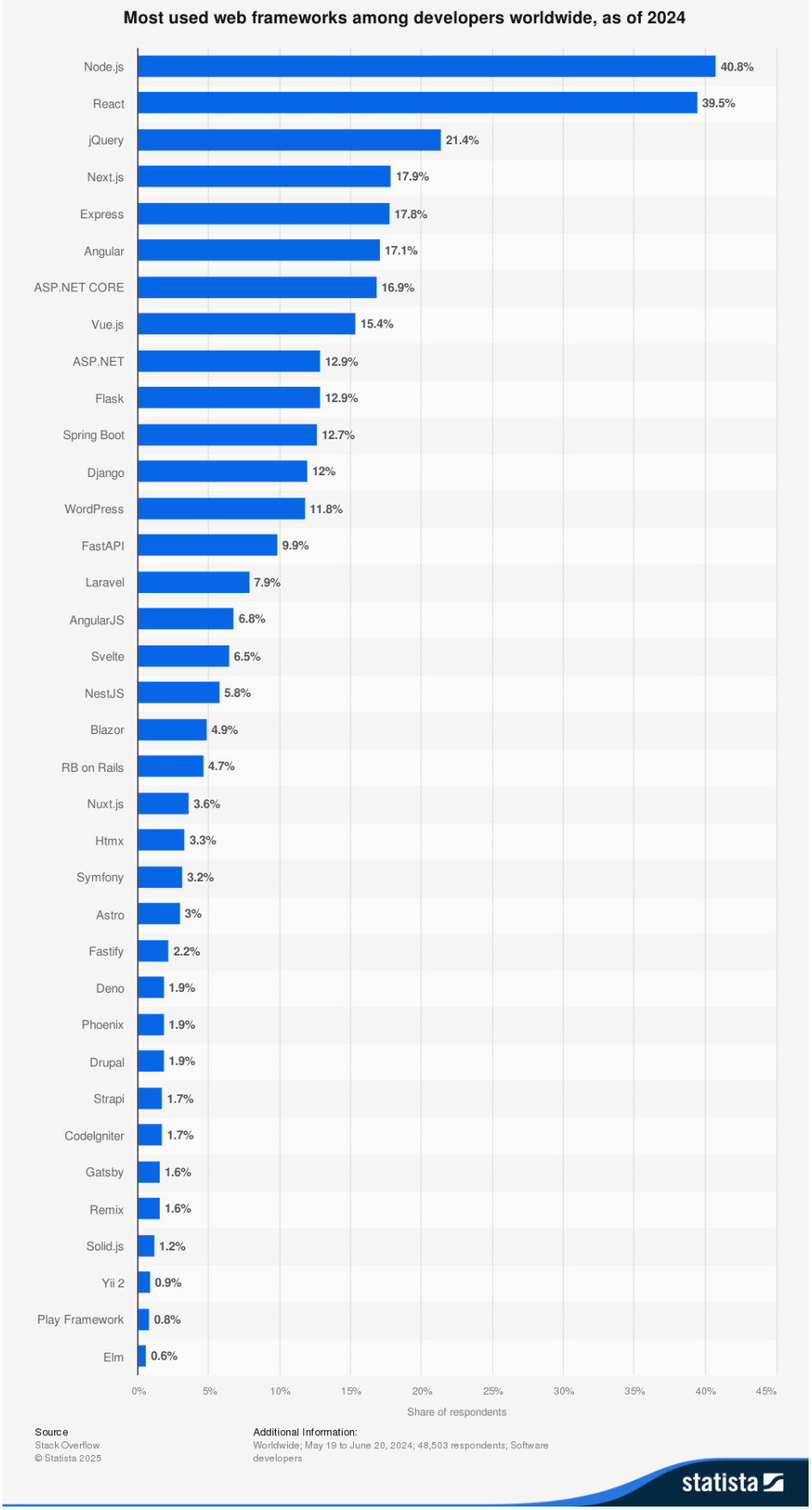
Το React είναι μια βιβλιοθήκη της JavaScript που αναπτύχθηκε από το Facebook το 2013, με σκοπό να λύσει το πρόβλημα της μη αποτελεσματικής ενημέρωσης της διεπαφής χρήστη σε μεγάλες δυναμικές εφαρμογές. Η μεγαλύτερη καινοτομία του React είναι η χρήση του εικονικού DOM ή αλλιώς Virtual Dom. Το εικονικό DOM είναι μια εικονική αναπαράσταση του DOM που επιτρέπει την αποτελεσματική και αποδοτική ενημέρωση των αλλαγών στο πραγματικό DOM, χωρίς να απαιτεί πλήρη ανανέωση της σελίδας.

Το React βασίζεται στην αρχιτεκτονική γύρω από την οργάνωση των δομικών στοιχείων, όπου η εφαρμογή δομείται και χωρίζεται σε μικρά και επαναχρησιμοποιήσιμα δομικά στοιχεία, τα οποία είναι ανεξάρτητα μεταξύ τους και είναι υπεύθυνα για την διαχείριση της κατάστασης state και την απόδοση της διεπαφής χρήστη. Αυτά τα δομικά στοιχεία μπορούν να συνδυαστούν και να αλληλεπιδράσουν μεταξύ τους για να δημιουργήσουν πιο σύνθετες διεπαφές, εξασφαλίζοντας παράλληλα την εμπειρία του χρήστη.

Η δυνατότητα επαναχρησιμοποίησης αυτών των δομικών στοιχείων καθιστά και συμβάλει στην αποδοτική ανάπτυξη και διαχείριση μεγάλων εφαρμογών, προσφέροντας ένα οργανωμένο περιβάλλον, αρθρωτό και επεκτάσιμο τρόπο σχεδιασμού. Επιπλέον, η προσέγγιση αυτή διευκολύνει τη συντήρηση του κώδικα, μειώνοντας σημαντικά την πολυπλοκότητα και ενισχύοντας αποτελεσματικά την επεκτασιμότητα της εφαρμογής. Το μεγαλύτερο της πλεονέκτημα είναι ότι τα δομικά στοιχεία μπορούν να αλλάξουν και να τροποποιηθούν χωρίς να επηρεάσουν τα υπόλοιπα γεγονός που οδηγεί σε μια πιο ευρεία υποστήριξη τόσο από τους πελάτες όσο και από τους παρόχους υπηρεσιών. [6]

Το React χρησιμοποιεί επίσης hooks, μια λειτουργία που εμφανίστηκε για πρώτη φορά στην έκδοση 16.8 η οποία προσφέρει την δυνατότητα χρήσης states και άλλων React χαρακτηριστικών χωρίς να απαιτείται η χρήση κλασικών δομικών στοιχείων. Χρησιμοποιείται στην εφαρμογή, για την δημιουργία και εμφάνιση των δομικών στοιχείων που απαρτίζουν την διεπαφή χρήστη, όπως είναι η μπάρα αναζήτησης, οι λίστες βιβλίων, η σελίδα με τις λεπτομέρειες κάθε βιβλίου αλλά και η ίδια εναλλαγή των σελίδων. [7]

Η χρήση του React μπορεί να μειώσει αποτελεσματικά τον χρόνο αλλά και το κόστος της ανάπτυξης της εφαρμογής. Μεγάλες εταιρείες όπως έχει προαναφερθεί είναι η Facebook , η Netflix και πολλές άλλες επιχειρήσεις που χρησιμοποιούν React λόγω των διαφόρων πλεονεκτημάτων που προσφέρει. Όταν η εφαρμογή απαιτεί σύνθετες αλληλεπιδράσεις, όπως είναι ενημερώσεις σε πραγματικό χρόνο ή σύνθετες φόρμες συμπλήρωσης. Η αρχιτεκτονική του React καθίσταται κατάλληλη για αρκετούς από αυτούς τους σκοπούς, παρέχει παράλληλα εργαλεία όπως είναι το React Router και το React Profiler και τον καταμερισμό του κώδικα για την βελτιστοποίηση της απόδοσης της εφαρμογής. [6]



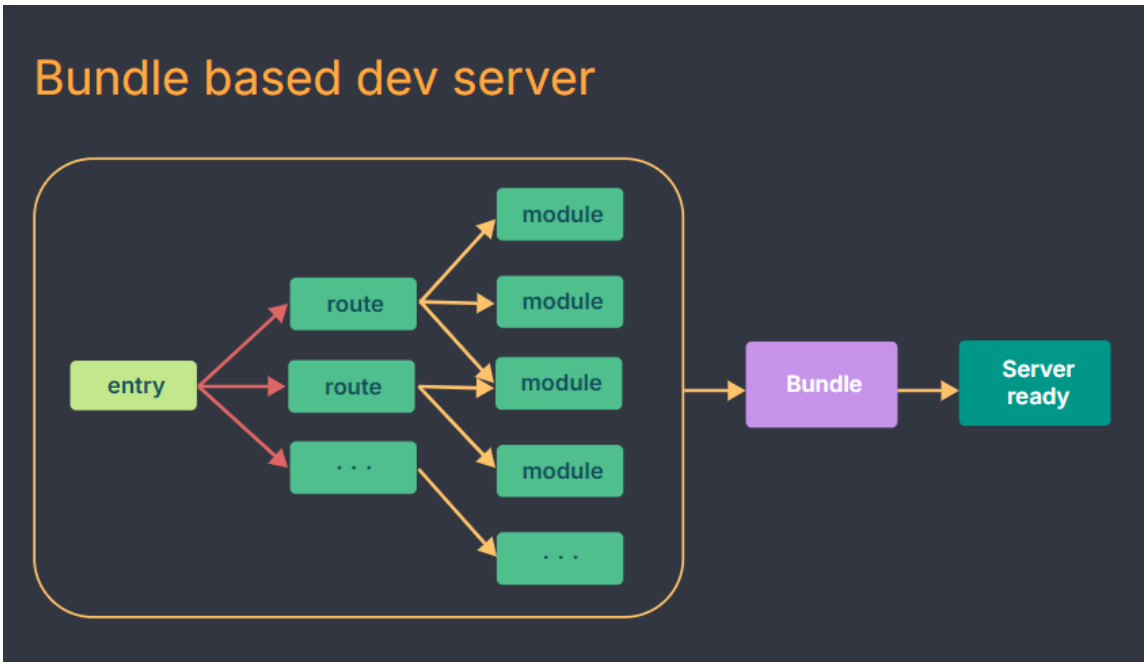
Σχήμα 2.2: React το πιο χρησιμοποιημένο διαδικτυακό σύστημα ανάπτυξης για το 2024.[8]

2.5 Ανάπτυξη του Vite

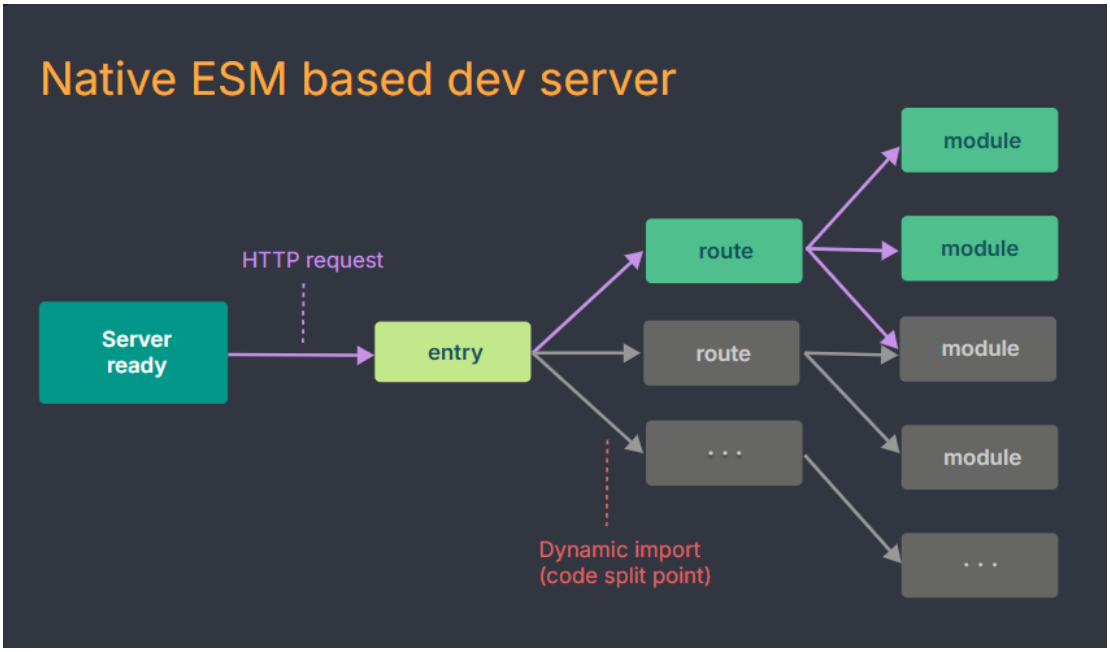
Το Vite είναι ένα JavaScript εργαλείο που αναπτύχθηκε από τον Evan You, πρόκειται για ένα σύγχρονο εργαλείο για την ανάπτυξη front-end εφαρμογών, όπου είναι και ο δημιουργός του Vue.js ένα προοδευτικό σύστημα ανάπτυξης JavaScript. Το Vite διαφέρει από τα παραδοσιακά εργαλεία ανάπτυξης, το οποίο επικεντρώνεται και έχει σαν κύριο χαρακτηριστικό την ταχύτητα και την αποδοτικότητα κατά τη διάρκεια της ανάπτυξης της εφαρμογής.

Όταν ξεκινάει η ανάπτυξη της εφαρμογής, ένα εργαλείο ανάπτυξης πρέπει να διαβάσει, να επεξεργαστεί και να δημιουργήσει ολόκληρη την εφαρμογή πριν να την "παρουσιάσει". Αυτού του είδος η διαδικασία μπορεί να απαιτεί αρκετό χρόνο, πόρους και υπολογιστική ισχύ. Αντίθετα το Vite βελτιώνει σημαντικά το χρόνο εκκίνησης του server χρησιμοποιώντας ESModules για τη φόρτωση των επιμέρους τμημάτων στον περιηγητή, επιτρέποντας με αυτόν τον τρόπο την επιπλέον ταχύτερη ανανέωση της σελίδας κατά την ανάπτυξη της. [9]

Η σημαντικότερη καινοτομία του Vite είναι η εξαιρετική του απόδοση στην ανάπτυξη του server. Όταν γίνεται αλλαγή στον κώδικα, το Vite ενημερώνει αποκλειστικά τα αρχεία που επηρεάζονται και σχετίζονται χωρίς να χρειάζεται να ξανά φορτωθεί από την αρχή ολόκληρη η σελίδα. Αυτό επιτυγχάνεται μέσω του χαρακτηριστικού Hot Module Replacement(HMR) που παρέχει. Το Vite, επομένως, είναι το εργαλείο που χρησιμοποιείται στην εφαρμογή εξασφαλίζοντας ταχύτητα, αποδοτικότητα και καινοτομία. [10]



Σχήμα 2.3: Διαδικασία ανάπτυξης του διακομιστή με δομή συγκέντρωσης. [9]



Σχήμα 2.4: Ο διακομιστής με ESModules για επαναφόρτωση και παρουσίαση επιμέρους τμημάτων. [9]

2.6 Συμπεράσματα

Η ανάλυση των τεχνολογιών που χρησιμοποιούνται για την ανάπτυξη της εφαρμογής φανερώνει τη σημαντική πρόοδο και εξέλιξη τους από την αρχική εμφάνιση της HTML το 1991 μέχρι την εμφάνιση εργαλείων όπως είναι το Vite και το React που διευκολύνουν τη δημιουργία σύγχρονων, δυναμικών εφαρμογών. Οι τεχνολογίες αυτές, σε συνδυασμό με τη JavaScript έχουν φέρει επανάσταση στην ανάπτυξη διαδικτυακών εφαρμογών και επιτρέπουν τη δημιουργία γρήγορων, αποδοτικών και διαδραστικών εμπειριών για τους χρήστες.

Η χρήση αυτών των εργαλείων στην εφαρμογή διασφαλίζει ότι η είναι σε θέση να ανταποκριθεί στις σύγχρονες και απαιτητικές ανάγκες του διαδικτύου, παρέχοντας στους χρήστες μια δυναμική και φιλική πλατφόρμα για τη διαχείριση βιβλίων.

Κεφάλαιο 3ο: Υλοποίηση του Συστήματος

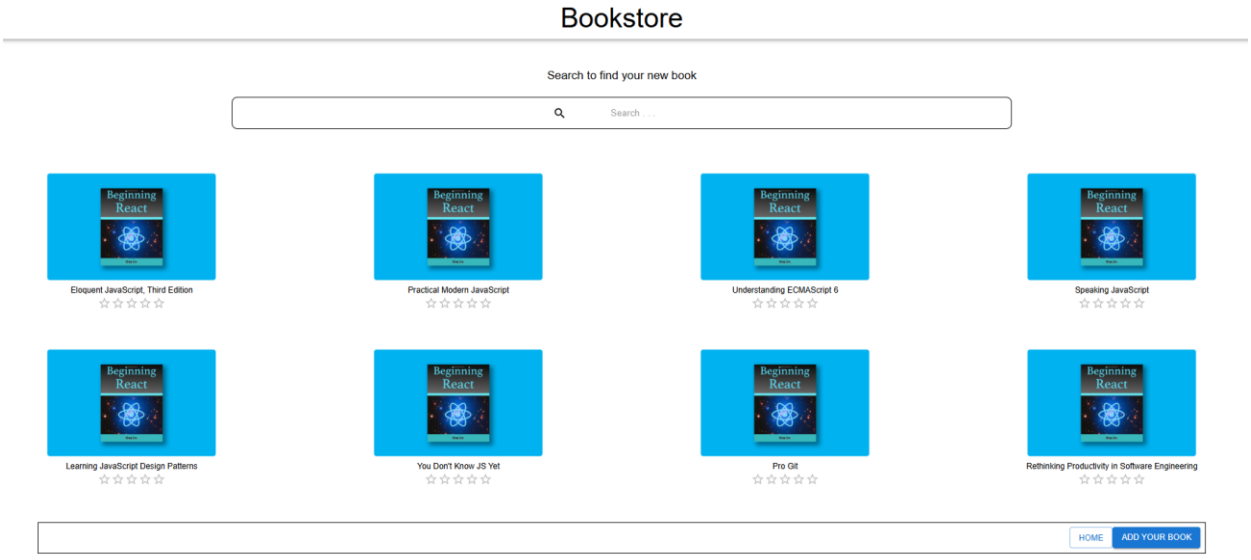
3.1 Ανάλυση της Υλοποίηση του Συστήματος

Ο σχεδιασμός του συστήματος εστιάζει στην ανάπτυξη μιας δυναμικής SPA εφαρμογής, η οποία διαχειρίζεται βιβλία και παρέχει πληροφορίες και λεπτομέρειες στους χρήστες. Η εφαρμογή χωρίζεται σε διάφορα επαναχρησιμοποιήσιμα δομικά στοιχεία που αλληλεπιδρούν και επικοινωνούν μεταξύ τους μέσω του state και των properties του React. Ο χρήστης μπορεί να αναζητήσει βιβλία με βάση τις επιθυμίες του, να καταχωρήσει νέα βιβλία και να μελετήσει τις λεπτομέρειες για το καθ' ένα βιβλίο μέσω μιας καθαρής και φιλικής ως προς τον χρήστη εφαρμογή.

Η αρχιτεκτονική του συστήματος είναι οργανωμένη γύρω από τα δομικά στοιχεία, με το καθένα να είναι υπεύθυνο για μια συγκεκριμένη λειτουργία και έχουν δημιουργηθεί με τέτοιο τρόπο ώστε να είναι δυναμικά και επαναχρησιμοποιήσιμα. Στην εφαρμογή χρησιμοποιούνται τα εξής δομικά στοιχεία:

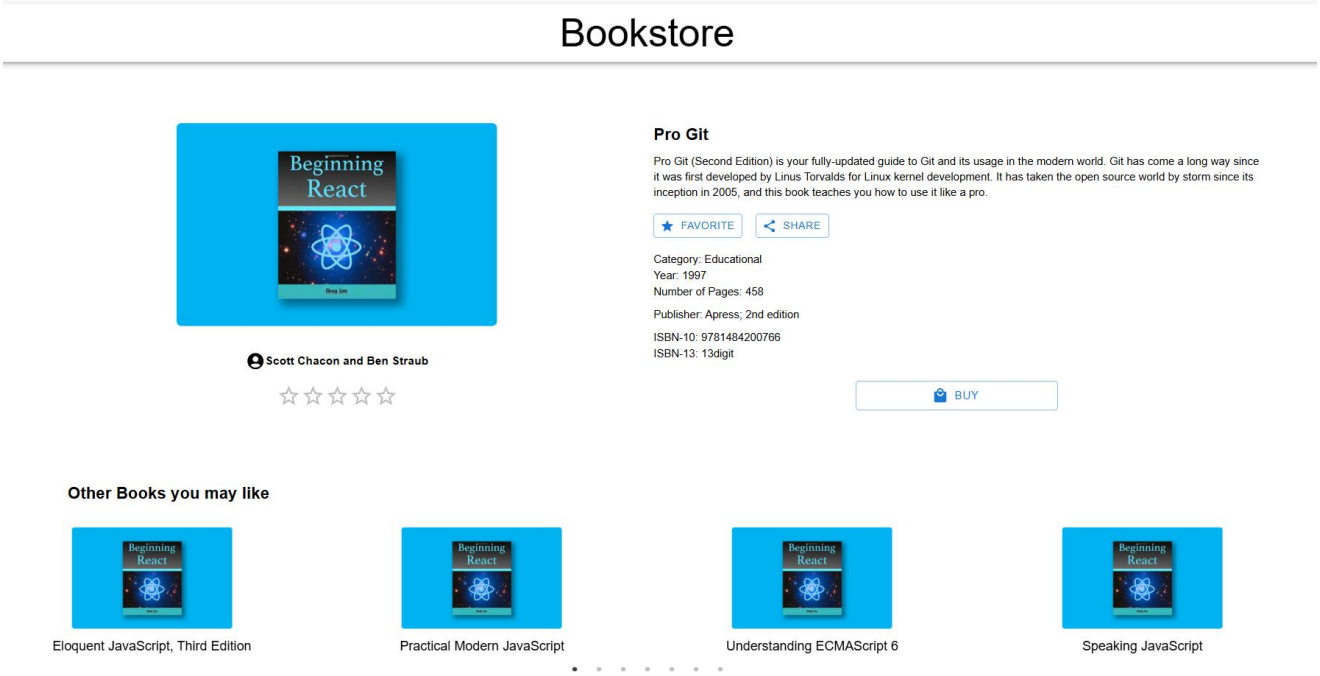
- **Library:** Αφορά την κεντρική σελίδα της εφαρμογής στην οποία ο χρήστης αναζητά τα βιβλία μέσω μιας δυναμικής και βέλτιστης δομικής μπάρα αναζήτησης. Το δομικό στοιχείο μπάρα αναζήτησης, επιτρέπει στον χρήστη να πληκτρολογεί με βάση τον τίτλο του βιβλίου, το όνομα του συγγραφέα, τα ISBN ή άλλων χαρακτηριστικών. Ο σχεδιασμός της αρχικής σελίδας Library ανταποκρίνεται αναλόγως, προσαρμόζοντας το περιεχόμενο σύμφωνα με την συσκευή που χρησιμοποιεί ο χρήστης. Κάθε βιβλίο που εμφανίζεται στην αρχική σελίδα της εφαρμογής έχει ιδιότητες και είναι σχεδιασμένο ως ένα κουμπί, το οποίο όταν επιλέγεται οδηγεί τον χρήστη στην αντίστοιχη σελίδα των λεπτομερειών του βιβλίου.

  Αυτό το χαρακτηριστικό επιτρέπει στους χρήστες να καθοδηγηθούν εύκολα και απλά στα κατοχυρωμένα βιβλία από την κεντρική σελίδα.



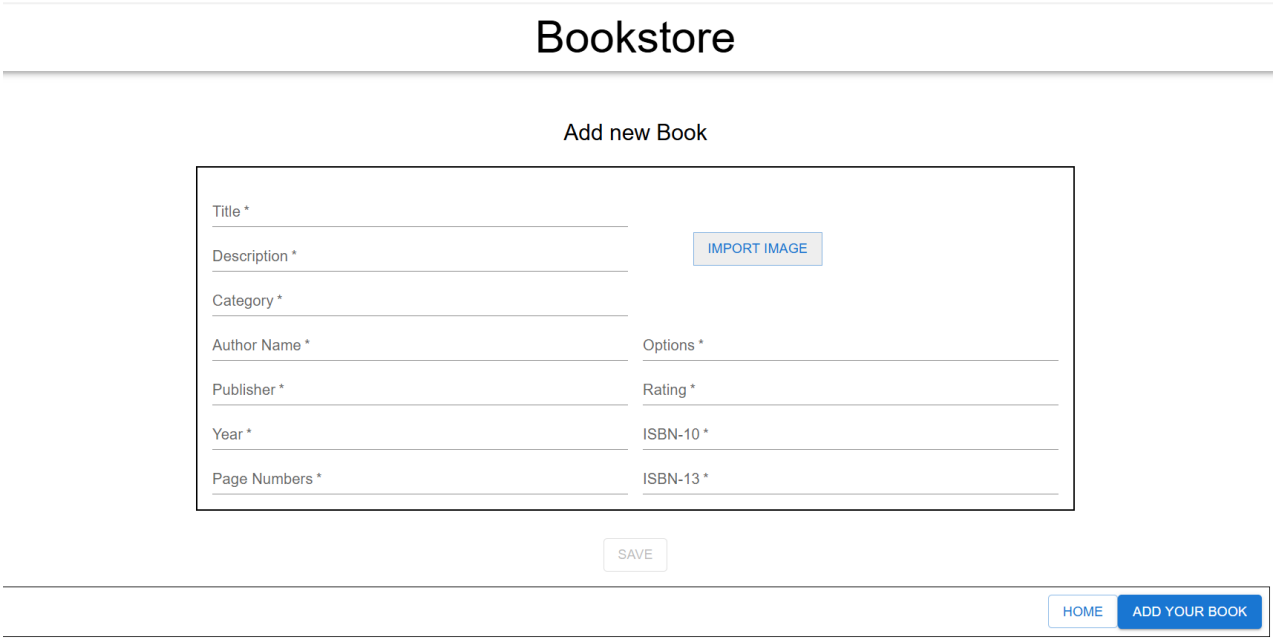
Σχήμα 3.1: Η αρχική σελίδα της εφαρμογής.

- **BookDetails:** Μετά την επιλογή ενός βιβλίου από την αρχική σελίδα, ο χρήστης μεταφέρεται στη σελίδα BookDetails. Στην συγκεκριμένη σελίδα παρουσιάζονται αναλυτικά όλες οι πληροφορίες και τα χαρακτηριστικά του επιλεγμένου βιβλίου όπως είναι ο τίτλος, ο συγγραφέας, η περιγραφή, η κατηγορία, η βαθμολογία, κλπ.



Σχήμα 3.2: Η σελίδα με τις λεπτομέρειες των βιβλίων.

- **NewBook:** Η σελίδα NewBook είναι η φόρμα καταχώρησης νέου βιβλίου όπου ο χρήστης συμπληρώνει τα κελιά τα οποία θα διαμορφώσουν τα χαρακτηριστικά και την δομή του βιβλίου που καταχωρεί. Η φόρμα περιλαμβάνει σημαντικά πεδία όπως είναι ο τίτλος του βιβλίου, η χρονολογία έκδοσης, τα μοναδικά ISBN, την κατηγορία, την βαθμολογία, κλπ. Επιπλέον, τα χαρακτηριστικά που είναι να καταχωρήσει ο χρήστης χρειάζεται να είναι κατάλληλα και επικυρωμένα προκειμένου να γίνουν αποδεκτά από το σύστημα. Αυτό διασφαλίζεται μέσω των βασικών ελέγχων επικύρωσης που εμφανίζονται στην φόρμα, με σκοπό την καθοδήγηση και την ενημέρωση του χρήστη για την σωστή συμπλήρωση των κελιών. Εφόσον η συμπλήρωση των στοιχείων είναι ορθή, ενεργοποιείται το κουμπί Save προκειμένου να ολοκληρωθεί η διαδικασία καταχώρησης του βιβλίου.



Σχήμα 3.3: Η σελίδα με την φόρμα καταχώρησης ενός βιβλίου.

- **Header:** Η κεφαλίδα της εφαρμογής με τον τίτλο “Bookstore”, το οποίο είναι η γραμμή εφαρμογής, ένα δομικό στοιχείο από την βιβλιοθήκη της Material-UI με ιδιότητες χαρτιού.

Σχήμα 3.4: Η κεφαλίδα των σελίδων.

- **Footer:** Το υποσέλιδο της εφαρμογής με ιδιότητες χαρτιού που παρέχει την δυνατότητα πλοήγησης στον χρήστη μεταξύ των σελίδων με την βοήθεια δομικών στοιχείων κουμπιών.

Σχήμα 3.5: Το υποσέλιδο των σελίδων.

Οι σελίδες αυτές επικοινωνούν μεταξύ τους χρησιμοποιώντας το React Router. Το React Router είναι μια βιβλιοθήκη JavaScript, για τη διαχείριση της ομαλής πλοήγησης, επιτρέποντας με αυτόν τον τρόπο στον χρήστη να μετακινείται μεταξύ διαφορετικών τμημάτων της εφαρμογής χωρίς να ανανεώνεται η σελίδα, αλλά μόνο το περιεχόμενο της.

3.2 Τεχνολογίες και Εργαλεία

Για την ανάπτυξη της εφαρμογής επιλέχθηκαν οι παρακάτω τεχνολογίες και εργαλεία λόγω των χαρακτηριστικών τους, τα οποία ταιριάζουν στις ανάγκες της εφαρμογής και στην πρόθεση να δημιουργηθεί μια γρήγορη, βέλτιστη αλλά και αποδοτική εμπειρία για τον χρήστη:

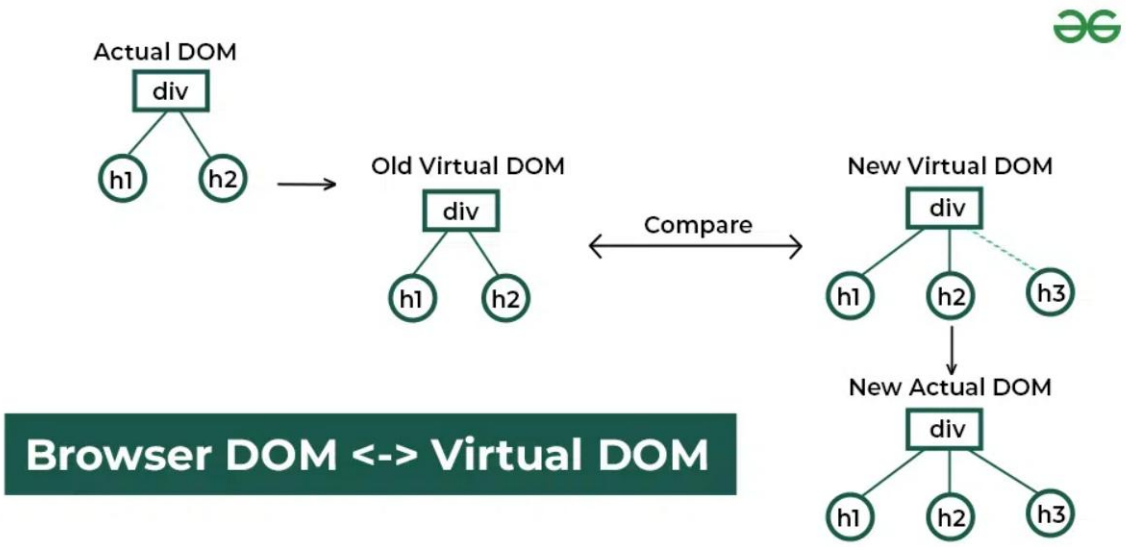
3.2.1 Τεχνολογίες του React

- **React:** Το React είναι η βιβλιοθήκη που επιλέχθηκε για την ανάπτυξη της εφαρμογής, καθώς επιτρέπει τη δημιουργία επαναχρησιμοποιημένων δομικών στοιχείων που διαχειρίζονται την εμφάνιση και αλληλεπίδραση με τον χρήστη. Τα πιο σημαντικά χαρακτηριστικά του React είναι το εικονικό DOM, η διαχείριση των state και τα properties που παρέχει στον προγραμματιστή.

- **Virtual DOM:** Το εικονικό DOM είναι ένα εικονικό αντίγραφο του πραγματικού DOM. Κάθε φορά που το state ή props ενός δομικού στοιχείου αλλάζει, το React δεν ανανεώνει αμέσως το πραγματικό DOM αλλά αντιθέτως, δημιουργεί ένα νέο εικονικό DOM και συγκρίνει τις διαφορές μεταξύ του παλιού και του νέου εικονικού DOM. Αυτή του είδος η διαδικασία ονομάζεται React Reconciliation που είναι υπεύθυνη για αυτή την σύγκριση που συμβαίνει μεταξύ δομικών στοιχείων και states. [11]

  Με βάση αυτή τη σύγκριση, το React υπολογίζει τα μέρη του πραγματικού DOM που χρειάζονται να ενημερωθούν και εφαρμόζει μόνο τις αναγκαίες και απαραίτητες αλλαγές, μειώνοντας έτσι με αυτόν τον τρόπο την ανάγκη για πλήρη ανανέωση της σελίδας. Αυτή η προσέγγιση βελτιώνει την αποδοτικότητα και την ταχύτητα της εφαρμογής. [12]

  Η χρήση του εικονικού DOM επιτρέπει στη εφαρμογή να είναι πολύ αποδοτικότερη. Αντί να ανανεώνεται το DOM πλήρως κάθε φορά που αλλάζει η κατάσταση της εφαρμογής όπως συμβαίνει και είναι σύνηθες με τις παραδοσιακές εφαρμογές, το React αντιθέτως ανανεώνει μόνο τα στοιχεία της διεπαφής χρήστη που έχουν τροποποιηθεί και επεξεργαστεί, εξοικονομώντας με αυτόν τον τρόπο χρόνο αλλά και πόρους.



Σχήμα 3.6: Η λειτουργεία του κανονικού DOM σε σύγκριση με το εικονικό του React.

Το εικονικό DOM είναι ένα κύριο χαρακτηριστικό της αποδοτικότητας του React. Χωρίς αυτό το χαρακτηριστικό, η εφαρμογή θα χρειαζόταν πολύ περισσότερους πόρους για την ανανέωση της διεπαφής χρήστη και δεν θα είχε την επιθυμητή απόδοση. Η δυνατότητα του React να ενημερώνει μόνο τις απαραίτητες περιοχές του DOM είναι ένας από τους λόγους που το React είναι τόσο δημοφιλές σύστημα ανάπτυξης μεταξύ προγραμματιστών.

Στην εφαρμογή, το εικονικό DOM βοηθάει στην γρήγορη εμφάνιση των αποτελεσμάτων αναζήτησης, όταν ο χρήστης πληκτρολογεί στην μπάρα αναζήτησης. Ανεξάρτητα από το πόσα βιβλία υπάρχουν στη βάση δεδομένων, το React θα ενημερώσει μόνο τα μέρη της διεπαφής χρήστη που επηρεάζονται από την αλλαγή, που στην προκειμένη περίπτωση επηρεάζεται σε πραγματικό χρόνο από την εισαγωγή κειμένου του χρήστη.

Παράδειγμα: Όταν ο χρήστης πληκτρολογεί στην αναζήτηση όπως αναφέρθηκε προηγουμένως, το state ενημερώνεται και το React χρησιμοποιεί το εικονικό DOM για να υπολογίσει τις αλλαγές. Αντί να ξαναφορτώσει όλα τα βιβλία από την αρχή, το React θα ενημερώσει μόνο τα σχετικά μέρη της σελίδας που επηρεάζονται από αυτήν την αλλαγή.

Πίνακας 3.1: Η Διαφορά του Εικονικού DOM της React σε σχέση με το Πραγματικό DOM.

Virtual DOM	**Real DOM**
Είναι μια ελαφριά έκδοση του πραγματικού DOM	Είναι μια δενδροειδής αναπαράσταση των HTML στοιχείων. Είναι η πραγματική αναπαράσταση της σελίδας στον φυλλομετρητή.
Διαχειρίζεται από βιβλιοθήκες JavaScript, όπως είναι η React.	Διαχειρίζεται από τον φυλλομετρητή μετά την ανάλυση των HTML στοιχείων μέσα στο DOM.
Μετά την επεξεργασία στοιχείων, δεν τροποποιείται ολόκληρο το DOM. Αντίθετα, μόνο τα τροποποιημένα μέρη ενημερώνονται στο πραγματικό DOM.	Κάθε αλλαγή στο DOM απαιτεί την ανασύνθεση ολόκληρου του DOM, ακόμα και στην τροποποίηση ενός στοιχείου.
Οι ενημερώσεις είναι ασύγχρονες και μόνο τα μέρη που χρειάζονται αλλαγή ενημερώνονται.	Οι ενημερώσεις στο πραγματικό DOM είναι συγχρονισμένες, που υποδηλώνει ότι κάθε αλλαγή των στοιχείων απαιτεί πλήρη ανανέωση του DOM.
Η απόδοση είναι γρήγορη.	Η απόδοση είναι αργή και χρειάζονται περισσότεροι πόροι, καθώς κάθε αλλαγή στο DOM οδηγεί σε πλήρη επαναφόρτωση των στοιχείων της σελίδας.
Πολύ αποτελεσματικό καθώς πραγματοποιεί μαζικές ενημερώσεις	Λιγότερο αποτελεσματικό λόγω των συνεχόμενων rendering.

- **React State:** Το state είναι ένα αντικείμενο που περιέχει δεδομένα που μπορεί να αλλάξουν κατά τη διάρκεια εκτέλεσης της εφαρμογής. Το state συνδέεται με την αλληλεπίδραση του χρήστη με την εφαρμογή και όταν πραγματοποιηθεί αλλαγή του, το React ενημερώνει την διεπαφή χρήστη αυτόματα.

  Είναι ιδιαίτερα χρήσιμο το state καθώς επιτρέπει στην εφαρμογή να αντιδρά σε αλλαγές δεδομένων σε πραγματικό χρόνο. Όταν το state αλλάζει, το React ανανεώνει την διεπαφή χρήστη χωρίς να χρειάζεται να φορτωθεί ξανά από την αρχή όλη η σελίδα. Είναι σημαντικά για τη λειτουργία οποιασδήποτε React εφαρμογής, καθώς διαχειρίζεται τα δεδομένα που αλλάζουν και επηρεάζουν την διεπαφή χρήστη. Είναι το κύριο εργαλείο που χρησιμοποιείται για την αποθήκευση και διαχείριση δυναμικών δεδομένων, όπως οι τιμές στις φόρμες συμπλήρωσης, τα αποτελέσματα αναζητήσεων.

  Το state επιτρέπει την ανανέωση της διεπαφής χρήστη όταν και όποτε αλλάζουν τα δεδομένα. Στην εφαρμογή η μπάρα αναζήτησης στο αρχείο Library χρησιμοποιεί το state για να αποθηκεύσει την αναζήτηση του χρήστη και να φιλτράρει τα βιβλία δυναμικά με βάση την εισαγωγή κειμένου του.

Στο παράρτημα του κώδικα της εφαρμογής για το state παρουσιάζεται αναλυτικά η επιρροή στο εικονικό DOM, στην ενέργεια του χρήστη να πληκτρολογήσει στο δομικό στοιχείο μπάρα αναζήτησης. Το state ενημερώνεται και το εικονικό DOM συγκρίνει τη νέα κατάσταση με την προηγουμένη, ενημερώνοντας μόνο τα μέρη της σελίδας που απαιτούν αλλαγή, όπως είναι τα αποτελέσματα της αναζήτησης.

- **React Properties:** Τα Properties ή αλλιώς props για την React είναι αντικείμενα που περιέχουν δεδομένα τα οποία μπορούν να περαστούν σε ένα δομικό στοιχείο από τον γονιό. Τα props επιτρέπουν στα δομικά στοιχεία να είναι πιο δυναμικά, καθώς μπορούν να διαχειρίζονται τα δεδομένα που τους δίνονται να τα επεξεργάζονται και να τα εμφανίζουν.

  Κάθε δομικό στοιχείο μπορεί να λάβει διαφορετικά props τα οποία είναι διαθέσιμα κυρίως για ανάγνωση από το ίδιο το δομικό στοιχείο που τα λαμβάνει. Σε περίπτωση που χρειάζεται να γίνει αλλαγή των δεδομένων των props, πρέπει να γίνει η αλλαγή στον γονιό του δομικού στοιχείου. [13]

Πίνακας 3.2: Η Διαφορά μεταξύ των Props και των States.

Props	**State**
Τα Props χρησιμοποιούνται για την μεταφορά δεδομένων από ένα δομικό στοιχείο σε ένα άλλο. Τα δεδομένα μεταδίδονται στα δομικά στοιχεία από τον πατέρα αυτών.	Το state αποθηκεύει δεδομένα μεταφέρονται μόνο εντός του δομικού στοιχείου που ορίζεται.
Είναι αμετάβλητα τα properties, δηλαδή δεν μπορούν να τροποποιηθούν μέσα στο δομικό στοιχείο που τα δέχεται.	Είναι μεταβλητά, δηλαδή μπορούν να τροποποιηθούν από το δομικό στοιχείο που το διαχειρίζεται.
Τα Props μπορούν να χρησιμοποιηθούν τόσο σε λειτουργικά δομικά στοιχεία όσο και σε δομικά στοιχεία κλάσεων.	Το state μπορεί να χρησιμοποιηθεί μόνο εντός του δομικού στοιχείου που ορίζεται. Βέβαια μέσω της διαδικασίας ανύψωσης state ή μέσω της χρήσης του γενικού πλαισίου μπορεί να μεταφερθεί και σε υπόλοιπα δομικά στοιχεία.
Τα Props είναι αποκλειστικά και μόνο για ανάγνωση.	Το state είναι για ανάγνωση αλλά και για εγγραφή.

3.2.2 Βιβλιοθήκη Vite

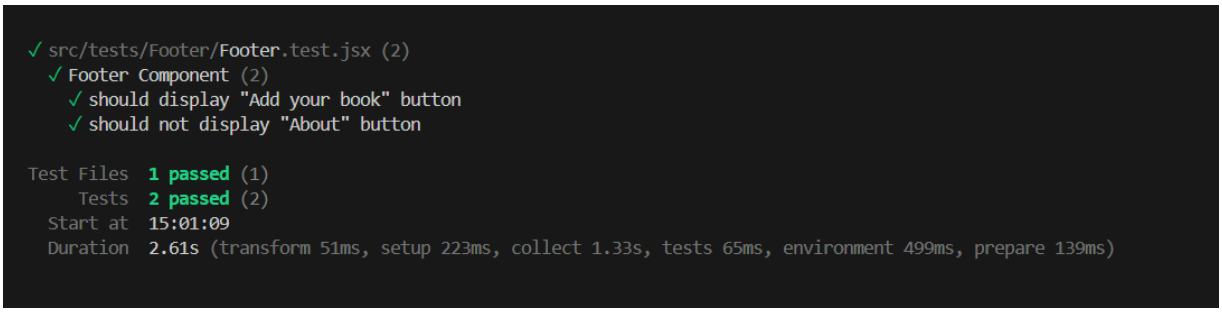
Το Vite είναι ένα σύγχρονο εργαλείο που χρησιμοποιείται για τη δημιουργία, την ανάπτυξη αλλά και την παραγωγή της εφαρμογής. Το Vite υποστηρίζει ταχύτερη ανάπτυξη μέσω της φόρτωσης αρχείων ES Modules και υποστηρίζει χαρακτηριστικά όπως είναι το Hot Module Replacement, το οποίο επιτρέπει την γρήγορη ενημέρωση της εφαρμογής χωρίς να χρειάζεται να γίνει ανανέωση όλο το περιβάλλον, καθώς η φόρτωση των επιμέρους τμημάτων γίνεται εξαιρετικά γρήγορα. Επιπλέον, διευκολύνει τους προγραμματιστές, γιατί μειώνει σημαντικά τους χρόνους ανανέωσης σε κάθε αλλαγή του κώδικα καθώς προσφέρει γρήγορη κατασκευή της εφαρμογής. [14]

3.2.3 Βιβλιοθήκη Vitest

Το Vitest είναι ένα σύγχρονο εργαλείο για την δημιουργία και εκτέλεση δοκιμών στην εφαρμογή. Πρόκειται για ταχύτατη και ευέλικτη αναπαραγωγή ελέγχων που χρησιμοποιείται κυρίως για την εκτέλεση δοκιμών κώδικα και λειτουργειών σε JavaScript και React εφαρμογές, προσφέροντας μια εξαιρετική εμπειρία προγραμματιστή. Επιπλέον, επιτρέπει τη συγγραφή και εκτέλεση δοκιμών σε περιβάλλον JavaScript Document Object Model(JSDOM). Παρέχει ενσωματωμένες δυνατότητες ψεύτικων δεδομένων για την προσομοίωση εξωτερικών υπηρεσιών και χρησιμοποιεί worker threads για να εκτελεί τις δοκιμές ταυτόχρονα, μειώνοντας τον χρόνο τους. Παράλληλα, έχει ενεργοποιημένο από προεπιλογή την λειτουργία παρατηρητή, όπου οι δοκιμές εκτελούνται αυτόματα με κάθε αποθήκευση του κώδικα, χωρίς να χρειάζεται να επαναληφθεί η διαδικασία εκτέλεσης τους.

Η ρύθμιση του Vitest διασφαλίζει ότι οι δοκιμές πραγματοποιούνται σε περιβάλλον React και υποστηρίζει την ασύγχρονη εκτέλεση των δοκιμών με API αιτήματα όπως είναι η ανάκτηση δεδομένων είτε είναι από έναν απομακρυσμένο διακομιστή ή είτε είναι από έναν τοπικό. [15]

Στο παράρτημα του κώδικα για την δοκιμή της εφαρμογής, παρουσιάζεται με λεπτομέρεια η υλοποίηση της σουίτας δοκιμής. Πραγματοποιείται έλεγχος μέσα στον κώδικα του υποσέλιδου δομικού στοιχείου με σκοπό να ελέγξει την ορθότητα και την λειτουργικότητα του. Τρέχοντας την δοκιμή στον τερματικό με την κατάλληλη εντολή, εμφανίζεται το αποτέλεσμα της.



Σχήμα 3.7: Το αποτέλεσμα της δοκιμής.

3.2.4 Ρυθμίσεις Vite and Vitest

- **React Plug-in:** Χρησιμοποιούμε τα επιμέρους τμήματα της βιβλιοθήκης Vite για τη σωστή υποστήριξη του συστήματος ανάπτυξης React στην εφαρμογή.

- **TEST Configuration:** Ορίζουμε το περιβάλλον JSDOM, το οποίο επιτρέπει τη δοκιμή του κώδικα React σαν να γίνεται η εκτέλεση του σε περιβάλλον περιήγησης. Επίσης, έχει ενεργοποιηθεί η υποστήριξη για CSS και έχει καθοριστεί το JavaScript setup αρχείο για τη ρύθμιση των βιβλιοθηκών όπως για παράδειγμα είναι το @testing-library/react.

- **Globals:** Ενεργοποιούμε τις παγκόσμιες μεταβλητές για τους ελέγχους όπως είναι το describe και το it.

- **JSDOM:** Ορίζουμε το JSDOM για την εκτέλεση των δοκιμών σε περιβάλλον φυλλομετρητή.

Το Vite κάνει την ανάπτυξη πιο γρήγορη και αποτελεσματική, επιτρέποντας ταχύτερη ανανέωση της σελίδας και στοχεύει στην καλύτερη εμπειρία του χρήστη.

Το JavaScript setup αρχείο είναι υπεύθυνο για την ρύθμιση των βιβλιοθηκών των δοκιμών πριν από την εκτέλεση των ελέγχων και τον καθαρισμό του DOM μετά από κάθε δοκιμή, χρησιμοποιώντας τη μέθοδο καθαρισμού από την βιβλιοθήκη δοκιμής.

Στο παράρτημα του κώδικα αναφέρεται αναλυτικά το αρχείο με τις ρυθμίσεις του Vite και Vitest προκειμένου να διασφαλιστούν οι κατάλληλες ρυθμίσεις για να τρέξουν αποτελεσματικά τα εργαλεία.

3.2.5 Βιβλιοθήκη Mock Service Worker

Το Mock Service Worker(MSW) είναι ένα εργαλείο που χρησιμοποιείται για την προσομοίωση αιτημάτων API και την παροχή ψεύτικων δεδομένων κατά τη διάρκεια ανάπτυξης της εφαρμογής. Αυτή η βιβλιοθήκη επιτρέπει στην εφαρμογή να λειτουργεί με ψεύτικα δεδομένα χωρίς να απαιτεί εξωτερικούς διακομιστές. Η διαμόρφωση για το MSW περιλαμβάνει την δημιουργία των χειριστών που προσομοιώνουν τα API, παρέχοντας τα δεδομένα από τα αρχεία JSON με ψεύτικα δεδομένα κατά τη διάρκεια της ανάπτυξης. Αυτό διασφαλίζει ότι η εφαρμογή μπορεί να ανακτά δεδομένα και να τα αξιοποιεί χωρίς να είναι αναγκαίες οι εξωτερικές εξαρτήσεις όπως είναι για παράδειγμα μια βάση δεδομένων.

Ο χειριστής του MSW είναι υπεύθυνος για τη ρύθμιση και διαχείριση API αιτημάτων και την συμπεριφορά που θέλουμε να προσομοιώσουμε στην εφαρμογή. Κάθε χειριστής καθορίζει τη λειτουργία που θα εκτελείται όταν γίνεται ένα αίτημα σε ένα συγκεκριμένο API άκρο. Με άλλα λόγια, ο χειριστής δηλώνει τον τρόπο που θα ενεργήσει ο διακομιστής με τα ψεύτικα δεδομένα που τρέχει όταν γίνει ένα αίτημα και τα δεδομένα που θα επιστρέψει όταν γίνει η ολοκλήρωση του αιτήματος. [16]

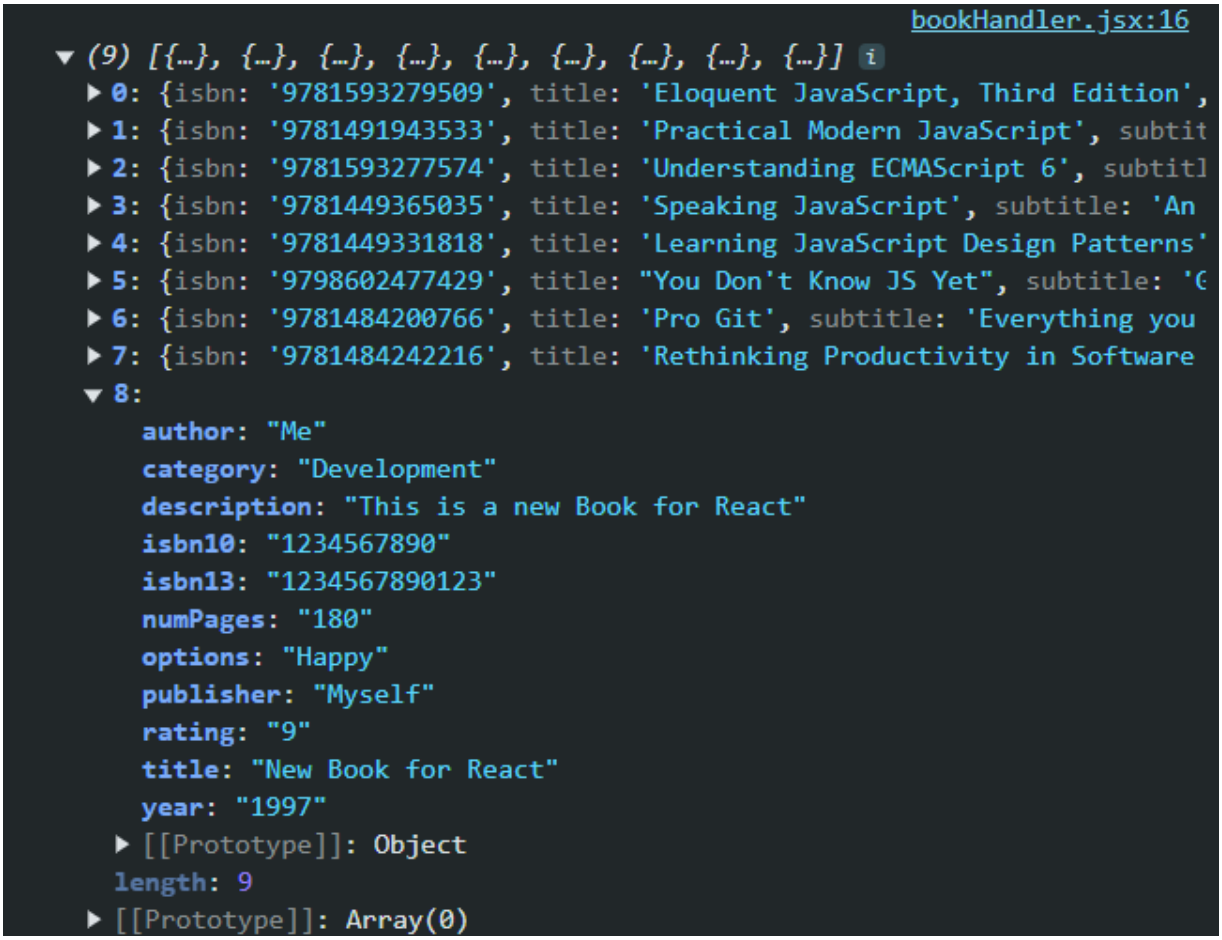
Οι χειριστές είναι κρίσιμοι και σημαντικοί για την επιτυχής ανάπτυξη του MSW, καθώς επιτρέπουν την προσομοίωση της επικοινωνίας με το πίσω μέρος των εφαρμογών και της βάσης δεδομένων χωρίς να χρειάζεται πραγματική σύνδεση με έναν εξωτερικό διακομιστή ή με μια απομακρυσμένη βάση δεδομένων.

Παράδειγμα: Η προσομοίωση που αναφέρεται στο παράρτημα του κώδικα της εφαρμογής είναι η ρύθμιση που πραγματοποιήθηκε με τον χειριστή υπεύθυνο για τα βιβλία, ο οποίος είναι αρμόδιος για τον χειρισμό και επεξεργασία των ψεύτικων δεδομένων των βιβλίων. Με λίγα λόγια, ορίζει τον τρόπο επεξεργασίας των αιτημάτων που γίνονται σε ένα API όπως είναι τα αιτήματα GET για την ανάκτηση δεδομένων και POST για να στείλει δεδομένα στον διακομιστή. Πιο συγκεκριμένα:

- Το GET αίτημα για το τεχνικό άκρο "/api/books" επιστρέφει τη λίστα βιβλίων από το BooksData αρχείο.

- Το POST αίτημα επιτρέπει την προσθήκη νέου βιβλίου στην ήδη υπάρχουσα λίστα βιβλίων μετά την έγκυρη καταχώρηση του.

Ο συγκεκριμένος αυτός χειριστής που αναλύθηκε και παρουσιάζεται στο παράρτημα, διαχειρίζεται τα αιτήματα που αφορούν την ανάκτηση και προσθήκη βιβλίων με την βοήθεια προσομοίωσης ενός τεχνικού API μέσω της βιβλιοθήκης MSW. Επιπλέον, έχει χρησιμοποιηθεί για να συλλέξει τα δεδομένα από ένα JSON αρχείο με ψεύτικα δεδομένα χωρίς να απαιτείται η σύνδεση με πραγματικό διακομιστή.



Σχήμα 3.8: Επιτυχής εισαγωγή βιβλίου και προσθήκη στο αρχείο JSON.

- **MSW Worker Setup:** Ο λόγος που χρησιμοποιούμε τον spread τελεστή για τον χειριστή στο MSW είναι για να εξασφαλίσουμε ότι μπορούμε να περάσουμε πολλαπλούς χειριστές στον ψεύτικο διακομιστή με τον ίδιο τρόπο, επιτρέποντας έτσι την επέκταση του ψεύτικου server για νέα API άκρα χωρίς να χρειάζεται να γίνει αλλαγή στο ήδη υπάρχον σύστημα.

  Ο spread τελεστής, μας επιτρέπει να περάσουμε πολλαπλές καταχωρίσεις από έναν μόλις πίνακα με χειριστές στον ψεύτικο διακομιστή. Αυτό κρίνεται ιδιαίτερα σημαντικό διότι, επιτρέπει την εύκολη και ομαλή προσθήκη νέων χειριστών χωρίς να τροποποιούμε τον κώδικα του worker κάθε φορά και διευκολύνει την διαχείριση πολλαπλών API άκρων, διαχωρίζοντας αποτελεσματικά την λογική της επεξεργασίας αιτημάτων.

Στο παράρτημα του κώδικα αναφέρεται η διαμόρφωση και ενεργοποίηση του Mock Service Worker στην εφαρμογή. Ο σκοπός αυτού του αρχείου είναι η ρύθμιση και το ξεκίνημα του MSW, το οποίο θα αναλάβει να καταγράψει τα αιτήματα HTTP κατά την εκτέλεση της εφαρμογής.

Αρχικά, έχει χρησιμοποιηθεί ο spread τελεστής, με σκοπό να περαστούν όλοι οι χειριστές από τον πίνακα στο setupWorker του MSW. Αυτός ο τρόπος, επιτρέπει την επέκταση του διακομιστή με νέους χειριστές όπως είναι για παράδειγμα άλλα άκρα, χωρίς να χρειάζεται να αλλάξουμε τη δομή του κώδικα για καθ' ένα καινούργιο αίτημα.

Το MSW παρέχει τη δυνατότητα να προσομοιώσουμε αιτήματα και να αναπτύσσουμε την εφαρμογή χωρίς την ανάγκη σύνδεσης με πραγματικά API. Τα πλεονεκτήματα του MSW είναι τα εξής:

1. **Ανεξαρτησία από εξωτερικές υπηρεσίες:** Δίνει την δυνατότητα ανάπτυξης και δοκιμής της εφαρμογή χωρίς να εξαρτάται και να παρέμβουν εξωτερικοί βάσεις δεδομένων.

2. **Γρήγορη και αποτελεσματική ανάπτυξη:** Δημιουργεί και προσομοιώνει API αιτήματα με πολύ γρήγορο τρόπο μέσα από την βοήθεια και χρήση τοπικών ψεύτικων δεδομένων.

3. **Ευκολία στην δοκιμή της εφαρμογής:** Το MSW προσφέρει την δυνατότητα να γίνεται έλεγχος της εφαρμογής στις αλληλεπιδράσεις με τα δεδομένα χωρίς να περιμένουμε απόκριση από εξωτερικούς διακομιστές.

4. **Αξιοπιστία:** Μειώνει δραστικά τον κίνδυνο αποτυχιών που σχετίζονται με εξωτερικές API υπηρεσίες επιτρέποντας με αυτόν τον τρόπο τον εντοπισμό σφαλμάτων στην εφαρμογή.

Καταλήγοντας, το MSW είναι ιδανικό εργαλείο για την ανάπτυξη και δοκιμή της εφαρμογής, καθώς προσομοιώνει αιτήματα API και επιτρέπει στην διαχείριση δεδομένων, χωρίς να περιμένουμε την απόκριση από μια εξωτερική πραγματική υπηρεσία. Είναι ιδιαίτερα χρήσιμο να αναφέρουμε πως μπορεί να γίνει η προσομοίωση των δεδομένων τοπικά χωρίς να γίνει σύνδεση εξωτερικών βάσεων δεδομένων, ενώ ταυτόχρονα παρέχει δυνατότητες για την προσομοίωση πιο περίπλοκων αιτημάτων και σφαλμάτων που θα εμφανιζόντουσαν κατά την διάρκεια ανάπτυξης της εφαρμογής.

3.2.6 Βιβλιοθήκη Stale While Revalidate

Το Stale While Revalidate ή αλλιώς SWR είναι μια βιβλιοθήκη που χρησιμοποιείται για την εύκολη και αποδοτική ανάκτηση των δεδομένων. Ένας από τους βασικούς λογούς που επιλέχθηκε η συγκεκριμένη βιβλιοθήκη είναι η υποστήριξη στην κατηγοριοποίηση και στην επικύρωση των δεδομένων. Αυτό έχει ως αποτέλεσμα όταν ο χρήστης ζητάει δεδομένα από τον διακομιστή, όταν τα δεδομένα είναι ήδη αποθηκευμένα στην μνήμη cache, η βιβλιοθήκη τα επιστρέφει αμέσως προσφέροντας μια γρήγορη εμπειρία για τον χρήστη.

Χρησιμοποιώντας το SWR τα δεδομένα αποθηκεύονται τοπικά στην μνήμη cache του φυλλομετρητή. Αυτό έχει ως κύριο σκοπό να μειώνει δραστικά το χρόνο ανάκτησης των δεδομένων ως προς τον διακομιστή με αποτέλεσμα ο χρήστης να μην είναι αναγκασμένος να περιμένει για κάθε αίτημα και να τον διακόπτει. Επιπλέον, προσφέρει το χαρακτηριστικό revalidateIfStale, ανανεώνοντας αυτόματα τα δεδομένα της εφαρμογής. Σε περίπτωση που ο διακομιστής είναι εκτός λειτουργίας ή αδυνατεί να επιστρέψει δεδομένα, το SWR θα προσπαθήσει ξανά αυτόματα να ανακτήσει τα δεδομένα μετά από κάποια χρονική περίοδο. Αυτό το χαρακτηριστικό διασφαλίζει στον χρήστη ότι η εφαρμογή παραμένει σε συνεχής λειτουργία και παρέχει την καλύτερη δυνατή εμπειρία ακόμα και μετά από μια αδρανής περίοδο. Είναι ένα χαρακτηριστικό που λείπει από αρκετές εφαρμογές γιατί εξασφαλίζει ότι τα δεδομένα της εφαρμογής συνεχώς είναι ενημερωμένα και δεν περιμένει η εφαρμογή από τον χρήστη να πραγματοποιήσει κάποιο αίτημα, αντιθέτως φροντίζει να γίνεται η παρακολούθηση των δεδομένων αυτόματα σε περίπτωση που αλλάξουν. [17]

Η χρήση των ψεύτικων δεδομένων που επιλέχθηκαν για την ανάπτυξη της εφαρμογής, χρησιμοποιήθηκαν από ένα αποθετήριο στο GitHub. Τα ψεύτικα βιβλία είναι της μορφής αρχείου JSON τα οποία έχουν όλα τα απαραίτητα χαρακτηριστικά προκειμένου να χρησιμοποιήσει η εφαρμογή.[18]

- **Διαχείριση Σφάλματος κατά την Ανάκτηση Δεδομένων:** Σε περίπτωση που παρουσιαστεί σφάλμα κατά την ανάκτηση των δεδομένων, η εφαρμογή χρησιμοποιεί την λειτουργία του SWR για να εμφανίσει το κατάλληλο μήνυμα σφάλματος. Το μήνυμα παρέχει σαφή ενημέρωση για την κατάσταση των δεδομένων προκειμένου να γίνει κατανοητή η πηγή του προβλήματος.

- **Διαχείριση Αναμονής Ανάκτησης Δεδομένων:** Σε περίπτωση που τα δεδομένα δεν έχουν εμφανιστεί και δεν έχουν ακόμη ανακτηθεί, η εμφάνιση κατάλληλου μηνύματος είναι σημαντικό, καθώς παρέχει στο χρήστη οπτική ένδειξη ότι η εφαρμογή δεν είναι αδρανής, αντιθέτως χρειάζεται χρόνο για να ανακτήσει τα δεδομένα.

Με αυτούς τους μηχανισμούς έχει διασφαλιστεί ότι κάθε στιγμή ο χρήστης έχει συνεχομένη ενημέρωση της τρέχουσας κατάστασης της εφαρμογής. Προσφέρουν σαφήνεια για την εξέλιξη των δεδομένων και παραμένει διαδραστική και εύχρηστη ακόμη και σε περιπτώσεις που η επικοινωνία με τον διακομιστή ενδέχεται να έχει προβλήματα. Βελτιώνουν την σωστή ανατροφοδότηση του χρήστη για το πρόβλημα. Στο παράρτημα του κώδικα παρουσιάζεται η διαμόρφωση και η λειτουργία ενός μέρους του κώδικα που περιλαμβάνει την χρήση της βιβλιοθήκης SWR για την ασύγχρονη ανάκτηση δεδομένων, καθώς και των προηγούμενων μηχανισμών που αναφέρθηκαν.

3.2.7 Βιβλιοθήκη Material UI

Το Material UI είναι μια βιβλιοθήκη η οποία προσφέρει δομικά στοιχεία διεπαφής χρήστη βασισμένα στο Material Design της Google. Η επιλογή της προήλθε από την ανάγκη της εφαρμογής να προσφέρει ευχρηστία στον χρήστη αλλά και ταυτόχρονα να παραμείνει συμβατή με τις τελευταίες τάσεις που αφορούν το σχεδιαστικό κομμάτι των εφαρμογών.

Παρέχει μια καλαίσθητη σχεδίαση που εναρμονίζεται με τις αρχές του Material Design, το οποίο διασφαλίζει ότι η εφαρμογή θα έχει επαγγελματικό χαρακτήρα σε όλες τις συσκευές που ενδέχεται να χρησιμοποιήσει ο χρήστης. Αυτό σημαίνει ότι τα δομικά στοιχεία ανταποκρίνονται και προσαρμόζονται αυτόματα σε διαφορά μεγέθη αναλόγως την ανάλυση της οθόνης, δίνοντας στον χρήστη την βεβαιότητα ότι θα απολαύσει μια καθαρότερη και πιο ομοιόμορφη εμπειρία όσο πλοηγείται στην εφαρμογή.

Η απλότητα κατά την διάρκεια χρήσης, καθώς και τα στοιχεία που προσφέρει μπορούν εύκολα να τροποποιηθούν και να επεξεργαστούν. Σκοπός αυτός είναι η ικανοποίηση των αναγκών του προγραμματιστή και της εφαρμογής χωρίς να χρειάζεται να γίνει η ανάπτυξη των δομικών στοιχείων από την αρχή. Η βιβλιοθήκη προσφέρει τη δυνατότητα στον προγραμματιστή μέσω των εντολών της CSS να αναπτύξει μια δημιουργική εμφάνιση στην εφαρμογή ελκύοντας και διευκολύνοντας τον χρήστη.

Σημαντικό είναι ότι η κοινότητα γύρω από την βιβλιοθήκη Material UI είναι αρκετά μεγάλη και ενεργή με αποτέλεσμα μονίμως να αναπτύσσεται και να εξελίσσεται. Παρέχει αναβαθμίσεις συνεχώς, κάτι που την καθιστά αξιόπιστη και με αυτόν τον τρόπο ενισχύεται η αίσθηση της ασφάλειας για τους προγραμματιστές που την επιλέγουν για την ανάπτυξη εφαρμογών. Τέλος, είναι σημαντικό να αναφερθεί πως οι προγραμματιστές μπορούν να βρουν εύκολες και γρήγορες λύσεις στα προβλήματα που ενδέχεται να προκύψουν κατά την διάρκεια ανάπτυξης της εφαρμογής. [19]

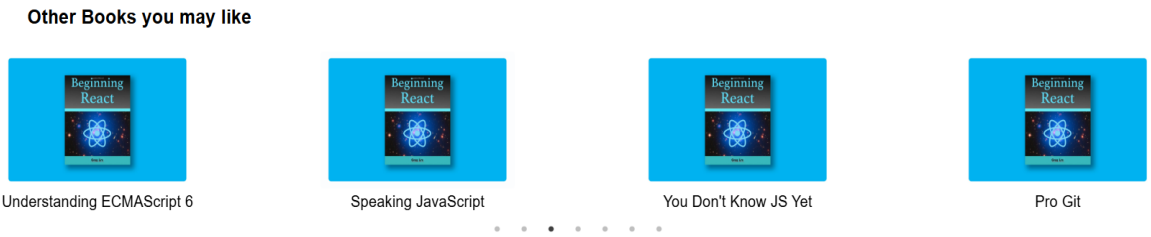
3.3 Σημαντικά Χαρακτηριστικά της Εφαρμογής για την Διεπαφή Χρήστη

Υπάρχουν ιδιαίτερα σημαντικά χαρακτηριστικά στην εφαρμογή που ενισχύουν την αλληλεπίδραση, την λειτουργικότητα και την εμπειρία του χρήστη. Αυτά τα χαρακτηριστικά είναι σχεδιασμένα για να προσφέρουν στον χρήστη μια πλήρη και αποτελεσματική διεπαφή με την εφαρμογή τα οποία είναι ενσωματωμένα σε αρκετές διάσημες εφαρμογές όπως είναι το Facebook, το Twitter, το Netflix κλπ.

3.3.1 Βιβλιοθήκη React Slick Carousel

Ένα από τα κύρια χαρακτηριστικά της εφαρμογής είναι η βιβλιοθήκη React Slick Carousel που δίνει την δυνατότητα προβολής προτεινόμενων βιβλίων. Είναι ένα από τα δημοφιλέστερα εργαλεία για την υλοποίηση του συγκεκριμένου τμήματος στη React, το οποίο προσφέρει στο χρήστη την δυνατότητα ομαλής εμφάνισης και πλοήγησης των βιβλίων με δυνατότητα κύλισης, χωρίς να απαιτεί την ανανέωση της σελίδας.

Χρησιμοποιείται στην εφαρμογή, στην σελίδα που σχετίζεται με τις λεπτομέρειες των βιβλίων. Σε αυτή τη σελίδα, μια ιδιαίτερη και σημαντική λειτουργία είναι ότι προτείνει βιβλία με βάση κάποια κοινά στοιχεία με το βιβλίο που εμφανίζεται και το εξαιρεί από το Carousel προκειμένου να αποφευχθεί η σύγχυση που ενδεχομένως να προκύψει από την πλευρά του χρήστη. Με αυτό το τρόπο διασφαλίζεται η παρουσίαση στο χρήστη, βιβλίων που δεν βλέπει ήδη στην σελίδα με τις λεπτομέρειες. Επιπλέον, κάθε εικόνα έχει τις ιδιότητες από ένα κουμπί που μπορεί να συνεχίσει τη πλοήγηση του χρήστη απευθείας στις λεπτομέρειες του επομένου βιβλίου που θα επιλέξει. Επιτρέπεται με αυτό τον τρόπο η γρήγορη εναλλαγή και πλοήγηση στην εφαρμογή.

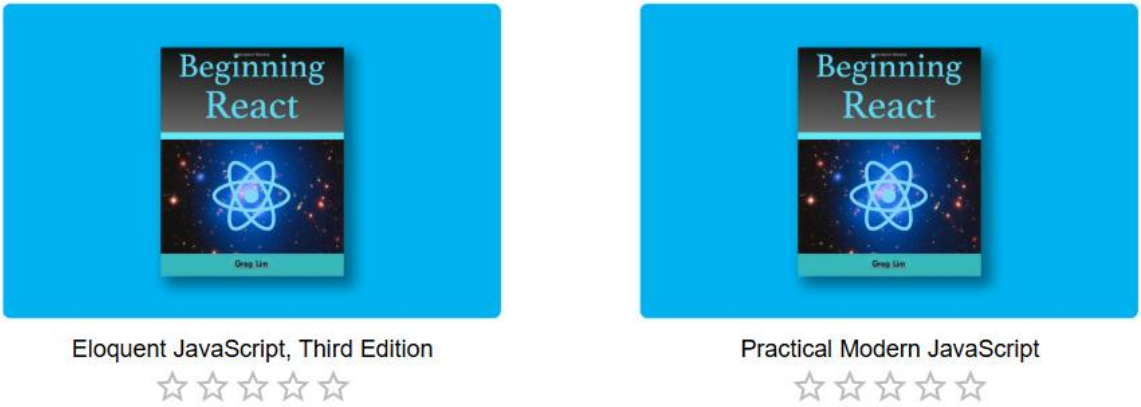


Σχήμα 3.11: Αναπαράσταση του Carousel στην σελίδα λεπτομερειών.

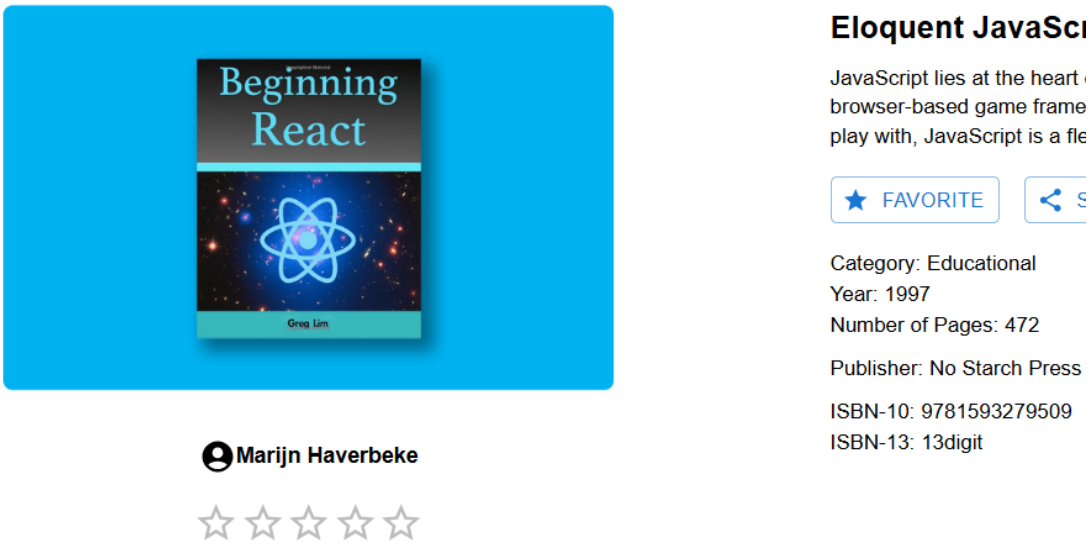
3.3.2 Ανάγνωση Αξιολογήσεων (Read Only Rating)

Ένα ακόμη χαρακτηριστικό της εφαρμογής είναι η πρόσβαση του χρήστη στο να παρατηρεί την αξιολόγηση των βιβλίων χωρίς να του δίνεται η δυνατότητα να την τροποποιήσει. Με αυτόν τον τρόπο μπορεί να παρατηρεί την υπάρχουσα αξιολόγηση ενός βιβλίου η οποία έχει οριστεί κατά την εισαγωγή του στο σύστημα.

Η εφαρμογή χρησιμοποιεί το δομικό στοιχείο Rating ή αλλιώς αξιολόγηση, της βιβλιοθήκης Material UI, το οποίο βρίσκεται σε διαφορετικά σημεία της εφαρμογής όπως είναι η αρχική σελίδα και η σελίδα με τις λεπτομέρειες των βιβλίων.



Σχήμα 3.12: Η αξιολόγηση στην Αρχική Σελίδα.



Σχήμα 3.13: Η αξιολόγηση στην σελίδα λεπτομερειών.

3.3.3 Προσαρμοσμένα Κουμπιά–Εικονίδια και Χρώματα

Τα κουμπιά και τα εικονίδια που χρησιμοποιούνται στην εφαρμογή είναι βασισμένα στα πρότυπα της βιβλιοθήκης Material UI και έχουν προσαρμοστεί με σκοπό να ταιριάζουν απόλυτα στο συνολικό ύφος της εφαρμογής.

Χρησιμοποιούνται για να εκτελούν τις ενέργειες και τις επιθυμίες του χρήστη όπως είναι η υποβολή φόρμας, η πλοήγηση μεταξύ των σελίδων, η επιλογή αγαπημένου βιβλίου και η κοινοποίηση του.

Επιπλέον περιλαμβάνει κουμπιά με ιδιότητες εφέ κατά την αιώρηση και κουμπιά που αλλάζουν χρώματα προκειμένου να είναι ξεκάθαρο στο χρήστη και να αλληλεπιδρά με ευκολία καθώς πλοηγείται στην εφαρμογή.

Σχήμα 3.14: Κουμπιά με χρήση εφέ και όμορφων σχεδιασμών.

Η επιλογή των χρωμάτων στην εφαρμογή ήταν ιδιαίτερα σημαντική για την δημιουργία μιας σοβαρής και επαγγελματικής εμπειρίας χρήστη. Επιλέχθηκε με βάση αυτά τα κριτήρια το λευκό φόντο, το οποίο παρέχει μια καθαρή εμφάνιση ενώ ταυτόχρονα δημιουργεί μια απαλή αντίθεση με τα υπόλοιπα στοιχεία της εφαρμογής, κάνοντας την εφαρμογή πιο ευχάριστη στο μάτι και ευανάγνωστη για τον χρήστη.

Το μπλε χρώμα από την άλλη που έχει επιλεχθεί για τα υπόλοιπα στοιχεία της εφαρμογής είναι το κύριο χρώμα με βάση τα πρότυπα της Ευρωπαϊκής Ένωσης, το οποίο συμβολίζει σταθερότητα και εμπιστοσύνη. Χρησιμοποιείται κυρίως για να δημιουργεί μια απαλή αντίθεση με το λευκό φόντο και να προσφέρει καλύτερη οπτική εμφάνιση στα σημαντικά στοιχεία της εφαρμογής όπως είναι τα εικονίδια και τα κουμπιά ενεργειών.

Σχήμα 3.15: Αντίθεση χρωμάτων λευκού-μπλε.

3.4 Συμπεράσματα

Σε αυτό το κεφάλαιο, έγινε αναφορά στην διαδικασία ανάπτυξης και υλοποίησης της εφαρμογής, η οποία δημιουργήθηκε για να παρέχει μια σύγχρονη, ευχάριστη και αξιόπιστη εμπειρία στον τομέα της διαχείρισης, εξερεύνησης αλλά και καταχώρησης βιβλίων. Η εφαρμογή βασίζεται στο περιβάλλον React για να ακολουθήσει την αρχιτεκτονική Single Page Application σε συνεργασία με το σύστημα ανάπτυξης Vite για γρήγορη ανάπτυξη και πλοήγηση της εφαρμογής, χωρίς να γίνεται ανανέωση της σελίδας.

Καθώς είναι Single Page Application η εφαρμογή έχει ανάγκη την βιβλιοθήκη React Router, η οποία επιτρέπει και δίνει την δυνατότητα στους χρήστες να πλοηγηθούν εύκολα μεταξύ των σελίδων της εφαρμογής και να εξερευνήσουν τις σελίδες όπως είναι η φόρμα καταχώρησης καινούργιων βιβλίων, η προβολή λεπτομερειών των βιβλίων αλλά και η αρχική σελίδα που είναι η ανασκόπηση όλων των βιβλίων. [20]

Προκειμένου η εφαρμογή να είναι λειτουργική, κρίθηκε σημαντική η προσομοίωση των API αιτημάτων με την βοήθεια της βιβλιοθήκης Mock Service Worker. Με αυτόν τον τρόπο η εφαρμογή δεν έχει την ανάγκη σύνδεσης με εξωτερικούς διακομιστές, παρέχοντας μεγαλύτερη ευελιξία κατά την διαδικασία ανάπτυξης της.

Καθώς, γίνεται προσομοίωση των API αιτημάτων, η χρήση του εργαλείου Stale While Revalidate ήταν απαραίτητη με σκοπό την ασύγχρονη ανάκτηση ψεύτικων δεδομένων. Επιτρέπεται η εμφάνιση των δεδομένων άμεσα και η παράλληλη ανανέωση τους στο παρασκήνιο, χωρίς την αναμονή του χρήστη. Με λίγα λόγια τα δεδομένα είναι ήδη διαθέσιμα χρησιμοποιώντας την μνήμη cache και επιστρέφει αμέσως τα δεδομένα στον χρήστη δίνοντας την δυνατότητα να αλληλεπιδρά με την εφαρμογή χωρίς καθυστερήσεις. Σε περίπτωση μεταβολής των δεδομένων εκτελεί την ανάκτηση δεδομένων και το εργαλείο Mock Service Worker ανανεώνει την cache αυτόματα με τα πιο πρόσφατα δεδομένα.

Προκειμένου, όλες αυτές οι διαδικασίες να γίνουν αντιληπτές από το χρήστη και τον ίδιο τον προγραμματιστή και να συνυπάρχουν, χρησιμοποιήθηκε η βιβλιοθήκη Material UI με σκοπό να ανταπεξέλθει στις απαιτήσεις της εφαρμογής, προσφέροντας ένα όμορφο περιβάλλον στην εφαρμογή. Επιπλέον, ο συνδυασμός των καινοτόμων χαρακτηριστικών και των επαγγελματικών προδιαγραφών ενισχύουν την αλληλεπίδραση με το χρήστη με αποτέλεσμα να αποκτά η εφαρμογή σοβαρότερο ύφος που αναδεικνύει την αξία και ποιότητα της εφαρμογής.

Παρουσιάστηκε μια γρήγορη και συνοπτική ανάλυση της διαδικασίας ανάπτυξης της εφαρμογής στην οποία έγινε η ανάλυση των λειτουργιών που χρησιμοποιήθηκαν για την ομαλή και πλήρη υλοποίηση της. Η εφαρμογή είναι επεκτάσιμη σε τυχόν μελλοντική εξέλιξη της, εύκολα μπορεί κανείς να την κατανοήσει και προσφέρει μια άριστη εμπειρία αλληλεπίδρασης. Τέλος, δεν εξυπηρετεί μόνο τις ανάγκες των χρηστών, αλλά ενσωματώνει και σύγχρονες τεχνολογίες και ευρωπαϊκά πρότυπα που τη θέτουν σε υψηλό επίπεδο.

Κεφάλαιο 4ο: Αρχιτεκτονική και Στρατηγική Υλοποίησης Εφαρμογής

4.1 Αρχιτεκτονική

Στο παρόν κεφάλαιο γίνεται η ανάλυση της αρχιτεκτονικής της εφαρμογής, παρέχοντας αναλυτική περιγραφή των εργαλείων και τεχνολογιών που χρησιμοποιήθηκαν. Επίσης, εξετάζονται οι εκδόσεις των εργαλείων και βιβλιοθηκών που χρησιμοποιήθηκαν και η συνεισφορά τους στη βελτίωση της απόδοσης της λειτουργικότητας της εφαρμογής.

4.1.1 Εκδόσεις Τεχνολογιών και Εργαλείων

Η επιλογή των κατάλληλων εργαλείων και τεχνολογιών ήταν σημαντική για την ανάπτυξη της εφαρμογής. Η χρήση των πιο πρόσφατων εκδόσεων διασφάλισε τη συμβατότητα και τη λειτουργικότητα της εφαρμογής. Παρακάτω γίνεται αναφορά αυτών που συντέλεσαν και χρησιμοποιήθηκαν για την ανάπτυξη:

- **React:** Η έκδοση 18.2.0 της React επιλέχθηκε λόγω της υποστήριξης για ταυτόχρονη απόδοση (Concurrent Rendering), η οποία βελτιώνει την απόδοση της εφαρμογής, καλύτερη διαχείρισης του state και της κατάστασης διεπαφής χρήστη.

- **Vite:** Η έκδοση 5.2.0 του Vite χρησιμοποιήθηκε για τη γρήγορη ανάπτυξη της εφαρμογής αλλά και την υποστήριξη του HMR, καθώς προσφέρει αρκετά γρήγορη ανανέωση των επιμέρους τμημάτων κατά την ανάπτυξης της.

- **MSW:** Η έκδοση 2.2.14 του MSW χρησιμοποιήθηκε για την προσομοίωση αιτημάτων API και την παροχή ψεύτικων δεδομένων στην εφαρμογή. Ιδιαίτερο χαρακτηριστικό είναι ότι διασφαλίζει η εφαρμογή μπορεί να λειτουργεί ανεξάρτητα από εξωτερικές υπηρεσίες και επιλέχθηκε η συγκεκριμένη έκδοση καθώς υποστηρίζεται από την React 18.

- **SWR:** Η έκδοση 2.2.5 του SWR χρησιμοποιήθηκε για την ανάκτηση δεδομένων στην εφαρμογή, καθώς ο μηχανισμός αυτός προσφέρει την ασύγχρονη ανάκτηση δεδομένων ενώ παράλληλα επιτρέπει τη συνεχή ανανέωση δεδομένων και την άμεση εμφάνιση των πιο προσφάτων αποτελεσμάτων. Ιδιαίτερο χαρακτηριστικό της είναι η δυνατότητα εκ νέου αναγνώρισης που επιτρέπει την εμφάνιση δεδομένων σε γρήγορο χρόνο με συνεχή επικοινωνία με τον διακομιστή και σε περίπτωση μεταβολής των δεδομένων τα διατηρεί πάντα ενημερωμένα.

- **Vitest:** Η έκδοση 1.5.3 του Vitest χρησιμοποιήθηκε για την εκτέλεση δοκιμών και την διασφάλιση της ακεραιότητας των λειτουργιών της εφαρμογής. Η συγκεκριμένη έκδοση επιλέχθηκε, γιατί υποστηρίζει δοκιμές σε περιβάλλον React και είναι συμβατή με τις τεχνολογίες που χρησιμοποιούνται όπως είναι το SWR και το MSW.

- **Material UI:** Η έκδοση 5.15.15 του Material UI χρησιμοποιήθηκε για την ανάπτυξη της διεπαφής του χρήστη. Προσφέρει απλοϊκότητα, δίνει την δυνατότητα στον προγραμματιστή να προσαρμόσει το περιβάλλον στις ανάγκες του και αρκετοί συνεργάτες ανοιχτού κώδικα έχουν συμβάλει στην εξέλιξη αυτής της βιβλιοθήκης.

```
"name": "react-project",
"private": true,
"version": "0.0.0",
"type": "module",
▷Debug
"scripts": {
  "dev": "vite",
  "build": "vite build",
  "lint": "eslint . --ext js,jsx --report-unused-disable-directives --max-warnings 0",
  "preview": "vite preview",
  "test": "vitest"
},
Start Xray scan
"dependencies": {
  "@emotion/react": "^11.11.4",
  "@emotion/styled": "^11.11.5",
  "@mui/icons-material": "^5.15.15",
  "@mui/material": "^5.15.15",
  "@testing-library/dom": "^10.1.0",
  "@types/jest-axe": "^3.5.9",
  "chai": "^5.1.0",
  "chai-react": "^4.0.0",
  "react": "^18.2.0",
  "react-dom": "^18.2.0",
  "react-responsive-carousel": "^3.2.23",
  "react-router-dom": "^6.22.3",
  "react-slick": "^0.30.2",
  "slick-carousel": "^1.8.1",
  "swiper": "^11.1.1",
  "swr": "^2.2.5"
},
```

Σχήμα 4.1: Οι εκδόσεις από τις τεχνολογίες και βιβλιοθήκες.

4.3 Συμπεράσματα

Πραγματοποιήθηκε η ανάλυση των βασικών σταδίων της αρχιτεκτονικής της εφαρμογής εστιάζοντας στις τεχνολογίες και στα εργαλεία που χρησιμοποιήθηκαν και αλληλοεπιδρούν μεταξύ τους. Η επιλογή των εργαλείων και των τεχνολογιών ήταν καθοριστική για την ομαλή λειτουργία της εφαρμογής. Οι εκδόσεις που επιλέχθηκαν με βάση την ιστοσελίδα και το πρόγραμμα Node Package Manager for JavaScript(NPMJS) που εξασφάλισε την συμβατότητα και την υψηλή απόδοση του συστήματος. [21]

Το διάγραμμα ροής της εφαρμογής παρουσιάζει συνοπτικά την αλληλουχία των ενεργειών από την στιγμή που ο χρήστης αλληλεπιδρά με την εφαρμογή μέχρι και την εμφάνιση των αποτελεσμάτων. Με αυτόν τον τρόπο κάθε βήμα της διαδικασίας είναι σαφές και κατανοητό. Τέλος, αναδεικνύεται το πλάνο που ακολουθήθηκε προκείμενου να υλοποιηθεί η εφαρμογή, εξηγώντας τα εργαλεία, τις τεχνολογικές και τις αρχιτεκτονικές αποφάσεις που λήφθηκαν καθ' όλη την ανάπτυξη της εφαρμογής.

Κεφάλαιο 5ο: Συζήτηση και Μελλοντική Έρευνα

5.1 Λειτουργικότητα

Η εφαρμογή "Bookstore" είναι ένα πρωτότυπο και σύγχρονο εργαλείο για την καταχώρηση και την εύκολη αναζήτηση βιβλίων με βασικά χαρακτηριστικά και λειτουργίες που εξυπηρετούν τον χρήστη για την διαχείριση βιβλιοπωλείου. Παρά την ολοκλήρωση της και την επιτυχία της στη βάση των απαιτήσεων της πτυχιακής, υπάρχουν αρκετά σημεία που μπορεί να βελτιωθούν, να επεκταθούν προς το καλύτερο ή ακόμα και να αλλάξουν προκειμένου να προσφέρουν μια πιο πλούσια και λειτουργική εμπειρία.

5.2 Θετικά Σημεία της Εφαρμογής

1. **Δυναμική Εφαρμογή με React:** Η χρήση του React για τη δημιουργία μιας SPA εφαρμογής επιτρέπει στον χρήστη να περιηγηθεί στην εφαρμογή χωρίς να χρειάζεται η σελίδα να φορτώσει από την αρχή. Η αλλαγή στις πληροφορίες όπως είναι η αναζήτηση βιβλίων ή την εμφάνιση των λεπτομερειών ενός βιβλίου, πραγματοποιείται χωρίς ανανέωση προσφέροντας μια ομαλότερη εμπειρία.

2. **Μαζική χρήση του state και των props:** Ο σωστός χειρισμός του state και των props καθιστά την εφαρμογή δυναμική και ελαστική. Το state επιτρέπει τη δυναμική αμφίδρομη επικοινωνία με την διεπαφή χρήστη, διασφαλίζοντας ότι ενημερώνεται κάθε φορά που αλλάζουν τα δεδομένα κατά την αλληλεπίδραση του χρήστη όπως είναι για παράδειγμα η αναζήτηση βιβλίων. Αντίθετα, τα properties μεταφέρουν δεδομένα μεταξύ του πατέρα του δομικού στοιχείου και του παιδιού δομικού στοιχείου επιτρέποντας την ευέλικτη παραμετροποίηση. Με αυτόν τον τρόπο επιτυγχάνεται η εύκολη επαναχρησιμοποίηση του κώδικα σε διαφορετικά μέρη της εφαρμογής κάνοντας την εύκολα διαχειριστική, καθώς το κάθε δομικό στοιχείο μπορεί να διαχειριστεί και να παρουσιάσει μόνο τις πληροφορίες που χρειάζεται χωρίς να έχει πρόσβαση σε όλη την κατάσταση του συστήματος.

3. **Χρήση του MSW για ψεύτικα δεδομένα:** Το Mock Service Worker έχει αποδειχθεί ιδιαίτερα χρήσιμο σε περιβάλλοντα ανάπτυξης, μειώνοντας τις καθυστερήσεις που ενδέχεται να υπάρξουν από τη σύνδεση με πραγματικούς διακομιστές επιτρέποντας παράλληλα τη δοκιμή της εφαρμογής με σταθερά δεδομένα. Η δυνατότητα προσομοίωσης των αιτημάτων API με το MSW εξασφαλίζει στον προγραμματιστή την σωστή και ελεγχόμενη δοκιμή της εφαρμογής, καθιστώντας την μια πλήρη ανεξάρτητη front-end εφαρμογή από εξωτερικούς διακομιστές.

4. **Χρήση του SWR για την ανάκτηση δεδομένων:** Το Stale While Revalidate χρησιμοποιείται για την ασύγχρονη ανάκτηση δεδομένων στην εφαρμογή. Ο μηχανισμός αυτός επιτρέπει στην εφαρμογή να επιστρέψει άμεσα τα δεδομένα από την μνήμη cache, με αυτήν την στρατηγική επιτρέπει στους χρήστες να παρατηρούν αμέσως τα δεδομένα και ταυτόχρονα να ανανεώνονται στο παρασκήνιο. Το SWR προσφέρει λειτουργίες για την διαχείριση σφαλμάτων και επιτρέπει την εύκολη και απλή ανανέωση των δεδομένων χωρίς να επηρεάζει την εμπειρία του χρήστη και να διαταράσσεται η λειτουργία της εφαρμογής.

5. **Χρήση των Υλικών του Material UI για εύκολη και σαφή σχεδίαση**: Η βιβλιοθήκη Material UI προσφέρει έτοιμα δομικά στοιχεία με μοντέρνα σχεδίαση. Η γραμμή εφαρμογής, το υποσέλιδο για την πλοήγηση , τα κουμπιά, τα κείμενα εισαγωγής αλλά και οι πίνακες είναι έτοιμοι για χρήση, μειώνοντας τον χρόνο ανάπτυξης και βελτιώνοντας την αισθητική και την λειτουργικότητα της εφαρμογής.

5.3 Αρνητικά Σημεία της Εφαρμογής

1. **Προβλήματα με το MSW:** Ένα σημαντικό πρόβλημα που υπάρχει με το MSW είναι ότι "ξεκινάει από την αρχή" επαναφέροντας τα δεδομένα από το αρχείο JSON. Αυτό σημαίνει ότι τα δεδομένα που έχουν προστεθεί ή τροποποιηθεί όπως είναι για παράδειγμα νέα βιβλία που προστίθενται μέσω της φόρμας καταχώρησης από τον χρήστη, χάνεται όταν ο worker κάνει επανεκκίνηση. Ενώ είναι αρκετά χρήσιμο εργαλείο για την ανάπτυξη, μπορεί να αποτελέσει περιορισμό για την παρακολούθηση της προόδου ή της συντήρησης των δεδομένων κατά τη διάρκεια της δοκιμής.

2. **Διαχείριση των Δεδομένων και Έλλειψη Βάσης Δεδομένων:** Η εφαρμογή χρησιμοποιεί το αρχείο JSON για την αποθήκευση των βιβλίων. Ωστόσο, η έλλειψη μιας πραγματικής βάσης δεδομένων σημαίνει ότι δεν υπάρχει μόνιμη αποθήκευση ή δυνατότητα ενημέρωσης των δεδομένων σε περίπτωση επανεκκίνησης του διακομιστή ή απώλειας δεδομένων. Ένας πιθανός τρόπος βελτίωσης αυτού θα ήταν η ενσωμάτωση μιας βάσης δεδομένων.

5.4 Μελλοντική Έρευνα και Επεκτάσεις

Προσθήκη Συγκρίσεων Βιβλίων: Η εφαρμογή θα μπορούσε να επεκταθεί με τη δυνατότητα να προσφέρει στον χρήστη την σύγκριση βιβλίων. Η προσθήκη αυτής της δυνατότητας θα επέτρεπε στους χρήστες να συγκρίνουν διάφορα βιβλία με βάση παραμέτρους και φίλτρα όπως είναι η αξιολόγηση, ο αριθμός σελίδων, τα περιεχόμενα αλλά και τις κατηγορίες των βιβλίων. Αυτό θα μπορούσε να γίνει με τη χρήση ενός state για την αποθήκευση των επιλεγμένων βιβλίων και την εμφάνιση τους σε μια σελίδα σύγκρισης.

Χρήση του React Hook Form για τη Διαχείριση της Φόρμας καταχώρησης βιβλίων: Ενώ η τρέχουσα φόρμα καταχώρησης βιβλίων λειτουργεί με τα σωστά επικυρωμένα πεδία συμπλήρωσης, θα ήταν δυνατή η βελτίωση της με τη χρήση της βιβλιοθήκης React Hook Form. Η συγκεκριμένη βιβλιοθήκη διευκολύνει τη διαχείριση της φόρμας, την επικύρωση των πεδίων συμπλήρωσης και την

αποστολή δεδομένων χωρίς την ανάγκη συνεχούς παρακολούθησης του state και των props. Αυτή η τεχνική μπορεί να μειώσει σε σημαντικό βαθμό την πολυπλοκότητα του κώδικα και να προσφέρει καλύτερη απόδοση.

Φιλτραρισμένη Αναζήτηση: Η δυνατότητα προσθήκης φίλτρων στην αναζήτηση είναι μια μελλοντική αναβάθμιση που θα βελτίωνε τη λειτουργικότητα της εφαρμογής. Η εφαρμογή δίνει την δυνατότητα αλλά και να επιτρέπει στους χρήστες να αναζητήσουν τα βιβλία βάση όλα τα χαρακτηριστικά που διαθέτουν, όμως θα ήταν μια ιδιαίτερη αναβάθμιση να είναι ξεκάθαρό ως προς τον χρήστη. Με λίγα λόγια να δίνεται η δυνατότητα στο χρήστη να ομαδοποίηση τα βιβλία ανά κατηγορία, ανά συγγραφέα, ανά έτος έκδοσης, αξιολόγησης ή ακόμα και με βάση τον αριθμό σελίδων του βιβλίου με την βοήθεια ενός φίλτρου αναζήτησης.

Διαχωρισμός Πελάτη και Διαχειριστή: Μια σημαντική βελτίωση για την εφαρμογή είναι ο διαχωρισμός του πελάτη από τον διαχειριστή του βιβλιοπωλείου, μέσω της προσθήκης ενός συστήματος ταυτοποίησης. Στόχος αυτής της λειτουργίας είναι να περιορίσει την πρόσβαση σε ορισμένες λειτουργίες της εφαρμογής τον πελάτη, εξασφαλίζοντας ότι μόνο οι διαχειριστές θα έχουν την δυνατότητα να προσθέτουν, να τροποποιούν και να διαγράφουν βιβλία. Αντιθέτως, οι πελάτες θα μπορούν να εξερευνήσουν κανονικά την υπόλοιπη εφαρμογή εξαιρώντας την καταχώρηση βιβλίων. Η προσθήκη ενός εργαλείου ελέγχου αυθεντικοποίησης θα εξασφαλίσει τον έλεγχο και θα επιτρέπεται η πρόσβαση μόνο από εξουσιοδοτημένα άτομα.

Προσθήκη Καλαθιού Αγορών(Shopping Cart): Ένα σημαντικό επόμενο βήμα για τη βελτίωση της εφαρμογής είναι η ενσωμάτωση ενός καλαθιού αγορών εφόσον έχει πραγματοποιηθεί ο διαχωρισμός του διαχειριστή του βιβλιοπωλείου από τον πελάτη. Ο πελάτης θα έχει την δυνατότητα προσθήκης των βιβλίων που επιθυμεί στο καλάθι αγορών και σε τελικό στάδιο η συναλλαγή. Για αυτή τη δυνατότητα χρειάζεται να γίνει η σύνδεση με το back-end, προκειμένου να διαχειριστεί αυτές τις λειτουργίες και να είναι αρμόδιο για την έγκυρη και ομαλή ολοκλήρωση των παραγγελιών του πελάτη.

Επέκταση σε Πραγματική Βάση Δεδομένων: Με την ανάπτυξη της εφαρμογής και με των όσων προαναφέρθηκαν, αφορμή γίνεται η εισαγωγή μιας πραγματικής βάσης δεδομένων όπως είναι ένα API back-end, για την μόνιμη αποθήκευση των βιβλίων. Παρέχει ασφάλεια των δεδομένων, γιατί η αποθήκευση των δεδομένων σε έναν διακομιστή διασφαλίζει την προστασία τους από κίνδυνους, καθώς επιτρέπει την εφαρμογή ισχυρών μηχανισμών ασφάλειας, όπως είναι η κρυπτογράφηση, η ασφαλής σύνδεση μέσω Hypertext Transfer Protocol Secure(HTTPS) και η χρήση διακριτικών. Τέλος, είναι ένας τρόπος δημιουργίας αντιγράφων ασφαλείας.

Εισαγωγή Βοηθού υποστήριξης του Συστήματος: Η υποστήριξη του συστήματος για τον χρήστη αποτελεί κρίσιμο και σημαντικό στοιχείο για την εξασφάλιση μιας συνολικής θετικής εμπειρίας χρήστη και της ευχρηστίας της εφαρμογής. Είναι σημαντικό να παρέχουμε πληροφορίες, οδηγίες αλλά και βοήθεια για την σωστή πλοήγηση και χρήση της εφαρμογής. Ειδικά για χρήστες που μπορεί να έχουν ειδικές ανάγκες. Η εφαρμογή θα μπορούσε να περιλαμβάνει έναν βοηθό υποστήριξης με τις πιο συχνές ερωτήσεις και απαντήσεις. Τέλος, την επιλογή να προσαρμόζεται το χρωματικό σχήμα της εφαρμογής, ώστε να γίνονται αναγνώσιμα όλα τα στοιχεία της διεπαφής.

5.5 Συμπεράσματα

Είναι μια δυναμική και λειτουργική εφαρμογή για την καταχώρηση και αναζήτηση βιβλίων. Ωστόσο, υπάρχουν αρκετές δυνατότητες για βελτίωση και εξέλιξη. Είτε πρόκειται για την προσθήκη προηγμένων λειτουργιών όπως είναι το καλάθι αγορών ή την σύγκριση βιβλίων, είτε για την ενσωμάτωση καλύτερης διαχείρισης φόρμας καταχώρησης δεδομένων μέσω του React Hook Form ή την εισαγωγή βοηθού υποστήριξης του συστήματος. Η εφαρμογή έχει αρκετό περιθώριο για μελλοντική εξέλιξη ανάπτυξη και βελτίωση.

Η χρήση εργαλείων όπως το MSW, το Vite, το SWR, το React, και το Material UI προσφέρει μεγάλη ευελιξία και επεκτασιμότητα για την εφαρμογή, ενώ παράλληλα υπάρχουν και περιορισμοί που μπορούν να αντιμετωπιστούν με τη χρήση μιας βάσης δεδομένων ή την υιοθέτηση νέων τεχνολογιών για τη διαχείριση δεδομένων.

Επίλογος

Η ανάπτυξη της δυναμικής εφαρμογής για την διαχείριση βιβλίων, η οποία έχει ως βασικό πυρήνα την δημιουργία μιας προηγμένης front-end εφαρμογής, κρίθηκε απαραίτητη για την καλύτερη οργάνωση, αναζήτηση καταχώρηση και παρουσίαση των βιβλίων, με στόχο την αποτελεσματική ενίσχυση της εμπειρίας των χρηστών. Η χρήση σύγχρονων τεχνολογιών όπως είναι το React, το Vite σε συνεργασία με το Vitest αλλά και το Material UI συντέλεσαν στην κατασκευή και υλοποίηση της εφαρμογής. Αποτέλεσαν απαραίτητα εργαλεία για να διαμορφώσουν ένα αποδοτικό σύστημα που ανταποκρίνεται στις ανάγκες των διαχειριστών του βιβλιοπωλείου. Η δημιουργία της εφαρμογής αυτής απαιτούσε τον συνδυασμό μεταξύ των διαφόρων τεχνολογιών και εργαλείων ανάπτυξης, με κύριο γνώμονα την επίτευξη ενός εύχρηστου και ευέλικτου συστήματος, το οποίο μπορεί να προσαρμοστεί στις διάφορες ανάγκες του χρήστη αλλά και να προσφέρει γρήγορη αναζήτηση βιβλίων, καταγραφή και προβολή τους με τρόπο οπτικά ελκυστικό και ευχάριστο.

Κατά τη διάρκεια ανάπτυξης της εφαρμογής, εντοπίστηκαν αρκετές προκλήσεις που αφορούσαν την εισαγωγή και διαμόρφωση των τεχνολογιών και βιβλιοθηκών που χρησιμοποιήθηκαν. Επιπλέον, η διαχείριση των δεδομένων, η δυνατότητα που προσφέρει ώστε να μπορεί να ανταποκριθεί σε μελλοντικές τεχνολογίες αλλά και η εξυπηρέτηση της στις ανάγκες και απαιτήσεις των χρηστών σε πραγματικό χρόνο ήταν μια ιδιαίτερη πρόκληση. Επιπλέον, η βιβλιογραφική ερευνά για τα πρότυπα και τις μεθόδους ανάπτυξης front-end εφαρμογών, συντέλεσαν στην επιτυχή ολοκλήρωση του έργου. Οι πληροφορίες που αντλήθηκαν από τα άρθρα και τα έγγραφα των ίδιων των τεχνολογιών, βοήθησαν στην καλύτερη κατανόηση και καθοδήγηση στην εφαρμογή τους, καθ’ όλη την διαδικασία ανάπτυξης της εφαρμογής.

Ολοκληρωτικά, το σύστημα προσφέρει μια πλήρη λειτουργική, αξιόπιστη και αποδοτική εφαρμογή για την διαχείριση βιβλιοπωλείου, η οποία δίνει και έχει την δυνατότητα να επεκταθεί και να εξελιχθεί ώστε να ανταποκριθεί ακόμη και στις μελλοντικές ανάγκες του τομέα.

Βιβλιογραφία

[1] MDN Web Docs. HTML. Available at: https://developer.mozilla.org/en-US/docs/Glossary/HTML.

[2] MDN Web Docs, “HTML: Hypertext Markup Language”. Available at: https://developer.mozilla.org/en-US/docs/Web/HTML.

[3] Siem Peters 2019. A Brief History of JavaScript: from Netscape to Frameworks. Available at: https://blog.bitsrc.io/a-brief-history-of-javascript-from-netscape-to-frameworks-74bf4774eeef.

[4] MDN Web Docs, “JavaScript Guide”. Available at: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

[5] Statista 2024. Most used programming languages among developers worldwide. Available at: https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/.

[6] Borusiuk, Y. 2022. Top 10 Benefits of React.js for Application Development. Online. NCube. Available at: https://ncube.com/blog/top-10-benefits-of-react-js-for-application-development.

[7] React Documentation, “Introduction to React”. Available at: https://legacy.reactjs.org/docs/getting-started.html .

[8] Statista 2024. Most used web frameworks among developers worldwide. Available at: https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/.

[9] Vite 2024. Slow Server Start. Available at: https://vite.dev/guide/why#slow-updates.

[10] Vite Official Documentation, “Vite”. Available at: https://vitejs.dev/.

[11] Geeks for Geeks 2023. ReactJS Reconciliation. Available at: https://www.geeksforgeeks.org/reactjs-reconciliation/.

[12] Geeks for Geeks 2024. ReactJS Virtual DOM. Available at: https://www.geeksforgeeks.org/reactjs-virtual-dom/.

[13] Geeks for Geeks 2025. ReactJS Props. Available at: https://www.geeksforgeeks.org/reactjs-props/.

[14] Anjana Madushan 2024. Vite vs Create React App in 2024. Medium. Available at: https://blog.bitsrc.io/vite-vs-create-react-app-326e8cc2c46b.

[15] Vitest. Why Vitest. Available at: https://vitest.dev/guide/why.html

[16] MDN Web Docs. Service Worker API. Available at: https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API.

[17] SWR Official Documentation. Available at: https://swr.vercel.app/docs/getting-started.

[18] Nanotaboada/Books.json. Mock book data for fetching. Available at: A small-sized (8 items) sample collection of free Books in valid JSON (RFC 8259) format · GitHub.

[19] MUI Official Documentation. Material UI- Overview. Available at: https://mui.com/material-ui/getting-started/.

[20] React Router Documentation. Available at: https://reactrouter.com/home .

[21] NPM Official Site for packages. Available at: https://www.npmjs.com/.

Παράρτημα του Κώδικα της Εφαρμογής

Η χρήση του state αλλά και το πως επηρεάζει σε αυτό το εικονικό DOM:

```
import { useState } from "react";


export default function Library() {
 const [query, setQuery] = useState("");  // Χρησιμοποιούμε το state για την αποθήκευση της αναζήτησης


 const search = (data) => {
  return data.filter((item) => {
   const lowerCaseQuery = query.toLowerCase();
   return Object.values(item).some((value) =>
    typeof value === "string"
     ? value.toLowerCase().includes(lowerCaseQuery)
     : false
   );
  });
 }; // Ενημερώνουμε το state όταν αλλάζει το input
      <StyledInputBase
       placeholder="Search . . ."
       inputProps={{ "aria-label": "search" }}
       onChange={(e) => setQuery(e.target.value)}  // Καλούμε την συνάρτηση όταν ο χρήστης πληκτρολογεί
      />
```

Η υλοποίηση της δοκιμής unit testing:

```
import { screen, render } from "@testing-library/react";
import Footer from "../../components/Footer";
import { MemoryRouter } from "react-router-dom";
import { expect, test } from "vitest";

describe("Footer Component", () => {
 test('should display "Add your book" button', async () => {
  render(
   <MemoryRouter>
    <Footer />
   </MemoryRouter>
  );
  const addButton = screen.getByText("Add your Book");
  expect(addButton).toBeInTheDocument();
 });

 test('should not display "About" button', async () => {
  render(
   <MemoryRouter>
    <Footer />
   </MemoryRouter>
  );
  const aboutButton = screen.queryByText("About");
  expect(aboutButton).not.toBeInTheDocument();
 });
});
```

Το αρχείο με τις ρυθμίσεις του Vite:

```
/// <reference types="vitest" />
/// <reference types="vite/client" />

import { defineConfig } from "vite";
import react from "@vitejs/plugin-react";

// https://vitejs.dev/config/
export default defineConfig({
 plugins: [react()],
 test: {
  globals: true,            //Παράμετρος που ενεργοποιεί τις παγκόσμιες μεταβλητές στα tests
  environment: "jsdom",          // Ορίζουμε το περιβάλλον των δοκιμών ως jsdom (προσομοίωση browser)
  css: true,
  setupFiles: "./src/tests/setup.js",      //path για τα modules του vitest τα οποια χρησιμοποιουνται για τα tests
 },
});
```

Το JavaScript setup αρχείο υπεύθυνο για την αρχικοποίηση του περιβάλλοντος δοκιμών:

```
import { expect, afterEach } from "vitest";
import { cleanup } from "@testing-library/react";
import * as matchers from "@testing-library/jest-dom/matchers";

expect.extend(matchers);

afterEach(() => {
 cleanup();      // Καθαρισμος του DOM μετα από κάθε δοκιμη
});
```

Η υλοποίηση του χειριστή υπεύθυνος για την διαχείριση αιτημάτων, ανάκτηση και προσθήκη βιβλίων:

```
import { http, HttpResponse } from "msw";

import BooksData from "../stubs/books.json";            // Γινεται import τα mock data για βιβλια

let NewBooksData = BooksData.books;

export const bookHandler = [
 http.get("/api/books", () => {
  return HttpResponse(BooksData);               // Επιστρεφει τα δεδομενα βιλβιων για το GET αιτημα
 }),

 http.post("/api/books", async ({ request }) => {
  try {
   const newBook = await request.json();       // Προσθηκη νεου βιλβιου
   NewBooksData.push(newBook);                 // Γινεται push στο object array των mock data βιβλιων
   console.log(NewBooksData);
   return HttpResponse.json(NewBooksData);          // Επιστρεφει τη νεα λιστα βιβλιων
  } catch (error) {
   return HttpResponse.error();          // Διαχειριση σφαλματων
  }
 }),
];
```

Η Διαμόρφωση και Ενεργοποίηση του MSW και το Όρισμα του χειριστή:

```
import { setupWorker } from "msw/browser";
import { bookHandler } from "./handlers/bookHandler";          //  Ρυθμιση του MSW

export const worker = setupWorker(...bookHandler);              //  Ξεκιναει τον mock server με τους handlers
```

Η Διαχείριση Σφαλμάτων και η Ασύγχρονη Ανάκτηση Δεδομένων κατά την επικοινωνία με το API:

```
const fetcher = (url) => fetch(url).then((res) => res.json());

export default function Library() {
 const [query, setQuery] = useState("");

 // using swr to fetch my data from the json file so i can get the title and if it had inside the img
 const { data: booksData, error } = useSWR(
  "../../resources/mocks/stubs/books.json",
  fetcher
 );

 if (error) return <div>Error loading data...</div>;
 if (!booksData) return <div>Loading...</div>;

 //here is the filter according to the data that i want
 const search = (data) => {
  return data.filter((item) => {
   //convert all fields from capital to lowercase incase insensitive search
   const lowerCaseQuery = query.toLowerCase();
   return Object.values(item).some((value) =>
    typeof value === "string"
     ? value.toLowerCase().includes(lowerCaseQuery)
     : false
   );
  });
 };
```