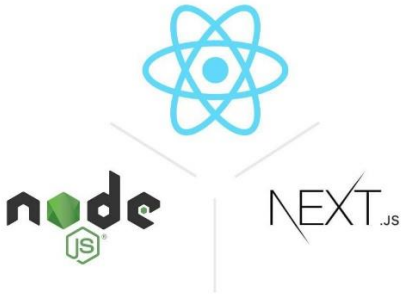
ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΕΛΛΑΔΟΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΕΦΑΡΜΟΓΩΝ ΣΥΣΤΗΜΑΤΩΝ ΑΥΤΟΜΑΤΟΥ ΕΛΕΓΧΟΥ ΚΑΙ ΠΡΟΓΡΑΜΜΑΤΙΖΟΜΕΝΩΝ ΛΟΓΙΚΩΝ ΕΛΕΓΚΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Δημιουργία ιστοσελίδας σε React για το Εργαστήριο Εφαρμογών Συστημάτων Αυτόματου Ελέγχου και Προγραμματιζόμενων Λογικών Ελεγκτών



Των Φοιτητών:
Δεμιτσόγλου Πέτρος - Φυλαχτός Χρήστος
Αρ. Μητρώου: 512112 – 052880

Επιβλέπων Καθηγητής:
Χρήστος Κ. Μανάβης
Λέκτορας

Θεσσαλονίκη 15/9/2021

Ευχαριστίες

Χωρίς την παρουσία , την υποστήριξη , την πίεση και την ανεκτικότητα κάποιων ανθρώπων δεν θα ήταν δυνατή η υλοποίηση αυτής της πτυχιακής εργασίας.

Γι’ αυτό λοιπόν θα θέλαμε να ευχαριστήσουμε θερμά τον επιβλέπων καθηγητή μας κ. Μανάβη Χρήστο που ήτανε δίπλα μας καθ’ όλη την διάρκεια της υλοποίησης της εργασίας και για όλες τις συμβουλές αλλά και την καθοδήγηση που μας έδωσε γιατί πραγματικά χωρίς αυτόν τίποτα δεν θα είχε τελειώσει στην ώρα του και όπως πρέπει.

Τέλος θα θέλαμε να ευχαριστήσουμε τις οικογένειες μας για την στήριξη και την υπομονή τους όλα αυτά τα χρόνια της φοιτητικής μας ζωής.

Περίληψη

Αντικείμενο της παρούσας πτυχιακής εργασίας είναι η ανάπτυξη της ιστοσελίδας του Εργαστηρίου Εφαρμογών Συστημάτων Αυτόματου Ελέγχου και Προγραμματιζόμενων Λογικών Ελεγκτών σε React. Για τον σκοπό αυτό, έγινε βιβλιογραφική έρευνα αναφορικά με την ιστορία της ανάπτυξης ιστοσελίδων και με τα υπάρχοντα JavaScript frameworks. Επίσης πραγματοποιήθηκε σύγκριση της βιβλιοθήκης React με το WordPress CMS και τα frameworks Angular.js και Vue.js. Στη συνέχεια, αναπτύχθηκε η νέα ιστοσελίδα του Εργαστηρίου με το framework Next.js. Η νέα ιστοσελίδα, η οποία βασίζεται στο theme eDemy, παρουσιάζει πολλά πλεονεκτήματα σε σχέση με την υπάρχουσα ιστοσελίδα του Εργαστηρίου, καθώς διαθέτει σύγχρονη responsive σχεδίαση, είναι πιο λειτουργική και ευέλικτη, παρέχει την δυνατότητα τριών επιπέδων χρήστη και αποτελεί ένα ολοκληρωμένο Σύστημα Διαχείρισης Μάθησης.

Λέξεις – κλειδιά*: React, Next.js, JavaScript, ιστοσελίδα, LMS, e-learning.*

Abstract

The subject of this thesis is the construction of a new website for the Automatic Control Systems Applications and Programmable Logic Controllers Laboratory with ReactJS. For this purpose, a literature review on the website development history and on the existing JavaScript frameworks has been conducted. In addition, ReactJS was compared to the WordPress CMS and to the Angular.js and Vue.js frameworks. The new website of the Laboratory was developed with Next.js framework. The new website based on the eDemy theme has a lot of advantages compared to the existing website of the Laboratory, since its design is modern and responsive, it is more practical and flexible, provides three user levels and constitutes a comprehensive Learning Management System.

Keywords: *React, Next.js, JavaScript, website, LMS, e-learning.*

Πίνακας Περιεχομένων

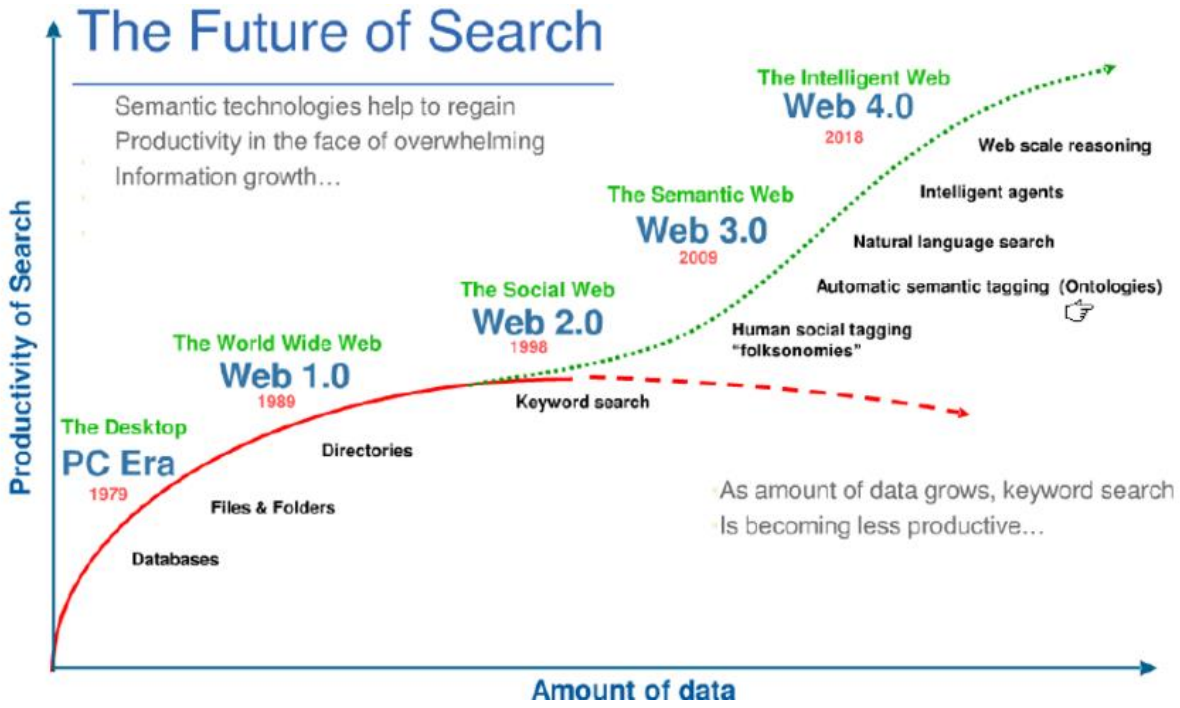
Ιστορική αναδρομή στην ανάπτυξη ιστοσελίδων

Ο ιστός (web) εφευρέθηκε από τον Βρετανό επιστήμονα Tim Berners – Lee και έκανε την εμφάνιση του στις αρχές του 1989 [3]. Ο ιστός αρχικά σχεδιάστηκε για να καλύπτει τις ανάγκες ανταλλαγής πληροφοριών μεταξύ πανεπιστημίων και ινστιτούτων επιστημόνων ανά τον κόσμο [3]. Ο Tim Berners – Lee ήταν επίσης ο εφερεύτης της HTML (HyperText Markup Language), του URL (Uniform Resource Locator) και του πρωτοκόλλου HTTP (Hyper Text Transfer Protocol), όπως και του πρώτου φυλλομετρητή ιστοσελίδων (web browser) [5].

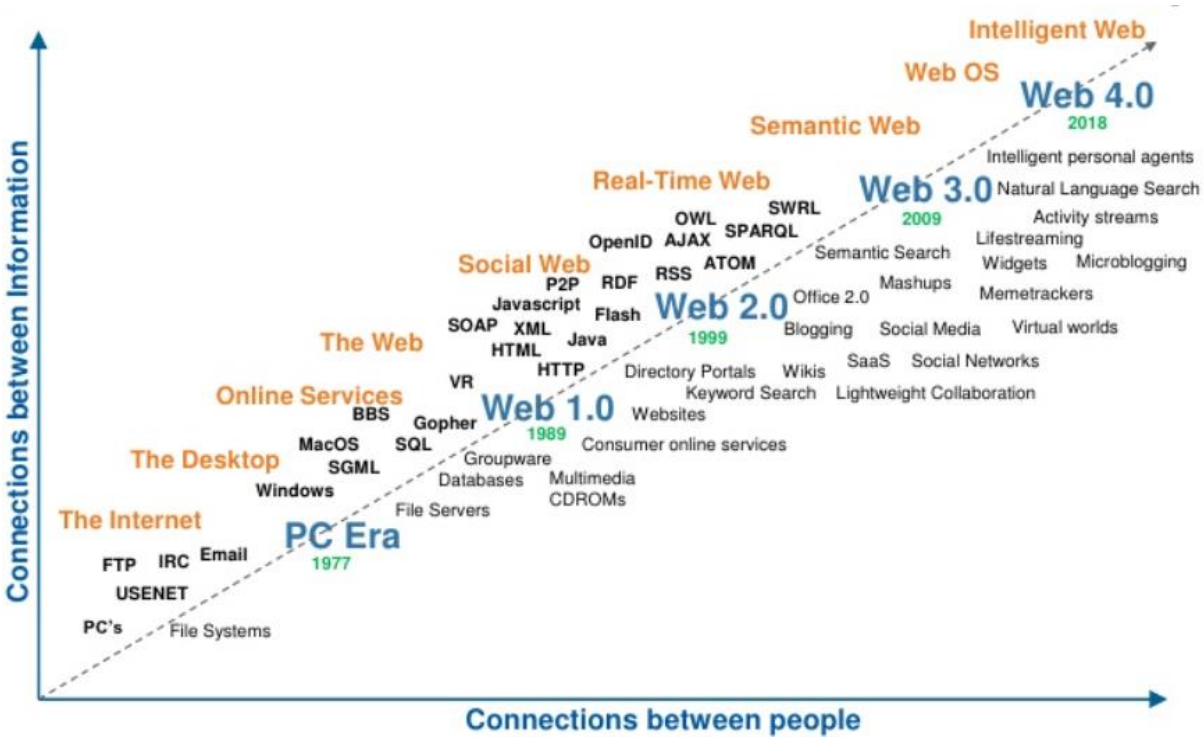
Η πρώτη έκδοση του ιστού ήταν η Web 1.0, η οποία εφευρέθηκε από τον Tim Berners – Lee και χρησιμοποιήθηκε μέχρι το 2003, ήταν μία ιστοσελίδα του παγκόσμιου ιστού (World Wide Web) με μη επεξεργασμένα δεδομένα (raw data). Ο χρήστης μπορούσε να αναζητήσει δεδομένα, χωρίς όμως να έχει την δυνατότητα να τα μοιραστεί ή να εισαγάγει νέα δεδομένα [3]. Οι τεχνολογίες που χρησιμοποίηθηκαν στην έκδοση Web 1.0 ήταν οι HTML, HTTP, URI, XML και XTML [3]. Άλλες τεχνολογίες που υποστηρίζονταν από την έκδοση Web 1.0 και παρείχαν αλληλεπίδραση μεταξύ του χρήστη και του server ήταν οι ASP, PHP, PERL, JSP και CGI [3]. Η έκδοση Web 1.0 λειτουργούσε πολύ αργά αναγκάζοντας τον χρήστη να ανανεώσει την ιστοσελίδα για να λάβει καινούργια δεδομένα και ταυτόχρονα δεν επέτρεπε στον χρήστη να τροποποιήσει τα δεδομένα της ιστοσελίδας [3].

Η δεύτερη έκδοση του ιστού Web 2.0 παρουσιάστηκε επίσημα το 2004 από τον αντιπρόεδρο της εταιρείας O'Reilly Media, Dale Dougherty [3]. Η έκδοση Web 2.0 κατέστησε τον ιστό αμφίδρομο, δηλαδή πλέον υπήρχε η δυνατότητα ανάγνωσης και εγγραφής δεδομένων, πέραν της δυνατότητας download και upload δεδομένων από τον διαχειριστή της ιστοσελίδας. Η τεχνολογική υποδομή της έκδοσης Web 2.0 περιλάμβανε τεχνολογίες όπως RSS, Atom και RDF για τη δημιουργία web services αλλά και τις τεχνολογίες Ajax, REST, DOM, CSS, XML και JavaScript [3]. Όμως η συγκεκριμένη έκδοση, αν και ενδείκνυται για την ανάπτυξη κοινωνικών δικτύων, παρουσιάζει προβλήματα αναφορικά με την ασφάλεια προσωπικών δεδομένων και με την ιδιωτικότητα (privacy) [3].

Η τελευταία και τρέχουσα έκδοση του ιστού Web 3.0 ή αλλιώς Σημασιολογικός Ιστός (Semantic Web) ξεκίνησε το 2014 ως εκτελέσιμος ιστός επιτρέποντας στον χρήστη να αλληλεπιδράσει με δυναμικές εφαρμογές [3]. Σύμφωνα με τη θεωρία Conrad Wolfram, η έκδοση Web 3.0 επιχειρεί να εφαρμόσει την τεχνητή νοημοσύνη στους υπολογιστές έτσι ώστε να σκέφτονται αντί των ανθρώπων στην αναζήτηση δεδομένων, όπως και να μετατρέψει τον παγκόσμιο ιστό σε μία τεράστια βάση δεδομένων [3]. Αν και η έκδοση Web 3.0 περιέχει πολλά καινούργια χαρακτηριστικά, υπάρχουν ακόμα προβλήματα αναφορικά με την ταυτοποίηση του χρήστη και τα αυξημένα αιτήματα (requests) του client προς τον server [3].



Εικόνα 1 Παραγωγικότητα ανάζητησης στις διαφορετικές εκδόσεις του ιστού [4]

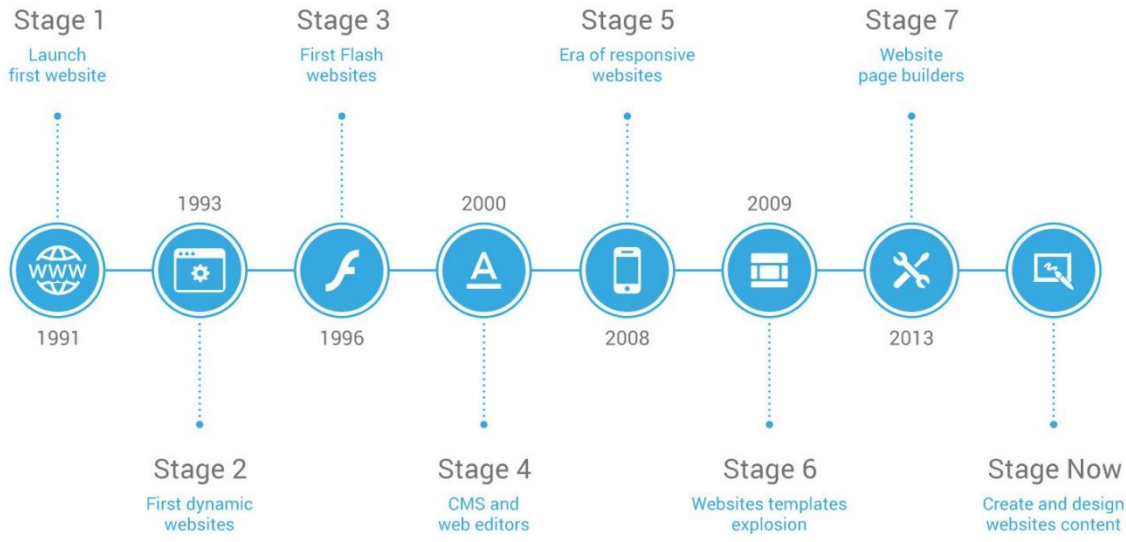


Εικόνα 2 Διάγραμμα συνδέσεων μεταξύ των πληροφοριών σε σχέση με τις συνδέσεις μεταξύ των ανθρώπων για τις διαφορετικές εκδόσεις του ιστού [4]

Η τεχνολογική στοίβα του ιστού (Web Technology Stack) είναι μία στοίβα πρωτοκόλλων που αποτελείται από τέσσερα επίπεδα που χρησιμοποιούνται για τον ορισμό, τον εντοπισμό, την εφαρμογή και την αλλεπίδραση των υπηρεσιών του ιστού μεταξύ τους [3]. Η τεχνική για τον εντοπισμό δεδομένων στον ιστό βασίζεται στο URI (Uniform Resource

Identifier), το οποίο είναι πληροφορία όπως μία ιστοσελίδα, ένα κείμενο, βίντεο ή ήχος [3]. Το πρωτόκολλο HTTP καθορίζει τους κανόνες της επικοινωνίας μεταξύ του client και server, ενώ το πρωτόκολλο FTP (File Transfer Protocol) χρησιμοποιείται για τη μεταφορά των δεδομένων [3]. Επιπλέον, το IP Security είναι ένα ακόμα πρωτόκολλο που χρησιμοποιείται για την κρυπτογράφηση και αποκρυπτογράφηση των δεδομένων που ανταλλάσονται στον ιστό μεταξύ του client και του server [3].

Από τις αρχές της δεκαετίας του 1990 αυξανόταν ο αριθμός των ιστοσελίδων και ταυτόχρονα η ανάγκη για αλληλεπίδραση με τις ιστοσελίδες [5]. Έτσι, με τη βοήθεια της JavaScript έγινε εφικτή η εισαγωγή διαδραστικού περιεχομένου στις ιστοσελίδες και η ανάπτυξη διαδραστικών ιστοσελίδων [5]. Το 1996 η τεχνολογία Flash έκανε την εμφάνισή της και κατέστησε δυνατή τη χρήση animation εικόνων και βίντεο στις ιστοσελίδες [5]. Το επόμενο σημαντικό βήμα στην ανάπτυξη ιστοσελίδων ήταν οι επεξεργαστές περιεχομένου ιστού (Web Content Editor), οι οποίοι ήταν HTML εργαλεία που επέτρεπαν το customization των ιστοσελίδων και μέχρι σήμερα αποτελούν αναπόσπαστα εργαλεία στα συστήματα διαχείρισης περιεχομένου (Content Management Systems) [5]. Η έκδοση Web 2.0 οδήγησε στην ανάπτυξη responsive ιστοσελίδων, βελτιστοποιημένων ειδικά για φορητές συσκευές [5]. Καθώς το περιεχόμενο των ιστοσελίδων αυξάνεται και πρέπει να προσπελαύνεται ταχύτερα, υπάρχει η τάση για CMS, όπως το WordPress, τα οποία δεν απαιτούν κώδικα και η κατασκευή ιστοσελίδων πραγματοποιείται μέσω drag-n-drop εργαλείων [5].



Εικόνα 3 Σημεία αναφοράς στην ανάπτυξη ιστοσελίδων [5]

Βιβλιοθήκη React

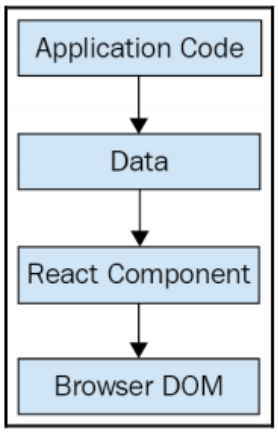
Η React είναι μία βιβλιοθήκη JavaScript που στοχεύει να απλοποιήσει την ανάπτυξη οπτικών διεπαφών [1]. Κατασκευάστηκε από την εταιρεία Facebook το 2013 και χρησιμοποιείται σε αρκετές παγκοσμίως γνωστές διαδικτυακές εφαρμογές όπως το Facebook και το Instagram. Ο βασικός στόχος της React είναι να απλοποιήσει τη σχεδίαση και την κατάσταση μίας διεπαφής σε κάθε στάδιο ανάπτυξης, διαιρώντας την διεπαφή χρήστη (User Interface) σε μία συλλογή από συστατικά στοιχεία [1].

Την εποχή που έκανε την εμφάνισή της η React, τα κυρίαρχα πλαίσια λογισμικού (frameworks) ήταν τα Ember.js και Angular 1.x . Η React είχε το πλεονέκτημα ότι μπορούσε ευκόλα να ενσωματώσει ένα project σε υπάρχον κώδικα. Επιπλέον, δεν απαιτούσε την εφαρμογή της αρχιτεκτονικής Model View Controller [1]. Στη συνέχεια, η έκδοση Angular 2.x παρουσίαζε προβλήματα ασυμβατότητας με προηγούμενες εκδόσεις, ενώ η React παρουσίαζε ταχύτερους χρόνους εκτέλεσης, το οποίο έπεισε τους προγραμματιστές να την δοκιμάσουν [1]. Αξίζει να αναφερθεί ότι η υποστήριξη της React από την Facebook και το γεγονός ότι η React παραμένει λογισμικό ανοιχτού κώδικα, καθιστούν την React ακόμα πιο ελκυστική για τους προγραμματιστές [1].

Η React διαθέτει μία μικρή Διεπαφή Προγραμματισμού Εφαρμογών (Application Programming Interface) και η κατανόησή της βασίζεται στις εξής έννοιες [1]:

I. **Component:** Είναι ένα ανεξάρτητο κομμάτι της διεπαφής. Για παράδειγμα, σε ένα blog μπορεί το sidebar ή η λίστα των posts να αποτελούν components. Κάθε component μπορεί να αποτελείται από πολλά components.
II. **JSX:** Είναι μία τεχνολογία που εμφανίστηκε πρώτη φορά στην React. Αν και η JSX μοιάζει να συνδυάζει JavaScript και HTML, στην πραγματικότητα η JSX χρησιμοποιεί σύνταξη δηλωτικού προγραμματισμού (declarative programming) με σκοπό να ορίσει πως πρέπει να είναι ένα component της διεπαφής χρήστη.
III. **State:** Είναι ένα αντικείμενο που περιέχει πληροφορίες για ένα component, οι οποίες μπορεί να μεταβληθούν κατά τη διάρκεια του κύκλου ζωής του.
IV. **Props:** Είναι ενα αντικείμενο που διατηρεί τις τιμές των πεδίων κάποιου tag, λειτουργεί όμοια με τα πεδία της HTML και χρησιμοποιείται για τη μεταφορά δεδομένων μεταξύ των components.

Η React γενικά θεωρείται ως το επίπεδο προβολής της εφαρμογής. Ακριβώς όπως η jQuery χρησιμοποιεί στοιχεία της διεπαφής χρήστη και εισαγάγει πρωτότυπα στοιχεία στην σελίδα, τα components της React αλλάζουν αυτό που ο χρήστης βλέπει [12]. Ο κώδικας της εφαρμογής δημιουργεί δεδομένα, τα οποία διαχειρίζεται κάποιο component της React. Αυτό με τη σειρά του μεταβάλλει τον κώδικα HTML στην σελίδα [12].



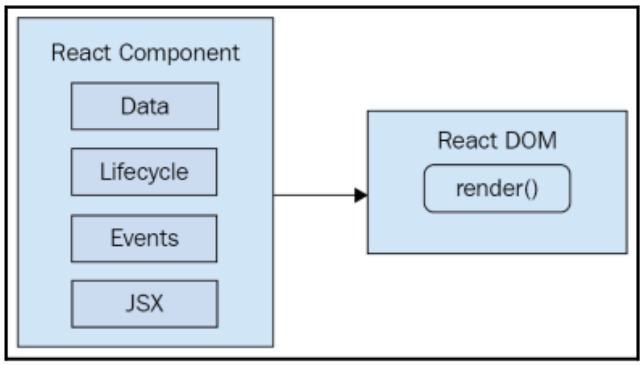
Εικόνα 4 Στάδια εκτέλεσης κώδικα της React [13]

Με βάση μία άλλη προσέγγιση, η React έχει δύο βασικές Διεπαφές Προγραμματισμού Εφαρμογών [13]:

- ✓ **React Component API**: Πρόκειται για τμήματα της σελίδας που υφίστανται επεξεργασία από το REACT DOM.
- ✓ **React DOM**: Είναι η Διεπαφή Προγραμματισμού Εφαρμογών που εκτελεί την πραγματική επεξεργασία της ιστοσελίδας.

Για κάθε component της React είναι σημαντική η κατανόηση των ακόλουθων εννοιών [13]:

- ✓ Data: Τα δεδομένα προέρχονται από κάποια πηγή και υφίστανται επεξεργασία από το component.
- ✓ Lifecycle: Ο κύκλος ζωής αποτελείται από μεθόδους που εφαρμόζονται και αντιστοιχούν στις φάσεις εισόδου και εξόδου του component στην διαδικασία επεξεργασίας που λαμβάνει χώρα κατά την διάρκεια εκτέλεσης της εφαρμογής.
- ✓ Events: Είναι ο κώδικας που καθορίζει την διάδραση με τον χρήστη.
- ✓ JSX: Είναι η σύνταξη των components που χρησιμοποιείται για να περιγράψει δομές της διεπαφής χρήστη.



Εικόνα 5 Βασικές έννοιες των components της React [13]

Ως επεξεργασία (rendering) στην React νοείται η διαδικασία του μετασχηματισμού των components σε στοιχεία του DOM (Document Object Model), τα οποία ο browser μπορεί να αναγνωρίσει και να απεικονίσει στην οθόνη. Το DOM αναπαριστά τον κώδικα HTML στον browser αφού έχει υποστεί επεξεργασία και και το DOM API αναφέρεται στο πως η JavaScript μπορεί να αλλάξει το περιεχόμενο της σελίδας [13].

Τα components της React βασίζονται στα δεδομένα που δέχονται ως εισόδους. Αυτά δεδομένα αναπαριστούν τα δυναμικά μέρη της διεπαφή χρήστη. Για παράδειγμα, ένα στοιχείο της διεπαφής χρήστης που υφίσταται επεξεργασία με βάση μία λογική τιμή, μπορεί να αλλάξει την επόμενη φορά που το component υφίσταται επεξεργασία [13]. Κάθε φορά που ένα component υφίσταται επεξεργασία, δέχεται ως είσοδο τον κώδικα JSX εκείνη την χρονική στιγμή.

Το εικονικό DOM της React διατηρεί μία αναπαράσταση των στοιχείων του πραγματικού DOM στην μνήμη. Με αυτόν τον τρόπο, όταν ένα component υφίσταται ξανά επεξεργασία, το εικονικό DOM συγκρίνει το καινούργιο περιεχόμενο με το περιεχόμενο που απεικονίζεται ήδη στη σελίδα. Με βάση τις διαφορές, το εικονικό DOM εκτελεί τις απαραίτητες ενέργειες για να εφαρμόσει τις αλλαγές στη σελίδα. Συνεπώς, όχι μόνο διατηρείται ο δηλωτικός κώδικας που χρειάζεται για την ενημέρωση της διεπαφής χρήστη, αλλά η React εξασφαλίζει ότι ο κώδικας εκτελείται με τον πλέον αποδοτικό τρόπο [13].

Όπως αναφέρθηκε, ο κώδικας JSX ερμηνεύεται σε διεργασίες που ενημερώσουν τα στοιχεία της διεπαφής χρήστη [13]. Ένας καλύτερος τρόπος να κατανοηθεί το πως η React ερμηνεύει τα δηλωτικά components της διεπαφής χρήστη είναι η παραδοχή ότι δεν παίζει ρόλο ποιος είναι ο στόχος επεξεργασίας. Αν και ο στόχος επεξεργασίας μπορεί να είναι το DOM του browser για την React, δεν περιορίζεται μόνο σε αυτό. Η React έχει τη δυνατότητα να χρησιμοποιηθεί για την δημιουργία οποιασδήποτε διεπαφής χρήστη σε οποιαδήποτε συσκευή. Προς το παρόν, η έκδοση React Native χρησιμοποιείται για τη δημιουργία εφαρμογών και διεπαφών χρήστη σε κινητές συσκευές [13].

Σύγκριση React με WordPress

Όταν εμφανίστηκε το WordPress το 2003, άλλαξε τα δεδομένα στην κατασκευή ιστοσελίδων, καθώς ήταν δυνατή η ανάπτυξη ιστοσελίδας από οποιονδήποτε με ή χωρίς γνώσεις PHP [6]. Αν και το WordPress είναι ιδανικό για κατασκευή blogs ή ιστοσελίδων εταιρειών, δεν ενδείκνυται για ιστοσελίδες που πρόκειται να ακολουθήσουν τα νέα δεδομένα, δεν παρέχει επεκτασιμότητα, ταχύτητα και ευελιξία [6]. Ειδικότερα, το WordPress [6]:

- ✓ Χρειάζεται plugins για τα πρόσθετα χαρακτηριστικά. Με την πάροδο του χρόνου, η διαχείριση της ιστοσελίδας γίνεται δύσκολη και ο αριθμός των plugins μειώνει τις επιδόσεις της ιστοσελίδας.

- ✓ Χρειάζεται αναβαθμίσεις τόσο το ίδιο το WordPress όσο και τα themes και τα plugins για την αντιμετώπιση κενών ασφαλείας και τη διόρθωση των bugs. Κάποιες φορές η αντιμετώπιση αυτών των προβλημάτων μπορεί να είναι δύσκολη.
- ✓ Δεν παρέχει γρήγορη φόρτωση των ιστοσελίδων με αποτέλεσμα να μην συνίσταται για περιπτώσεις π.χ. εμπορικές ιστοσελίδες, όπου οι πελάτες επιθυμούν γρήγορη πρόσβαση στα δεδομένα της ιστοσελίδας. Επιπλέον, η ταχύτητα της ιστοσελίδας επηρεάζει σημαντικά τα αποτελέσματα του SEO (Search Engine Optimization).
- ✓ Έχει κενά ασφαλείας τα οποία μπορεί να εκμεταλλευτούν hackers ή spammers.
- ✓ Εμφανίζει συχνές επιπλοκές (downtime) με συνέπεια η ιστοσελίδα να τίθεται offline, το οποίο, ειδικά για τις επιχειρήσεις, έχει μεγάλο κόστος.

Από την άλλη πλευρά, η ιστοσελίδα η οποία αναπτύχθηκε με React φορτώνει το περιεχόμενο της ιστοσελίδας μέσω CDN (Content Delivery Network). Ειδικότερα, ο επισκέπτης της ιστοσελίδας λαμβάνει τα δεδομένα της ιστοσελίδας από τον server που βρίσκεται πιο κοντά του και όχι από τον server της ιστοσελίδας, με αποτέλεσμα να αυξάνεται σημαντικά ο χρόνος φόρτωσης της ιστοσελίδας και το SEO να παρέχει καλύτερα αποτελέσματα [6]. Η ιστοσελίδα, το περιεχομένο της οποίας προέρχεται από πολλούς servers CDN, παρουσιάζει επίσης αυξημένη ασφάλεια [6].

Η χρήση JavaScript διευκολύνει την ανάπτυξη της ιστοσελίδας, καθώς η JavaScript παρουσιάζει πολύ λιγότερα bugs από το WordPress CMS [6]. Η JavaScript επίσης καθιστά εύκολη την ενσωμάτωση κώδικα εφαρμογών και εύκολες τις προσωποιημένες ρυθμίσεις στην ιστοσελίδα [6].

Η χρήση του JSX (JavaScript) από την React επιτρέπει την ενσωμάτωση κώδικα HTML εντός των components της React και με αυτόν τον τρόπο γίνεται εύκολη η δημιουργία δυναμικών διαδικτυακών εφαρμογών [7]. Ένα ακόμα πλεονέκτημα της React είναι η χρηστικότητα (reusability) των components, το οποίο διευκολύνει την ανάπτυξη σύνθετων εφαρμογών [7].

Το DOM (Document Object Model) της JavaScript είναι ένα API που διαχειρίζεται κώδικα HTML, XML και XHTML. Όταν το DOM ενημερώνεται, οι επιδόσεις της εφαρμογής μειώνονται. Η React λύνει αυτό το πρόβλημα με ένα εικονικό DOM (Virtual DOM) που υπάρχει αποκλειστικά στη μνήμη, ενώ στον browser υπάρχει η αναπαράστασή του [7]. Γενικά οι εφαρμογές σε React δεν χρειάζεται ενημερώσεις για να είναι λειτουργικές [7].

Σύγκριση React με Angular

Το ανοιχτού κώδικα πλαίσιο λογισμικού Angular αναπτύχθηκε από την εταιρεία Google. Αν και τόσο η React όσο και το Angular είναι κατάλληλα για ανάπτυξη front – end εφαρμογών, η βασική διαφορά τους είναι ότι η React χρησιμοποιεί το Virtual DOM, ενώ το Angular χρησιμοποιεί το DOM [7]. Αυτό σημαίνει ότι όταν γίνεται αίτημα στο Angular για τροποποίηση της πληροφορίας σε ένα πίνακα HTML, τότε χρειάζεται να ενημερωθεί όλη η δομή των πινάκων HTML εώς ότου βρεθεί η πληροφορία για την οποία έγινε το αίτημα [8].

Το Virtual DOM της React εγγυάται ταχύτερη εκτέλεση των εφαρμογών γιατί τροποποιεί την ζητούμενη πληροφορία χωρίς να χρειάζεται να ενημερωθεί όλο το αρχείο HTML [8].

Στην React η γλώσσα προγραμματισμού είναι η JavaScript, ενώ στο Angular η TypeScript [8][9]. Η React υποστήριζει το one – way data binding που σημαίνει ότι ένα στοιχείο της διεπαφής χρήστη δεν μπορεί να μεταβάλλει το state ενός component, ενώ στο Angular οι μεταβολές στα στοιχεία της διεπαφής χρήστης και στα components επηρεάζουν το ένα το άλλο αμφίδρομα [9].

Σχετικά με τη συμβατότητα το Angular, χρειάζονται αναβαθμίσεις ώστε να υποστηρίζει παλαιότερες εκδόσεις, ενώ η React υποστηρίζει πλήρως τις προηγούμενες εκδόσεις χωρίς να απαιτούνται ενημερώσεις [8]. Η React είναι πιο εύκολη στην εκμάθηση καθώς ο κώδικας του component υπάρχει σε ένα αρχείο. Από την άλλη πλευρά, το Angular περιλαμβάνει πολλές έννοιες και κανόνες σύνταξης [8]. Η React είναι τέσσερις φορές πιο δημοφιλής από το Angular με βάση τα downloads (100 εκ. Angular / 400 εκ. React) [9].

Εικόνα 6 Δημοφιλείς διαδκτυακές εφαρμογές που αναπτύχθηκαν με Angular [10]

Σύγκριση React με Vue

Το ανοιχτού κώδικα πλαίσιο λογισμικού JavaScript, Vue, αναπτύχθηκε από τον πρώην υπάλληλο της Google, Evan You, με σκοπό να συνδυάσει χαρακτηριστικά από Angular, React και Ember έτσι ώστε η ανάπτυξη front – end διαδικτυακών εφαρμογών να γίνει ευκολότερη, ταχύτερη και πιο ευχάριστη [11].

Μία από τις βασικότερες διαφορές μεταξύ της React και του Vue, είναι ότι το Vue, σε αντίθεση με τη React, χρησιμοποιεί HTML πρότυπα και JSX στο επίπεδο προβολής (view layer), ενώ η React μόνο JSX [11]. Η React ενδείκνυται για την ανάπτυξη τόσο σύνθετων

διαδικτυακών όσο και κινητών εφαρμογών, ενώ το Vue προτείνεται μόνο για την ανάπτυξη απλών διαδικτυακών εφαρμογών [12].

Η React παρέχει μεγαλύτερο έλεγχο στον προγραμματιστή αξιοποιώντας τις αρχές του συναρτησιακού προγραμματισμού και την επικοινωνία μεταξύ των components [12]. Από την άλλη πλευρά, το Vue περιλαμβάνει περισσότερα ενσωματωμένα χαρακτηριστικά και βιβλιοθήκες μετατρέποντας την ανάπτυξη της ιστοσελίδας πιο κατανοητή. Για παράδειγμα, το Vue επιτρέπει την γρήγορη εισαγωγή της βιβλιοθήκης πυρήνα (core library) και του κώδικα στις υπάρχουσες ιστοσελίδες, και δεν απαιτεί τη χρήση components για απλά χαρακτηριστικά [12].

Σχετικά με τις επιδόσεις της ιστοσελίδας, το Vue έχει ελαφρώς ταχύτερο χρόνο εκκίνησης (start-up time) κατανέμοντας την μνήμη καλύτερα, ενώ η React έχει ταχύτερους χρόνους εκτέλεσης [12].

Αν και η εκμάθηση του Vue είναι εύκολη και το Vue διαθέτει μία μεγάλη διαδικτυακή κοινότητα για την παροχή υποστήριξης αλλά και πληρέστερη βιβλιογραφία, η React παραμένει πολύ πιο δημοφιλής [11].

Εικόνα 7 Δημοφιλείς διαδικτυακές εφαρμογές που αναπτύχθηκαν με Vue και React [12]

Ανάπτυξη διαδικτυακών εφαρμογών σε React

Προαπαιτούμενα

Για την σύνταξη του κώδικα της εφαρμογής είναι απαραίτητη η χρήση ενός επεξεργαστή, το οποίο παρέχει όλα τα εργαλεία που χρειάζεται ένας προγραμματιστής και υποστηρίζει λειτουργίες όπως debugging, εκτέλεση διεργασιών και version control [14]. Για

όλα τα projects σε React, χρειάζεται να είναι εγκατεστημένο το περιβάλλον Node.js μαζί με το npm (Node Package Manager) [14]. Μερικές φορές για την εκτέλεση των εφαρμογών χρειάζονται βιβλιοθήκες από πηγές τρίτων, οι οποίες εγκαθίστανται με τη βοήθεια του Git. Συστήνεται επίσης η χρήση του φυλλομετρητή ιστοσελίδων Google Chrome [14]. Συγκεκριμένα, στα λειτουργικά συστήματα Windows πρέπει να εγκατασταθεί η υπηρεσία IIS (Internet Information Services) προκειμένου να τρέχουν τοπικά οι servers δικτύου [14].

JavaScript

Η JavaScript είναι γλώσσα του διαδικτύου που υποστηρίζεται από πολλούς φυλλομετρητές ιστοσελίδων όπως ο Google Chrome, ο Firefox, ο Safari, ο Microsoft Edge και ο Internet Explorer. Κάθε φυλλομετρητής ιστοσελίδων έχει τον δικό του διερμηνέα για την εκτέλεση του κώδικα JavaScript [14]. Η δημοφιλία της JavaScript οδήγησε στη δημιουργία του προτύπου ECMAScript ή ES. Η πέμπτη έκδοση του προτύπου είναι η ES5 και το 2009 υιοθετήθηκε από όλους τους γνωστούς φυλλομετρητές ιστοσελίδων [14]. Η έκτη έκδοση του προτύπου ES6, η οποία ολοκληρώθηκε το 2015 και έκτοτε αναβαθμίζεται, περιείχε πολλά νέα χαρακτηριστικά, περίπου 24, και διέφερε σημαντικά από την πέμπτη έκδοση του προτύπου. Η έκδοση ES7 προσέθεσε μόνο δύο νέα χαρακτηριστικά [14].

Ανάπτυξη δοκιμαστικής εφαρμογής

Σε αυτήν την ενότητα αναπτύσσεται μία αριθμομηχανή σε React [15] ως δοκιμαστική εφαρμογή με σκοπό την επεξήγηση της ανάπτυξης εφαρμογών σε React.

Η εφαρμογή διαθέτει δύο components, τα αρχεία των οποίων βρίσκονται στον υποφάκελο components. Το ResultComponent, το οποίο εμπεριέχεται στην κλάση Component, εμφανίζει το περιεχόμενο που υπάρχει εντός του tag της παραγράφου <p> και παριστάνει το αποτέλεσμα των πράξεων της αριθμομηχανής. Με την εντολή import εισάγονται οι απαραίτητες βιβλιοθήκες και κλάσεις για την εκτέλεση του κώδικα.

```
import React, {Component} from 'react';

class ResultComponent extends Component {

    render() {
        let {result} = this.props;
        return (
            <div className="result">
                <p>{result}</p>
            </div>
    );

    }
}

export default ResultComponent;
```

Εικόνα 8 ResultComponent της εφαρμογής Calculator σε React [15]

Το KeyPadComponent παριστάνει το πληκτρολόγιο της αριθμομηχανής. Κάθε πλήκτρο της αριθμομηχανής εμφανίζεται στην διεπαφή χρήστη μέσω του στοιχείου <button></button> και διακρίνεται από τα υπόλοιπα πλήκτρα μέσω της παραμέτρου name. Η συνάρτηση OnClick() ενεργοποιείται από το πάτημα του πλήκτρου, δέχεται ως όρισμα το όνομα του πλήκτρου και καλεί την αντίστοιχη συνάρτηση που ορίζεται στην γονική κλάση App.

```
import React, {Component} from 'react';

class KeyPadComponent extends Component {

    render() {
        return (
            <div className="button">
                <button name="(" onClick={e => this.props.onClick(e.target.name)}>(</button>
                <button name="CE" onClick={e => this.props.onClick(e.target.name)}>CE</button>
                <button name=")" onClick={e => this.props.onClick(e.target.name)}>)</button>
                <button name="C" onClick={e => this.props.onClick(e.target.name)}>C</button><br/>
                <button name="1" onClick={e => this.props.onClick(e.target.name)}>1</button>
                <button name="2" onClick={e => this.props.onClick(e.target.name)}>2</button>
                <button name="3" onClick={e => this.props.onClick(e.target.name)}>3</button>
                <button name="+" onClick={e => this.props.onClick(e.target.name)}>+</button><br/>
                <button name="4" onClick={e => this.props.onClick(e.target.name)}>4</button>
                <button name="5" onClick={e => this.props.onClick(e.target.name)}>5</button>
                <button name="6" onClick={e => this.props.onClick(e.target.name)}>6</button>
                <button name="-" onClick={e => this.props.onClick(e.target.name)}>-</button><br/>
                <button name="7" onClick={e => this.props.onClick(e.target.name)}>7</button>
                <button name="8" onClick={e => this.props.onClick(e.target.name)}>8</button>
                <button name="9" onClick={e => this.props.onClick(e.target.name)}>9</button>
                <button name="*" onClick={e => this.props.onClick(e.target.name)}>x</button><br/>
                <button name="." onClick={e => this.props.onClick(e.target.name)}>.</button>
                <button name="0" onClick={e => this.props.onClick(e.target.name)}>0</button>
                <button name="=" onClick={e => this.props.onClick(e.target.name)}>=</button>
                <button name="/" onClick={e => this.props.onClick(e.target.name)}>÷</button><br/>
            </div>
        );
    }
}


export default KeyPadComponent;
```

Εικόνα 9 KeypadComponent της εφαρμογής Calculator σε React [15]

Το αρχείο App.js είναι το γονικό component των προηγούμενων components και το βασικό αρχείο της εφαρμογής, στο οποίο γίνεται η επεξεργασία των components. Τα ResultComponent και KeyPadComponent δηλώνονται στην αρχή του αρχείου.

Η τιμή της μεταβλητής result που αντιστοιχεί στη τιμή του αποτελέσματος του state περνιέται στο ResultComponent. Η μεταβλητή result αρχικοποιείται στον constructor της κλασης. Η συνάρτηση this.calculate() εκτελείται όταν πατιέται το πλήκτρο “=”, η συνάρτηση this.reset() εκτελείται όταν πατιέται το πλήκτρο "C”, η συνάρτηση this.backspace() εκτελείται όταν πατιέται το πλήκτρο “CE”, ενώ σε κάθε άλλη περίπτωση προστίθεται στην μεταβλητή result η τιμή του πλήκτρου. Οι προηγούμενες λειτουργίες ορίζονται με το onClick event.

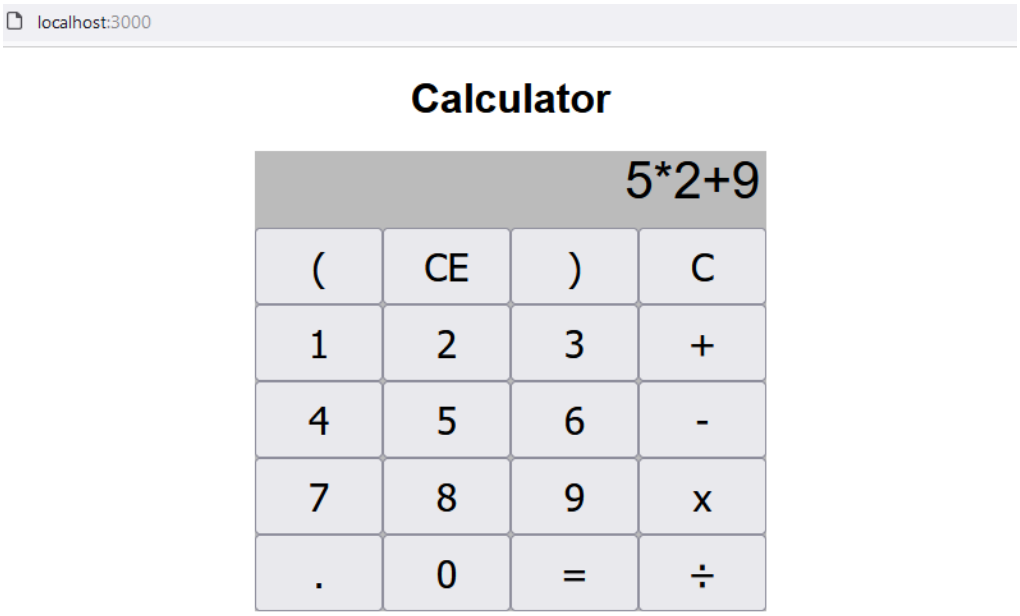
Η συνάρτηση this.calculate() υπολογίζει την τιμή της έκφρασης στην αριθμομηχανής με χρήση της συνάρτησης eval(), συνάρτηση this.reset() αναθέτει στη μεταβλητή result την τιμή “ ” (κενό) και η συνάρτηση this.backspace() με χρήση της συνάρτησης slice() αφαιρεί από το string της result ένα χαρακτήρα.

Το αρχείο App.css της εφαρμογής περιέχει τον κώδικα CSS για την μορφοποίηση των στοιχείων της εφαρμογής (χρώμα, συντεταγμένες, περιθώρια κοκ). Επιπλέον, τα πακέτα και οι βιβλιοθήκες που είναι απαραίτητα για την εκτέλεση της εφαρμογής δηλώνονται στο αρχείο package.json.

Η εφαρμογή μπορεί να εκτελεστεί τοπικά με την εντολή “npm start” από το τερματικό του Visual Studio Code.

```css
.result {
  height: 60px;
  background-color: #bbb;
  width: 100%;
  text-align: right;
}

.result p {
  font-size: 40px;
  margin: 5px;

}

.calculator-body {
  max-width: 400px;
  margin: auto;
}

.button {
  display: block;
  background-color: #bbb;
}

button {
  width: 25%;
  height: 60px;
  font-size: 30px;
}
```

Εικόνα 10 Αρχείο App.css της εφαρμογής Calculator σε React [15]



Εικόνα 11 Εφαρμογή Calculator σε React [15]

```
import React, { Component } from 'react';
import './App.css';
import ResultComponent from './components/ResultComponent';
import KeyPadComponent from "./components/KeyPadComponent";

class App extends Component {
    constructor(){
        super();

        this.state = {
            result: ""
        }
    }

    onClick = button => {

        if(button === "="){
            this.calculate()
        }

        else if(button === "C"){
            this.reset()
        }
        else if(button --- "CC"){
            this.backspace()
        }

        else {
            this.setState({
                result: this.state.result + button
            })
        }
    };


    calculate = () => {
        var checkResult = ''
        if(this.state.result.includes('--')){
            checkResult = this.state.result.replace('--','+')
        }

        else {
            checkResult = this.state.result
        }

        try {
            this.setState({
                result: (eval(checkResult) || "" ) + ""
            })
        } catch (e) {
            this.setState({
                result: "error"
            })

        }
    };

    reset = () => {
        this.setState({
            result: ""
        })
    };

    backspace = () => {
        this.setState({
            result: this.state.result.slice(0, -1)
        })
    };

    render() {
        return (
            <div>
                <div className="calculator-body">
                    <h1 align="center">Calculator</h1>
                    <ResultComponent result={this.state.result}/>
                    <KeyPadComponent onClick={this.onClick}/>
                </div>
            </div>
        );
    }
}

export default App;
```

Εικόνα 12 App.js αρχείο της εφαρμογής Calculator σε React [15]

Next.js

Το Next.js είναι ένα JavaScript framework ανοιχτού κώδικα, το οποίο κατασκευάστηκε από την εταιρεία Vercel [16]. Βασίζεται στο back-end περιβάλλον εκτέλεσης JavaScript Node.js και στον μεταγλωττιστή JavaScript Babel [16]. Σε συνδυασμό με την React, καθιστά την επεξεργασία από την πλευρά του server πολύ εύκολη [16]. Εκτός από την δυνατότητα στατικής εξαγωγής (static export) και του pre-rendering, το Next.js υποστηρίζει την αυτόματη βελτιστοποίηση μεγέθους της αναπτυσσόμενης εφαρμογής, την προεπισκόπηση και την γρηγορότερη σύνθεση της εφαρμογής [16]. Το Next.js διαθέτει την απαραίτητη λειτουργικότητα για τη δημιουργία μιας εκτελέσιμης εφαρμογής. Επιπλέον, διαθέτει πλήρη βιβιογραφία και γίνεται διαρκώς πιο δημοφιλές στους προγραμματιστές [16].

Η δημιουργία ενός project με React απαιτεί τη δημιουργία του component και την προσθήκη του στο router, ενώ με το Next.js αρκεί η προσθήκη της ιστοσελίδας στον φάκελο με τον κατάλληλο υπερσύνδεσμο του component στο header [16]. Η δημιουργία εφαρμογών με το Next.js είναι εξαιρετικά γρήγορη χάρη στις στατικές σελίδες και στην επεξεργασία από την πλευρά του server (server-side rendering). Η React υποστηρίζει μόνο client – side rendering, το οποίο δεν προσφέρει υψηλές αποδόσεις για τις εφαρμογές [16].

Το Next.js βελτιώνει σημαντικά το User Experience καθώς [17]:

1. Προσφέρει ελευθερία. Η λειτουργικότητά του δεν στηρίζεται σε plugins όπως συμβαίνει με τα CMS. Αντίθετα, επιτρέπει την τροποποίηση του front-end χωρίς περιορισμούς ανάλογα με τις εκάστοτε ανάγκες.
2. Είναι responsive. Οι ιστοσελίδες και οι διαδικτυακές εφαρμογές που αναπτύσσονται με το Next.js λειτουργούν σε οποιαδήποτε συσκευή και προσαρμόζονται στην ανάλυση και το μέγεθος της συσκευής.
3. Βελτιώνει τον χρόνο φόρτωσης της ιστοσελίδας. Οι ιστοσελίδες που αναπτύσσονται με Next.js είναι πολύ γρήγορες γιατί είναι στατικές.
4. Παρέχει ασφάλεια δεδομένων. Στην περίπτωση των στατικών ιστοσελίδων δεν υφίσταται απευθείας σύνδεση με τη βάση δεδομένων, με dependencies και άλλες ευαίσθητες πληροφορίες.

Η χρήση του Next.js για την ανάπτυξη ενός React JS project προτείνεται στις παρακάτω περιπτώσεις [18]:

- ✓ Η εφαρμογή περιλαμβάνει τμήμα server και τμήμα client.
- ✓ Η εφαρμογή React έχει αναπτυχθεί ήδη σε Express.js και χρειάζεται να επεκταθεί στην πλευρά του client.
- ✓ Η εφαρμογή είναι μία δυναμική ιστοσελίδα.
- ✓ Η ιστοσελίδα χρειάζεται SEO (Search Engine Optimization) και έχει δυναμικό περιεχόμενο.

Table 1 Σύγκριση React με Next.js [16]

Παράμετρος	Next.js	React
Κώδικας	Λιγότερος κώδικας και το project αναπτύσσεται εύκολα	Γρήγορη ανάπτυξη με την βοήθεια του Create-React-App
Απόδοση	Γρήγορες εφαρμογές χάρη στις στατικές σελίδες και στο server-side rendering	Αξιοπρεπής
Χαρακτηριστικά	Static export, pre-rendering, αυτόματη βελτιστοποίηση μεγέθους της αναπτυσσόμενης εφαρμογής, προεπισκόπηση, γρήγορη σύνθεση εφαρμογής	Routing, πρότυπα διαχείρισης κατάστασης (state management patterns), επεκτασιμότητα
Βιβλιογραφία	Καλογραμμένη	Καλογραμμένη
Κοινότητα	Μικρή και φιλική	Μεγάλη και φιλική
Εκμάθηση	Εύκολη	Εύκολη
Κόστος ανάπτυξης εφαρμογής	Χαμηλό	Χαμηλό

Στη συνέχεια αναλύονται βασικές έννοιες και στοιχεία του framework Next.js.

Σελίδες στο Next.js

Στο Next.js μία σελίδα είναι ένα React Component που εξάγεται από ένα αρχείο τύπου .js, .jsx, .ts ή .tsx στον φάκελο pages. Κάθε σελίδα συνδέεται με μία διαδρομή με βάση το όνομα του αρχείου [20]. Για παράδειγμα, η σελίδα “pages/about.js” εξάγει ένα React Component και είναι προσβάσιμη στην διαδρομή “/about”.

```
function About() {
  return <div>About</div>
}

export default About
```

Εικόνα 13 Παράδειγμα εξαγωγής React Component από σελίδα στο Next.js [20]

Επιπλέον το Next.js υποστηρίζει σελίδες με δυναμικές διαδρομές. Για παράδειγμα, η σελίδα που αντιστοιχεί στο αρχείο “pages/posts/[id].js”, είναι προσβάσιμη στις διαδρομές “posts/1”, “posts/2” κ.ο.κ [20].

Το Next.js παράγει HTML κώδικα για κάθε σελίδα εκ των προτέρων, αντί αυτό να πραγματοποιείται από την JavaScript στην πλευρά του client, με αποτέλεσμα τις καλύτερες επιδόσεις και το βελτιωμένο SEO [20]. Κάθε παραγόμενος HTML κώδικας συνδέεται με τον ελάχιστα απαιτούμενο κώδικα JavaScript για την σελίδα. Κάθε φορά που η σελίδα φορτώνεται στον browser, ο κώδικας JavaScript εκτελείται και η σελίδα γίνεται πλήρως διαδραστική. Η διαδικασία αυτή λέγεται *hydration* [20].

Το Next.js έχει δύο μορφές προεπεξεργασίας (pre-rendering) [20]:

1. **Static generation**: Ο HTML κώδικας παράγεται κατά την διάρκεια της μεταγλώττισης και χρησιμοποιείται ξανά σε κάθε request.
2. **Server-side rendering**: Ο HTML κώδικας παράγεται σε κάθε request.

Το Next.js επιτρέπει ποια μορφή προεπεξεργασίας θα χρησιμοποιηθεί για κάθε σελίδα. Μία υβριδική Next.js εφαρμογή μπορεί να χρησιμοποιεί Static generation για τις περισσότερες σελίδες και Server-side rendering για τις υπόλοιπες. Συστήνεται το Static generation για λόγους επιδόσεων, γιατί οι σελίδες που παράγονται με Static generation μπορούν να αποθηκευτούν στην cache από CDNs και με αυτό τον τρόπο να ενισχυθούν οι επιδόσεις [20]. Όμως, σε μερικές περιπτώσεις το Server-side rendering μπορεί να είναι η μοναδική επιλογή. Επιπλέον, είναι δυνατόν κάποια τμήματα της σελίδας να υφίστανται επεξεργασία αποκλειστικά από την JavaScript στην πλευρά του client [20].

Μερικές σελίδες χρειάζονται την λήψη εξωτερικών δεδομένων (fetching) για την προεπξεργασία. Υπάρχουν δύο σενάρια και σε κάθε περίπτωση χρησιμοποιούνται αναλόγως οι συνάρτησεις του Next.js [20]:

1. Το περιεχόμενο της σελίδας εξαρτάται από εξωτερικά δεδομένα. Για παράδειγμα, μία σελίδα blog χρειάζεται να λάβει τη λίστα με τα posts από ένα CMS. Για να ληφθούν τα δεδομένα στην προεπεξεργασία, το Next.js επιτρέπεται την εξαγωγή μίας ασύγχρονης συνάρτησης που ονομάζεται “getStaticProps” από το ίδιο αρχείο. Η συνάρτηση καλείται κατά την διάρκεια της μεταγλώττισης και επιτρέπεται να περάσουν τα ληφθέντα δεδομένα στις ιδιότητες της σελίδας.

```
function Blog({ posts }) {
  // Render posts...
}

// This function gets called at build time
export async function getStaticProps() {
  // Call an external API endpoint to get posts
  const res = await fetch('https://.../posts')
  const posts = await res.json()

  // By returning { props: { posts } }, the Blog component
  // will receive `posts` as a prop at build time
  return {
    props: {
      posts,
    },
  }
}

export default Blog
```

Εικόνα 14 Παράδειγμα λήψης εξωτερικών δεδομένων από σελίδα με τη συνάρτηση getStaticProps() στο Next.js

2. Οι διαδρομές της σελίδας και η προεπεξεργασία που υφίστανται εξαρτώνται από εξωτερικά δεδομένα. Προκειμένου να διαχειριστεί αυτό το θέμα, το Next.js επιτρέπει την εξαγωγή μίας ασύγχρονης συνάρτησης που ονομάζεται “getStaticPaths” από μία δυναμική σελίδα. Η συνάρτηση καλείται κατά την διάρκεια της μεταγλώττισης και καθορίζει ποιες διαδρομές θα υποστούν προεπεξεργασία. Στο παράδειγμα της σελίδας “pages/posts/[id].js”, εξάγεται επιπλέον η συνάρτηση getStaticProps() προκειμένου να ληφθούν τα δεδομένα σχετικά με το post με το συγκεκριμένο id, το οποίο θα χρησιμοποιηθεί στην προεπεξεργασία της σελίδας.

```
// This function gets called at build time
export async function getStaticPaths() {
  // Call an external API endpoint to get posts
  const res = await fetch('https://.../posts')
  const posts = await res.json()

  // Get the paths we want to pre-render based on posts
  const paths = posts.map((post) => ({
    params: { id: post.id },
  }))

  // We'll pre-render only these paths at build time.
  // { fallback: false } means other routes should 404.
  return { paths, fallback: false }
}
```

Εικόνα 15 Παράδειγμα ασύγχρονης συνάρτησης getStaticPaths() στο Next.js [20]

```
function Post({ post }) {
  // Render post...
}

export async function getStaticPaths() {
  // ...
}

// This also gets called at build time
export async function getStaticProps({ params }) {
  // params contains the post `id`.
  // If the route is like /posts/1, then params.id is 1
  const res = await fetch(`https://.../posts/${params.id}`)
  const post = await res.json()

  // Pass post data to the page via props
  return { props: { post } }
}

export default Post
```

Εικόνα 16 Παράδειγμα ταυτόχρονης χρήσης των ασύγχρονων συναρτήσεων getStaticProps() και getStaticPaths() στο Next.js [20]

Προκειμένου να χρησιμοποιηθεί το Server-side rendering σε μία σελίδα, πρέπει να εξαχθεί μία ασύγχρονη συνάρτηση που ονομάζεται “getServerSideProps”. Η συνάρτηση αυτή καλείται από τον server σε κάθε request [20]. Για παράδειγμα, αν η σελίδα χρειάζεται να προεπεξεργαστεί πρόσφατα ανανεωμένα δεδομένα, πρέπει να χρησιμοποιηθεί η συνάρτηση getServerSideProps(), η οποία λαμβάνει τα δεδομένα και τα περνάει στη σελίδα σε κάθε request.

```
function Page({ data }) {
  // Render data...
}

// This gets called on every request
export async function getServerSideProps() {
  // Fetch data from external API
  const res = await fetch(`https://.../data`)
  const data = await res.json()

  // Pass data to the page via props
  return { props: { data } }
}

export default Page
```

Εικόνα 17 Παράδειγμα της ασύγχρονης συνάρτησης getServerSideProps() στο Next.js [20]

Ενσωματωμένη υποστήριξη CSS στο Next.js

Το Next.js υποστηρίζει την εισαγωγή CSS αρχείων από ένα JavaScript αρχείο. Για την προσθήκη ενός stylesheet στην εφαρμογή, εισάγεται το CSS αρχείο με την εντολή import στο αρχείο “pages/_app.js”. Για παράδειγμα, δημιουργείται το stylesheet με όνομα “styles.css” και κατόπιν εισάγεται στο αρχείο “_app.js” [20].

```
body {
  font-family: 'SF Pro Text', 'SF Pro Icons', 'Helvetica Neue', 'Helvetica',
    'Arial', sans-serif;
  padding: 20px 20px 60px;
  max-width: 680px;
  margin: 0 auto;
}
```

Εικόνα 18 Παράδειγμα αρχείου stylesheet στο Next.js [20]

```
import '../styles.css'

// This default export is required in a new `pages/_app.js` file.
export default function MyApp({ Component, pageProps }) {
  return <Component {...pageProps} />
}
```

Εικόνα 19 Παράδειγμα εισαγωγής stylesheet CSS στο αρχείο _app.js στο Next.js [20]

Τα stylesheets εφαρμόζονται σε όλες τις σελίδες και τα components της εφαρμογής. Λόγω της φύσης των stylesheets, πρέπει να εισάγονται μόνο στο _app.js. Κατά την διάρκεια της παραγωγής (production), όλα τα αρχεία CSS συγχωνεύονται σε ένα ενιαίο αρχείο CSS [20].

Επιπλέον, επιτρέπεται η εισαγωγή ενός CSS αρχείου από τα node modules οπουδήποτε στην εφαρμογή. Για καθολικά stylesheets, όπως το bootstrap ή το nprogress, πρέπει να γίνει πρώτα εισαγωγή του αρχείου CSS στο _app.js. Για την εισαγωγή CSS που απαιτείται από third-party component, η εισαγωγή του αρχείου CSS μπορεί να γίνει στο component [20].

```jsx
// components/ExampleDialog.js
import { useState } from 'react'
import { Dialog } from '@reach/dialog'
import VisuallyHidden from '@reach/visually-hidden'
import '@reach/dialog/styles.css'

function ExampleDialog(props) {
  const [showDialog, setShowDialog] = useState(false)
  const open = () => setShowDialog(true)
  const close = () => setShowDialog(false)

  return (
    <div>
      <button onClick={open}>Open Dialog</button>
      <Dialog isOpen={showDialog} onDismiss={close}>
        <button className="close-button" onClick={close}>
          <VisuallyHidden>Close</VisuallyHidden>
          <span aria-hidden>x</span>
        </button>
        <p>Hello there. I am a dialog</p>
      </Dialog>
    </div>
  )
}
```

Εικόνα 20 Παράδειγμα εισαγωγής CSS αρχείου σε Component στο Next.js [20]

Το Next.js υποστηρίζει CSS modules τα οποία λαμβάνουν όνομα σύμφωνα με το πρότυπο “[name].module.css”. Τα CSS module καθορίζουν τοπικά τους κανόνες CSS δημιουργώντας αυτόματα μοναδικές κλάσεις ονομάτων. Αυτό επιτρέπει να χρησιμοποιηθούν τα ίδια όνοματα κλάσεων CSS σε διαφορετικά αρχεία χωρίς να υπάρχουν εμπλοκές [20]. Αυτή η συμπεριφορά καθιστά τα CSS modules ιδανικά για τον καθορισμό των κανόνων CSS στα Components. Με άλλα λόγια, μπορεί να εισαχθεί απευθείας το stylesheet στο Component οπουδήποτε στην εφαρμογή [20].

```css
/*
You do not need to worry about .error {} colliding with any other `.css` or
`.module.css` files!
*/
.error {
  color: white;
  background-color: red;
}
```

Εικόνα 21 Παράδειγμα CSS module στο Next.js [20]

```
import styles from './Button.module.css'

export function Button() {
  return (
    <button
      type="button"
      // Note how the "error" class is accessed as a property on the imported
      // `styles` object.
      className={styles.error}
    >
      Destroy
    </button>
  )
}
```

Εικόνα 22 Παράδειγμα εφαρμογής CSS module σε Component στο Next.js [20]

Κατά την διάρκεια της παραγωγής τα αρχεία CSS modules συγχωνεύονται αυτόματα σε πολλά ελαχιστοποιημένα αρχεία CSS. Αυτά τα αρχεία CSS αναπαριστούν διαδρομές στην εφαρμογή και εξασφαλίζουν ότι φορτώνεται η ελάχιστη ποσότητα CSS για την εκτέλεση της εφαρμογής [20].

Layouts στο Next.js

Το μοντέλο React συνθέτει την σελίδα με μία σειρά από Components. Πολλά από αυτά τα Components χρησιμοποιούνται σε περισσότερες από μία σελίδες. Για παράδειγμα, μπορεί να χρησιμοποιείται το ίδιο Component μπάρας πλοήγησης και το ίδιο Component για το footer σε κάθε σελίδα [20].

```
// components/layout.js

import Navbar from './navbar'
import Footer from './footer'

export default function Layout({ children }) {
  return (
    <>
      <Navbar />
      <main>{children}</main>
      <Footer />
    </>
  )
}
```

Εικόνα 23 Παράδειγμα δήλωσης Components στο layout στο Next.js [20]

Στην περίπτωση που υπάρχει ένα μόνο layout για την εφαρμογή, δημιουργείται ένα layout και εισάγεται στο _app.js. Επειδή το Component <Layout /> επαναχρησιμοποιείται όταν αλλάζει η σελίδα, η κατάστασή του διατηρείται σταθερή. Στην περίπτωση που υπάρχουν πολλά layouts για την εφαρμογή, προστίθεται η ιδιότητα “getLayout” στη σελίδα. Με αυτόν τον τρόπο επιστρέφεται ένα Component για το layout και προσδιορίζεται διαφορετικό layout για κάθε σελίδα [20]. Κατά την πλοήγηση μεταξύ των σελίδων, είναι επιθυμητό να παραμένει σταθερή η κατάσταση της σελίδας (π.χ. τιμές εισόδου, θέση scroll) προκειμένου να επιτευχθεί εμπειρία Single-Page Application. Η προηγούμενη τεχνική του layout ενεργοποιεί την εμπειρία Single-Page Application καθώς διατηρείται το δέντρο των React Components κατά την πλοήγηση μεταξύ των σελίδων και η React μπορεί να καταλάβει ποια στοιχεία άλλαξαν, με αποτέλεσμα να να διατηρήσει την κατάσταση της σελίδας [20].

```jsx
// pages/index.js

import Layout from '../components/layout'
import NestedLayout from '../components/nested-layout'

export default function Page() {
  return {
    /** Your content */
  }
}

Page.getLayout = function getLayout(page) {
  return (
    <Layout>
      <NestedLayout>{page}</NestedLayout>
    </Layout>
  )
}
```

```jsx
// pages/_app.js

export default function MyApp({ Component, pageProps }) {
  // Use the layout defined at the page level, if available
  const getLayout = Component.getLayout || ((page) => page)

  return getLayout(<Component {...pageProps} />)
}
```

Εικόνα 24 Παράδειγμα πολλαπλών layouts σε εφαρμογή στο Next.js [20]

Static File Serving στο Next.js

Το Next.js μπορεί να παρέχει στατικά αρχεία, όπως εικόνες, από τον φάκελο “public” που βρίσκεται στον ριζικό φάκελο. Τα αρχεία του φακέλου “public” μπορούν να κληθούν από τον κώδικα με χρήση του URL που ξεκινάει με τον χαρακτήρα “/”. Ο συγκεκριμένος φάκελος είναι επίσης χρήσιμος για το “robots.txt”, για το “favicon.ico”, για Google Site Verification και κάθε άλλο στατικό αρχείο [20]. Ο φάκελος “public” δεν επιτρέπεται να αλλάξει όνομα καθώς είναι ο μοναδικός φάκελος, ο οποίος χρησιμοποιείται για την παροχή στατικών αρχείων [20].

```
import Image from 'next/image'

function Avatar() {
  return <Image src="/me.png" alt="me" width="64" height="64" />
}

export default Avatar
```

Εικόνα 25 Παράδειγμα χρήσης στατικού αρχείου από τον φάκελο "public" στο Next.js [20]

Το Script Component

Οι ιστοσελίδες συχνά χρειάζονται third-party scripts για analytics, διαφημίσεις, widgets υποστήριξης πελατών και αποδοχή όρων χρήσης. Όμως, αυτά τα scripts χρειάζονται αρκετό χρόνο για να φορτωθούν και επηρεάζουν την εμπειρία του χρήστη. Το Script Component στο Next.js δίνει την δυνατότητα στους προγραμματιστές να καθορίσουν την προτεραιότητα φόρτωσης των third-party scripts και έτσι βελτιώνεται ο χρόνος φόρτωσης [20].

Με το “next/script” προσδιορίζεται η ιδιότητα “strategy” και το Next.js βελτιστοποιεί την φόρτωση του script [20]:

- “beforeInteractive”: Αφορά scripts που πρέπει να ληφθούν και να εκτελεστούν προτού η σελίδα γίνει διαδραστική, όπως scripts για την διαχείριση των bots και για την αποδοχή των όρων χρήσης.
- “afterInteractive” (προεπιλογή): Αφορά scripts που πρέπει να ληφθούν και να εκτελεστούν αφού η σελίδα γίνει διαδραστική, όπως scripts για την διαχείριση των tag και για την αποδοχή των όρων χρήσης.
- “lazyOnLoad” : Αφορά scripts που μπορούν να περιμένουν να φορτωθούν κατά τη διάρκεια του χρόνου αδράνειας, όπως scripts για chat και widgets για social media.

```
import Script from 'next/script'

export default function Home() {
  return (
    <>
      <Script
        src="https://polyfill.io/v3/polyfill.min.js?features=IntersectionO
        strategy="beforeInteractive"
      />
    </>
  )
}
```

Εικόνα 26 Παράδειγμα script με φόρτωση προτού η σελίδα γίνει διαδραστική στο Next.js [20]

```
// After

// pages/index.js
import Script from 'next/script'

export default function Home() {
  return (
    <>
      <Script src="https://www.google-analytics.com/analytics.js" />
    </>
  )
}
```

Εικόνα 27 Παράδειγμα script με φόρτωση αφού η σελίδα γίνει διαδραστική στο Next.js [20]

```
import Script from 'next/script'

export default function Home() {
  return (
    <>
      <Script
        src="https://connect.facebook.net/en_US/sdk.js"
        strategy="lazyOnload"
      />
    </>
  )
}
```

Εικόνα 28 Παράδειγμα script με φόρτωση lazy-loading [20]

Γρήγορο Refresh και Διαχείριση Σφαλμάτων

Το γρήγορο Refresh είναι ένα χαρακτηριστικό του Next.js το οποίο παρέχει στιγμιαίο feedback για τις αλλαγές που λαμβάνουν χώρα στα React Components. Με ενεργοποιημένο του γρήγορο Refresh, οι αλλαγές γίνονται ορατές άμεσα, χωρίς να μεταβάλλεται η κατάσταση των Components [20].

Στην περίπτωση που τροποποιηθεί ένα αρχείο που μόνο εξάγει React Components, το γρήγορο Refresh θα ενημερώσει τον κώδικα για το αρχείο αυτό και θα επανεπεξεργαστεί το Component. Οι αλλαγές περιλαμβάνουν ο,τιδήποτε από μεταβολές στο style εώς αλλαγές στη δομή του κώδικα και αλλαγές σε event handlers [20]. Στην περίπτωση που τροποποιηθεί αρχείο που δεν εξάγει Component, το γρήγορο Refresh θα εκτελέσει αυτό το αρχείο, αλλά και τα αρχεία στα οποία γίνεται εισαγωγή αυτό το αρχείο. Στην περίπτωση που τροποποιηθεί αρχείο που εισάγεται από αρχείο εκτός του δέντρου React, το γρήγορο Refresh θα εκτελέσει μία πλήρη φόρτωση των αρχείων. Για παράδειγμα, μπορεί ένα αρχείο να επεξεργάζεται ένα React Component και ταυτόχρονα να εξάγει μία τιμή, η οποία εισάγεται από ένα Component που δεν ανήκει στο δέντρο React [20].

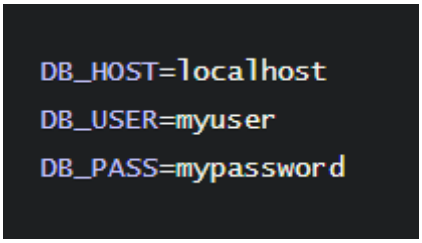
Τα συντακτικά λάθη κατά τη διάρκεια της ανάπτυξης της εφαρμογής επιλύονται με την διόρθωση τους και την εκ νέου αποθήκευση της εφαρμογής. Με αυτόν τον τρόπο, το σφάλμα εξαφανίζεται και δεν απαιτείται η επαναφόρτωση της εφαρμογής, ενώ διατηρείται η κατάσταση των Components [20]. Στην περίπτωση που υπάρχει λάθος στην εφαρμογή, το οποίο προκαλεί σφάλμα εκτέλεσης, η κατάσταση των Components διατηρείται μόνο αν το σφάλμα δεν συνέβη κατά την διάρκεια της επεξεργασίας. Μία καλή τεχνική για την εύρεση των σφαλμάτων είναι η χρήση Error Boundaries, τα οποία είναι React Components που εντοπίζουν και καταγράφουν σφάλματα JavaScript καθ'όλη την διάρκεια επεξεργασίας και του κύκλου ζωής του Component [20].

Το γρήγορο Refresh επιχειρεί να διατηρήσει τοπικά την κατάσταση του Component που υφίσταται επεξεργασία, εκτός εάν [20]:

- Η κατάσταση δεν αποθηκεύεται για το συγκεκριμένο Component.
- Το αρχείο εξάγει και άλλες τιμές εκτός από αυτή του Component.
- Το επιστρεφόμενο Component είναι τύπου κλάσης.
- Εφαρμόζονται ανώνυμες συναρτήσεις.

Μεταβλητές Περιβάλλοντος

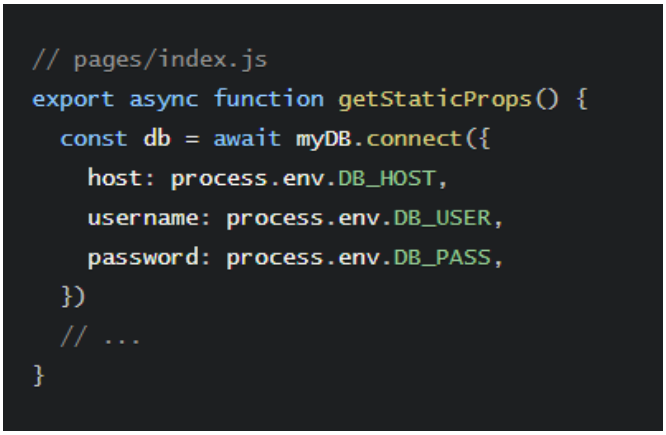
Το Next.js υποστηρίζει την φόρτωση μεταβλητών περιβάλλοντος από το ".env.local" στο "process.env".



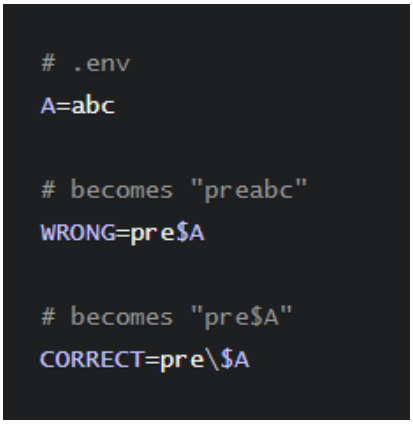
Εικόνα 29 Παράδειγμα ".env.local" στο Next.js [20]

Το ανωτέρω παράδειγμα φορτώνει τις μεταβλητές “process.env.DB_HOST”, “process.env.DB_USER” και “process.env.DB_PASS” αυτόματα στο περιβάλλον Node.js, πράγμα το οποίο επιτρέπει την χρήση των μεταβλητών από μεθόδους λήψης δεδομένων (data fetching methods) και διαδρομές API [20].

Προκειμένου να διατηρηθούν τα δεδομένα που αφορούν τον server ασφαλή, το Next.js αντικαθιστά το “process.env.*” με τις σωστές τιμές κατά την διάρκεια της μεταγλώττισης. Αυτό σημαίνει ότι το “process.env” δεν είναι ένα τυπικό αντικείμενο JavaScript [20].



Εικόνα 30 Παράδειγμα κλήσης μεταβλητών περιβάλλοντος από ασύγχρονη συνάρτηση στο Next.js [20]



Εικόνα 31 Παράδειγμα αναφοράς μεταβλητών περιβάλλοντος εντός άλλων μεταβλητών περιβάλλοντος στο Next.js [20]

Οι μεταβλητές περιβάλλοντος φορτώνονται μέσω του ".env.local" στο περιβάλλον Node.js και δεν είναι διαθέσιμες στον browser. Προκειμένου μία μεταβλητή περιβάλλοντος να εμφανιστεί στον browser, πρέπει να χρησιμοποιήσει το πρόθεμα "NEXT_PUBLIC_" [20].

Γενικά χρειάζεται μόνο ένα αρχείο ".env.local". Μερικές φορές όμως μπορεί να χρειαστεί να προστεθούν μεταβλητές περιβάλλοντος για την ανάπτυξη ή την παραγωγή της εφαρμογής. Σε αυτή την περίπτωση ορίζονται τα αρχεία ".env.development" ή ".env.production". Εκτός από τα περιβάλλοντα ανάπτυξης και παραγωγής της εφαρμογής, μπορεί να οριστεί επίσης περιβάλλον για την δοκιμή της εφαρμογής μέσω του αρχείου ".env.test". Το τελευταίο είναι χρήσιμο στην περίπτωση που εκτελούνται εργαλεία όπως το jest ή το cypress, όπου απαιτείται ο ορισμός μεταβλητών περιβάλλοντος για τον έλεγχο της εφαρμογής [20]. Οι προεπιλεγμένες τιμές για τις δοκιμές φορτώνονται αν η παράμετρος "NODE_ENV" λάβει την τιμή "test". Όμοια με τις προεπιλεγμένες μεταβλητές περιβάλλοντος, το αρχείο ".env.test" πρέπει να συμπεριληφθεί στο αποθετήριο, αλλά όχι το αρχείο ".env.test.local", γιατί τα αρχεία ".env*.local" παραβλέπονται λόγω του ".gitingore". Αξίζει να σημειωθεί ότι για την εκτέλεση unit tests, οι μεταβλητές περιβάλλοντος φορτώνονται με χρήση της συνάρτησης "loadEnvConfig" από το πακέτο "@next/env" [20].

```
// The below can be used in a Jest global setup file or similar for your test
import { loadEnvConfig } from '@next/env'

export default async () => {
  const projectDir = process.cwd()
  loadEnvConfig(projectDir)
}
```

Εικόνα 32 Φόρτωση μεταβλητών περιβάλλοντος για test της εφαρμογής με χρήση της συνάρτησης loadEnvConfig() στο Next.js [20]

Διαδρομές API

Οι διαδρομές API παρέχουν μία λύση για την κατασκευή του API με το Next.js. Οποιοδήποτε αρχείο στο φάκελο "pages/api" αντιστοιχεί στο "/api/*" και αντιμετωπίζεται σαν API endpoint αντί για σελίδα [20]. Για να λειτουργήσει μία διαδρομή API πρέπει να εξαχθεί μία συνάρτηση ή αλλιώς request handler, η οποία στην συνέχεια λαμβάνει τις παρακάτω παραμέτρους [20]:

- req: Είναι το http.IncomingMessage συν κάποια προκατασκευασμένα middleware, ή αλλιώς η είσοδος.
- res: Είναι το http.ServerResponse συν κάποιες συναρτήσεις τύπου helper, ή αλλιώς η απάντηση του server.

Για τον χειρισμό διαφορετικών HTTP μεθόδων σε μία API διαδρομή, χρησιμοποιείται η μέθοδος req.method.

```
export default function handler(req, res) {
  if (req.method === 'POST') {
    // Process a POST request
  } else {
    // Handle any other HTTP method
  }
}
```

Εικόνα 33 Παράδειγμα χειρισμού HTTP μεθόδων σε μία διαδρομή API στο Next.js [20]

Οι διαδρομές API υποστηρίζουν δυναμικές διαδρομές (dynamic routes) και ακολουθούν τους ίδιους κανόνες ονοματοδοσίας που ισχύουν για τις σελίδες. Για παράδειγμα, η δυναμική διαδρομή “pages/api/post/[pid].js” έχει τον παρακάτω κώδικα με αποτέλεσμα η απόκριση στο request “api/post/abc” να είναι “Post: abc”.

```
export default function handler(req, res) {
  const { pid } = req.query
  res.end(`Post: ${pid}`)
}
```

Εικόνα 34 Παράδειγμα δυναμικής διαδρομής API στο Next.js [20]

Μία κοινή RESTful τεχνική για τον καθορισμών διαδρομών είναι η ακόλουθη:

- ✓ GET api/posts: Λήψη της λίστας με όλα τα posts.
- ✓ GET api/posts/12345: Λήψη του post με id 12345.

Αυτό μοντελοποιείται με δύο τρόπους:

1. /api/posts.js, /api/posts/[postId].js
2. /api/posts/index.js, /api/posts/[postId].js

Και οι δύο τρόποι είναι ισοδύναμοι. Η χρήση μόνο του /api/posts/[postId].js δεν είναι έγκυρη γιατί οι δυναμικές διαδρομές δεν μπορούν να λάβουν ακαθόριστες τιμές και το GET api/posts δεν θα ταιριάζει με το /api/posts/[postId.js].

Παραγωγή της εφαρμογής

Πριν από το στάδιο της παραγωγής της εφαρμογής καλό θα ήταν να εφαρμοστούν τα παρακάτω προκειμένου να βελτιωθεί η εμπειρία του χρήστη [20]:

- ✓ Χρήση caching όπου είναι εφικτό.
- ✓ Η ανάπτυξη της βάσης δεδομένων και του back-end λαμβάνουν χώρα στην ίδια περιοχή.
- ✓ Χρήση της λιγότερης δυνατής JavaScript.
- ✓ Αποφυγή φόρτωσης μεγάλου κώδικα JavaScript αν δεν είναι απολύτως απαραίτητο.
- ✓ Ενεργοποίηση του logging.
- ✓ Η διαχείριση σφαλμάτων (error handling) είναι ρυθμισμένη.
- ✓ Ρύθμιση των σελίδων 404 (Not Found) και 500 (Error Page).
- ✓ Μέτρηση επιδόσεων εφαρμογής.

Το caching βελτιώνει τους χρόνους απόκρισης και μειώνει τον αριθμό των requests σε εξωτερικές υπηρεσίες. Το Next.js προσθέτει αυτόματα caching headers στα αμετάβλητα αντικείμενα που εξυπηρετούνται από την διαδρομή "_next/static" συμπεριλαμβανομένων αντικειμένων JavaScript, CSS, στατικών εικόνων και άλλων πολυμέσων [20]. Οι Cache-Control headers που ορίζονται στο "next.config.js" επανεγγράφονται κατά την παραγωγή ούτως ώστε τα στατικά αντικείμενα να μπορούν να κληθούν αποτελεσματικά. Αν χρειάζεται η επιβεβαίωση της cache μίας σελίδας που έχει δημιουργηθεί στατικά, τότε ορίζεται η παράμετρος "revalidate" στην συνάρτηση getStaticProps() της σελίδας. Αν χρησιμοποιείται το "next/image", υπάρχουν επίσης συγκεκριμένοι κανόνες caching για τον προεπιλεγμένο βελτιστοποιητή εικόνων (Image Optimization loader) [20]. Στην περίπτωση που η εφαρμογή εκτελείται τοπικά με το "next dev", οι headers επανεγγράφονται για να αποτραπεί τοπικό caching.

Προκειμένου να μειωθεί η ποσότητα JavaScript που αποστέλλεται στον browser, χρησιμοποιούνται τα παρακάτω εργαλεία για να γίνει κατανοητό τι περιέχει κάθε δέσμη κώδικα JavaScript [20]:

- ✓ Import Cost: Απεικονίζει το μέγεθος του εισαγόμενου πακέτου στο VSCode.
- ✓ Package Phobia: Βρίσκει το κόστος της προσθήκης μίας νέας dependency στο project.
- ✓ Bundle Phobia: Αναλύει κατά πόσο μία dependency μπορεί να αυξήσει το μέγεθος του κώδικα.
- ✓ Webpack Bundle Analyzer: Απεικονίζει σε ένα διαδραστικό δέντρο το μέγεθος των αρχείων εξόδου του πακέτου Webpack.

Κάθε αρχείο στον φάκελο "pages/" αυτομάτως παράγει την δική του δέσμη κώδικα JavaScript κατά την εκτέλεση της εντολής "next build". Μπορούν επίσης να χρησιμοποιηθούν

δυναμικές εισαγωγές (dynamic imports) για την φόρτωση των Components και των βιβλιοθηκών με την τεχνική lazy-loading [20].

Επειδή το Next.js εκτελείται τόσο στον server όσο και στον client, το logging μπορεί να πραγματοποιηθεί στον browser μέσω του “console.log” και στον server μέσω του “stdout” [20]. Επιπλέον για την καταγραφή μπορεί να χρησιμοποιηθεί το πακέτο “Pino”.

Component next/link

Οι μεταβάσεις μεταξύ των διαδρομών στην πλευρά του client μπορούν να ενεργοποιηθούν μέσω του Component “Link” το οποίο εξάγεται από το next/link. Για παράδειγμα, αν θεωρηθεί ο φάκελος pages με τα αρχεία “pages/index.js”, “pages/about.js” και “pages/blog/[slug].js”, μπορεί να υπάρχει ένα link για κάθε σελίδα όπως στο παράδειγμα της Εικόνας 35 [20].

Το “Link” δέχεται τις ακόλουθες παραμέτρους [20]:

- href: Η διαδρομή η του URL στο οποίο είναι επιθυμητή η πλοήγηση.
- as: Προαιρετικός decorator για την διαδρομή που θα εμφανιστεί στην μπάρα διευθύνσεων του browser.
- passHref: Αναγκάζει το “Link” να στείλει την ιδιότητα “href” στο «παιδί» του. Η προεπιλεγμένη τιμή είναι false.
- prefetch: Προφόρτωση της σελίδας στο background. Η προεπιλεγμένη τιμή της παραμέτρου είναι true. Η προφόρτωση ενεργοποιείται στο στάδιο της παραγωγής.
- replace: Αντικαθιστά την τρέχουσα κατάσταση του “history” αντί να προστίθεται ένα νέο url στην στοίβα. Η προεπιλεγμένη τιμή της παραμέτρου είναι false.
- scroll: Επιστροφή στην κορυφή της σελίδας μετά το τέλος της πλοήγησης. Η προεπιλεγμένη τιμή της παραμέτρου είναι true.
- shallow: Ενημερώνει την διαδρομή της τρέχουσας σελίδας χωρίς να εκτελούνται ξανά οι συναρτήσεις getStaticProps(), getServerSideProps() και getInitialProps(). Η προεπιλεγμένη τιμή της παραμέτρου είναι false.
- locale: Επιτρέπει τον προσδιορισμό διαφορετικού locale.

Το “Link” μπορεί επίσης να λάβει ένα URL αντικείμενο και να το διαμορφώσει έτσι ώστε να σχηματιστεί το string του url. Το παράδειγμα της Εικόνας 36 έχει μία προκαθορισμένη διαδρομή (“/about?name=test”) και μία δυναμική διαδρομή (“/blog/my-post”) [20].

```
function Home() {
  return (
    <ul>
      <li>
        <Link href="/">
          <a>Home</a>
        </Link>
      </li>
      <li>
        <Link href="/about">
          <a>About Us</a>
        </Link>
      </li>
      <li>
        <Link href="/blog/hello-world">
          <a>Blog Post</a>
        </Link>
      </li>
    </ul>
  )
}

export default Home
```

Εικόνα 35 Παράδειγμα εφαρμογής του Component "Link" στο Next.js [20]

```
import Link from 'next/link'

function Home() {
  return (
    <ul>
      <li>
        <Link
          href={{
            pathname: '/about',
            query: { name: 'test' },
          }}
        >
          <a>About us</a>
        </Link>
      </li>
      <li>
        <Link
          href={{
            pathname: '/blog/[slug]',
            query: { slug: 'my-post' },
          }}
        >
          <a>Blog Post</a>
        </Link>
      </li>
    </ul>
  )
}

export default Home
```

Εικόνα 36 Παράδειγμα διαμόρφωσης URL string και δυναμικής διαδρομής με το Component "Link" στο Next.js [20]

Το αρχείο next.config.js

Για την εξατομικευμένη συμπεριφορά του Next.js, χρησιμοποιείται το αρχείο next.config.js που βρίσκεται στο ριζικό φάκελο της εφαρμογής μαζί με το αρχείο package.json. Το αρχείο next.config.js είναι ένα τυπικό Node.js module και όχι αρχείο JSON. Αν και χρησιμοποιείται από τον Next.js server και σε διάφορα στάδια της μεταγλώττισης, δεν σχετίζεται με την κατασκευή της σελίδας στον browser [20].

```
const { PHASE_DEVELOPMENT_SERVER } = require('next/constants')

module.exports = (phase, { defaultConfig }) => {
  if (phase === PHASE_DEVELOPMENT_SERVER) {
    return {
      /* development only config options here */
    }
  }

  return {
    /* config options for all phases except development here */
  }
}
```

Εικόνα 37 Παράδειγμα αρχείου next.config.js στο Next.js [20]

Υλοποίηση διαδικτυακής εφαρμογής

Για την υλοποίηση της ιστοσελίδας του Εργαστηρίου sae.el.teithe.gr σε React χρησιμοποιήθηκε το theme eDemy [21]. Το eDemy είναι κατάλληλο για εκπαιδευτικές ιστοσελίδες, για ιστοσελίδες εκπαιδευτικών ιδρυμάτων, για παροχή online εκπαίδευσης, LMS (Learning Management System) εφαρμογές κ.α. Το template βασίζεται στις τεχνολογίες ReactJS, NextJS, PostgreSQL, Bootstrap και SASS (Syntactically Awesome Style Sheets) [21]. Η προοδευτική εφαρμογή ιστού (Progressive Web App) ενεργοποιείται με το Stripe Payment το οποίο είναι ενσωματωμένο στο eDemy. Επιπλέον, το eDemy υποστηρίζει την δυναμική διαχείριση μαθημάτων, τις δυναμικές βιντεο-διαλέξεις και τρία επίπεδα χρηστών (Admin, Καθηγητής, Μαθητής) [21].

Η παρούσα εργασία συνοδεύεται με τον κώδικα των σελίδων της διαδικτυακής εφαρμογής του Εργαστηρίου σε React και τον κώδικα των αρχείων του theme που τροποποιήθηκαν.

Προαπαιτούμενα

Για την εκτέλεση της εφαρμογής απαιτείται έκδοση Node.js 10.13 ή νεότερη. Ως ασύγχρονο περιβάλλον εκτέλεσης JavaScript που λειτουργεί με βάση τα events, το Node.js είναι σχεδιασμένο για την ανάπτυξη επεκτάσιμων δικτυακών εφαρμογών [22]. Σε αντίθεση με την τεχνολογία thread-based networking, το Node.js απενεργοποιείται στην περίπτωση που δεν υπάρχει ενεργή σύνδεση ή κλήση. Επιπλέον, οι συναρτήσεις του Node.js δεν εφαρμόζουν απευθείας ενέργειες I/O με αποτέλεσμα να μην μπλοκάρει το Node.js, το οποίο και είναι ο λόγος που συστήνεται για επεκτάσιμες εφαρμογές [22]. Αν και ο σχεδιασμός του Node.js ομοιάζει με το Event Machine της Ruby και το Twisted της Python, εφαρμόζει τον βρόγχο επανάληψης event σαν εκτελέσιμη δομή και όχι ως βιβλιοθήκη. Συγκεκριμένα, στο Node.js δεν υπάρχει κλήση που να ξεκινάει τον βρόγχο επανάληψης των events, αλλά ο βρόγχος επανάληψης εκκινεί με την εκτέλεση του script εισόδου [22]. Το Node.js σταματά τον βρόγχο επανάληψης των events όταν δεν υπάρχουν πλέον callbacks για να εκτελέσει. Επιπρόσθετα η τεχνολογία HTTP χρησιμοποιείται ευρέως στο Node.js παρέχοντας streaming και χαμηλή καθυστέρηση. Αν και το Node.js είναι σχεδιασμένο χωρίς threads, αυτό δεν σημαίνει ότι δεν είναι δυνατή η εκμετάλλευση πολλών επεξεργαστών. Αντιθέτως, το cluster module του Node.js επιτρέπει τον διαμοιρασμό των sockets μεταξύ των διεργασιών με σκοπό να ισορροπηθεί ο φόρτος εργασίας [22].

Το eDemy χρησιμοποιεί για την βάση δεδομένων του το PostgreSQL [23]. Το PostgreSQL είναι ένα σύστημα σχεσιακής βάσης δεδομένων ανοιχτού κώδικα, το οποίο χρησιμοποιεί και επεκτείνει την SQL σε συνδυασμό με χαρακτηριστικά που αποθηκεύουν και επεκτείνουν τις πιο σύνθετες διεργασίες επεξεργασίας δεδομένων [23]. Το PostgreSQL παρέχει αξιοπιστία, ευελιξία (διαχείριση της βάσης δεδομένων ανεξαρτήτου μεγέθους), διασφάλιση της ακεραιότητας των δεδομένων, ανεκτικότητα στα σφάλματα, δυνατότητα δημιουργίας εξατομικευμένων τύπων δεδομένων και εξατομικευμένων συναρτήσεων, συμβατότητα με τις περισσότερες γλώσσες προγραμματισμού και με τα περισσότερα λειτουργικά συστήματα. Μεταξύ άλλων, το PostgreSQL υποστηρίζει JSON δεδομένα, SQL/JSON εκφράσεις, ρουτίνες, συναρτήσεις, Authentication με GSSAPI, SSPI, LDAP, SCRAM-SHA-256 και Certificate [23].

Επιπλέον, χρειάζεται λογαριασμός Cloudinary για την παροχή cloud υπηρεσιών διαχείρισης εικόνων και βίντεο [24] και λογαριασμός Stripe [25] για τις online πληρωμές (στην περίπτωση που λειτουργήσει το online κατάστημα). Αφού δημιουργηθούν οι λογαριασμοί, συμπληρώνονται κατάλληλα οι μεταβλητές περιβάλλοντος στο αρχείο next.config.js στον ριζικό φάκελο όπως φαίνεται παρακάτω:

```
CLOUDINARY_URL: "https://api.cloudinary.com.....",

CLOUDINARY_VIDEO_URL: "https://api.cloudinary.com.....",

STRIPE_SECRET_KEY: "sk_test_3D.....",

STRIPE_PUBLISHABLE_KEY: "pk_test_Zy....."
```

Στη συνέχεια ενημερώνονται οι μεταβλητές upload_preset και cloud_name στα αρχεία “/pages/teacher/course/create.js” και “/pages/teacher/course/upload-course-video.js”. Τέλος ενημερώνεται το αρχείο “/config/config.json” με τις απαραίτητες πληροφορίες όπως usernames και passwords για τα στάδια της παραγωγής, ανάπτυξης και δοκιμής της εφαρμογής:

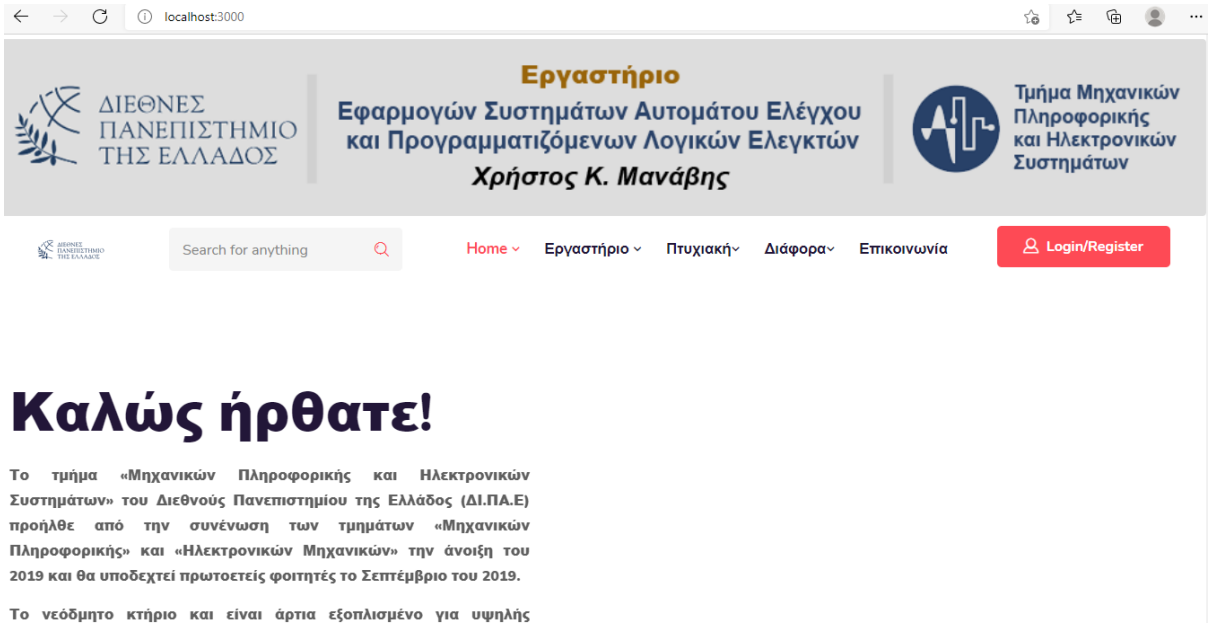
```
{
    "development": {
        "username": "postgres",
        "password": "123456",
        "database": "edemy_app_development",
        "host": "127.0.0.1",
        "dialect": "postgres"
    },
    "test": {
        "username": "postgres",
        "password": "123456",
        "database": "edemy_database_test",
        "host": "127.0.0.1",
        "dialect": "postgres"
    },
    "production": {
        "username": "username",
        "password": "pass&word",
        "database": "edemy_database_production",
        "host": "127.0.0.1",
        "dialect": "postgres"
    }}
```

Για την συγγραφή και την τροποποίηση του κώδικα χρησιμοποιείται το Visual Studio Code [26].

Εκτέλεση εφαρμογής

Για την εκτέλεση της εφαρμογής ακολοθούνται τα εξής βήματα:

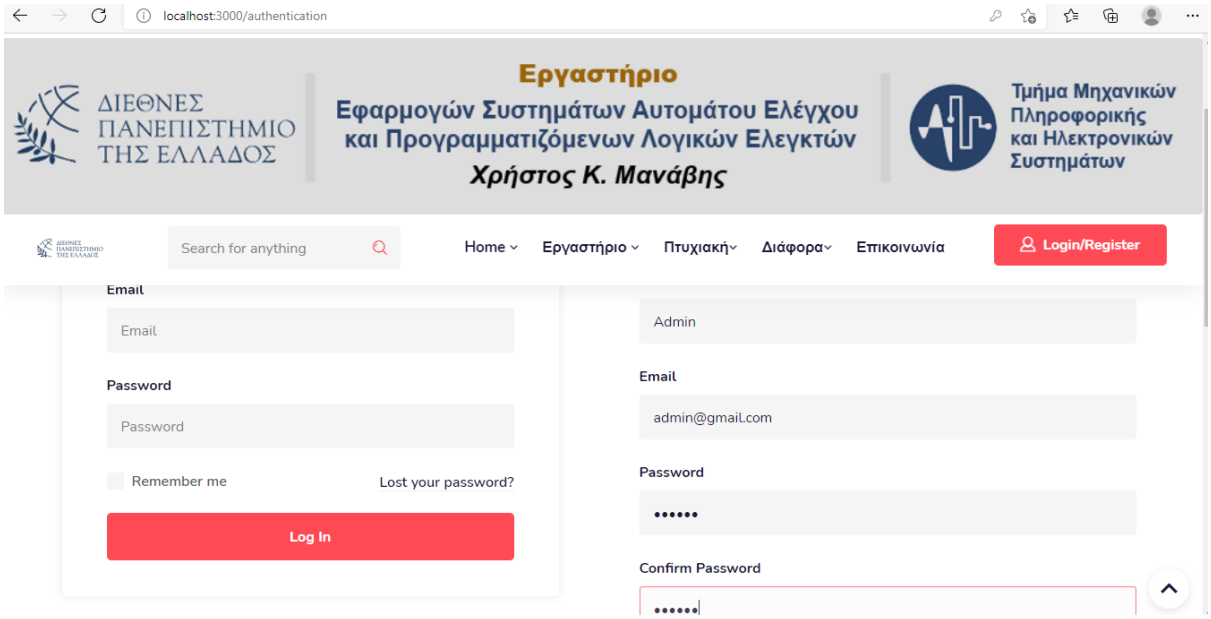
- ✓ Άνοιγμα του ριζικού φακέλου “edemy-react-next” στο Visual Studio Code.
- ✓ Άνοιγμα τερματικού από “Terminal → New Terminal”.
- ✓ Εκτέλεση της εντολής “npm install –legacy-peer-deps” στο τερματικό.
- ✓ Εκτέλεση της εντολής “npx sequelize-cli db:create” στο τερματικό.
- ✓ Εκτέλεση της εντολής “npx sequelize-cli db:migrate” στο τερματικό.
- ✓ Εκτέλεση της εντολής “npm run dev” στο τερματικό.
- ✓ Άνοιγμα της διεύθυνσης http://localhost:3000 στον browser.



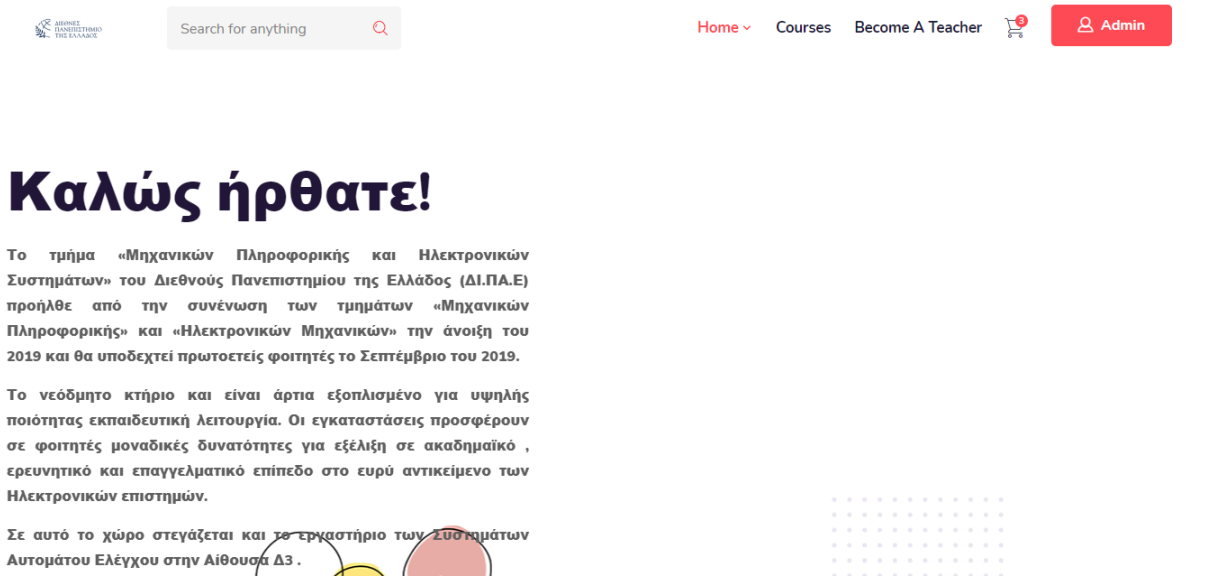
Εικόνα 38 Αρχική σελίδα της διαδικτυακής εφαρμογής του Εργαστηρίου σε React

Δημιουργία λογαριασμού Admin

Για την δημιουργία λογαριασμού επιπέδου admin, αρχικά δημιουργείται ένας λογαριασμός από την σελίδα “Authentication” με τον ίδιο τρόπο που δημιουργείται λογαριασμός για χρήστη επιπέδου “student” και “teacher”.

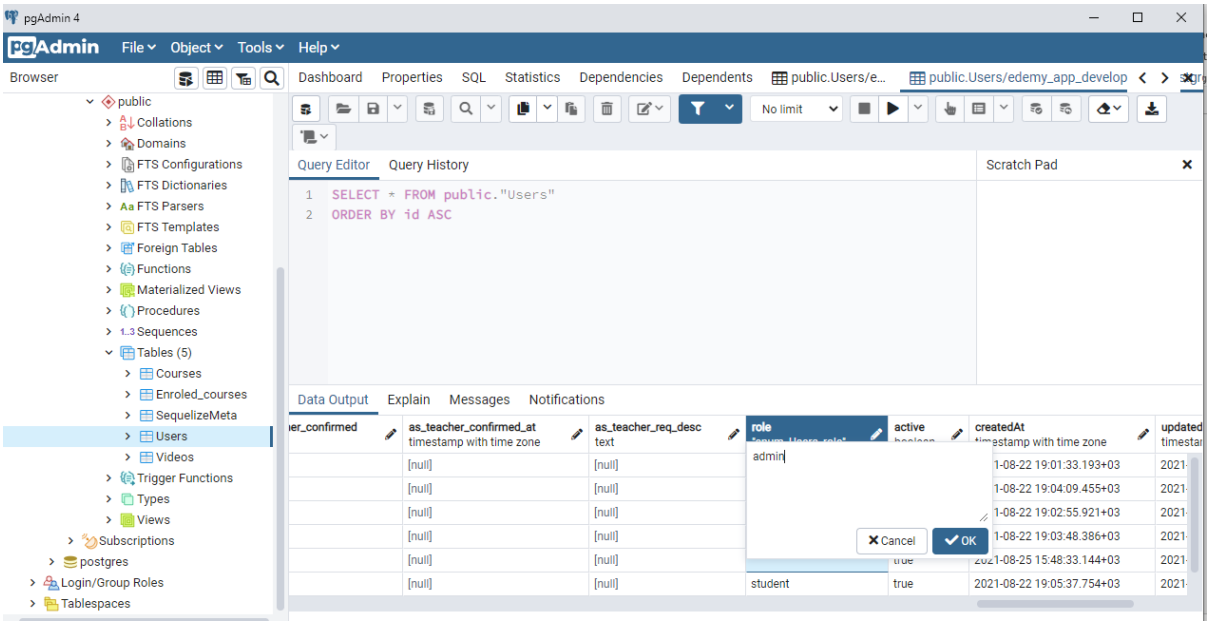


Εικόνα 39 Δημιουργία λογαριασμού admin στη διαδικτυακή εφαρμογή του Εργαστηρίου σε React

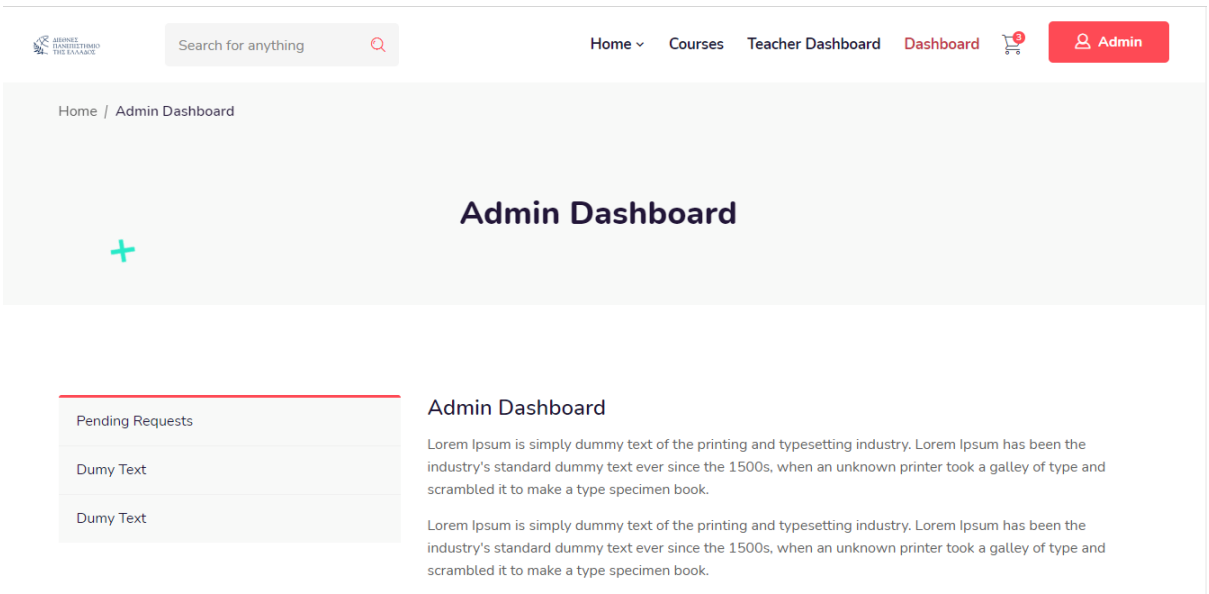


Εικόνα 40 Αρχική σελίδα αφού ο χρήστης κάνει login με τον λογαριασμό του με επίπεδο χρήστη student στη διαδικτυακή εφαρμογή του Εργαστηρίου σε React

Στη συνέχεια εντοπίζεται με το εργαλείο pgAdmin 4 ο νέος λογαριασμός στον πίνακα Users της βάσης δεδομένων της εφαρμογής και τροποποιείται η στήλη “role” έτσι ώστε η τιμή της να είναι “admin” και αποθηκεύονται οι αλλαγές. Συγκρίνοντας τις Εικόνες 40 και 42 παρατηρείται ότι μεταβλήθηκε η μπάρα πλοήγησης αφού άλλαξε το επίπεδο του χρήστη. Επιπλέον, ανάλογα με το επίπεδο του χρήστη, αλλάζουν τα δικαιώματα και οι δυνατότητες του χρήστη στην εφαρμογή.



Εικόνα 41 Αλλαγή του επιπέδου του χρήστη σε admin στη διαδικτυακή εφαρμογή του Εργαστηρίου σε React με το εργαλείο pgAdmin 4



Εικόνα 42 Σελίδες αφού ο χρήστης κάνει login με τον λογαριασμό του με επίπεδο χρήστη admin στη διαδικτυακή εφαρμογή του Εργαστηρίου σε React

Συμπεράσματα

Η νέα ιστοσελίδα του Εργαστήριου σε σύγκριση με την υπάρχουσα διαθέτει σύγχρονη responsive σχεδίαση κάνοντας χρήση των τεχνολογιών ReactJS, NextJS, PostgreSQL, Bootstrap και SASS. Το περιεχόμενο της νέας ιστοσελίδας είναι το ίδιο με αυτό της υπάρχουσας ιστοσελίδας του εργαστηρίου, ενώ η νέα ιστοσελίδα είναι πιο λειτουργική, καθώς συγχωνεύτηκαν ή καταργήθηκαν ενότητες. Συγκεκριμένα, διαθέτει πέντε αντί για οκτώ tabs στην μπάρα πλοήγησης, η οποία δεν είναι στατική αλλά ακολουθεί τον χρήστη κατά το scrolling-down. Οι ενότητες «Υποδομή», «Φωτογραφίες», «Ο Διδάσκων» της υπάρχουσας ιστοσελίδας εμπεριέχονται στα tabs της νέας ιστοσελίδας καθιστώντας την πλοήγηση πιο εύκολη. Επιπλέον, η νέα ιστοσελίδα διαθέτει μπάρα αναζήτησης προκειμένου ο επισκέπτης να εντοπίσει γρήγορα την πληροφορία που επιθυμεί.

Η νέα ιστοσελίδα του Εργαστηρίου διαθέτει footer, το οποίο περιέχει links σε σελίδες της εφαρμογής, πληροφορίες για το Εργαστήριο και παραπομπές σε εξωτερικούς συνδέσμους. Οι προηγούμενες πληροφορίες στην υπάρχουσα ιστοσελίδα είναι διάσπαρτες σε διάφορα σημεία τις ιστοσελίδας δυσκολεύοντας τον χρήστη κατά την πλοήγηση. Επιπρόσθετα, στο footer υπάρχουν εικονίδια που παραπέμπουν στα δημοφιλή social media Facebook, Twitter, Instagram και Linkedin, στα οποία το Εργαστήριο μπορεί να δημιουργήσει προφίλ.

Η νέα ιστοσελίδα του Εργαστηρίου διαθέτει το menu «Διάφορα» που περιέχει τις σελίδες Blog και FAQs. Στη σελίδα Blog εγγράφονται άρθρα για διάφορα θέματα και οι χρήστες μπορούν να αλληλεπιδράσουν με την εφαρμογή γράφοντας σχόλια και λαμβάνοντας απαντήσεις. Στη σελίδα FAQs (Frequently Asked Questions) περιλαμβάνονται τυπικές ερωτήσεις και απαντήσεις για βασικά θέματα της εφαρμογής. Η τελευταία ενδείκνυται για την αντικατάσταση αρκετών σελίδων της υπάρχουσας ιστοσελίδας όπως οι «Γενικά», «Πρόγραμμα Διδασκαλίας», «Κανονισμός – Αξιολόγηση» και «Εξετάσεις». Επίσης η νέα ιστοσελίδα διαθέτει φόρμα επικοινωνίας στη σελίδα «Επικοινωνία» προκειμένου οι επισκέπτες να επικοινωνούν άμεσα με το Εργαστήριο.

Η νέα ιστοσελίδα του Εργαστηρίου διαθέτει τρία επίπεδα χρηστών: Μαθητής, Καθηγητής, Διαχειριστής. Ο χρήστης μπορεί να εγγραφεί μέσω της φορμάς SignUp και να πραγματοποιήσει Login με τα προσωπικά του στοιχεία. Ανάλογα με το επίπεδο του χρήστη, ο χρήστης έχει διαβαθμισμένα δικαιώματα και πρόσβαση σε διαφορετικό υλικό. Για παράδειγμα, ο Διαχειριστής μπορεί να αλλάξει το επίπεδο του χρήστη, ο Καθηγητής να ορίσει Μαθήματα και να ανεβάσει αρχεία όπως βίντεο και αρχεία κειμένου, και ο Μαθητής να αποκτήσει πρόσβαση σε συγκεκριμένα μαθήματα και στο αντίστοιχο υλικό.

Η νέα ιστοσελίδα του Εργαστήριου αποτελεί ένα ολοκληρωμένο Σύστημα Διαχείρισης Μάθησης (LMS), καθώς παρέχει την δυνατότητα για δυναμική διαχείριση μαθημάτων, για δυναμικές βιντεο-διαλέξεις, ειδικές ενότητες για τη δημιουργία εκπαιδευτικών κατευθύνσεων, βαθμίδες εγγραφής μελών, ηλεκτρονικό κατάστημα και Gateway για την πώληση προϊόντων π.χ. Online μαθημάτων. Το γεγονός ότι η εφαρμογή

βασίζεται σε REST API παρέχει την δυνατότητα μεταφοράς δεδομένων και ενσωμάτωσης της εφαρμογής σε οποιαδήποτε άλλη διαδικτυακή εφαρμογή.

Κατά την ανάπτυξη της νέας ιστοσελίδας του Εργαστηρίου χρειάστηκε εμβάθυνση στις γλώσσες προγραμματισμού JavaScript, HTML και CSS όπως και στο framework Next.js. Σε αντίθεση με το WordPress CMS, η ανάπτυξη της ιστοσελίδας με το Next.js χρειάζεται συγγραφή κώδικα, ενώ δεν υπάρχει η λογική του “drag-n-drop”.

Ως επέκταση της παρούσας εργασίας προτείνεται η περαιτέρω μελέτη της ιστοσελίδας του Εργαστηρίου και η μετατροπή της σε ένα e-learning platform, το οποίο θα συνοδεύεται από μία κινητή εφαρμογή κατασκευασμένη σε ένα JavaScript framework όπως το Next.js ή το Ionic framework.

Βιβλιογραφία

[1] F. Copes, *The React Handbook*.

[2] "ReactJS Tutorial for Beginners: Learn ReactJS with Example", *Guru99.com*, 2021. [Online]. Available: https://www.guru99.com/reactjs-tutorial.html .

[3] K. Jacksi and S. Abass, "Development History Of The World Wide Web", *INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH*, vol. 8, no. 9, 2019.

[4] "ResearchGate | Find and share research", *ResearchGate*, 2021. [Online]. Available: https://www.researchgate.net/ .

[5] "The History of Websites 101", *Linkedin.com*, 2021. [Online]. Available: https://www.linkedin.com/pulse/history-websites-101-sami-alsayyed/ .

[6] "What should you choose? WordPress vs. React Frameworks", *Withloveinternet.com*, 2021. [Online]. Available: https://withloveinternet.com/blog/what-should-you-choose-wordpress-vs.-react-frameworks .

[7] "Wordpress Vs Reactjs: What Should You Choose?", *Mindbowser*, 2021. [Online]. Available: https://www.mindbowser.com/react-vs-wordpress/ .

[8] "Angular vs. React: Which is Better and Why?", *Cleveroad Inc. - Web and App development company*, 2021. [Online]. Available: https://www.cleveroad.com/blog/angular-vs-react .

[9] "Angular vs React: which framework to pick?", *Blog | Imaginary Cloud*, 2021. [Online]. Available: https://www.imaginarycloud.com/blog/angular-vs-react/#Differences .

[10] H. Atha and H. Atha, "8 Angular Apps Examples to Inspire Your Next App Project - Moveo Apps", *Moveo Apps*, 2021. [Online]. Available: https://www.moveoapps.com/blog/angular-apps-examples/ .

[11] "Vue vs React in 2021: Which Framework to Choose and When", *Monterail.com*, 2021. [Online]. Available: https://www.monterail.com/blog/vue-vs-react-2021 .

[12] 2021. [Online]. Available: https://www.mindk.com/blog/react-vs-vue/ .

[13] A. Boduch and R. Derks, *React and React Native - Third Edition*. 2020.

[14] A. Accomazzo, N. Murray, A. Lerner, C. Allsopp, D. Guttman and T. McGinnis, *Fullstack React*. 2020.

[15] "How To Build A Simple Calculator Application With React.JS", *Medium*, 2021. [Online]. Available: https://medium.com/@nitinpatel_20236/how-to-build-a-simple-calculator-application-with-react-js-bc10a4568bbd .

[16] R. Krajewski, "Next.js vs React: Which JavaScript Framework Is Better For your Front-end?", *Ideamotive.co*, 2021. [Online]. Available: https://www.ideamotive.co/blog/nextjs-vs-react-which-javascript-framework-is-better-for-your-front-end .

[17] *Pagepro.co*, 2021. [Online]. Available: https://pagepro.co/blog/what-is-nextjs/ .

[18] M. Liberati, "Kick Off a React JS Project: CRA, Next.js or Gatsby? -", *Codemotion Magazine*, 2021. [Online]. Available: https://www.codemotion.com/magazine/dev-hub/web-developer/react-project-cra-nextjs-gatsby/ .
[19] "Next.js by Vercel - The React Framework", *Nextjs.org*, 2021. [Online]. Available: https://nextjs.org/ .
[20] "Next.js by Vercel - The React Framework", *Nextjs.org*, 2021. [Online]. Available: https://nextjs.org/ .
[21] "WordPress Themes & Website Templates from ThemeForest", *ThemeForest*, 2021. [Online]. Available: https://themeforest.net/ .
[22] "Node.js", *Node.js*, 2021. [Online]. Available: https://nodejs.org/ .
[23] "PostgreSQL", *PostgreSQL*, 2021. [Online]. Available: https://www.postgresql.org/ .
[24] "Image and Video Upload, Storage, Optimization and CDN", *Cloudinary*, 2021. [Online]. Available: https://cloudinary.com/
[25] "Online payment processing for internet businesses – Stripe", *Stripe.com*, 2021. [Online]. Available: https://stripe.com/ .
[26] V. Code, "Visual Studio Code - Code Editing. Redefined", *Code.visualstudio.com*, 2021. [Online]. Available: https://code.visualstudio.com/ .