UNIVERSITY OF THE AEGEAN
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ -
ΠΛΗΡΟΦΟΡΙΑΚΑ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

Ποια είναι τα καλύτερα πλαίσια ανάπτυξης εφαρμογών ιστού για την γλώσσα Javascript;

Η Διπλωματική εργασία κατατέθηκε στο τμήμα
Μηχανικών Πληροφοριακών & Επικοινωνιακών Συστημάτων
της Πολυτεχνικής Σχολής του Πανεπιστημίου Αιγαίου

Μαρία Χατζάκη, Βαλεντίνα Σαλακίδου

Επιτροπή

Επιβλέπων: Αναπληρωτής Καθηγητής Κυριάκος Κρητικός
Καθηγητής Σπυρίδων Κοκολάκης
Αναπληρωτής Καθηγητής Γκουμόπουλος Χρήστος

Σάμος, Φεβρουάριος 2024

UNIVERSITY OF THE AEGEAN
DEPARTMENT OF INFORMATION & COMMUNICATION SYSTEMS ENGINEERING
MSc PROGRAMME -
INFORMATION & COMMUNICATION SYSTEMS

Which are the best web development frameworks for the Javascript language?

A thesis statement
submitted to the Department of Information & Communication Systems Engineering
School of Engineering of the University Of The Aegean

Maria Chatzaki, Valentina Salakidou

Committee

Supervisor: Associate Professor Kyriakos Kritikos
Professor Kokolakis Spyridon
Associate Professor Gkoumopoulos Christos

Samos, February 2024

Δήλωση Αυθεντικότητας

Βεβαιώνω ότι είμαι συγγραφέας αυτής της διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης έχω αναφέρει τις πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Τέλος, βεβαιώνω ότι αυτή η διπλωματική εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του μεταπτυχιακού προγράμματος σπουδών του Τμήματος Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων του Πανεπιστημίου Αιγαίου στη Σάμο.

Καρλόβασι, 19 Φεβρουαρίου 2024

Μαρία Χατζάκη

(Υπογραφή)

Βαλεντίνα Σαλακίδου

(Υπογραφή)

Statement of Authenticity

I declare that this thesis is my own work and was written without literature other than the sources indicated in the bibliography. Information used from the published or unpublished work of others has been acknowledged in the text and has been explicitly referred to in the given list of references. This thesis has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education.

Karlovasi, 19th February 2024

Maria Chatzaki

(Signature)

Valentina Salakidou

(Signature)

Περίληψη

Αυτή η διατριβή διερευνά και συγκρίνει διάφορα πλαίσια ανάπτυξης ιστού για τη γλώσσα προγραμματισμού JavaScript, που περιλαμβάνει λύσεις front-end, back-end και full-stack. Η συγκριτική ανάλυση βασίζεται σε διάφορες πηγές, όπως βιβλιογραφία, τεκμηρίωση, ιστότοπους πλαισίων και πλατφόρμες σύγκρισης. Η σύγκριση πραγματοποιείται τόσο μέσω θεωρητικού όσο και πρακτικού φακού, με ιδιαίτερη έμφαση στην ποιότητα του λογισμικού και τις πτυχές ασφάλειας. Μια πρωταρχική πτυχή της πρακτικής αξιολόγησης περιλαμβάνει τον εντοπισμό και την ανάλυση τρωτών σημείων και ζητημάτων ασφάλειας στον πηγαίο κώδικα των πλαισίων που χρησιμοποιούν εργαλεία ανάλυσης στατικού κώδικα. Η αξιολόγηση της κρισιμότητας αυτών των τρωτών σημείων ενημερώνει τη διαδικασία συγκριτικής επιλογής πλαισίου.

Επιπλέον, η εφαρμογή εργαλείων στατικής ανάλυσης διευκολύνει την εξαγωγή μετρήσεων ποιότητας όπως η διπλή και η κυκλωμική πολυπλοκότητα, παρέχοντας πληροφορίες για τη συνολική ποιότητα λογισμικού των υπό εξέταση πλαισίων. Τα συγκριτικά αποτελέσματα αναλύονται αυστηρά για να διακρίνονται αξιοσημείωτα μοτίβα και να εξαχθούν σημαντικά συμπεράσματα σχετικά με τα ανώτερα πλαίσια, τις δυνάμεις τους σε διαφορετικά κριτήρια, καθώς και τομείς για πιθανή βελτίωση. Τα ευρήματα αυτής της έρευνας συμβάλλουν στον εντοπισμό κατευθύνσεων για μελλοντική έρευνα με στόχο τη βελτίωση των πλαισίων ανάπτυξης διαδικτυακών εφαρμογών.

Μαρία Χατζάκη, Βαλεντίνα Σαλακίδου
Τμήμα Μηχανικών Πληροφοριακών & Επικοινωνιακών Συστημάτων
Πανεπιστήμιο Αιγαίου

Abstract

This thesis investigates and compares various web development frameworks for the JavaScript programming language, encompassing both front-end, back-end, and full-stack solutions. The comparative analysis draws upon diverse sources including literature, documentation, framework websites, and comparison platforms. The comparison is conducted through both theoretical and practical lenses, with a specific focus on software quality and security aspects. A paramount aspect of the practical evaluation involves the identification and analysis of vulnerabilities and security issues within the source code of the frameworks utilizing static code analysis tools. Criticality assessment of these vulnerabilities informs the comparative framework selection process.

Moreover, the application of static analysis tools facilitates the derivation of quality metrics such as duplication and cyclomatic complexity, providing insights into the overall software quality of the frameworks under examination. The comparative results are rigorously analyzed to discern notable patterns and draw meaningful conclusions regarding the superior frameworks, their strengths across distinct criteria, as well as areas for potential improvement. The findings of this research contribute to the identification of directions for future research aimed at enhancing web application development frameworks.

Maria Chatzaki, Valentina Salakidou
Department of Information and Communication Systems Engineering
University of the Aegean

Περιεχόμενα

Λίστα Σχημάτων

Λίστα Ακρωνυμίων

HTML . . . . . Hypertext Markup Language

CSS . . . . . Cascading Style Sheets

HTTP . . . . . Hypertext Transfer Protocol

GDPR . . . . . General Data Protection Regulation

PHP . . . . . PHP: Hypertext Preprocessor

WWW . . . . . World Wide Web

W3C . . . . . World Wide Web Consortium

DOM . . . . . Document Object Model

API . . . . . Application Programming Interface

Κεφάλαιο 1

Εισαγωγή

1.1 Εισαγωγικά Στοιχεία

Η διπλωματική εργασία εστιάζει σε έναν δυναμικό και σημαντικό χώρο στον τομέα των τεχνολογιών έρευνας και πληροφορικής. Ο χώρος αυτός χαρακτηρίζεται από τη συνεχή εξέλιξη της τεχνολογίας και τον ραγδαίο ρυθμό ανάπτυξης του Διαδικτύου. Η JavaScript, ως γλώσσα προγραμματισμού, κατέχει κεντρική θέση στη δημιουργία διαδικτυακών εφαρμογών, ενώ τα πλαίσια ανάπτυξης συμβάλλουν στην ευκολότερη και αποτελεσματικότερη διαδικασία ανάπτυξης.

Στον εν λόγω χώρο, εμφανίζονται χαρακτηριστικά που αφορούν τόσο τον προγραμματιστή όσο και τον τελικό χρήστη. Συνεπώς, η ανάγκη για γρήγορη ανάπτυξη, ευελιξία, καλή διαχείριση του κώδικα, και ασφάλεια αποτελούν βασικά χαρακτηριστικά που ορίζουν τις απαιτήσεις του πεδίου. Παράλληλα, η ανάδυση νέων προκλήσεων, όπως η ασφάλεια των εφαρμογών και η βελτιστοποίηση της απόδοσης, αντιπροσωπεύει προκλήσεις που οι ερευνητές και οι προγραμματιστές πρέπει να αντιμετωπίσουν. Στη συνέχεια της διπλωματικής εργασίας, θα εξεταστούν πιο λεπτομερώς αυτά τα χαρακτηριστικά και οι προκλήσεις, προσφέροντας λύσεις και προτάσεις για τη βελτίωση της ανάπτυξης εφαρμογών ιστού σε περιβάλλον JavaScript.

1.2 Αντικείμενο διπλωματικής

Η παρούσα διπλωματική εργασία επικεντρώνεται στη συστηματική ανάλυση και σύγκριση πλαισίων ανάπτυξης εφαρμογών ιστού για τη γλώσσα προγραμματισμού Javascript. Ο στόχος είναι να αναδείξει τις δυνατότητες και τους περιορισμούς των πλαισίων αυτών, εστιάζοντας τόσο στη front-end όσο και στη back-end πλευρά ανάπτυξης. Με έμφαση στην πρακτική πλευρά, η εργασία διερευνά την ασφάλεια του πηγαίου κώδικα μέσω εργαλείων στατικής ανάλυσης, αναζητώντας τρωτότητες και ζητήματα ασφάλειας. Επιπλέον, μέσω μετρικών ποιότητας, όπως η διπλό-εγγραφή και η κυκλωματική πολυπλοκότητα,

αξιολογείται η συνολική ποιότητα του λογισμικού των πλαισίων.

Τα αποτελέσματα της σύγκρισης αναλύονται λεπτομερώς, προσφέροντας σημαντικά συμπεράσματα για την καλύτερη κατανόηση των πλεονεκτημάτων και των αδυναμιών κάθε πλαισίου. Με βάση αυτήν την ανάλυση, η διπλωματική εργασία διαμορφώνει προτάσεις για μελλοντικές κατευθύνσεις έρευνας που θα ενισχύσουν την ποιότητα και την ασφάλεια των πλαισίων ανάπτυξης εφαρμογών ιστού.

1.3 Δομή Διπλωματικής

Η διπλωματική εργασία διακρίνεται στις επόμενες εννιά θεματικές ενότητες. Αρχικά, για την καλύτερη κατανόηση της παρούσας εργασίας κρίνεται αναγκαίο στο 2ο κεφάλαιο, και πρωταρχικό του κυρίου μέρους, να χτιστεί το απαραίτητο υπόβαθρο. Στην συνέχεια, τα πλαίσια ανάπτυξης των εφαρμογών ιστού και τα βασικά χαρακτηριστικά τους αναλύονται στο 3ο Κεφάλαιο. Η γλώσσα JavaScript, τα χαρακτηριστικά και τα πλεονεκτήματα της περιγράφονται στο 4ο Κεφάλαιο.

Στο Κεφάλαιο 5, παρατίθεται η μεθοδολογία που ακολουθήθηκε για την επιλογή των πλαισίων ανάπτυξης εφαρμογών ιστού στην JavaScript, καθώς και οι τρόποι αναζήτησης και φιλτραρίσματος αυτών των πλαισίων. Το 6ο Κεφάλαιο περιλαμβάνει την σύντομη περιγραφή των πλαισίων που επιλέχθηκαν, ενώ στο 7ο Κεφάλαιο κατηγοριοποιούνται τα επιλεγμένα πλαίσια βάσει διαστάσεων. Η παρουσίαση και η επεξήγηση των κριτηρίων σύγκρισης των πλαισίων συμπεριλαμβάνονται στο Κεφάλαιο 8 και στο Κεφάλαιο 9 λαμβάνει χώρα η ίδια η σύγκριση των πλαισίων ανάπτυξης εφαρμογών ιστού στην JavaScript. Στο 10ο και τελευταίο Κεφάλαιο παρουσιάζονται τα συμπεράσματα της εργασίας, καθώς και προτάσεις για μελλοντικές ενέργειες.

Κεφάλαιο 2

Υπόβαθρο

2.1 Πλαίσια Ανάπτυξης Εφαρμογών Ιστού

Τα πλαίσια ανάπτυξης εφαρμογών ιστού αποτελούν θεμέλιο στον κόσμο της προγραμματιστικής κοινότητας, παρέχοντας ένα εξελιγμένο σύνολο εργαλείων και λειτουργιών που διευκολύνουν την ανάπτυξη εφαρμογών. Αυτά καλύπτουν διάφορες πτυχές του κύκλου ζωής των εφαρμογών, ξεκινώντας από την αρχική σχεδίαση και φτάνοντας μέχρι τη συντήρηση και την επέκταση του λογισμικού, ενώ το βασικό χαρακτηριστικό που καθιστά τα πλαίσια τόσο χρήσιμα είναι η δομημένη αρχιτεκτονική που παρέχουν, επιτρέποντας την ανάπτυξη εφαρμογών σε μεγάλη κλίμακα με συνοχή και αποτελεσματικότητα.

2.1.1 Εισαγωγή στα Πλαίσια Ανάπτυξης Εφαρμογών Ιστού

Κατά τη διάρκεια του κύκλου ζωής των εφαρμογών, τα πλαίσια καλύπτουν κρίσιμες πτυχές. Παρέχουν μηχανισμούς για τη διαχείριση του κώδικα, επιτρέποντας την ευκολότερη συντήρηση και τη διαχείριση αυξανόμενης πολυπλοκότητας. Επίσης, παρέχουν λύσεις για τη διαχείριση της εφαρμογικής λογικής, την αλληλεπίδραση με βάσεις δεδομένων και τη διαχείριση των αιτημάτων των χρηστών.

Τα πλαίσια ανάπτυξης εφαρμογών ιστού επίσης διακρίνονται για την επαναχρησιμοποίηση του κώδικα, προωθώντας τη σταθερότητα και την αξιοπιστία στις εφαρμογές. Επιπλέον, τίθενται ως πηγή καλών πρακτικών, παρέχοντας έτοιμες λύσεις για κοινά προβλήματα, όπως η ασφάλεια, η διαχείριση χρηστών και η επεκτασιμότητα. Συνολικά, η ευρεία διάδοση των πλαισίων ανάπτυξης συμβάλλει στην επιτυχία και τη σταθερότητα των σύγχρονων εφαρμογών, παρέχοντας εξελιγμένες και συνεκτικές λύσεις.

Ένα ακόμη βασικό χαρακτηριστικό των πλαισίων ανάπτυξης είναι η δυνατότητα να παρέχουν λειτουργικότητα χωρίς την ανάγκη να ξεκινήσουν τα πάντα από την αρχή. Με την ύπαρξη προεγκατεστημένων λύσεων και προκαθορισμένων παραμέτρων, οι προγραμματιστές εξοικονομούν χρόνο και προσπάθεια, επικεντρώνοντας την προσοχή τους στα ειδικά χαρακτηριστικά και τις ανάγκες του συγκεκριμένου έργου.

Σε πολλές περιπτώσεις, τα πλαίσια ανάπτυξης στοχεύουν στη δημιουργία ενός ολοκληρωμένου οικοσυστήματος, προσφέροντας όχι μόνο τη βασική υποδομή για την ανάπτυξη, αλλά και εργαλεία για τη διαχείριση, τον έλεγχο και την ανάλυση της εφαρμογής. Αυτό επιτρέπει στις ομάδες ανάπτυξης να επικεντρώνονται σε εξειδικευμένες λειτουργίες, ενώ ταυτόχρονα επωφελούνται από τη συνοχή και την ασφάλεια που παρέχει το πλαίσιο.

Συνεπώς, τα πλαίσια ανάπτυξης εφαρμογών ιστού εξελίσσονται διαρκώς για να αντιμετωπίσουν τις εξελισσόμενες απαιτήσεις του κόσμου της τεχνολογίας. Με τις αναβαθμίσεις και τις νέες εκδόσεις, προσφέρουν βελτιωμένες επιδόσεις, νέες λειτουργίες και προηγμένες δυνατότητες. Αυτό εξασφαλίζει ότι οι προγραμματιστές έχουν πρόσβαση σε τεχνολογικά προηγμένες λύσεις και παραμένουν ενημερωμένοι σε έναν ραγδαία εξελισσόμενο τομέα.

2.1.2 Οι Κατηγορίες των Πλαισίων Ανάπτυξης Εφαρμογών Ιστού

Τα πλαίσια ανάπτυξης εφαρμογών ιστού διακρίνονται σε δύο βασικές κατηγορίες με βάση τον ρόλο τους στη διαδικασία ανάπτυξης και λειτουργίας της εφαρμογής. Οι δύο αυτές βασικές κατηγορίες είναι τα Front-end Frameworks και τα Back-end Frameworks.

2.1.2.1 Front-end Πλαίσια Ανάπτυξης

Τα Front-end πλαίσια ανάπτυξης αποτελούν απαραίτητα εργαλεία στην ανάπτυξη ιστού παρέχοντας δομημένη βάση για τη δημιουργία διεπαφών χρήστη (User Interface - UI). Αυτά τα πλαίσια αποτελούνται από μια συλλογή προγραμμένου κώδικα, βιβλιοθηκών και συμβάσεων, που εξορθολογίζουν τη διαδικασία ανάπτυξης και επιτρέπουν στους προγραμματιστές να δημιουργούν συνεπείς, ανταποκρινόμενους και οπτικά ελκυστικούς ιστότοπους και εφαρμογές Ιστού. Ο πρωταρχικός σκοπός των πλαισίων διεπαφής είναι η αντιμετώπιση κοινών προκλήσεων στην ανάπτυξη διεπαφής χρήστη, όπως η συμβατότητα μεταξύ προγραμμάτων περιήγησης, ο αποκριτικός σχεδιασμός και η οργάνωση κώδικα.

Αυτά τα πλαίσια περιλαμβάνουν συνήθως ένα σύνολο τυποποιημένων στοιχείων, όπως πλέγματα, γραμμές πλοήγησης, κουμπιά και στοιχεία φόρμας, τα οποία οι προγραμματιστές μπορούν να αξιοποιήσουν για να επιταχύνουν τη διαδικασία ανάπτυξης. Με την υιοθέτηση ενός πλαισίου διεπαφής, οι προγραμματιστές μπορούν να επικεντρωθούν περισσότερο σε λειτουργίες και λειτουργικότητα που αφορούν συγκεκριμένες εφαρμογές αντί να ξοδεύουν πολύ χρόνο σε επαναλαμβανόμενες εργασίες. Η χρήση ενός πλαισίου συμβάλλει επίσης στη συντηρησιμότητα και την επεκτασιμότητα, καθώς οι ενημερώσεις και οι βελτιώσεις στο πλαίσιο μπορούν να ενσωματωθούν απρόσκοπτα σε υπάρχοντα έργα.

Τα πλαίσια Front-end συχνά ακολουθούν μια αρθρωτή αρχιτεκτονική που βασίζεται σε στοιχεία, προάγοντας την επαναχρησιμοποίηση και τη συντηρησιμότητα κώδικα. Αυτή η προσέγγιση ενισχύει τη συνεργασία μεταξύ των ομάδων ανάπτυξης και επιτρέπει την

αποτελεσματική δημιουργία πολύπλοκων διεπαφών χρήστη. Τα δημοφιλή πλαίσια Front-end περιλαμβάνουν το Bootstrap, το React, το Angular και το Vue.js, το καθένα με τα μοναδικά χαρακτηριστικά και τα δυνατά του σημεία που καλύπτουν διαφορετικές προτιμήσεις ανάπτυξης και απαιτήσεις έργου.

Τα πλαίσια ανάπτυξης Front-end διαδραματίζουν κρίσιμο ρόλο στη βελτίωση της συνολικής εμπειρίας χρήστη (UX) ιστότοπων και ιστοεφαρμογών. Διαθέτουν εγγενή στοιχεία και μοτίβα σχεδίασης βελτιστοποιημένα για χρηστικότητα και προσβασιμότητα, εξασφαλίζοντας μια συνεπή και απολαυστική εμπειρία σε διαφορετικές συσκευές και αναλύσεις οθόνης.

Επιπλέον, συμβάλλουν στη συνοχή του κώδικα και τις βέλτιστες πρακτικές, επιβάλλοντας συμβάσεις κωδικοποίησης και αρχές σχεδιασμού που διευκολύνουν τους προγραμματιστές να ακολουθούν τα πρότυπα του κλάδου. Αυτή η συνέπεια είναι πολύτιμη όταν πολλοί προγραμματιστές συνεργάζονται σε ένα έργο ή κατά τη μετάβαση έργων μεταξύ μελών της ομάδας.

Ένα σημαντικό πλεονέκτημα των πλαισίων Front-end είναι η δυνατότητα που παρέχουν για ανταπόκριση (responsive design). Με την ποικιλία συσκευών και αναλύσεων οθόνης, η δημιουργία ιστοσελίδων που προσαρμόζονται απροβλημάτιστα είναι πολύπλοκη. Τα πλαίσια Front-end διαθέτουν έτοιμες λειτουργίες ανταπόκρισης, επιτρέποντας στους προγραμματιστές να δημιουργούν ιστότοπους με άριστη εμφάνιση και λειτουργικότητα σε υπολογιστές, τάμπλετ και κινητές συσκευές.

Όσον αφορά την υποστήριξη και την τεκμηρίωση, τα δημοφιλή πλαίσια Front-end επωφελούνται από μεγάλη βάση χρηστών, εκτεταμένη τεκμηρίωση και ενεργές διαδικτυακές κοινότητες. Έτσι, οι προγραμματιστές μπορούν εύκολα να βρουν λύσεις σε κοινά ζητήματα, να μοιραστούν γνώσεις και να αξιοποιήσουν πλούσιους πόρους.

Κάθε πλαίσιο έχει τα δυνατά του σημεία και περιπτώσεις χρήσης, με την επιλογή να εξαρτάται από τις απαιτήσεις του έργου, την τεχνογνωσία της ομάδας και την επιθυμητή εμπειρία χρήστη. Παράγοντες όπως η ευκολία εκμάθησης, η απόδοση, η επεκτασιμότητα και η ενοποίηση με άλλα εργαλεία πρέπει να ληφθούν υπόψη.

Συμπερασματικά, τα πλαίσια Front-end συμβάλλουν σημαντικά στην αποτελεσματικότητα, συνέπεια και ποιότητα των έργων ανάπτυξης ιστού. Είτε για γρήγορη πρωτοτυποποίηση, δημιουργία διαδραστικών διεπαφών, ισχυρές εφαρμογές μιας σελίδας ή απλότητα και ευελιξία, αυτά τα πλαίσια παρέχουν στους προγραμματιστές τη δυνατότητα να δημιουργούν σύγχρονες, πλούσιες και οπτικά ελκυστικές ιστοεφαρμογές.

2.1.2.2 Back-end πλαίσια ανάπτυξης

Τα πλαίσια Back-end αποτελούν θεμέλια στη σύγχρονη ανάπτυξη ιστού, προσφέροντας στους προγραμματιστές μια δομημένη προσέγγιση για την κατασκευή της λογικής και της υποδομής από την πλευρά του διακομιστή των εφαρμογών ιστού. Σε αντίθεση με τα πλαίσια Front-end, τα οποία επικεντρώνονται στη σχεδίαση διεπαφής χρήστη, τα

πλαίσια Back-end αντιμετωπίζουν την πολυπλοκότητα της διαχείρισης δεδομένων, την εκτέλεση επιχειρηματικής λογικής και την απρόσκοπτη αλληλεπίδραση με βάσεις δεδομένων.

Ένας σημαντικός πυλώνας των πλαισίων υποστήριξης είναι η παροχή μηχανισμών δρομολόγησης, οι οποίοι επιτρέπουν στους προγραμματιστές να ορίζουν και να διαχειρίζονται αποτελεσματικά διάφορα τελικά σημεία αιτημάτων HTTP. Αυτή η δυνατότητα απλοποιεί την οργάνωση του κώδικα από την πλευρά του διακομιστή και διευκολύνει τη δημιουργία RESTful API, επιτρέποντας την επικοινωνία μεταξύ του front-end και του back-end. Επιπλέον, η υποστήριξη του Middleware είναι ένα άλλο κρίσιμο χαρακτηριστικό που παρέχει μία ευέλικτη εργαλειοθήκη για την εκτέλεση λειτουργιών όπως ο έλεγχος ταυτότητας, η καταγραφή και ο χειρισμός σφαλμάτων σε διαφορετικά στάδια του κύκλου αιτήματος-απόκρισης. Αυτή η αρθρωτή προσέγγιση ενισχύει την οργάνωση του κώδικα και προωθεί την υλοποίηση βασικών λειτουργιών με δομημένο τρόπο.

Επιπλέον, η εστίαση στην ασφάλεια ξεχωρίζει τα πλαίσια Back-end, με ενσωματωμένες δυνατότητες και συμμόρφωση με τις βέλτιστες πρακτικές που προστατεύουν τις εφαρμογές από κοινά τρωτά σημεία. Από την άμυνα έναντι της δέσμης ενεργειών μεταξύ τοποθεσιών (XSS) έως την αποτροπή της πλαστογράφησης αιτημάτων μεταξύ τοποθεσιών (CSRF) και της έγχυσης SQL, αυτά τα πλαίσια συμβάλλουν στην ευρωστία των εφαρμογών ιστού. Κατά αυτόν τον τρόπο, η εν λόγω δέσμευση για την ασφάλεια όχι μόνο προστατεύει ευαίσθητα δεδομένα, αλλά επίσης ενισχύει τη συνολική ανθεκτικότητα της εφαρμογής έναντι των εξελισσόμενων απειλών στον κυβερνοχώρο.

Προχωρώντας, η επεκτασιμότητα και η βελτιστοποίηση απόδοσης αποτελούν πρωταρχικά ζητήματα στη σύγχρονη ανάπτυξη ιστού, δεδομένης της αυξανόμενης ζήτησης για αποκριτικές και αποτελεσματικές εφαρμογές. Τα πλαίσια Back-end αντιμετωπίζουν αυτές τις προκλήσεις παρέχοντας εργαλεία και μοτίβα για αποτελεσματική διαχείριση περιόδων σύνδεσης, ταυτόχρονο χειρισμό αιτημάτων χρήστη και ασύγχρονες λειτουργίες. Αυτό διασφαλίζει ότι οι εφαρμογές ιστού διατηρούν μια αποκριτική και αξιόπιστη εμπειρία χρήστη ακόμη και υπό συνθήκες υψηλής επισκεψιμότητας, συμβάλλοντας στην επεκτασιμότητα και τη συνολική τους απόδοση.

Γίνεται φανερό πόσο κρίσιμο ρόλο διαδραματίζουν τα πλαίσια Back-end στη συνολική διαδικασία ανάπτυξης ιστού. Από τη διαχείριση δεδομένων και την υλοποίηση επιχειρηματικής λογικής μέχρι τη διασφάλιση της ασφάλειας και την επιδίωξη της επεκτασιμότητας, αυτά τα πλαίσια αποτελούν το θεμέλιο των διακομιστών που υποστηρίζουν τις εφαρμογές ιστού. Με ιδιαίτερη προσοχή στην αποδοτικότητα, την ασφάλεια και την ευελιξία, τα πλαίσια Back-end ενδυναμώνουν τους προγραμματιστές να δημιουργούν εφαρμογές που όχι μόνο ανταποκρίνονται στις σύγχρονες απαιτήσεις, αλλά επίσης ξεπερνούν τις προσδοκίες των χρηστών.

2.2 Η γλώσσα JavaScript

Η JavaScript είναι μια γλώσσα υψηλού επιπέδου, ερμηνευμένη και δυναμική, που σημαίνει ότι όχι μόνο δεν χρειάζεται να μεταγλωττιστεί πριν από την εκτέλεση αλλά και ότι μπορεί να αλλάξει τη συμπεριφορά της κατά το χρόνο εκτέλεσης. Υποστηρίζει πολλαπλά παραδείγματα προγραμματισμού, όπως επιτακτικά, λειτουργικά και αντικειμενοστραφή. Έχει επίσης μερικά μοναδικά χαρακτηριστικά, όπως πρωτότυπη κληρονομικότητα και λειτουργίες πρώτης κατηγορίας. Επιπλέον, η JavaScript μπορεί να χειριστεί το μοντέλο αντικειμένου εγγράφου (DOM), το οποίο είναι μια αναπαράσταση της δομής και του περιεχομένου της ιστοσελίδας, και το Μοντέλο αντικειμένου προγράμματος περιήγησης(BOM), το οποίο είναι μια συλλογή αντικειμένων που παρέχουν πρόσβαση στη λειτουργικότητα του προγράμματος περιήγησης.

2.2.1 Η ιστορία της JavaScript

Η ιστορία της JavaScript[1] ξεκινά από τη συνεργασία της Netscape[2] και της Sun Microsystems [3]. Πριν από την εισαγωγή της, οι περιηγητές ιστού ήταν βασικά εργαλεία για την προβολή υπερκειμενικών εγγράφων. Η πρόθεση πίσω από τη δημιουργία της JavaScript ήταν να βελτιώσει τις ιστοσελίδες και να εισαγάγει διαδραστικότητα. Η πρώτη έκδοση, JavaScript 1.0, κυκλοφόρησε το 1995 με το Netscape Navigator 2.

Εκείνη την εποχή, το Netscape Navigator κυριαρχούσε στην αγορά περιηγητών. Η Microsoft, προσπαθώντας να καλύψει το χάσμα με τον Internet Explorer, ακολούθησε εισάγοντας τη VBScript και μια δική της έκδοση JavaScript με την ονομασία JScript, στον Internet Explorer 3.

Ως απάντηση, η Netscape, η Sun Microsystems και η Ευρωπαϊκή Ένωση Κατασκευαστών Υπολογιστών (ECMA) συνεργάστηκαν για την τυποποίηση της γλώσσας, δημιουργώντας το ECMAScript, έναν εναλλακτικό όρο για την ίδια γλώσσα. Έτσι, παρά το γεγονός ότι δεν έχει ευρεία αποδοχή, είναι πιο ακριβές να αναφέρεται η JavaScript ως ECMAScript.

Είναι σημαντικό να τονιστεί ότι η JavaScript δεν σχετίζεται με τη γλώσσα προγραμματισμού Java, που αναπτύχθηκε από τη Sun Microsystems. Αρχικά ονομαζόταν LiveScript, αλλά μετονομάστηκε σε "JavaScript" για να δημιουργήσει μια σύνδεση με την Java, οδηγώντας σε σύγχυση. Αυτή η σύγχυση επιδεινώθηκε περαιτέρω από τους περιηγητές ιστού που υποστήριζαν μια μορφή Java στην πλευρά του πελάτη. Ωστόσο, ενώ η Java είναι ευέλικτη και θεωρητικά μπορεί να αναπτυχθεί σε οποιοδήποτε περιβάλλον, η JavaScript σχεδιάστηκε ειδικά για περιβάλλοντα περιηγητών ιστού.

Η JavaScript λειτουργεί ως γλώσσα σεναρίων, καθοδηγώντας τον περιηγητή ιστού για τις ενέργειες που πρέπει να εκτελέσει. Ο περιηγητής ερμηνεύει το σενάριο και εκτελεί τις απαραίτητες εργασίες. Αυτή η λειτουργία οδηγεί συχνά σε δυσμενείς συγκρίσεις με μεταγλωττισμένες γλώσσες όπως η Java και η C++. Ωστόσο, η απλότητα της JavaScript,

με χαμηλό εμπόδιο εισόδου, απευθύνθηκε σε μη προγραμματιστές που μπορούσαν εύκολα να ενσωματώσουν σενάρια στις ιστοσελίδες τους.

Επιπλέον, η JavaScript δίνει στους προγραμματιστές τη δυνατότητα να χειρίζονται διάφορες πτυχές του περιηγητή ιστού, όπως η τροποποίηση των ιδιοτήτων του παραθύρου του περιηγητή, γνωστή ως Μοντέλο Αντικειμένου Περιηγητή (BOM).

2.2.2 Τα κύρια χαρακτηριστικά & πλεονεκτήματα της JavaScript

Η JavaScript αποτελεί μια από τις πιο ευέλικτες και υψηλού επιπέδου γλώσσες προγραμματισμού. Η εν λόγω γλώσσα θεωρείται ως μία από τις κυριότερες γλώσσες ανάπτυξης εφαρμογών Ιστού και αυτό γιατί διαθέτει κάποια βασικά χαρακτηριστικά και πλεονεκτήματα τα οποία έχουν συμβάλλει σε αυτό. Πιο συγκεκριμένα αυτά αναλύονται παρακάτω [4]:

1. Συμβατότητα μεταξύ πλατφορμών: Η JavaScript εκτελείται από την πλευρά του πελάτη, συνήθως σε προγράμματα περιήγησης ιστού, καθιστώντας τη γλώσσα μεταξύ πλατφορμών. Έτσι, επιτρέπει στους προγραμματιστές να δημιουργούν διαδραστικές και δυναμικές διεπαφές χρήστη που λειτουργούν απρόσκοπτα σε διάφορες συσκευές και λειτουργικά συστήματα.

2. Υψηλή δυναμική: Ως γλώσσα δυναμικής πληκτρολόγησης μπορεί να χαρακτηριστεί η JavaScript, αφού επιτρέπει στις μεταβλητές να προσαρμόζονται σε διαφορετικούς τύπους δεδομένων κατά τη διάρκεια του χρόνου εκτέλεσης. Αυτή η ευελιξία απλοποιεί την κωδικοποίηση και επιτρέπει τη δημιουργία δυναμικών εφαρμογών με απόκριση.

3. Προγραμματισμός βάσει συμβάντων: HJavaScript βασίζεται σε συμβάντα. Αυτό πρακτικά σημαίνει ότι ανταποκρίνεται σε ενέργειες χρήστη ή συμβάντα συστήματος αντί να εκτελεί κώδικα σε γραμμική ακολουθία. Το χαρακτηριστικό αυτό, είναι ζωτικής σημασίας για τη δημιουργία δια-δραστικών εμπειριών χρηστών, καθώς οι προγραμματιστές μπορούν να ορίσουν ενέργειες που ενεργοποιούνται από συμβάντα όπως κλικ, αλλαγές εισόδου ή ακόμη και φόρτωση σελίδων.

4. Χειρισμός DOM: Η JavaScript είναι η κύρια γλώσσα για τον χειρισμό του Μοντέλου Αντικειμένου Εγγράφου (Document Object Model- DOM), την αναπαράσταση της δομής μιας ιστοσελίδας. Αυτή η δυνατότητα επιτρέπει στους προγραμματιστές να ενημερώνουν και να τροποποιούν δυναμικά το περιεχόμενο, τη δομή και το στυλ μιας ιστοσελίδας ως απόκριση στις αλληλεπιδράσεις των χρηστών.

5. Βιβλιοθήκες και πλαίσια: Ένα από τα κυριότερα πλεονεκτήματα της JavaScript είναι ότι διαθέτει ένα τεράστιο οικοσύστημα βιβλιοθηκών και πλαισίων, όπως jQuery, React, Angular και Vue.js. Αυτά τα εργαλεία απλοποιούν την ανάπτυξη παρέχοντας

προκατασκευασμένα στοιχεία, βοηθητικά προγράμματα και αφαιρέσεις που επιταχύνουν κοινές εργασίες και ενισχύουν την οργάνωση του κώδικα.

6. Διαλειτουργικότητα: Ένα σημαντικό πλεονέκτημα της JavaScript είναι πως μπορεί εύκολα να ενσωματωθεί με άλλες γλώσσες και τεχνολογίες. Κατά αυτόν τον τρόπο, μπορεί να επικοινωνεί με διακομιστές υποστήριξης μέσω API, επιτρέποντας την απρόσκοπτη ανταλλαγή δεδομένων μεταξύ του πελάτη και του διακομιστή.

7. Ασφάλεια: Τέλος, οι σύγχρονες υλοποιήσεις JavaScript περιλαμβάνουν χαρακτηριστικά ασφαλείας και τα προγράμματα περιήγησης επιβάλλουν πολιτικές ίδιας προέλευσης για την πρόληψη κακόβουλων δραστηριοτήτων. Επιπλέον, η χρήση πλαισίων και βιβλιοθηκών συχνά ενθαρρύνει τις βέλτιστες πρακτικές, μειώνοντας τον κίνδυνο κοινών τρωτών σημείων ασφαλείας.

Συνεπώς, γίνεται κατανοητό ότι η προσαρμοστικότητα και η ευρεία χρήση της JavaScript την καθιστούν μια θεμελιώδη γλώσσα για την ανάπτυξη ιστού, διαδραματίζοντας κρίσιμο ρόλο στη δημιουργία δια-δραστικών και πλούσιων σε δυνατότητες εφαρμογών ιστού.

2.2.3 Οι λόγοι που οδηγούν στην υιοθέτηση της JavaScript για την ανάπτυξη εφαρμογών ιστού

Έχοντας αναλύσει τα βασικότερα χαρακτηριστικά και πλεονεκτήματα της γλώσσας προγραμματισμού JavaScript νωρίτερα, κατανοεί κανείς πως μέσα από αυτά τα γνωρίσματα δημιουργούνται οι λόγοι για έναν προγραμματιστή ή μία εταιρεία να επιλέξει την JavaScript ως γλώσσα προγραμματισμού για την ανάπτυξη εφαρμογών ιστού. Παρακάτω αναλύονται οι λόγοι [5] που οδηγούν στην επιλογή της στην σημερινή εποχή.

Στη σύγχρονη ανάπτυξη ιστού, η JavaScript παραμένει η βάση για τη δημιουργία ισχυρών και διαδραστικών ιστοεφαρμογών, χάρη σε αρκετούς παράγοντες που συμβάλλουν στη συνεχή υιοθέτησή της.

Ένας βασικός λόγος είναι η ευρεία υποστήριξή της από τους σύγχρονους περιηγητές ιστού. Ως γλώσσα προγραμματισμού πλευράς πελάτη, η JavaScript εκτελείται απευθείας στους περιηγητές των χρηστών, διασφαλίζοντας τη συμβατότητα σε διάφορες πλατφόρμες και συσκευές. Αυτή η διαθεσιμότητα επιτρέπει στους προγραμματιστές να δημιουργούν εφαρμογές που προσεγγίζουν ένα ευρύ κοινό χωρίς ανησυχίες συμβατότητας μεταξύ περιηγητών.

Επιπλέον, η άνοδος των Εφαρμογών Ενιαίας Σελίδας (SPA) τονίζει τη σημασία της JavaScript. Πλαίσια όπως το React, Angular και Vue.js διευκολύνουν την ανάπτυξη SPA, όπου μια μεμονωμένη σελίδα HTML ενημερώνεται δυναμικά κατά την αλληλεπίδραση των χρηστών με την εφαρμογή. Αυτή η προσέγγιση οδηγεί σε ταχύτερους χρόνους φόρτωσης, ομαλές μεταβάσεις και ανταποκρινόμενη εμπειρία χρήστη. Η ικανότητα της

JavaScript να χειρίζεται ασύγχρονες λειτουργίες και να διαχειρίζεται το DOM την καθιστά κατάλληλη για αυτές τις σύγχρονες, δυναμικές ιστοεφαρμογές.

Η αρχιτεκτονική χωρίς διακομιστή και η αυξημένη χρήση APIs συμβάλλουν επίσης στη συνάφεια της JavaScript. Με τεχνολογίες όπως το Node.js, οι προγραμματιστές μπορούν να χρησιμοποιούν συνεπώς την JavaScript τόσο για την πλευρά πελάτη όσο και για την πλευρά διακομιστή των εφαρμογών τους. Αυτή η ενοποίηση απλοποιεί την ανάπτυξη, προωθεί την επαναχρησιμοποίηση κώδικα και διευκολύνει την αποτελεσματική ανταλλαγή δεδομένων μεταξύ πελάτη και διακομιστή, συμβάλλοντας στην ευελιξία.

Το ζωντανό οικοσύστημα βιβλιοθηκών, πλαισίων και εργαλείων για την JavaScript είναι ένας ακόμη επιτακτικός λόγος υιοθέτησής της. Οι προγραμματιστές μπορούν να αξιοποιήσουν πλαίσια όπως το Express.js για την πλευρά διακομιστή, το React για διεπαφές χρήστη και διάφορες βοηθητικές βιβλιοθήκες. Αυτό το πλούσιο οικοσύστημα παρέχει τα κατάλληλα εργαλεία για κάθε έργο, ενισχύοντας την παραγωγικότητα και ενθαρρύνοντας την ανάπτυξη επεκτάσιμων και διατηρήσιμων ιστοεφαρμογών.

Συνολικά, η ευελιξία, η υποστήριξη της κοινότητας και η ευθυγράμμιση της JavaScript με τις σύγχρονες πρακτικές ανάπτυξης ιστού, την καθιστούν θεμελιώδη γλώσσα για τη δημιουργία εντυπωσιακών ιστοεφαρμογών στον σημερινό ψηφιακό κόσμο.

Κεφάλαιο 3

Μεθοδολογία Επιλογής Πλαισίων Ανάπτυξης Εφαρμογών Ιστού στην Javascript

3.1 Τρόπος αναζήτησης των πλαισίων

Ο εντοπισμός των καταλληλότερων πλαισίων ανάπτυξης ιστού απαιτεί μια ολοκληρωμένη στρατηγική αναζήτησης, η οποία πρέπει να περιλαμβάνει διάφορες πηγές και να χρησιμοποιεί αποτελεσματικά ερωτήματα αναζήτησης. Οι ακόλουθες υποενότητες περιγράφουν τις πηγές που χρησιμοποιήθηκαν και τα βασικά ερωτήματα αναζήτησης που χρησιμοποιήθηκαν κατά τη διαδικασία αναζήτησης.

3.1.1 Πηγές Ανάκτησης

1. Ακαδημαϊκά Περιοδικά και Εργασίες: Ακαδημαϊκές βάσεις δεδομένων, όπως το IEEE Xplore, η Ψηφιακή Βιβλιοθήκη της ACM [6], καθώς και το Google Scholar, υποβλήθηκαν σε έρευνα για τη διερεύνηση επιστημονικών άρθρων, ερευνητικών εργασιών και πρακτικών συνεδρίων που σχετίζονται με πλαίσια ανάπτυξης ιστού σε JavaScript. Αυτές οι πηγές παρέχουν πληροφορίες για τις πιο πρόσφατες εξελίξεις, συγκρίσεις και αξιολογήσεις διαφόρων πλαισίων.

2. Επίσημα Τεκμήρια και Ιστότοποι: Η επίσημη τεκμηρίωση και οι ιστότοποι δημοφιλών πλαισίων JavaScript, όπως το React, το Angular, το Vue.js και άλλα, εξετάστηκαν εκτενώς. Αυτές οι πηγές παρέχουν λεπτομερείς πληροφορίες σχετικά με τις δυνατότητες, τις λειτουργίες, τις μετρήσεις απόδοσης και την υποστήριξη της κοινότητας για κάθε πλαίσιο.

3. Κοινότητες και Φόρουμ Προγραμματιστών στο Διαδίκτυο: Πλατφόρμες όπως το Stack Overflow, το Reddit, το GitHub και το Quora ερευνήθηκαν προκειμένου να

αποκτηθούν πληροφορίες σχετικά με τις εμπειρίες, τις απόψεις και τις συστάσεις των προγραμματιστών σχετικά με τα πλαίσια ανάπτυξης ιστού JavaScript. Η συμμετοχή σε συζητήσεις, η ανάγνωση των νημάτων του φόρουμ και η ανάλυση των σχολίων της κοινότητας παρείχαν πολύτιμες προοπτικές σχετικά με τη χρηστικότητα, την επεκτασιμότητα και την ευκολία υιο

3.1.2 Βασικά Ερωτήματα Αναζήτησης

- "Καλύτερα πλαίσια ανάπτυξης ιστού JavaScript": Αυτό το ευρύ ερώτημα είχε ως στόχο τον εντοπισμό περιεκτικών λιστών, ταξινομήσεων και συγκρίσεων δημοφιλών πλαισίων JavaScript με βάση διάφορα κριτήρια όπως απόδοση, ευελιξία, οικοσύστημα και υποστήριξη κοινότητας.

- "Σύγκριση πλαισίων JavaScript": Αυτό το ερώτημα στόχευε άρθρα, αναρτήσεις ιστολογίου και συζητήσεις που συγκρίνονταν τα χαρακτηριστικά, τα πλεονεκτήματα και τα μειονεκτήματα διαφορετικών πλαισίων JavaScript. Αυτό βοήθησε να αποκαλυφθούν διαφοροποιημένες γνώσεις για τα δυνατά και τα αδύνατα σημεία κάθε πλαισίου από τεχνική άποψη.

- "Κορυφαία πλαίσια Frontend/Backend σε JavaScript": Εστίαση σε πλαίσια Frontend και Backend ξεχωριστά, αυτό το ερώτημα διευκόλυνε την εξερεύνηση εξειδικευμένων πλαισίων προσαρμοσμένων σε συγκεκριμένες απαιτήσεις ανάπτυξης, όπως σχεδιασμός διεπαφής χρήστη, διαχείριση δεδομένων και διακομιστής λογικής.

- Ερωτήματα για συγκεκριμένα πλαίσια: Τα προσαρμοσμένα ερωτήματα για συγκεκριμένα πλαίσια, όπως η "Σύγκριση React vs. Angular" ή η "Ανάλυση Απόδοσης του Vue.js", επέτρεψαν λεπτομερείς αξιολογήσεις και συγκρίσεις μεταξύ μεμονωμένων πλαισίων, βοηθώντας στη λήψη τεκμηριωμένων αποφάσεων.

Με βάση αυτές τις πηγές και τα ερωτήματα αναζήτησης, πραγματοποιήσαμε μια συστηματική εξερεύνηση του τοπίου των πλαισίων ανάπτυξης ιστού σε JavaScript, επιτρέποντάς μας να αξιολογήσουμε και να επιλέξουμε τα καταλληλότερα πλαίσια για τους ερευνητικούς μας στόχους.

3.2 Τρόπος Φιλτραρίσματος των Πλαισίων Ανάπτυξης Ιστού

Σε αυτήν την υποενότητα, αναλύονται τα κριτήρια που χρησιμοποιήθηκαν για το φιλτράρισμα και την αξιολόγηση πλαισίων ανάπτυξης ιστού για JavaScript. Η διαδικασία φιλτραρίσματος περιλαμβάνει τόσο την επιλογή των πλαισίων που ευθυγραμμίζονται με

τους ερευνητικούς στόχους, όσο και τον αποκλεισμό εκείνων που δεν πληρούν συγκεκριμένες προϋποθέσεις.

3.2.1 Κριτήρια Επιλογής

1. Λειτουργικότητα και Δυνατότητες: Τα επιλεγμένα πλαίσια πρέπει να προσφέρουν ένα ολοκληρωμένο σύνολο χαρακτηριστικών και λειτουργιών, απαραίτητων για την ανάπτυξη σύγχρονων διαδικτυακών εφαρμογών. Τέτοια κριτήρια περιλαμβάνουν υποστήριξη για αρχιτεκτονική βασισμένη σε στοιχεία, διαχείριση κατάστασης, δρομολόγηση, σύνδεση δεδομένων και ενσωμάτωση με βιβλιοθήκες ή APIs τρίτων.

2. Απόδοση και Επεκτασιμότητα: Τα πλαίσια πρέπει να επιδεικνύουν αποδοτική λειτουργία, ελάχιστο περιττό φορτίο και επεκτασιμότητα, ώστε να μπορούν να διαχειριστούν αυξανόμενα φορτία χρηστών και πολύπλοκες απαιτήσεις εφαρμογών. Κατά την αξιολόγηση, λαμβάνονται υπόψη μετρήσεις απόδοσης όπως η ταχύτητα απόκρισης, η κατανάλωση μνήμης και η αποτελεσματικότητα δικτύου.

3. Υποστήριξη Κοινότητας: Η ύπαρξη μιας ενεργής και δυναμικής κοινότητας είναι απαραίτητη για την εξασφάλιση υποστήριξης, πόρων και ενημερώσεων για τα επιλεγμένα πλαίσια. Τα κριτήρια περιλαμβάνουν το μέγεθος της κοινότητας προγραμματιστών, τη διαθεσιμότητα τεκμηρίωσης, πόρων εκμάθησης, προσθηκών και επεκτάσεων, καθώς και τη συχνότητα ενημερώσεων και νέων εκδόσεων.

4. Ευκολία Εκμάθησης και Υιοθέτησης: Τα πλαίσια θα πρέπει να προσφέρουν διαισθητικά APIs, σαφή τεκμηρίωση και καλά καθορισμένες βέλτιστες πρακτικές, ώστε να διευκολύνεται η εκμάθηση και η υιοθέτησή τους από προγραμματιστές με διαφορετικά επίπεδα δεξιοτήτων. Στα κριτήρια συμπεριλαμβάνονται η διαθεσιμότητα πόρων εκμάθησης, φόρουμ κοινότητας, διαδικτυακών σεμιναρίων και η ευκολία ενσωμάτωσης με υπάρχουσες βάσεις κώδικα ή έργα.

5. Συμβατότητα και Διαλειτουργικότητα: Τα επιλεγμένα πλαίσια θα πρέπει να ενσωματώνονται απρόσκοπτα με άλλα εργαλεία, βιβλιοθήκες και τεχνολογίες που χρησιμοποιούνται συχνά στην ανάπτυξη ιστού, όπως συστήματα κατασκευής, διαχειριστές πακέτων, πλαίσια δοκιμών και πλατφόρμες ανάπτυξης. Τα κριτήρια περιλαμβάνουν τη συμβατότητα με δημοφιλή προγράμματα περιήγησης, λειτουργικά συστήματα και περιβάλλοντα ανάπτυξης.

"

3.2.2 Κριτήρια Αποκλεισμού

1. Παρωχημένα ή Εγκαταλελειμμένα Πλαίσια: Τα πλαίσια που δεν διατηρούνται πλέον ενεργά, στερούνται υποστήριξης από την κοινότητα ή έχουν καταστεί ξεπερα-

σμένα λόγω τεχνολογικών εξελίξεων, αποκλείονται από την αξιολόγηση. Τέτοια κριτήρια περιλαμβάνουν την παλιά ή ελλιπή τεκμηρίωση, τις σπάνιες ενημερώσεις και τις φθίνουσες τάσεις χρήσης στην κοινότητα προγραμματιστών.

2. Χαμηλή Απόδοση και Ασταθής Λειτουργία: Πλαίσια που παρουσιάζουν σημαντικά προβλήματα απόδοσης, ζητήματα σταθερότητας ή αρχιτεκτονικούς περιορισμούς, αποκλείονται από περαιτέρω αξιολόγηση. Στα κριτήρια περιλαμβάνονται συχνές διακοπές λειτουργίας, διαρροές μνήμης, απρόβλεπτη συμπεριφορά και ανεπαρκής υποστήριξη για σύγχρονα πρότυπα ιστού και δυνατότητες των προγραμμάτων περιήγησης.

3. Περιορισμένη Υιοθέτηση από την Κοινότητα: Πλαίσια με μικρή ή στάσιμη βάση χρηστών, περιορισμένη συμμετοχή της κοινότητας και ανεπαρκείς πόρους για εκμάθηση και υποστήριξη, αποκλείονται. Τέτοια κριτήρια περιλαμβάνουν τη χαμηλή δραστηριότητα σε φόρουμ κοινότητας, την ελλιπή τεκμηρίωση και την έλλειψη προσθηκών ή επεκτάσεων τρίτων.

4. Περιορισμοί Αδειοδότησης και Εξάρτησης από Προμηθευτή: Τα πλαίσια που επιβάλλουν περιοριστικές συμφωνίες αδειοδότησης, εξάρτηση από συγκεκριμένο προμηθευτή ή αποκλειστικές τεχνολογίες, αποκλείονται. Στα κριτήρια συμπεριλαμβάνονται το απαγορευτικό κόστος, οι περιορισμένες δυνατότητες προσαρμογής και οι νομικές επιπτώσεις που σχετίζονται με τη χρήση ιδιόκτητου λογισμικού.

Με την εφαρμογή αυτών των κριτηρίων επιλογής και αποκλεισμού, διασφαλίζεται μια συστηματική και αυστηρή διαδικασία αξιολόγησης για τον εντοπισμό των καταλληλότερων πλαισίων ανάπτυξης ιστού για JavaScript, τα οποία ευθυγραμμίζονται με τους ερευνητικούς στόχους και τις απαιτήσεις του έργου.

3.3 Επιλεγμένα Πλαίσια Ανάπτυξης Ιστού

Το τοπίο της ανάπτυξης ιστοσελίδων εξελίσσεται συνεχώς, καθοδηγούμενο από καινοτομίες στην τεχνολογία, αλλαγές στις προσδοκίες των χρηστών και την εμφάνιση νέων παραδειγμάτων και μεθοδολογιών. Εντός αυτού του δυναμικού οικοσυστήματος, η επιλογή των κατάλληλων πλαισίων διαδραματίζει καθοριστικό ρόλο στην επιτυχία των έργων ανάπτυξης διαδικτυακών εφαρμογών. Σε αυτήν την ενότητα, παρουσιάζονται τα επιλεγμένα πλαίσια που προέκυψαν από τη διαδικασία αναζήτησης και φιλτραρίσματος για πλαίσια JavaScript, περιλαμβάνοντας πλαίσια frontend, backend και fullstack.

3.3.1 Frontend Πλαίσια JavaScript

Στην κατηγορία Frontend επιλέχθηκαν πέντε πλαίσια ανάπτυξης. Αρχικά, το Angular είναι ένα ολοκληρωμένο πλαίσιο frontend που διατηρεί η Google . Προσφέρει ένα ισχυρό σύνολο δυνατοτήτων για τη δημιουργία εφαρμογών μιας σελίδας (Single Page Application-SPA) και προοδευτικών εφαρμογών ιστού (Progressive Web Apps-PWA) . Επιπλέον, το Angular ακολουθεί την αρχιτεκτονική που βασίζεται σε στοιχεία και παρέχει ισχυρά εργαλεία για σύνδεση δεδομένων, έγχυση εξάρτησης και δρομολόγηση. Ταυτόχρονα, προσφέρει μία πλούσια συλλογή βιβλιοθηκών, εργαλείων και επεκτάσεων, επιτρέποντας στους προγραμματιστές να δημιουργούν διαδραστικές και ανταποκρινόμενες διεπαφές χρήστη.

Το δεύτερο πλαίσιο που επιλέχθηκε για αυτήν την κατηγορία είναι το React , το οποίο είναι μια δημοφιλής βιβλιοθήκη JavaScript που αναπτύχθηκε από το Facebook για τη δημιουργία διεπαφών χρήστη. Ακολουθεί μια προσέγγιση που βασίζεται σε στοιχεία, επιτρέποντας στους προγραμματιστές να δημιουργούν επαναχρησιμοποιήσιμα στοιχεία διεπαφής χρήστη και να διαχειρίζονται την κατάσταση αποτελεσματικά. Ακόμη, το React χρησιμοποιεί ένα εικονικό DOM για αποτελεσματική απόδοση, έχοντας ως συνέπεια την υψηλή απόδοση και ανταπόκριση. Αξίζει να σημειωθεί πως το συγκεκριμένο πλαίσιο χρησιμοποιείται ευρέως για τη δημιουργία δυναμικών εφαρμογών ιστού, εφαρμογών για κινητά και στοιχείων διεπαφής χρήστη.

Στην συνέχεια, το επόμενο frontend πλαίσιο ονομάζεται Ember . Το Ember είναι ένα παραγωγικό πλαίσιο frontend σχεδιασμένο για τη δημιουργία μεγαλεπήβολων διαδικτυακών εφαρμογών. Μέσα από αυτό, παρέχεται μια ισχυρή προσέγγιση συμβατικών παραμέτρων, απλοποιώντας τη διαδικασία ανάπτυξης και μειώνοντας τον κώδικα boilerplate .Το Ember προσφέρει επίσης ενσωματωμένη υποστήριξη για τη διαχείριση δεδομένων, τη δρομολόγηση και τη διαχείριση κατάστασης, επιτρέποντας στους προγραμματιστές να επικεντρωθούν στα χαρακτηριστικά του building και όχι στην υποδομή. Επιπλέον, παρέχει ένα πλούσιο οικοσύστημα από πρόσθετα, ενισχύοντας την παραγωγικότητα των προγραμματιστών και τη διατήρηση του κώδικα.

Έπειτα, το Svelte αποτελεί ένα από το επιλεγμένα πλαίσια που θα μελετηθούν. Το selectlanguageenglishSvelte είναι ένα σύγχρονο πλαίσιο JavaScript που μετατοπίζει τη βαρύτητα των πλαισίων διεπαφής χρήστη από το χρόνο εκτέλεσης στο στάδιο μεταγλώττισης. Έτσι, μεταγλωττίζει στοιχεία σε εξαιρετικά βελτιστοποιημένο κώδικα JavaScript που ενημερώνει αποτελεσματικά το DOM . Η προσέγγιση του Svelte επιτρέπει μικρότερα μεγέθη bundle και καλύτερη απόδοση σε σύγκριση με τα παραδοσιακά πλαίσια. Συνεπώς, προσφέρει μια απλή και διαισθητική σύνταξη για τη δημιουργία αντιδραστικών στοιχείων, καθιστώντας την εύκολη εκμάθηση και χρήση.

Το Vue.js αποτελεί το τελευταίο frontent πλαίσιο που επιλέχθηκε. Το Vue.js είναι ένα προοδευτικό πλαίσιο JavaScript για τη δημιουργία διεπαφών χρήστη καθώς παρέχει

μία ευέλικτη και προσιτή βιβλιοθήκη για τη δημιουργία διαδραστικών διαδικτυακών εφαρμογών. Επιπλέον, το εν λόγω πλαίσιο, ακολουθεί μια αρχιτεκτονική βασισμένη σε συστατικά, προσφέρει αντιδραστική δέσμευση δεδομένων, υπολογισμένες ιδιότητες και δηλωτικά πρότυπα αλλά και δίνει έμφαση στην απλότητα και την επεκτασιμότητα, καθιστώντας το κατάλληλο τόσο για έργα μικρής κλίμακας όσο και για εφαρμογές μεγάλης κλίμακας.

3.3.2 Backend Πλαίσια JavaScript

Ένα από τα επιλεγμένα πλαίσια backend είναι το Express [7] . Αυτό το πλαίσιο αποτελεί ένα από τα πιο μινιμαλιστικά πλαίσια ιστού για το Node.js , σχεδιασμένο για τη δημιουργία εφαρμογών ιστού και API . Το Express παρέχει ένα ισχυρό σύνολο λειτουργιών για δρομολόγηση, ενδιάμεσο λογισμικό και χειρισμό αιτημάτων. Επίσης, ακολουθεί το μοτίβο "μεσαίου λογισμικού", επιτρέποντας στους προγραμματιστές να συνθέτουν αρθρωτά και επαναχρησιμοποιήσιμα στοιχεία για το χειρισμό αιτημάτων και απαντήσεων HTTP , προσφέροντας μια ελαφριά και ευέλικτη αρχιτεκτονική, καθιστώντας το ιδανικό για την κατασκευή μικροϋπηρεσιών και εφαρμογών από την πλευρά του διακομιστή.

Προχωρώντας, το Koa [8] είναι ένα σύγχρονο πλαίσιο web για το Node.js , που αναπτύχθηκε από τους δημιουργούς του Express . Με την χρήση αυτού του πλαισίου, προσφέρεται ένα πιο εκφραστικό και βελτιωμένο API σε σύγκριση με το Express , αξιοποιώντας χαρακτηριστικά ES6 όπως το async/wait για το χειρισμό ασύγχρονων λειτουργιών. Το Koa , επίσης, δίνει έμφαση στη σύνθεση ενδιάμεσου λογισμικού και στον χειρισμό σφαλμάτων, επιτρέποντας την ανάπτυξη καθαρού και διατηρήσιμου κώδικα, επιτρέποντας στους προγραμματιστές να χρησιμοποιούν μόνο τις δυνατότητες που χρειάζονται.

Το προτελευταίο επιλεγμένο backend είναι γνωστό με το όνομα Fastify [9] . Αυτό το πλαίσιο, αποτελεί ένα γρήγορο και χαμηλού κόστους πλαίσιο web για το Node.js , βελτιστοποιημένο για απόδοση και αποδοτικότητα. Προσφέρει μια αρχιτεκτονική που βασίζεται σε πρόσθετα που επιτρέπει στους προγραμματιστές να επεκτείνουν και να προσαρμόσουν εύκολα τη λειτουργικότητα. Το Fastify δίνει έμφαση στην απόδοση, παρέχοντας ενσωματωμένη υποστήριξη για λειτουργίες όπως η επικύρωση βάσει σχήματος, η σειριοποίηση και η προσωρινή αποθήκευση. Είναι κατάλληλο για τη δημιουργία API και μικροϋπηρεσιών υψηλής απόδοσης, προσφέροντας ισορροπία μεταξύ ταχύτητας, ευελιξίας και εμπειρίας προγραμματιστή.

Ακολούθως, το τελευταίο πλαίσιο που επιλέχθηκε λέγεται Hono [10] . Το Hono είναι ένα ελαφρύ και αρθρωτό πλαίσιο υποστήριξης για το Node.js , σχεδιασμένο για τη δημιουργία μικροϋπηρεσιών και εφαρμογών χωρίς διακομιστή, παρέχοντας μια μινιμαλιστική και σκεπτόμενη προσέγγιση για την ανάπτυξη backend , εστιάζοντας στην απλότητα και την απόδοση. Το Hono προσφέρει ενσωματωμένη υποστήριξη για λειτουργίες όπως δρομολόγηση, διαχείριση αιτημάτων και διαχείριση σφαλμάτων. Πέρα από αυτά, δίνει

έμφαση στην επεκτασιμότητα, επιτρέποντας στους προγραμματιστές να ενσωματώνουν εύκολα προσθήκες και ενδιάμεσο λογισμικό τρίτων.

3.3.3 Fullstack Πλαίσια JavaScript

Όσον αφορά τα fullstack πλαίσια, επιλέχθηκαν τρία εξ' αυτών προς ανάλυση. Το πρώτο από αυτά ονομάζεται Meteor [11] . Το Meteor είναι ένα πλαίσιο Fullstack JavaScript για τη δημιουργία σύγχρονων εφαρμογών ιστού και κινητών. Παρέχει μια ολοκληρωμένη λύση για την ανάπτυξη και τη διαχείριση εφαρμογών, συμπεριλαμβανομένων και των στοιχείων frontend και backend . Το Meteor προσφέρει ένα αντιδραστικό σύστημα δεδομένων, ενημερώσεις σε πραγματικό χρόνο και απρόσκοπτη ενσωμάτωση με δημοφιλείς βιβλιοθήκες frontend όπως React και Vue.js .Έτσι, προσφέρει μια ενοποιημένη εμπειρία ανάπτυξης, επιτρέποντας στους προγραμματιστές να δημιουργούν και να αναπτύσσουν εφαρμογές με ελάχιστη διαμόρφωση.

Κατόπιν, το Feathers [12] συνιστά ένα ελαφρύ και ευέλικτο πλαίσιο για τη δημιουργία εφαρμογών σε πραγματικό χρόνο και RESTful API με JavaScript και TypeScript . Αυτό το πλαίσιο, διαθέτει μια αρχιτεκτονική που επιτρέπει στους προγραμματιστές να συνδυάζουν και να ταιριάζουν χαρακτηριστικά με βάση τις απαιτήσεις του έργου. Επιπλέον, το Feathers παρέχει ενσωματωμένη υποστήριξη για έλεγχο ταυτότητας, εξουσιοδότηση και ενσωμάτωση βάσεων δεδομένων, καθιστώντας εύκολη τη δημιουργία ασφαλών και επεκτάσιμων εφαρμογών.Συνολικά, αυτό το πλαίσιο προσφέρει απρόσκοπτη ενοποίηση με πλαίσια frontend όπως το Angular [13] και το Vue.js [14] .

Τέλος, το Next.js [15] είναι ένα πλαίσιο με επιρροή σχεδιασμένο για τη δημιουργία εφαρμογών React που αποδίδονται στον διακομιστή. Με την χρήση αυτού του πλαισίου προσφέρεται μια ολοκληρωμένη λύση για την κατασκευή διαδικτυακών εφαρμογών έτοιμων για παραγωγή, με απόδοση από την πλευρά του διακομιστή, δημιουργία στατικών τοποθεσιών και αυτοματοποιημένη κατάτμηση κώδικα. Το Next.js βελτιστοποιεί τη διαδικασία ανάπτυξης με λειτουργίες όπως η αντικατάσταση hot module , η αυτόματη δρομολόγηση και η εγγενής υποστήριξη CSS . Οι εξαιρετικές του επιδόσεις, οι δυνατότητες βελτιστοποίησης μηχανών αναζήτησης και τα φιλικά προς τους προγραμματιστές χαρακτηριστικά του έχουν σταθεροποιήσει την κατάστασή του ως προτιμώμενη επιλογή για την ανάπτυξη σύγχρονων εφαρμογών Ιστού.

Το Next.JS είναι ιδιάιτερα διαδεμένο σήμερα ανάμεσα στα JS πλαίσια, και ο λόγος είναι κυρίως η απόδοσή του. Το Next.js παρέχει τρεις κύριες στρατηγικές απόδοσης, καθεμία από τις οποίες προσφέρει μοναδικές συνέπειες για το SEO και την απόδοση:

Στατική απεικόνιση: Εδώ, οι σελίδες δημιουργούνται κατά το χρόνο κατασκευής, αποθηκεύονται στην προσωρινή μνήμη και διανέμονται μέσω ενός Δικτύου Παράδοσης Περιεχομένου (CDN). Χάρη στην προβλέψιμη φύση του και στους γρήγορους χρόνους φόρτωσης, είναι ιδανικό για περιεχόμενο που δεν αλλάζει συχνά, όπως άρθρα ιστολογίου.

Δυναμική απεικόνιση: Οι σελίδες δημιουργούνται ανά αίτημα χρήστη, διευκολύνοντας το εξατομικευμένο περιεχόμενο. Αν και αυτό προσφέρει προσαρμοσμένες εμπειρίες χρήστη, μπορεί να είναι λίγο πιο αργό, επηρεάζοντας δυνητικά το SEO. Ωστόσο, με τις σωστές τροποποιήσεις, μπορεί να διατηρηθεί η βέλτιστη απόδοση.

Ροή: Ο διακομιστής αποδίδει σταδιακά τη διεπαφή χρήστη σε τμήματα. Αυτή η μέθοδος μπορεί να βελτιώσει την αντιληπτή απόδοση, επιτρέποντας στους χρήστες να αλληλεπιδρούν με τμήματα μιας σελίδας ακόμη και πριν ολοκληρωθεί η απεικόνιση του πλήρους περιεχομένου — ένα απόλυτο πλεονέκτημα για περιεχόμενο που βασίζεται σε πιο αργές ανακτήσεις δεδομένων [16].

Κεφάλαιο 4

Ανάλυση Πλαισίων Ανάπτυξης Εφαρμογών Ιστού

4.1 Κατηγοριοποίηση Πλαισίων

4.1.1 Σύνδεση δεδομένων

Η σύνδεση δεδομένων (Data Binding) είναι μια τεχνική που χρησιμοποιείται στην ανάπτυξη λογισμικού για τη δημιουργία σύνδεσης ή συγχρονισμού μεταξύ του UI και του μοντέλου δεδομένων της εφαρμογής. Επιτρέπει στις αλλαγές δεδομένων να αντικατοπτρίζονται αυτόματα στη διεπαφή χρήστη και αντίστροφα, χωρίς να απαιτείται μη αυτόματη παρέμβαση για την ενημέρωση των στοιχείων διεπαφής χρήστη [17].

Σε ένα τυπικό σενάριο δέσμευσης δεδομένων:

Πηγή δεδομένων: Εδώ βρίσκονται τα δεδομένα της εφαρμογής. Στην κατηγορία αυτή ανήκει για παράδειγμα κάποια βάση δεδομένων, ένα αντικείμενο, ένα αρχείο ή οποιαδήποτε άλλη πηγή δεδομένων.

Στοιχείο διεπαφής χρήστη: Αυτά είναι τα οπτικά στοιχεία της διεπαφής χρήστη της εφαρμογής, όπως πλαίσια κειμένου, ετικέτες, κουμπιά κ.λπ.

Binding Engine: Αυτό είναι το στοιχείο που είναι υπεύθυνο για τη δημιουργία της σύνδεσης μεταξύ της πηγής δεδομένων και των στοιχείων διεπαφής χρήστη.

Υπάρχουν δύο κύριοι τύποι δέσμευσης δεδομένων:

Μονόδρομη δέσμευση δεδομένων(One-way Data Binding): Σε αυτόν τον τύπο, τα δεδομένα ρέουν μόνο προς μία κατεύθυνση, είτε από την πηγή δεδομένων προς το στοιχείο διεπαφής χρήστη (γνωστή ως δέσμευση πηγής προς στόχο) είτε από το στοιχείο διεπαφής χρήστη προς την πηγή δεδομένων (γνωστή ως στόχος- δέσμευση προς την πηγή).

Αμφίδρομη σύνδεση δεδομένων(Two-way Data Binding): Αυτός ο τύπος δέσμευσης δεδομένων επιτρέπει την αμφίδρομη μετάδοση των αλλαγών δεδομένων μεταξύ της πηγής δεδομένων και του στοιχείου διεπαφής χρήστη. Οποιαδήποτε αλλαγή στο στοιχείο διεπαφής χρήστη ενημερώνει την προέλευση δεδομένων και οποιαδήποτε αλλαγή στην

πηγή δεδομένων ενημερώνει το στοιχείο διεπαφής χρήστη.

Η σύνδεση δεδομένων απλοποιεί την ανάπτυξη μειώνοντας την ποσότητα του κώδικα που απαιτείται για τη διαχείριση του συγχρονισμού μεταξύ της διεπαφής χρήστη και των υποκείμενων δεδομένων.

Σύγκριση Τύπων Σύνδεσης δεδομένων

Όσον αφορά τη σύγκριση μεταξύ της μονόδρομης και της αμφίδρομης σύνδεσης δεδομένων στα πλαίσια μιας εφαρμογής, δεν υπάρχει μια απόλυτη απάντηση για το ποια είναι καλύτερη ή πιο προτιμότερη. Εξαρτάται από τις απαιτήσεις και τις ανάγκες της συγκεκριμένης εφαρμογής.

Η μονόδρομη σύνδεση είναι πιο απλή και λιγότερο σύνθετη, κατάλληλη για περιπτώσεις όπου τα δεδομένα είναι μόνο για ανάγνωση ή όταν η προβολή δεν χρειάζεται να τροποποιεί τα δεδομένα. Είναι πιο εύκολη στην υλοποίηση και διαχείριση, με λιγότερες πιθανότητες για ανεπιθύμητες παρενέργειες. Ωστόσο, περιορίζει τη διαδραστικότητα της εφαρμογής, καθώς ο χρήστης δεν μπορεί να τροποποιήσει τα δεδομένα μέσω της προβολής.

Από την άλλη πλευρά, η αμφίδρομη σύνδεση είναι πιο σύνθετη, αλλά απαραίτητη για διαδραστικές εφαρμογές όπου ο χρήστης πρέπει να μπορεί να επεξεργάζεται και να τροποποιεί τα δεδομένα μέσω της προβολής. Επιτρέπει τη ροή δεδομένων και προς τις δύο κατευθύνσεις, διευκολύνοντας τη δυναμική αλληλεπίδραση μεταξύ μοντέλου και προβολής. Ωστόσο, απαιτεί περισσότερη προσοχή και έλεγχο για την αποφυγή ανεπιθύμητων παρενεργειών ή κύκλων ενημέρωσης δεδομένων.

Συνεπώς, η επιλογή μεταξύ μονόδρομης και αμφίδρομης σύνδεσης εξαρτάται από το εάν η εφαρμογή απαιτεί διαδραστικότητα και επεξεργασία δεδομένων από τον χρήστη ή όχι. Εάν η διαδραστικότητα είναι κρίσιμη, τότε η αμφίδρομη σύνδεση είναι προτιμότερη, παρά τη μεγαλύτερη πολυπλοκότητά της. Εάν όμως τα δεδομένα είναι στατικά και μόνο για ανάγνωση, η μονόδρομη σύνδεση μπορεί να είναι πιο κατάλληλη λόγω της απλότητάς της.

4.1.1.1 Σύνδεση δεδομένων στα Front-End πλαίσια

Στον Πίνακα 4.1 παρουσιάζεται μια σύγκριση των Front-End πλαισίων όσον αφορά την σύνδεση δεδομένων τους.

4.1.1.2 Σύνδεση δεδομένων στα Back-End πλαίσια

Τα πλαίσια Express, Koa, Hono και Fastify δεν χρησιμοποιούν άμεσα τις έννοιες της αμφίδρομης ή μονόδρομης σύνδεσης δεδομένων καθώς αυτές οι έννοιες σχετίζονται περισσότερο με εφαρμογές με γραφικό περιβάλλον χρήστη.

Πλαίσιο	Σύνδεσης Δεδομένων	Λεπτομέρειες
Angular	Αμφίδρομη	Σύνδεση δεδομένων με την χρήση της σύνταξης (ngModel).
React	Μονόδρομη	Δέσμευση δεδομένων με την χρήση των props και state.
Ember	Αμφίδρομη	Σύνδεση δεδομένων με την χρήση της εντολής computed.alias() η οποία ορίζει το μονοπάτι προς κάποιο άλλο αντικείμενο.
Svelte	Μονόδρομη	Παρακολουθεί τις εξαρτήσεις μεταξύ αντιδραστικών μεταβλητών και στοιχείων DOM. Όταν μια μεταβλητή αλλάζει, γνωρίζει ποια στοιχεία είναι συνδεδεμένα σε αυτήν και τα ενημερώνει αποτελεσματικά.
Vue	Αμφίδρομη	Σύνδεση δεδομένων με την οδηγία v-model, υποστηρίζει επίσης τη μονόδρομη δέσμευση δεδομένων.

Πίνακας 4.1: Σύγκριση των FrontEnd JavaScript πλαισίων σχετικά με την σύνδεση δεδομένων

Αυτά τα πλαίσια εστιάζουν στη δημιουργία διακομιστών (server-side) και τη διαχείριση αιτημάτων HTTP. Δεν έχουν ενσωματωμένο μηχανισμό για σύνδεση δεδομένων με τα Views στην πλευρά του πελάτη.

Ωστόσο, όταν αυτά τα πλαίσια χρησιμοποιούνται σε συνδυασμό με βιβλιοθήκες στην πλευρά του πελάτη, όπως το React, Angular ή Vue.js, τότε η σύνδεση δεδομένων θα καθορίζεται από την αρχιτεκτονική και τον μηχανισμό της συγκεκριμένης βιβλιοθήκης.

4.1.1.3 Σύνδεση δεδομένων στα Full-Stack πλαίσια

Όσον αφορά τη σύνδεση δεδομένων, τα full-stack πλαίσια Feathers, Next.js και Meteor δεν έχουν ενσωματωμένη υποστήριξη για αμφίδρομη ή μονόδρομη σύνδεση δεδομένων. Αυτές οι έννοιες σχετίζονται περισσότερο με το χειρισμό δεδομένων και την ενημέρωση των προβολών στην πλευρά του πελάτη.

Τα Feathers, Next.js και Meteor είναι full-stack πλαίσια που καλύπτουν τόσο την πλευρά του διακομιστή όσο και την πλευρά του πελάτη. Ωστόσο, η σύνδεση δεδομένων στην πλευρά του πελάτη θα εξαρτηθεί από το πλαίσιο ή τη βιβλιοθήκη που χρησιμοποιείται για την ανάπτυξη της διεπαφής χρήστη.

4.1.2 Αρχιτεκτονικές Front-End Πλαισίων

Όσον αφορά τα front-end πλαίσια, τα είδη αρχιτεκτονικών που συναντάμε είναι τρία. Η πιο συνηθισμένη και διαδεδομένη αρχιτεκτονική σχεδίασης με κάποιο πλαίσιο είναι

η MVC, που χωρίζει την υλοποίηση σε τρία βασικά κομμάτια, το τι βλέπει ο χρήστης, το πως η εφαρμογή διαχειρίζεται την είσοδο του χρήστη και πως αλληλεπιδρά με τρίτες υπηρεσίας (π.χ μια βάση δεδομένων) και τα δεδομένα της εφαρμογής. Ενώ η παραπάνω αρχιτεκτονική είναι η πιο διαδεδομένη, συναντάται σε πλαίσια που παρέχουν μια πιο ολοκληρωμένη λειτουργικότητα. Ωστόσο, τα περισσότερα front-end πλαίσια, όπως είναι λογικό, επικεντρώνονται στην υλοποίηση και ανάπτυξη προγραμμάτων που αφορούν το UI, και συνεπώς δεν παρέχουν κάποια από τις συνηθισμένες αρχιτεκτονικές. Ωστόσο, έχει ενδιαφέρον να παρατηρήσουμε, τον τρόπο με τον οποίο κάθε πλαίσιο επιτρέπει στον προγραμματιστεί να αναπτύξει εφαρμογές και την προσέγγιση που κάθε ένα από αυτά χρησιμοποιεί. Παρακάτω θα περιγράψουμε εν συντομία τις διαφορετικές αρχιτεκτονικές που θα συναντήσουμε.

Αρχιτεκτονική MVC

Το μοντέλο Model–view–controller (MVC) σχεδιάστηκε αρχικά για διασυνδέσεις χρήστη σε εφαρμογές που υλοποιήθηκαν με Smalltalk, αλλά από τότε έχει εξελιχθεί σε ένα μοντέλο σχεδίασης για διασυνδέσεις χρήστη ανεξαρτήτως γλώσσας υλοποίησης και για web εφαρμογές των οποίων τα στοιχεία ελέγχου αλλάζουν συχνά. Η αρχιτεκτονική MVC χωρίζει το διαδραστικό σύστημα σε τρία εξειδικευμένα στοιχεία. Το Model περιέχει τα δεδομένα της εφαρμογής και διαχειρίζεται τη βασική λειτουργικότητα. Το View διαχειρίζεται την οπτική απεικόνιση του μοντέλου και την ανατροφοδότηση προς τον χρήστη. Ο Controller ερμηνεύει τις εισόδους του ποντικιού και του πληκτρολογίου από τον χρήστη, δίνοντας εντολές στο μοντέλο και την προβολή να αλλάξουν κατάλληλα[18].

Αρχιτεκτονικη MVVM

Το Model-View-ViewModel (MVVM) είναι ένα πρότυπο αρχιτεκτονικής λογισμικού που χρησιμοποιείται κυρίως στην ανάπτυξη διασυνδέσεων χρήστη. Σκοπεύει στον διαχωρισμό της λογικής της διασύνδεσης χρήστη από τα υποκείμενα δεδομένα και τη λογική επιχειρηματικής λειτουργίας, προωθώντας έναν πιο αρθρωτό και διατηρήσιμο κώδικα.

Στο MVVM, το Model αντιπροσωπεύει τα δεδομένα και τη λογική επιχειρηματικής λειτουργίας της εφαρμογής. Περιλαμβάνει τις οντότητες δεδομένων, όπως τα στοιχεία χρήστη ή τις λεπτομέρειες προϊόντων, και ορίζει τη συμπεριφορά τους, όπως τον έλεγχο δεδομένων ή την ανάκτηση από μια βάση δεδομένων.

Το View αντιπροσωπεύει τα στοιχεία της διασύνδεσης χρήστη με τα οποία αλληλεπιδρούν οι χρήστες, όπως κουμπιά, φόρμες ή λίστες. Σε αντίθεση με τα παραδοσιακά πρότυπα MVC, το View στο MVVM είναι συνήθως πιο παθητική και χωρίς λογική εφαρμογής. Αντ' αυτού, συνδέεται απευθείας με το ViewModel. Το ViewModel λειτουργεί ως μεσολαβητής μεταξύ του Model και του View. Εκθέτει τα δεδομένα και τις λειτουργίες που απαιτούνται από το View, μετατρέποντας τα δεδομένα από το Model σε μια μορφή

που μπορεί εύκολα να εμφανιστεί και να αλληλεπιδράσει το View. Το ViewModel ενσωματώνει επίσης συνήθως λογική παρουσίασης, όπως τη μορφοποίηση ημερομηνιών ή τον χειρισμό του ελέγχου των δεδομένων εισόδου του χρήστη. Με τον διαχωρισμό των ευθυνών με αυτόν τον τρόπο, το MVVM προωθεί την επαναχρησιμοποίηση, τη δυνατότητα δοκιμής και τη διατήρηση του κώδικα, καθιστώντας το μια δημοφιλή επιλογή για την ανάπτυξη σύγχρονων εφαρμογών web και κινητών.

Αρχιτεκτονική Βασισμένη σε Στοιχεία

Η Αρχιτεκτονική βασισμένη σε στοιχεία (Component-Based Architecture-CBA) αποτελεί μια ολοένα και πιο δημοφιλή προσέγγιση στον σχεδιασμό και την υλοποίηση λογισμικού. Βασίζεται στην ιδέα της δημιουργίας λογισμικού από επαναχρησιμοποιήσιμα, ανεξάρτητα συστατικά, τα οποία ονομάζονται components. Κάθε συστατικό αποτελεί μια αυτόνομη μονάδα με συγκεκριμένη λειτουργικότητα, ενώ η σύνδεση και ο συνδυασμός τους επιτρέπει την ανάπτυξη σύνθετων εφαρμογών. Η αρχιτεκτονική CBA προσφέρει μια σειρά από πλεονεκτήματα, όπως η επαναχρησιμοποίηση κώδικα, η αυξημένη συντηρησιμότητα, η ευελιξία και η επεκτασιμότητα. Ως αποτέλεσμα, η CBA υιοθετείται ευρέως σε διάφορους τομείς, όπως οι web εφαρμογές, τα κινητά apps και οι εφαρμογές υπολογιστή[19].

Αρχιτεκτονική Προσανατολισμένη στις Υπηρεσίες

Η Αρχιτεκτονική Προσανατολισμένη στις Υπηρεσίες (Service-Oriented Architecture-SOA) αποτελεί ένα αρχιτεκτονικό στυλ λογισμικού που εστιάζει σε διακριτές υπηρεσίες, αντί για ένα μονολιθικό σχεδιασμό. Βασίζεται στην ιδέα της δημιουργίας αυτόνομων, επαναχρησιμοποιούμενων υπηρεσιών που μπορούν να συνδυαστούν για την ανάπτυξη σύνθετων εφαρμογών.

Κάθε υπηρεσία SOA υλοποιεί μια συγκεκριμένη λειτουργικότητα και προσφέρει ένα τυποποιημένο interface για την επικοινωνία με άλλες υπηρεσίες. Η επικοινωνία πραγματοποιείται μέσω πρωτοκόλλων δικτύου, όπως HTTP, SOAP και REST, καθιστώντας τις υπηρεσίες SOA ανεξάρτητες από την πλατφόρμα και τη γλώσσα προγραμματισμού.

Η SOA προσφέρει πλήθος πλεονεκτημάτων, όπως:

Ευελιξία: Η SOA επιτρέπει την εύκολη τροποποίηση και επέκταση εφαρμογών, καθώς οι υπηρεσίες μπορούν να προστεθούν, να αφαιρεθούν ή να τροποποιηθούν χωρίς να επηρεαστεί το υπόλοιπο σύστημα. **Επαναχρησιμοποίηση:** Οι υπηρεσίες SOA μπορούν να επαναχρησιμοποιηθούν σε διάφορες εφαρμογές, μειώνοντας τον χρόνο ανάπτυξης και το κόστος. **Συνδεσιμότητα:** Η SOA επιτρέπει την εύκολη ενσωμάτωση με υπάρχοντα συστήματα και εφαρμογές. **Κλιμακωσιμότητα:** Η SOA επιτρέπει την εύκολη κλιμάκωση εφαρμογών για να καλύψουν αυξημένες απαιτήσεις.

4.1.2.1 Αρχιτεκτονικές στα Front-End πλαίσια

Όσον αφορά την καλύτερη και ποιοτικότερη αρχιτεκτονική, δεν υπάρχει μια απόλυτη απάντηση που να ισχύει για όλες τις περιπτώσεις. Ωστόσο, μπορούμε να εξετάσουμε τα πλεονεκτήματα και τα μειονεκτήματα της καθεμίας για να κατανοήσουμε ποια ταιριάζει καλύτερα σε διαφορετικά σενάρια.

Η Αρχιτεκτονική βασισμένη σε Στοιχεία είναι μια πολύ δημοφιλής και ευέλικτη προσέγγιση που ενδείκνυται για εφαρμογές με πλούσια διεπαφή χρήστη. Τα επαναχρησιμοποιήσιμα στοιχεία διευκολύνουν την ανάπτυξη και τη συντήρηση, ενώ η ανεξαρτησία των στοιχείων προσφέρει μεγαλύτερη επεκτασιμότητα. Ωστόσο, η διαχείριση της κατάστασης και της αλληλεπίδρασης μεταξύ των στοιχείων μπορεί να γίνει πολύπλοκη σε μεγάλες εφαρμογές.

Η κλασική Αρχιτεκτονική MVC είναι μια δοκιμασμένη και καλά κατανοητή προσέγγιση που λειτουργεί καλά για εφαρμογές με έντονη αλληλεπίδραση με τον χρήστη. Ο διαχωρισμός της λογικής σε Model, View και Controller διευκολύνει τη συντήρηση και την επαναχρησιμοποίηση. Ωστόσο, σε πολύπλοκες εφαρμογές, οι Controllers μπορεί να γίνουν δύσκολοι στη διαχείριση και την τροποποίηση.

Η Αρχιτεκτονική MVVM είναι μια πιο σύγχρονη προσέγγιση που συνδυάζει πλεονεκτήματα από τις άλλες δύο. Το ViewModel διαχωρίζει τη λογική παρουσίασης από την προβολή, διευκολύνοντας τη δοκιμή και την επαναχρησιμοποίηση. Αυτό την καθιστά ιδανική για εφαρμογές με πολύπλοκες λογικές προβολής και πλούσια διεπαφή χρήστη. Ωστόσο, η υιοθέτηση του MVVM μπορεί να απαιτήσει περισσότερο χρόνο και προσπάθεια στην αρχική ανάπτυξη.

Συνολικά, η Αρχιτεκτονική βασισμένη σε Στοιχεία είναι προτιμότερη για εφαρμογές με έμφαση στη διεπαφή χρήστη και την επαναχρησιμοποίηση στοιχείων. Η MVC είναι μια καλή επιλογή για εφαρμογές με έντονες αλληλεπιδράσεις με τον χρήστη. Η MVVM αποτελεί μια πιο σύγχρονη προσέγγιση και είναι προτιμότερο για εφαρμογές με συνδυασμό πλούσιας διεπαφής και πολύπλοκων λογικών παρουσίασης, παρέχοντας καλύτερη διαχωρισιμότητα και δυνατότητα δοκιμής. Η τελική επιλογή εξαρτάται από τις ανάγκες του έργου, τις δεξιότητες της ομάδας και τις προτεραιότητες όπως η ευελιξία, η επεκτασιμότητα και η ευκολία συντήρησης.

Vue Στο Vue.js, οι εφαρμογές δημιουργούνται συνθέτοντας επαναχρησιμοποιήσιμα και αυτόνομα στοιχεία. Ένα στοιχείο στο Vue.js εγκλωβίζει τόσο τη δομή HTML (πρότυπο), τη λογική JavaScript (script) και προαιρετικά τα στυλ CSS (styles), παρέχοντας μια συνεκτική μονάδα λειτουργικότητας. Το Vue.js αξιοποιεί την αρχιτεκτονική του βασισμένη σε στοιχεία για να προωθήσει τη διαμόρφωση, την επαναχρησιμοποίηση και τη διατήρηση του κώδικα. Τα στοιχεία μπορούν να συνδυαστούν ιεραρχικά, επιτρέποντας στους developers να δημιουργήσουν σύνθετες διασυνδέσεις χρήστη (UI) συναρμολογώντας μικρότερα, διαχειρίσιμα κομμάτια. Επιπλέον, το Vue.js παρέχει διάφορα χαρακτηριστικά

για να διευκολύνει την επικοινωνία μεταξύ των στοιχείων, όπως:

1. **props**: για τη μεταφορά δεδομένων από γονικά σε θυγατρικά στοιχεία
2. **custom events**: για την επικοινωνία από θυγατρικά σε γονικά στοιχεία
3. **event bus ή Vuex**: για τη διαχείριση της κατάστασης σε ολόκληρη την εφαρμογή.

Svelte

Το Svelte είναι επίσης βασισμένο σε στοιχεία, επιτρέποντας στους developers να δημιουργούν επαναχρησιμοποιήσιμα στοιχεία UI εγκλωβισμένα μέσα σε αυτά. Ωστόσο, σε αντίθεση με άλλα frameworks όπου τα στοιχεία ερμηνεύονται κατά τη διάρκεια της εκτέλεσης, τα στοιχεία του Svelte μεταγλωττίζονται σε εξαιρετικά αποδοτικό κώδικα που χειρίζεται άμεσα το DOM όταν χρειάζεται.

Αυτή η μοναδική προσέγγιση στην ανάπτυξη UI στο Svelte οδηγεί σε εξαιρετικά αποδοτικές και υψηλών επιδόσεων διαδικτυακές εφαρμογές, καθώς και σε ένα απλούστερο διανοητικό μοντέλο για τους developers. Τα στοιχεία στο Svelte εγκλωβίζουν όχι μόνο τη δομή και τη συμπεριφορά, αλλά και τον τρόπο με τον οποίο ενημερώνουν το DOM, με αποτέλεσμα έναν καθαρότερο και πιο προβλέψιμο κώδικα.

React

Η αρχιτεκτονική βασισμένη σε στοιχεία και το μοτίβο ροής δεδομένων μιας κατεύθυνσης του React το καθιστούν εξαιρετικά αρθρωτό, επαναχρησιμοποιήσιμο και διατηρήσιμο. Τα στοιχεία μπορούν εύκολα να συνδυαστούν και να φωλιάσουν, επιτρέποντας στους developers να δημιουργήσουν σύνθετα UI διασπώντας τα σε μικρότερα, διαχειρίσιμα κομμάτια. Το οικοσύστημα του React περιλαμβάνει επίσης εργαλεία όπως το React Router για τη διαχείριση της δρομολόγησης της εφαρμογής και το Redux για τη διαχείριση της κατάστασης της εφαρμογής, βελτιώνοντας περαιτέρω τις δυνατότητές του για τη δημιουργία κλιμακωτών και διατηρήσιμων διαδικτυακών εφαρμογών.

Angular

Στην Angular, η αρχιτεκτονική MVC υλοποιείται μέσω των βασικών εννοιών της, δηλαδή τα στοιχεία, τις υπηρεσίες και τα πρότυπα. Τα στοιχεία στην Angular λειτουργούν ως οι οικοδομικοί λίθοι του περιβάλλοντος χρήστη της εφαρμογής και εγκλωβίζουν τη λογική που σχετίζεται με ένα συγκεκριμένο τμήμα του περιβάλλοντος χρήστη. Κάθε στοιχείο αποτελείται από μια κλάση TypeScript που αντιπροσωπεύει τον ελεγκτή(λογική), ένα πρότυπο HTML που αντιπροσωπεύει την τοποθέτηση και εμφάνιση των στοιχείων και ένα αρχείο CSS για την μορφοποίηση. Η διαχείριση του μοντέλο δεδομένων ή της κατάστασης πραγματοποιείται μέσα στην αντίστοιχη κλάση του στοιχείου, λειτουργώντας ως

το μοντέλο στην αρχιτεκτονική MVC. Οι υπηρεσίες Angular χρησιμοποιούνται για την εγκλωβισμό κοινής επιχειρηματικής λογικής, τον χειρισμό δεδομένων ή την επικοινωνία με εξωτερικές υπηρεσίες, λειτουργώντας ως ελεγκτής στο μοτίβο MVC. Τα πρότυπα στο Angular αντιπροσωπεύουν την άποψη, παρέχοντας την σήμανση UI και τη σύνδεση με τα δεδομένα και τις μεθόδους του στοιχείου. Ο μηχανισμός έγχυσης εξαρτήσεων του Angular διευκολύνει την επικοινωνία μεταξύ διαφορετικών στοιχείων και υπηρεσιών, επιτρέποντας χαλαρή σύνδεση(coupling) και αρθρωτότητα(modularity).

Ember.js

Στο Ember.js, το MVVM υλοποιείται μέσω ενός συνδυασμού του δρομολογητή (router), των πρότυπων (views), των ελεγκτών (view-models) και των model του. Ο δρομολογητής διαχειρίζεται τις μεταβάσεις κατάστασης της εφαρμογής και την δρομολόγηση URL, ενώ τα πρότυπα ορίζουν τη δομή του περιβάλλοντος UI χρησιμοποιώντας σύνταξη Handlebars[20] με δυναμικές συνδέσεις δεδομένων στους ελεγκτές. Οι ελεγκτές λειτουργούν ως μοντέλα παρουσίασης, κρατώντας την κατάσταση και τη συμπεριφορά της εφαρμογής συγκεκριμένα για το UI, μεσολαβώντας μεταξύ των model και των View. Τα model αντιπροσωπεύουν τα υποκείμενα δεδομένα και τη λογική επιχειρηματικής λειτουργίας. Αυτή η αρχιτεκτονική επιτρέπει έναν σαφή διαχωρισμό των ευθυνών, επιτρέποντας στους developers να διαμορφώνουν αποτελεσματικά τον κώδικά τους, να βελτιώνουν τη διατήρησή του και να διευκολύνουν τη δοκιμή. Επιπλέον, η αμφίδρομη σύνδεση δεδομένων του Ember διασφαλίζει ότι οι αλλαγές στα View στοιχεία ενημερώνουν αυτόματα το model και το αντίστροφο, παρέχοντας μια ομαλή εμπειρία χρήστη.

Τα παραπάνω πλαίσια και οι αντίστοιχες αρχιτεκτονικές τους έχουν συλλεχθεί και παρουσιάζονται στον Πίνακα 4.2.

Framework	Architecture
Vue	Component-based
Svelte	Component-based (Compiler-based)
React	Component-based
Angular	MVC (Model-View-Controller)
Ember	MVVM (Model-View-View Mode)

Πίνακας 4.2: Κατηγοριοποίηση πλαισίων βάσει της αρχιτεκτονικής τους

4.1.2.2 Αρχιτεκτονικές στα Back-End πλαίσια

Τα πλαίσια Express, Fastify, Hono και Koa δεν χρησιμοποιούν κάποια συγκεκριμένη αρχιτεκτονική όπως την MVC ή MVVM. Αυτές οι αρχιτεκτονικές σχετίζονται περισσότερο με την ανάπτυξη εφαρμογών με γραφικό περιβάλλον χρήστη. Τα προαναφερθέντα πλαίσια είναι πλαίσια για τη δημιουργία διακομιστών (server-side frameworks) και

επικεντρώνονται στη διαχείριση αιτημάτων HTTP, τη δρομολόγηση URL, τη διαχείριση middleware και τη δημιουργία web APIs. Ακολουθούν μια αρχιτεκτονική βασισμένη σε αιτήματα/απαντήσεις (request/response) και ροή ελέγχου (control flow).

Πιο συγκεκριμένα:

1. **Express:** Είναι ένα ευέλικτο και ελαφρύ πλαίσιο που δεν επιβάλλει κάποια συγκεκριμένη αρχιτεκτονική. Παρέχει έναν απλό τρόπο δρομολόγησης και διαχείρισης middleware.

2. **Fastify:** Σχεδιάστηκε για υψηλές επιδόσεις και απλότητα. Χρησιμοποιεί μια αρχιτεκτονική plug-in που επιτρέπει την εύκολη επέκταση των δυνατοτήτων του.

3. **Hono:** Είναι ένα σχετικά νέο πλαίσιο που εστιάζει στην απλότητα, την αποδοτικότητα και την εύκολη ενσωμάτωση με άλλες βιβλιοθήκες.

4. **Koa:** Χτίστηκε πάνω στο Express αλλά με μια πιο μοντέρνα προσέγγιση χρησιμοποιώντας ES6 async/await. Βασίζεται επίσης σε μια αρχιτεκτονική middleware.

Αντί για MVC ή MVVM, αυτά τα πλαίσια ακολουθούν μια πιο απλή και άμεση προσέγγιση στη διαχείριση αιτημάτων και την κατασκευή APIs. Η οργάνωση της λογικής εφαρμογής και η διαχείριση του μοντέλου δεδομένων εναπόκειται στον προγραμματιστή.

4.1.2.3 Αρχιτεκτονικές στα Full-Stack πλαίσια

Τα πλαισία Next.js, Feathers και Meteor χρησιμοποιούν διαφορετικές αρχιτεκτονικές προτύπων, αλλά όλα τους έχουν στόχο την καλύτερη οργάνωση και διαχωρισμό των διαφόρων μερών μιας εφαρμογής.

Next.js

Το Next.js είναι ένα πλαίσιο React για τη δημιουργία εφαρμογών διαδικτύου στην πλευρά του server. Ακολουθεί μια προσέγγιση βασισμένη σε εξαρτήματα (component-based), η οποία είναι συνεπής με την φιλοσοφία του React. Δεν υιοθετεί άμεσα την αρχιτεκτονική MVC ή MVVM, αλλά επιτρέπει την οργάνωση του κώδικα με τρόπους που μοιάζουν με αυτές τις προσεγγίσεις.

Feathers

Το Feathers είναι ένα πλαίσιο για τη δημιουργία εφαρμογών real-time στην πλευρά του server και του client, χρησιμοποιώντας JavaScript ή TypeScript. Ακολουθεί μια αρχιτεκτονική βασισμένη σε υπηρεσίες (service-oriented architecture), όπου κάθε υπηρεσία είναι υπεύθυνη για έναν συγκεκριμένο πόρο ή λειτουργία της εφαρμογής. Αυτό το πρότυπο διαχωρίζει τη λογική από την παρουσίαση, αλλά δεν ακολουθεί ρητά την MVC ή MVVM.

Meteor

Το Meteor είναι ένα πλήρες πλαίσιο για τη δημιουργία εφαρμογών διαδικτύου σε JavaScript, τόσο στην πλευρά του server όσο και στην πλευρά του client. Ενώ δεν ακολουθεί αυστηρά την αρχιτεκτονική MVC ή MVVM, προωθεί έναν διαχωρισμό ανάμεσα στη λογική και την παρουσίαση μέσω της χρήσης συλλογών για τα δεδομένα και προτύπων για την απεικόνιση στο UI.

Συνοψίζοντας, ενώ κανένα από αυτά τα πλαίσια δεν υιοθετεί ρητά την MVC ή MVVM, όλα τους παρέχουν τρόπους για τον καλύτερο διαχωρισμό των διαφόρων μερών της εφαρμογής, ακολουθώντας αρχές όπως η αρχιτεκτονική βασισμένη σε εξαρτήματα ή υπηρεσίες.

4.1.3 Κοινότητα και Υποστήριξη

Η διαθεσιμότητα κοινότητας και υποστήριξης παίζει καθοριστικό ρόλο στην υιοθέτηση και τη μακροβιότητα των διαφορετικών πλαισίων. Αυτή η κατηγοριοποίηση ομαδοποιεί τα πλαίσια με βάση το μέγεθος, την ωριμότητα και το επίπεδο δραστηριότητας των κοινοτήτων τους.

Το μέγεθος της κοινότητας και η διαθέσιμη υποστήριξη για ένα JavaScript framework είναι σημαντικά για αρκετούς λόγους:

1. **Επίλυση Προβλημάτων και Βοήθεια:** Μια μεγαλύτερη και ενεργή κοινότητα σημαίνει ότι είναι πιο πιθανό να βρείτε λύσεις για τα προβλήματα που αντιμετωπίζετε, είτε σε φόρουμ, είτε σε ιστότοπους ερωτήσεων/απαντήσεων όπως το Stack Overflow. Με περισσότερους προγραμματιστές που χρησιμοποιούν το ίδιο πλαίσιο, αυξάνονται οι πιθανότητες να έχουν απαντηθεί παρόμοια ερωτήματα.

2. **Τεκμηρίωση και Πόροι Μάθησης:** Πλαίσια με μεγάλη κοινότητα τείνουν να έχουν καλύτερη τεκμηρίωση, περισσότερα βοηθητικά βιβλία, οδηγούς, βίντεο και άλλους πόρους μάθησης διαθέσιμους. Αυτό καθιστά ευκολότερη την εκμάθηση και την υιοθέτηση του πλαισίου.

3. **Συντήρηση και Ανάπτυξη:** Τα πλαίσια με μεγάλη κοινότητα έχουν συνήθως πιο ενεργή ανάπτυξη, με συχνές ενημερώσεις, διορθώσεις σφαλμάτων και νέες δυνατότητες. Μια μικρή κοινότητα μπορεί να οδηγήσει σε αργή ανάπτυξη ή εγκατάλειψη του έργου.

4. **Πρόσθετα και Βιβλιοθήκες:** Τα δημοφιλή πλαίσια τείνουν να έχουν ένα πλούσιο οικοσύστημα πρόσθετων, βιβλιοθηκών και εργαλείων που αναπτύσσονται από την κοινότητα, προσφέροντας επιπλέον λειτουργικότητα και επεκτασιμότητα.

5. **Απασχόληση και Ευκαιρίες Σταδιοδρομίας:** Τα πλαίσια με μεγάλη υιοθέτηση από εταιρείες και οργανισμούς μπορούν να προσφέρουν περισσότερες ευκαιρίες απασχόλησης και σταδιοδρομίας για προγραμματιστές με εμπειρία σε αυτά τα πλαίσια.

6. **Κοινότητα Υποστήριξης:** Μια ενεργή κοινότητα μπορεί να παρέχει χρήσιμες συμβουλές, βέλτιστες πρακτικές και συζητήσεις γύρω από το πλαίσιο, βοηθώντας τους προγραμματιστές να βελτιώσουν τις δεξιότητές τους και να μοιραστούν γνώσεις.

Συνολικά, ένα JavaScript framework με μεγάλη και ενεργή κοινότητα μπορεί να προσφέρει πολλά πλεονεκτήματα, όπως ευκολότερη επίλυση προβλημάτων, πρόσβαση σε πολλούς πόρους μάθησης, συνεχή ανάπτυξη και βελτίωση, καθώς και περισσότερες ευκαιρίες απασχόλησης και υποστήριξη από άλλους προγραμματιστές.

4.1.3.1 Κοινότητα και Υποστήριξη των Front-End πλαισίων

Μεγάλα και ενεργά πλαίσια όπως το Angular και το React διαθέτουν πολύ μεγάλες και ενεργές κοινότητες, με εκτενή τεκμηρίωση, αφιερωμένους διαύλους υποστήριξης και ένα τεράστιο οικοσύστημα πόρων. Το Vue επίσης απολαμβάνει μια μεγάλη και ενεργή κοινότητα, προσφέροντας ένα καλά ισορροπημένο οικοσύστημα επίσημης και υποστήριξης τρίτων. Το Svelte, αν και νεότερο, διαθέτει μια ταχέως αναπτυσσόμενη και ενεργή κοινότητα, παρέχοντας χρήσιμους πόρους και αφοσιωμένη ανάπτυξη. Το Ember, παρόλο που διαθέτει μια μικρότερη κοινότητα σε σύγκριση με τα άλλα, διατηρεί έναν ενεργό κύκλο ανάπτυξης και προσφέρει αφιερωμένους διαύλους υποστήριξης. Αυτή η κατηγοριοποίηση βοηθά τους προγραμματιστές να κατανοήσουν το επίπεδο υποστήριξης και των διαθέσιμων πόρων για κάθε πλαίσιο, λαμβάνοντας έτσι συνειδητές αποφάσεις με βάση τις ανάγκες και τις απαιτήσεις των projects τους. Στον Πίνακα 4.3 εμφανίζουμε την κατηγοριοποίηση των πλαισίων βάσει της κοινότητας και της υποστήριξης που παρέχουν.

Πλαίσιο	**Κοινότητα**	**Λεπτομέρειες**
Vue	Μεγάλη	Μεγάλη και ενεργή κοινότητα, που προσφέρει διάφορους πόρους και κανάλια υποστήριξης. Καλά ισορροπημένο οικοσύστημα με επίσημη τεκμηρίωση και βιβλιοθήκες τρίτων.
Svelte	Αναπτυσσόμενη	Σχετικά νέο πλαίσιο αλλά ταχέως αναπτυσσόμενη κοινότητα με ενεργή ανάπτυξη και χρήσιμους πόρους. Μικρότερη κοινότητα σε σύγκριση με άλλες.
React	Πολύ μεγάλη	Τεράστια κοινότητα και οικοσύστημα με άφθονους πόρους, βιβλιοθήκες και ενεργό ανάπτυξη. Οι βιβλιοθήκες τρίτων ενδέχεται να απαιτούν ανεξάρτητη υποστήριξη.
Angular	Μεγάλη	Εκτεταμένη τεκμηρίωση, οδηγοί χρήσης και τεράστια κοινότητα για υποστήριξη και συνεισφορά. Υποστηρίζεται από την Google, εξασφαλίζοντας μακροπρόθεσμη σταθερότητα.
Ember	Μέτρια	Αφιερωμένη κοινότητα με χρήσιμους πόρους και ενεργό ανάπτυξη. Μικρότερη κοινότητα σε σύγκριση με μεγαλύτερα πλαίσια, αλλά εξακολουθεί να είναι ενεργή.

Πίνακας 4.3: Κατηγοροιοποίηση των Front-End JavaScript πλαισίων βάσει της κοινότητας και υποστήριξης

4.1.3.2 Κοινότητα και Υποστήριξη Back-End Πλαισίων

Ο Πίνακας 4.4 κατηγοριοποιεί τα τέσσερα βασικά back-end πλαίσια της παρούσας διπλωματικής με βάση την ωριμότητα και το μέγεθος των κοινοτήτων τους: Το Express διαθέτει την πιο καλά εδραιωμένη κοινότητα, προσφέροντας εκτεταμένη τεκμηρίωση, οδηγούς και ένα τεράστιο αριθμό προγραμματιστών έτοιμο να προσφέρει βοήθεια. Τα Koa και Fastify αντιπροσωπεύουν πλαίσια με ισχυρές και αναπτυσσόμενες κοινότητες, παρέχοντας ενεργή ανάπτυξη και χρήσιμους πόρους. Τέλος, το Hono, ως αναδυόμενο πλαίσιο, διαθέτει μια αναπτυσσόμενη κοινότητα και οικοσύστημα υποστήριξης που αναπτύσσεται σταθερά. Αυτή η κατηγοριοποίηση βοηθά τους προγραμματιστές να κατανοήσουν το επίπεδο υποστήριξης που μπορούν να περιμένουν από τη βάση χρηστών κάθε πλαισίου.

Πλαίσιο	Κοινότητα	Λεπτομέρειες
Express	Τεράστια	Εκτεταμένη τεκμηρίωση, οδηγοί χρήσης και μια τεράστια κοινότητα για υποστήριξη και συνεισφορά
Koa	Ενεργή	Αναπτυσσόμενη κοινότητα, με ενεργή ανάπτυξη και χρήσιμους διαθέσιμους πόρους.
Fastify	Ενεργή	Ενεργή κοινότητα και αναπτυσσόμενη, καθώς και κανάλια υποστήριξης και αυξανόμενους πόρους
Hono	Αναπτυσσόμενη	Σχετικά νέο πλαίσιο με μια αναπτυσσόμενη κοινότητα και οικοσύστημα υποστήριξης.

Πίνακας 4.4: Κατηγοροιοποίηση των Back-End JavaScript πλαισίων σχετικά με την κοινότητα και υποστήριξη

4.1.3.3 Κοινότητα και Υποστήριξη Full-Stack Πλαισίων

Όσον αφορά την κοινότητα και υποστήριξη στα Full-Stack πλαίσια, το Next.js ξεχωρίζει με τη μεγάλη και ενεργή του κοινότητα, την άφθονη τεκμηρίωση, τους οδηγούς χρήσης και την ενεργή ανάπτυξη από την Vercel. Το Meteor διαθέτει επίσης μια ενεργή κοινότητα με πολλούς διαθέσιμους πόρους και υποστήριξη, αν και μικρότερη σε μέγεθος από το Next.js. Από την άλλη πλευρά, το Feathers έχει μια σχετικά μικρή αλλά ενεργή κοινότητα, με περιορισμένους διαθέσιμους πόρους και υποστήριξη. Ως εκ τούτου, η επιλογή ενός πλαισίου με μεγαλύτερη κοινότητα και καλύτερη υποστήριξη, όπως το Next.js ή το Meteor, μπορεί να διευκολύνει την επίλυση προβλημάτων, την εκμάθηση και την υιοθέτηση του πλαισίου

Ο Πίνακας 4.5 εμφανίζει την κατηγοροιοποίηση όσων αναφέρθηκαν παραπάνω.

Πλαίσιο	Κοινότητα	Λεπτομέρειες
Next.js	Μεγάλη	Έχει μια μεγάλη και ενεργή κοινότητα, με πολλούς διαθέσιμους πόρους μάθησης, οδηγούς και ενεργή ανάπτυξη από την Vercel.
Meteor	Ενεργή	Διαθέτει μια ενεργή κοινότητα με πολλούς διαθέσιμους πόρους και υποστήριξη, ωστόσο μικρότερη από άλλα δημοφιλή πλαίσια.
Feathers	Μικρή	Έχει μια σχετικά μικρή αλλά ενεργή κοινότητα, με περιορισμένους διαθέσιμους πόρους και υποστήριξη.

Πίνακας 4.5: Κατηγοριοποίηση των Full-Stack JavaScript πλαισίων σχετικά με την κοινότητα και υποστήριξη

4.1.4 Διαχείριση ασύγχρονων λειτουργιών: Callback vs. Async/Await

Όταν εργαζόμαστε με ασύγχρονες λειτουργίες στον προγραμματισμό, υπάρχουν δύο κύριοι τρόποι χειρισμού τους: με την χρήση callbacks και με την χρήση async/await εντολών. Ας τα αναλύσουμε για να κατανοήσουμε πώς εκτελείται ο κώδικάς σας σε κάθε σενάριο.

Χρήση Callback

Ας δώσουμε ένα πιο καθημερινό παράδειγμα για να κατανοήσουμε την χρήση των Callbacks. Ας φανταστούμε ότι παραγγέλνουμε φαγητό σε ένα εστιατόριο. Δίνουμε την παραγγελία μας στον σερβιτόρο ώστε να ετοιμαστεί. Αλλά αντί να περιμένουμε στον πάγκο μέχρι να ετοιμαστεί το φαγητό, δίνουμε εντολή στον σερβιτόρο να μας ειδοποιήσει μόλις το φαγητό μας ετοιμαστεί. Με έναν παρόμοιο τρόπο λειτουργεί και η χρήση μιας συνάρτησης callback. Η παραγγελία είναι ουσιαστικά η συνάρτηση που καλείται από κάποιο σημείο του κώδικα με σκοπό να ολοκληρώσει κάποια συγκεκριμένη λειτουργία, για παράδειγμα το κατέβασμα ενός αρχείου. Η συνάρτηση αυτή μπορεί να πάρει αρκετή ώρα μέχρι να ολοκληρωθεί, όπως στην περίπτωση που το αρχείο είναι μεγάλο. Για να μην καθυστερεί ο υπόλοιπος κώδικας, συνεχίζει κανονικά την εκτέλεσή του. Η callback συνάρτηση είναι αυτή που θα εκτελεστεί όταν ολοκληρωθεί το κατέβασμα του αρχείου, και μπορεί να περιέχει ένα άλλο σύνολο εντολών, για παράδειγμα την εμφάνιση ενός μηνύματος ή το άνοιγμα του αρχείου.

Παρακάτω παρουσιάζουμε την τεχνική σειρά:

1. Καλούμε μια συνάρτηση που εκτελεί μια ασύγχρονη λειτουργία
2. Παρέχουμε ως είσοδο της συνάρτησης μια δεύτερη συνάρτηση (την callback) η οποία θα εκτελεστεί αργότερα με το τελικό αποτέλεσμα (ή κάποιο λάθος) μόλις η λειτουργία ολοκληρωθεί.
3. Ο βασικός κώδικας συνεχίζει να εκτελείται χωρίς αναμονή των παραπάνω

Χρήση Async/Await εντολών

Η χρήση των συγκεκριμένων εντολών έρχεται σε πλήρη αντιπαράθεση με την χρήση των συναρτήσεων callback. Ως παράδειγμα μπορούμε να φανταστούμε την αναμονή του χρήστη σε μια ουρά σε ένα κατάστημα. Ο χρήστης ζητάει από τον ταμία τι θέλει να αγοράσει και αυτή τη φορά, αντί να ενημερωθεί όταν η παραγγελία του ετοιμαστεί, θα πρέπει να περιμένει στην ουρά, χωρίς να μπορεί να εκτελέσει άλλες ενέργειες. Έτσι λοιπόν η εντολή await ουσιαστικά διακόπτει προσωρινά τον κώδικα στο σημείο που καλείται η συνάρτηση που θέλουμε και ο κώδικας θα συνεχιστεί μόλις αυτή ολοκληρωθεί. Σε αυτήν την περίπτωση η συνάρτηση που καλείται πρέπει να δηλωθεί ως async.

Παρακάτω παρουσιάζουμε την τεχνική σειρά:

1. Δηλώνουμε μια συνάρτηση ως async για να υποδείξουμε ότι εμπεριέχει ασύγχρονες λειτουργίες.
2. Χρησιμοποιούμε την await εντολή πριν την κλήση της ασύγχρονης συνάρτησης.
3. Ο βασικός κώδικας διακόπτεται προσωρινά με την χρήση της await εντολής μέχρι να ολοκληρωθεί η async συνάρτηση που καλέστηκε.

Σύγκριση Async/Await εντολών

Όσον αφορά τη χρήση callbacks και async/await στην JavaScript, η μέθοδος async/await θεωρείται γενικά καλύτερη και προτιμότερη για τους ακόλουθους λόγους:

1. **Ευκολότερη ανάγνωση και κατανόηση του κώδικα:** Ο κώδικας που χρησιμοποιεί async/await είναι πιο γραμμικός και ακολουθεί μια δομή παρόμοια με τον συγχρονισμένο κώδικα, καθιστώντας τον πιο εύκολο στην ανάγνωση και την κατανόηση.
2. **Διαχείριση λαθών:** Με τα async/await, μπορείτε να χρησιμοποιήσετε την κλασική δομή try/catch για τη διαχείριση λαθών, η οποία είναι πιο οικεία και ευκολότερη στη χρήση από τη διαχείριση λαθών με callbacks.
3. **Αποφυγή callback hell:** Όταν υπάρχουν πολλαπλά ασύγχρονα βήματα, η χρήση callbacks μπορεί να οδηγήσει στο λεγόμενο "callback hell", όπου ο κώδικας γίνεται δυσανάγνωστος και δύσκολος στη συντήρηση λόγω των πολλαπλών ενσωματωμένων callbacks. Τα async/await βοηθούν στην αποφυγή αυτού του προβλήματος.
4. **Βελτιωμένη διαχείριση ροής ελέγχου:** Με τα async/await, μπορείτε να χρησιμοποιήσετε συνθήκες και βρόχους για να ελέγξετε τη ροή εκτέλεσης του κώδικα σας, κάτι που είναι δύσκολο με τα callbacks.
5. **Διαχείριση σφαλμάτων με τη χρήση try/catch:** Η χρήση async/await επιτρέπει τη χρήση της δήλωσης try/catch για τη διαχείριση σφαλμάτων, κάτι που είναι πιο ευανάγνωστο και ευκολότερο στη διαχείριση από την επεξεργασία σφαλμάτων με callbacks.
6. **Απλούστερη συντήρηση:** Ο κώδικας που χρησιμοποιεί async/await είναι συνήθως πιο εύκολος στη συντήρηση, καθώς είναι πιο δομημένος και ευκολότερος στην ανάγνωση.

Ωστόσο, τα callbacks εξακολουθούν να είναι χρήσιμα σε ορισμένες περιπτώσεις, όπως όταν χρειάζεται να διαχειριστείτε ασύγχρονα γεγονότα σε πραγματικό χρόνο ή όταν εργάζεστε με παλαιότερες εκδόσεις του JavaScript που δεν υποστηρίζουν τα async/await.

Συνολικά, η προτίμηση για τη χρήση async/await έναντι των callbacks είναι ευρέως αποδεκτή στην κοινότητα των προγραμματιστών JavaScript, καθώς βελτιώνει την ευκολία ανάγνωσης, τη διαχείριση λαθών και τη συντήρηση του κώδικα.

4.1.4.1 Διαχείριση Ασύγχρονων Λειτουργιών από τα Front-End πλαίσια

Express:

Ενώ το Express δεν επιβάλλει συγκεκριμένο στυλ, μπορεί να χρησιμοποιηθεί άνετα τόσο με προσέγγιση callback όσο και με async/await. Η φιλοσοφία ανάπτυξής του αποκλίνει προς τη σύμβαση έναντι της διαμόρφωσης, προσφέροντας μια οικεία δομή για τη δημιουργία web εφαρμογών. Πολλοί προγραμματιστές θεωρούν το Express ένα καλό σημείο εκκίνησης λόγω των εκτεταμένων πόρων και της μεγάλης κοινότητάς του.

Fastify:

Σχεδιασμένο για ταχύτητα και ευελιξία, το Fastify υιοθετεί async/await από προεπιλογή. Αυτό οδηγεί σε πιο καθαρό και πιο ευανάγνωστο κώδικα όταν χειρίζεται ασυγχρονες λειτουργίες. Εμπνέεται από το Express όσον αφορά την προσφορά ενός ισχυρού συστήματος plugin για την επέκταση των λειτουργιών, ενώ εστιάζει στη βελτιστοποίηση των επιδόσεων για απαιτητικές εφαρμογές.

Koa:

Συχνά θεωρείται ως διάδοχος του Express, το Koa δίνει έμφαση στην ευελιξία και μια πιο σύγχρονη προσέγγιση. Υποστηρίζει άμεσα τόσο στυλ callback όσο και async/await, δίνοντας στους προγραμματιστές έλεγχο πάνω στην προτιμώμενη μέθοδό τους. Το Koa αντλεί έμπνευση από τις βασικές έννοιες του Express, αλλά αποσκοπεί σε μια πιο ελαφριά και μινιμαλιστική βάση για τη δημιουργία web εφαρμογών και API.

Hono:

Κατασκευασμένο ειδικά για τους χρόνους εκτέλεσης Deno και Bun, το Hono δίνει προτεραιότητα στις επιδόσεις και την απλότητα. Χρησιμοποιεί async/await από προεπιλογή, παρόμοια με το Fastify. Ενώ το Hono προσφέρει ενσωματωμένες δυνατότητες δρομολόγησης και middleware, διατηρεί τη βασική του λειτουργικότητα συμπαγή, επιτρέποντας στους προγραμματιστές να επιλέγουν πρόσθετα χαρακτηριστικά ανάλογα με τις ανάγκες. Αυτή η εστίαση στον μινιμαλισμό καθιστά το Hono μια καλή επιλογή για τη δημιουργία αποδοτικών και απλοποιημένων web εφαρμογών.

Στον Πίνακα 4.6 παρουσιάζονται τα Front-End πλαίσια και ο τρόπος με τον οποίο διαχειρίζονται τις ασύγχρονες λειτουργίες.

Πλαίσιο	Στυλ Ανάπτυξης	Λεπτομέρειες
Express	Βασισμένη στα Callback (μπορεί να χρησιμοποιήσει και async/await)	Προσφέρει ευελιξία στη χρήση είτε στυλ callback (παραδοσιακό) είτε async/await (πιο σύγχρονο) για τον χειρισμό ασυγχρονων λειτουργιών
Fastify	Async/await (εξ ορισμού)	Δίνει προτεραιότητα στο async/await για μια πιο καθαρή και ευανάγνωστη προσέγγιση στις ασυγχρονες λειτουργίες
Koa	Ταυτόχρονη υποστήριξη και callback και async/await	Επιτρέπει στους προγραμματιστές να επιλέξουν ευέλικτα τον προτιμώμενο τους τρόπο για τον χειρισμό ασυγχρονων λειτουργιών.
Hono	Async/await (εξ ορισμού)	Χρησιμοποιεί async/await εξ αρχής για μια απλοποιημένη και ευανάγνωστη εμπειρία ανάπτυξης.

Πίνακας 4.6: Κατηγοροιοποίηση των Back-End JavaScript πλαισίων σχετικά με τη διαχείριση ασύγχρονων λειτουργιών.

4.1.4.2 Διαχείριση Ασύγχρονων Λειτουργιών από τα Front-End πλαίσια

Angular:

Η Angular χρησιμοποιεί async/await εκτενώς για τη διαχείριση ασύγχρονων λειτουργιών, όπως κλήσεις HTTP και αλληλεπιδράσεις με βάσεις δεδομένων. Επίσης, υποστηρίζει την έννοια των Observables από την βιβλιοθήκη RxJS, η οποία μπορεί να συνδυαστεί με async/await.

React:

Το React από μόνο του δεν έχει ενσωματωμένη υποστήριξη για async/await, αλλά οι προγραμματιστές μπορούν να τα χρησιμοποιήσουν σε συνδυασμό με συναρτήσεις άγκιστρου (hook functions) όπως η useEffect και η useCallback. Επίσης, βιβλιοθήκες όπως η React Query και η SWR διευκολύνουν τη χρήση async/await στο React.

Vue:

Το Vue.js υποστηρίζει πλήρως τη χρήση async/await σε συνδυασμό με τις επιλογές ζωής (lifecycle options) των στοιχείων και τις αναμεταδιδόμενες ιδιότητες (computed properties).

Ember:

Το Ember.js προωθεί τη χρήση async/await για τη διαχείριση ασύγχρονων λειτουργιών, παρέχοντας επίσης υποστήριξη για Promises και Observables.

Svelte:

Το Svelte υποστηρίζει τη χρήση async/await σε διάφορα σημεία, όπως σε συναρτήσεις ζωής (lifecycle functions) και σε αναδραστικές δηλώσεις (reactive statements).

Στον Πίνακα 4.7 παρουσιάζονται τα Front-End πλαίσια και ο τρόπος με τον οποίο διαχειρίζονται τις ασύγχρονες λειτουργίες.

Πλαίσιο	**Στυλ Ανάπτυξης**	**Λεπτομέρειες**
Angular	Async/await (εξ ορισμού)	Το Angular χρησιμοποιεί εκτενώς async/await για τη διαχείριση ασύγχρονων λειτουργιών, όπως κλήσεις HTTP και αλληλεπιδράσεις με βάσεις δεδομένων. Επίσης, υποστηρίζει τη χρήση Observables από τη βιβλιοθήκη RxJS σε συνδυασμό με async/await.
React	Async/await (εξ ορισμού)	Το React από μόνο του δεν έχει ενσωματωμένη υποστήριξη για async/await, αλλά οι προγραμματιστές μπορούν να τα χρησιμοποιήσουν σε συνδυασμό με συναρτήσεις άγκιστρου όπως η useEffect και η useCallback. Βιβλιοθήκες όπως η React Query και η SWR διευκολύνουν τη χρήση async/await στο React.
Vue.js	Async/await (εξ ορισμού)	Το Vue.js υποστηρίζει πλήρως τη χρήση async/await σε συνδυασμό με τις επιλογές ζωής των στοιχείων και τις αναμεταδιδόμενες ιδιότητες (computed properties).
Ember.js	Async/await (εξ ορισμού)	Το Ember.js προωθεί τη χρήση async/await για τη διαχείριση ασύγχρονων λειτουργιών, παρέχοντας επίσης υποστήριξη για Promises και Observables.
Svelte	Async/await (εξ ορισμού)	Το Svelte υποστηρίζει τη χρήση async/await σε διάφορα σημεία, όπως σε συναρτήσεις ζωής και σε αναδραστικές δηλώσεις.

Πίνακας 4.7: Κατηγοροιοποίηση των Front-End JavaScript πλαισίων σχετικά με τη διαχείριση ασύγχρονων λειτουργιών.

4.1.4.3 Διαχείριση Ασύγχρονων Λειτουργιών από τα Full-Stack πλαίσια

Τα full-stack πλαίσια Feathers, Next.js και Meteor υποστηρίζουν και χρησιμοποιούν κυρίως async/await για τη διαχείριση ασύγχρονων λειτουργιών, αντί για callbacks.

Feathers:

Το Feathers.js είναι ένα πλαίσιο για τη δημιουργία εφαρμογών πραγματικού χρόνου και REST APIs. Υποστηρίζει πλήρως τη χρήση async/await, τόσο στην πλευρά του διακο-

μιστή (server-side) όσο και στην πλευρά του πελάτη (client-side). Παρέχει ένα σύγχρονο και ευανάγνωστο τρόπο για τη διαχείριση ασύγχρονων λειτουργιών, όπως κλήσεις βάσης δεδομένων και αιτήματα HTTP.

Next.js:

Το Next.js είναι ένα πλαίσιο για τη δημιουργία εφαρμογών React με τη μέθοδο του στατικού δικτύου (static site generation) και της αποδοτικής αποτύπωσης από την πλευρά του διακομιστή (server-side rendering). Υποστηρίζει πλήρως τη χρήση async/await, τόσο στην πλευρά του διακομιστή όσο και στην πλευρά του πελάτη. Συγκεκριμένα, το Next.js χρησιμοποιεί async/await για τη διαχείριση ασύγχρονων λειτουργιών όπως η προφόρτωση δεδομένων (data fetching) και η διακομιστική αποκωδικοποίηση.

Meteor:

Meteor: Το Meteor είναι ένα full-stack πλαίσιο για τη δημιουργία εφαρμογών πραγματικού χρόνου σε JavaScript. Υποστηρίζει τη χρήση async/await τόσο στην πλευρά του διακομιστή (με τη βιβλιοθήκη Fibers) όσο και στην πλευρά του πελάτη. Οι προγραμματιστές μπορούν να χρησιμοποιήσουν async/await για τη διαχείριση ασύγχρονων λειτουργιών, όπως κλήσεις σε βάσεις δεδομένων, αιτήματα HTTP και αλληλεπιδράσεις με APIs.

Συνολικά, τα full-stack πλαίσια Feathers, Next.js και Meteor έχουν υιοθετήσει την προσέγγιση async/await για τη διαχείριση ασύγχρονων λειτουργιών, καθώς προσφέρει ένα πιο ευανάγνωστο και ευκολότερο στη συντήρηση τρόπο διαχείρισης σε σύγκριση με τα callbacks. Αυτό ευθυγραμμίζεται με τις γενικές τάσεις και βέλτιστες πρακτικές στην ανάπτυξη σύγχρονων εφαρμογών JavaScript.

Στον Πίνακα 4.8 παρουσιάζονται τα Full-Stack πλαίσια και ο τρόπος με τον οποίο διαχειρίζονται τις ασύγχρονες λειτουργίες.

4.2 Σύγκριση Πλαισίων

4.2.1 Κριτήρια Σύγκρισης

4.2.1.1 Καμπύλη Εκμάθησης

Η καμπύλη εκμάθησης για τα πλαίσια JavaScript μπορεί να ποικίλλει σημαντικά ανάλογα με παράγοντες όπως η πολυπλοκότητα του πλαισίου, η υπάρχουσα γνώση και εμπειρία του προγραμματιστή με τη JavaScript, η εξοικείωσή του με παρόμοια πλαίσια και οι διαθέσιμοι πόροι για εκμάθηση. Ακολουθεί μια γενική επισκόπηση:

Βασικές γνώσεις JavaScript: Προτού γίνει η μετάβαση σε οποιοδήποτε πλαίσιο JavaScript, είναι απαραίτητο να υπάρχει μια σταθερή κατανόηση των βασικών εννοιών

Πλαίσιο	Στυλ Ανάπτυξης	Λεπτομέρειες
Feathers.js	Async/await (εξ ορισμού)	Το Feathers.js υποστηρίζει πλήρως τη χρήση async/await, τόσο στην πλευρά του διακομιστή όσο και στην πλευρά του πελάτη, για τη διαχείριση ασύγχρονων λειτουργιών, όπως κλήσεις βάσης δεδομένων και αιτήματα HTTP, προσφέροντας ένα σύγχρονο και ευανάγνωστο τρόπο ανάπτυξης.
Next.js	Async/await (εξ ορισμού)	Το Next.js, ένα πλαίσιο για εφαρμογές React, χρησιμοποιεί async/await για τη διαχείριση ασύγχρονων λειτουργιών, όπως η προ-φόρτωση δεδομένων και η διακομιστική αποκωδικοποίηση, τόσο στην πλευρά του διακομιστή όσο και στην πλευρά του πελάτη.
Meteor	Async/await (εξ ορισμού)	Το Meteor, ένα full-stack πλαίσιο για εφαρμογές πραγματικού χρόνου, υποστηρίζει τη χρήση async/await τόσο στην πλευρά του διακομιστή (με τη βιβλιοθήκη Fibers) όσο και στην πλευρά του πελάτη, επιτρέποντας στους προγραμματιστές να διαχειρίζονται ασύγχρονες λειτουργίες, όπως κλήσεις σε βάσεις δεδομένων και αιτήματα HTTP, με ένα ευανάγνωστο τρόπο.

Πίνακας 4.8: Κατηγοροιοποίηση των Full-Stack JavaScript πλαισίων σχετικά με τη διαχείριση ασύγχρονων λειτουργιών.

JavaScript, όπως μεταβλητές, συναρτήσεις, αντικείμενα, πίνακες, βρόχους και δηλώσεις υπό όρους.

Βασικά πλαίσια: Αρχικά, θα πρέπει να υπάρχει γνώση των βασικών αρχών του πλαισίου, συμπεριλαμβανομένης της σύνταξης, των εννοιών και των βασικών χαρακτηριστικών του. Αυτό συνήθως περιλαμβάνει την κατανόηση του τρόπου ρύθμισης ενός έργου, τη δημιουργία στοιχείων ή λειτουργικών μονάδων, τη διαχείριση κατάστασης, τη διαχείρισης δρομολόγησης (αν υπάρχει) και αλληλεπίδρασης με το οικοσύστημα του πλαισίου.

Τεκμηρίωση και σεμινάρια: Τα πιο δημοφιλή πλαίσια διαθέτουν εκτενή τεκμηρίωση και πλήθος διαδικτυακών οδηγών εκμάθησης και διαθέσιμων μαθημάτων. Η αφιέρωση χρόνου στη μελέτη της επίσημης τεκμηρίωσης και στην επεξεργασία σεμιναρίων είναι ζωτικής σημασίας για την απόκτηση μιας σταθερής κατανόησης των δυνατοτήτων και των βέλτιστων πρακτικών του πλαισίου.

Πρακτική και έργα: Η πρακτική εξάσκηση είναι απαραίτητη για την εκμάθηση οποιουδήποτε πλαισίου. Η δημιουργία μικρών έργων ή η συμβολή σε έργα ανοιχτού κώδικα μπορεί να βοηθήσει στην ενίσχυση της εκμάθησης από τον προγραμματιστή και να εμβαθύνει την κατανόησή του για τις έννοιες και τα πρότυπα του πλαισίου.

Προηγμένα θέματα: Καθώς η ικανότητα προς τα βασικά αυξάνεται, ακολουθεί η εκ-

μάθηση των πιο προηγμένων θέματων όπως η βελτιστοποίηση απόδοσης, οι δοκιμές, η απόδοση από την πλευρά του διακομιστή, τα μοτίβα διαχείρισης κατάστασης (π.χ. Redux στην React) και ενοποίηση με άλλες τεχνολογίες (π.χ. , API, βάσεις δεδομένων, κλπ.).

Στην τρέχουσα ενότητα θα συγκρίνουμε τα πλαίσια ως προς την καμπύλη εκμάθησης του κάθε πλαισίου, λαμβάνοντας υπόψην τις παραπάνω παραμέτρους. Θα χρησιμοποιήσουμε μια κλίμακα από το 1 εώς το 5, με το 1 να αποδίδεται στην περίπτωση που το πλαίσιο έχει πολύ ήπια και μικρή καμπύλη εκμάθησης, ενώ το 5 στην περίπτωση που το πλαίσιο χρειάζεται αρκετό χρόνο για την πλήρη εκμάθησής του παρέχοντας μεγάλη και απότομη καμπύλη.

4.2.1.2 Δημοφιλία Πλαισίων

Ενώ η δημοτικότητα μπορεί να φαίνεται ελκυστικός μοναδικός παράγοντας για την επιλογή ενός Javascript πλαισίου, είναι σημαντικό να λάβουμε υπόψη τις λεπτομέρειες του. Από την θετική πλευρά, τα δημοφιλή πλαίσια διαθέτουν συχνά μια ζωντανή κοινότητα, που προσφέρει πλούσιους πόρους όπως τεκμηρίωση, tutorials και βιβλιοθήκες. Αυτό μεταφράζεται σε ευκολότερες καμπύλες μάθησης και εύκολα διαθέσιμες λύσεις για τους developers.

Ωστόσο, η δημοτικότητα δεν εγγυάται την τέλεια εφαρμογή. Τα πολύ δημοφιλή πλαίσια μπορεί να είναι υπερβολικά για απλούστερα έργα, απαιτώντας περιττή πολυπλοκότητα. Αντιστρόφως, τα πιο απλά πλαίσια που δεν παρέχουν υπερβολικές δυνατότητες, ενώ μπορεί να είναι ιδανικά για συγκεκριμένες ανάγκες, ενδέχεται να στερούνται των εκτεταμένων πόρων ή της υποστήριξης της κοινότητας που είναι κρίσιμα για πολύπλοκες εφαρμογές. Επομένως, η προσεκτική αξιολόγηση των μοναδικών απαιτήσεων του αντίστοιχου έργου προς υλοποίηση είναι απαραίτητη πριν βασιστούμε αποκλειστικά στη δημοτικότητα ενός πλαισίου κατά τη λήψη επιλογής του.

Στην τρέχουσα διπλωματική για να συγκρίνουμε τα πλαίσια βάσει της δημοφιλίας τους, χρησιμοποιήσαμε την ιστοσελίδα npmtrends.com [21], η οποία εμφανίζει για ένα επιλεγμένο χρονικό διάστημα την δημοφιλία μεταξύ πλαισίων που θα επιλέξουμε.

4.2.2 Σύγκριση Βάσει Καμπύλης Εκμάθησης Πλαισίων

4.2.2.1 Καμπύλη Εκμάθησης Front-End πλαισίων

Angular:

Η Angular διαθέτει ολοκληρωμένα χαρακτηριστικά, αλλά απαιτεί πιο απότομη καμπύλη μάθησης. Η εξάρτησή του από το TypeScript, ένα υπερσύνολο της JavaScript, προσθέτει ένα επιπλέον επίπεδο πολυπλοκότητας. Η εξοικείωση με τον αντικειμενοστρεφή προγραμματισμό και τα πρότυπα σχεδίασης είναι επίσης επωφελής. Ωστόσο, η δομημένη

προσέγγιση του Angular προσφέρει ισχυρή τεκμηρίωση, μεγάλη κοινότητα και εκτεταμένους πόρους, καθιστώντας πιο εύκολη την υπερπήδηση των αρχικών εμποδίων.

React:

Σε σύγκριση με την Angular, το React παρουσιάζει μια πιο ήπια καμπύλη μάθησης. Οι βασικές του έννοιες, όπως τα στοιχεία και η σύνταξη JSX, είναι σχετικά πιο απλές στην κατανόηση. Ωστόσο, η δημιουργία πολύπλοκων εφαρμογών συχνά απαιτεί την ενσωμάτωση πρόσθετων βιβλιοθηκών και εργαλείων, τα οποία μπορούν να αυξήσουν το βάρος της μάθησης. Συνιστάται προηγούμενη εμπειρία με JavaScript και κατανόηση του χειρισμού του εικονικού DOM για ομαλότερη πρόοδο.

Vue:

Το Vue.js θεωρείται ευρέως το πιο φιλικό προς τους αρχάριους πλαίσιο. Η βασική του σύνταξη μοιάζει πολύ με την HTML και την JavaScript, καθιστώντας την πιο εύκολη στην εκμάθηση για τους νεοεισερχόμενους. Επιπλέον, το Vue προσφέρει μια ευέλικτη προσέγγιση, επιτρέποντάς στους προγραμματιστές να το χρησιμοποιήσουν προοδευτικά ή ως πλήρες πλαίσιο. Αυτή η ευελιξία το καθιστά κατάλληλο για προγραμματιστές διαφορετικών επιπέδων δεξιοτήτων.

Ember

Το Ember.js ακολουθεί μια προσέγγιση με γνώμες, παρέχοντας μια ισχυρή δομή εξ αρχής. Αυτή η δομή μπορεί να είναι επωφελής για γρήγορη ανάπτυξη, αλλά με το κόστος μιας πιο απότομης καμπύλης μάθησης. Η κατανόηση των συμβάσεων και των εννοιών του Ember, όπως το Ember Data και το Ember CLI, είναι απαραίτητη για την αποτελεσματική ανάπτυξη. Η προηγούμενη εμπειρία με άλλα frontend frameworks μπορεί να διευκολύνει τη διαδικασία μάθησης.

Svelte

Το Svelte παρουσιάζει μια μοναδική προσέγγιση, μεταγλωττίζοντας τον κώδικα κατά τη διάρκεια της δημιουργίας, με αποτέλεσμα μικρότερα μεγέθη πακέτων και πιθανώς ταχύτερες εφαρμογές. Ενώ οι βασικές του έννοιες θεωρούνται σχετικά εύκολες στην εκμάθηση, η σύνταξη και ο ασύγχρονος χαρακτήρας του μπορεί να είναι άγνωστα σε προγραμματιστές που έχουν συνηθίσει σε παραδοσιακά frameworks. Επιπλέον, η κοινότητα και οι πόροι του εξακολουθούν να αναπτύσσονται σε σύγκριση με πιο καθιερωμένα πλαίσια.

Στον Πίνακα 4.10 εμφανίζεται μια συνοπτική σύγκριση των front-end πλαισίων της JavaScript [22].

Πλαίσιο	**Βαθμός**	**Καμπύλη εκμάθησης**
Angular	5	Μεγάλη και απότομη καμπύλη εκμάθησης λόγω του ολοκληρωμένου συνόλου χαρακτηριστικών και της πολύπλοκης αρχιτεκτονικής της. Απαιτεί κατανόηση λειτουργικών μονάδων, στοιχείων, υπηρεσιών, έγχυσης εξάρτησης (dependency injection) και RxJS για αντιδραστικό προγραμματισμό.
React	2	Σχετικά ήπια καμπύλη εκμάθησης, ειδικά για προγραμματιστές που είναι εξοικειωμένοι με JavaScript και HTML. Οι βασικές έννοιες όπως τα στοιχεία, η σύνταξη JSX και το εικονικό DOM είναι εύκολο να κατανοηθούν. Η διαχείριση κατάστασης με τις συναρτήσεις άγκιστρου της React και τις μεθόδους κύκλου ζωής εξαρτημάτων ενδέχεται να απαιτεί πρόσθετη εκμάθηση, αλλά είναι γενικά διαχειρίσιμη.
Ember	4	Μέτρια καμπύλη μάθησης, που εντάσεται μεταξύ Angular και React. Η προσέγγιση convention-over-configuration του Ember μπορεί να απλοποιήσει την ανάπτυξη όταν αυτή γίνει κατανοητή, αλλά στα αρχικά στάδια ίσως να φαίνεται περιοριστική. Η μάθηση περιλαμβάνει την κατανόηση διαδρομών, ελεγκτών, προτύπων και δεδομένων του Ember για διαχείριση δεδομένων. Οι ισχυρές απόψεις και τα ενσωματωμένα εργαλεία απλοποιούν την ανάπτυξη, αλλά ενδέχεται να απαιτούν προσαρμογή για προγραμματιστές που προέρχονται από άλλα πλαίσια.
Svelte	1	Μία από τις απλούστερες καμπύλες εκμάθησης μεταξύ των σύγχρονων πλαισίων JavaScript. Η προσέγγιση της Svelteγια τη μεταγλώττιση στοιχείων κατά το χρόνο κατασκευής και την εστίαση στη vanilla JavaScript το καθιστά διαισθητικό για προγραμματιστές που είναι εξοικειωμένοι με τις βασικές αρχές της ανάπτυξης ιστού. Η μάθηση περιλαμβάνει την κατανόηση των αντιδραστικών δηλώσεων, της δομής των στοιχείων και της σύνταξης που σχετίζεται με το Svelte, όπως τα stores και τα actions. Η συνοπτική τεκμηρίωση και τα διαδραστικά σεμινάρια διευκολύνουν την ταχεία μάθηση και τον πειραματισμό.
Vue	2	Σχετικά ήπια καμπύλη εκμάθησης, παρόμοια με το React αλλά με πρόσθετο syntactic sugar και ενσωματωμένα χαρακτηριστικά. Η προοδευτική υιοθέτηση του Vue επιτρέπει στους προγραμματιστές να το ενσωματώνουν σταδιακά σε υπάρχοντα έργα, καθιστώντας το προσβάσιμο σε προγραμματιστές διαφορετικών επιπέδων δεξιοτήτων. Η μάθηση περιλαμβάνει την κατανόηση των στοιχείων Vue, των οδηγιών, του Vue Router για τη δρομολόγηση και του Vuex για τη διαχείριση κατάστασης. Η καλά δομημένη τεκμηρίωση, η υποστηρικτική κοινότητα και το εκτεταμένο οικοσύστημα συμβάλλουν στην ευκολία εκμάθησής του.

Πίνακας 4.9: Σύγκριση πλαισίων front-end ως προς την καμπύλη εκμάθησης

4.2.2.2 Καμπύλη Εκμάθησης Back-End πλαισίων

Express:

Το Express θεωρείται ευρέως ως ένα από τα πιο φιλικά προς τους αρχάριους frameworks Node.js. Ο μινιμαλιστικός του σχεδιασμός και η εκτεταμένη τεκμηρίωση το κάνουν σχετικά εύκολο για τους developers να κατανοήσουν γρήγορα τις βασικές έννοιες. Η απλότητα του συστήματος middleware του Express επιτρέπει στους developers να κατανοήσουν και να εφαρμόσουν εύκολα τον δρομολογήση, το middleware και άλλα χαρακτηριστικά. Με μια μεγάλη κοινότητα και άφθονους πόρους, η εύρεση λύσεων σε κοινά προβλήματα είναι απλή, διευκολύνοντας περαιτέρω τη διαδικασία μάθησης.

Fastify:

Το Fastify είναι γνωστό για την επίδοσή του και το χαμηλό του λειτουργικό κόστος. Ενώ η καμπύλη μάθησης του μπορεί να είναι ελαφρώς πιο απότομη σε σύγκριση με το Express λόγω της εστίασής του στη βελτιστοποίηση των επιδόσεων και σε ορισμένα προηγμένα χαρακτηριστικά, το Fastify προσφέρει ολοκληρωμένη τεκμηρίωση και μια υποστηρικτική κοινότητα. Οι developers που είναι εξοικειωμένοι με το Express θα βρουν τη μετάβαση στο Fastify σχετικά ομαλή, καθώς πολλές έννοιες είναι παρόμοιες. Ωστόσο, η κατανόηση της έμφασης του Fastify στη βελτιστοποίηση των επιδόσεων, τα plugins και τους διακοσμητές μπορεί να απαιτήσει επιπλέον προσπάθεια.

Hapi:

Το Hapi δίνει προτεραιότητα στη διαμόρφωση και τις συμβάσεις, προσφέροντας ένα ισχυρό framework για τη δημιουργία σύνθετων εφαρμογών. Η καμπύλη μάθησης του μπορεί να είναι μέτρια, ειδικά για αρχάριους, λόγω της αυστηρής δομής του και του ολοκληρωμένου συνόλου χαρακτηριστικών. Η εκτεταμένη τεκμηρίωση του Hapi και η ισχυρή έμφαση στις βέλτιστες πρακτικές μπορούν να βοηθήσουν τους developers να μάθουν τις συμβάσεις του αποτελεσματικά. Ωστόσο, ο πλήρης χειρισμός των λεπτομερειών του Hapi, όπως η διαμόρφωση διαδρομών και η διαχείριση των plugin, μπορεί να απαιτήσει περισσότερο χρόνο σε σύγκριση με απλούστερα frameworks όπως το Express.

Koa:

Το Koa είναι γνωστό για τον ελαφρύ σχεδιασμό του και τη σύγχρονη ροή middleware που βασίζεται σε async/await. Η καμπύλη μάθησης του μπορεί να είναι απότομη για όσους δεν είναι εξοικειωμένοι με τα πρότυπα ασύγχρονης JavaScript και τις generator functions. Οι developers που κάνουν μετάβαση από το Express μπορεί να βρουν διαφορετική την προσέγγιση του Koa στο middleware και στον χειρισμό σφαλμάτων, απαιτώντας επιπλέον χρόνο προσαρμογής. Ωστόσο, η απλότητα του Koa και η εστίασή του στην αξιοποίηση

των σύγχρονων χαρακτηριστικών του JavaScript μπορούν να το κάνουν ελκυστικό σε προγραμματιστές που αναζητούν ένα πιο σύγχρονο πλαίσιο. Η ολοκληρωμένη τεκμηρίωση και μια χρήσιμη κοινότητα συμβάλλουν στην απλοποίηση της διαδικασίας μάθησης.

Πλαίσιο	**Βαθμός**	**Καμπύλη εκμάθησης**
Express	2	Φιλικό προς τους αρχάριους με μινιμαλιστικό σχεδιασμό και άμεσα διαθέσιμη εκτεταμένη τεκμηρίωση. Απλό σύστημα ενδιάμεσου λογισμικού για δρομολόγηση και άλλες δυνατότητες. Μεγάλη κοινοτική υποστήριξη.
Fastify	3	Έμφαση στην απόδοση και χαμηλό κόστος λειτουργίας. Ολοκληρωμένη τεκμηρίωση με πλήρεις οδηγούς χρήσης και υποστηρικτική κοινότητα. Παρόμοιες έννοιες με το Express αλλά απαιτεί κατανόηση βελτιστοποιήσεων.
Hapi	4	Προσδίδει ιδιαίτερη σημασία στη διαμόρφωση και τις συμβάσεις. - Μέτρια καμπύλη μάθησης λόγω της ξεκάθαρης δομής. - Εκτενής τεκμηρίωση και έμφαση στις βέλτιστες πρακτικές.
Koa	3	Ελαφρύ με σύγχρονη ροή middleware που βασίζεται σε async/await. - Απότομη καμπύλη μάθησης για patterns async JavaScript. - Διαφορετική προσέγγιση στο middleware σε σύγκριση με το Express.

Πίνακας 4.10: Σύγκριση πλαισίων Back-End ως προς την καμπύλη εκμάθησης

4.2.2.3 Καμπύλη Εκμάθησης Full-Stack πλαισίων

Τα Meteor, Feather, και Next.js είναι τρία δημοφιλή εργαλεία για τη δημιουργία web εφαρμογών. Ωστόσο, η καμπύλη εκμάθησής τους διαφέρει:

Meteor:

Η καμπύλη εκμάθησης του Meteor είναι σχετικά χαμηλή, καθώς παρέχει ένα ολοκληρωμένο περιβάλλον για την ανάπτυξη εφαρμογών. Με το Meteor, μπορείτε να δημιουργήσετε γρήγορα και εύκολα εφαρμογές, χωρίς την ανάγκη για πολλά διαφορετικά εργαλεία. Η εκμάθηση του Meteor απαιτεί κυρίως γνώσεις στη JavaScript.

Feathers:

Το Feathers είναι ένα ελαφρύ framework για τη δημιουργία των API. Η καμπύλη εκμάθησης του είναι σχετικά χαμηλή, καθώς παρέχει έναν απλό και ευέλικτο τρόπο για τη δημιουργία API. Είναι ιδανικό για ανάπτυξη back-end λύσεων με ελάχιστη πολυπλοκότητα.

Next.js:

Η καμπύλη εκμάθησης του Next.js μπορεί να είναι μέτρια προς υψηλή, καθώς προσφέρει περισσότερες λειτουργίες και δυνατότητες σε σύγκριση με τα παραπάνω. Ωστόσο, με το Next.js μπορείτε να δημιουργήσετε πολύ πιο πολύπλοκες και ευέλικτες εφαρμογές, καθώς παρέχει προηγμένες λειτουργίες όπως την αποδοτική αποτύπωση από την πλευρά του διακομιστή και την εκτέλεση κώδικα στην πλευρά του server. Η εκμάθηση του Next.js απαιτεί γνώσεις στη JavaScript και στην ανάπτυξη web εφαρμογών.

Στον Πίνακα 4.11, παρουσιάζουμε την καμπύλη εκμάθησης των παραπάνω πλαισίων.

Πλαίσιο	**Βαθμός**	**Καμπύλη εκμάθησης**
Meteor	2	Παρέχει ένα ολοκληρωμένο περιβάλλον ανάπτυξης με προεγκατεστημένα πακέτα και λειτουργίες. Βασίζεται σε JavaScript.
Feathers	2	Είναι ένα ελαφρύ framework για τη δημιουργία απλών και γρήγορων back-end λύσεων.
Next.js	3	Παρέχει προηγμένες λειτουργίες όπως την αποδοτική αποτύπωση από την πλευρά του διακομιστή, δυναμική δρομολόγηση και στατική παραγωγή ιστοσελίδων. Απαιτεί γνώσεις στο React.

Πίνακας 4.11: Σύγκριση πλαισίων Full-Stack ως προς την καμπύλη εκμάθησης

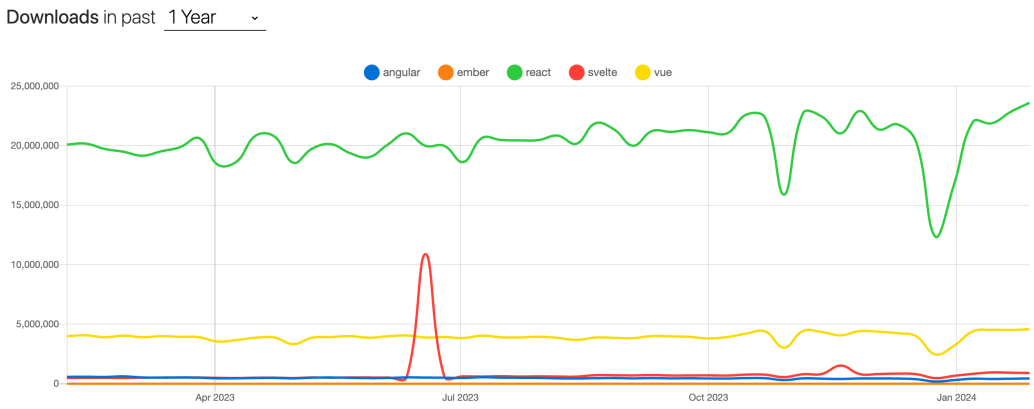
4.2.3 Σύγκριση Βάσει Δημοφιλίας Πλαισίων

4.2.3.1 Δημοφιλία Front End πλαισίων

Σύμφωνα με το npmtrends.com(Εικόνα 4.1) τους τελευταίους 12 μήνες η React είναι εμφανώς η πιο δημοφιλής σε μεγάλη απόσταση από το δεύτερο πλαίσιο. Αυτό το οποίο κάνει εντύπωση φυσικά, είναι η ραγδαία ανάπτυξη του Vue το οποίο βρίσκεται στην δεύτερη θέση, ενώ το Svelte έχει ξεπεράσει το Vue και Angular, το οποίο είναι αξιοσημείωτο βάσει του γεγονότος ότι η Angular είναι πλαίσιο που έχει τα τελευταία χρόνια την εποποτεία της Google και παρέχει πολλές δυνατότητες. Συμπερένουμε λοιπόν ότι η ευκολία εκμάθησης ενός πλαισίου παίζει μείζονα σημασία στην δημοφιλία του. Αν και η δημοφιλία της React είναι ένας πολύτιμος δείκτης, είναι σημαντικό να λάβουμε υπόψη μια ευρύτερη προοπτική κατά την αξιολόγηση των τάσεων:

- Ενώ το React οδηγεί στις λήψεις, δεν το καθιστά αυτόματα την "καλύτερη" επιλογή για κάθε έργο. Κάθε πλαίσιο έχει τα δυνατά και τα αδύνατα σημεία του και η βέλτιστη επιλογή εξαρτάται από συγκεκριμένες ανάγκες του έργου, την τεχνογνωσία της ομάδας και τα επιθυμητά χαρακτηριστικά.

- Ενώ το React βρίσκεται επί του παρόντος στην κορυφή, πλαίσια όπως το Svelte και το Vue γνωρίζουν σημαντική ανάπτυξη. Η καινοτόμος προσέγγιση της Svelte και η ισορροπία απλότητας και ευελιξίας του Vue προσελκύουν προγραμματιστές.
- Είναι δύσκολο να προβλεφθεί το μέλλον, αλλά οι αναδυόμενες τάσεις όπως το WebAssembly και τα micro frontends θα μπορούσαν να επηρεάσουν τις προτιμήσεις του πλαισίου.
- Οι ανάγκες της επιχείρησης, οι συγκεκριμένες απαιτήσεις απόδοσης και οι προτιμήσεις της ομάδας παίζουν επίσης ρόλο στην επιλογή ενός πλαισίου.



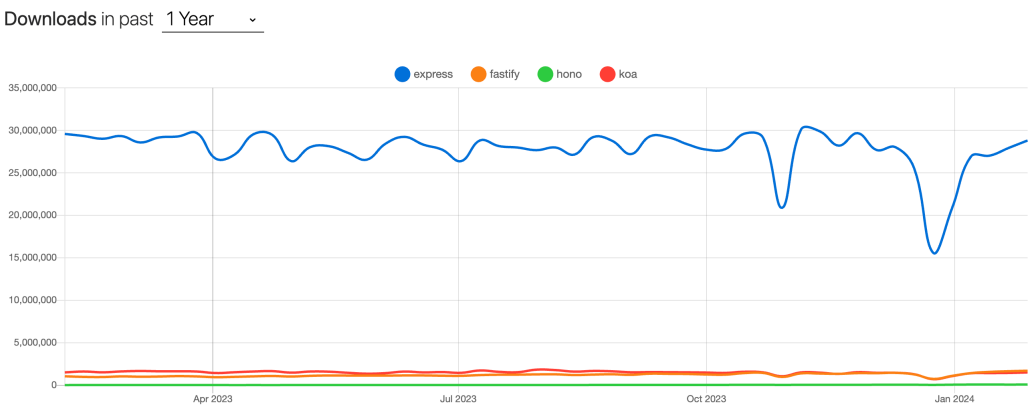
Σχήμα 4.1: Τάσεις τους τελευταίους 12 μήνες στα front-end πλαίσια

4.2.3.2 Δημοφιλία Back-End πλαισίων

Αντίστοιχα, αναζητήσαμε την δημοφιλία των back-end πλαισίων στον ιστότοπο του npmtrends, εφόσον όλα τα πλαίσια μπορούν να βρεθούν στον διαχειριστή πακέτων npm. Η σύγκριση λοιπόν δείχνει πως το express είναι το πλαίσιο με την μακράν μεγαλύτερη δημοφιλία. Αυτό μπορεί να είναι ένα αναμενόμενο γεγονός, εφόσον το express είναι το παλαιότερο από τα συγκρινόμενα πλαίσια.

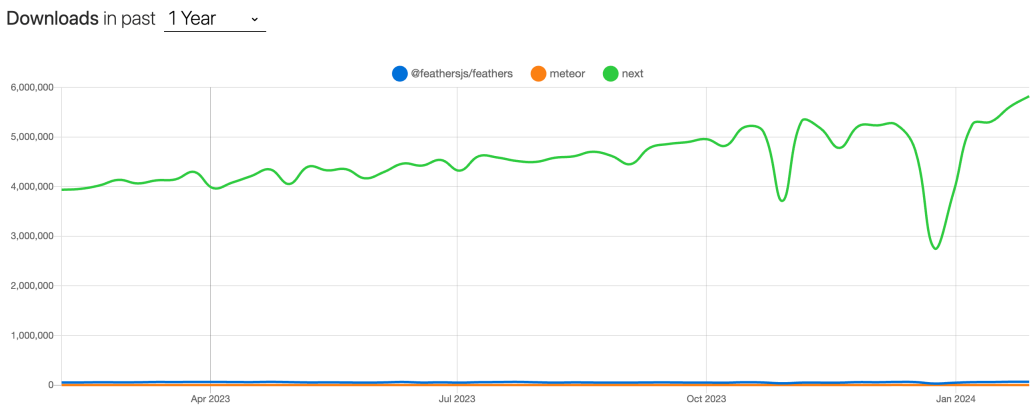
4.2.3.3 Δημοφιλία Full-Stack πλαισίων

Όσον αφορά τη δημιουργία συναρπαστικών εμπειριών ιστού, η επιλογή του σωστού full-stack πλαισίου είναι ζωτικής σημασίας. Ενώ το Next.js φαίνεται πως κυριαρχεί σε δημοτικότητα, όπως φαίνεται και στην Εικόνα 4.3, άλλα πλαίσια όπως το Meteor και το FeathersJS προσφέρουν επίσης μοναδικά πλεονεκτήματα. Ωστόσο, το Next.js, ένα πλαίσιο που φημίζεται για την ικανότητά του να χειρίζεται πλατφόρμες μεγάλης κλίμακας, υψηλής επισκεψιμότητας, έχει συγκεντρώσει αξιοσημείωτη προσοχή από τις μεγάλες επιχειρήσεις του κλάδου. Η ενσωμάτωση του Next.js σε μεγάλου εύρους έργα και η υιοθέτησή του για αυτά από γνωστές εταιρίες όπως η Washington Post, η Loom και το Netflix



Σχήμα 4.2: Τάσεις τους τελευταίους 12 μήνες στα back-end πλαίσια

υπογραμμίζει την ικανότητά του να ανταποκρίνεται στις απαιτητικές απαιτήσεις των σύγχρονων ψηφιακών περιβαλλόντων. Αυτή η ευρεία υιοθέτηση επεκτείνεται περαιτέρω σε πλατφόρμες με επιρροή όπως οι TikTok, Twitch και Notion, ενισχύοντας το Next.js ως όχι απλώς ένα πλαίσιο, αλλά ως έναν αξιόπιστο σύμμαχο για επιχειρήσεις και δημιουργούς που πλοηγούνται στην πρώτη γραμμή της ψηφιακής επανάστασης [16].



Σχήμα 4.3: Τάσεις τους τελευταίους 12 μήνες στα full-stack πλαίσια

4.3 Στατική ανάλυση κώδικα

Στο συνεχώς εξελισσόμενο τοπίο της ανάπτυξης Ιστού, η JavaScript είναι μια τεχνολογία πλέον που μπορεί να χαρακτηριστεί ως ακρογωνιαίος λίθος, δίνοντας τη δυνατότητα στους προγραμματιστές να δημιουργούν δυναμικές και διαδραστικές εφαρμογές ιστού. Ωστόσο, η ευρεία υιοθέτηση της JavaScript προκαλεί επίσης μια έντονη ανησυχία για την ασφάλεια, καθώς η γλώσσα σε πολλές περιπτώσεις, εάν και όχι πάντα πλέον, εκτελείται από την πλευρά του πελάτη, εκθέτοντας τις εφαρμογές σε διάφορα τρωτά σημεία. Καθώς αυξάνεται η ζήτηση για πλούσιες σε χαρακτηριστικά και πολύπλοκες εφαρμογές Ιστού, αυξάνεται και η ανάγκη αξιολόγησης και ενίσχυσης της θέσης ασφαλείας των υ-

ποκείμενων πλαισίων και βιβλιοθηκών.

Αυτό το κεφάλαιο εμβαθύνει στην κριτική ανάλυση των τρωτών σημείων ασφαλείας στα πλαίσια JavaScript, αλλά και στον έλεγχο ποιότητας τους, χρησιμοποιώντας προηγμένα εργαλεία σάρωσης όπως το SonarQube. Στόχος είναι να εξεταστούν τα εγγενή χαρακτηριστικά ασφαλείας, οι πιθανές αδυναμίες που υπάρχουν στα δημοφιλή πλαίσια JavaScript, καθώς και πιθανά σφάλματα (bugs)ρίχνοντας φως σε τομείς που απαιτούν προσοχή από προγραμματιστές, επαγγελματίες ασφαλείας και την ευρύτερη κοινότητα ανάπτυξης λογισμικού.

Η σημασία αυτής της έρευνας έγκειται στον προληπτικό εντοπισμό και τον μετριασμό των κινδύνων ασφάλειας, συμβάλλοντας τελικά στη δημιουργία πιο ισχυρών και ανθεκτικών διαδικτυακών εφαρμογών. Αξιοποιώντας αυτοματοποιημένα εργαλεία όπως το SonarQube, στοχεύουμε να διεξαγάγουμε μια ολοκληρωμένη εξέταση των πλαισίων JavaScript, διευκρινίζοντας τόσο κοινές όσο και συγκεκριμένες ευπάθειες που θα μπορούσαν να θέσουν σε κίνδυνο την ακεραιότητα, την εμπιστευτικότητα και τη διαθεσιμότητα των εφαρμογών Ιστού.

Οι επόμενες υποενότητες θα παρέχουν μια επισκόπηση της μεθοδολογίας που χρησιμοποιείται για την αξιολόγηση ευπάθειας, τα κριτήρια επιλογής για πλαίσια JavaScript υπό έλεγχο και μια λεπτομερή παρουσίαση των ευρημάτων. Μέσω αυτής της εξερεύνησης, επιδιώκουμε να δώσουμε στους προγραμματιστές πληροφορίες για πιθανές απειλές, δίνοντάς τους τη δυνατότητα να λαμβάνουν τεκμηριωμένες αποφάσεις σχετικά με την επιλογή πλαισίου, τη διαμόρφωση και τις στρατηγικές υλοποίησης που ευθυγραμμίζονται με τις σύγχρονες βέλτιστες πρακτικές ασφάλειας.

SonarQube

Το SonarQube είναι μια πλατφόρμα ανοιχτού κώδικα που έχει σχεδιαστεί για να επιθεωρεί και να αναλύει συνεχώς την ποιότητα του πηγαίου κώδικα καθ' όλη τη διάρκεια της διαδικασίας ανάπτυξης λογισμικού. Χρησιμεύει ως ένα ισχυρό εργαλείο ανάλυσης στατικού κώδικα που εντοπίζει και αντιμετωπίζει ζητήματα ποιότητας κώδικα, ευπάθειες ασφαλείας και ανεπιθύμητες ανωμαλίες ή σφάλματα στον κώδικα. Αρχικά αναπτύχθηκε από τη SonarSource, το SonarQube έχει γίνει ένα ευρέως χρησιμοποιούμενο εργαλείο στην κοινότητα ανάπτυξης λογισμικού για τη διατήρηση και τη βελτίωση της συνολικής υγείας του κώδικα των προγραμμάτων.

Ένα από τα βασικά χαρακτηριστικά του SonarQube είναι η ικανότητά του να υποστηρίζει πολλαπλές γλώσσες προγραμματισμού, καθιστώντας το μια ευέλικτη λύση για διαφορετικά προγράμματα λογισμικού. Παρέχει ένα ολοκληρωμένο σύνολο κανόνων και ελέγχων προσαρμοσμένων σε κάθε υποστηριζόμενη γλώσσα, επιτρέποντας στους προγραμματιστές να επιβάλλουν πρότυπα κωδικοποίησης και βέλτιστες πρακτικές. Επιπλέον, το SonarQube προσφέρει ανατροφοδότηση σε πραγματικό χρόνο και λεπτομερείς αναφορές, επιτρέποντας στις ομάδες ανάπτυξης να εντοπίζουν γρήγορα και να δίνουν προτε-

ραιότητα σε θέματα που χρειάζονται προσοχή.

Η ανάλυση του SonarQube καλύπτει ένα ευρύ φάσμα πτυχών ποιότητας κώδικα, συμπεριλαμβανομένης της συντήρησης, της αξιοπιστίας, της ασφάλειας και των αρχείων δοκιμών. Χρησιμοποιεί μια ποικιλία τεχνικών στατικής ανάλυσης για να εξετάσει τον κώδικα χωρίς να τον εκτελέσει, προσφέροντας πληροφορίες για πιθανά σφάλματα, αλληλεπικαλύψεις κώδικα και ευπάθειες ασφαλείας. Η πλατφόρμα υποστηρίζει επίσης την ενσωμάτωση με δημοφιλή συστήματα CI/CD (Continuous Integration/Continuous Deployment), επιτρέποντας την αυτοματοποιημένη ανάλυση κώδικα ως μέρος του αγωγού ανάπτυξης.

Η διεπαφή SonarQube παρέχει έναν διαισθητικό πίνακα εργαλείων που εμφανίζει βασικές μετρήσεις, τάσεις και οπτικές αναπαραστάσεις ποιότητας κώδικα με την πάροδο του χρόνου. Αυτό βοηθά τις ομάδες ανάπτυξης να παρακολουθούν την πρόοδό τους στην αντιμετώπιση ζητημάτων κώδικα και τη διατήρηση υψηλού επιπέδου ποιότητας κώδικα. Η πλατφόρμα διευκολύνει επίσης τη συνεργασία μεταξύ των μελών της ομάδας, παρέχοντας χαρακτηριστικά όπως η διακλάδωση έργων και η ενσωμάτωση με δημοφιλή εργαλεία ανάπτυξης.

Εκτός από τα βασικά χαρακτηριστικά του, το SonarQube υποστηρίζει επεκτασιμότητα μέσω plugins, επιτρέποντας στους χρήστες να προσαρμόσουν και να επεκτείνουν τη λειτουργικότητά του ώστε να ανταποκρίνονται σε συγκεκριμένες απαιτήσεις του έργου. Αυτή η προσαρμοστικότητα έχει συμβάλει στη δημοτικότητα της SonarQube ως εργαλείο για οργανισμούς που προσπαθούν να επιτύχουν και να διατηρήσουν υψηλά πρότυπα ποιότητας κώδικα στις διαδικασίες ανάπτυξης λογισμικού τους. Συνολικά, το SonarQube διαδραματίζει κρίσιμο ρόλο στην προώθηση μιας κουλτούρας συνεχούς βελτίωσης και παροχής λογισμικού με λιγότερα ελαττώματα και ενισχυμένη ασφάλεια.

Το SonarQube κατά την ολοκλήρωση του ελέγχου ενός έργου, εμφανίζει διάφορα αποτελέσματα τα οποία χωρίζονται σε 6 διαφορετικές κατηγορίες. Την αξιοπιστία, την συντηρησιμότητα, την ασφάλεια, το επίπεδο ελέγχου ασφάλειας, την κάλυψη και τα διπλότυπα κώδικα. Στις επόμενες υποενότητες θα επικεντρωθούμε σε κάποιες από αυτές τις μετρικές για να μπορέσουμε να αξιολογήσουμε και να συγκρίνουμε τα πρωτόκολλα JavaScript. Πιο συγκεκριμένα θα χρησιμοποιήσουμε τα αποτελέσματα της ασφάλειας για να μπορέσουμε να αξιολογήσουμε τα επίπεδα ασφάλειας των πλαισίων, αλλά και έναν συνδιασμό της αξιοπιστίας, της συντηρησιμότητας, της ασφάλειας και της κάλυψης για να εξάγουμε έναν συνολικό δείκτη ποιότητας κώδικα. Στην συνέχεια θα εξηγήσουμε την κάθε κατηγορία ξεχωριστά, η οποία αποτελεί μέρος του δείκτη ποιότητας κώδικα που θα δημιουργήσουμε:

Αξιοπιστία: Αυτός ο δείκτης αξιολογεί την ικανότητα του κώδικα να λειτουργεί σωστά και σταθερά. Λαμβάνει υπόψη κακές πρακτικές κώδικα (code smells), πιθανές τοποθεσίες σφαλμάτων και τη γενική υγεία του κώδικα. Χαμηλότερες τιμές συνήθως υποδεικνύουν υψηλότερη αξιοπιστία και χαμηλότερο κίνδυνο σφαλμάτων.

Συντηρησιμότητα: Αυτός ο δείκτης μετράει την προσπάθεια που απαιτείται για την κατανόηση, τροποποίηση και επέκταση του κώδικα. Περιλαμβάνει παράγοντες όπως η πολυπλοκότητα του κώδικα, οι συμβάσεις ονοματοδοσίας και τα σχόλια. Χαμηλότερες τιμές υποδεικνύουν ευκολότερη συντήρηση και μειωμένο μελλοντικό τεχνικό χρέος.

Ασφάλεια: Αυτός ο δείκτης εντοπίζει πιθανές ευπάθειες ασφαλείας στον κώδικα, όπως ελαττώματα εγχύσεων, ανασφαλείς αναφορές σε αντικείμενα και προβλήματα κρυπτογράφησης. Χαμηλότερες τιμές υποδεικνύουν λιγότερους κινδύνους ασφαλείας και μια πιο ασφαλή βάση κώδικα.

Κάλυψη: Αυτός ο δείκτης δείχνει το ποσοστό του κώδικα που καλύπτεται από αυτοματοποιημένες δοκιμές. Ενώ η υψηλή κάλυψη δεν εγγυάται την έλλειψη σφαλμάτων, αυξάνει την εμπιστοσύνη στη λειτουργικότητα του κώδικα και αποκαλύπτει λιγότερο δοκιμασμένες περιοχές.

Επιπλέον, κάθε έργο μπορεί να έχει διαφορετικά πρότυπα ποιότητας και να δίνει προτεραιότητα σε συγκεκριμένους δείκτες ανάλογα με τη φύση και τις απαιτήσεις του. Ο ορισμός ορίων και η παρακολούθηση τάσεων με την πάροδο του χρόνου παρέχει πολύτιμες πληροφορίες για την εξέλιξη και τη βελτίωση της ποιότητας του κώδικα.

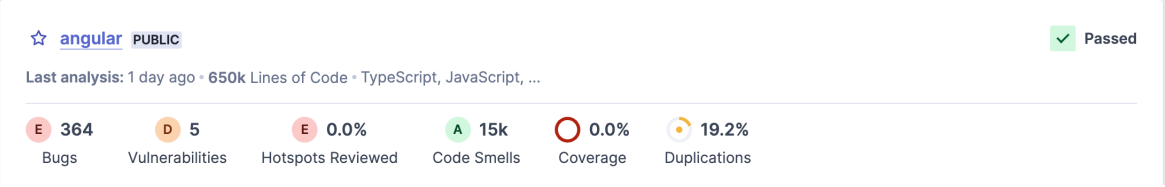
4.3.1 Σύγκριση Ασφάλειας Πλαισίων

4.3.1.1 Σύγκριση Ασφάλειας Front-End Πλαισίων

Με την χρήση του SonarQube ελέγξαμε τις τρωτότητες των Front-End πλαισίων και τα αποτελέσματα παρουσιάζονται στην τρέχουσα υποενότητα.

Τρωτότητες στην Angular

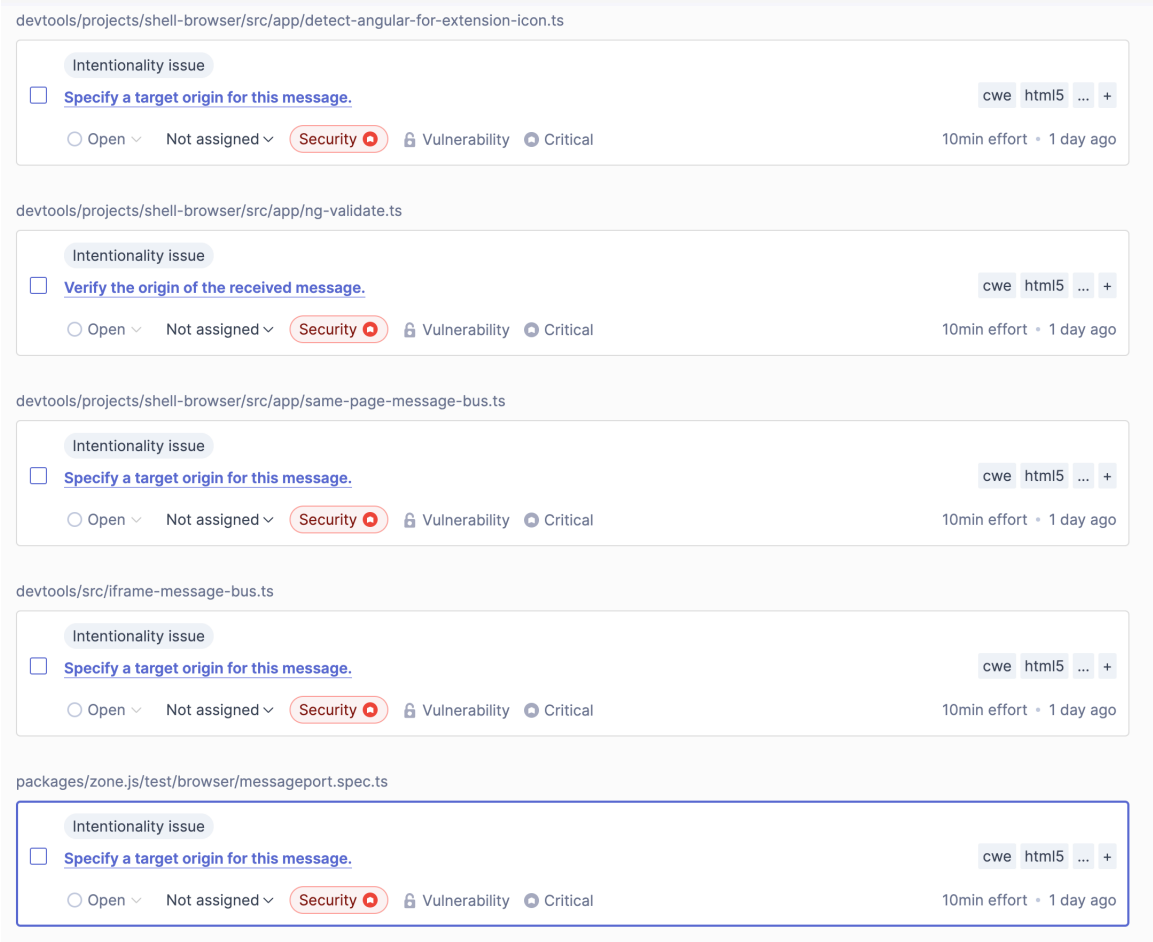
Σε παρόμοια κλίμακα με τα runtime πλαίσια που ελέγξαμε και πιο πάνω, ο έλεγχος της Angular απέφερε έναν μικρό αριθμό από ευπάθειες, ο οποίος όμως σε αυτην την περίπτωση είναι μικρότερος από τις 10. Πιο συγκεκριμένα, όπως φαίνεται και από την Εικόνα 4.4, έχουν ανακαλυφθεί 5 ευπάθειες και 448 security hotspots.



Σχήμα 4.4: Αποτελέσματα ανάλυσης του SonarQube για το πλαίσιο Angular

Είναι σημαντικό να παρατηρήσουμε ότι και οι 5 ευπάθειες του πλαισίου δεν αφορούν κάποιο ζωτικό κομμάτι του ίδιου του πλαισίου, αλλά κομμάτια των αρχείων δοκιμών. Πιο συγκεκριμένα, οι 4 ευπάθειες αφορούν τα Angular DevTools, τα οποία είναι μια επέκταση

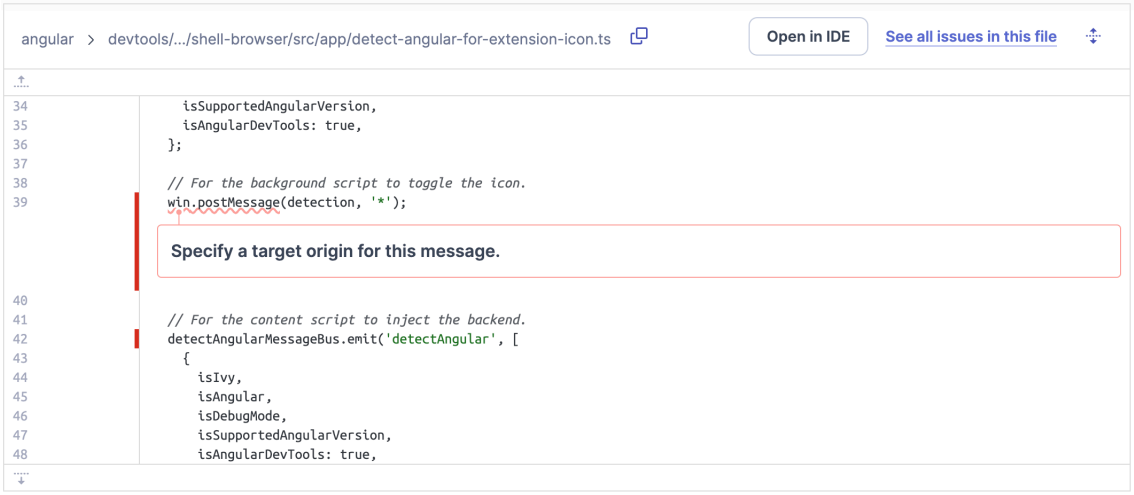
DevTools για τον φυλλομετρητή που χρησιμεύει στον εντοπισμό σφαλμάτων και τη δημιουργία προφίλ επιδόσεων για εφαρμογές Angular, ενώ η τελευταία αφορά αποκλειστικά ένα αρχείο δοκιμών, όπως φαίνεται και στην Εικόνα 4.5.



Σχήμα 4.5: Λίστα ευπαθειών για το πλαίσιο Angular

Οι 4 από τις 5 ευπάθειες αφορούν το ίδιο πρόβλημα, δηλαδή τον προσδιορισμό του στόχου προέλευσης για το μήνυμα. Πιο συγκεκριμένα, το σφάλμα αφορά σε πιθανό θέμα ασφαλείας που σχετίζεται με την επικοινωνία μεταξύ διαφορετικών τμημάτων του κώδικα, ειδικά με τη χρήση της συνάρτησης postMessage, όπως απεικονίζεται στην Εικόνα 4.6.

- **Συνάρτηση postMessage:** Αυτή η συνάρτηση επιτρέπει την επικοινωνία μεταξύ διαφορετικών παραθύρων (φυλλομεταφυλλιστών ή πλαισίων) εντός του ίδιου τομέα. Στέλνει ένα μήνυμα από ένα παράθυρο σε άλλο, παρέχοντας έναν τρόπο για αλληλεπίδραση διαφορετικών τμημάτων κώδικα.
- **Στόχος προέλευσης:** Αυτό καθορίζει τον ιστότοπο ή τον τομέα στον οποίο προορίζεται το μήνυμα. Διασφαλίζει ότι η επικοινωνία πραγματοποιείται μόνο μεταξύ των προβλεπόμενων μερών και αποτρέπει την αποστολή μη εξουσιοδοτημένων μηνυμάτων στην εφαρμογή.



Σχήμα 4.6: Ευπάθεια 1: "Προσδιορισμός του στόχου προέλευσης για το μήνυμα"

Το μήνυμα σφάλματος προκύπτει επειδή η συνάρτηση postMessage χρησιμοποιείται χωρίς να καθορίζεται τον στόχο προέλευσης. Αυτό σημαίνει ότι αποστέλλεται ένα μήνυμα χωρίς να δηλώνεται ρητά ποιος είναι ο παραλήπτης. Σε ένα ασφαλές περιβάλλον, αυτό θεωρείται πιθανή ευπάθεια για τους εξής λόγους:

- **Μη προβλεπόμενοι παραλήπτες:** Εάν ο στόχος προέλευσης δεν καθοριστεί, οποιοδήποτε website ή σενάριο που φορτώνεται στην ίδια σελίδα θα μπορούσε δυνητικά να λάβει το μήνυμα. Αυτό θα μπορούσε να οδηγήσει σε εισβολείς που εισάγουν κακόβουλα σενάρια, αποκτώντας ενδεχομένως πρόσβαση στα δεδομένα του εξυπηρετητή ή χειραγωγώντας την εφαρμογή/πλαίσιο.
- **Εκθέσεις δεδομένων:** Το περιεχόμενο του μηνύματος μπορεί να περιέχει ευαίσθητες πληροφορίες, όπως δεδομένα χρήστη ή κατάσταση εφαρμογής. Η αποστολή του χωρίς σωστή επαλήθευση προέλευσης αυξάνει τον κίνδυνο πρόσβασης μη εξουσιοδοτημένων φορέων σε αυτά τα δεδομένα.

Επίλυση προβλήματος

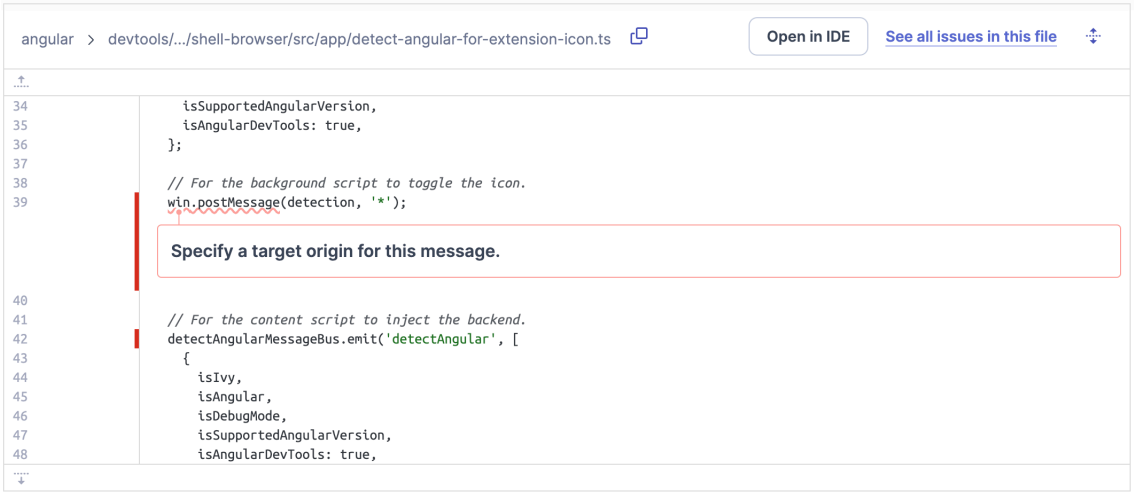
Προσδιορισμός στόχου προέλευσης: Όταν χρησιμοποιείται η συνάρτηση postMessage, θα πρέπει να ορίζεται ρητά ο ιστότοπος ή ο τομέας του επιδιωκόμενου παραλήπτη. Αυτό βοηθά στον περιορισμό της ροής των μηνυμάτων και στον μετριασμό πιθανών κινδύνων ασφαλείας.

Εξέταση του περιβάλλοντος: Ανάλυση του κώδικα όπου εμφανίζεται το σφάλμα και κατανόηση του λόγου που δεν καθορίζεται ο στόχος προέλευσης. Αναζήτηση πιθανών προβλημάτων όπου ο επιδιωκόμενος παραλήπτης μπορεί να μην είναι σαφής ή όπου το μήνυμα ενδέχεται να φτάνει σε μη προβλεπόμενους στόχους.

Χρήση βέλτιστων πρακτικών: Συμβουλευτείτε την τεκμηρίωση και τις οδηγίες κωδικοποίησης σχετικά με τη χρήση του postMessage με ασφάλεια. Θα πρέπει ο κώδικας να

συμμορφώνεται με ασφαλείς πρακτικές επικοινωνίας, συμπεριλαμβανομένης της σωστής επαλήθευσης προέλευσης και χειρισμού δεδομένων.

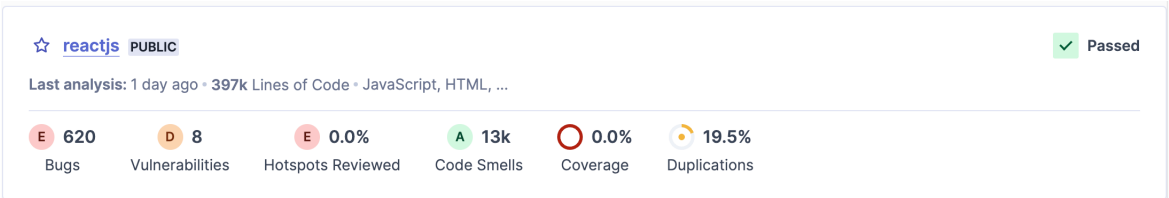
Η τελευταία ευπάθεια αφορά την επαλήθευση της προέλευσης του ληφθέντος μηνύματος, όπως απεικονίζεται στην Εικόνα 4.7. Το μήνυμα σφάλματος υποδηλώνει πιθανή ευπάθεια ασφαλείας στον κώδικα του πλαισίου, πιο συγκεκριμένα στα αρχεία δοκιμών. Προειδοποιεί για κώδικα που αλληλεπιδρά με μηνύματα από εξωτερικές πηγές χωρίς να επαληθεύει σωστά την προέλευσή τους. Η ευπάθεια αυτή είναι παρόμοια με τις προηγούμενες 4, για τον λόγο αυτό η επίλυση του προβλήματος είναι ίδια με την Ευπάθεια 1.



Σχήμα 4.7: Ευπάθεια 2: "Επαλήθευση της προέλευσης του ληφθέντος μηνύματος"

Τρωτότητες στο React

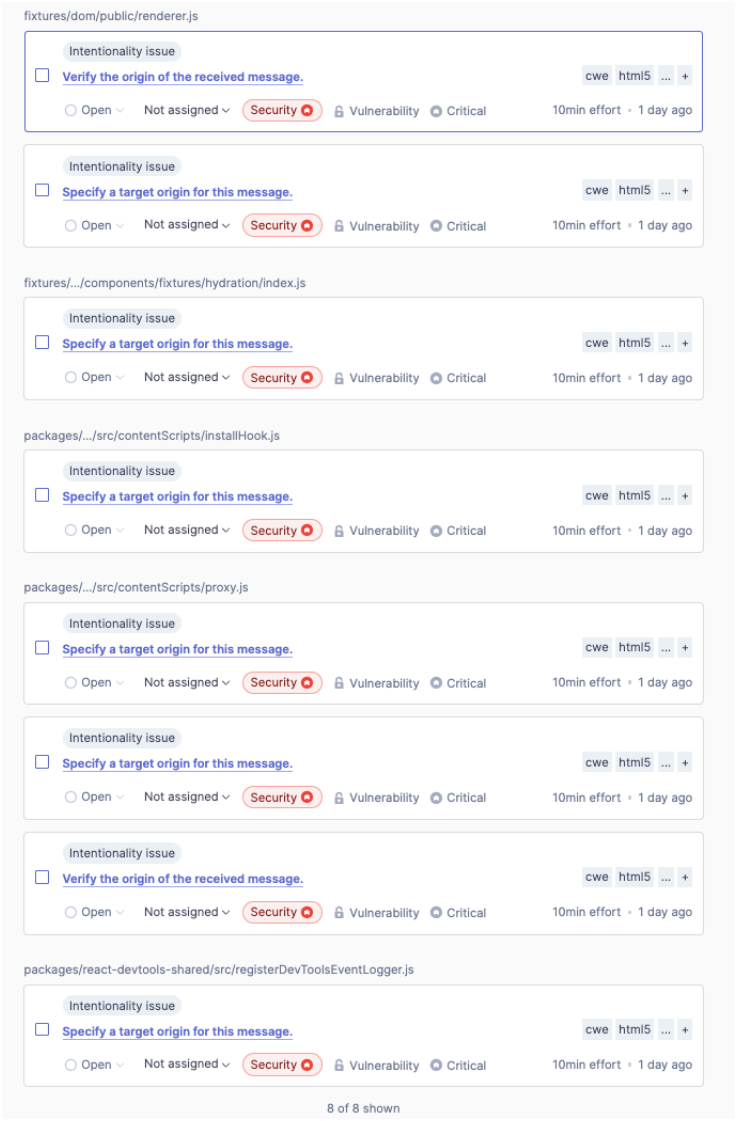
Το React παρουσιάζει 8 ευπάθειες και 239 security hotspots σύμφωνα με το SonarQube, όπως εμφανίζονται και στην Εικόνα 4.8 και για τον λόγο αυτό έχει βαθμολογηθεί από το εργαλείο ανάλυσης με βαθμό D (εφόσον υπάρχει τουλάχιστον μια ευπάθεια κρίσιμου επιπέδου) για τις ευπάθειες και με E για τα security hotspots.



Σχήμα 4.8: Αποτελέσματα ανάλυσης του SonarQube για το πλαίσιο React

Όλες οι ευπάθειες αφορούν τα δύο είδη ευπαθειών που αναλύθηκαν και στο πλαίσιο της Angular, για αυτόν τον λόγο δεν θα αναλύσουμε περαιτέρω τα συγκεκριμένα αποτελέσματα. Όπως παρατηρούμε και από την Εικόνα 4.9, όλες οι ευπάθειες και σε αυτήν

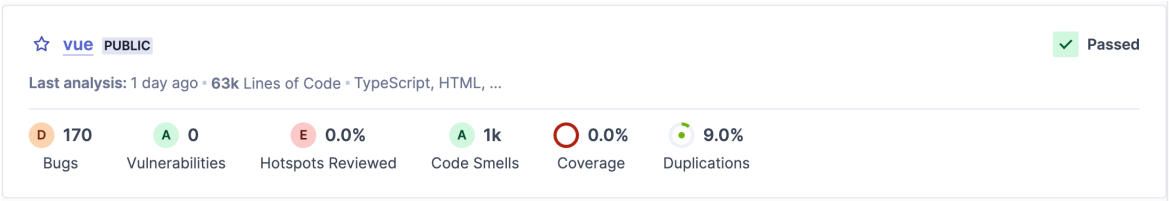
την περίπτωση αφορούν αρχεία δοκιμών του πλαισίου. Για παράδειγμα οι πρώτες 2 ευπάθειες αφορούν το αρχείο renderer.js που βρίσκεται στον φάκελο fixtures/dom/public. Ο φάκελος αυτός σύμφωνα με το επίσημο αποθετήριο του πλαισίου αφορά ένα σύνολο περιπτώσεων δοκιμών στο δέντρο DOM για γρήγορο εντοπισμό προβλημάτων περιηγητή.



Σχήμα 4.9: Αποτελέσματα ανάλυσης του SonarQube για το πλαίσιο React

Τρωτότητες στο Vue

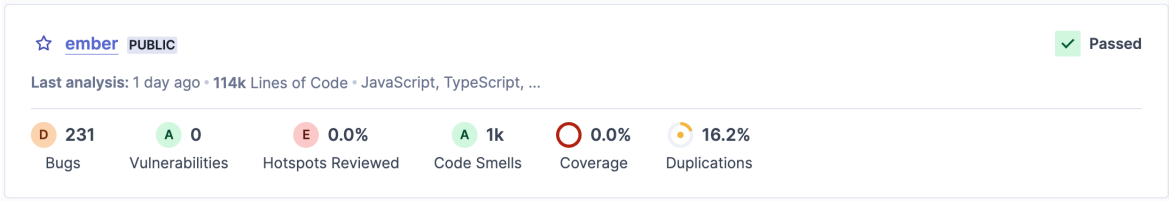
Το Vue είναι το πρώτο πλαίσιο από όσα έχουμε ελέγξει στο οποίο δεν έχουν ανακαλυφθεί καθόλου ευπάθειες από το SonarQube. Αυτό ίσως είναι λογικό δεδομένου ότι το Vue έχει τον μικρότερο αριθμό γραμμών κώδικα, που φτάνει τις 63 χιλιάδες. Το vue επίσης έχει έναν πολύ μικρό αριθμό από security hotspots που είναι 57.



Σχήμα 4.10: Αποτελέσματα ανάλυσης του SonarQube για το πλαίσιο Vue

Τρωτότητες στο Ember

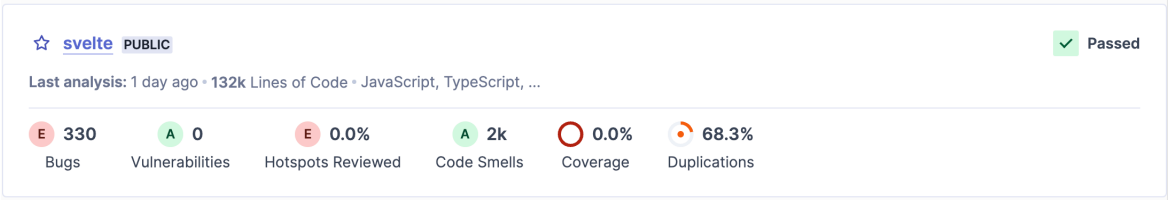
Το ember επίσης είναι ένα πλαίσιο που δεν έχει καθόλου ευπάθειες βάσει του SonarQube. Ανακαλύφθηκαν επίσης 40 security hotspots.



Σχήμα 4.11: Αποτελέσματα ανάλυσης του SonarQube για το πλαίσιο Ember

Τρωτότητες στο Svelte

Το τελευταίο πλαίσιο από τα πλαίσια Front-End που ελέγθηκαν, εμφάνισε επίσης 0 ευπάθειες. Το SonarQube ανακάλυψε επίσης 45 security hotspots.



Σχήμα 4.12: Αποτελέσματα ανάλυσης του SonarQube για το πλαίσιο Svelte

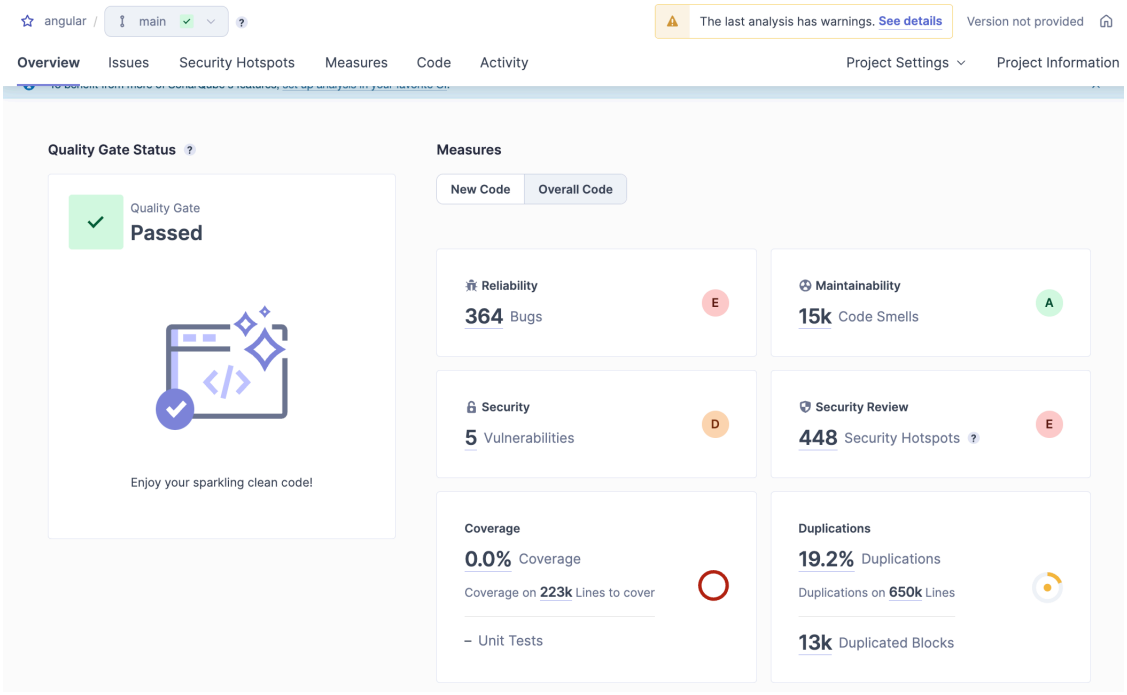
4.3.2 Σύγκριση Ποιότητας Κώδικα Πλαισίων

4.3.2.1 Σύγκριση Ποιότητας Κώδικα Front-End Πλαισίων

Στην συγκεκριμένη υποενότητα θα συγκρίνουμε τα πλαίσια ως προς την ποιότητα κώδικά τους. Τα αποτελέσματα του SonarQube θα μας βοηθήσουν να εξάγουμε πιο ασφαλή συμπεράσματα σχετικά με την ύπαρξη bugs και σφαλμάτων στον κώδικα του κάθε πλαισίου, όπως και διπλότυπα ή κακές πρακτικές χρήσης κώδικα.

Ποιότητα Κώδικα Angular

Με βάση τα αποτελέσματα του SonarQube, όπως εμφανίζονται και στην Εικόνα 4.13 η γενική ποιότητα του κώδικα φαίνεται να είναι καλή. Η Πύλη Ποιότητας έχει περάσει, το οποίο δείχνει ότι ο κώδικας πληροί τα διαμορφωμένα πρότυπα ποιότητας. Δεν υπάρχουν νέα ζητήματα κώδικα, πράγμα που σημαίνει ότι δεν έχουν εισαχθεί νέα σφάλματα ή ευπάθειες στον πιο πρόσφατο κώδικα.



Σχήμα 4.13: Αποτελέσματα γενικής ποιότητας κώδικα για το πλαίσιο Angular

Εντούτοις, υπάρχουν ορισμένοι τομείς για βελτίωση. Εξακολουθούν να εντοπίζονται 364 σφάλματα και 15.000 κακές πρακτικές κώδικα στον κώδικα. Αυτό υποδηλώνει ότι μπορεί να υπάρχουν πιθανά προβλήματα στον κώδικα που θα μπορούσαν να δυσκολέψουν τη συντήρηση ή την κατανόησή του στο μέλλον. Επιπλέον, η κάλυψη κώδικα είναι στο 0%, το οποίο σημαίνει ότι κανένας από τον κώδικα δεν δοκιμάζεται με unit tests. Αυτό θα μπορούσε να δυσκολέψει την εύρεση και την επιδιόρθωση σφαλμάτων στο μέλλον.

Συνολικά, ο κώδικας φαίνεται να είναι λειτουργικός και να πληροί τα βασικά πρότυπα ποιότητας. Ωστόσο, υπάρχουν ορισμένοι τομείς για βελτίωση, όπως η μείωση του αριθμού των σφαλμάτων και των κακών πρακτικών κώδικα, καθώς και η αύξηση της κάλυψης κώδικα.

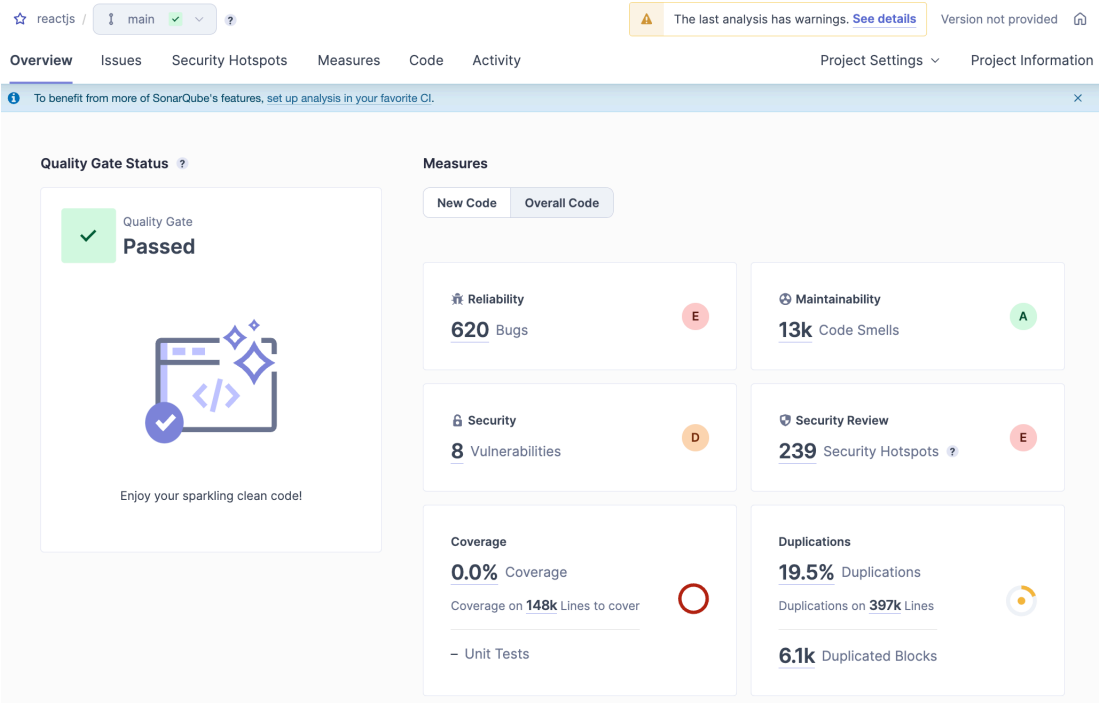
Ποιότητα Κώδικα React

Σύμφωνα με τα αποτελέσματα του SonarQube, η ποιότητα του κώδικα React φαίνεται να είναι καλή. Η Πύλη Ποιότητας πέρασε, πράγμα που σημαίνει ότι δεν εντοπίστηκαν

κρίσιμα ζητήματα. Ωστόσο, υπάρχουν ορισμένοι τομείς για βελτίωση, όπως υποδεικνύουν οι προειδοποιήσεις.

Ένας τομέας προς βελτίωση είναι η συντήρηση. Η βαθμολογία συντήρησης βαθμολογείται με "E" που θεωρείται "χαμηλή". Αυτό μπορεί να οφείλεται σε παράγοντες όπως η υψηλή πολυπλοκότητα του κώδικα ή η έλλειψη σχολίων. Ένας άλλος τομέας προς βελτίωση είναι η επανάληψη κώδικα. Η επανάληψη κώδικα είναι στο 19,5%, η οποία είναι αρκετά υψηλή. Αυτό σημαίνει ότι ένα σημαντικό τμήμα του κώδικα επαναλαμβάνεται, γεγονός που μπορεί να δυσκολέψει τη συντήρηση και την ενημέρωση του κώδικα.

Συνολικά, ο κώδικας React φαίνεται να είναι λειτουργικός χωρίς κρίσιμα ζητήματα, αλλά μπορεί να βελτιωθεί όσον αφορά τη συντήρηση και την επανάληψη κώδικα.



Σχήμα 4.14: Αποτελέσματα γενικής ποιότητας κώδικα για το πλαίσιο React

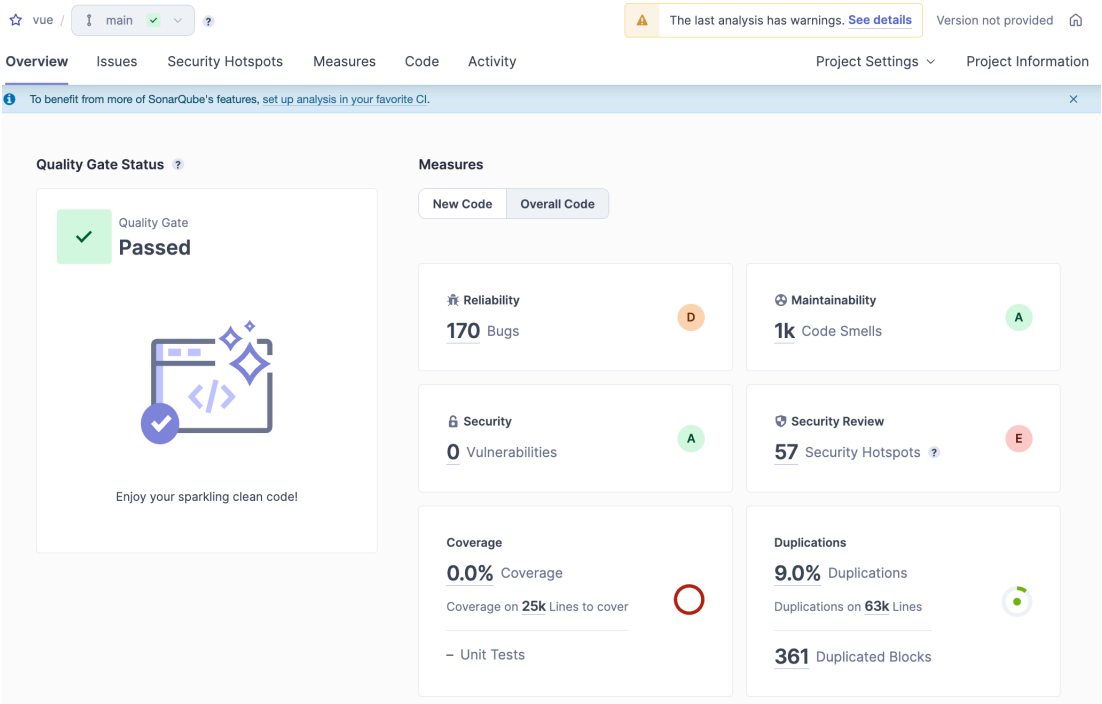
Ποιότητα Κώδικα Vue

Με βάση τον πίνακα εργαλείων SonarQube, η ποιότητα του κώδικα Vue φαίνεται να είναι αρκετά καλή. Η Πύλη Ποιότητας έχει περάσει, υποδεικνύοντας ότι ο κώδικας πληροί τα διαμορφωμένα πρότυπα ποιότητας. Στο συγκεκριμένο πλαίσιο εντοπίστηκαν 170 σφάλματα, ενώ και 1.000 κακές πρακτικές κώδικα. Ενώ αυτοί οι αριθμοί μπορεί να φαίνονται υψηλοί, είναι σημαντικό να λάβετε υπόψη το μέγεθος του έργου. Το SonarQube δεν παρέχει βαθμολογία σοβαρότητας για σφάλματα ή κακές πρακτικές κώδικα, επομένως είναι δύσκολο να αξιολογηθεί η κρισιμότητα αυτών των ευρημάτων χωρίς περαιτέρω ανάλυση.

Ένας τομέας προς βελτίωση είναι η κάλυψη κώδικα. Ο πίνακας εργαλείων δείχνει

κάλυψη 0,0%, υποδεικνύοντας ότι κανένας από τον κώδικα δεν καλύπτεται από μοναδιαίες δοκιμές. Εφόσον οι μοναδιαίες δοκιμές βοηθούν στην ανίχνευση σφαλμάτων και οπισθοδρομήσεων κατά την ανάπτυξη, η αύξηση της κάλυψης των δοκιμών είναι μια συνιστώμενη βελτίωση.

Συνολικά, τα αποτελέσματα του SonarQube υποδηλώνουν ότι ο κώδικας του Vue.js έχει μια καλή βάση ποιότητας. Η αντιμετώπιση των κακών πρακτικών κώδικα και η εφαρμογή μοναδιαίων δοκιμών θα βελτίωνε περαιτέρω τη συντήρηση και την αξιοπιστία του κώδικα.



Σχήμα 4.15: Αποτελέσματα γενικής ποιότητας κώδικα για το πλαίσιο Vue

Ποιότητα Κώδικα Svelte

Βασισμένο στον πίνακα εργαλείων SonarQube, όπως εμφανίζεται και στην Εικόνα 4.16, η ποιότητα του κώδικα Svelte φαίνεται γενικά καλή, αλλά υπάρχει περιθώριο βελτίωσης σε ορισμένους τομείς. Το Quality Gate έχει περάσει, υποδεικνύοντας ότι ο κώδικας πληροί τα διαμορφωμένα πρότυπα ποιότητας.

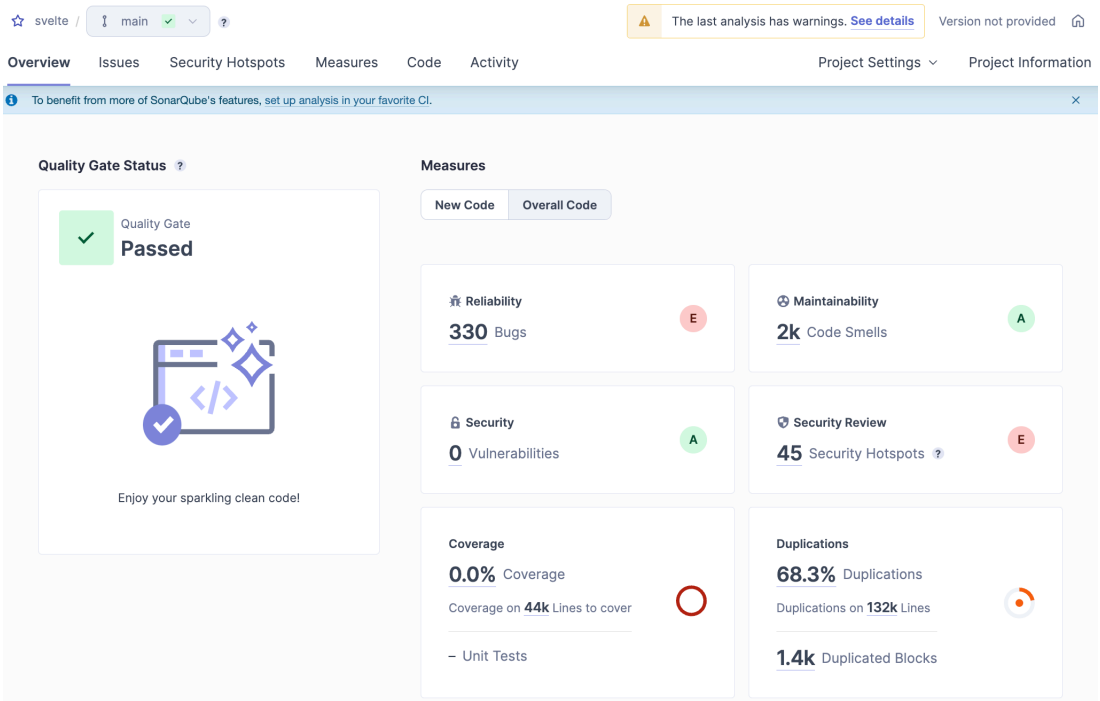
Ακολουθούν μερικοί από τους δείκτες που εμφανίζονται στον πίνακα εργαλείων:

- Σφάλματα: Έχουν εντοπιστεί 330 σφάλματα.
- Οσμές Κώδικα: Βρέθηκαν 2.000 οσμές κώδικα.
- Διπλοτυπία: Ο βαθμός διπλοτυπίας του κώδικα είναι 68,3%.
- Κάλυψη: Η κάλυψη κώδικα είναι στο 0,0

Παρόλο που η Πύλη Ποιότητας πέρασε, υπάρχει ένας σημαντικός αριθμός σφαλμάτων και κακών πρακτικών κώδικα που θα μπορούσαν δυνητικά να επηρεάσουν τη λειτουργικότητα και τη συντήρηση της βάσης κώδικα. Το υψηλό ποσοστό διπλοτυπίας (68,3%) υποδηλώνει ότι υπάρχουν επαναλαμβανόμενα μπλοκ κώδικα σε όλο τον κώδικα του πλαισίου. Αυτό μπορεί να δυσκολέψει την κατανόηση και τη συντήρηση του κώδικα και επίσης να αυξήσει τον κίνδυνο εισαγωγής σφαλμάτων εάν χρειαστεί να γίνει αλλαγή σε ένα σημείο.

Η έλλειψη κάλυψης κώδικα (0,0%) αποτελεί επίσης ανησυχία. Οι δοκιμές μονάδας μπορούν να βοηθήσουν στον εντοπισμό σφαλμάτων και οπισθοδρομήσεων που μπορεί να εισαχθούν κατά τη διάρκεια της ανάπτυξης. Χωρίς δοκιμές μονάδας, μπορεί να είναι πιο δύσκολο να διασφαλιστεί ότι ο κώδικας θα συνεχίσει να λειτουργεί σωστά καθώς γίνονται αλλαγές.

Συνολικά, τα αποτελέσματα του SonarQube υποδηλώνουν ότι ο κώδικας Svelte έχει ορισμένους τομείς προς βελτίωση, αλλά γενικά φαίνεται να είναι μια καλά λειτουργούσα βάση κώδικα. Η εστίαση στη μείωση των σφαλμάτων, των κακών πρακτικών κώδικα και της διπλοτυπίας κώδικα, καθώς και η εφαρμογή δοκιμών μονάδας θα βελτιώσουν περαιτέρω την ποιότητα και τη συντήρηση του έργου Svelte.



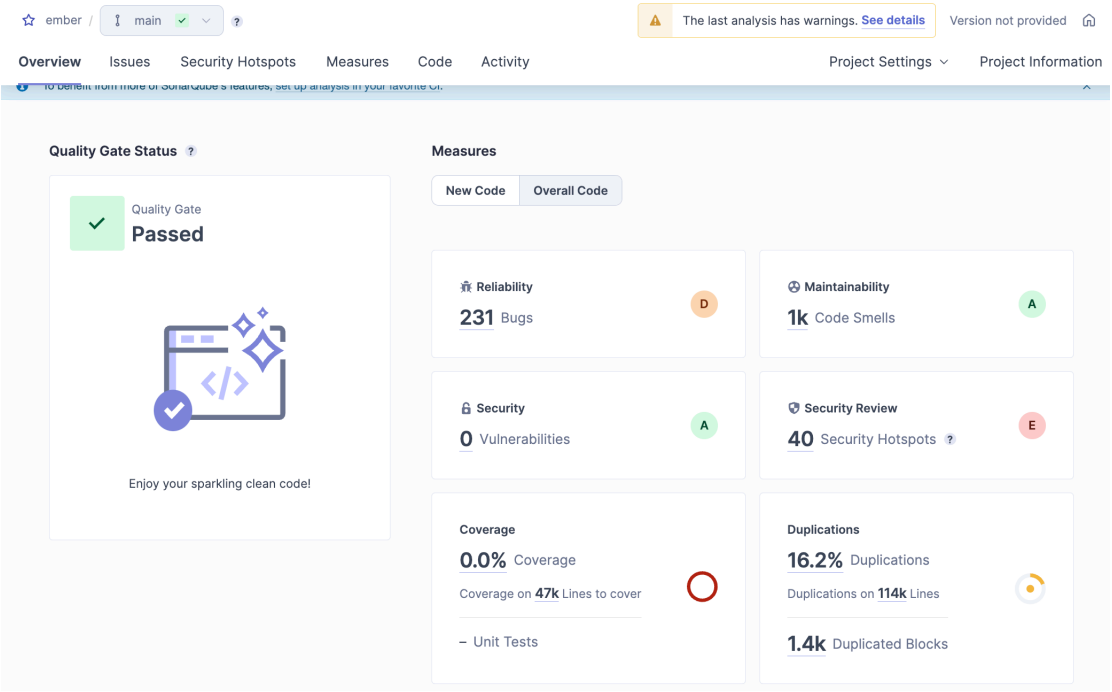
Σχήμα 4.16: Αποτελέσματα γενικής ποιότητας κώδικα για το πλαίσιο Svelte

Ποιότητα Κώδικα Ember

Η ποιότητα του κώδικα του Ember φαίνεται να είναι καλή, αλλά υπάρχουν κάποια περιθώρια βελτίωσης. Ακολουθεί μια ανάλυση των αποτελεσμάτων:

- **Πύλη Ποιότητας:** Το πλαίσιο πέρασε την Πύλη ποιότητας, συνεπώς η ποιότητα του κώδικα είναι καλή.
- **Αξιοπιστία:** Έχουν εντοπιστεί 231 σφάλματα.
- **Διατηρησιμότητα:** Εντοπίστηκαν 1.000 οσμές κώδικα.
- **Διπλοτυπία:** Ο βαθμός διπλοτυπίας του κώδικα είναι 16,2%.
- **Κάλυψη:** Η κάλυψη κώδικα είναι 0,0%, πράγμα που σημαίνει ότι κανένας κώδικας δεν καλύπτεται από μοναδιαίες δοκιμές.

Συνολικά, ο κώδικας Ember φαίνεται να είναι καλά συντηρημένος με χαμηλό ποσοστό διπλοτυπίας. Ωστόσο, υπάρχουν ακόμα σφάλματα που πρέπει να διορθωθούν και πλήρη έλλειψη κάλυψης δοκιμών μονάδας, κάτι που θα μπορούσε να υποδηλώσει πιθανά προβλήματα στο μέλλον. Η εστίαση στη βελτίωση της κάλυψης κώδικα και την επίλυση των προειδοποιήσεων πιθανότατα θα βελτιώσει τη συνολική ποιότητα του κώδικα.



Σχήμα 4.17: Αποτελέσματα γενικής ποιότητας κώδικα για το πλαίσιο Svelte

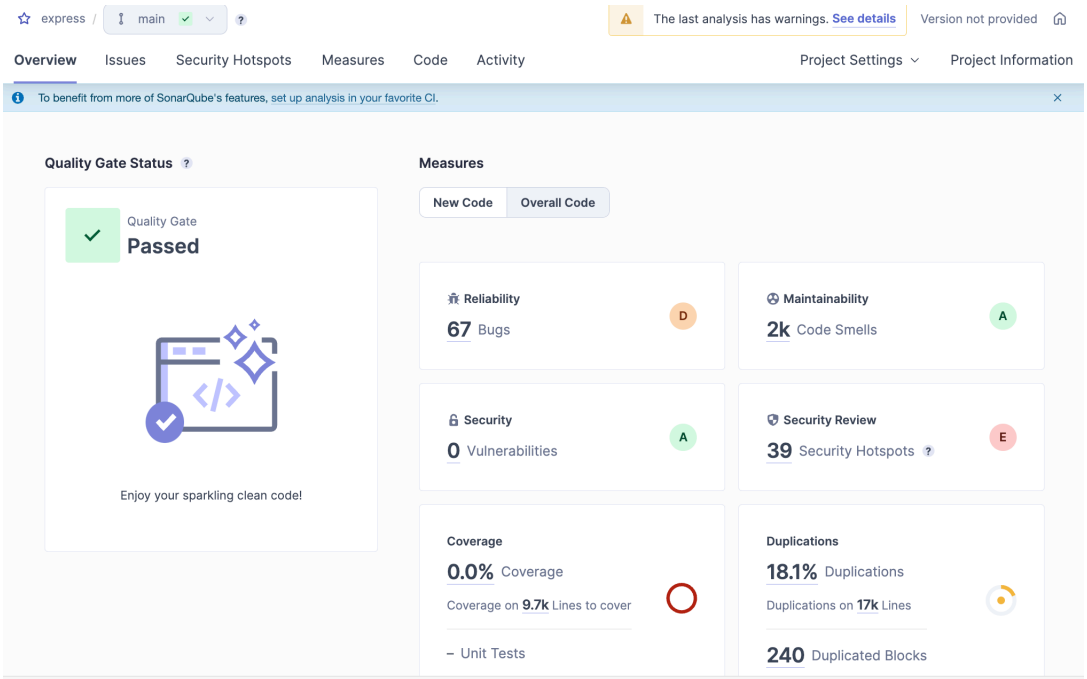
4.3.2.2 Σύγκριση Ποιότητας Κώδικα Back-End Πλαισίων

Ποιότητα Κώδικα Express

Η γενική ποιότητα του κώδικα φαίνεται να είναι καλή. Τα αποτελέσματα απεικονίζονται στην Εικόνα 4.18. Ακολουθεί μια ανάλυση βασισμένη στον πίνακα εργαλείων του SonarQube στην εικόνα:

- Σφάλματα: Έχουν εντοπιστεί 67 σφάλματα. Αυτό βαθμολογείται με D, το οποίο δεν είναι ιδανικό, αλλά αποδεκτό.
- Κακές Πρακτικές Κώδικα: Υπάρχουν 2.000 κακές πρακτικές κώδικα, κάτι που βαθμολογείται με A, δηλαδή πολύ καλό. Οι οσμές κώδικα είναι αδυναμίες στον κώδικα που δεν προκαλούν σφάλματα αλλά μπορούν να δυσκολέψουν τη συντήρηση του κώδικα στο μέλλον. Επιθυμητός είναι ένας χαμηλός αριθμός.
- Επανάληψη Κώδικα: Υπάρχει 18,1% επανάληψη στον κώδικα. Αυτό είναι αιτία ανησυχίας. Ο υψηλός βαθμός επανάληψης σημαίνει ότι υπάρχει επαναλαμβανόμενος κώδικας, ο οποίος μπορεί να δυσκολέψει τη συντήρηση του κώδικα και να τον κάνει πιο επιρρεπή σε σφάλματα. Επίσης, μπορεί να αυξήσει το μέγεθος της βάσης του κώδικα.

Συνολικά, ο κώδικας φαίνεται να είναι λειτουργικός με χαμηλό κίνδυνο ευπαθειών ασφαλείας, αλλά υπάρχουν περιθώρια βελτίωσης όσον αφορά τα σφάλματα και την επανάληψη του κώδικα.



Σχήμα 4.18: Αποτελέσματα γενικής ποιότητας κώδικα για το πλαίσιο Express

Ποιότητα Κώδικα Hono

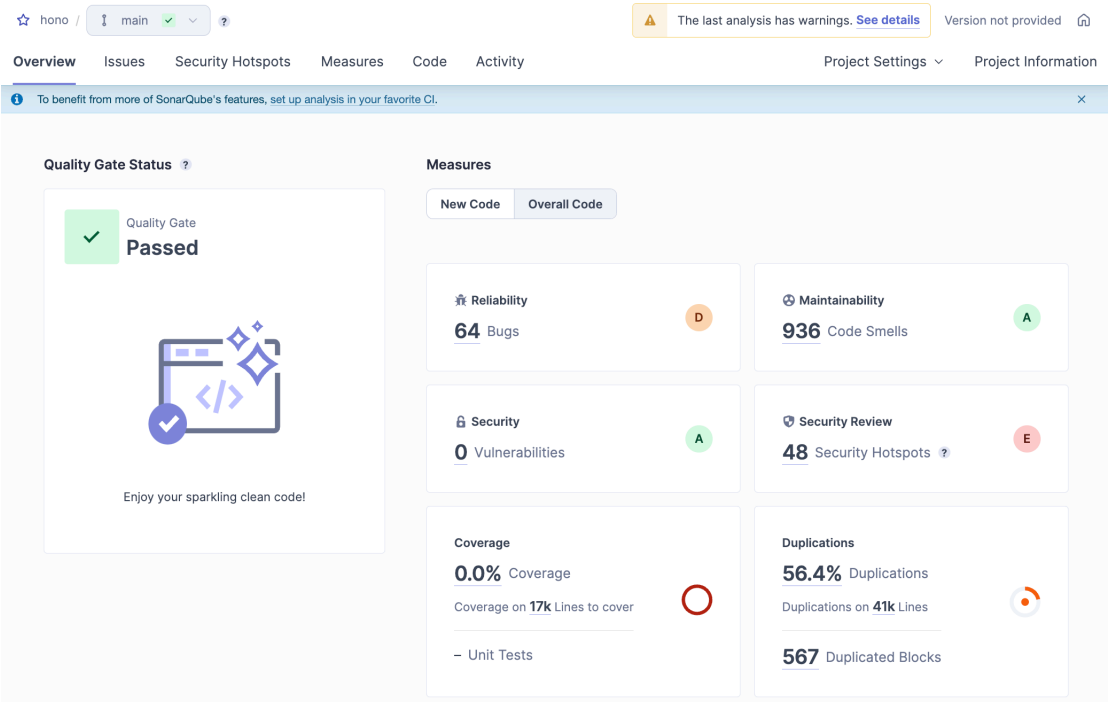
Τα αποτελέσματα του πλαισίου Hono παρουσιάζονται στην Εικόνα 4.19 και αναλύονται παρακάτω:

- Αξιοπιστία: Έχουν εντοπιστεί 64 σφάλματα. Αυτό βαθμολογείται με D από το SonarQube, το οποίο δεν είναι ιδανικό αλλά εξακολουθεί να είναι αποδεκτό.

- Διατηρησιμότητα: Έχουν εντοπιστεί 936 κακές πρακτικές κώδικα, ωστόσο η κατηγορία αυτή βαθμολογείται με Α, άρα η διατηρησιμότητα θεωρείται πολύ καλή. Όπως έχουμε προαναφέρει, οι κακές πρακτικές κώδικα είναι αδυναμίες που μπορούν να δυσκολέψουν τη μελλοντική συντήρηση του κώδικα, και συνεπώς επιθυμητό είναι ο αριθμός κακών πρακτικών κώδικα να είναι όσο χαμηλότερος γίνεται.

- Επαναλήψεις Κώδικα: Αυτός είναι ένας σημαντικός τομέας για βελτίωση. Ο κώδικας έχει 56,4% επαναληψιμότητα, η οποία είναι αρκετά υψηλή. Αυτό σημαίνει ότι υπάρχει πολύς επαναλαμβανόμενος κώδικας που μπορεί να είναι δύσκολο στη συντήρηση και μπορεί επίσης να αυξήσει το μέγεθος της βάσης του κώδικα.

Συνολικά, ο κώδικας φαίνεται λειτουργικός, αλλά υπάρχει περιθώριο βελτίωσης για να γίνει πιο διατηρήσιμος. Η αναδιοργάνωση για τη μείωση των επαναλήψεων θα ήταν ένα καλό επόμενο βήμα.



Σχήμα 4.19: Αποτελέσματα γενικής ποιότητας κώδικα για το πλαίσιο Hono

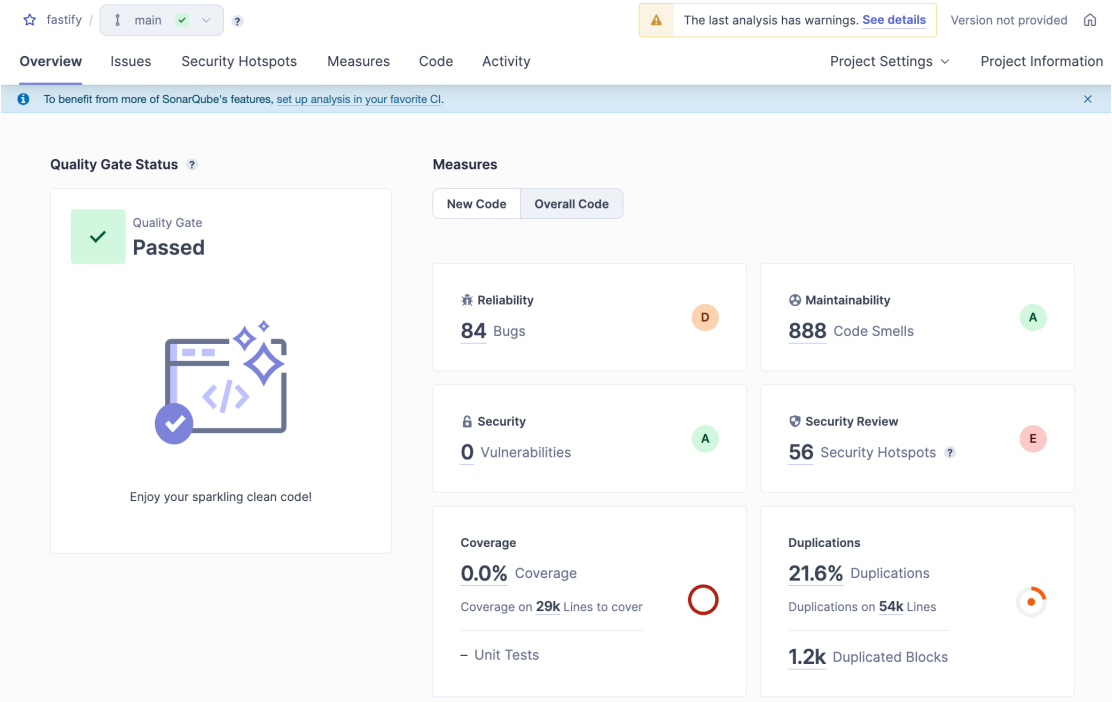
Ποιότητα Κώδικα Fastify

Βασισμένο στον πίνακα εργαλείων του SonarQube στην Εικόνα 4.19, η ποιότητα του κώδικα Fastify φαίνεται πολύ καλή. Ακολουθεί μια ανάλυση των σχετικών μετρήσεων:

- Σφάλματα: Έχουν εντοπιστεί 84 σφάλματα. Αυτό βαθμολογείται με D από το SonarQube, το οποίο δεν είναι ιδανικό αλλά εξακολουθεί να είναι αποδεκτό.

- Κακές Πρακτικές Κώδικα: Υπάρχουν μόνο 888 κακές πρακτικές κώδικα, οι οποίες βαθμολογούνται με Α, που είναι πολύ καλό.
- Επαναλήψεις Κώδικα: Η επανάληψη του κώδικα είναι 21,6%, γεγονός που αποτελεί αιτία ανησυχίας. Ο υψηλός βαθμός επανάληψης σημαίνει ότι υπάρχει επαναλαμβανόμενος κώδικας, ο οποίος μπορεί να δυσκολέψει τη συντήρηση του κώδικα και να τον κάνει πιο επιρρεπή σε σφάλματα.

Συνολικά, ο κώδικας Fastify φαίνεται να είναι λειτουργικός με χαμηλό κίνδυνο ευπαθειών ασφαλείας, αλλά υπάρχει περιθώριο βελτίωσης για τη μείωση της επανάληψης του κώδικα.



Σχήμα 4.20: Αποτελέσματα γενικής ποιότητας κώδικα για το πλαίσιο Fastify

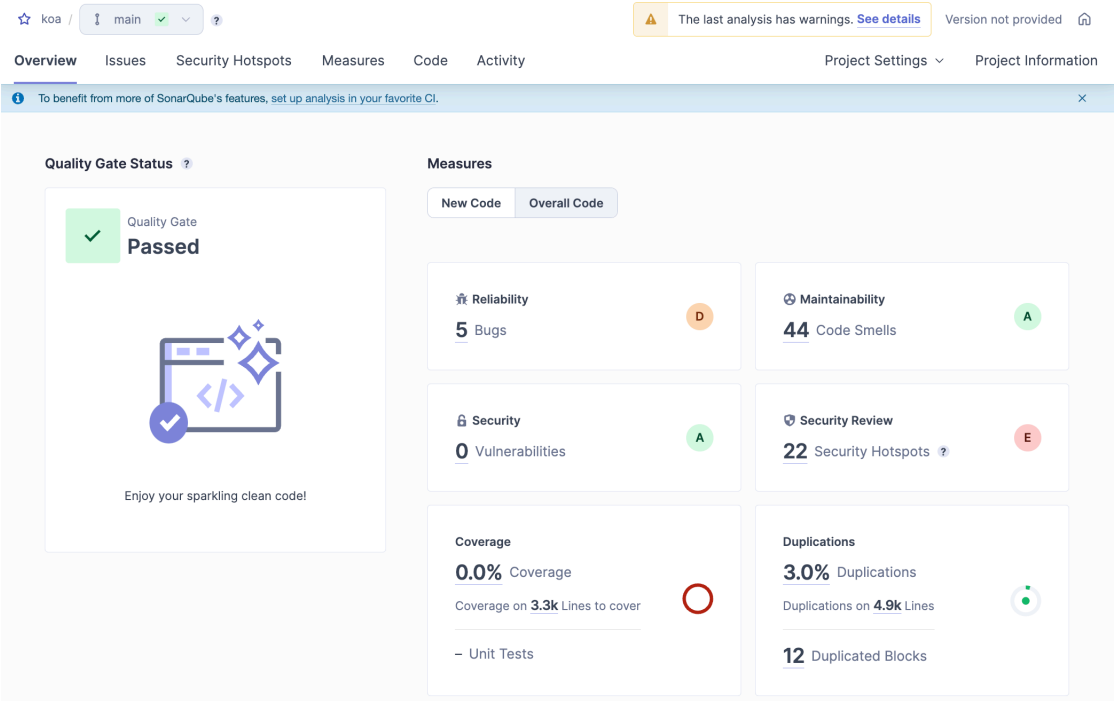
Ποιότητα Κώδικα Koa

Βασισμένο στον πίνακα εργαλείων του SonarQube για το έργο Koa στην Εικόνα 4.21, η ποιότητα του κώδικα φαίνεται γενικά καλή. Ακολουθεί μια ανάλυση ορισμένων μετρήσεων:

- Σφάλματα: Έχουν εντοπιστεί 5 σφάλματα. Αυτός είναι ένας πολύ χαμηλός αριθμός και θα θεωρούνταν καλός.
- Κακές Πρακτικές Κώδικα: Υπάρχουν 44 κακές πρακτικές κώδικα, οι οποίες βαθμολογούνται με D. Ο βαθμός αυτός δεν είναι ο ιδανικός, αλλά εξακολουθεί να είναι σε αποδεκτά επίπεδα.

- Επαναλήψεις Κώδικα: Η επανάληψη του κώδικα κυμαίνεται στο 3%, το οποίο αποτελεί πολύ καλό αποτέλεσμα όσον αφορά τον επαναλβανόμενο κώδικα και την συντηρησιμότητα του.

Συνολικά, ο κώδικας Koa φαίνεται να είναι καλογραμμένος με χαμηλό κίνδυνο σφαλμάτων και ελάχιστη επανάληψη κώδικα. Ωστόσο, υπάρχει κάποιο περιθώριο βελτίωσης όσον αφορά τις οσμές κώδικα.



Σχήμα 4.21: Αποτελέσματα γενικής ποιότητας κώδικα για το πλαίσιο Koa

4.3.2.3 Σύγκριση Ποιότητας Κώδικα Full-Stack Πλαισίων

Ποιότητα Κώδικα Next

Βασισμένο στον πίνακα εργαλείων του SonarQube για το έργο Next.js, όπως φαίνεται και στην Εικόνα 4.22, η ποιότητα του κώδικα φαίνεται γενικά καλή. Ακολουθεί μια ανάλυση ορισμένων μετρήσεων:

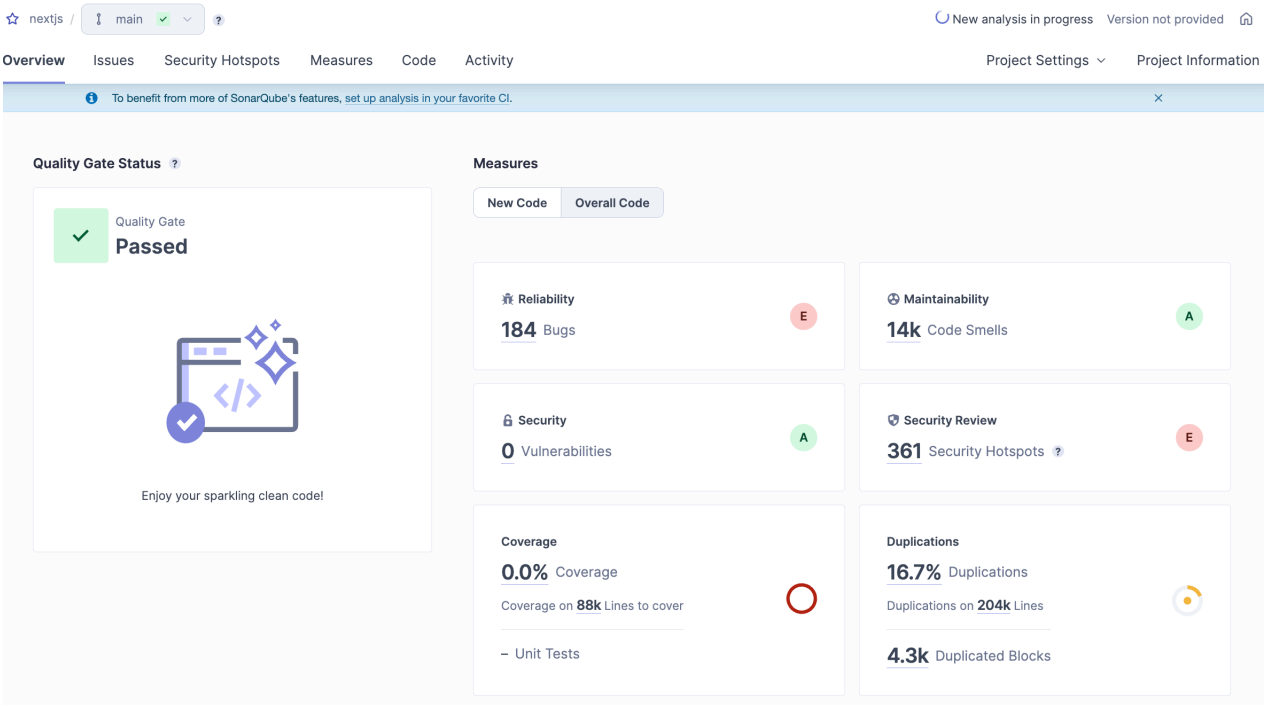
- Συνολική Πύλη Ποιότητας: Η Πύλη Ποιότητας έχει περάσει, πράγμα που σημαίνει ότι όλες οι μετρήσεις πληρούν τα διαμορφωμένα όρια. Σφάλματα: Έχουν εντοπιστεί 184 σφάλματα. Αυτό βαθμολογείται με Ε από το SonarQube, το οποίο δεν είναι ιδανικό αλλά εξακολουθεί να είναι αποδεκτό.

- Κακές Πρακτικές Κώδικα: Υπάρχουν 14.000 κακές πρακτικές κώδικα, οι οποίες βαθμολογούνται με Ε. Ενώ ο βαθμός είναι σε χαμηλά επίπεδα, δεν σημαίνει απαραίτητα ότι ο κώδικας είναι κακός. Αυτή η μέτρηση αναζητά πρακτικές κωδικοποίησης που ενδεχομένως να δυσκολέψουν τη συντήρηση του κώδικα στο μέλλον

και η βελτίωση της μέτρησης μπορεί να επιτευχθεί με κάποια αναδιοργάνωση του κώδικα (refactoring).

- Επαναλήψεις Κώδικα: Η επανάληψη του κώδικα κυμαίνεται στο 16,7%, που είναι ένα μέτριο ποσοστό όσον αφορά τον επαναλαμβανόμενο κώδικα. Αυτό σημαίνει ότι η συντήρηση του κώδικα είναι σχετικά δύσκολη.

Συνολικά, ο κώδικας Next φαίνεται να είναι καλογραμμένος με χαμηλό κίνδυνο σφαλμάτων και ελάχιστη επανάληψη κώδικα. Ωστόσο, υπάρχει κάποιο περιθώριο βελτίωσης όσον αφορά τις οσμές κώδικα.



Σχήμα 4.22: Αποτελέσματα γενικής ποιότητας κώδικα για το πλαίσιο Next.js

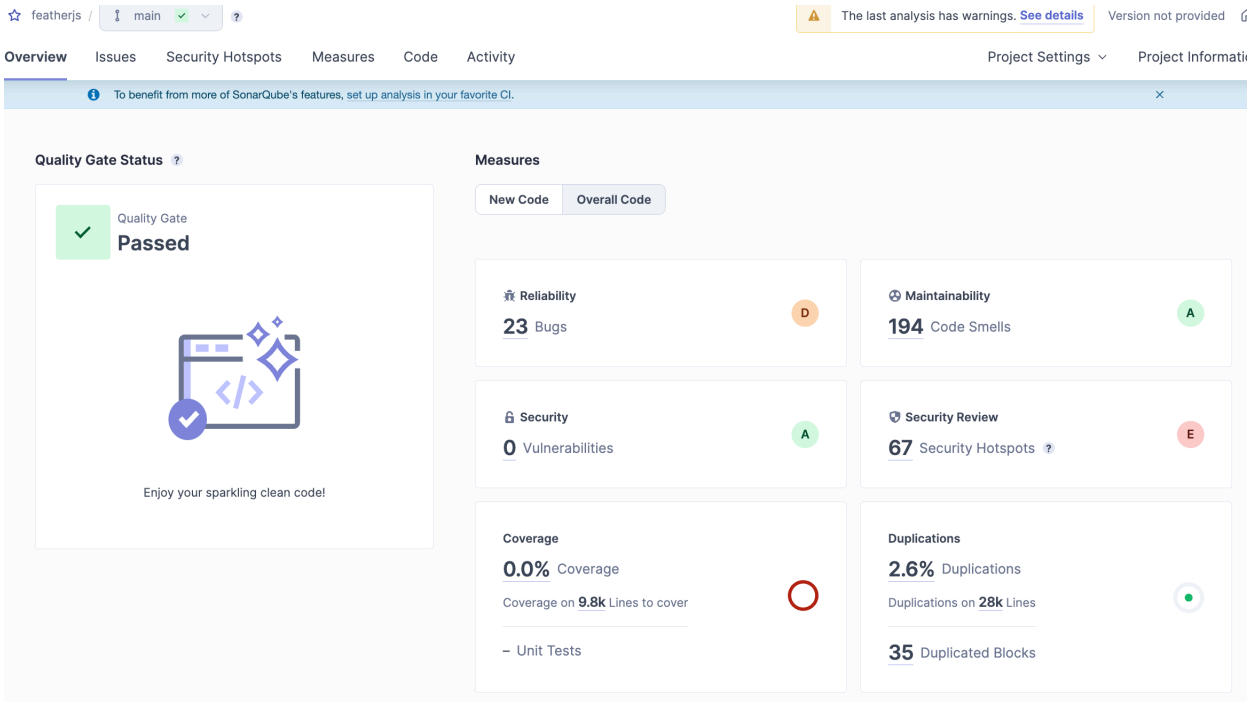
Ποιότητα Κώδικα Feathers

Βασισμένο στον πίνακα εργαλείων του SonarQube για το έργο Feather, όπως φαίνεται και στην Εικόνα 4.23 η ποιότητα του κώδικα φαίνεται γενικά καλή. Ακολουθεί μια ανάλυση ορισμένων μετρήσεων:

- Σφάλματα: Έχουν εντοπιστεί 23 σφάλματα. Αυτό βαθμολογείται με D από το SonarQube, το οποίο δεν είναι ιδανικό, αλλά εξακολουθεί να είναι αποδεκτό.
- Κακές Πρακτικές Κώδικα: Υπάρχουν 194 κακές πρακτικές κώδικα, οι οποίες βαθμολογούνται με C, που είναι αποδεκτό. Θυμηθείτε ότι οι οσμές κώδικα είναι αδυναμίες στον κώδικα που μπορούν να δυσκολέψουν τη μελλοντική συντήρηση του, επομένως επιθυμητός είναι ένας χαμηλότερος αριθμός.

- Επαναλήψεις Κώδικα: Η επανάληψη του κώδικα είναι 2,6%, που είναι πολύ καλό αποτέλεσμα. Αυτό σημαίνει ότι υπάρχει πολύ λίγος επαναλαμβανόμενος κώδικας, καθιστώντας τον κώδικα πιο εύκολο στη συντήρηση και λιγότερο επιρρεπή σε σφάλματα.

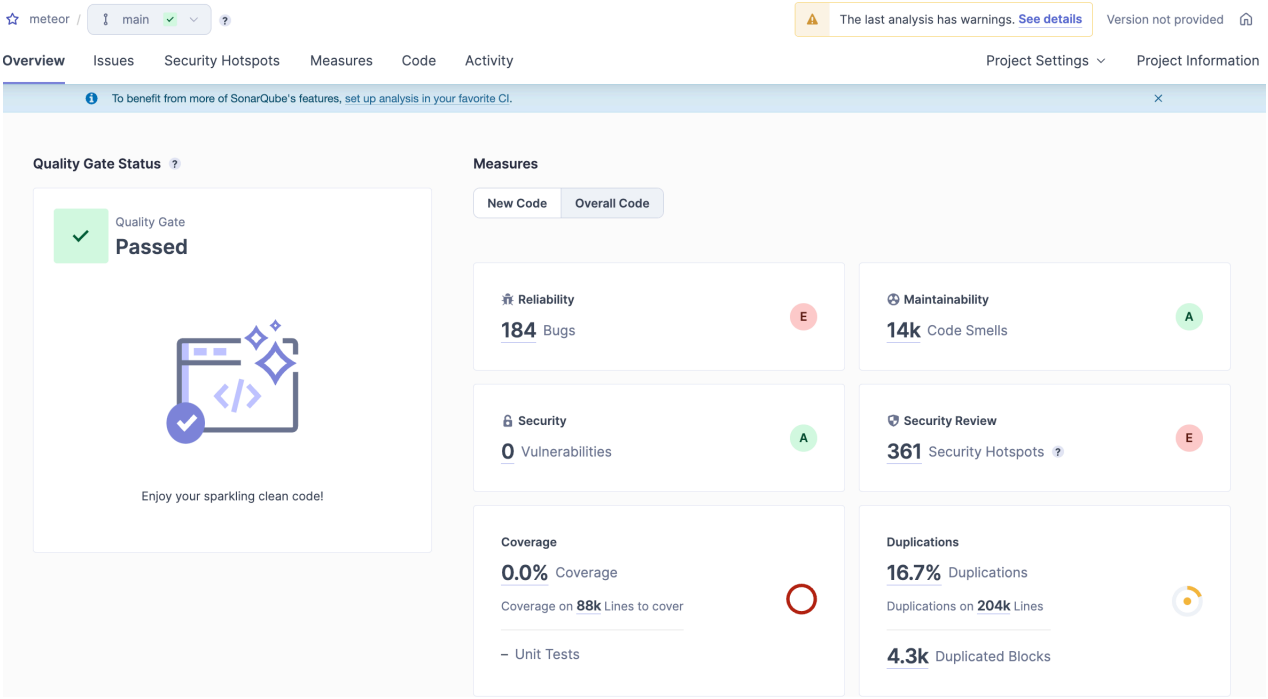
Συνολικά, ο κώδικας Feather φαίνεται να είναι καλογραμμένος με χαμηλό κίνδυνο σφαλμάτων και ελάχιστη επανάληψη κώδικα. Υπάρχει κάποιο περιθώριο βελτίωσης όσον αφορά τις οσμές κώδικα, αλλά γενικά η ποιότητα φαίνεται καλή.



Σχήμα 4.23: Αποτελέσματα γενικής ποιότητας κώδικα για το πλαίσιο Feathers.js

Ποιότητα Κώδικα Meteor

Η ποιότητα του κώδικα για το έργο Meteor φαίνεται πολύ καλή βάσει των αποτελεσμάτων του SonarQube, όπως παρουσιάζεται και στην Εικόνα 4.24. Εμφανίζεται να έχει περάσει τον γενικό έλεγχο ποιότητας. Ενώ ο ακριβής αριθμός σφαλμάτων δεν είναι διαθέσιμος, υπάρχουν προειδοποιήσεις που υποδεικνύουν ορισμένα πιθανά προβλήματα. Οι οσμές κώδικα είναι χαμηλές, πράγμα που αποτελεί θετικό σημάδι για τη συντηρησιμότητα. Υπάρχει κάποιο περιθώριο βελτίωσης μειώνοντας τη διπλοτυπία του κώδικα (που επί του παρόντος είναι 16,7%). Συνολικά, ο κώδικας Meteor φαίνεται λειτουργικός και καλά δομημένος.



Σχήμα 4.24: Αποτελέσματα γενικής ποιότητας κώδικα για το πλαίσιο Meteor.js

Κεφάλαιο 5

Συμπεράσματα και μελλοντική εργασία

Η παρούσα διατριβή αποτέλεσε μια ολοκληρωμένη εξερεύνηση της βαθιάς επίδρασης της JavaScript στην ανάπτυξη ιστοσελίδων και του μεγάλου εύρους πλαισίων που έχουν προκύψει για να διαμορφώσουν το σύγχρονο τοπίο. Αναλύσαμε τις ποικίλες λειτουργίες αυτών των πλαισίων, κατηγοριοποιώντας τα με βάση την αρχιτεκτονική, τη σύνδεση δεδομένων και άλλους σχετικούς παράγοντες. Επιπλέον, διεξαγάγαμε συγκριτική ανάλυση, εξετάζοντας τη δημοτικότητά τους, την υποστήριξη της κοινότητας και κρίσιμες πτυχές όπως η ασφάλεια και η ποιότητα του κώδικα. Αυτή η συγκριτική ανάλυση εξοπλίζει τους προγραμματιστές με τις απαραίτητες γνώσεις και δεξιότητες κριτικής σκέψης για να λαμβάνουν ενημερωμένες αποφάσεις κατά την επιλογή του πιο κατάλληλου πλαισίου για τις συγκεκριμένες απαιτήσεις και το πλαίσιο του έργου τους.

Η κατηγοριοποίηση των διαφόρων πλαισίων JavaScript, μέσω του διαχωρισμού τους σε διάφορες ομάδες βάσει συγκεκριμένων χαρακτηριστικών, μπορεί να βοηθήσει τους προγραμματιστές να αποφασίσουν ποιο είναι το καταλληλότερο πλαίσιο για την υλοποίηση της εφαρμογής τους. Δεν υπάρχει πλαίσιο που να θεωρείται καλύτερο από τα υπόλοιπα. Ωστόσο, κάθε πλαίσιο είναι πιο κατάλληλο ανάλογα με την περίπτωση και τις ανάγκες υλοποίησης. Η επιλογή του σωστού πλαισίου εξαρτάται από παράγοντες όπως το μέγεθος του έργου, οι απαιτήσεις απόδοσης, η μεταφερσιμότητα, η ευκολία μάθησης και η ανάπτυξη εφαρμογών.

Όσον αφορά τη σύγκριση των πλαισίων, ορισμένα πλαίσια JavaScript φαίνεται να συγκεντρώνουν υψηλότερη βαθμολογία στα συστήματα αξιολόγησης σε σχέση με άλλα. Αυτό οφείλεται κυρίως στη μεγαλύτερη διάδοσή τους και την ευρύτερη υιοθέτησή τους από την κοινότητα των προγραμματιστών. Συγκεκριμένα, τα πλαίσια React.js και Next.js θεωρούνται από τα πλέον δημοφιλή τα τελευταία χρόνια, σύμφωνα με τα στατιστικά στοιχεία λήψεων από το αποθετήριο GitHub και το σύστημα διαχείρισης πακέτων λογισμικού npm.

Η ευρεία υιοθέτηση ενός πλαισίου από την κοινότητα των προγραμματιστών αποτελεί σημαντικό παράγοντα για την περαιτέρω ανάπτυξη και βελτίωσή του. Όσο περισσότεροι προγραμματιστές χρησιμοποιούν ένα συγκεκριμένο πλαίσιο, τόσο περισσότερη υποστή-

ριξη και συνεισφορά λαμβάνει από την κοινότητα, γεγονός που συμβάλλει στη συνεχή ενημέρωση, βελτιστοποίηση και επέκτασή του. Επιπλέον, η ευρεία διάδοση ενός πλαισίου διευκολύνει την εύρεση πόρων, τεκμηρίωσης και υποστήριξης από την κοινότητα, καθιστώντας την εκμάθηση και χρήση του πιο προσιτή.

Μία ακόμη σύγκριση που διενεργήσαμε στα επιλεγμένα πλαίσια JavaScript αφορούσε την ασφάλεια των πλαισίων. Για να επιβεβαιώσουμε ότι τα πιο διαδεδομένα πλαίσια της JavaScript είναι αρκετά ασφαλή και δεν εμφανίζουν σημαντικά κενά ασφαλείας που θα έθεταν την εφαρμογή προς υλοποίηση σε κίνδυνο, εκτελέσαμε δύο διαφορετικές αξιολογήσεις στα πλαίσια, μέσω του εργαλείου SonarQube, το οποίο είναι ένα από τα πιο διαδεδομένα εργαλεία ανάλυσης στατικού κώδικα. Η πρώτη αξιολόγηση αφορούσε την ασφάλεια των πλαισίων, ενώ η δεύτερη την ποιότητα του κώδικά τους. Στην πρώτη αξιολόγηση ασφάλειας, δεν εντοπίστηκαν σημαντικά ζητήματα ασφαλείας που θα μπορούσαν να θέσουν σε κίνδυνο τη λειτουργία των πλαισίων, παρά μόνο κάποιες ευπάθειες που αφορούσαν τα αρχεία δοκιμών. Ακόμη, η δεύτερη αξιολόγηση αφορούσε την ποιότητα του κώδικα. Όσον αφορά αυτό το μέρος, σε γενικές γραμμές ο κώδικας των πλαισίων ήταν σε αρκετά υψηλά επίπεδα. Εντούτοις, εντοπίστηκαν τόσο αρκετές επαναλήψεις κώδικα όσο και κακές πρακτικές κώδικα, οι οποίες συνιστούν βελτίωση για την βέλτιστη συντήρηση του κώδικα.

Η εμφάνιση νέων πλαισίων και η συνεχής εξέλιξη των υπαρχόντων απαιτεί συνεχή εξερεύνηση και έρευνα για να παραμένουμε ενημερωμένοι με τις τελευταίες τάσεις και τις βέλτιστες πρακτικές. Με την ενεργή συμμετοχή στην ζωντανή και δυναμική κοινότητα της JavaScript και την παρακολούθηση αυτών των εξελίξεων, οι προγραμματιστές μπορούν να διασφαλίσουν ότι είναι καλά εξοπλισμένοι για να αντιμετωπίσουν τις προκλήσεις και τις ευκαιρίες που τους περιμένουν, διαμορφώνοντας το μέλλον της ανάπτυξης ιστοσελίδων με αυτά τα ισχυρά εργαλεία στη διάθεσή τους.

Bibliography

[1] 8 advantages and disadvantages of javascript explained // unstop. https://unstop.com/blog/advantages-and-disadvantages-of-javascript. (Accessed on 02-28-2024).

[2] Netscape. https://en.wikipedia.org/wiki/Netscape. (Accessed on 02-28-2024).

[3] Sun microsystems. https://en.wikipedia.org/wiki/SunMicrosystems. (Accessed on 02-28-2024).

[4] History of javascript. https://softteco.com/blog/history-of-javascript. (Accessed on 02-28-2024).

[5] Judi Toledo. Why Millions of Developers use JavaScript for Web Application Development? — torquemag.io. https://torquemag.io/2018/06/why-millions-of-developers-use-javascript-for-web- [Accessed 28-02-2024].

[6] ACM Digital Library — dl.acm.org. https://dl.acm.org/. [Accessed 28-02-2024].

[7] Ethan Brown. *Web development with node and express: leveraging the JavaScript stack*. O'Reilly Media, 2019.

[8] Olayinka Omole. *Server Side development with Node. js and Koa. js Quick Start Guide: Build robust and scalable web applications with modern JavaScript techniques*. Packt Publishing Ltd, 2018.

[9] Fundamentals of Fastify: The Best Node.js Framework — radixweb.com. https://radixweb.com/blog/introducing-fastify-nodejs-framework. [Accessed 28-02-2024].

[10] GitHub - honojs/hono: Fast, Lightweight, Web-standards — github.com. https://github.com/honojs/hono. [Accessed 28-02-2024].

[11] What Is Meteor? — builtin.com. https://builtin.com/software-engineering-perspectives/ [Accessed 28-02-2024].

[12] daffl, marshallswain, and FeathersJS contributors. feathersjs. https://feathersjs.com/. [Accessed 28-02-2024].

[13] AngularJS — docs.angularjs.org. https://docs.angularjs.org/guide/introduction. [Accessed 28-02-2024].

[14] Konrad Bielak, Bartłomiej Borek, and Małgorzata Plechawska-Wójcik. Web application performance analysis using angular, react and vue. js frameworks. *Journal of Computer Sciences Institute*, 23:77–83, 2022.

[15] VKR Ballamudi, Karu Lal, Harshith Desamsetti, and Sreekanth Dekkati. Getting started modern web development with next. js: An indispensable react framework. *Digitalization & Sustainability Review*, 1(1):1–11, 2021.

[16] felix vemmer. The rise of next.js: Why it's the full-stack framework of choice for modern websites. URL https://www.felixvemmer.com/en/blog/why-next-js.

[17] Adegeo. Data binding overview - wpf .net. URL https://learn.microsoft.com/en-us/dotnet/desktop/wpf/data/?view=netdesktop-8.0.

[18] Steve Burbeck. Applications programming in smalltalk-80 (tm): How to use model-view-controller (mvc). *Smalltalk-80 v2*, 5:1–11, 1992.

[19] Michael Jiang and Allan Willey. Architecting systems with components and services. In *IRI-2005 IEEE International Conference on Information Reuse and Integration, Conf, 2005.*, pages 259–264. IEEE, 2005.

[20] Handlebars — handlebarsjs.com. https://handlebarsjs.com/. [Accessed 24-02-2024].

[21] npm trends: Compare npm package downloads. https://npmtrends.com/. (Accessed on 02/29/2024).

[22] Apr 2023. URL https://kruschecompany.com/ember-jquery-angular-react-vue-what-to-

Appendices