ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Web-Based Σύστημα Θεατρικών Παραστάσεων - Front End»

Του φοιτητή
Δημητρίου Αναστασιάδη
Αρ. Μητρώου: 154429

Επιβλέπων
Μιχαήλ Σαλαμπάσης
Καθηγητής

Θεσσαλονίκη 2023

Τίτλος Δ.Ε.: Web-Based Σύστημα Θεατρικών Παραστάσεων - Front End
Κωδικός Δ.Ε.: 21104
Ονοματεπώνυμο φοιτητή: Δημήτριος Αναστασιάδης
Ονοματεπώνυμο εισηγητή: Μιχαήλ Σαλαμπάσης
Ημερομηνία ανάληψης Δ.Ε.: 20-02-2021
Ημερομηνία περάτωσης Δ.Ε.: 19-01-2023

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

Πρόλογος

Το διαδίκτυο και οι ιστοσελίδες αποτελούν πλέον αναπόσπαστο κομμάτι της καθημερινότητάς μας. Συναντάται σε όλες τις πτυχές της ζωής μας: στην οικονομία, την ενημέρωση, την εκπαίδευση, την επικοινωνία αλλά και τον πολιτισμό και την ψυχαγωγία.

Αυτή η αυξανόμενη ανάγκη για περισσότερα web συστήματα τα οποία καλύπτουν τις απαιτήσεις μας, αλλά ταυτοχρόνως είναι εύχρηστα και με υψηλές επιδόσεις, έχει οδηγήσει στην ραγδαία εξέλιξη των γλωσσών και των τεχνολογιών του διαδικτύου. Οι τεχνολογίες αυτές έχουν ως σκοπό την ταχεία ανάπτυξη εφαρμογών με το ελάχιστο δυνατό κόστος, καθώς και να κάνουν την διαδικασία ανάπτυξης λογισμικού πιο φιλική προς τους προγραμματιστές.

Ο λόγος που επέλεξα αυτήν την διπλωματική είναι η αγάπη μου για αυτόν τον κόσμο της τεχνολογίας που συνεχώς αλλάζει. Λίγα πράγματα συγκρίνονται με την αίσθηση ικανοποίησης που προσφέρει η δημιουργία μιας εφαρμογής «από το μηδέν», καθώς και την ελευθερία να εκφράσεις δημιουργικά τον εαυτό σου μέσω αυτού που έχτισες.

Περίληψη

Το θέατρο είναι μία από τις αρχαιότερες μορφές τέχνης και έχει τις ρίζες του στην Αρχαία Ελλάδα. Είναι ένα μέσω επικοινωνίας και κοινωνικοποίησης, αφού μέσω του θεάτρου ο άνθρωπος έρχεται πιο κοντά ο ένας με τον άλλο. Μέσω του θεάτρου ο άνθρωπος μορφώνεται πνευματικά και ηθικά, και καλλιεργεί τον εσωτερικό του κόσμο και την κοινωνική του συνείδηση. Αποτελεί μία μορφή ψυχαγωγίας η οποία του επιτρέπει να δραπετεύσει από την ρουτίνα και τις δυσκολίες της καθημερινότητας.

Σε μία εποχή που κυριαρχούν η τηλεόραση και οι υπερπαραγωγές του κινηματογράφου, δεν πρέπει να ξεχνάμε την τέχνη από την οποία προέκυψαν, το θέατρο. Για αυτόν τον λόγο είναι σημαντικό να χρησιμοποιούμε τα εργαλεία που μας παρέχει η τεχνολογία για να αναδεικνύουμε τα πολιτισμικά μας αγαθά.

Το αντικείμενο αυτής της διπλωματικής εργασίας είναι η ανάπτυξη μίας διαδικτυακής εφαρμογής, η οποία θα προσφέρει στους χρήστες την δυνατότητα να ενημερώνονται για θεατρικές παραστάσεις, θεατρικούς χώρους, ηθοποιούς, σεναριογράφους, σκηνοθέτες και άλλους ανθρώπους του θεάματος. Επιπλέον οι χρήστες θα μπορούν να ανακαλύπτουν παραστάσεις με βάση την απόστασή τους από τον θεατρικό χώρο, καθώς και η προβολή στατιστικών για χρονικές περιόδους αλλά και η σύγκριση χρονικών περιόδων μεταξύ τους.

Για την υλοποίηση της παραπάνω εφαρμογής χρησιμοποιήθηκαν οι πλέον σύγχρονες τεχνολογίες και τεχνικές του διαδικτύου, η μελέτη και η ανάλυση των οποίων αποτελεί ένα σημαντικό μέρος της παρούσας διπλωματικής και θα παρουσιαστούν εις βάθος.

«Web-Based System for Theatrical Plays - Front End»

«Dimitrios Anastasiadis»

Abstract

Theater is one of the most ancient forms of art and it has its origins in Ancient Greece. It is a way for people to communicate and socialize as through it we come closer to each other. Through theater we grow as people spiritually and ethically, and we nourish our inner world, as well as our social consciousness. It is a form of entertainment that allows us to escape from the routine and hardship of everyday life.

In our day and age, where television and big cinema productions are taking over the world, we must not forget the art from which they originated from. For that reason, it is important to use the tool that technology has gifted us to highlight our cultural treasures.

The subject of this thesis is the development of a web application, through which the users will have the opportunity to be informed about theatrical plays, theatrical venues, actors, screenwriters, directors, and all sorts of people of the industry. Users will also be able to discover plays based on their location and view stats and compare time periods.

To achieve the creation of the application, some of the most modern technologies and techniques of the web were used, the study and breakdown of which make up a significant part of this thesis and will be presented thoroughly.

Περιεχόμενα

Κατάλογος Σχημάτων

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
JSON	JavaScript Object Notation
API	Application Programming Interface
ID	Identifier
URL	Uniform Resource Locator
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
JS	JavaScript
DOM	Document Object Model
XML	Extensible Markup Language
JSX	JavaScript XML
UI	User Interface
SEO	Search Engine Optimization
CDN	Content Delivery Network
ISR	Incremental Static Regeneration
SSL	Secure Sockets Layer
CI/CD	Continuous Integration / Continuous Delivery
UX	User Experience
WCAG	Web Content Accessibility Guidelines
SPA	Single Page Application

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα παρουσιαστεί μία γενική περιγραφή της εφαρμογής, καθώς και θα περιγραφούν τα δεδομένα που αυτή χρησιμοποιεί και ο τρόπος και η πηγή άντλησής τους. Επιπροσθέτως, θα γίνει μία αναφορά στην δομή της εργασίας και θα παρουσιαστούν τα επιμέρους κεφάλαια που την αποτελούν και μία συνοπτική περιγραφή τους.

1.2 Τι είναι μια web εφαρμογή

Μια web εφαρμογή είναι λογισμικό, το οποίο είναι αποθηκευμένο σε έναν απομακρυσμένο διακομιστή (server) και η πρόσβαση σε αυτό γίνεται μέσω ενός φυλλομετρητή. Ο διακομιστής ανταλλάσσει δεδομένα με τους πελάτες (clients), ύστερα από αίτησή τους μέσω του διαδικτύου. Οι εφαρμογές αυτές είναι σχεδιασμένες με γνώμονα την αρχιτεκτονική client-server, όπου παρέχονται στον πελάτη υπηρεσίες μέσω διακομιστών.

Η διαφορά με τις παραδοσιακές desktop εφαρμογές είναι ότι οι web εφαρμογές δεν χρειάζεται να εγκατασταθούν στη συσκευή του πελάτη τοπικά. Η πρόσβαση σε αυτές γίνεται μέσω ενός φυλλομετρητή εισάγοντας την διεύθυνση της εφαρμογής στην γραμμή διευθύνσεων. Ο διακομιστής στέλνει στον φυλλομετρητή ένα σύνολο από HTML, CSS και JavaScript αρχεία, τα οποία καθορίζουν την δομή, την εμφάνιση και την λειτουργικότητα της εφαρμογής. Ο φυλλομετρητής αφού λάβει αυτά τα αρχεία τα διερμηνεύει και τα παρουσιάζει σε μια κατανοητή, για τον άνθρωπο, μορφή. [1]

Επειδή οι web εφαρμογές «ζουν» στον διαδίκτυο, είναι προσβάσιμες από μεγάλο αριθμό χρηστών, ανεξαρτήτως από το που βρίσκονται στον κόσμο και τον τύπο της συσκευής τους. Το μόνο που απαιτείται είναι μία σύνδεση στο διαδίκτυο και ένας φυλλομετρητής. Οι χρήστες δεν χρειάζεται να ανησυχούν για τυχόν απώλεια δεδομένων αφού αυτά αποθηκεύονται σε μία απομακρυσμένη τοποθεσία. Έτσι προστίθεται ένα επίπεδο ασφάλειας στις web εφαρμογές από πιθανή καταστροφή της συσκευής του χρήστη ή κακόβουλο λογισμικό.

Την σήμερον ημέρα οι web εφαρμογές βρίσκονται παντού γύρω μας και θα ήταν δύσκολο να φανταστούμε τη ζωή μας χωρίς αυτές. Μερικά παραδείγματα web εφαρμογών είναι: τα webmail, τα ηλεκτρονικά καταστήματα, το e-banking, υπηρεσίες διαμοιρασμού και κοινοποίησης βίντεο και φωτογραφιών, τα μέσα κοινωνικής δικτύωσης.

1.3 Ο Στόχος της Διπλωματικής

Ο στόχος της διπλωματικής είναι ο σχεδιασμός και η υλοποίηση μίας web εφαρμογής, και πιο συγκεκριμένα το front-end κομμάτι αυτής, που θα προσφέρει στους χρήστες πρόσβαση σε business analytics σχετικά με τον χώρο του θεάτρου. Οι χρήστες θα μπορούν να προβάλλουν και να εξερευνούν στατιστικά σχετικά με το θέατρο σε μια συγκεκριμένη χρονική περίοδο της επιλογής τους, όπως για παράδειγμα πόσες παραστάσεις παίχτηκαν ή ποια ήταν η ακριβότερη παράσταση. Επιπλέον, θα μπορούν να πραγματοποιούν σύγκριση μεταξύ χρονικών περιόδων, η οποία θα παρέχεται σε μορφή γραφημάτων, για την ευκολότερη κατανόηση των δεδομένων και την ταχεία εξαγωγή συμπερασμάτων.

Οι χρήστες θα μπορούν, επίσης, να αναζητούν και να ανακαλύπτουν θεατρικές παραστάσεις, θεατρικούς χώρους, ηθοποιούς, σκηνοθέτες, σεναριογράφους, καθώς και να προβάλλουν λεπτομέρειες

για αυτούς, όπως για παράδειγμα πόσες φορές και που παίχτηκε μία παράσταση, ποιοι ήταν οι συντελεστές της, σε ποιες παραστάσεις έχει παίξει ένας συγκεκριμένος ηθοποιός, που βρίσκεται ένας θεατρικός χώρος.

Σκοπός της εφαρμογής δεν είναι απλά η δημιουργία μιας έγκυρης πηγής πληροφόρησης για ανθρώπους που αγαπούν το θέατρο, αλλά και η ανάπτυξη ενός συστήματος που θα είναι εύκολο στην χρήση, την κατανόηση των απεικονιζόμενων δεδομένων και γενικά θα προσφέρει στον χρήστη μία ευχάριστη εμπειρία.

1.4 Front-end vs Back-end

Το front end είναι το κομμάτι μίας διαδικτυακής εφαρμογής το οποίο οι χρήστες βλέπουν στις οθόνες τους μέσω του φυλλομετρητή και αλληλοεπιδρούν με αυτό. Οτιδήποτε συμβαίνει στην οθόνη του χρήστη, όπως για παράδειγμα το πάτημα ενός κουμπιού, ελέγχεται από τον κώδικα του front end, ο οποίος εκτελείται στον φυλλομετρητή στην συσκευή του χρήστη.

Τα δεδομένα όμως που παρουσιάζονται, όπως έχει προαναφερθεί, δεν είναι αποθηκευμένα τοπικά. Είναι αποθηκευμένα σε κάποια απομακρυσμένη βάση δεδομένων και λαμβάνονται μέσω του διαδικτύου. Το κομμάτι της εφαρμογής που είναι υπεύθυνο για την αποστολή, την επεξεργασία, την ενημέρωση και την αποθήκευση των δεδομένων αυτών είναι το back end. Ο κώδικας του back end εκτελείται σε κάποιον απομακρυσμένο διακομιστή και στη συνέχεια τα δεδομένα στέλνονται, συνήθως σε μορφή JSON (JavaScript Object Notation), στον φυλλομετρητή του χρήστη. Η επικοινωνία αυτή επιτυγχάνεται μέσω ενός API (Application Programming Interface). [2]

Το API είναι η διεπαφή μεταξύ δύο υπολογιστών ή και μεταξύ δύο προγραμμάτων. Μέσω αυτού ένα λογισμικό παρέχει υπηρεσίες σε ένα άλλο. Τα APIs έχουν ένα σύνολο από λειτουργείες που μπορούν να εκτελεστούν στο back end και τις οποίες μπορεί ένας πελάτης να επικαλεστεί μέσω των endpoints. Για παράδειγμα, ένα endpoint θα μπορούσε να είναι η λήψη όλων των πληροφοριών για έναν συγκεκριμένο ηθοποιό.

Ένα σημαντικό πλεονέκτημα που προκύπτει από αυτόν τον διαχωρισμό του front end και του back end είναι το γεγονός ότι το ίδιο back end μπορεί μέσω του API να εξυπηρετεί πολλές διαφορετικές εφαρμογές, που έχουν ωστόσο ανάγκη από τα ίδια δεδομένα και υπηρεσίες. Μια web εφαρμογή, μια android εφαρμογή και μία iOS εφαρμογή έχουν διαφορετικό κώδικα στο front end, μπορούν όμως να εξυπηρετηθούν από το ίδιο, ένα και μοναδικό, back end.

Επιπλέον, επιτυγχάνεται καλύτερη αξιοποίηση των πόρων του διακομιστή, αφού η μόνη του δουλειά είναι η λήψη των δεδομένων από την βάση και η αποστολή τους στον πελάτη. Δεν χρειάζεται δηλαδή να χτίζει μια HTML σελίδα κάθε φορά που λαμβάνει ένα αίτημα από έναν πελάτη. Συγχρόνως, από την μεριά του πελάτη, ο φυλλομετρητής δεν χρειάζεται να κατεβάσει ολόκληρη την σελίδα κάθε φορά που κάποιο δεδομένο αλλάζει. Αντιθέτως, αλλάζουν δυναμικά μόνο τα κομμάτια τα οποία είναι απαραίτητα. [3]

1.5 Δεδομένα

Τα δεδομένα της εφαρμογής αντλούνται από το HTML Scrapping της ιστοσελίδας viva.gr. HTML Scrapping είναι η διαδικασία κατά την οποία εξειδικευμένο λογισμικό κάνει εξόρυξη πληροφορίας από μια ιστοσελίδα, εκμεταλλευόμενου του γεγονότος ότι η δομή της κάθε ιστοσελίδας περιγράφεται στον HTML κώδικά της. Στη συνέχεια τα δεδομένα αυτά μπορούν να αποθηκευτούν σε μία βάση δεδομένων και να χρησιμοποιηθούν από τρίτες εφαρμογές. Η υλοποίηση του scrapping έγινε στα πλαίσια δύο

άλλων εργασιών του τμήματος, παρ' όλα αυτά είναι σημαντικό να αναφερθεί η πηγή και η δομή των δεδομένων αυτών.

1.5.1 Το Μοντέλο Δεδομένων

Τα δεδομένα της εφαρμογής είναι αποθηκευμένα σε μία σχεσιακή βάση δεδομένων. Σχεσιακή ονομάζεται μία βάση δεδομένων η οποία έχει οργανωθεί σε πίνακες, όπου κάθε ένας από αυτούς αποτελεί μία ξεχωριστή οντότητα. Ο κάθε πίνακας είναι ένα σύνολο από αντικείμενα του ίδιου τύπου. Κάθε τέτοιο αντικείμενο είναι μία ξεχωριστή γραμμή στον πίνακα, στην οποία προσδίδεται ένα μοναδικό κύριο κλειδί, για την ταυτοποίηση του αντικειμένου. Οι στήλες του πίνακα περιγράφουν τα χαρακτηριστικά ή τις ιδιότητες της οντότητας. Το κύριο χαρακτηριστικό μίας σχεσιακής βάσης δεδομένων είναι οι συνδέσεις ή σχέσεις μεταξύ των πινάκων, με την χρήση ξένων κλειδιών. Το κύριο κλειδί ενός πίνακα μπορεί να χρησιμοποιηθεί ως ξένο κλειδί σε έναν άλλον, και με αυτόν τον τρόπο να επιτευχθεί η συσχέτιση γραμμών του ενός πίνακα με τον άλλον.

Παρακάτω θα γίνει μία σύντομη περιγραφή των πινάκων, καθώς και των συσχετίσεων μεταξύ αυτών. Η βάση απαρτίζεται από τους εξής πίνακες:

- **Πίνακας production**. Ο πίνακας production είναι ο βασικός πίνακας της βάσης. Κάθε εγγραφή αυτού αντιπροσωπεύει μία θεατρική παραγωγή και περιγράφεται από τις εξής πεδία: 1) ID – το κύριο κλειδί του πίνακα, 2) Τίτλος Παραγωγής 3) Περιγραφή της εφαρμογής 4) URL – ο σύνδεσμος για την σελίδα της παραγωγής στο viva.gr 5) MediaURL – ο σύνδεσμος προς την φωτογραφία της παραγωγής 6) Χρονική Διάρκεια.

- **Πίνακας persons**. Κάθε εγγραφή του πίνακα persons αντιπροσωπεύει έναν ξεχωριστό άνθρωπο που συνεισφέρει στην υλοποίηση μία θεατρικής παραγωγής. Ο πίνακας περιέχει το πεδίο ID, που είναι και το κύριο κλειδί του πίνακα, και το πεδίο Fullname, δηλαδή το ονοματεπώνυμο του κάθε ανθρώπου.

- **Πίνακας images**. Κάθε εγγραφή του πίνακα images αντιπροσωπεύει μία φωτογραφία ενός ανθρώπου. Οι πίνακες persons και images συσχετίζονται μεταξύ τους με σχέση 'ένα προς πολλά', δηλαδή κάθε άνθρωπος μπορεί να έχει πολλές φωτογραφίες. Ο πίνακας αποτελείται από τα εξής πεδία: 1) ID – το κύριο κλειδί του πίνακα 2) imageURL – ο σύνδεσμος προς την φωτογραφία 3) personID – το ξένο κλειδί που συσχετίζει τους πίνακες persons και images.

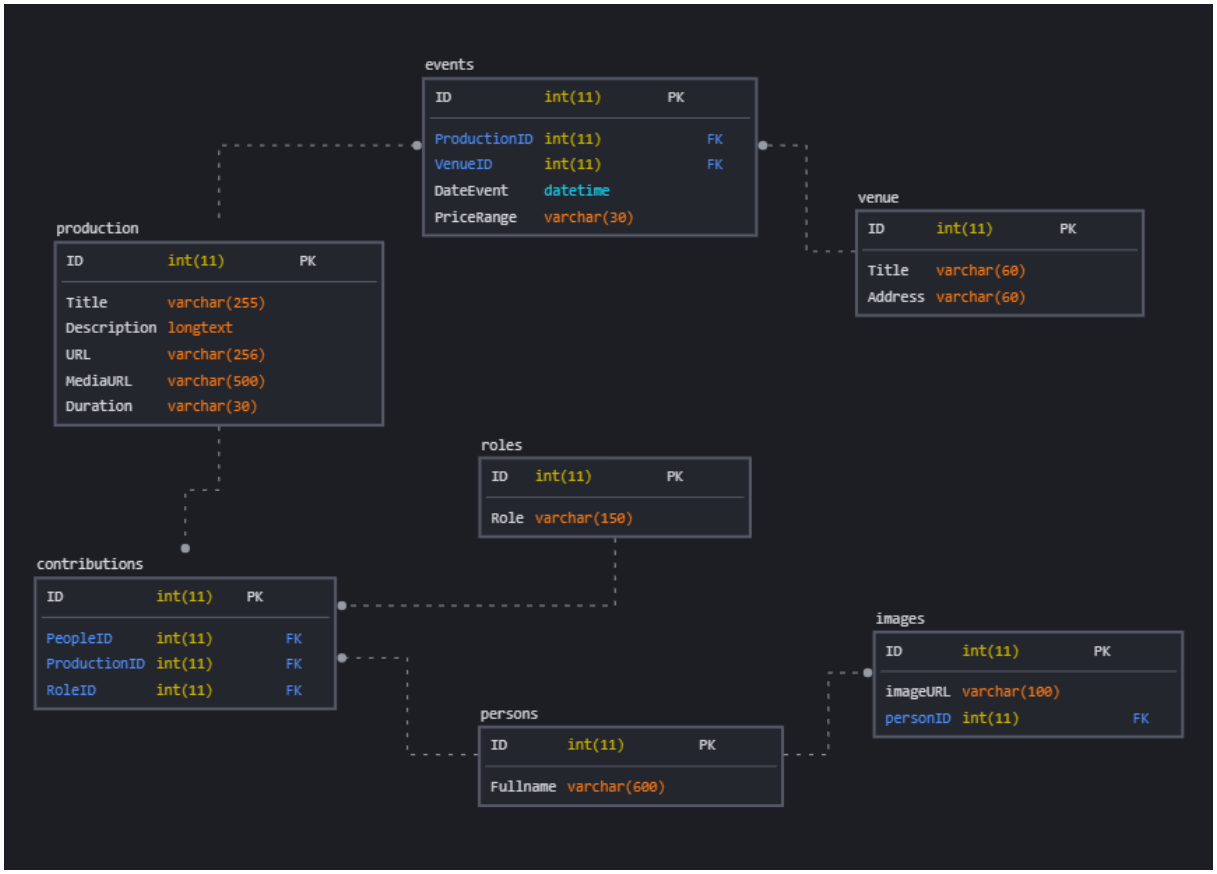
- **Πίνακας roles**. Κάθε γραμμή αυτού του πίνακα περιγράφει τους διάφορους ρόλους που μπορεί να έχει ένας άνθρωπος με την συνεισφορά του σε μία θεατρική παραγωγή. Για παράδειγμα, ένας άνθρωπος μπορεί να είναι ηθοποιός σε μία θεατρική παραγωγή. Ο πίνακας αποτελείται από τα πεδίο ID, που είναι το κύριο κλειδί, και το πεδίο role, δηλαδή τον τίτλο του ρόλου.

- **Πίνακας contributions**. Ο πίνακας contributions είναι ένας πίνακας συσχέτισης. Πιο συγκεκριμένα τα πεδία του PeopleID, ProductionID, RoleID είναι ξένα κλειδιά στους πίνακες persons, production και roles αντίστοιχα. Με αυτόν τον τρόπο επιτυγχάνεται η δημιουργία σχέσης 'πολλά προς πολλά' μεταξύ αυτών τον πινάκων. Ένας άνθρωπος μπορεί να συνεισφέρει σε πολλές διαφορετικές θεατρικές παραγωγές και πιθανών με διαφορετικούς ρόλους. Αντίστοιχα σε μία παραγωγή συνεισφέρουν πολλοί άνθρωποι.

- **Πίνακας venue**. Κάθε γραμμή του πίνακα venue αντιπροσωπεύει έναν θεατρικό χώρο στον οποίο μπορεί να λάβει μέρος μία παράσταση. Αποτελείται από τα πεδία: 1) ID – το κύριο κλειδί του πίνακα 2) Title – η ονομασία του θεατρικού χώρου 3) Την διεύθυνση του θεατρικού χώρου.

- **Πίνακας events**. Κάθε εγγραφή αυτού του πίνακα αντιπροσωπεύει μία μεμονωμένη εκδήλωση μίας θεατρικής παραγωγής, δηλαδή μία παράσταση. Ο πίνακας events συσχετίζεται με τον

πίνακα production με σχέση ‘ένα προς πολλά’ αφού κάθε παραγωγή μπορεί να έχει πάνω από μια παράσταση. Επίσης συσχετίζεται με τον πίνακα venue με σχέση ‘ένα προς ένα’ αφού κάθε παράσταση παίζετε σε έναν θεατρικό χώρο. Ο πίνακας αποτελείται από τις εξής στήλες: 1) ID – το κύριο κλειδί του πίνακα 2) ProductionID – το ξένο κλειδί που συσχετίζει τον πίνακα events με τον πίνακα production 3) VenueID – το ξένο κλειδί που συσχετίζει τον πίνακα events με τον πίνακα venue 4) Ημερομηνία της παράστασης 5) Εύρος τιμής εισιτηρίων.



Εικόνα 1.1: Διάγραμμα Οντοτήτων – Συσχετίσεων

1.6 Δομή της Εργασίας

Η εργασία είναι χωρισμένη σε πέντε κεφάλαιο, κάθε ένα από τα οποία αποτελεί και μια ξεχωριστή θεματική ενότητα. Στο πρώτο, τρέχων, κεφάλαιο γίνεται μία παρουσίαση των γενικών εννοιών που σχετίζονται με την εφαρμογή, καθώς και των δεδομένων πάνω στα οποία βασίζεται η εφαρμογή.

Στο δεύτερο κεφάλαιο, θα γίνει μία αναλυτική περιγραφή των τεχνολογιών που χρησιμοποιήθηκαν για την δημιουργία της εφαρμογής, τα χαρακτηριστικά τους, οι λειτουργίες τους και οι δυνατότητες που προσφέρουν για την γρηγορότερη και πιο αποτελεσματική ανάπτυξη μίας σύγχρονης web εφαρμογής.

Στο τρίτο κεφάλαιο, θα περιγραφεί η αρχιτεκτονική την οποία ακολουθεί η εφαρμογή ενώ θα γίνει και μία αναφορά στα APIs, τα οποία χρησιμοποιεί για την λειτουργία της. Θα γίνει επίσης μία αναλυτική περιγραφή των εργαλείων που χρησιμοποιήθηκαν για τον σχεδιασμό των διεπαφών χρήστη, έτσι ώστε να δημιουργήσουν την καλύτερη δυνατή εμπειρία για τον χρήστη.

Στο τέταρτο κεφάλαιο, θα γίνει μία αναλυτική περιγραφή της εφαρμογής. Θα παρουσιαστούν οι επιμέρους σελίδες, καθώς και τα components που χρησιμοποιήθηκαν για την δημιουργία τους, ενώ παράλληλα θα παρουσιαστούν κάποια κομμάτια κώδικα, τα οποία εκτελούν κάποια βασική λειτουργία στην εφαρμογή.

Στο πέμπτο κεφάλαιο, θα παρουσιαστούν συνοπτικά κάποιες προτάσεις βελτίωσης της εφαρμογής, οι οποίες αποσκοπούν είτε στην βελτίωση της εμπειρίας του χρήστη, είτε στην πιο αποδοτική ανάπτυξη της εφαρμογής, με την ενσωμάτωση νέων τεχνολογιών.

1.7 Επίλογος

Στο κεφάλαιο αυτό, έγινε μία παρουσίαση των βασικών εννοιών που σχετίζονται με την εφαρμογή, όπως ο διαχωρισμός front-end και back-end, και ο ορισμός μίας web εφαρμογής. Στην συνέχεια, παρουσιάστηκαν τα δεδομένα που είναι αποθηκευμένα στην βάση δεδομένων και τέλος έγινα μία αναφορά στην δομή της εργασίας, τα κεφάλαια που την απαρτίζουν και τι εμπεριέχουν αυτά.

Κεφάλαιο 2ο: Τεχνολογίες

2.1 Εισαγωγή

Στο παρακάτω κεφάλαιο θα γίνει μία αναλυτική περιγραφή των γλωσσών και των τεχνολογιών που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής. Θα αναφερθούν οι δυνατότητες και τα πλεονεκτήματά τους, καθώς και οι λόγοι για τους οποίους επιλέχθηκαν και κρίθηκαν κατάλληλες για την αποτελεσματική ανάπτυξη της εφαρμογής.

2.2 HTML

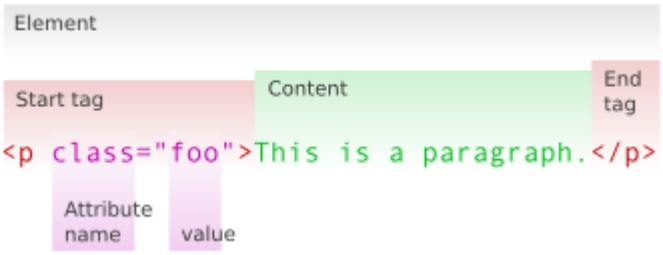
Η HTML (HyperText Markup Language) είναι η βασική γλώσσα του διαδικτύου και η βάση πάνω στην οποία χτίζονται όλες οι σύγχρονες ιστοσελίδες και web εφαρμογές. Είναι μία γλώσσα σήμανσης με την οποία περιγράφεται η δομή της σελίδας. Αυτό επιτυγχάνεται με μία σειρά από HTML elements μέσα στα οποία περικλείεται το περιεχόμενο της σελίδας. Ανάλογα με το είδος του element αλλάζει ο τρόπος με τον οποίο εμφανίζεται το περιεχόμενο που περικλείεται μέσα σε αυτό.

Με τον όρο HyperText (υπερκείμενα) εννοούμε ότι γίνεται η χρήση υπερσυνδέσμων για την σύνδεση σελίδων μεταξύ τους, μέσα στην ίδια ιστοσελίδα, αλλά και ανάμεσα σε διαφορετικές ιστοσελίδες.

2.2.1 HTML Στοιχεία

Κάθε σελίδα αποτελείται από μια σειρά HTML στοιχείων τα οποία περιγράφουν την δομή της. Το κάθε στοιχείο αποτελείται με την σειρά του από μια σήμανση ανοίγματος, το περιεχόμενο ανάμεσα στις σημάνσεις, και μία σήμανση κλεισίματος.

Οι σημάνσεις περικλείονται ανάμεσα σε αγκύλες (<element>) ενώ ο διαχωρισμός ανάμεσα στην σήμανση κλεισίματος και ανοίγματος γίνεται με την χρήση καθέτου πριν το όνομα του στοιχείου (</element>).



[4]

Εικόνα 2.1: Δομή HTML Στοιχείου

Όπως φαίνεται από το παραπάνω σχήμα εκτός από τις σημάνσεις αρχής και τέλους και το περιεχόμενο, τα στοιχεία μπορεί να έχουν και κάποιες παραμέτρους (attributes), οι οποίες δίνουν πρόσθετες ιδιότητες στο στοιχείο. Ένα attribute περικλείεται μέσα στην σήμανση αρχής και πρέπει ανάμεσα στο όνομα του της σήμανσης και στα attributes να υπάρχει κενό. Για παράδειγμα, η πηγή της φωτογραφίας σε ένα στοιχείο ‘img’ δίδεται μέσω του attribute ‘src’, το οποίο παίρνει ως τιμή το URL της εικόνας, εάν η

πρόσβαση σε αυτήν γίνεται μέσω του διαδικτύου, ή το μονοπάτι προς το αρχείο εάν η πρόσβαση γίνεται τοπικά. [5]

2.2.2 Βασικότερες HTML Σημάνσεις

Παρακάτω θα γίνει μία αναφορά στις βασικότερες και πιο συχνά χρησιμοποιούμενες HTML σημάνσεις καθώς και θα εξηγηθεί εν συντομία η χρήση τους.

Η σήμανση **<html>** είναι η ρίζα του HTML αρχείου μέσα στην οποία περιλαμβάνονται όλα τα άλλα elements. Μία καλή πρακτική είναι να ορίζεται πάντα η παράμετρος 'lang' του **<html>** στοιχείου, δηλαδή η γλώσσα στην οποία είναι γραμμένο το περιεχόμενο της ιστοσελίδας.

Συνεχίζοντας μέσα στην ιεραρχία ενός HTML αρχείου συνήθως συναντάμε το στοιχείο με την σήμανση **<head>**. Μέσα σε αυτό το στοιχείο εμπεριέχονται άλλα στοιχεία που προσδίδουν στην σελίδα πρόσθετες πληροφορίες που δεν είναι άμεσα ορατές στον χρήστη μέσω του περιηγητή. Για παράδειγμα, εδώ μπορεί να οριστεί ο τίτλος της σελίδας, αναφορές σε εξωτερικά αρχεία, και άλλα μεταδεδομένα.

Η επόμενη βασική σήμανση είναι η **<body>**. Μέσα στο στοιχείο με αυτήν την σήμανση εμπεριέχονται όλα τα άλλα στοιχεία τα οποία μπορεί να δει και να αλληλοεπιδράσει μαζί τους ο χρήστης μέσω του παραθύρου του περιηγητή. Αποτελεί, δηλαδή, το σώμα της σελίδας.

Οι σημάνσεις **<h1>** έως **<h6>** χρησιμοποιούνται για τον ορισμό επικεφαλίδων μέσα στην σελίδα. Η σήμανση **<h1>** ορίζει την πιο σημαντική επικεφαλίδα, ενώ η **<h6>** την λιγότερο σημαντική. Μέσα σε μία σελίδα θα πρέπει να υπάρχει μόνο μία σήμανση **<h1>**, που θα περιγράφει το κύριο θέμα της σελίδας, ενώ παράλληλα καλό θα ήταν να μην παραλείπονται επίπεδα επικεφαλίδων, δηλαδή να μην γίνεται χρήση της σήμανσης **<h3>** εάν δεν έχει προηγηθεί μία σήμανση **<h2>**.

Η σήμανση **<p>** χρησιμοποιείται για τον ορισμό μίας παραγράφου, ενώ οι σημάνσεις **<b>** και <**i**> για να μετατραπεί το κείμενο σε bold και italics αντίστοιχα.

Η δημιουργία λιστών πραγματοποιείται με την χρήση των σημάνσεων **<ul>** και **<ol>**. Με την **<ul>** (unordered list) μπορούμε να δημιουργήσουμε λίστες χωρίς αρίθμηση των στοιχείων, για παράδειγμα με την χρήση bullet point. Με την <**ol**> (ordered list) ορίζεται λίστα στην οποία υπάρχει αρίθμηση των στοιχείων. Τα στοιχεία της λίστας ορίζονται και στις δύο περιπτώσεις με την σήμανση **li**, τα οποία πρέπει να περιέχονται μέσα σε ένα στοιχείο με σήμανση **<ul>** ή **<ol>**.

Με την σήμανση <**a**> ορίζεται ένας υπερσύνδεσμος, που χρησιμοποιείται για την μετάβαση από μία σελίδα σε μία άλλη. Η πιο σημαντική παράμετρος ενός υπερσυνδέσμου είναι η 'href', με την οποία προσδιορίζεται ο προορισμός του υπερσυνδέσμου.

Η σήμανση **<img>** χρησιμοποιείται για την προβολή μίας εικόνας μέσα στην ιστοσελίδα και έχει δύο υποχρεωτικές παραμέτρους, την 'src' και την 'alt'. Με την ιδιότητα 'src' προσδιορίζεται το μονοπάτι προς την εικόνα, ενώ η 'alt' προσδιορίζει το κείμενο που θα προβληθεί αντί της εικόνας εάν για κάποιο λόγο δεν είναι δυνατή η φόρτωσή της ή η προβολή της.

Με την σήμανση ** ** ορίζεται ένα τμήμα της ιστοσελίδας. Χρησιμοποιείται συνήθως ως ένα container. Μέσα σε ένα ** ** μπορεί να εμπεριέχεται οποιοδήποτε άλλο HTML στοιχείο. Παράλληλα μπορεί με ευκολία να αλλαχθεί η εμφάνιση ενός ** ** με την χρήση CSS, στην οποία θα αναφερθώ σε παρακάτω στο κεφάλαιο.

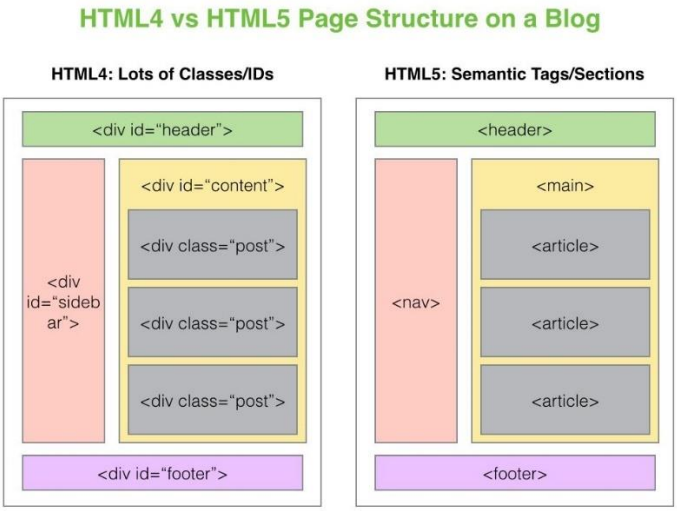
2.2.3 Σημασιολογική HTML

Σε μια εποχή που χαρακτηρίζεται από την ραγδαία ανάπτυξη της τεχνολογίας και την συνεχή ανάπτυξη νέου λογισμικού, έχει δημιουργηθεί η ανάγκη για την συγγραφή κώδικα, ο οποίος είναι πιο εύκολα κατανοητός και ευανάγνωστος, όχι μόνο από τους ανθρώπους αλλά και από τις ίδιες τις μηχανές.

Πολλές σημάνσεις, εκτός από την περιγραφή της δομής και της εμφάνισης των HTML εγγράφων, προσδίδουν νόημα στο περιεχόμενό τους. Για παράδειγμα, με τις σημάνσεις **<h1>** έως **<h6>** γίνεται εύκολα κατανοητό ότι το περιεχόμενο τους αποτελεί επικεφαλίδα, ενώ με την σήμανση **<button>** καταλαβαίνουμε ότι πρόκειται για κάποιο κουμπί με το οποίο ο χρήστης μπορεί να αλληλοεπιδράσει.

Με την έκδοση HTML 5, προστέθηκαν στις ήδη υπάρχουσες, πολλές νέες σημάνσεις με σημασιολογικό χαρακτήρα. Μερικές από αυτές είναι: η σήμανση **<main>** που εμπεριέχει το κύριο περιεχόμενο της σελίδας, η **<nav>** που εμπεριέχει τους συνδέσμους για την πλοήγηση μέσα στην ιστοσελίδα, οι σημάνσεις **<header>** και **<footer>,** η σήμανση **<section>** με την οποία χωρίζεται το περιεχόμενο της σελίδας σε εννοιολογικά συναφή τμήματα.

Με την χρήση της σημασιολογικής HTML, αποφεύγεται η υπερβολική χρήση των ** ** στοιχείων, η οποία κάνει σε πολλές περιπτώσεις τον κώδικα δυσανάγνωστο και δύσκολο στην κατανόηση.



Εικόνα 2.2: Απλή HTML vs Σημασιολογική HTML

Ένα ακόμη πλεονέκτημα της σημασιολογικής HTML είναι το γεγονός ότι παρέχει καλύτερη προσβασιμότητα (accessibility) στους χρήστες. Όλο και περισσότεροι άνθρωποι χρησιμοποιούν σε καθημερινή βάση τις δυνατότητες του διαδικτύου κι έτσι είναι αναπόφευκτο κάποιοι από αυτούς να έχουν κάποιο πρόβλημα στην όραση ή στην κινητικότητα. Η σημασιολογική HTML βοηθά τεχνολογίες, όπως τους αναγνώστες οθονών, να κατανοήσουν καλύτερα την δομή και το περιεχόμενο της σελίδας και να παρέχουν στον χρήστη την καλύτερη δυνατή εμπειρία. [6]

Παράλληλα, η χρήση σημασιολογικών σημάνσεων παίζει σημαντικό ρόλο στην κατάταξη μίας σελίδας από τις μηχανές αναζήτησης. Οι αλγόριθμοι που χρησιμοποιούν οι μηχανές αναζήτησης για την κατάταξη των σελίδων γίνονται όλο και πιο περίπλοκοι και ένας από τους παράγοντες που λαμβάνουν υπόψιν είναι ο βαθμός προσβασιμότητας της σελίδας.

2.3 CSS

Η CSS (Cascading Style Sheets ή διαδοχικά φύλλα στυλ) είναι η γλώσσα που χρησιμοποιείται για να αλλάξουμε την εμφάνιση ενός HTML εγγράφου ή γενικώς ενός εγγράφου που είναι γραμμένο σε κάποια γλώσσα σήμανσης. Η CSS και η HTML αποτελούν τις δύο βασικότερες γλώσσες του διαδικτύου. Η CSS σχεδιάστηκε με σκοπό τον διαχωρισμό του κώδικα που είναι υπεύθυνος για την εμφάνιση ενός HTML αρχείου από αυτόν που είναι υπεύθυνος για την δομή του.

Κάθε αρχείο CSS αποτελείται από ένα σύνολο από κανόνες, οι οποίοι έχουν ως σκοπό την επιλογή κάποιου HTML στοιχείου έτσι ώστε να επηρεάσουν την μορφή του. Ο κάθε κανόνας αποτελείται από έναν selector και ένα μπλοκ δηλώσεων, μέσα στο οποίο βρίσκονται ζεύγη ιδιοτήτων-τιμών που θέλουμε να αλλάξουμε.

```
h1 {
  font-size: 30px;
}
```

Εικόνα 2.3: Παράδειγμα Κανόνα CSS

2.3.1 CSS Selectors

Με τους CSS selectors επιλέγουμε ποια στοιχεία θέλουμε να επηρεάσει ο συγκεκριμένος κανόνας. Ένας selector μπορεί να επιλέξει μια ολόκληρη ομάδα στοιχείων με την ίδια σήμανση. Για παράδειγμα, ο selector **h1** επιλέγει όλα τα **<h1>** HTML στοιχεία.

Ένας άλλος τρόπος επιλογής στοιχείων είναι με selectors που βασίζονται σε κάποια HTML παράμετρο. Πιο συγκεκριμένα στις παραμέτρους **id** και **class**. Για να γίνει η επιλογή ενός στοιχείου με βάση την κλάση του ο selector θα πρέπει να έχει την μορφή '**.class-name**', δηλαδή τελεία ακολουθούμενη από το όνομα της κλάσης. Αντίστοιχα για την παράμετρο **id** η μορφή είναι '**#id-value**'.

Η επιλογή ενός στοιχείου μπορεί επίσης να γίνει ανάλογα με το ποια είναι η θέση του μέσα στο αρχείο. Παραδείγματος χάρη, ο selector '**h1:nth-child(2)**' επιλέγει όλα τα στοιχεία **h1**, τα οποία είναι δεύτερα παιδιά του στοιχείου-γονέα τους.

Η λέξη κλειδί **nth-child(n)** αποτελεί μία ψευδό-κλάση και έτσι δεν χρειάζεται να δημιουργούμε δικιά μας και να την προσθέσουμε σαν παράμετρο σε κάθε στοιχείο που θέλουμε να επηρεαστεί. Μία ακόμη χρήσιμη ψευδό-κλάση είναι η **hover** η οποία ενεργοποιείται όταν ο χρήστης τοποθετεί τον κέρσορα του ποντικιού πάνω από ένα στοιχείο.

Εκτός από αυτές τις απλές περιπτώσεις ένας selector μπορεί να επιλέξει πολλές κατηγορίες στοιχείων ταυτόχρονα ή και να συνδυάσει κατηγορίες με διάφορα ειδικά σύμβολα. Για την επιλογή πολλαπλών στοιχείων αρκεί ο διαχωρισμός τους με κόμμα. Ο selector '**h1, h2, h3**' επιλέγει όλα τα στοιχεία με σήμανση είτε **h1** είτε **h2** είτε **h3**.

Ο selector '**section h1**' επιλέγει όλα τα στοιχεία **<h1>**, τα οποία είναι απόγονοι ενός στοιχείου **<section>**, ενώ ο selector '**section>h1**' επιλέγει όλα τα στοιχεία **<h1>**, που είναι άμεσα παιδιά ενός στοιχείου **<section>**.

2.3.2 CSS Pseudo-Elements (Ψευδό-Στοιχεία)

Τα ψευδό-στοιχεία είναι λέξεις κλειδιά που μπορούν να τοποθετηθούν στο τέλος ενός selector, έτσι ώστε να αλλάξει η εμφάνιση ενός μόνο μέρους του επιλεγμένου στοιχείου. Για παράδειγμα, το ψευδό-στοιχείο **::first-line** μπορεί να χρησιμοποιηθεί για να αλλάξει η εμφάνιση της πρώτης μόνο γραμμής του επιλεγμένου στοιχείου. [7]

Ιδιαίτερο ενδιαφέρον παρουσιάζουν τα ψευδό-στοιχεία **::before** και **::after**, με τα οποία μπορούμε να προσθέσουμε επιπλέον περιεχόμενο στο επιλεγμένο στοιχείο. Με το **::before** προσθέτουμε περιεχόμενο ως το πρώτο παιδί του επιλεγμένου στοιχείου, ενώ με το **::after** το περιεχόμενο που θα προστεθεί θα είναι το τελευταίο παιδί του στοιχείου.

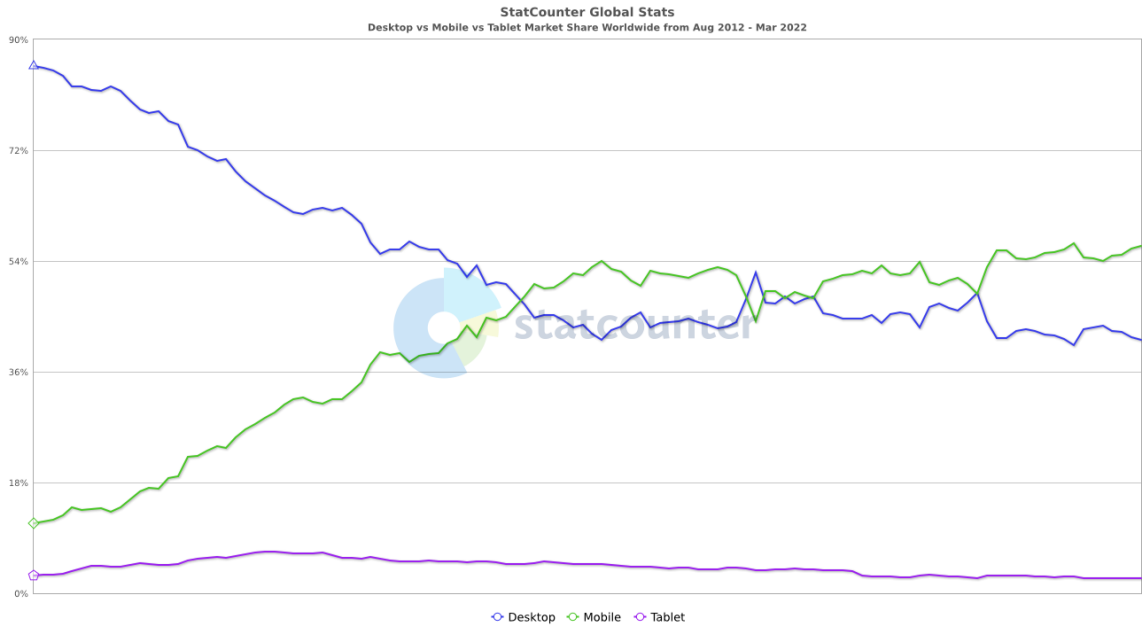
Ένα απλό παράδειγμα είναι η τοποθέτηση εισαγωγικών σε μία παράγραφο, χρησιμοποιώντας την ιδιότητα **content** για να ορίσουμε το περιεχόμενο των ψευδό-στοιχείων.

```
p::before {
  content: "«";
}
```

Εικόνα 2.4: Παράδειγμα Χρήσης Ψευδό-Στοιχείων

2.3.3 Media Queries

Την τελευταία δεκαετία η χρήση των κινητών συσκευών για την πρόσβαση στο διαδίκτυο έχει αυξηθεί κατακόρυφα. Όλο και περισσότεροι χρήστες του διαδικτύου χρησιμοποιούν το κινητό τους για τις καθημερινές τους ανάγκες, ενώ η χρήση των παραδοσιακών σταθερών υπολογιστών μειώνεται συνεχώς. Μάλιστα, τα τελευταία χρόνια η χρήση κινητών συσκευών ξεπέρασε αυτήν των desktop.



Εικόνα 2.5: Μερίδιο Αγοράς Desktop vs Mobile

Όπως φαίνεται από το παραπάνω γράφημα, το 2012 το μερίδιο αγοράς σταθερών υπολογιστών ήταν σχεδόν 90% ενώ οι κινητές συσκευές είχαν ποσοστό μικρότερο του 18%. Αντιθέτως, σήμερα οι κινητές συσκευές καταλαμβάνουν ποσοστό 56%, ξεπερνώντας τους σταθερούς υπολογιστές με ποσοστό 41%. [8]

Κάθε χρόνο βγαίνουν στην αγορά εκατοντάδες νέες συσκευές, καθεμία με διαφορετικά τεχνικά χαρακτηριστικά και διαστάσεις οθόνης. Υπάρχουν πάρα πολλοί συνδυασμοί ανάλυσης οθόνης, χωρίς να υπάρχει κάποιο συγκεκριμένο πρότυπο, γεγονός που καθιστά απίθανο τον σχεδιασμό και την ανάπτυξη μίας web εφαρμογής για ένα μόνο τύπο και μέγεθος οθόνης.

Εάν ο σχεδιασμός γίνει με βάση μια μεγάλη οθόνη, τότε στις κινητές συσκευές όλα θα φαίνονται πολύ μικρά και στριμωχτά. Αντίθετα, αν σχεδιάσουμε την εφαρμογή για μικρότερες οθόνες, σε μία οθόνη σταθερού υπολογιστή όλα τα στοιχεία θα φαίνονται πολύ μεγάλα.

Για την αντιμετώπιση αυτού του προβλήματος, χρησιμοποιούντα τα media queries. Με την χρήση τους μπορούμε να επιλέξουμε συγκεκριμένους τύπους συσκευών με συγκεκριμένα χαρακτηριστικά και να εφαρμόσουμε CSS κανόνες μόνο στις συσκευές που ικανοποιούν αυτές τις συνθήκες. Με τον τρόπο αυτό η εφαρμογή μας γίνεται responsive. Ανταποκρίνεται δηλαδή στα χαρακτηριστικά της συσκευής και η εμφάνιση της αλλάζει με βάση αυτά.

Ένα media query μπορεί να επιλέξει έναν τύπο συσκευής, screen για οθόνες και print για εκτυπωτές, χαρακτηριστικά συσκευής μέσα σε παρένθεση, αλλά και να συνδυάσει παραπάνω από ένα από αυτά με την χρήση των λογικών τελεστών **not**, **and** και **only**.

Μία από τις πιο συνηθισμένες λειτουργίες των media queries είναι η επιλογή συσκευών, των οποίων το παράθυρο του περιηγητή ξεπερνά κάποιο συγκεκριμένο πλάτος και η εφαρμογή κάποιων κανόνων CSS σε αυτές. Για παράδειγμα, το media query '**@media only screen and (min-width: 360px) { ... }**' επιλέγει μόνο οθόνες των οποίων το πλάτος του παραθύρου ξεπερνάει τα 360 pixels.

Με αυτό τον τρόπο είναι δυνατή η διαφοροποίηση της web εφαρμογής, έτσι ώστε να αυτή να έχει διαφορετική διάταξη και εμφάνιση σε ένα κινητό, σε ένα tablet ή σε ένα laptop ή σταθερό υπολογιστή. Αυτό έχει ως αποτέλεσμα την καλύτερη αξιοποίηση των δυνατοτήτων κάθε συσκευής και, συγχρόνως, αυξάνει την ευχρηστία και την προσβασιμότητα των χρηστών.

Μια εφαρμογή μπορεί σε ένα κινητό να έχει διάταξη με μία στήλη και μικρά διαστήματα ανάμεσα στα στοιχεία, για να αξιοποιήσει καλύτερα τον περιορισμένο χώρο της οθόνης, ενώ σε έναν σταθερό υπολογιστή μπορεί να έχει διάταξη τριών στηλών και τα διαστήματα να είναι μεγαλύτερα, για να μην κουράζει τον χρήστη και το κείμενο να είναι πιο ευανάγνωστο.

2.4 JavaScript

Η JavaScript είναι μία γλώσσα προγραμματισμού, που δημιουργήθηκε το 1995, και είναι μία από τις βασικές τεχνολογίες του διαδικτύου μαζί με την HTML και την CSS. Πάνω από το 97% όλων των ιστοσελίδων χρησιμοποιούν JavaScript, ενώ όλοι οι σύγχρονοι περιηγητές διαθέτουν εξειδικευμένο λογισμικό για την εκτέλεση JavaScript κώδικα.

Η δημιουργία της είχε σκοπό να κάνει τις ιστοσελίδες πιο δυναμικές και να αυξήσει την αλληλεπίδραση του χρήστη με αυτές. Μερικές από τις λειτουργίες της είναι:

- Η φόρτωση νέου περιεχομένου στην σελίδα χωρίς την ανανέωσή της, με την χρήση AJAX (Asynchronous JavaScript and XML) ή WebSocket. Για παράδειγμα, οι χρήστες μίας σελίδας

κοινωνικής δικτύωσης μπορούν να στέλνουν και να λαμβάνουν μηνύματα χωρίς να χρειάζεται να ανανεώσουν την ιστοσελίδα.
- Παίξιμο παιχνιδιών που εκτελούνται μέσα στον περιηγητή.
- Ο έλεγχος της αναπαραγωγής βίντεο και ήχων.
- Η επικύρωση της ορθότητας των δεδομένων που έχουν εισαχθεί σε μία φόρμα, πριν αυτά σταλούν στον διακομιστή.
- Η καταγραφή των κινήσεων και της συμπεριφοράς του χρήστη μέσα στην ιστοσελίδα και η αποστολή τους σε έναν διακομιστή. Ο ιδιοκτήτης της ιστοσελίδας μπορεί στη συνέχεια να αναλύσει αυτά τα δεδομένα και να τα χρησιμοποιήσει για στοχευμένη διαφήμιση και εξατομίκευση της ιστοσελίδας στις απαιτήσεις των χρηστών.
- Η ανακατεύθυνση του χρήστη από μία σελίδα σε μία άλλη. [9]

Η αρχική έκδοση της JavaScript σχεδιάστηκε από τον Brendan Eich για τον περιηγητή της Netscape μέσα σε δέκα ημέρες. Η αρχική της ονομασία ήταν Mocha, αλλά μετονομάστηκε σε LiveScript λίγους μήνες μετά την δημιουργία της και τελικά πήρε την σημερινή της ονομασία τον Δεκέμβριο του 1995. [10]

Η Microsoft ανέπτυξε την δικιά της εκδοχή της JavaScript με το όνομα JScript, για την χρησιμοποίησή της στον περιηγητή της εταιρίας Internet Explorer. Παρά το γεγονός ότι οι δύο γλώσσες έμοιαζαν πολύ μεταξύ τους, οι διαφορές που παρουσίαζαν έκαναν δύσκολη την ανάπτυξη ιστοσελίδων που να δούλευαν το ίδιο καλά και στους δύο περιηγητές.

Έτσι έγινε γρήγορα κατανοητό ότι έπρεπε να αναπτυχθεί ένα σύνολο προτύπων, στα οποία θα πρέπει να συμμορφώνονται όλες οι εκδοχές της γλώσσας. Η Netscape απευθύνθηκε στην ECMA η οποία είναι μία μη κερδοσκοπική οργάνωση, που αναπτύσσει και θέτει πρότυπα για συστήματα στον τομέα της πληροφορικής. Τον Ιούνιο του 1997 η ECMA δημοσίευσε την πρώτη επίσημη προδιαγραφή της γλώσσας, με το όνομα ECMAScript.

Σήμερα η JavaScript έχει επικρατήσει ως η κυρίαρχη γλώσσα που χρησιμοποιούν όλοι οι σύγχρονοι περιηγητές, ενώ ακόμα και ο νεότερος περιηγητής της Microsoft, ο Microsoft Edge, χρησιμοποιεί JavaScript.

2.4.1 Χαρακτηριστικά της JavaScript

Η JavaScript είναι μία υψηλού επιπέδου, διερμηνευμένη ή JIT (just-in-time) μεταγλωττισμένη γλώσσα προγραμματισμού, δηλαδή ο κώδικας μεταγλωττίζεται κατά την εκτέλεση του προγράμματος και όχι πιο πριν.

Έχει dynamic typing, που σημαίνει ότι ο τύπος των μεταβλητών ελέγχεται κατά την εκτέλεση του προγράμματος. Οι μεταβλητές ορίζονται με τις λέξεις **var**, **const** και **let** και δεν χρειάζεται να προσδιορίζεται ο τύπος τους (int, string, boolean κτλ.).

Οι συναρτήσεις στην JavaScript είναι first-class, δηλαδή είναι δυνατή η ανάθεση τους σε μία μεταβλητή και μπορούν επίσης να περαστούν ως παράμετροι σε μία άλλη συνάρτηση, καθώς και η επιστροφή τους ως το αποτέλεσμα μιας άλλης συνάρτησης.

Είναι βασισμένη σε πρωτότυπα αντικειμένων (prototype-based), δηλαδή τα πάντα στην JavaScript, που δεν είναι πρωτόγονου τύπου (string, number, bigint, boolean, undefined, symbol, null), είναι αντικείμενα, συμπεριλαμβανομένων των πινάκων και των συναρτήσεων. Όλα τα αντικείμενα είναι συνδεδεμένα με ένα αρχικό αντικείμενο, το πρωτότυπο, από το οποίο μπορούν να κληρονομήσουν τις λειτουργίες τους.

Είναι μια multi-paradigm γλώσσα αφού υποστηρίζει πολλούς διαφορετικούς τύπους προγραμματισμού, και αφήνει στην ευχέρεια του προγραμματιστή την επιλογή του κατάλληλου, ανάλογα με τις ανάγκες και τις απαιτήσεις της εκάστοτε εφαρμογής.

Η JavaScript είναι επηρεασμένη, όσον αφορά το συντακτικό, από άλλες δημοφιλείς γλώσσες της εποχής που δημιουργήθηκε, όπως η C++ και η Java. Αυτό βοήθησε στο να διευκολύνει την εκμάθησή της γλώσσας. Παρά τις ομοιότητες της JavaScript και της Java, όσον αφορά την ονομασία και το συντακτικό, οι δύο γλώσσες δεν μοιάζουν καθόλου μεταξύ τους και παρουσιάζουν μεγάλες διαφορές ως προς τον σχεδιασμό και την φιλοσοφία τους.

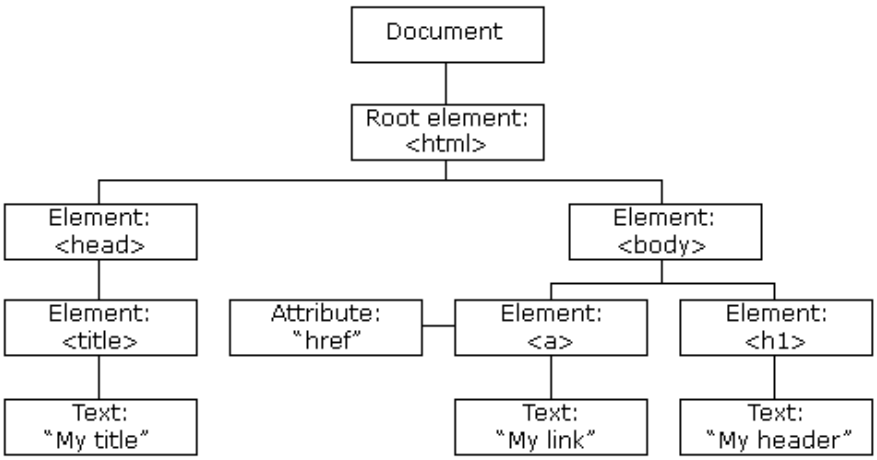
Παρόλο που η JavaScript ξεκίνησε ως μία client-side (από την πλευρά του πελάτη) γλώσσα προγραμματισμού, τώρα πια παρατηρείται μία συνεχόμενη αύξηση της χρήσης της γλώσσας και σε server-side εφαρμογές. Το πιο δημοφιλές περιβάλλον εκτέλεσης JavaScript εφαρμογών στην πλευρά του διακομιστή είναι το Node.js.

Η JavaScript έχει στην διάθεσή της μία μεγάλη ποικιλία από APIs τα οποία επεκτείνουν τις δυνατότητές και τις λειτουργίες της. Το πιο σημαντικό από αυτά, είναι το DOM (Document Object Model) API, που της επιτρέπει να επεξεργάζεται το HTML έγγραφο και να έχει πρόσβαση στα στοιχεία του.

2.4.2 Η JavaScript και το DOM

Το Document Object Model, ή DOM, είναι μια προγραμματιστική διεπαφή που επιτρέπει σε γλώσσες προγραμματισμού, όπως η JavaScript, να έχουν πρόσβαση στα στοιχεία του εγγράφου, να αλλάζουν τις ιδιότητες τους, να δημιουργούν και να διαγράφουν στοιχεία καθώς και να προσθέτουν ή να αφαιρούν events.

Το HTML DOM είναι κατασκευασμένο ως ένα δέντρο φτιαγμένο από αντικείμενα. Κάθε φύλλο αυτού του δέντρου αντιστοιχεί σε ένα HTML element. [11]



Εικόνα 2.6: Το HTML DOM – Δέντρο Αντικειμένων

Το αντικείμενο document είναι η ρίζα του δέντρου και αποτελεί ιδιότητα του αντικειμένου window.

Το HTML DOM δίνει στην JavaScript όλα τα εργαλεία που χρειάζεται για να δημιουργήσει δυναμικές ιστοσελίδες. Πιο συγκεκριμένα:

- Δίνει πρόσβαση σε όλα τα στοιχεία του HTML εγγράφου. Η εύρεση τους μπορεί να γίνει με τις μεθόδους getElementById(), getElementsByTagName(), getElementsByClassName() καθώς και με CSS query selectors, querySelector() και querySelectorAll().
- Δίνει την δυνατότητα αλλαγής των ιδιοτήτων ενός HTML στοιχείου. Για παράδειγμα, με την εντολή 'element.innerHTML = new html content' μπορούμε να αλλάξουμε το περιεχόμενο ενός στοιχείου. Με παρόμοιο τρόπο μπορεί να γίνει η αλλαγή οποιασδήποτε ιδιότητας με την εντολή 'element.attribute = new value'.
- Δίνει την δυνατότητα προσθήκης και διαγραφής HTML στοιχείων. Για παράδειγμα, με την μέθοδο document.createElement(element) δημιουργούμε ένα HTML στοιχείο και στη συνέχει μπορούμε να το προσθέσουμε στο έγγραφο με την μέθοδο document.appendChild(element). Αντίστοιχα υπάρχουν οι μέθοδοι removeChild(), για την αφαίρεση ενός στοιχείου, και replaceChild(new, old) για την αντικατάσταση ενός στοιχείου με ένα άλλο.
- Δίνει την δυνατότητα προσθήκης event listener σε οποιοδήποτε στοιχείο του εγγράφου με την μέθοδο element.addEventListener(event, function).

Το DOM δεν αποτελεί κομμάτι της JavaScript, παρόλα αυτά ο συνδυασμός των δύο τεχνολογιών αποτελεί βασικό θεμέλιο για την ανάπτυξη δυναμικών διαδικτυακών εφαρμογών.

2.4.3 Εξέλιξη της JavaScript

Η JavaScript συνεχώς εξελίσσεται και επεκτείνει τις λειτουργίες και τις δυνατότητες της, έτσι ώστε να μειώνεται ο χρόνος ανάπτυξης νέων εφαρμογών, να είναι πιο εύχρηστη και κατανοητή από τους προγραμματιστές και γενικά για την επίτευξη ενός καλύτερου developer experience.

Τα τελευταία χρόνια λαμβάνει συνεχείς ενημερώσεις. Η πιο μεγάλη και σημαντική νέα έκδοση ήταν η **ECMAScript 2015**.

Αυτή η έκδοση πρόσθεσε τις λέξεις κλειδιά **let** και **const**, που χρησιμοποιούνται για την δημιουργία μεταβλητών. Διαφέρουν από την **var** γιατί οι μεταβλητές που ορίζονται με αυτές είναι προσβάσιμες μόνο μέσα στο ίδιο μπλοκ κώδικα και όχι έξω από αυτό.

Επίσης προστέθηκε ένας νέος τρόπος δημιουργίας συναρτήσεων τα arrow functions (`() => {...}`), που επιτρέπουν τον ορισμό συναρτήσεων με πιο συνοπτικό τρόπο και σε λιγότερες γραμμές κώδικα.

Έγινε δυνατή η δημιουργία κλάσεων αντικειμένων με την λέξη class, η δημιουργία promises καθώς και προστέθηκε η δυνατότητα χρησιμοποίησης default παραμέτρων σε μία συνάρτηση. Επίσης προστέθηκαν οι συλλογές map, που είναι ένα σύνολο από ζεύγη κλειδιών-τιμών (key-value pairs), και set, που είναι ένα σύνολο μοναδικών δεδομένων, δηλαδή κάθε τιμή δεν επαναλαμβάνεται μέσα σε ένα set.

Προστέθηκαν επίσης τα ES modules τα οποία επιτρέπουν την εξαγωγή μιας μεθόδου ή μιας μεταβλητής από ένα αρχείο με την λέξη **export** και την εισαγωγή της σε ένα άλλο διαφορετικό αρχείο με την λέξη **import**.

Με την έκδοση **ECMAScript 2016** προστέθηκε ο τελεστής ******, για τον υπολογισμό ενός αριθμού υψωμένο σε κάποιον εκθέτη, καθώς και η μέθοδος **array.includes()** για τον έλεγχο ύπαρξης μίας τιμής μέσα σε ένα πίνακα.

Η **ECMAScript 2017** πρόσθεσε τις μεθόδους αντικειμένων **object.entries()** και **object.values()** αλλά και την δυνατότητα χρησιμοποίησης των λέξεων-κλειδιών **async** και **await** για την εκτέλεση ασύγχρονου κώδικα, σε πιο απλή και καθαρή μορφή και όχι με την χρησιμοποίηση περίπλοκων Promise chains.

Στην έκδοση **ECMAScript 2018** προστέθηκε ο τελεστής spread (`...`) που κάνει την αντιγραφή αντικειμένων ευκολότερη, ενώ με την **ECMAScript 2020** προστέθηκε ο τύπος δεδομένων **BigInt** για την διαχείριση πολύ μεγάλων αριθμών.

2.4.4 JavaScript Object Notation

Η JavaScript Object Notation ή JSON είναι μία μορφή αρχείου που χρησιμοποιείται για την αποθήκευση και την ανταλλαγή δεδομένων μεταξύ εφαρμογών. Βασίζεται στην JavaScript αλλά είναι ανεξάρτητη από αυτήν και μπορεί να χρησιμοποιηθεί από οποιαδήποτε γλώσσα προγραμματισμού. [12]

Είναι εύκολη στην συγγραφή και την ανάγνωση από τους ανθρώπους αλλά και στην δημιουργία και την επεξεργασία από τις μηχανές. Βασίζεται στις δομές δεδομένων αντικειμένων και πινάκων, ενώ συχνά ένα JSON αρχείο έχει την μορφή πίνακα αντικειμένων.

```json
[
  {
    "ID": "301",
    "Title": "Christmas Theater",
    "Address": "Γαλάτσι, Αττική"
  },
  {
    "ID": "302",
    "Title": "Νέος Ακάδημος",
    "Address": "Athens, Αττική"
  },
  {
    "ID": "303",
    "Title": "Θέατρο Δημήτρης Χορν",
    "Address": "Κολωνάκι, Αττική"
  }
]
```

Εικόνα 2.7: Παράδειγμα Αρχείου JSON

2.5 React

Η React είναι μία ανοιχτού κώδικα βιβλιοθήκη της JavaScript, η οποία χρησιμοποιείται για την δημιουργία user interface (διεπαφή χρήστη) και δημιουργήθηκε από την Meta (πρώην Facebook). Με την React ο κώδικας χωρίζεται σε μικρότερα, πιο διαχειρίσιμα κομμάτια, τα components, και όταν συνδυάζονται μας δίνουν ολόκληρη την εφαρμογή.

2.5.1 JSX

Η JSX (JavaScript XML) είναι μία επέκταση του συντακτικού της JavaScript, που χρησιμοποιεί η React για την περιγραφή του user interface, η οποία επιτρέπει την συγγραφή κώδικα HTML μέσα σε ένα αρχείο JavaScript.

Είναι στην ουσία μία μίξη της HTML και της JavaScript, αφού μέσα στις ετικέτες σημάνσεων της JSX μπορούμε να προσθέσουμε κώδικα JavaScript, ανοίγοντας αγκύλες.

Η JSX δεν αποτελεί προϋπόθεση για να χρησιμοποιήσει κάποιος τις δυνατότητες της React, παρόλα αυτά η χρήση της μειώνει τον χρόνο που απαιτείται για την συγγραφή του κώδικα και συγχρόνως τον κάνει πιο συνοπτικό και ευανάγνωστο.

```
const dateElement = <h1 className="date">The date is {Date()}</h1>
```

Εικόνα 2.8: Παράδειγμα Έκφρασης JSX

Η παραπάνω JSX έκφραση μεταφράζεται στην εντολή **React.createElement()**, η οποία παίρνει ως παραμέτρους τον τύπο του στοιχείο, τις ιδιότητές του και το περιεχόμενο του. Για παράδειγμα, η παραπάνω JSX έκφραση ισοδυναμεί με:

```
const dateElement = React.createElement(
  'h1',
  {className: 'date'},
  `The date is ${Date()}`
)
```

Εικόνα 2.9: Ισοδύναμο JSX έκφρασης με χρήση React.createElement()

Η εντολή αυτή δημιουργεί αντικείμενα τα οποία ονομάζονται React elements, τα οποία περιγράφουν τι πρέπει να υπάρχει στην οθόνη. Η React διαβάζει τα αντικείμενα αυτά και τα χρησιμοποιεί για να κατασκευάσει και να ενημερώσει το DOM. [13]

Η JSX έχει μερικούς πρόσθετους περιορισμούς σε σχέση με την HTML. Πιο συγκεκριμένα κάθε JSX έκφραση πρέπει να επιστρέφει ένα μόνο στοιχείο στην ρίζα του. Για να επιστραφούν πάνω από δύο elements πρέπει να τα περικλείσουμε σε κενά tags <> </> που ονομάζονται React fragments.

Επίσης στην JSX όλα τα στοιχεία θα πρέπει είτε να έχουν ετικέτα κλεισίματος είτε να είναι self-closing. Τέλος, στην ονομασία των ιδιοτήτων δεν επιτρέπεται η χρήση της παύλας και χρησιμοποιείται το στυλ γραφής camelCase. Για παράδειγμα, η ιδιότητα **tab-index** ενός HTML στοιχείου θα γραφεί **tabIndex** στην React. [14]

Τα πλεονεκτήματα της React και της JSX γίνονται πιο εμφανή εάν προσπαθήσουμε να υλοποιήσουμε το παραπάνω HTML στοιχείο με την χρήση απλής JavaScript και των μεθόδων του DOM.

Αρχικά θα πρέπει να βρούμε το HTML element, στο οποίο θέλουμε να προσθέσουμε το νέο στοιχείο, με την εντολή **document.getElementById()**. Στη συνέχεια δημιουργούμε το στοιχείο με την εντολή **document.createElement()** και το κείμενο με την εντολή **document.createTextNode()**. Τέλος, προσθέτουμε το κείμενο με την μέθοδο **appendChild()** στο στοιχείο που μόλις δημιουργήσαμε αυτό με την σειρά του το προσθέτουμε στο root element.

```
const app = document.getElementById('app')
const header = document.createElement('h1')
const text = document.createTextNode(`The date is ${Date()}`)
header.className = 'date'
header.appendChild(headerContent)
app.appendChild(header)
```

Εικόνα 2.10: Προσθήκη στοιχείου με χρήση απλής JavaScript

Το αποτέλεσμα των του παραπάνω κώδικα μπορεί να επιτευχθεί πιο απλά, με λιγότερες γραμμές κώδικα με την χρήση της React και της JSX.

Αρχικά καλούμε την μέθοδο **createRoot** της βιβλιοθήκης **ReactDOM** περνώντας ως παράμετρο το HTML element, στο οποίο θέλουμε να προσθέσουμε το νέο στοιχείο και στη συνέχεια καλούμε την μέθοδο **render** για να το εμφανίσουμε στην οθόνη**.**

```
const app = ReactDOM.createRoot(document.getElementById('app'))
app.render(<h1 className='date'>The date is {Date()}</h1>)
```

Εικόνα 2.11: Προσθήκη στοιχείου με χρήση React και JSX

Παρατηρείται ότι στο πρώτο παράδειγμα δίνουμε οδηγίες για τον τρόπο με τον οποίο θα δημιουργηθεί το στοιχείο, χρησιμοποιώντας μεθόδους του DOM. Το στυλ προγραμματισμού σε αυτήν την περίπτωση είναι προστακτικό (imperative programming).

Αντίθετα, στο παράδειγμα της React χρησιμοποιείται ένα δηλωτικό στυλ προγραμματισμού (declarative programming), δηλαδή δηλώνουμε το αποτέλεσμα που επιθυμούμε να εμφανιστεί στην οθόνη, χωρίς να μας ενδιαφέρει ποια βήματα ακολούθησε η React για να το φέρει εις πέρας.

Για την δημιουργία διεπαφών χρήστη προτιμάται συνήθως το δηλωτικό στυλ προγραμματισμού αφού μειώνονται κατά πολύ οι γραμμές κώδικα που απαιτούνται για την επίτευξη του ίδιου αποτελέσματος και αποφεύγεται η συνεχής επανάληψη συχνά χρησιμοποιούμενων εντολών.

2.5.2 Components

Όπως προαναφέρθηκε, το user interface στην React αποτελείται από μικρότερα, ανεξάρτητα μεταξύ τους, κομμάτια, τα components, με τα οποία χτίζεται η εφαρμογή. Για παράδειγμα, σε μία εφαρμογή η μπάρα πλοήγησης μπορεί να είναι ξεχωριστό component από μια φόρμα ή ένα κουμπί.

Μας επιτρέπουν να γράφουμε κώδικα, ο οποίος είναι επαναχρησιμοποιούμενος και πιο εύκολος στην συντήρηση. Ο διαχωρισμός αυτός του κώδικα κάνει δυνατή αλλαγή και ενημέρωση ενός κομματιού της εφαρμογής χωρίς να επηρεάζονται τα υπόλοιπα components, διευκολύνοντας συγχρόνως και την αποσφαλμάτωση.

Τα components στην React είναι στην ουσία JavaScript μέθοδοι, οι οποίες επιστρέφουν στην έξοδό τους τα HTML στοιχεία τα οποία θα εμφανίζονται στην οθόνη, συνήθως σε JSX. Κάθε τέτοια μέθοδος,

μπορεί να έχει δικιές της μεθόδους και μεταβλητές που χρησιμοποιεί για να επεξεργαστεί την είσοδο που λαμβάνει, έτσι ώστε να παραγάγει την κατάλληλη έξοδο.

```
export default function Greeting(props) {
  return (
    <h1>Hello, {props.name}!</h1>
  )
}
```

Εικόνα 2.12: Παράδειγμα απλού React Component

Κάθε component ορίζεται συνήθως σε δικό του ξεχωριστό αρχείο και εξάγεται από αυτό ως ένα module με την χρήση της λέξης-κλειδί export. Με αυτό τον τρόπο, γίνεται δυνατή η εισαγωγή και η χρησιμοποίηση του component σε άλλο αρχείο με την λέξη-κλειδί import.

Η κλήση ενός component από ένα άλλο γίνεται όπως ακριβώς θα καλούσαμε και ένα HTML στοιχείο, με την χρήση σημάνσεων. Η μόνη διαφορά είναι ότι το όνομα ενός component θα πρέπει να ξεκινάει με κεφαλαίο γράμμα, έτσι ώστε να διαφοροποιούνται τα component, που έχουν δημιουργηθεί από τον προγραμματιστή, από τα native HTML στοιχεία.

Η κλήση του component του παραπάνω παραδείγματος θα γινόταν **<Greeting name=’John’ />**. Όπως στην HTML έτσι και στην React είναι δυνατή η ενθυλάκωση ενός component μέσα σε ένα άλλο. Για την ακρίβεια, θα μπορούσε κανείς να πει ότι είναι η βασική αρχή πάνω στην οποία χτίζονται οι εφαρμογές στην React. Η δυνατότητα αυτή επιτρέπει την εύκολη ενημέρωση και επέκταση της εφαρμογής, καθώς και την δημιουργία εφαρμογών, που παρά το μέγεθος και την πολυπλοκότητα τους, είναι εύκολο να συντηρηθούν.

Στην React η ροή των δεδομένων ακολουθεί μία κατεύθυνση, από τα components που βρίσκονται υψηλότερα στην ιεραρχία προς αυτά που βρίσκονται χαμηλότερα. Αυτό επιτυγχάνεται με το πέρασμα ιδιοτήτων στα components από τον γονέα τους, τα props.

Τα props (σύντομο για properties, δηλαδή ιδιότητες) είναι οι ιδιότητες που έχει ορίσει ο προγραμματιστής για ένα component και χρησιμοποιούνται ως οι είσοδοί του. Τα props ενός component είναι read-only, δηλαδή δεν είναι δυνατή η αλλαγή τους από το ίδιο το component, και λειτουργούν με παρόμοιο τρόπο με τις ιδιότητες ενός HTML στοιχείου.

Στο παραπάνω παράδειγμα, περάσαμε ένα prop με το όνομα ‘name’ και τιμή ‘John’. Το component Greeting λαμβάνει ως παράμετρο ένα αντικείμενο με το όνομα props που περιέχει όλες τις ιδιότητες που έχουν περαστεί, στην προκειμένη περίπτωση την ιδιότητα ‘name’. Το component στη συνέχεια χρησιμοποιεί την τιμή του prop ‘name’, και την εμφανίζει μέσα σε ένα HTML στοιχείο.

2.5.3 State και Event Handlers

Τα props των components είναι read-only πληροφορία η οποία περνιέται ως παράμετρος από τον γονέα του component. Για να γίνει όμως μία web εφαρμογή διαδραστική θα πρέπει κάπου να αποθηκεύεται πληροφορία, η οποία να μπορεί να αλλάξει ανάλογα με τις ενέργειες του χρήστη, έτσι ώστε να αντιπροσωπεύει την παρούσα κατάσταση του component.

Αυτό επιτυγχάνεται με την χρήση του **useState** hook. Τα hooks είναι ειδικές μέθοδοι της React, που ξεκινάνε με την λέξη use, και μας επιτρέπουν να προσθέσουμε επιπλέον λειτουργικότητα σε ένα component, όπως για παράδειγμα την κατάστασή του. Τα hooks δεν μπορούν να κληθούν μέσα σε ένα if statement ή μέσα σε ένα loop αλλά ούτε και από το εσωτερικό μίας μεθόδου.

Το useState hook μας επιτρέπει να αποθηκεύουμε και να ενημερώνουμε πληροφορία, η οποία δεν χάνεται μεταξύ των renders (η απεικόνιση του component στην οθόνη) του component. Παίρνει ως παράμετρο την αρχική τιμή που θέλουμε να έχει η συγκεκριμένη κατάσταση και επιστρέφει έναν πίνακα με δύο στοιχεία. Το πρώτο είναι η μεταβλητή που αντιπροσωπεύει την τιμή της κατάστασης, και το δεύτερο είναι μία μέθοδος που επιτρέπει την ενημέρωσή της.

```
export default function Count() {
  const [count, setCount] = React.useState(0)

  function handleClick() {
    setCount(prevCount => prevCount + 1)
  }

  return (
    <>
      <p>The count is {count}</p>
      <button onClick={handleClick}>Increment</button>
    </>
  )
}
```

Εικόνα 2.13: Παράδειγμα χρήσης useState hook

Στο παραπάνω παράδειγμα δημιουργήσαμε ένα component, το οποίο εμφανίζει την οθόνη την τιμή της μεταβλητής count και ένα κουμπί που αυξάνει την τιμή της κατά ένα.

Η μέθοδος **handleClick** είναι ένας event handler, δηλαδή εκτελείται όταν πυροδοτείται κάποιο event, στην προκειμένη περίπτωση το event onClick του κουμπιού. Αναθέσαμε την συγκεκριμένη μέθοδο στο onClick event δίνοντάς την ως τιμή στην ιδιότητα onClick του κουμπιού. Υπενθυμίζεται ότι για την εισαγωγή JavaScript κώδικα μέσα σε JSX απαιτείται το άνοιγμα αγκύλων.

Η μέθοδος **handleClick** χρησιμοποιεί την μέθοδο **setCount** που επιστράφηκε από το **useState** hook για να ενημερώσει την μεταβλητή count, αυξάνοντάς την κατά ένα.

Ένα ακόμα πολύ σημαντικό hook είναι το **useEffect**, που μας επιτρέπει να εκτελούμε κάποια μέθοδο κάθε φορά που το component κάνει re-render. Μπορούμε επίσης να ορίσουμε εμείς πότε θέλουμε να εκτελείται η μέθοδος, περνώντας ως παράμετρο στην useEffect έναν πίνακα με μεταβλητές. Όποτε μία από αυτές τις μεταβλητές αλλάζει εκτελείται και η μέθοδος.

2.5.4 Πλεονεκτήματα της React

Η React προσφέρει πολλά πλεονεκτήματα και αυτό γίνεται εμφανές από το γεγονός ότι μεγάλες εταιρίες την υιοθετούν για την ανάπτυξη των web εφαρμογών τους, όπως για παράδειγμα το Netflix, το Instagram, το DropBox, το Airbnb. [15]

Μερικά από τα πλεονεκτήματα της React είναι:

- Η απλότητά της και η ευκολία στην εκμάθησή της. Η React είναι μία JavaScript βιβλιοθήκη. Ένας προγραμματιστής που γνωρίζει καλά JavaScript είναι εύκολο να μάθει τις λίγες ιδιαιτερότητες της React και να την χρησιμοποιήσει.
- Ο διαχωρισμός του κώδικα σε components επιτρέπει την εύκολη επαναχρησιμοποίηση και επεκτασιμότητά του, μειώνοντας δραστικά τον χρόνο ανάπτυξης web εφαρμογών.
- Τα components είναι ανεξάρτητα το ένα από το άλλο, που σημαίνει αλλαγές και προβλήματα που μπορεί να προκύψουν σε ένα component, δεν επηρεάζουν ολόκληρη την εφαρμογή.
- Είναι μια μικρή σε μέγεθος και γρήγορη βιβλιοθήκη. Η React κάνει χρήση του virtual DOM, μιας ‘ελαφριάς’ έκδοσης του πραγματικού DOM, που είναι πολύ πιο γρήγορο να ενημερωθεί.
- Αποφεύγεται η απευθείας χρήση των μεθόδων του DOM, οι οποίες επαναλαμβάνονται συχνά και μπορούν να περιπλέξουν τον κώδικα.
- Η μεγάλη δημοτικότητά της ανάμεσα στους προγραμματιστές κάνει εύκολη την εύρεση πληροφοριών για τυχόν προβλήματα που μπορεί να προκύψουν κατά την ανάπτυξη μιας εφαρμογής, διευκολύνοντας έτσι την αποσφαλμάτωση.
- Το μεγάλο εύρος open source βιβλιοθηκών που έχουν αναπτυχθεί για την React κάνει εύκολη την εύρεση έτοιμων λύσεων για συνηθισμένα προβλήματα που προκύπτουν κατά την ανάπτυξη web εφαρμογών.
- Έχει μεγάλη ζήτηση στην αγορά εργασίας.
- Εκτός από την ανάπτυξη web εφαρμογών, μπορεί να χρησιμοποιηθεί και για την ανάπτυξη mobile εφαρμογών με την χρήση της React Native.

2.6 Material UI

Το Material UI (ή MUI) είναι μία ανοιχτού κώδικα βιβλιοθήκη, που δημιουργήθηκε το 2014, και είναι βασισμένη στο Material Design της Google. Το Material Design είναι ένα σύνολο από αρχές και πρότυπα που έχει αναπτύξει η Google για τον βέλτιστο σχεδιασμό web και mobile εφαρμογών.

Η βιβλιοθήκη παρέχει έτοιμα React components, τα οποία έχουν μια ορισμένη εμφάνιση που ακολουθεί τα πρότυπα του Material Design, και μπορούν να ενσωματωθούν γρήγορα και εύκολα σε οποιαδήποτε React εφαρμογή, με ελάχιστες γραμμές κώδικα.

Υπάρχουν components τα οποία χρησιμοποιούνται για την δημιουργία του layout μιας σελίδας, όπως για παράδειγμα το grid, components για την παρουσίαση δεδομένων, όπως το list και το table και components για την εισαγωγή δεδομένων από τον χρήστη, όπως το autocomplete ή το transfer list, τα οποία παρέχουν πιο εξειδικευμένες λειτουργίες από τα native HTML inputs.

Εκτός από τα components, που παρέχουν κάποιο εμφανές κομμάτι UI, υπάρχουν και components, τα οποία παρέχουν βοηθητικές λειτουργίες. Το component **ClickAwayListener** μας επιτρέπει να ανιχνεύσουμε εάν ο χρήστης έκανε click εκτός κάποιου συγκεκριμένου στοιχείου και να εκτελέσουμε κάποια μέθοδο όταν αυτό συμβεί. Με αυτό, για παράδειγμα, μπορούμε να δημιουργήσουμε κάποιο αναδυόμενο παράθυρο, το οποίο να κλείνει όταν ο χρήστης κάνει click έξω από αυτό.

Μια άλλη βοηθητική λειτουργία της βιβλιοθήκης είναι το **useMediaQuery** hook, το οποίο επιτρέπει την εύκολη χρήση των media queries. Με αυτό μπορούμε να αποκρύψουμε τελείως κάποια στοιχεία,

ανάλογα με το μέγεθος της συσκευής, αυξάνοντας έτσι την ταχύτητα της εφαρμογής, αφού ο χρήστης θα χρειαστεί να κατεβάσει λιγότερο όγκο δεδομένων.

Η βιβλιοθήκη επίσης παρέχει την δυνατότητα εξατομίκευσης του κάθε component, έτσι ώστε να προσαρμόζεται η εμφάνισή τους στο στυλ της κάθε εφαρμογής. Εκτός από την μεμονωμένη αλλαγή του κάθε component είναι δυνατή η δημιουργία θεμάτων, που μας επιτρέπουν να αλλάξουμε την εμφάνιση ολόκληρης της εφαρμογής, ορίζοντας, για παράδειγμα, τον τύπο της γραμματοσειράς, το μέγεθός της ή και τα χρώματα που θέλουμε να χρησιμοποιήσουμε.

2.7 Next.js

Η Next.js είναι ένα ανοιχτού κώδικα λογισμικό, το οποίο αναπτύχθηκε από την Vercel, και δημοσιεύτηκε τον Οκτώβριο του 2016. Είναι ένα framework ανάπτυξης web εφαρμογών, που επεκτείνει τις δυνατότητες της React, προσφέροντάς της λειτουργίες όπως routing, server-side rendering, static site generation, καθώς επίσης βελτιστοποιεί την εφαρμογή, έτσι ώστε να φορτώνει πιο γρήγορα και με καλύτερες επιδόσεις. [16]

2.7.1 Σελίδες και Ανάκτηση Δεδομένων

Στην Next.js, μία σελίδα είναι ένα React component, το οποίο το έχουμε εξάγει από ένα JavaScript αρχείο, που βρίσκεται στον φάκελο ‘pages’ της εφαρμογής. Κάθε σελίδα αντιστοιχεί σε μια διαδρομή και βασίζεται στον όνομα του αρχείου. [17]

Για παράδειγμα, εάν δημιουργήσουμε ένα αρχείο με όνομα ‘home.js’, το οποίο εξάγει ένα React component, μέσα στον φάκελο ‘pages’, τότε η σελίδα αυτή θα είναι προσβάσιμη από την διαδρομή ‘/home’.

Η Next.js κάνει pre-render κάθε σελίδα. Αυτό σημαίνει ότι η μετατροπή των React components σε HTML, καθώς και η ανάκτηση των δεδομένων της σελίδας, γίνεται στον server πριν το αποτέλεσμα σταλεί στον πελάτη. Αυτό έχει ως αποτέλεσμα καλύτερες επιδόσεις όσον αφορά την ταχύτητα της εφαρμογής αλλά και το SEO (Search Engine Optimization).

Αντίθετα, σε μία τυπική React εφαρμογή πραγματοποιείται client-side rendering. Δηλαδή, ο πελάτης δέχεται ένα σχεδόν κενό HTML αρχείο και τον JavaScript κώδικα, ο οποίος είναι υπεύθυνος για την δημιουργία των HTML στοιχείων και την συμπλήρωσή τους με τα δεδομένα. Όλα αυτά γίνονται στον περιηγητή και την συσκευή του χρήστη. Η Next.js στέλνει έτοιμο το HTML αρχείο ενώ ο JavaScript κώδικας κάνει την σελίδα διαδραστική.

Υπάρχουν δύο τρόποι με τους οποίους η Next.js πραγματοποιεί το pre-rendering. Ο ένας είναι το server-side rendering και ο άλλος το static site generation. Η διαφορά των δύο βρίσκεται στο πότε ο καθένας παράγει τα HTML αρχεία.

Με το server-side rendering ο HTML κώδικας παράγεται κάθε φορά που ένας χρήστης πλοηγείται σε μία σελίδα, κατά την εκτέλεση της εφαρμογής (runtime). Για να δημιουργήσουμε μία σελίδα, η οποία χρησιμοποιεί server-side rendering, θα πρέπει από το αρχείο της σελίδας (από όπου εξάγεται και το page component) να εξάγουμε μία ασύγχρονη μέθοδο με το όνομα **getServerSideProps.**

Η μέθοδος **getServerSideProps** εκτελείται στον server και δεν επιβαρύνει τον χρήστη. Μέσα σε αυτήν την μέθοδο γίνεται συνήθως η ανάκτηση και η επεξεργασία των δεδομένων, που εμφανίζονται στην σελίδα. Τα δεδομένα αυτά επιστρέφονται από την μέθοδο ως JSON, και περνιούνται στο page component ως η παράμετρος props. Ο τρόπος αυτός rendering χρησιμοποιείται όταν τα δεδομένα της

σελίδας αλλάζουν πολύ συχνά ή όταν αυτά εξαρτώνται από επιλογές του χρήστη και ο αριθμός όλων των πιθανών συνδυασμών είναι μεγάλος.

```
export default 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 } }
}
```

Εικόνα 2.14: Χρήση Μεθόδου getServerSideProps

Με το static site generation, ο HTML κώδικας της σελίδας δημιουργείται μόνο κατά το αρχικό χτίσιμο της web εφαρμογής (build time), και επαναχρησιμοποιείται κάθε φορά που ένας χρήστης πλοηγείται σε αυτήν την σελίδα. Όλες οι σελίδες στην Next.js που δεν χρησιμοποιούν server-side rendering είναι statically generated.

Για να ανακτήσουμε τα δεδομένα που θα χρησιμοποιηθούν σε μία σελίδα με static generation, θα πρέπει να εξάγουμε μία ασύγχρονη μέθοδο με το όνομα **getStaticProps** από το αρχείο της σελίδας. Η **getStaticProps** έχει σχεδόν πανομοιότυπη λειτουργία με την **getServerSideProps**, με την διαφορά ότι εκτελείται μόνο μία φορά, κατά το χτίσιμο της εφαρμογής.

Αυτός ο τρόπος rendering προτιμάται έναντι του server-side, γιατί αφού χτιστεί η σελίδα, αποθηκεύεται σε ένα CDN (Content Delivery Network). Τα CDNs αποθηκεύουν στατικό περιεχόμενο, όπως για παράδειγμα εικόνες και HTML αρχεία, σε διάφορες τοποθεσίες διασκορπισμένες σε ολόκληρο τον κόσμο. Όταν πραγματοποιείται ένα καινούριο αίτημα, η απάντηση προέρχεται από την κοντινότερη γεωγραφική τοποθεσία προς τον χρήστη, με αποτέλεσμα η σελίδα να φορτώσει ταχύτερα και με μικρότερο χρόνο απόκρισης. [18]

Με το static site generation για να γίνει μία μικρή αλλαγή στα δεδομένα μίας σελίδας θα πρέπει δημιουργήσουμε ένα καινούριο build της εφαρμογής έτσι ώστε να εκτελεστεί η μέθοδος **getStaticProps**. Αυτό δεν είναι καθόλου ευέλικτο και μπορεί να γίνει χρονοβόρο σε πολύ μεγάλες εφαρμογές. Η Next.js παρέχει έναν τρόπο για να ανανεώνονται τα δεδομένα κάθε σελίδας ξεχωριστά με την χρήση του Incremental Static Regeneration (σταδιακή στατική αναζωογόνηση).

Το ISR επιτρέπει την επανεκτέλεση της μεθόδου **getStaticProps**, έτσι ώστε να ανακτηθούν τα ενημερωμένα δεδομένα, και στη συνέχεια παράγει εκ νέου τον HTML κώδικα, ο οποίος αντικαθιστά τον παλιό στο CDN.

Για να ενεργοποιηθεί αυτή η λειτουργία θα πρέπει η μέθοδος **getStaticProps**, εκτός από τα props (που περιέχουν τα δεδομένα) της σελίδας, να επιστρέφει ένα επιπλέον πεδίο με όνομα ‘revalidate’ και τιμή

έναν αριθμό, που αντιπροσωπεύει το χρονική συχνότητα σε δευτερόλεπτα, κατά την οποία θέλουμε να γίνεται η ανανέωση της σελίδας.

Για παράδειγμα, εάν μία σελίδα έχει ‘revalidate: 60’, τότε όταν κάποιος χρήστης πλοηγηθεί σε αυτήν, αρχικά θα εμφανιστεί η έκδοση της σελίδας που είναι αποθηκευμένη στο CDN. Στη συνέχεια, αν έχουν περάσει 60 δευτερόλεπτα από την τελευταία ανανέωση, η Next.js ξαναχτίζει τη σελίδα στο παρασκήνιο. Την επόμενη φορά που κάποιος χρήστης πλοηγηθεί σε αυτήν την σελίδα, εμφανίζεται η πιο πρόσφατη, ενημερωμένη, έκδοση. Η ανανέωση πραγματοποιείται το πολύ κάθε 60 δευτερόλεπτα, και μόνο αν κάποιος χρήστης επισκεφθεί την συγκεκριμένη σελίδα.

Πολλές φορές σε μία εφαρμογή, οι διαδρομές είναι δυναμικές, δηλαδή εξαρτώνται από τα δεδομένα της. Παραδείγματος χάρη, αν στα δεδομένα μίας εφαρμογής έχουμε πολλούς ηθοποιούς, τους οποίους θέλουμε να παρουσιάσουμε, δεν θα δημιουργήσουμε για τον καθένα ξεχωριστή σελίδα αλλά μία σελίδα που θα λειτουργεί ως template και θα συμπληρώνεται με τα δεδομένα του κάθε ηθοποιού, ανάλογα την διαδρομή που έχουμε επισκεφθεί.

Για να γνωρίζει η Next.js ποιες σελίδες να δημιουργήσει πρέπει να εξάγουμε από το αρχείο της σελίδας μία ακόμα μέθοδο με το όνομα **getStaticPaths**. Αυτή η μέθοδος λειτουργεί σε συνεργασία με την **getStaticProps**. Μέσα στην μέθοδο αυτή ανακτούμε όλες τις διαδρομές που θέλουμε να δημιουργηθούν κατά το build time (π.χ. από μια βάση δεδομένων), και επιστρέφει έναν πίνακα με όλα τα δυνατά μονοπάτια, ο οποίος περνιέται ως παράμετρος στην **getStaticProps**. Για κάθε μονοπάτι του πίνακα εκτελείται η μέθοδος **getStaticProps** και δημιουργείται η σελίδα που αποθηκεύεται στο CDN.

Σε μεγάλες εφαρμογές δεν συνιστάται να επιστρέφουμε όλα τα δυνατά μονοπάτια από την μέθοδο **getStaticPaths**, γιατί αυτό θα είχε ως αποτέλεσμα το χτίσιμο της εφαρμογής να πάρει υπερβολικά πολύ χρόνο. Αντί αυτού, είναι προτιμότερο να επιλέξουμε ένα υποσύνολο των διαδρομών που θα επιστρέφονται, για παράδειγμα τις πιο δημοφιλείς σελίδες, και ένα επιπλέον πεδίο με όνομα ‘fallback’ και τιμή ‘true’.

Με το πεδίο ‘fallback: true’, όταν ένας χρήστης επισκεφθεί μία διαδρομή η οποία δεν επιστράφηκε από την **getStaticPaths**, και η σελίδα δεν έχει δημιουργηθεί ακόμα, του εμφανίζεται μία εναλλακτική σελίδα. Εν τω μεταξύ η Next.js αρχίζει να χτίζει την σελίδα στο παρασκήνιο και όταν τελειώσει ανακατευθύνει τον χρήστη σε αυτήν. Με τον τρόπο αυτό επιτυγχάνεται η εύκολα επεκτασιμότητα με την Next.js σε μια εφαρμογή, ακόμα κι αν αυτή αποτελείται από χιλιάδες σελίδες.

Η Next.js υποστηρίζει παράλληλα και client-side rendering σε σελίδες που ενδεχομένως να μην χρειάζεται να γίνουν pre-render, όπως για παράδειγμα ένα admin panel ή μία σελίδα ρυθμίσεων μίας εφαρμογής. Αυτό μπορεί να επιτευχθεί κάνοντας την κλήση προς τον server, για την ανάκτηση των δεδομένων, μέσα σε μία **useEffect**. Με τον τρόπο αυτό τα δεδομένα ανακτώνται την στιγμή που ο χρήστης ανοίξει μία σελίδα, και τα περιεχόμενά της ενημερώνονται καθώς τα δεδομένα αλλάζουν.

2.7.2 Βελτιστοποίηση Εικόνων

Η Next.js παρέχει ένα δικό της ειδικό component, με το οποίο μπορεί να γίνει η εμφάνιση των εικόνων σε μία εφαρμογή και χρησιμοποιείται έναντι του html στοιχείου **img**. Το component αυτό παρέχει αυτόματη βελτιστοποίηση των εικόνων όσων αφορά την ταχύτητα με την οποία αυτές φορτώνονται στην σελίδα και βοηθά στην επίτευξη καλύτερων Core Web Vitals.

Τα Core Web Vitals είναι κάποιες μετρικές με τις οποίες υπολογίζεται το πόσο γρήγορη και με καλές επιδόσεις είναι μία web εφαρμογή και επηρεάζει σε μεγάλο βαθμό το SEO (Search Engine

Optimization) και την κατάταξη μιας σελίδας στα αποτελέσματα αναζήτησης της Google, αλλά και το UX (User Experience).

Μερικές από τις βελτιστοποιήσεις που παρέχει το Image component είναι:

- **Βελτιωμένη απόδοση**. Προβάλλεται πάντα εικόνα σωστού μεγέθους για κάθε συσκευή, χρησιμοποιώντας σύγχρονες μορφές εικόνας.
- **Οπτική σταθερότητα**. Το περιεχόμενο μιας σελίδας δεν μετατοπίζεται καθώς οι εικόνες φορτώνονται.
- **Ταχύτερη φόρτωση σελίδας**. Οι εικόνες φορτώνονται μόνο όταν γίνονται ορατές στον χρήστη (lazy loading).
- **Μεγαλύτερη Ευελιξία**. Αλλαγή μεγέθους εικόνας ανά περίσταση, ακόμη και για εικόνες που είναι αποθηκευμένες σε απομακρυσμένους διακομιστές. [19]

Για να γίνει δυνατή η χρησιμοποίηση εικόνων που προέρχονται από εξωτερικές πηγές θα πρέπει να προσθέσουμε το domain name της κάθε πηγής στο αρχείο **next.config.js**, που βρίσκεται μέσα στο project. Μόνο εικόνες από τις πηγές που έχουν συμπεριληφθεί θα φορτωθούν στην εφαρμογή μας. Με αυτόν το τρόπο προσθέτουμε ένα επιπλέον επίπεδο προστασίας ενάντια σε κακόβουλους χρήστες.

2.7.3 API Routes

Η Next.js μας επιτρέπει να δημιουργήσουμε ένα API για την εφαρμογή μας χωρίς να χρειαστεί να στήσουμε από το μηδέν ένα ολόκληρο backend. Αυτό επιτυγχάνεται με τα API routes, τα οποία είναι στην ουσία JavaScript αρχεία που βρίσκονται στον φάκελο `pages/api`. Αυτά τα αρχεία εκτελούνται μόνο στον server κι έτσι δεν επηρεάζει το μέγεθος της client-side εφαρμογής.

Κάθε τέτοιο αρχείο αποτελεί ένα endpoint, το οποίο μπορεί να εκτελεστεί όταν κάνουμε μία κλήση προς το αντίστοιχο μονοπάτι: `/api/{όνομα αρχείου}`. Για παράδειγμα το αρχείο `/pages/api/hello.js` θα εκτελεστεί στον διακομιστή όταν κάνουμε μία κλήση στο μονοπάτι `/api/hello`.

```
export default function handler(req, res) {
  res.status(200).json({ content: "Hello World" });
}
```

Εικόνα 2.15 Παράδειγμα αρχείου API Route

Το παραπάνω μονοπάτι θα επιστρέψει μία JSON απάντηση με κωδικό 200. Κάθε αρχείο θα πρέπει να εξάγει μία default συνάρτηση η οποία και θα εκτελείται όταν καλείτε το συγκεκριμένο endpoint. Η συνάρτηση αυτή δέχεται δύο παραμέτρους, τις req (request) και res (response). Μέσα σε αυτήν μπορεί να εκτελεστεί στον διακομιστή JavaScript κώδικας με την χρήση της node.js, να ανακτηθούν δεδομένα και στη συνέχεια να επιστραφεί το αποτέλεσμα στον client.

Εκτός από την δημιουργία ενός ολοκληρωμένου API, τα API routes μπορούν να χρησιμοποιηθούν και μεμονωμένα για την επίτευξη διάφορων λειτουργιών.

Μια από τις πιο συχνές χρήσεις τους είναι η απόκρυψη ευαίσθητων δεδομένων από τον χρήστη. Για παράδειγμα, εάν χρησιμοποιείται στην εφαρμογή κάποιο microservice, το οποίο απαιτεί κάποιο μυστικό

κλειδί για να υπάρχει πρόσβαση σε αυτό, τότε η κλήση προς αυτό το microservice θα μπορούσε να γίνει μέσω ενός API route, αποκρύπτοντας έτσι το μυστικό κλειδί από τους χρήστες.

2.8 Επίλογος

Σε αυτό το κεφάλαιο έγινε μία αναλυτική περιγραφή των γλωσσών και των τεχνολογιών που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής. Παρουσιάστηκαν τα βασικά τους στοιχεία, οι δυνατότητες και τα πλεονεκτήματά τους, καθώς και ο τρόπος με τον οποίο αυτές εξελίχθηκαν στην πάροδο του χρόνο και διαμόρφωσαν το οικοσύστημα που χρησιμοποιείται σήμερα για την ανάπτυξη web εφαρμογών

Κεφάλαιο 3ο: Αρχιτεκτονική και Σχεδιασμός

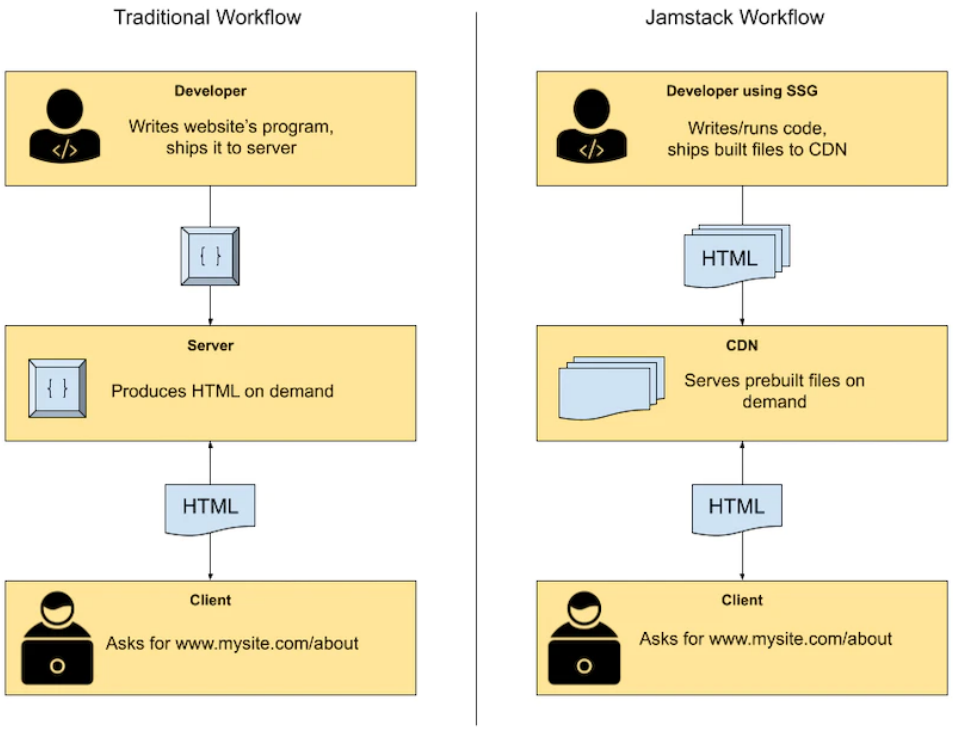
3.1 Εισαγωγή

Στο κεφάλαιο που ακολουθεί θα περιγραφεί η αρχιτεκτονική την οποία ακολουθεί η εφαρμογή καθώς και θα γίνει μία αναφορά στα APIs τα οποία χρησιμοποιεί για την λειτουργία της. Θα αναφερθούν επίσης τα εργαλεία και ο τρόπος με τον οποίο σχεδιάστηκαν οι διεπαφές χρήστη (User Interface) έτσι ώστε να δημιουργήσουν την καλύτερη δυνατή εμπειρία για τον χρήστη.

3.2 Αρχιτεκτονική Jamstack

Το Jamstack είναι μία αρχιτεκτονική, που ακολουθούν πολλές σύγχρονες web εφαρμογές, και έχει σχεδιαστεί έτσι ώστε να τις κάνει γρηγορότερες, με μεγαλύτερη ασφάλεια και ευκολότερο να επεκταθούν. Χρησιμοποιεί πολλά από τα εργαλεία και τις έννοιες που αρέσει στους developers να χρησιμοποιούν, αυξάνοντας έτσι την παραγωγικότητά τους. [20]

Η βασική αρχή της συγκεκριμένης αρχιτεκτονικής είναι ότι οι σελίδες της εφαρμογής παραδίδονται στατικά μέσω ενός CDN, ενώ το δυναμικό περιεχόμενο και η διαδραστικότητα της εφαρμογής παρέχεται μέσω της JavaScript. Η διεπαφή χρήστη είναι αποσυνδεδεμένη από τα υπόλοιπα συστήματα της εφαρμογής και η επικοινωνία με αυτά γίνεται μέσω API.



Εικόνα 3.1 Παραδοσιακή Ιστοσελίδα vs Jamstack [21]

3.2.1 Βασικά Στοιχεία

Το όνομα Jamstack προέρχεται από τα τρία βασικά στοιχεία που έχει κάθε εφαρμογή όταν ακολουθεί την συγκεκριμένη αρχιτεκτονική:

1) **JavaScript**. Η βασική γλώσσα του διαδικτύου την οποία χρησιμοποιούν οι περιηγητές και εμπλουτίζει το στατικό περιεχόμενο της σελίδας με δυναμικά δεδομένα και διαδραστικότητα. Παράλληλα μέσω αυτής είναι δυνατή η επικοινωνία με τα backend συστήματα μίας εφαρμογής**.** Εδώ εντάσσονται και οι βιβλιοθήκες δημιουργίας διεπαφών χρήστη όπως είναι η React, η Vue και η Angular.

2) **APIs**. Η μεγάλη πληθώρα από backend services και συστήματα, τα οποία κάνουν διαθέσιμη την λειτουργία τους στον κόσμο μέσω ενός API, είναι ένας από τους σημαντικότερους παράγοντες της τόσο ταχείας ανάπτυξης της Jamstack αρχιτεκτονικής. Η δυνατότητα αξιοποίησης αυτών των υπηρεσιών, οι οποίες έχουν δημιουργηθεί από ειδικούς του εκάστοτε τομέα, κάνει την ανάπτυξη νέων εφαρμογών πολύ γρηγορότερη και με λιγότερο ρίσκο. Πολλές λειτουργίες, οι οποίες θα ήταν χρονοβόρες και περίπλοκες για να υλοποιηθούν, μπορούν να ανατεθούν σε τρίτους. Παράδειγμα τέτοιων λειτουργιών είναι η αυθεντικοποίηση των χρηστών, η πραγματοποίηση πληρωμών, η αναζήτηση και πολλά άλλα.

3) **Markup**. Το HTML περιεχόμενο της κάθε σελίδας. Όπως έχει ήδη αναφερθεί, στην Jamstack αρχιτεκτονική, η κάθε HTML σελίδα είναι στατική, δηλαδή έχει δημιουργηθεί εκ των προτέρων, και είναι αποθηκευμένη σε κάποιο CDN έτοιμο να παραδοθεί σε κάποιο client. Συνήθως ο HTML κώδικας δεν γράφεται από κάποιο developer, αλλά παράγεται από εξειδικευμένα frameworks, τα static site generators. Μερικά από τα πιο γνωστά τέτοια frameworks είναι η Next.js, το Gatsby, η Nuxt και το Eleventy.

3.2.2 Σημασία των APIs

Τα APIs είναι η καρδιά της αρχιτεκτονικής. Το πλεονέκτημα τους είναι ότι είναι όλα ανεξάρτητα μεταξύ τους και η αλλαγή του ενός δεν επηρεάζει την λειτουργία του άλλου. Έτσι μπορεί να επιτευχθεί η αρμονική συνεργασία ενός μεγάλου εύρους συστημάτων.

Στην εφαρμογή των θεατρικών παραστάσεων χρησιμοποιείται πληθώρα API, τα οποία και χρησιμοποιούνται σαν κομμάτια σε ένα παζλ για να δημιουργηθεί το τελικό αποτέλεσμα.

Η ανάκληση των δεδομένων, τα οποία αποτελούνται από θεατρικές παραστάσεις, ηθοποιούς και θέατρα γίνεται μέσω ενός API. Το API αυτό είναι υπεύθυνο για την επικοινωνία με την βάση δεδομένων, καθώς και την επεξεργασία τους, ανάλογα με το endpoint που έχει κληθεί, πριν την τελική τους αποστολή στην εφαρμογή.

Με αυτόν τον τρόπο αποφεύγεται η απευθείας επικοινωνία της εφαρμογής με την βάση δεδομένων, προσδίδοντάς της έτσι ένα επιπλέον επίπεδο ασφαλείας. Περίπλοκες λειτουργίες και υπολογισμοί πραγματοποιούνται από το API μειώνοντας έτσι την πολυπλοκότητα της εφαρμογής. Ο διαχωρισμός αυτός κάνει πιο ξεκάθαρο τον ρόλο του κάθε συστήματος και απλοποιεί σημαντικά την λογική και την αναγνωσιμότητα του κώδικα.

Εκτός από το παραπάνω βασικό API, χρησιμοποιούνται από την εφαρμογή και διάφορα άλλα APIs, που εξυπηρετούν άλλες λειτουργίες.

Το Google Maps API (maps.googleapis.com) χρησιμοποιήθηκε για τον υπολογισμό αποστάσεων από θέατρα, καθώς και για την εμφάνιση δυναμικών χαρτών. Η ανάπτυξη τέτοιων λειτουργιών από την αρχή

θα ήταν εξαιρετικά περίπλοκη και χρονοβόρα, αλλά η χρήση αυτού του API κάνει την προσθήκη τους εύκολη και απλή.

Χρησιμοποιήθηκε επίσης το The Movie Database API (api.themoviedb.org), για την προβολή διάφορων πληροφοριών για τους καλλιτέχνες, όπως για παράδειγμα βιογραφικά σημειώματα, ημερομηνίες γέννησης, φωτογραφίες και άλλα.

Με το News API (newsapi.org) γίνεται δυνατή η εύρεση διάφορων νέων και ειδήσεων, σχετικών με τον χώρο του θεάτρου από πολλαπλές πηγές, έτσι ώστε να προβληθούν μέσα στην εφαρμογή.

Από τα παραπάνω, γίνεται εύκολα κατανοητό ότι, με την χρήση των API, επιταχύνεται σημαντικά η ανάπτυξη νέων εφαρμογών. Μέσω των APIs μπορούν να ενσωματωθούν σε οποιαδήποτε εφαρμογή μεμονωμένες λειτουργίες, που θα χρειάζονταν πολύ χρόνο για να υλοποιηθούν εξαρχής.

3.2.3 Αυθεντικοποίηση API

Για να ελέγχεται ποιος έχει πρόσβαση στα APIs, αλλά και να περιορίζεται ο αριθμός των κλήσεων που κάθε χρήστης μπορεί να πραγματοποιήσει προς αυτά, είναι απαραίτητο να υπάρχει ένας τρόπος αυθεντικοποίησης των χρηστών, ιδιαίτερα αν το API είναι προσβάσιμο μέσω του διαδικτύου.

Ένας τρόπος για να επιτευχθεί η παραπάνω λειτουργία είναι με την χρήση API keys. Ένα API key είναι ένα μοναδικό αναγνωριστικό, το οποίο δημιουργείται για έναν προγραμματιστή ή μία εφαρμογή και στην συνέχεια περνιέται ως παράμετρος με κάθε request προς το API. Στη συνέχεια, το API key ελέγχεται από τον διακομιστή για να διασφαλιστεί ότι το αίτημα προέρχεται από εξουσιοδοτημένο χρήστη και μόνο τότε εκτελείται το αίτημά του.

Ένας άλλος τρόπος είναι με την χρήση username και password. Αυτός ο τρόπος αυθεντικοποίησης είναι γνωστός ως basic authentication. Με κάθε αίτημα προς το API, περνιούνται μέσα στο header το username και το password, τα οποία ελέγχονται από τον διακομιστή, ο οποίος στην συνέχεια στέλνει μία απάντηση. Συνιστάται η χρήση HTTPS για την κρυπτογράφηση των δεδομένων που ανταλλάσσονται μεταξύ πελάτη και διακομιστή.

3.2.4 Πλεονεκτήματα Jamstack

Η Jamstack αρχιτεκτονική επιφέρει πολλά πλεονεκτήματα, τα οποία και συμβάλλουν στην γρήγορη ανάπτυξή της και στην ταχεία αύξηση της δημοτικότητάς της. Μερικά από αυτά τα πλεονεκτήματα είναι:

- **Ασφάλεια**. Η αρχιτεκτονική μειώνει τα συστήματα και τα εργαλεία που πρέπει ο προγραμματιστής να διατηρεί από μόνος του, μειώνοντας έτσι τους διακομιστές που απαιτούνται για την λειτουργία μιας εφαρμογής. Έτσι οι χρήστες έχουν λιγότερες ευκαιρίες για κακόβουλες ενέργειες.

  Δυναμικά συστήματα και εργαλεία παρέχονται από τρίτους, οι οποίοι είναι εξειδικευμένοι και έχουν μεριμνήσει για την ασφάλεια τον υπηρεσιών που παρέχουν. [22]

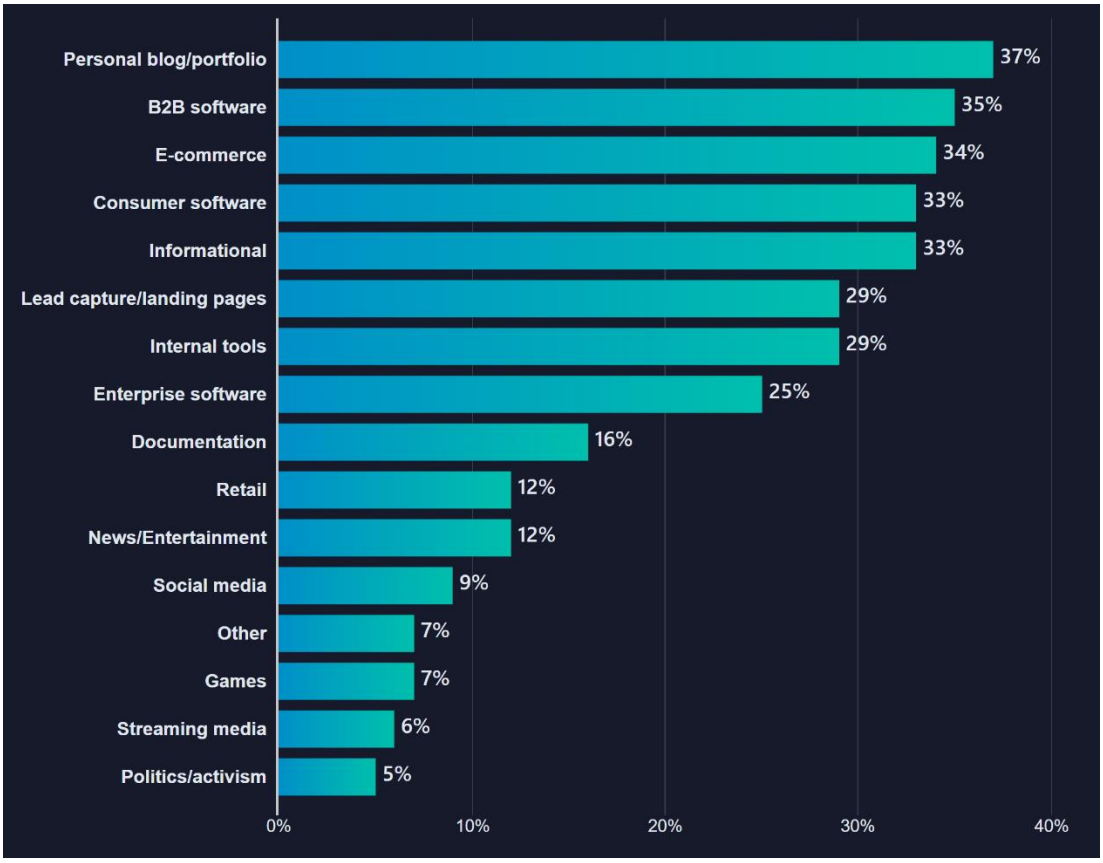
- **Επεκτασιμότητα (Scalability)**. Άλλες αρχιτεκτονικές χειρίζονται τους μεγάλους αριθμούς χρηστών κάνοντας cache τις πιο δημοφιλείς σελίδες. Η Jamstack το παρέχει αυτό από μόνη της κάνοντας cache την εφαρμογή σε ένα CDN. Οι εφαρμογές αποτελούνται από στατικές σελίδες και είναι εύκολο να προστεθούν νέες, χωρίς ιδιαίτερο κόστος και δυσκολία.

- **Καλύτερη απόδοση**. Το γεγονός ότι οι Jamstack εφαρμογές εξυπηρετούνται από κάποιο CDN σημαίνει ότι ο κάθε χρήστης λαμβάνει τα αρχεία από κάποιον κοντινό διακομιστή με μεγάλη ταχύτητα και απόκριση, το οποία συμβάλει σε μία καλύτερη εμπειρία για τον χρήστη.

- **Συντηρησιμότητα**. Υπάρχουν πολυάριθμες υπηρεσίες που κάνουν host Jamstack εφαρμογές και έτσι απλοποιείται η διαδικασία της ανάρτησης της εφαρμογής στο διαδίκτυο και η ανάγκη για συντήρηση κάποιου διακομιστή.

- **Κόστος**. Το hosting των στατικών αρχείων μίας Jamstack εφαρμογής είναι σημαντικά μικρότερο σε σχέση με παραδοσιακές εφαρμογές που απαιτούν έναν αφοσιωμένο διακομιστή να τις εξυπηρετεί.

Τα παραπάνω πλεονεκτήματα έχουν συμβάλλει στην υιοθέτηση της Jamstack αρχιτεκτονικής για την ανάπτυξη εφαρμογών από ένα ευρύ φάσμα κατηγοριών. Από μικρά blogs και προσωπικές σελίδες, μέχρι και μεγάλα ηλεκτρονικά καταστήματα και πλατφόρμες κοινωνικής δικτύωσης.



Εικόνα 3.2 Είδη Εφαρμογών που χρησιμοποιείται η Jamstack [23]

3.2.5 Deployment Εφαρμογής

Το deployment είναι η διαδικασία κατά την οποία ο κώδικας που έχουμε γράψει μεταφέρεται από τον υπολογιστή μας ή κάποια αποθήκη κώδικα (repository) σε κάποιον διακομιστή, έτσι ώστε να χτιστεί η εφαρμογή και να γίνει διαθέσιμη για προβολή στο διαδίκτυο.

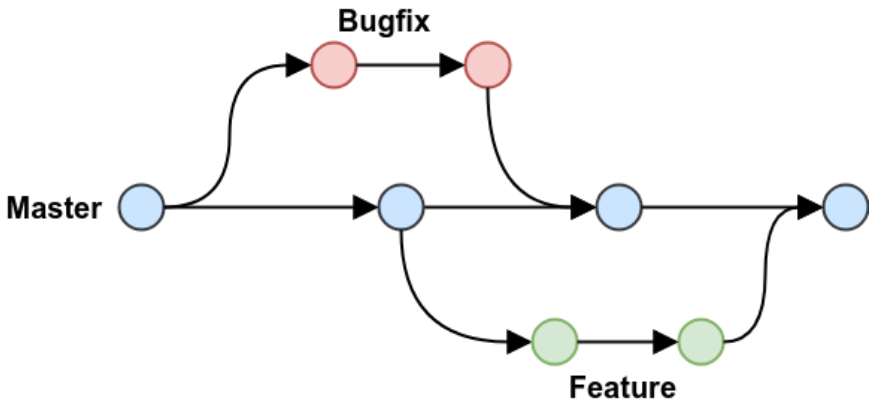
Έχουν αναπτυχθεί διάφορες υπηρεσίες, οι οποίες απλοποιούν την παραπάνω διαδικασία, αφού αυτοματοποιούν πολλές από τις ενέργειες που απαιτούνται για την πραγματοποίησή της. Μία από αυτές τις υπηρεσίες είναι το Netlify. Μέσω αυτού μπορεί, πολύ εύκολα, κάποιος να κάνει deploy μία εφαρμογή στο διαδίκτυο. Το μόνο που απαιτείται είναι η σύνδεση του Netlify με ένα GitHub repository, το οποίο περιέχει τον κώδικα της εφαρμογής.

Το GitHub είναι μία web εφαρμογή, η οποία φιλοξενεί Git repositories στο διαδίκτυο. Το Git είναι ένα ανοιχτού κώδικα λογισμικό, το οποίο χρησιμοποιείται ως σύστημα ελέγχου εκδόσεων. Μέσω αυτού γίνεται δυνατή η παρακολούθηση των αλλαγών μιας εφαρμογής στην πάροδο του χρόνο και η καλύτερη οργάνωση και διαχείριση του κώδικα.

Το Git επιτρέπει σε πολλούς προγραμματιστές να δουλεύουν πάνω στον ίδιο κώδικα χωρίς να δημιουργείται χάος, ενώ παράλληλα εξασφαλίζεται η ακεραιότητα των δεδομένων και προστίθεται μια δικλίδα ασφαλείας, σε περίπτωση που κάτι πάει στραβά, αφού μπορεί ανά πάσα ώρα να γίνει η ανάκτηση μίας προηγούμενης έκδοσης της εφαρμογής.

Κάθε Git repository εμπεριέχει ένα βασικό branch, το οποίο αντιπροσωπεύει και την εφαρμογή που είναι διαθέσιμη στο διαδίκτυο. Οι προγραμματιστές στη συνέχεια μπορούν να δημιουργήσουν νέα branches, αντιγράφοντας στην ουσία το βασικό σε κάποια συγκεκριμένη χρονική στιγμή.

Ο κάθε προγραμματιστής εργάζεται στο δικό του branch, για παράδειγμα για να αναπτύξει κάποια καινούρια λειτουργία ή να διορθώσει κάποιο σφάλμα, απομονωμένα από την υπόλοιπη εφαρμογή, δηλαδή χωρίς να επηρεάζουν την εφαρμογή που είναι αναρτημένη στο διαδίκτυο ή και τα υπόλοιπά branches. Όταν οριστικοποιηθούν οι αλλαγές του branch και διασφαλιστεί η λειτουργία του, τότε ενσωματώνεται στο κύριο branch.



Εικόνα 3.3 Git Branches

Έτσι, με την συνεργασία του GitHub και του Netlify, μπορεί κάποιος να κάνει deploy μία web εφαρμογή στο διαδίκτυο, με μία απλοποιημένη και εύχρηστη διαδικασία, χωρίς να απαιτούνται εξειδικευμένες γνώσεις.

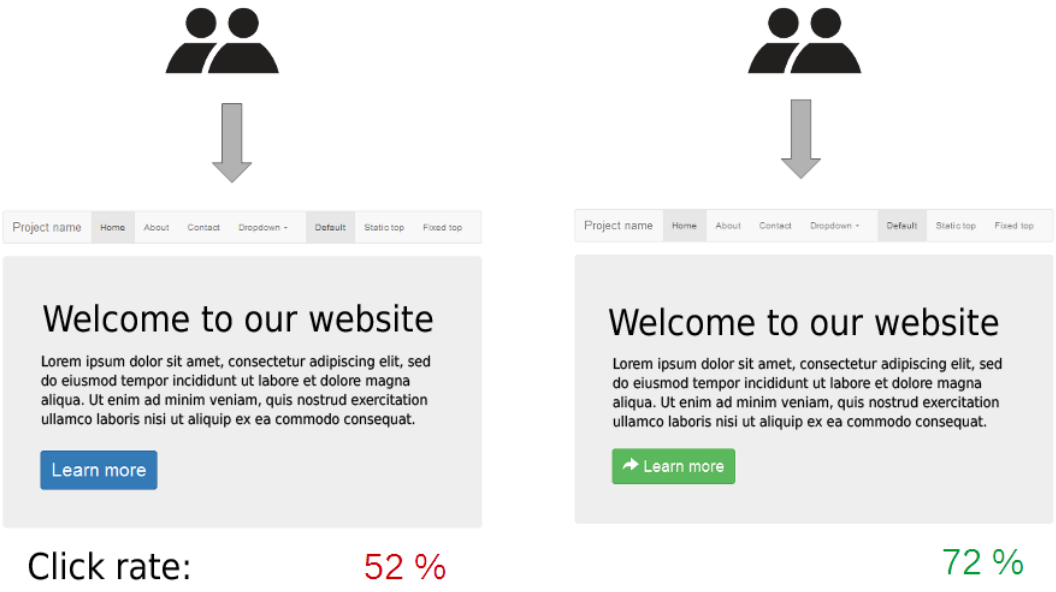
Το Netlify σαρώνει τον κώδικα του συνδεδεμένου repository και προσδιορίζει αυτόματα τι είδους εφαρμογή εμπεριέχεται σε αυτό. Στη συνέχεια, ανάλογα με τα αποτελέσματα του παραπάνω βήματος, προχωράει στο βήμα του χτισίματος της εφαρμογής, οι οποία και τελικά δημοσιεύεται στο διαδίκτυο. Το Netlify παρέχει δωρεάν domain name για την εφαρμογή, καθώς και δωρεάν πιστοποιητικό SSL

(Secure Sockets Layer) για την κρυπτογράφηση των δεδομένων της εφαρμογής, ενισχύοντας έτσι την ασφάλεια των δεδομένων των χρηστών.

Μία από τις σημαντικότερες λειτουργίες, που παρέχει το Netlify, είναι το αυτόματο CI/CD (Continuous Integration / Continuous Delivery). Με την λειτουργία αυτή, κάθε φορά που ανιχνεύεται κάποια αλλαγή στον κώδικα του κύριου branch, τότε το Netlify αυτόματα χτίζει εκ νέου την εφαρμογή, ενσωματώνοντας αυτές τις αλλαγές και κάνει deploy την εφαρμογή, χωρίς να απαιτείται η παρέμβαση κάποιου προγραμματιστή. Με αυτόν τον τρόπο αυξάνεται η παραγωγικότητα και επιταχύνεται η ανάπτυξη της εφαρμογής.

Μία άλλη σημαντική λειτουργία είναι η δυνατότητα προεπισκόπησης των branches, πριν αυτά ενσωματωθούν στο κύριο branch, το οποίο αντιπροσωπεύει και την τελική εφαρμογή. Κάθε branch, για το οποίο έχει γίνει pull request (αίτημα για την ενσωμάτωσή του στο κύριο branch), χτίζεται ξεχωριστά από το κύριο, και γίνεται διαθέσιμο για προεπισκόπηση, έτσι ώστε να ανιχνευτούν τυχόν σφάλματα, που προέκυψαν κατά την διαδικασία, πριν αυτά φτάσουν στον τελικό χρήστη.

Παρέχεται επίσης η δυνατότητα πραγματοποίησης A/B split testing. Με το A/B testing δημιουργούνται δύο εκδόσεις της ίδιας εφαρμογής, οι οποίες διαφέρουν μεταξύ τους σε κάποια κομμάτια, έτσι ώστε να αποφασιστεί ποια έκδοση είναι καλύτερη και να βελτιωθεί το user experience. Για παράδειγμα, μπορεί ανάμεσα στις δύο εκδόσεις, να διαφοροποιούνται τα χρώματα της εφαρμογής ή η διάταξη των στοιχείων μίας σελίδας.



Εικόνα 3.4 Παράδειγμα A/B Testing [24]

Οι μισοί χρήστες της εφαρμογής βλέπουν την μία έκδοση, ενώ οι υπόλοιποι την άλλη. Στη συνέχεια, μετριούνται διάφορα στατιστικά, όπως πόση ώρα χρησιμοποίησε την εφαρμογή ο κάθε χρήστης, πόσοι χρήστες επέστρεψαν στην εφαρμογή μετά την πρώτη τους επίσκεψη ή ακόμα και πόσες πωλήσεις πραγματοποιήθηκαν, αν πρόκειται για κάποιο ηλεκτρονικό κατάστημα. Τα στατιστικά αυτά

συγκρίνονται ανάμεσα στις δύο εκδόσεις, κι έτσι μπορεί να βγει ένα συμπέρασμα για το ποια έκδοση έχει τις καλύτερες επιδόσεις και θα πρέπει να συνεχίσει να χρησιμοποιείται.

3.3 Σχεδιασμός

Ο αρχικός σχεδιασμός μίας εφαρμογής παίζει σημαντικό ρόλο καθ' όλη τη διάρκεια ανάπτυξής της, και καθορίζει σε μεγάλο βαθμό το πόσο επιτυχημένη είναι. Η εμφάνιση μιας εφαρμογής, είναι σε πολλές περιπτώσεις εξίσου σημαντική με την λειτουργικότητά της, αφού είναι το πρώτο πράγμα που βλέπει ο χρήστης και σχηματίζει την πρώτη του εντύπωση.

Όλο και περισσότερες επιχειρήσεις επενδύουν σημαντικά στον καλύτερο σχεδιασμό των web εφαρμογών τους, έτσι να τις κάνουν πιο εύχρηστες και να προσφέρουν καλύτερη εμπειρία στους χρήστες, και να αυξήσουν τις πωλήσεις της επιχείρησης.

3.3.1 Βασικές Αρχές

Παρακάτω θα γίνει μία αναφορά σε κάποιες από τις βασικές αρχές σχεδιασμού, που όταν ακολουθούνται συμβάλλουν σε μία πιο, οπτικά, ευχάριστη εμπειρία για τους χρήστες, αφού κάνει το περιεχόμενο κάθε σελίδας πιο ευανάγνωστο και την εφαρμογή πιο εύκολη στην πλοήγηση. Αυτές οι αρχές είναι:

- **Κενό διάστημα**. Η χρήση κενού διαστήματος ανάμεσα στα στοιχεία μίας σελίδας, αλλά και σε άλλα σημεία, όπως για παράδειγμα η απόσταση του περιεχομένου από την άκρη της σελίδας, είναι από τις πιο σημαντικές ενέργειες που μπορεί να κάνει κάποιος για να βελτιώσει την εμφάνιση μίας εφαρμογής.

- **Στοίχιση**. Η σωστή στοίχιση και τοποθέτηση των στοιχείων μέσα σε μία σελίδα, κάνει την εφαρμογή πιο εύχρηστη και προσδίδει έναν επαγγελματισμό, ενώ αντίθετα μια λανθασμένη στοίχιση μπορεί να την κάνει να φανεί πρόχειρη και, πολλές φορές, έχει ως αποτέλεσμα να μπερδεύει τον χρήστη.

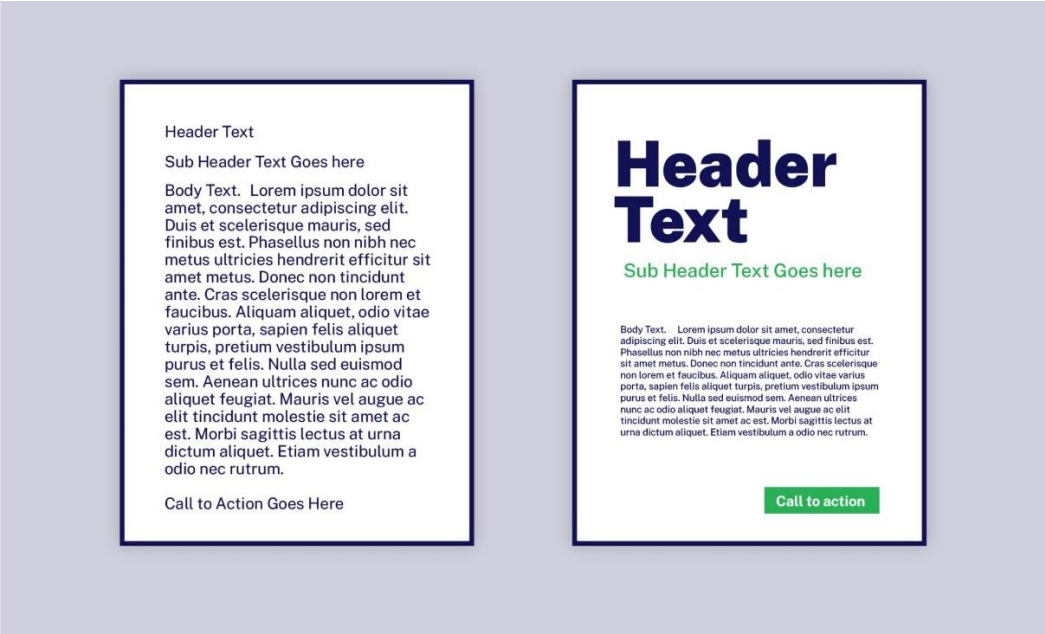
- **Αντίθεση**. Η ύπαρξη ενός ικανοποιητικού επιπέδου αντίθεσης ανάμεσα στα γράμματα, αλλά και σε όλα τα στοιχεία μίας σελίδας, και στον φόντο, έτσι ώστε να είναι ξεκάθαρος ο διαχωρισμός ανάμεσά τους, δεν είναι σημαντική μόνο για την επίτευξη ενός καλύτερου οπτικού αποτελέσματος, αλλά επιτρέπει και σε χρήστες με προβλήματα όρασης να χρησιμοποιούν την εφαρμογή με ευκολία.

- **Κλίμακα**. Το μέγεθος των στοιχείων μίας σελίδας παίζει σημαντικό ρόλο, αφού μέσω αυτού μπορούμε να δείξουμε ότι μία ομάδα στοιχείων, που έχουν ίδιο μέγεθος είναι όμοια με κάποιον τρόπο, ή αντιθέτως μια ομάδα στοιχείων με διαφορετικό μέγεθος δεν ανήκουν στην ίδια κατηγορία. Για παράδειγμα, οι γραμματοσειρά των τίτλων σε μία σελίδα είναι μεγαλύτερη από την γραμματοσειρά μίας παραγράφου.

- **Τυπογραφία**. Η επιλογή της κατάλληλης γραμματοσειράς, ανάλογα με το είδος της εφαρμογής, είναι ένα από τα σημαντικότερα κομμάτια κατά την διαδικασία του σχεδιασμού. Ιδιαίτερη σημασία θα πρέπει να δοθεί και στο μέγεθος της γραμματοσειράς, την απόσταση των γραμμάτων (letter spacing), καθώς και στο ύψος της κάθε γραμμής (line height). Δεν συνιστάται να χρησιμοποιούνται πάνω από δύο διαφορετικές γραμματοσειρές στην ίδια σελίδα, αφού με αυτόν τον τρόπο το περιεχόμενο είναι πιθανό να γίνει κουραστικό και δυσνόητο. Η εφαρμογή διάφορων γραμματοσειρών σε μία εφαρμογή έχει διευκολυνθεί με υπηρεσίες, όπως το Google Fonts, που επιτρέπει το κατέβασμα και την χρήση μιας γραμματοσειράς, κατά την φόρτωση της σελίδας, ακόμα και αν αυτή δεν είναι εγκατεστημένη στο σύστημα του χρήστη.

- **Χρώμα**. Τα χρώματα είναι από τα πρώτα στοιχεία που αντιλαμβάνεται ο χρήστης όταν επισκέπτεται μία ιστοσελίδα, και επηρεάζει σε μεγάλο βαθμό την εμπειρία του. Η χρήση μεγάλου αριθμού χρωμάτων στην ίδια σελίδα είναι καλό να αποφεύγεται. Κάθε χρώμα μπορεί να προκαλέσει διάφορα συναισθήματα στον χρήστη, ενώ παράλληλα έχει και συμβολικό χαρακτήρα. Τα ζεστά χρώματα, όπως το κόκκινο, το κίτρινο και το πορτοκαλί, μπορούν να προκαλέσουν συναισθήματα όπως άνεση και ζεστασιά, ενώ τα ψυχρά χρώματα, όπως το πράσινο, το μπλε και το μοβ, συνδέονται με συναισθήματα ηρεμίας. [25]

- **Οπτική ιεραρχία**. Κάθε στοιχείο μέσα σε μία σελίδα έχει διαφορετικό επίπεδο σημασίας. Για παράδειγμα, ο τίτλος μίας ιστοσελίδας είναι πιο σημαντικός από μία παράγραφο. Η οπτική ιεραρχία είναι ο τρόπος με τον οποίο προσδιορίζουμε το πόσο σημαντικό είναι το κάθε στοιχείο σε σχέση με τα άλλα. Στο παραπάνω παράδειγμα, ο τίτλος έχει μεγαλύτερο μέγεθος γραμματοσειράς από την παράγραφο. Η οπτική ιεραρχία, εκτός από το μέγεθος των στοιχείων, μπορεί να επιτευχθεί και με την χρήση χρωμάτων, αντίθεσης, ή και με συνδυασμό αυτών. Με αυτόν τον τρόπο μπορούμε να εστιάσουμε την προσοχή του χρήστη σε συγκεκριμένα στοιχεία, και βοηθάμε στον καλύτερο προσανατολισμό του μέσα στην εφαρμογή.



Εικόνα 3.5 Παράδειγμα Οπτικής Ιεραρχίας [26]

3.3.2 User Experience

Το user experience έχει να κάνει με την εμπειρία που αποκομίζει ο χρήστης κατά την αλληλεπίδρασή του με την εφαρμογή, και έχει ως στόχο την βελτίωση αυτής της εμπειρίας έτσι ώστε οι χρήστες να συνεχίσουν να την χρησιμοποιούν.

Ο σχεδιασμός UX διαφέρει από τον σχεδιασμό διεπαφών χρήστη, αν και συχνά οι όροι συγχέονται μιας και είναι παρόμοιοι. Παρότι ο σχεδιασμός διεπαφών χρήστη είναι μέρος της εμπειρίας του χρήστη, αναφέρεται μόνο στο οπτικό κομμάτι της εφαρμογής. Το UX είναι μία πιο ευρεία έννοια που επικεντρώνεται περισσότερο στον τρόπο με τον οποίο ο χρήστης αλληλοεπιδρά με την εφαρμογή, καθώς και το πόσο ευχάριστη είναι αυτή η αλληλεπίδραση.

Σύμφωνα με τον Peter Morville, πρωτοπόρο στους κλάδους της αρχιτεκτονικής πληροφοριών και της εμπειρίας χρήστη, ένας καλός UX σχεδιασμός προϋποθέτει τα παρακάτω βασικά χαρακτηριστικά [27]. Η εφαρμογή θα πρέπει να είναι:

- **Χρήσιμη**. Η εφαρμογή θα πρέπει να είναι χρήσιμη προς τον χρήστη και να καλύπτει κάποια του ανάγκη.
- **Χρησιμοποιήσιμη**. Η εφαρμογή θα πρέπει να είναι απλή και εύκολη στην χρήση.
- **Επιθυμητή**. Η εφαρμογή θα πρέπει να είναι ευπαρουσίαστη και να προκαλεί θετικά συναισθήματα στον χρήστη.
- **Findable**. Η πλοήγηση θα πρέπει να είναι απλή και η δυνατότητα για εντοπισμό και αναζήτηση πληροφοριών να γίνεται με ευκολία.
- **Προσβάσιμη**. Το περιεχόμενο θα πρέπει να είναι προσβάσιμο σε όλους, ακόμα και σε άτομα με ειδικές ανάγκες.
- **Αξιόπιστη**. Οι χρήστες θα πρέπει να πιστεύουν τις πληροφορίες που παρουσιάζονται στην εφαρμογή και να την εμπιστεύονται.

3.3.3 Προσβασιμότητα

Το διαδίκτυο είναι σχεδιασμένο έτσι ώστε να μπορεί να χρησιμοποιείται από όλους τους ανθρώπους, ανεξάρτητα από το hardware, το λογισμικό, τη γλώσσα, τον τόπο που ζούνε ή τις ικανότητές τους. Όταν το διαδίκτυο μπορεί να χρησιμοποιηθεί με ευκολία από άτομα με ειδικές ανάγκες, τότε μπορούμε να πούμε ότι έχει επιτύχει τον παραπάνω στόχο.

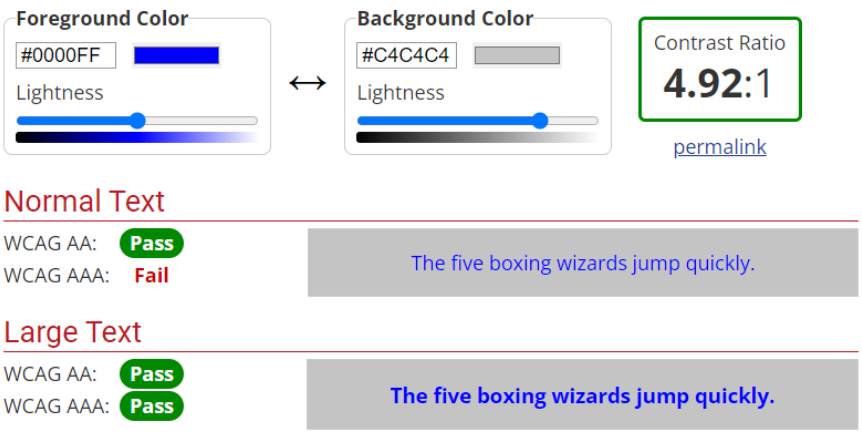
Η ανάπτυξη του διαδικτύου έχει αφαιρέσει πολλά από τα εμπόδια στην επικοινωνία και την αλληλεπίδραση που αντιμετωπίζουν πολλοί άνθρωποι στον φυσικό κόσμο. Ωστόσο, όταν οι ιστοσελίδες, οι εφαρμογές, οι τεχνολογίες ή τα εργαλεία έχουν σχεδιαστεί με κακό τρόπο, μπορούν να δημιουργήσουν εμπόδια που αποκλείουν κάποιους ανθρώπους από τη χρήση τους. [28]

Η προσβασιμότητα θα πρέπει να αποτελεί κύριο μέλημα για προγραμματιστές που θέλουν να δημιουργήσουν ιστοσελίδες και web εφαρμογές υψηλής ποιότητας, τα οποία δεν αποκλείουν άτομα από τη χρήση των προϊόντων και των υπηρεσιών τους.

Για την πιο αποτελεσματική και ομοιόμορφη επίτευξη της προσβασιμότητας στο διαδίκτυο, έχουν αναπτυχθεί ένα σύνολο προτύπων και κανόνων, που ονομάζονται Web Content Accessibility Guidelines (WCAG).

Οι κανόνες αυτοί είναι οργανωμένοι σε τέσσερις κατηγορίες: 1) αντίληψη, πόσο εύκολο είναι ο χρήστης να δει το περιεχόμενο, 2) λειτουργικότητα, δηλαδή η ευκολία στην πλοήγηση, όχι μόνο με το ποντίκι, αλλά και με το πληκτρολόγιο, 3) κατανόηση, το περιεχόμενο θα πρέπει να είναι εύκολα κατανοητό, 4) στιβαρότητα, το σύστημα θα πρέπει να είναι συμβατό με τις συσκευές του χρήστη.

Ο κάθε κανόνας έχει κάποια κριτήρια επιτυχίας, τα οποία βαθμολογούνται σε τρία επίπεδα: Α, ΑΑ, και ΑΑΑ. Για παράδειγμα, όσον αφορά την αντίθεση των στοιχείων, θα πρέπει για κανονικού μεγέθους γράμματα να έχει αναλογία 4.5:1 για να είναι στο επίπεδο ΑΑ. ενώ θα πρέπει να είναι 7:1 για να επιτευχθεί επίπεδο ΑΑΑ.



Εικόνα 3.6 Εργαλείο μέτρησης αντίθεσης γραμμάτων στην ιστοσελίδα https://webaim.org/

Ένα ακόμη παράδειγμα καλής προσβασιμότητας είναι η παροχή περιγραφής για περιεχόμενο που δεν είναι κείμενο, έτσι ώστε η πληροφορία να είναι προσβάσιμη σε ανθρώπους με προβλήματα όρασης. Η δομή της εφαρμογής θα πρέπει να είναι τέτοια, έτσι ώστε το περιεχόμενο να είναι κατανοητό και προσπελάσιμο με το πληκτρολόγιο, για ανθρώπους που χρησιμοποιούν βοηθητικές τεχνολογίες, όπως η μετατροπή του κειμένου σε ομιλία (text to speech).

Υπάρχουν πολλά διαδικτυακά εργαλεία, με τα οποία μπορεί κάποιος να εντοπίσει τυχόν προβλήματα προσβασιμότητας σε μία εφαρμογή, όπως για παράδειγμα το Google Lighthouse. Το εργαλείο αυτό βαθμολογεί την εφαρμογή σε θέμα προσβασιμότητας από το μηδέν έως το εκατό, ενώ παράλληλα εντοπίζει τα προβλήματα και προτείνει λύσεις για αυτά.

3.3.4 Responsive Web Design

Το responsive web design είναι ένα σύνολο μεθόδων και τεχνικών που έχει ως στόχο την ανάπτυξη διαδικτυακών εφαρμογών, οι οποίες μπορούν να προβληθούν και να προσαρμόζουν την εμφάνισή τους στο μέγεθος της κάθε οθόνης.

Παλαιότερα οι ιστοσελίδες σχεδιάζονταν για ένα συγκεκριμένο μέγεθος οθόνης, με αποτέλεσμα να φαίνονται άσχημες σε άλλα μεγέθη. Με την ανάπτυξη της κινητής τηλεφωνίας, όλο και περισσότερα μεγέθη οθονών εμφανίζονταν στην αγορά, κι έτσι εμφανίστηκε η έννοια του responsive web design, για την δημιουργία εφαρμογών που είναι εύχρηστες και εμφανίσιμες, ανεξάρτητα από το μέγεθος της οθόνης.

Το responsive web design δεν είναι μία ξεχωριστή τεχνολογία αλλά μια προσέγγιση σχεδιασμού, η οποία χρησιμοποιεί τις ήδη υπάρχουσες τεχνολογίες, όπως την HTML και την CSS, για να αλλάξει την δομή και την διάταξη της σελίδας.

Ο όρος responsive design επινοήθηκε από τον Ethan Marcotte το 2010 και περιγράφει τη χρήση τριών τεχνικών σε συνδυασμό [29]:

1. Η πρώτη τεχνική είναι η χρήση fluid grid, όπως για παράδειγμα το CSS grid ή το flexbox, τα οποία μεγαλώνουν και μικραίνουν ανάλογα με τον διαθέσιμο χώρο της σελίδας.
2. Η δεύτερη τεχνική είναι η χρήση fluid εικόνων. Χρησιμοποιώντας την CSS ιδιότητα max-width: 100% σε μία εικόνα εξασφαλίζεται ότι η εικόνα δεν θα μεγαλώσει πέρα του αρχικού της μεγέθους, υποβαθμίζοντας την ποιότητα της στην πορεία. Συγχρόνως, όταν το στοιχείο που περιέχει την εικόνα γίνεται στενό, τότε αυτή μικραίνει και δεν βγαίνει έξω από τα όριά του.

3. Το τρίτο βασικό στοιχείο είναι τα media queries. Αντί να υπάρχει μία διάταξη για όλα τα μεγέθη οθόνης, η διάταξη αλλάζει. Έτσι, για παράδειγμα, μπορούμε να έχουμε μία διάταξη με τρεις ή περισσότερες στήλες για έναν desktop υπολογιστή, και διάταξη με μία στήλη για μία κινητή συσκευή.

Σχεδόν σε όλες τις σύγχρονες ιστοσελίδες υπάρχει το HTML meta tag:

<meta name="viewport" content="width=device-width,initial-scale=1" />

Αυτό το tag ορίζει το viewport width στον περιηγητή, έτσι ώστε να είναι ίσο με το πραγματικό πλάτος της συσκευής. Αυτό είναι αναγκαίο, αφού οι κινητές συσκευές δεν ορίζουν από μόνες τους το πραγματικό τους πλάτος. Όταν κυκλοφόρησαν οι πρώτες κινητές συσκευές οι περισσότερες ιστοσελίδες δεν ήταν mobile friendly. Έτσι, τα κινητά όριζαν το viewport width στα 980 pixel, και οι ιστοσελίδες εμφανίζονταν ως μία zoomed-out εκδοχή της desktop ιστοσελίδας. Οι χρήστες στην συνέχεια μπορούσαν να κάνουν zoom στα κομμάτια που τους ενδιέφεραν. [30]

Αυτό στη συνέχεια δημιούργησε προβλήματα, αφού με την εξάπλωση του responsive web design, τα media queries δεν λειτουργούσαν με τον αναμενόμενο τρόπο στα κινητά. Για παράδειγμα, εάν έχει οριστεί ένα media query για οθόνες μικρότερες των 480 pixel, τότε αυτό δεν θα εφαρμοστεί ποτέ σε ένα κινητό, αφού τα κινητά θέτουν από προεπιλογή το πλάτος τους στα 980 pixel. Ορίζοντας width=device-width, παρακάμπτετε αυτό το προεπιλεγμένο πλάτος με το πραγματικό πλάτος της συσκευής και τα media queries αρχίζουν να δουλεύουν κανονικά.

3.3.5 Figma

Το Figma, είναι μία web εφαρμογή, που χρησιμοποιείται για την σχεδίαση διεπαφών χρήστη και την δημιουργία πρωτοτύπων. Εστιάζει κυρίως στην συνεργασία πολλών σχεδιαστών σε πραγματικό χρόνο, αφού τους επιτρέπει να δουλεύουν στο ίδιο αρχείο ταυτόχρονα. Το γεγονός ότι εκτελείται στον περιηγητή σημαίνει ότι μπορεί να το χρησιμοποιήσει οποιοσδήποτε, ανεξαρτήτως λειτουργικού συστήματος, αφού δεν απαιτεί την εγκατάσταση κάποιου προγράμματος. Ο διαμοιρασμός των σχεδίων γίνεται με ευκολία, αφού για κάθε Figma αρχείο δημιουργείται ένας υπερσύνδεσμος, ο οποίος μπορεί να σταλεί σε οποιονδήποτε μέσω του διαδικτύου.

Η δημιουργία πρωτοτύπων σχεδίων μίας εφαρμογής είναι μία πολύ σημαντική διαδικασία, αφού επιτρέπει την προσομοίωση του τελικού προϊόντος, πριν αρχίσει η ανάπτυξή της. Η δημιουργία πρωτοτύπων είναι μία πιο γρήγορη διαδικασία από αυτήν της ανάπτυξης, και έτσι μπορεί να σώσει εξοικονομήσει σημαντικό χρόνο και να μειώσει το κόστος.

Μέσω τον πρωτοτύπων, ο πελάτης μπορεί να δει πώς θα μοιάζει η τελική εφαρμογή και να ζητήσει αλλαγές σε αυτό το στάδιο, όπου αυτές μπορούν να υλοποιηθούν με σχετική ευκολία. Μπορεί, επίσης, να εκτιμηθεί με μεγαλύτερη ακρίβεια το χρόνος και το κόστος της κάθε εφαρμογής, και το κατά πόσο είναι δυνατή η υλοποίηση των απαιτήσεων του πελάτη.

Τα πρωτότυπα μπορούν να δοθούν σε χρήστες για δοκιμή, έτσι ώστε να εντοπιστούν τυχόν προβλήματα εμφάνισης και λειτουργικότητας της εφαρμογής. Με αυτόν τον τρόπο μπορεί να βελτιωθεί σημαντικά η ποιότητα της εφαρμογής και η γενική εμπειρία του χρήστη. Η γνώμη του τελικού χρήστη παίζει καθοριστικό ρόλο στην πορεία της εφαρμογής, αφού εάν οι χρήστες σταματήσουν να την χρησιμοποιούν τότε θα πάψει να υπάρχει.

Από την άλλη πλευρά, οι προγραμματιστές αποκτούν μία πιο γενική εικόνα της εφαρμογής και μπορούν να διασπάσουν το σύνολο του έργου σε επιμέρους, πιο διαχειρίσιμες, λειτουργίες. Έτσι επιτυγχάνεται

η καλύτερη οργάνωση της ανάπτυξης της εφαρμογής, αφού οι προγραμματιστές μπορούν να δουλεύουν παράλληλα σε διαφορετικά τμήματα της, χωρίς να επηρεάζουν ο ένας τον άλλο, επιταχύνοντας την διαδικασία.

Οι προγραμματιστές δεν χρειάζεται να δαπανούν χρόνο για την επιλογή χρωμάτων, γραμματοσειρών ή για την δημιουργία της διάταξης της κάθε σελίδας και των διαστημάτων ανάμεσα στα στοιχεία. Μπορούν έτσι να επικεντρωθούν στην υλοποίηση των σχεδίων, αφού όλες οι αποφάσεις έχουν ήδη παρθεί από εξειδικευμένους web σχεδιαστές. Μέσω του Figma είναι δυνατή η εξαγωγή CSS κώδικα από τα σχέδια. Για παράδειγμα, εάν υπάρχει ένα κουμπί στα σχέδια είναι δυνατόν να εξαχθούν οι CSS ιδιότητες του όπως το padding, το background color, το font-size, το box-shadow και πολλά άλλα.

3.4 Επίλογος

Σε αυτό το κεφάλαιο αναλύθηκε η αρχιτεκτονική και ο σχεδιασμός της εφαρμογής. Πιο συγκεκριμένα, αναφερθήκαμε στην αρχιτεκτονική Jamstack, και αναλύθηκαν τα βασικά της στοιχεία, καθώς και τα πλεονεκτήματά της, ενώ δόθηκε ιδιαίτερη έμφαση στην σημασία των APIs που είναι ένα από βασικά στοιχεία της αρχιτεκτονικής.

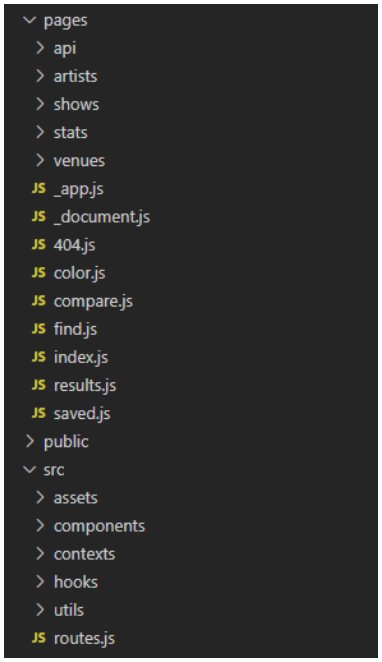
Στο δεύτερο κομμάτι του κεφαλαίου, επικεντρωθήκαμε στον σχεδιασμό της εφαρμογής. Περιεγράφηκαν οι βασικές αρχές ενός καλού σχεδιασμού, ενώ τονίστηκε η σημασία του user experience, της προσβασιμότητας και του responsive web design για την δημιουργία μιας επιτυχημένης εφαρμογής. Τέλος, έγινε μία αναφορά στα οφέλη που προσφέρει το Figma, ένα εργαλείο σχεδίασης διεπαφών χρήστη, στα αρχικά στάδια δημιουργίας της εφαρμογής

Κεφάλαιο 4ο: Περιγραφή της Εφαρμογής

Σε αυτό το κεφάλαιο θα γίνει μία αναλυτική περιγραφή της εφαρμογής. Θα παρουσιαστούν οι επιμέρους σελίδες, καθώς και τα components που χρησιμοποιήθηκαν για την δημιουργία τους.

4.1 Δομή της Εφαρμογής

Στο παρακάτω σχήμα περιγράφεται η ιεραρχία των αρχείων της εφαρμογής. Κάποιες από τις επιλογές στα ονόματα των αρχείων είναι από επιλογή του προγραμματιστή, όπως για παράδειγμα ο φάκελος src που περιέχει ένα μεγάλο μέρος του κώδικα της εφαρμογής. Άλλοι φάκελοι, όπως ο φάκελος pages, έχει προκαθοριστεί από την Next.js και χρησιμοποιείται για την δημιουργία των διαδρομών της εφαρμογής. Κάθε αρχείο μέσα στον φάκελο pages αντιπροσωπεύει μία σελίδα.



Εικόνα 4.1 Ιεραρχία των Φακέλων

Κάποια JavaScript αρχεία έχουν επίσης συγκεκριμένες ονομασίες. Το αρχείο **404.js** είναι μία σελίδα, η οποία αντικαθιστά την προεπιλεγμένη σελίδα σε περίπτωση που μία διαδρομή της εφαρμογής δεν υπάρχει. Με αυτόν τον τρόπο μπορούμε να εξατομικεύσουμε τις σελίδες σφάλματων, έτσι ώστε να ταιριάζουν οπτικά με την υπόλοιπη εφαρμογή. Παρόμοια λειτουργία μπορεί να επιτευχθεί και για την αντικατάσταση της σελίδας σε περίπτωση σφάλματος του διακομιστή, δημιουργώντας μία σελίδα με όνομα **500.js**.

Ιδιαίτερη σημασία παρουσιάζει και το αρχείο **_document.js**, μέσω του οποίου μπορούμε να προσθέσουμε meta tags στο head του HTML αρχείου. Το αρχείο αυτό εκτελείται στον διακομιστή κι έτσι δεν υπάρχει πρόσβαση σε API του client, όπως για παράδειγμα οι onClick ιδιότητες ή το local storage.

```
class MyDocument extends Document {
  render() {
    return (
      <Html>
        <Head>
          <link rel="icon" href="/favicon.ico" sizes="any" />
          <link rel="icon" href="/icon.svg" type="image/svg+xml" />
          <link rel="apple-touch-icon" sizes="180x180" href="/apple-touch-icon.png" />
          <link rel="icon" type="image/png" sizes="32x32" href="/favicon-32x32.png" />
          <link rel="icon" type="image/png" sizes="16x16" href="/favicon-16x16.png" />
          <link rel="manifest" href="/site.webmanifest" />
          <link rel="preconnect" href="https://fonts.googleapis.com" />
          <link rel="preconnect" href="https://fonts.gstatic.com" crossOrigin="true" />
          <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;500;700&display=swap" rel="stylesheet" />
        </Head>
        <body>
          <Main />
          <NextScript />
        </body>
      </Html>
    )
  }
}
```

Εικόνα 4.2 Αρχείο _document.js

Μέσα στο αρχείο document θα πρέπει να τοποθετούνται tags που χρησιμοποιούνται ευρέως στην εφαρμογή, όπως για παράδειγμα για τον ορισμό favicon ή για την φόρτωση γραμματοσειρών από εξωτερικές πηγές, στην προκείμενη περίπτωση από το Google Fonts. Η προσθήκη τίτλου σελίδας και meta description θα πρέπει να γίνεται σε επίπεδο σελίδας, αφού αυτά είναι διαφορετικά σε κάθε περίπτωση.

Στον φάκελο folder μπορεί να γίνει αποθήκευση στατικών αρχείων, όπως για παράδειγμα εικόνων. Εδώ αποθηκεύονται επίσης τα εικονίδια που χρησιμοποιούνται για το favicon, καθώς και το αρχείο robots.txt, το οποίο δίνει πληροφορίες στις μηχανές αναζήτησες για το ποιες σελίδες της εφαρμογής χρειάζεται να επισκεφθούν.

Ο φάκελος **pages/api** είναι ακόμη ένας προκαθορισμένος φάκελος της Next.js. Τα αρχεία που περιέχει λειτουργούν ως endpoints και ο κώδικάς τους εκτελείται στον διακομιστή, δηλαδή ο client δεν έχει πρόσβαση σε αυτά και δεν επιβαρύνει το μέγεθος της εφαρμογής που εκτελείται στον περιηγητή.

Στον φάκελο **assets** είναι αποθηκευμένα CSS αρχεία, καθώς και τα JSS αρχεία (CSS in JS), που είναι ένας τρόπος δημιουργίας στυλ συνδυάζοντας την CSS με την JavaScript, και χρησιμοποιούνται για να εξατομικεύσουν την εμφάνιση των διαφόρων επαναχρησιμοποιούμενων React components που βρίσκονται στον αντίστοιχο φάκελο.

Στον φάκελο **hooks** φιλοξενούνται αρχεία, τα οποία εμπεριέχουν λογική που είναι κοινή σε διάφορα components, κι έτσι με την χρησιμοποίησή τους αποφεύγεται η αχρείαστη επανάληψη κώδικα. Αντίστοιχα στον φάκελο **utils** υπάρχουν διάφορες JavaScript μέθοδοι που χρησιμοποιούνται σε ολόκληρη την εφαρμογή.

4.1.1 Η Ρίζα της Εφαρμογής

Το αρχείο **_app.js** αποτελεί το entry point όλης της εφαρμογής, αφού εμπεριέχει όλα τα άλλα components και είναι υπεύθυνο για την αρχικοποίηση της κάθε σελίδας. Μέσα σε αυτό το αρχείο μπορεί να γίνει η εισαγωγή global CSS, καθώς και ο καθορισμός της διάταξης της εφαρμογής, δηλαδή

στοιχείων της διεπαφής που είναι κοινά σε όλες τις σελίδες. Ένα τέτοιο παράδειγμα σε μία web εφαρμογή είναι η μπάρα πλοήγησης, που βρίσκεται συνήθως στην κορυφή της κάθε σελίδας.

```
import Layout from "../src/components/Layout"
import "../src/assets/css/global.css"
import { ThemeContextProvider } from "../src/contexts/ThemeContext"

function MyApp({ Component, pageProps }) {
  return (
    <ThemeContextProvider>
      <Layout>
        <Component {...pageProps} />
      </Layout>
    </ThemeContextProvider>
  )
}
```

Εικόνα 4.3 Αρχείο _app.js

Το **ThemeContextProvider** κάνει διαθέσιμα δεδομένα για το θέμα της εφαρμογής, όπως για παράδειγμα τα χρώματα και το μέγεθος της γραμματοσειράς, σε όλα τα στοιχεία παιδιά του. Στην προκειμένη περίπτωση περικλείει ολόκληρη την εφαρμογή, που σημαίνει ότι τα δεδομένα είναι διαθέσιμα σε όλα τα components.

Το context είναι μία λειτουργία της React που επιτρέπει στα components να έχουν πρόσβαση σε κάποιο συγκεκριμένο state, χωρίς την μεταβίβασή του από γονέα σε παιδί, μέσω των props, όπως συμβαίνει τυπικά σε μία React εφαρμογή. Η λειτουργία αυτή είναι ιδιαίτερα χρήσιμη για δεδομένα που χρησιμοποιούνται από όλη την εφαρμογή, όπως είναι τα θέματα, καθώς έτσι αποφεύγεται η μεταβίβασή τους μέσω των props, που είναι ιδιαίτερα δύσκολη όταν το component βρίσκεται βαθιά στην ιεραρχία της εφαρμογής.

```
export const ThemeContext = createContext();

export function ThemeContextProvider(props){
    const [secondaryColor, setSecondaryColor] = useState();

    return (
        <ThemeProvider theme={DarkTheme(secondaryColor)}>
            <ThemeContext.Provider value={{ secondaryColor, setSecondaryColor }}>
                {props.children}
            </ThemeContext.Provider>
        </ThemeProvider>
    )
}
```

Εικόνα 4.4 Παράδειγμα Χρήσης React Context

Για την δημιουργία React context, απαιτείται η χρησιμοποίηση της μεθόδου **createContext()**, η οποία επιστρέφει ένα context αντικείμενο. Το αντικείμενο αυτό έχει την ιδιότητα Provider, το οποίο είναι ένα React component, όπου θα περικλείσουμε όλα τα άλλα components που θέλουμε να έχουν πρόσβαση στο συγκεκριμένο context.

Στη προκειμένη περίπτωση έχουμε τις μεταβλητές **secondaryColor** και **setSecondaryColor** τις οποίες περνάμε στο value prop του **ThemeContext.Provider**. Οι μεταβλητές αυτές μπορούν στη συνέχεια να χρησιμοποιηθούν από οποιοδήποτε component παιδί, με την χρήση του hook της React **useContext**. Για παράδειγμα μέσα σε ένα component θα καλούσαμε την εντολή: { secondaryColor, setSecondaryColor } = useContext(ThemeContext).

Το component **ThemeProvider** είναι ένα component της βιβλιοθήκης Material UI και είναι απαραίτητο για την χρήση της. Δέχεται ως prop ένα αντικείμενο που περιέχει τις πληροφορίες του θέματος και είναι αναγκαίες για την σωστή λειτουργία των υπόλοιπων component που προσφέρει η βιβλιοθήκη.

4.1.2 Layout Component

Το Layout είναι ένα component το οποίο παρέχει στην εφαρμογή την διάταξη της και είναι ο γονέας της κάθε σελίδας. Περιέχει δύο άλλα components το Navbar και το Sidebar, τα οποία είναι πάντα ορατά ανεξάρτητα σε ποια διαδρομή της εφαρμογής βρισκόμαστε.

```
const Layout = ({ children }) => {
  return (
    <>
      <CssBaseline />
      <DrawerContextProvider>
        <Navbar />
        <Sidebar />
      </DrawerContextProvider>
      <main>
        {children}
      </main>
    </>
  );
}
```

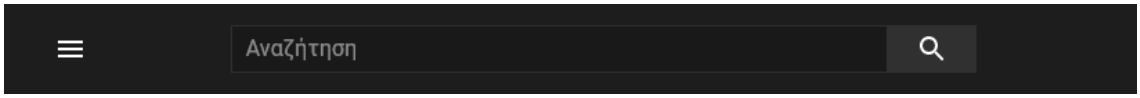
Εικόνα 4.5 Layout Component

Όλα τα React components πρέπει να περικλείονται σε ένα και μόνο στοιχείο γονέα, για αυτό και στο παραπάνω παράδειγμα γίνεται η χρήση των κενών tag. Τα κενά tags είναι μία συντομογραφία του <React.Fragment> και χρησιμοποιείται για να αντιμετωπιστεί το παραπάνω πρόβλημα. Τα React fragments χρησιμοποιούνται για την διευκόλυνσή μας και δεν φαίνονται στην τελική σελίδα ως HTML στοιχεία.

Το CssBaseline είναι ένα component του Material UI και χρησιμοποιείται για την κανονικοποίηση των διάφορων CSS κανόνων που υπάρχουν σε κάθε περιηγητή από προεπιλογή, έτσι ώστε να εξασφαλίζεται μια συνεπής εμφάνιση της εφαρμογής, ανάμεσα στους περιηγητές που αυτή εκτελείται.

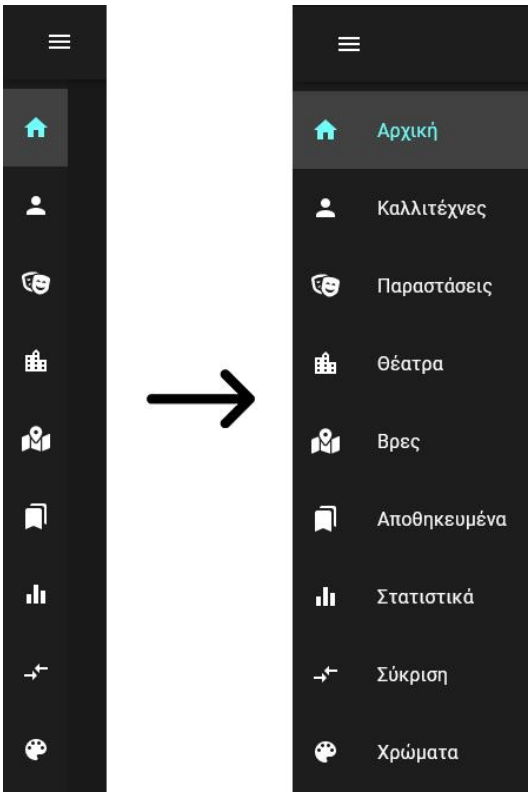
Όλα τα React components έχουν πρόσβαση σε ένα ειδικό prop που ονομάζεται **children**. Αυτό το prop αντιπροσωπεύει όλα τα στοιχεία παιδιά του component. Το **Layout** component παίρνει το children prop, σε αυτήν την περίπτωση την κάθε σελίδα, και τα εμφανίζει μέσα στο **main** tag.

Το Navbar είναι ένα component που εμφανίζεται στην κορυφή της κάθε σελίδας και περιέχει την μπάρα αναζήτησης, καθώς και ένα κουμπί για να ανοιγοκλείνει το sidebar. Η αναζήτηση πραγματοποιείται περνώντας τον όρο αναζήτησης ως query στην διαδρομή της σελίδας με τα αποτελέσματα.



Εικόνα 4.6 Μπάρα Αναζήτησης

Το Sidebar component εξυπηρετεί την πλοήγηση του χρήστη ανάμεσα στις διάφορες σελίδες. Αποτελείται από μία λίστα από τις πιθανές διαδρομές της εφαρμογής. Η λίστα είναι κρυμμένη από προεπιλογή, που σημαίνει ότι είναι εμφανή μόνο τα εικονίδια της κάθε διαδρομής, χωρίς δηλαδή να φαίνεται η ονομασία τους. Σε μικρότερες οθόνες είναι τελείως κρυμμένη έτσι ώστε να αξιοποιηθεί κατάλληλα ο χώρος.



Εικόνα 4.7 Sidebar Component

Το component **DrawerContextProvider** επιτρέπει την επικοινωνία ανάμεσα στο Navbar και το Sidebar, αφού το κουμπί για το άνοιγμα της λίστας βρίσκεται στο Navbar.

Όλες οι διαδρομές της εφαρμογής είναι αποθηκευμένες ως ένας πίνακας αντικειμένων στο αρχείο **routes.js**. Κάθε τέτοιο αντικείμενο έχει τρεις ιδιότητες: 1) την ονομασία της διαδρομής, 2) το μονοπάτι της διαδρομής στο οποίο θα κατευθυνθεί ο χρήστης, 3) το εικονίδιο που εμφανίζεται δίπλα στην ονομασία.

Ένα από τα πιο συχνά μοτίβα που συναντάει κάποιος σχεδόν σε όλες τις React εφαρμογές είναι το mapping τέτοιων πινάκων αντικειμένων σε HTML στοιχεία (ή React component) με την χρήση της JavaScript μεθόδου map. Με τον τρόπο αυτό αποφεύγεται η επανάληψη κώδικα αφού αρκεί αυτός να γραφτεί μία φορά, αλλάζοντας δυναμικά μόνο τα μεταβαλλόμενα στοιχεία του κάθε αντικειμένου.

```
<List>
  {routes.map(route =>
    <Link key={route.name} href={route.path}>
      <a className="linksNoDecoration">
        <ListItem
          className={classes.item}
          classes={{ selected: classes.selected }}
          selected={
            route.path === "/" ? router.pathname === "/" :
              router.pathname.startsWith(route.path)
          }
          button
        >
          <ListItemIcon>{route.icon}</ListItemIcon>
          <ListItemText primary={route.name} />
        </ListItem>
      </a>
    </Link>
  )}
</List>
```

Εικόνα 4.8 Παράδειγμα mapping πίνακα αντικειμένων σε React components

Από το παραπάνω παράδειγμα, γίνεται εύκολα κατανοητό πώς με την χρήση της μεθόδου map μπορούμε να δημιουργήσουμε μία λίστα από επαναλαμβανόμενα στοιχεία της διεπαφής. Για κάθε διαδρομή δημιουργείται δυναμικά ένα link που οδηγεί στην αντίστοιχη οθόνη. Τα πεδία που αλλάζουν είναι το **href** του link, η ονομασία του και το εικονίδιο.

Κάθε τέτοιο link πρέπει υποχρεωτικά να έχει το key prop, το οποίο πρέπει να είναι και μοναδικό ανάμεσα σε όλα τα στοιχεία. Αυτό το prop χρησιμοποιείται εσωτερικά από την React και την βοηθάει να εντοπίζει ποια στοιχεία, σε μία λίστα που δημιουργήθηκε με map, έχουν αλλάξει, προστεθεί ή αφαιρεθεί.

Το Link είναι ένα component της Next.js που επιτρέπει την πλοήγηση του χρήστη από την μία σελίδα στην άλλη, χωρίς να πραγματοποιείται η φόρτωση όλης της εφαρμογής εκ νέου. Με το κλασσικό <a> tag η πλοήγηση γίνεται φορτώνοντας όλα τα στοιχεία της σελίδας από την αρχή, κάτι που δεν θέλουμε να συμβαίνει σε ένα SPA (Single Page Application). Αντίθετα, χρησιμοποιώντας το Link, αλλάζουν μόνο τα στοιχεία του DOM που διαφέρουν από την προηγούμενη σελίδα, κι έτσι τα κοινά στοιχεία, όπως η μπάρα αναζήτησης, δεν θα φορτωθούν ξανά.

Τα **ListItem**, **ListIcon**, **ListText** είναι components του MaterialUI και χρησιμοποιούνται για να δημιουργήσουν το κάθε στοιχείο της λίστας. Το ListItem παίρνει ένα boolean prop, το selected, που δηλώνει εάν η διαδρομή στην οποία βρισκόμαστε αντιστοιχεί στο στοιχείο της λίστας. Στη συνέχεια αυτό το prop χρησιμοποιείται για να αλλάξει το στυλ του ListItem μέσω των classes, έτσι ώστε ο χρήστης να μπορεί με μια ματιά να καταλάβει σε ποιο μέρος της εφαρμογής βρίσκεται.

4.1.3 HTTP Requests

Για τις κλήσεις προς τα διάφορα API χρησιμοποιείται η δημοφιλής βιβλιοθήκη axios. Το axios διευκολύνει την πραγματοποίηση HTTP και προσθέτει ένα ακόμα επίπεδο προστασίας στην εφαρμογή. Μία από τις λειτουργίες τους είναι η δημιουργία instance για ένα συγκεκριμένο API.

```
const newsAxios = axios.create({
    baseURL: "https://newsapi.org/v2",
    params: {
        apiKey: process.env.NEWS_API
    }
})

export const newsFetcher = async url => {
    try{
        const response = await newsAxios.get(url);
        const data = response.data;
        return data;
    }catch(error){
        console.log(error)
    }
}
```

Εικόνα 4.9 Παράδειγμα Axios Instance

Με την χρήση ενός instance, δημιουργείται στην ουσία ένα αντικείμενο, το οποίο εμπεριέχει κάποιες βασικές πληροφορίες, που θα χρησιμοποιηθούν από όλες τις κλήσεις, αποφεύγοντας την επανάληψη κώδικα. Έτσι, στο παραπάνω παράδειγμα, δημιουργήθηκε ένα instance για το News API στο οποίο προστέθηκε το base URL καθώς και η παράμετρος apiKey. Τα API keys είναι ένας τρόπος για τους ιδιοκτήτες των APIs να ελέγχουν ποιος έχει πρόσβαση σε αυτά, καθώς και να περιορίζουν τον αριθμό των κλήσεων που κάθε χρήστης έχει σε αυτά, για την αποφυγή κατάχρησης.

Το API key είναι αποθηκευμένο σε ένα αρχείο .env, και η πρόσβαση σε αυτό γίνεται μέσω του αντικειμένου process.env. Το αρχείο .env δεν αποθηκεύεται στο GitHub κι έτσι το API key δεν εκτίθεται στο κοινό.

Το newsAxios instance στη συνέχεια χρησιμοποιείται για να πραγματοποιηθεί μία κλήση get περνώντας ως παράμετρο μόνο το endpoint που θέλουμε να χρησιμοποιήσουμε, χωρίς δηλαδή να πρέπει να περνάμε κάθε φορά το base URL και το API key. Από την απάντηση της κλήσης επιστρέφονται τα δεδομένα, τα οποία και χρησιμοποιούνται στην εφαρμογή. Παρόμοια instances έχουν δημιουργηθεί και για άλλα APIs της εφαρμογής, όπως για παράδειγμα το βασικό API από το οποίο αντλούμε τις πληροφορίες των παραστάσεων ή το TMDB (The Movie Database) API, από το οποίο παίρνουμε πληροφορίες, όπως ημερομηνίες γεννήσεως καλλιτεχνών, βιογραφικά σημειώματα, φωτογραφίες.

4.2 Αρχική Σελίδα

Αυτή είναι η σελίδα που οι χρήστες βλέπουν όταν επισκέπτονται την βασική διαδρομή της εφαρμογής. Περιέχει στοιχεία από διάφορες κατηγορίες, όπως άρθρα σχετικά με τον χώρο του θεάτρου, δημοφιλείς ηθοποιούς, νεότερες παραστάσεις, έτσι ώστε να παροτρύνει τον επισκέπτη να εξερευνήσει την εφαρμογή.

Η σελίδα είναι στατική, που σημαίνει ότι χρησιμοποιείται η μέθοδος της Next.js getStaticProps() για την ανάκτηση των δεδομένων. Στην συνέχεια τα δεδομένα αυτά χρησιμοποιούνται για να δημιουργήσουν ολόκληρη την HTML σελίδα, κατά το χτίσιμο της εφαρμογής, η οποία και αποθηκεύεται σε κάποιο CDN, έτοιμη να σταλεί σε κάποιον επισκέπτη.

```
export const getStaticProps = async () => {
    cloudinary.config({
        cloud_name: 'dpxyl5vyr',
        secure: true
    })

    const articlesResponse = await newsFetcher(encodeURI("/everything?q=παράσταση θέατρο&sortBy=publishedAt&pageSize=3"))
    const articles = articlesResponse.articles

    articles.forEach(article => {
        article.urlToImage = cloudinary.url(
            encodeURI(article.urlToImage),
            { type: "fetch", width: 320, fetch_format: "auto", crop: "scale", quality: "auto"
        })
    })

    const artists = await mainFetcher("/people?page=0&size=10")
    let latestShows = await mainFetcher(`/productions/latest?page=0&size=10`)

    latestShows = latestShows?.content?.map(show => ({
        id: show.id,
        title: show.title,
        image: getShowImage(show.mediaURL)
    }))

    return {
        props: { artists, latestShows, articles },
        revalidate: 60 * 15
    }
}
```

Εικόνα 4.10 Μέθοδος getStaticProps της Αρχικής Σελίδας

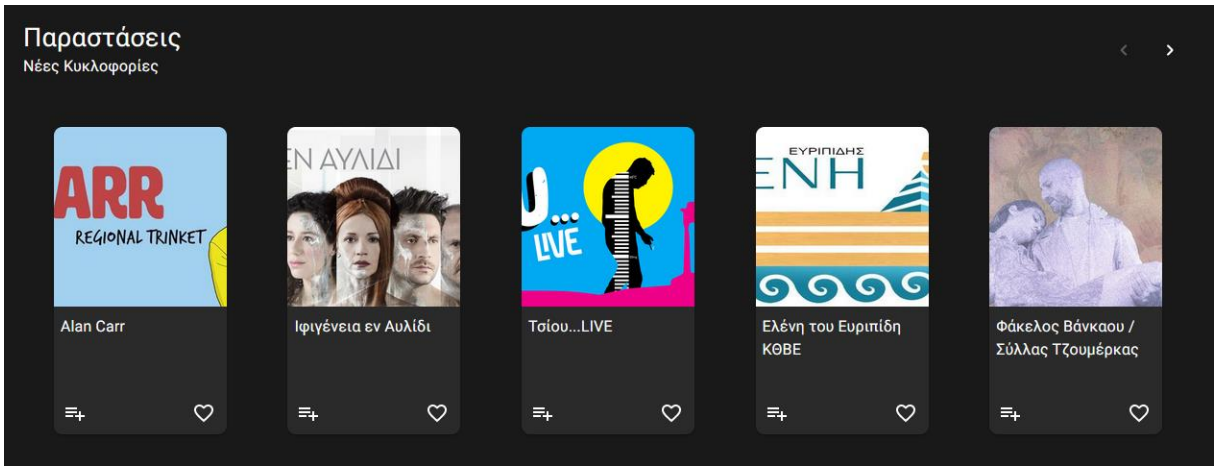
Το γεγονός ότι η σελίδα δημιουργείται κατά το χτίσιμο της εφαρμογής δεν σημαίνει απαραίτητα ότι τα δεδομένα της δεν μπορούν να ανανεωθούν. Περνώντας την ιδιότητα revalidate, ορίζουμε τη χρονική συχνότητα την οποία θέλουμε να δημιουργείται η σελίδα εκ νέου με τα ανανεωμένα δεδομένα. Στο παραπάνω παράδειγμα ορίσαμε τη συχνότητα αυτή στα εννιακόσια δευτερόλεπτα, που σημαίνει ότι η σελίδα θα ανανεωθεί το πολύ κάθε ένα τέταρτο, και μόνο αν κάποιος χρήστης την επισκεφθεί.

Το News API χρησιμοποιείται για την ανεύρεση άρθρων και ειδήσεων σχετικά με τον χώρο του θέατρου. Η αναζήτηση αυτή μπορεί να περιοριστεί με διάφορα φίλτρα, όπως την χώρα με την οποία σχετίζεται το άρθρο, την γλώσσα, την ημερομηνία δημοσίευσής του και την πηγή του. Στην προκειμένη περίπτωση, πραγματοποιήσαμε μία αναζήτηση για τους όρους «παράσταση» και «θέατρο», περιορίσαμε τον αριθμό τον αποτελεσμάτων με την παράμετρο «pageSize» στα τρία, ενώ παράλληλα τα ταξινομήσαμε έτσι ώστε να παίρνουμε πάντα τα νεότερα άρθρα.

Στη συνέχεια η φωτογραφία κάθε άρθρου μετασχηματίζεται, με την χρήση του cloudinary API, έτσι ώστε αυτές να πιάνουν λιγότερο χώρο και η σελίδα να φορτώνει ταχύτερα. Ο κύριος λόγος χρήσης του cloudinary όμως, είναι το γεγονός ότι επιστρέφει την εικόνα με ένα νέο URL, του οποίου γνωρίζουμε το domain name εκ των προτέρων.

Για την χρήση του Image component, πρέπει να προσθέσουμε όλα τα domain names των πηγών των φωτογραφιών στο αρχείο next.config.js για λόγους ασφάλειας, κάτι που θα ήταν απίθανο για τα άρθρα που ανακτάμε μέσω του News API, αφού αυτά μπορεί να προέρχονται από οποιαδήποτε ιστοσελίδα. Με την χρήση του cloudinary, όλα τα URL ξεκινούν από «res.cloudinary.com», το οποίο μπορούμε και να προσθέσουμε στην λίστα με τα επιτρεπόμενα domains.

Εκτός από τα άρθρα, κάνουμε ανάκτηση και δέκα ηθοποιών και παραστάσεων, από το βασικό API της εφαρμογής, και στη συνέχεια όλα αυτά τα δεδομένα περνιούνται ως props στην σελίδα και χρησιμοποιούνται για την δημιουργία της. Για την απεικόνιση αυτών των δεδομένων έχουν δημιουργηθεί επαναχρησιμοποιούμενα components για την κάθε κατηγορία, τα οποία δέχονται ως props τις πληροφορίες, στην περίπτωση των ηθοποιών για παράδειγμα το όνομα, την φωτογραφία και το ID τους. Πατώντας πάνω σε έναν ηθοποιό, ο χρήστης πλοηγείται στην σελίδα με τις λεπτομέρειές του.



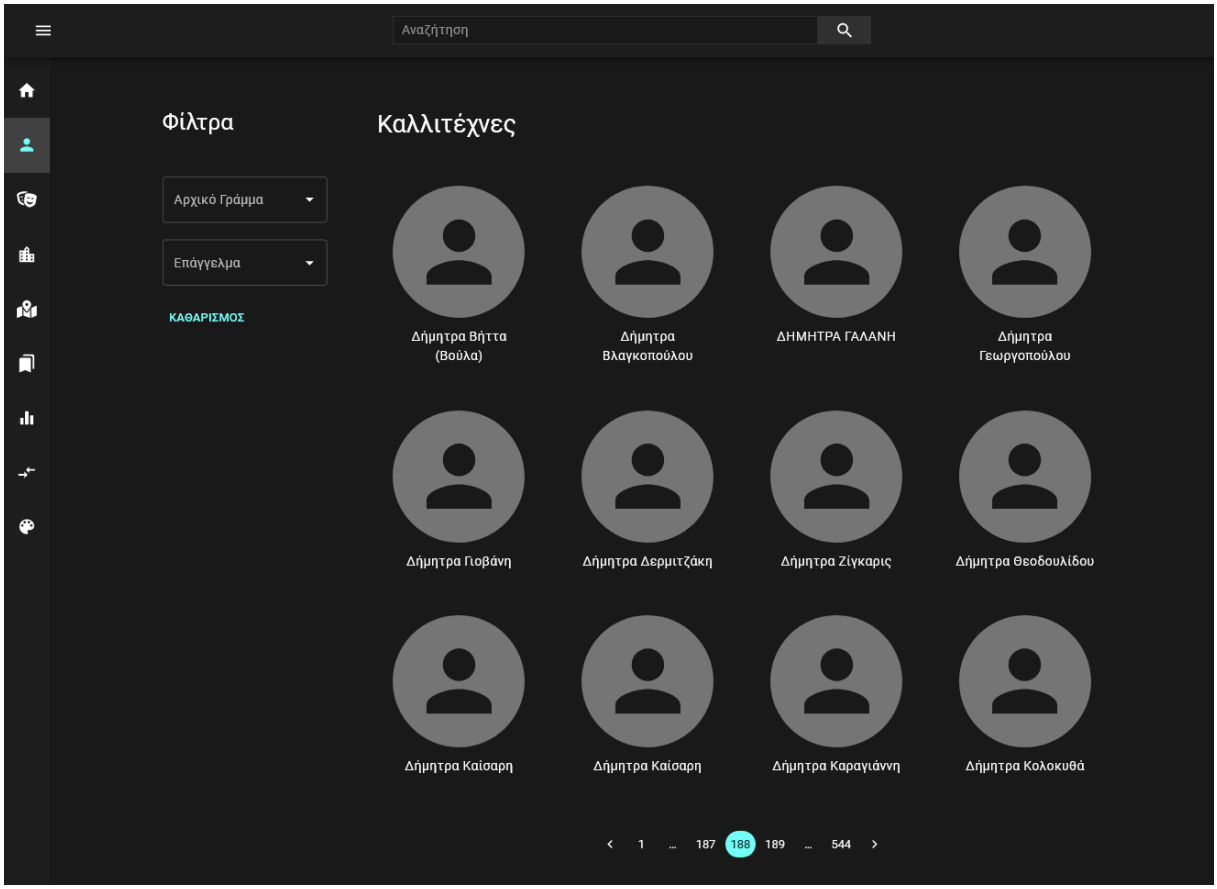
Εικόνα 4.11 Προβολή λίστας θεατρικών παραστάσεων με την χρήση component ContentSlider

Το ContentSlider είναι ένα custom component το οποίο χρησιμοποιεί την βιβλιοθήκη swiper.js και προβάλει μία λίστα των στοιχείων που δέχεται στο εσωτερικό του, μέσω του children prop, ενώ παράλληλα επιτρέπει την οριζόντια κύλισή τους. Με την χρήση του swiper.js είναι δυνατή η δημιουργία του component με ελάχιστο κώδικα, χωρίς όμως να περιορίζει τον βαθμό στον οποίο μπορούμε να το προσαρμόσουμε, ώστε να καλύπτει τις ανάγκες της εκάστοτε εφαρμογής.

Ένα από τα σημαντικότερα πλεονεκτήματα της βιβλιοθήκης, είναι το γεγονός ότι το component είναι mobile friendly, αφού υποστηρίζει την κύλιση των στοιχείων με την χρήση touch screen. Η κύλιση των στοιχείων μπορεί επίσης να επιτευχθεί με την χρήση κουμπιών, έτσι ώστε να είναι δυνατή η πλοήγηση με την χρήση του πληκτρολογίου για την επίτευξη καλύτερης προσβασιμότητας της εφαρμογής.

4.3 Σελίδες Pagination Καλλιτεχνών, Παραστάσεων και Θεάτρων

Οι σελίδες αυτές χρησιμοποιούνται για την προβολή όλων των καλλιτεχνών, παραστάσεων και θεάτρων που υπάρχουν στην βάση δεδομένων της εφαρμογής. Στο page component γίνεται η ανάκτηση των δεδομένων ανάλογα με τον αριθμό της σελίδας και τα φίλτρα που έχουν εφαρμοστεί και στην συνέχεια αυτά χρησιμοποιούνται για την προβολή της λίστας. Οι τρεις αυτές σελίδες υλοποιούνται με παρόμοιο τρόπο με ελάχιστες διαφοροποιήσεις.



Εικόνα 4.12 Σελίδα Pagination Καλλιτεχνών

Το pagination είναι μία τεχνική που χρησιμοποιείται για την διάσπαση του συνόλου των δεδομένων σε επιμέρους τμήματα. Με τον τρόπο αυτό επιτυγχάνεται εξοικονόμηση των δεδομένων που ανταλλάσσονται μέσω του διαδικτύου, το οποίο έχει ως αποτέλεσμα και την ταχύτερη φόρτωση της εφαρμογής. Εάν επιχειρούσαμε να προβάλουμε το σύνολο των δεδομένων αυτό θα είχε ως αποτέλεσμα την κακή απόδοση της εφαρμογής λόγω του μεγάλου πλήθους στοιχείων στην οθόνη, ενώ παράλληλα θα δυσχέραινε την εμπειρία του χρήστη αφού δεν είναι πρακτική η περιήγηση σε μία τέτοια σελίδα.

Η ανάκτηση των δεδομένων με pagination γίνεται περνώντας τις παραμέτρους page και size, η οποίες εκτός από τον αριθμό της σελίδας που θέλουμε να ανακτήσουμε, μας επιτρέπουν να παραμετροποιήσουμε και το μέγεθος της κάθε σελίδας, δηλαδή των αριθμό των στοιχείων που ανακτάμε με κάθε κλήση. Για παράδειγμα, για την ανάκτηση των πρώτων δέκα καλλιτεχνών θα κάναμε μία κλήση στο endpoint “/people?page=0&size=10”.

Από το page component περνιούνται ως props στο PaginationPage component τα δεδομένα, ο αριθμός της τρέχουσας σελίδας, ο συνολικός αριθμός σελίδων καθώς και η διαδρομή στην οποία βρισκόμαστε. Το PaginationPage component μπορεί να επαναχρησιμοποιηθεί ανεξάρτητα από την κατηγορία των δεδομένων που θέλουμε να προβάλλουμε, έτσι είναι απαραίτητο να γνωρίζει την διαδρομή στην οποία βρισκόμαστε ώστε να καθοριστεί ποιο component θα χρησιμοποιηθεί για την απεικόνιση των δεδομένων. Για παράδειγμα, για τους καλλιτέχνες θα χρησιμοποιηθεί το ArtistCard component, ενώ για τα θέατρα το VenueCard.

Για την εναλλαγή των σελίδων και την προβολή της τρέχουσας, αλλά και του συνόλου των σελίδων χρησιμοποιείται το Pagination component του MaterialUI. Εκτός από τα παραπάνω δεδομένα, το component δέχεται ένα function μέσω του onChange prop το οποίο εκτελείται κάθε φορά που πατιέται κάποιο κουμπί και δέχεται ως παράμετρο τον αριθμό της σελίδας στην οποία επιθυμείται να γίνει μετάβαση.

```
const handleChange = (_event, value) => {
    router.push({
        pathname: path,
        query: {
            ...router.query,
            page: value
        }
    })
}
```

Εικόνα 4.13 Χρήση Next Router για αλλαγή διαδρομής

Με την χρήση του router της Next.js, μπορεί να γίνει η μετάβαση σε μια διαδρομή της εφαρμογής με προγραμματιστικό τρόπο, δηλαδή χωρίς την χρήση υπερσυνδέσμων. Χρησιμοποιώντας την μέθοδο push, περνάμε ως παράμετρο ένα αντικείμενο με τις διάφορες επιλογές της διαδρομής στην οποία θέλουμε να μεταβούμε. Στην προκειμένη περίπτωση, το μόνο που θέλουμε να αλλαχτεί είναι η παράμετρος page του query, αφήνοντας τις υπόλοιπες παραμέτρους ως έχουν. Με την χρήση του spread operator (…router.query), θέτουμε τις ιδιότητες του νέου query object ίδιες με το προ υπάρχον και στην συνέχεια αντικαθίσταται η ιδιότητα page με την νέα τιμή.

4.4 Σελίδα Λεπτομερειών Καλλιτέχνη

Σε αυτήν την σελίδα παρουσιάζονται η λεπτομέρειες ενός συγκεκριμένου καλλιτέχνη, όπως για παράδειγμα ένα βιογραφικό σημείωμα, φωτογραφίες, ημερομηνία γέννησης και οι παραστάσεις στις οποίες έχει υπάρξει συντελεστής.

Η ανάκτηση των δεδομένων γίνεται μέσα στην μέθοδο getStaticProps, η οποία τρέχει κατά το χτίσιμο της εφαρμογής και δημιουργεί την σελίδα. Για τις φωτογραφίες και τις διάφορες πληροφορίες του χρήστη χρησιμοποιήθηκε το The Movie Database API.

Η δημιουργία μίας HTML σελίδας για κάθε καλλιτέχνη θα έκανε την διαδικασία του χτισίματος της εφαρμογής ιδιαίτερα χρονοβόρα. Για αυτόν τον λόγο επιλέγεται ένα υποσύνολο των καλλιτεχνών, για τους οποίους θα δημιουργηθεί η σελίδα αρχικά. Οι υπόλοιπες σελίδες θα χτίζονται την στιγμή που κάποιος χρήστης τις επισκέπτεται και στην συνέχεια θα αποθηκεύονται σε κάποιο CDN.

```
export const getStaticPaths = async () => {
  const artists = await mainFetcher('/productions?page=0&size=10');

  const paths = artists.content.map(artist => ({
    params: { id: artist.id.toString() }
  }))

  return {
    paths,
    fallback: true
  }
}
```

Εικόνα 4.14 Παράδειγμα Μεθόδου getStaticPaths

Για την επιλογή αυτού του υποσυνόλου των διαδρομών, οι οποίες θα δημιουργηθούν κατά το χτίσιμο, χρησιμοποιείται η μέθοδος getStaticPaths. Η μέθοδος αυτή εκτελείται στον διακομιστή και επιστρέφει ένα αντικείμενο το οποίο περιέχει τα ids όλων των μονοπατιών που θα δημιουργηθούν, καθώς και την ιδιότητα fallback με τιμή true.

Με την ιδιότητα αυτήν επισημαίνουμε ότι εάν κάποιος χρήστης επισκεφθεί μία σελίδα με id που δεν υπάρχει στα μονοπάτια που έχουμε ορίσει, τότε θα προβληθεί κάποιο fallback component, για παράδειγμα ένα loading screen, όσην ώρα δημιουργείται η σελίδα στο παρασκήνιο.

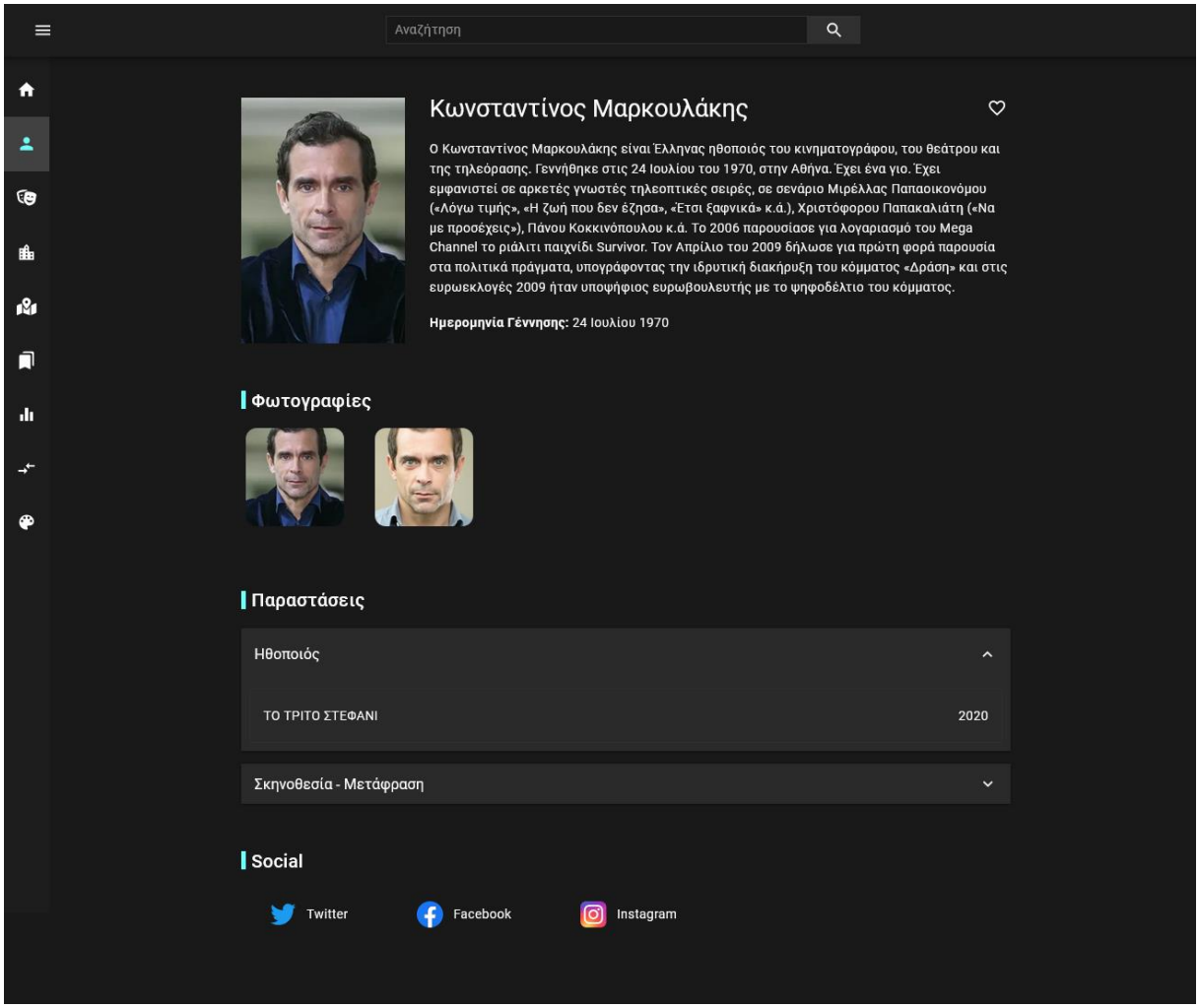
Για κάθε καλλιτέχνη υπάρχει η δυνατότητα προσθήκης του στην λίστα αγαπημένων ενός χρήστη. Η λειτουργία αυτή επιτυγχάνεται μέσω ενός custom hook, το useFavoriteArtist.

```
export default function useFavoriteArtist(id){
  const [isFavorite, setIsFavorite] = useState(false);

  useEffect(() => {
    if (localStorage.getItem("favoriteArtists") === null){
      localStorage.setItem("favoriteArtists", JSON.stringify([]));
    }else{
      const favoriteArtists = JSON.parse(localStorage.favoriteArtists);
      if (favoriteArtists.includes(id)){
        setIsFavorite(true);
      }else{
        setIsFavorite(false);
      }
    }
  }, [id])

  useEffect(() => {
    let favoriteArtists = JSON.parse(localStorage.favoriteArtists);
    if (isFavorite){
      favoriteArtists.push(id);
    }else{
      favoriteArtists = favoriteArtists.filter(item => item !== id);
    }
    localStorage.favoriteArtists = JSON.stringify(favoriteArtists);
  }, [isFavorite, id])

  return { isFavorite, setIsFavorite }
}
```

Εικόνα 4.15 Custom Hook useFavoriteArtist

Το hook αυτό δέχεται ως παράμετρο το ID του καλλιτέχνη και περιέχει δυο useEffect hooks. Το πρώτο από αυτά εκτελείται την πρώτη φορά που καλείται το hook και διαβάζει από το local storage την εγγραφή με κλειδί “favoriteArtists”. Εάν η εγγραφή δεν υπάρχει τότε αρχικοποιείται ως κενός πίνακας, σε αντίθετη περίπτωση διαπιστώνεται εάν το συγκεκριμένο ID εμπεριέχεται στον πίνακα των αγαπημένων καλλιτεχνών, έτσι ώστε να γίνει η κατάλληλη αρχικοποίηση του state isFavorite.

Το δεύτερο useEffect χρησιμοποιείται για την ενημέρωση του local storage όσο ο χρήστης αλληλοεπιδρά με την εφαρμογή και εκτελείται κάθε φορά που ένας καλλιτέχνης προστίθεται ή αφαιρείται από τα αγαπημένα.



Εικόνα 4.16 Οθόνη Λεπτομερειών Καλλιτέχνη

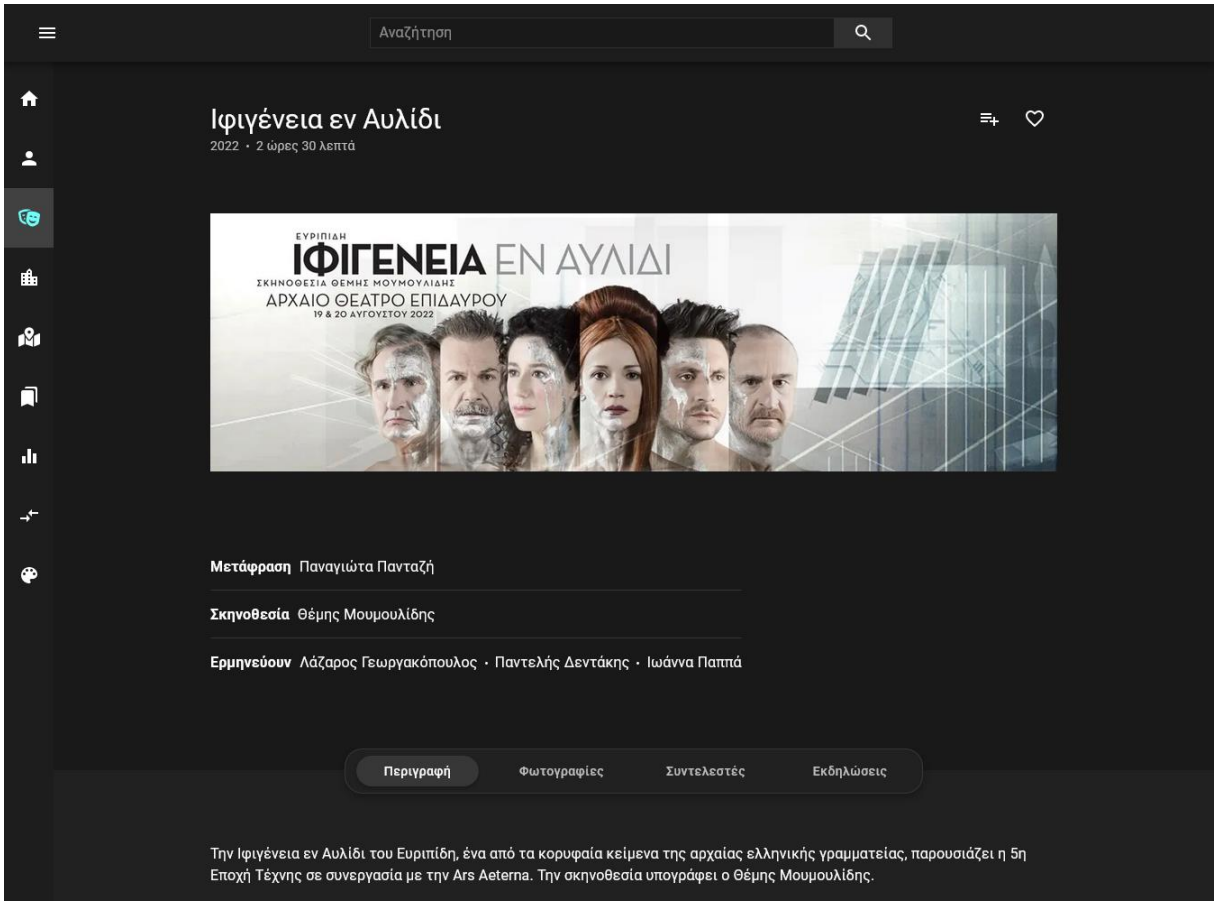
Στην οθόνη αυτήν υπάρχει μία λίστα από φωτογραφίες του καλλιτέχνη, τις οποίες ο χρήστης μπορεί να προβάλει μεγεθυμένες κάνοντας κλικ σε αυτές. Η προβολή αυτή γίνεται μέσω του component **MediaViewer**, το οποίο δέχεται ως props έναν πίνακα με τους υπερσυνδέσμους των φωτογραφιών, καθώς και το index της τρέχουσας φωτογραφίας.

Ο χρήστης μπορεί επίσης να εξερευνήσει όλες τις θεατρικές παραστάσεις στις οποίες έχει λάβει μέρος ο καλλιτέχνης. Οι παραστάσεις είναι ομαδοποιημένες ανάλογα με τον ρόλο που είχε ο συγκεκριμένος καλλιτέχνης. Η κάθε κατηγορία μπορεί να επεκταθεί, ώστε να εμφανιστούν τα ονόματα όλων των

παραστάσεων του συγκεκριμένου ρόλου, τα οποία λειτουργούν ως υπερσύνδεσμοι που οδηγούν στην αντίστοιχη σελίδα λεπτομερειών της παράστασης.

4.5 Σελίδα Λεπτομερειών Παράστασης

Στην σελίδα αυτήν ο χρήστης μπορεί να δει όλες τις λεπτομέρειες μίας συγκεκριμένης παράστασης. Στο πάνω μέρος της οθόνης υπάρχει μία γενική εικόνα της παράστασης, που συμπεριλαμβάνει τον τίτλο, μια φωτογραφία, καθώς και κουμπιά για την προσθαφαίρεση της παράστασης από το watchlist και τα αγαπημένα.



Εικόνα 4.17 Οθόνη Λεπτομερειών Παράστασης

Όπως και στην σελίδα λεπτομερειών καλλιτέχνη, η ανάκτηση των δεδομένων γίνεται στον διακομιστή, μέσω της μεθόδου getStaticProps, και στην συνέχεια αυτά περνιούνται ως props στο page component για την δημιουργία της σελίδας.

Οι πληροφορίες είναι χωρισμένες σε υποενότητες, κάθε μια από τις οποίες εμφανίζεται σε ξεχωριστές καρτέλες, στις οποίες ο χρήστης μπορεί να πλοηγηθεί. Οι καρτέλες έχουν δημιουργηθεί χρησιμοποιώντας τα components **Tabs** και **TabPanel** της βιβλιοθήκης MaterialUI. Το **Tabs** component χρησιμοποιείται για την δημιουργία της μπάρας πλοήγησης που χρησιμοποιεί ο χρήστης για να μεταφερθεί από την μία καρτέλα στην άλλη. Το **TabPanel** χρησιμοποιείται ως ο γονέας του περιεχομένου κάθε καρτέλας.

Για να αποφασιστεί ποια καρτέλα προβάλλεται, σε μια συγκεκριμένη χρονική στιγμή, χρησιμοποιείται ένα useState hook, το οποίο αποθηκεύει το index της τρέχουσας ενεργής καρτέλας. Κάθε **TabPanel** δέχεται ως props το index και το value. Το index είναι ο αριθμός της καρτέλας ενώ στο value περνάμε το state του παραπάνω useState hook. Όταν αυτές οι δύο τιμές συμπίπτουν, αυτό σημαίνει ότι η καρτέλα είναι ενεργή κι έτσι προβάλλεται το περιεχόμενό της.

Η πρώτη καρτέλα εμπεριέχει μία γενικές πληροφορίες για την παράστασης σε μορφή κειμένου, όπως αυτό εμφανιζόταν στο viva.gr από όπου έγινε το scrapping των πληροφοριών. Η πληροφορία αυτή είναι αποθηκευμένη στην βάση δεδομένων ως ένα μονοκόμματο string. Για τον λόγο αυτό έγινε η χρήση της μεθόδου split περνώντας ως παράμετρο το string **\n** που συμβολίζει την έναρξη μίας νέας γραμμής.

Η μέθοδος αυτή εντοπίζει μέσα στο αρχικό string την παράμετρο που δέχτηκε και το διαχωρίζει το σε επιμέρους κομμάτια, τα οποία αποθηκεύονται σε έναν πίνακα από strings. Στην συνέχεια χρησιμοποιείται η μέθοδος map για να δημιουργηθεί για κάθε string του πίνακα μία HTML παράγραφος.

Η δεύτερη καρτέλα περιέχει μια λίστα από φωτογραφίες της παράστασης τις οποίες ο χρήστης μπορεί να επισκοπήσει πατώντας πάνω σε αυτές, με την χρήση του **MediaViewer**, όπως είδαμε προηγουμένως. Η τρίτη καρτέλα περιέχει όλους τους συντελεστές που έλαβαν μέρος στην παράσταση, χωρισμένους ανάλογα με τον ρόλο που είχαν στην παράσταση (π.χ. ηθοποιοί, σκηνοθεσία, σενάριο). Πατώντας πάνω σε κάποιο συντελεστή ο χρήστης μεταφέρεται στην σελίδα με τις λεπτομέρειές του.

Η τέταρτη καρτέλα εμφανίζει όλες τις μεμονωμένες εκδηλώσεις μίας παράστασης χωρισμένες σε δύο πίνακες, τις μελλοντικές εκδηλώσεις, καθώς και το ιστορικό των εκδηλώσεων της παράστασης. Οι δύο αυτοί πίνακες γεμίζονται δυναμικά, συγκρίνοντας της ημερομηνία της εκάστοτε εκδήλωσης με την τωρινή ημερομηνία, για να διαπιστωθεί εάν αυτή βρίσκεται στο παρελθόν ή στο μέλλον, και στην συνέχεια τοποθετείται στον κατάλληλο πίνακα.

Κάθε γραμμή του πίνακα αντιπροσωπεύει μία εκδήλωση και εμπεριέχει πληροφορίες, όπως την ημερομηνία και την ώρα της εκδήλωσης, την τιμή του εισιτηρίου και τον θεατρικό χώρο στον οποίο αυτή πήρε μέρος. Κάνοντας κλικ στο όνομα του θεατρικού χώρου, ο χρήστης μπορεί να μεταβεί στην σελίδα με τις πληροφορίες του.

4.6 Σελίδα Λεπτομερειών Θεατρικού Χώρου

Η σελίδα αυτή παρέχει κάποιες πληροφορίες για τον θεατρικό χώρο, καθώς και επιτρέπει τον χρήστη να περιηγηθεί σε όλες τις παραστάσεις που έχουν παιχτεί στο συγκεκριμένο θέατρο. Οι παραστάσεις εμφανίζονται ως μια λίστα και χρησιμοποιείται το **ContentSlider** component για την προβολή τους. Γίνεται επίσης προβολή ενός χάρτη με την τοποθεσία του θεατρικού χώρου, με την βοήθεια δύο API που παρέχει η Google, το Geocoding API και το Maps Embed API.

Το geocoding είναι η διαδικασία μετατροπής μίας διεύθυνσης σε γεωγραφικές συντεταγμένες (όπως γεωγραφικό πλάτος 37.5960, και γεωγραφικό μήκος 23.0792). Το αντίστροφο geocoding είναι η διαδικασία μετατροπής των γεωγραφικών συντεταγμένων σε μια αναγνώσιμη από τον άνθρωπο διεύθυνση. [31]

Στην βάση δεδομένων της εφαρμογής δεν υπάρχει πληροφορία για την διεύθυνση ενός θεατρικού χώρου σε καμία μορφή. Παρ' όλα αυτά, κάνοντας μία κλήση στο Geocoding API, μπορούμε περνώντας σαν παράμετρο address τον τίτλο του θεατρικού χώρου (το όνομα του θεάτρου), να λάβουμε ως απάντηση ένα JSON αρχείο με τις πληροφορίες της τοποθεσίας του, όπως για παράδειγμα την διεύθυνση, τις

συντεταγμένες και το place id του, που μπορεί να χρησιμοποιηθεί σε άλλα APIs της Google για την ταυτοποίηση μίας τοποθεσίας.

Στην συνέχεια, το place id του θεατρικού χώρου χρησιμοποιείται για την ενσωμάτωση ενός διαδραστικού Google χάρτη, με την τοποθεσία του. Η ενσωμάτωση αυτή γίνεται με την χρήση ενός iframe HTML στοιχείου, βάζοντας στην ιδιότητα src τον κατάλληλο υπερσύνδεσμο.

```
{location &&
  <section>
    <Typography className={classes.sectionTitle} variant="h3">Χάρτης</Typography>
    <iframe
      width="100%"
      height="400"
      style={{ border: 0 }}
      loading="lazy"
      allowFullScreen
      src={`https://www.google.com/maps/embed/v1/place?q=place_id:${location.place_id}&key=${process.env.NEXT_PUBLIC_MAPS_EMBED_API}`}
    />
  </section>
}
```

Εικόνα 4.18 Ενσωμάτωση Google χάρτη με χρήση iframe

Ο χάρτης εμφανίζεται μόνο εάν η μεταβλητή location, που περιέχει την απάντηση από την κλήση που πραγματοποιήθηκε προς το geocoding API, έχει κάποια τιμή. Ένα iframe στοιχείο χρησιμοποιείται για την φόρτωση μίας εξωτερικής HTML σελίδας μέσα στην εφαρμογή. Στην συγκεκριμένη περίπτωση φορτώνουμε έναν χάρτη της Google, χρησιμοποιώντας το Embed API ως πηγή του iframe και περνώντας ως παράμετρο το place id του μέρους που θέλουμε να προβάλουμε, καθώς και το API key, που είναι απαραίτητο για την χρήση της υπηρεσίας.

4.7 Σελίδες Αποτελεσμάτων Αναζήτησης και Εύρεσης Παράστασης

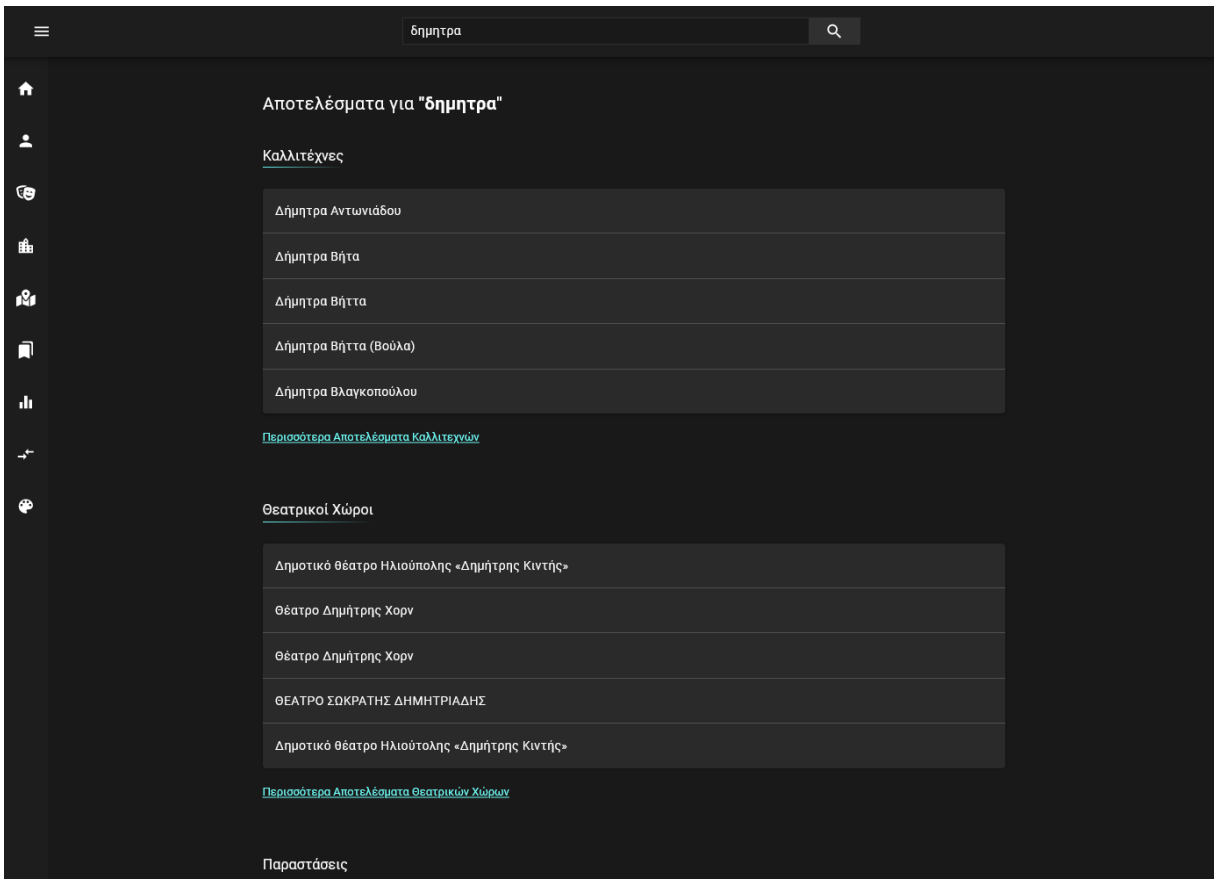
4.7.1 Full Text Αναζήτηση

Η εφαρμογή παρέχει την δυνατότητα πραγματοποίησης full text αναζήτησης, μέσω της μπάρας αναζήτησης που βρίσκεται στην κορυφή κάθε οθόνης. Ο χρήστης μπορεί οποιαδήποτε στιγμή να πληκτρολογήσει έναν όρο αναζήτησης και υποβάλλοντάς τον να μεταφερθεί στην σελίδα των αποτελεσμάτων.

Η πραγματοποίηση της αναζήτησης πραγματοποιείται μέσω του Algolia, το οποίο είναι μία μηχανή αναζήτησης και παρέχει τις υπηρεσίες του μέσω ενός API. Για την χρήση του απαιτείται το άνοιγμα ενός λογαριασμού και στην συνέχεια η φόρτωση των εγγραφών πάνω στις οποίες θέλουμε να πραγματοποιείται η αναζήτηση. Η φόρτωση αυτή μπορεί να γίνει είτε ανεβάζοντας ένα JSON αρχείο, το οποίο περιέχει τα δεδομένα, είτε με δυναμικό τρόπο καλώντας το API της εφαρμογής μέσω της ιστοσελίδας του Algolia, ώστε τα δεδομένα να ενημερώνονται αυτόματα.

Το Algolia παρέχει μέσω της ιστοσελίδας του πολλές ρυθμίσεις και επιλογές για την παραμετροποίηση της λειτουργίας της αναζήτησης. Παρέχει την επιλογή για το ποιες ιδιότητες του JSON αντικειμένου θέλουμε να συμπεριλάβουμε στην αναζήτηση. Για παράδειγμα, στις θεατρικές παραστάσεις, εκτός από τον τίτλο της παράστασης, το Algolia λαμβάνει υπόψιν και το κείμενο που αποτελεί την περιγραφή της παράστασης.

Παρέχονται επίσης δυνατότητες εξατομίκευσης του τρόπου με τον οποίο γίνεται η κατάταξη των αποτελεσμάτων, ενώ παράλληλα είναι ανεκτικό σε τυχόν λάθη που μπορεί να κάνει ο χρήστης κατά την πληκτρολόγηση. Παρέχει επίσης την δυνατότητα ορισμού συνώνυμων λέξεων, έτσι ώστε λέξεις που έχουν παρόμοια σημασία να έχουν και παρόμοια βαρύτητα στις αναζητήσεις.



Εικόνα 4.19 Σελίδα Αποτελεσμάτων Αναζήτησης

Αρχικά, για κάθε κατηγορία αποτελεσμάτων (καλλιτέχνες, θεατρικοί χώροι, παραστάσεις) εμφανίζονται τα κορυφαία πέντε αποτελέσματα σε τρεις διαφορετικούς πίνακες. Στην συνέχεια, εφόσον το πλήθος των αποτελεσμάτων για μία συγκεκριμένη κατηγορία είναι πάνω από πέντε, ο χρήστης έχει την επιλογή να προβάλει όλα τα αποτελέσματα της κατηγορίας, τα οποία εμφανίζονται με την χρήση pagination.

4.7.2 Εύρεση Παραστάσεων Βάση Ημερομηνίας και Τοποθεσίας

Εκτός από full text αναζήτηση, η εφαρμογή επιτρέπει στον χρήστη να αναζητήσει παραστάσεις με βάση την ημερομηνία διεξαγωγής τους, ή ακόμα και με την απόσταση του θεατρικού χώρου από κάποια διεύθυνση που επέλεξε ο χρήστης.

Για την επίτευξη των ανωτέρω λειτουργιών, χρησιμοποιήθηκαν δυο ακόμα API της google, το places autocomplete, το οποίο παρέχει προτάσεις αυτόματης συμπλήρωσης, καθώς ο χρήστης πληκτρολογεί κάποια διεύθυνση, και το distance matrix API, με το οποίο μπορεί να υπολογιστεί η απόστασης μεταξύ δύο, ή περισσότερων τοποθεσιών.

```
<Script
  src={`https://maps.googleapis.com/maps/api/js?key=${process.env.NEXT_PUBLIC_MAPS_JAVASCRIPT_API}&libraries=places`}
  onLoad={() => handleScriptLoad(setAutocompleteService)}
/>

function handleScriptLoad(setService) {
  sessionToken = new google.maps.places.AutocompleteSessionToken();
  const autocompleteService = new google.maps.places.AutocompleteService();
  setService(autocompleteService);
}
```

Εικόνα 4.20 Χρήση Script Component για φόρτωση εξωτερικού script

Η Next.js επιτρέπει την φόρτωση εξωτερικών script με την χρήση του component Script. Το component αυτό τοποθετείται μέσα στο return statement ενός React component, και εκτελείται όταν αυτό γίνεται mount. Η Next ανιχνεύει αυτόματα ποια scripts έχουν φορτωθεί ήδη, ώστε να αποφευχθεί το περιττό κατέβασμα εξωτερικών script.

Το Script component δέχεται το prop src, το οποίο είναι το URL από το οποίο φορτώνεται το script. Δέχεται επίσης το prop onLoad, με το οποίο μπορούμε να εκτελέσουμε κάποιο κομμάτι κώδικα όταν το script εκτελεστεί επιτυχώς. Στην προκειμένη περίπτωση όταν εκτελεστεί το script δημιουργούμε ένα session token, το οποίο απαιτείται για την χρήση του API, καθώς και ένα instance του autocomplete service, που περιέχει τις μεθόδους που χρησιμοποιούνται για το autocomplete. Στη συνέχεια, το instance αυτό αποθηκεύεται σε ένα React state, για την πιο εύκολη πρόσβασή του μέσα από το component.

```
useEffect(() => {
  const debounce = setTimeout(() => {
    if (autocompleteService && (typeof sessionToken !== "undefined")) {
      autocompleteService.getPlacePredictions({
        input: formData.address,
        sessionToken: sessionToken,
        componentRestrictions: {
          country: 'gr'
        }
      },
        (predictions) => { setPredictions(predictions) });
    }
  }, 500)

  return () => {
    clearTimeout(debounce)
  }
}, [autocompleteService, formData.address])
```

Εικόνα 4.21 Χρήση του autocomplete service

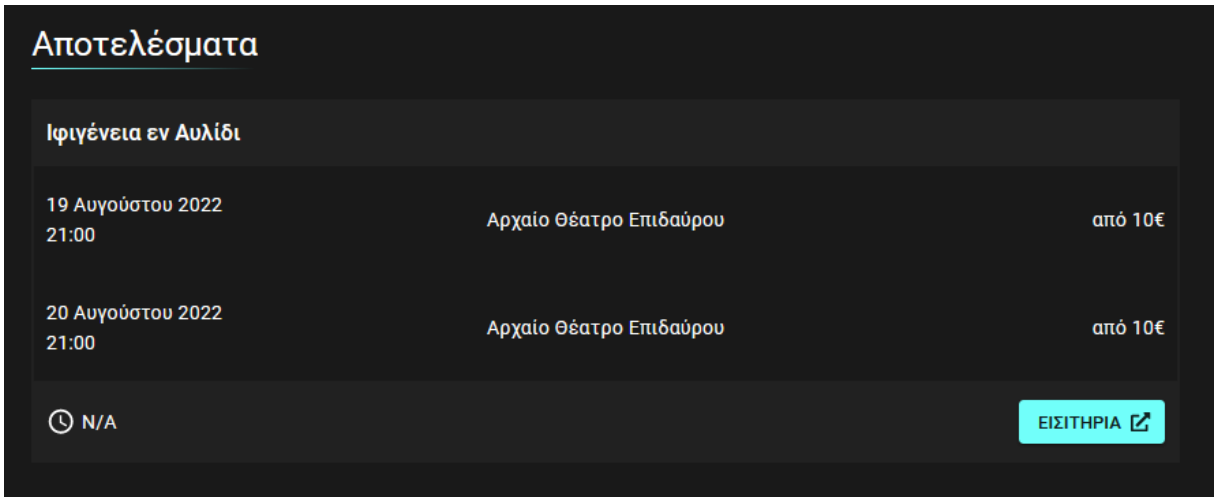
Η χρήση του autocomplete service γίνεται μέσα από μία useEffect, η οποία εκτελείται κάθε φορά που αλλάζει το περιεχόμενο του πεδίου της διεύθυνσης, δηλαδή κάθε φορά που ο χρήστης πληκτρολογεί ένα γράμμα. Μέσα στην useEffect καλείται η μέθοδος getPlacePredictions, στην οποία περνάμε ως παράμετρο την διεύθυνση που έχει εισάγει ο χρήστης, το session token, καθώς και τον περιορισμό να λαμβάνουμε αποτελέσματα μόνο ελληνικών διευθύνσεων. Η μέθοδος αυτή πραγματοποιεί την κλήση

προς το API και επιστρέφει τα αποτελέσματα με τις προτάσεις αυτόματης συμπλήρωσης, τα οποία αποθηκεύονται στο state, και στην συνέχεια προβάλλονται στον χρήστη.

Τα APIs της google κοστολογούνται ανάλογα με τον αριθμό τον κλήσεων προς αυτά. Για την μείωση των κλήσεων προς το autocomplete API χρησιμοποιείται η τεχνική debounce. Το debouncing είναι μία προγραμματιστική τεχνική κατά την οποία καθυστερείται η εκτέλεση μίας μεθόδου κατά ένα συγκεκριμένο χρονικό διάστημα, έτσι ώστε αυτή να μην εκτελείται πολλαπλές φορές σε σύντομο χρονικό διάστημα, βελτιώνοντας έτσι την απόδοση του κώδικα.

Στο παραπάνω παράδειγμα, ο κώδικας που εκτελείται μέσα στην useEffect έχει τοποθετηθεί μέσα σε ένα setTimeout με καθυστέρηση 500 millisecond, ενώ στο return statement της useEffect, καλούμε την μέθοδο clearTimeout η οποία ακυρώνει το timeout που δημιουργήσαμε νωρίτερα. Με τον τρόπο αυτό, η κλήση προς το API πραγματοποιείται με καθυστέρηση μισού δευτερολέπτου, και μόνο αν ο χρήστης δεν πληκτρολογήσει νέο χαρακτήρα στο διάστημα αυτό.

Η διεύθυνση αυτή χρησιμοποιείται, σε συνδυασμό με τις διευθύνσεις των θεατρικών χώρων, ως παράμετροι στην κλήση προς το distance matrix API, με το οποίο γίνεται ο υπολογισμός της απόστασης των δύο σημείων. Στη συνέχεια, ανάλογα με την μέγιστη απόσταση και την ημερομηνία που έχει επιλέξει ο χρήστης, φιλτράρονται οι παραστάσεις και προβάλλονται στον χρήστη.

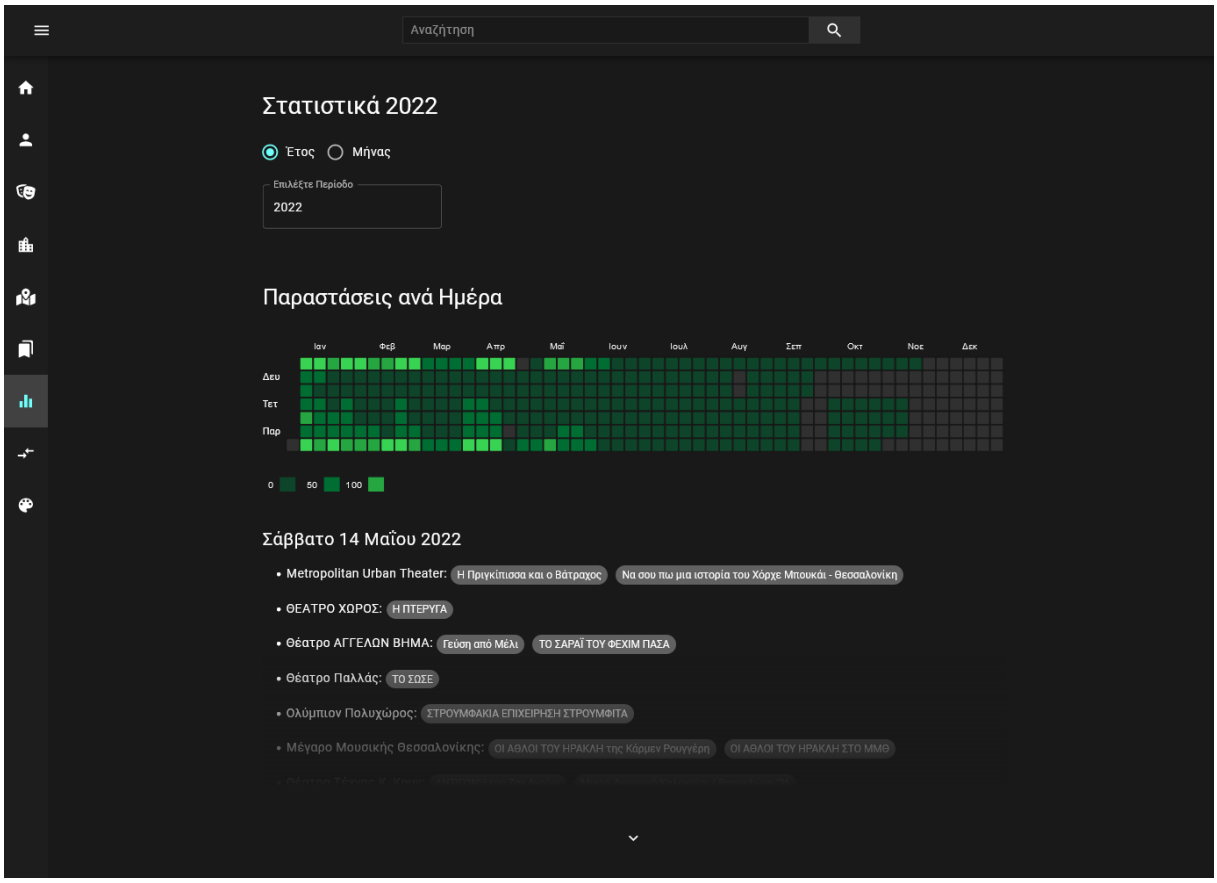


Εικόνα 4.22 Παράδειγμα αποτελέσματος εύρεσης παράστασης

Για κάθε παράσταση, προβάλλεται μία λίστα με τις μεμονωμένες εκδηλώσεις, καθώς και η ημερομηνία και ώρα της κάθε μίας, ο θεατρικός χώρος στον οποίο αυτή θα πραγματοποιηθεί και η τιμή του εισιτηρίου. Ο χρήστης μπορεί επίσης να μεταφερθεί στην ιστοσελίδα viva.gr, μέσω της οποίας μπορεί να πραγματοποιήσει την αγορά εισιτηρίου για κάποια παράσταση.

4.8 Σελίδες Στατιστικών

Ο χρήστης μέσω της εφαρμογής έχει την δυνατότητα να ανακαλύψει διάφορα στατιστικά επιλέγοντας μία συγκεκριμένη περίοδο. Για την δημιουργία των γραφημάτων χρησιμοποιήθηκαν οι βιβλιοθήκες rechart και nivo, οι οποίες παρέχουν έτοιμα react components, τα οποία δέχονται δεδομένα και μπορούν να παραμετροποιηθούν μέσω των props, για την διευκόλυνση της οπτικοποίησης των δεδομένων.



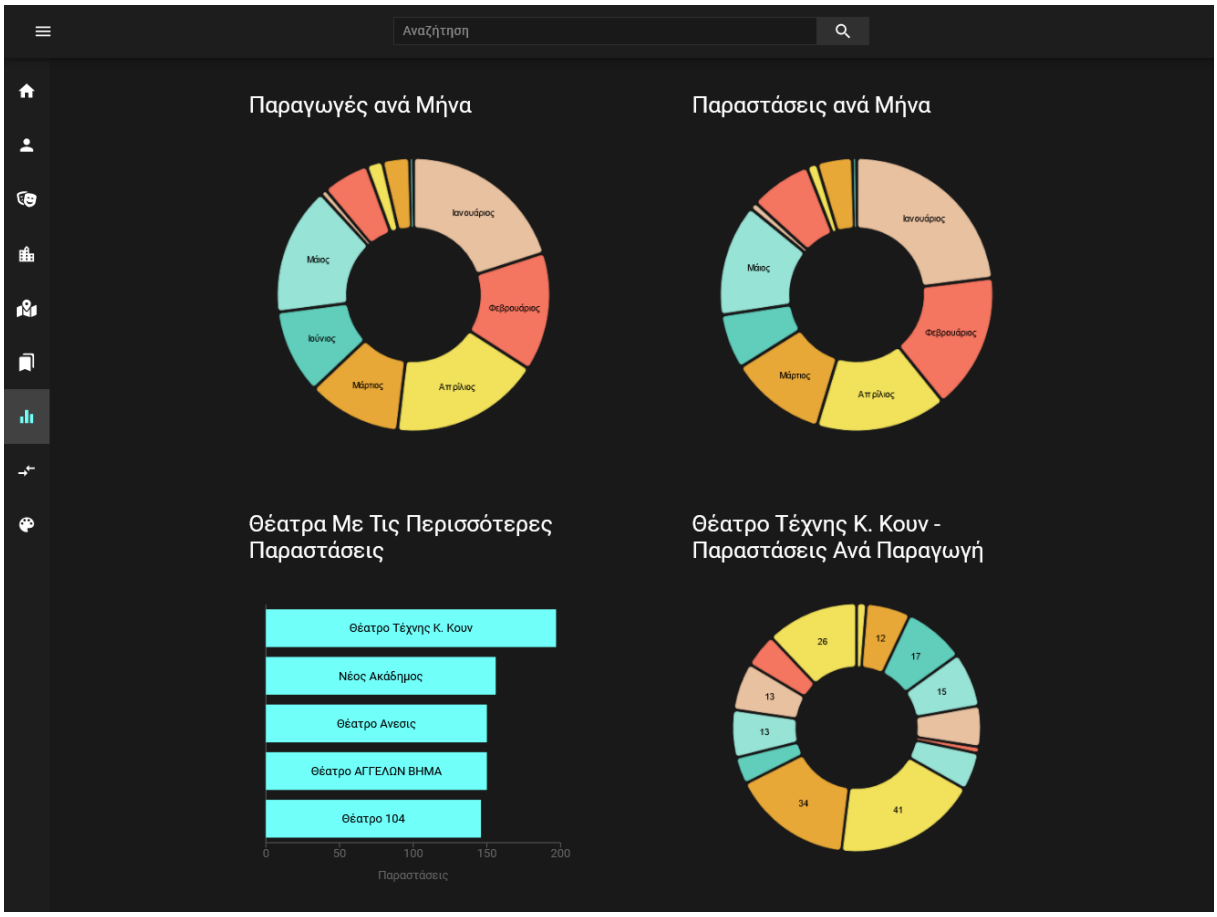
Εικόνα 4.23 Σελίδα Στατιστικών

Ο χρήστης μπορεί να επιλέξει μία περίοδο ενός έτους ή ενός μήνα. Στην κορυφή της σελίδας υπάρχει ένα γράφημα σε στυλ ημερολογίου, το οποίο δείχνει τον αριθμό τον παραστάσεων για κάθε ημέρα του χρόνου. Όσο πιο φωτεινό είναι ένα κουτάκι, τόσες περισσότερες παραστάσεις έλαβαν μέρος την συγκεκριμένη ημέρα. Ο χρήστης μπορεί επίσης, τοποθετώντας τον κέρσορα πάνω από ένα κουτάκι, να δει ολόκληρη την ημερομηνία, καθώς και τον ακριβή αριθμό των παραστάσεων.

Κάνοντας κλικ σε ένα από τα κουτάκια, ανακτώνται όλες οι παραστάσεις που παίχτηκαν την συγκεκριμένη ημέρα και εμφανίζονται στην λίστα κάτω από το ημερολόγιο. Κάθε στοιχείο της λίστας αποτελείται από τον θεατρικό χώρο και τις παραστάσεις που παίχτηκαν αυτόν, τα οποία είναι υπερσύνδεσμοι που οδηγούν στην αντίστοιχη σελίδα λεπτομερειών.

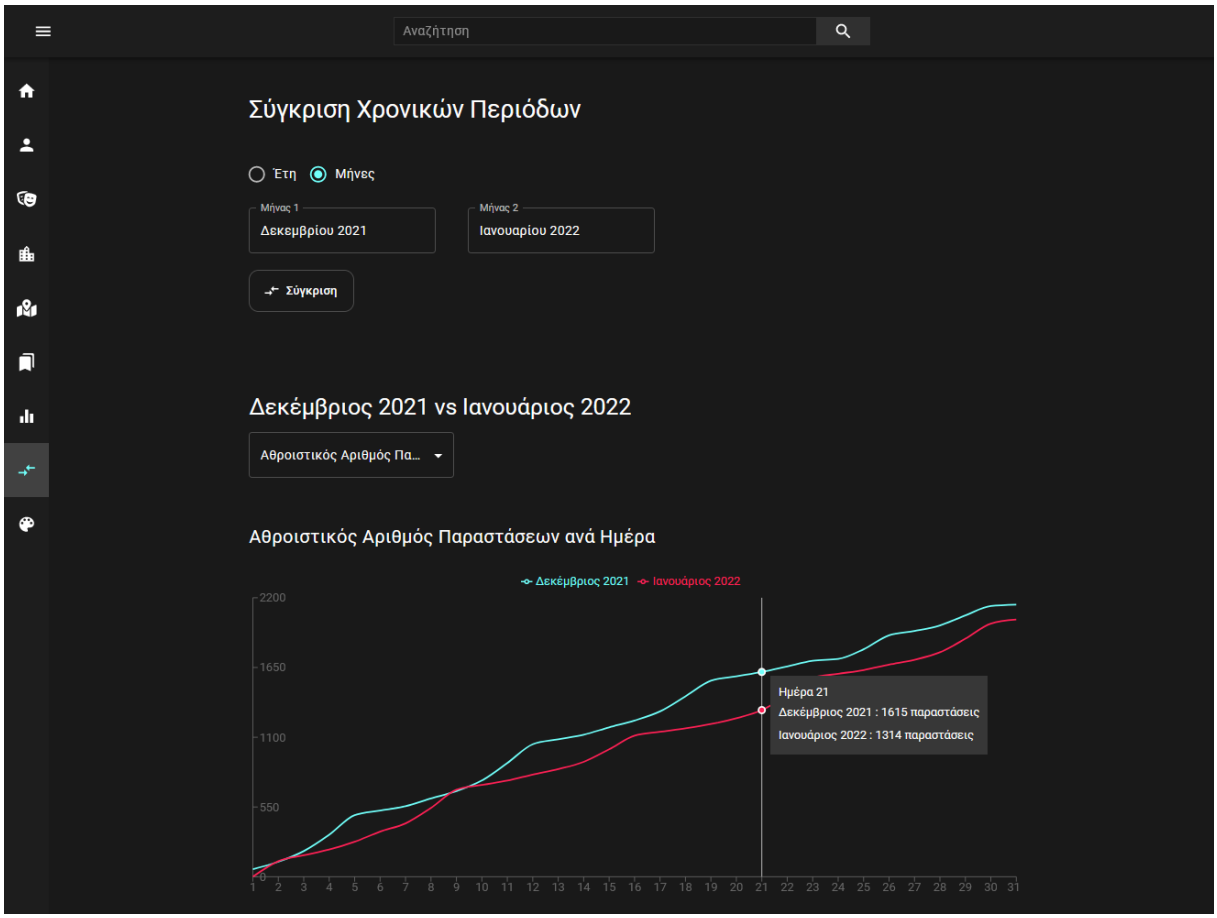
Εκτός από το γράφημα σε μορφή ημερολογίου, στην σελίδα υπάρχουν και γραφήματα πίτας και ράβδων, για διάφορα στατιστικά όπως παραγωγές ανά μήνα, παραστάσεις ανά μήνα, παραγωγές με τις περισσότερες παραστάσεις, θέατρα με τις περισσότερες παραστάσεις.

Στο γράφημα που απεικονίζει τα θέατρα με τις περισσότερες παραστάσεις, ο χρήστης έχει την δυνατότητα, πατώντας πάνω σε κάποιο από τα θέατρα να προβάλει ένα νέο γράφημα, το οποίο δείχνει τις παραγωγές με τις περισσότερες παραστάσεις στο συγκεκριμένο θέατρο, την συγκεκριμένη χρονική περίοδο. Ο χρήστης, τοποθετώντας τον κέρσορα πάνω από κάποια ράβδο, μπορεί να δει πιο λεπτομερή στοιχεία για το συγκεκριμένο στατιστικό.



Εικόνα 4.24 Γραφήματα σελίδας στατιστικών

Εκτός από την προβολή των στατιστικών μίας χρονικής περιόδου, παρέχεται η δυνατότητα σύγκρισης αυτών μεταξύ δύο διαφορετικών περιόδων. Ο χρήστης, μέσω της σελίδας σύγκρισης, επιλέγει δύο διαφορετικούς μήνες ή έτη για σύγκριση, και στην συνέχεια επιλέγει το στατιστικό που θέλει να προβάλλει.



Εικόνα 4.25 Σελίδα σύγκρισης στατιστικών

4.9 Επίλογος

Στο κεφάλαιο αυτό έγινε μία αναλυτική περιγραφή της εφαρμογής, ξεκινώντας από την δομή της και στην συνέχεια αναλύθηκαν η μεμονωμένες σελίδες που την απαρτίζουν. Έγινε επίσης μία ανάλυση της χρήσης των APIs που χρησιμοποιήθηκαν για την επίτευξη των λειτουργιών της εφαρμογής, καθώς και η παρουσίαση επιμέρους κομματιών αξιοσημείωτου κώδικα.

Κεφάλαιο 5ο: Προτάσεις Βελτίωσης

Ο κόσμος της ανάπτυξης διαδικτυακών εφαρμογών συνεχώς αλλάζει και εξελίσσεται, καθώς νέες τεχνολογίες και βιβλιοθήκες εμφανίζονται συνεχώς, οι οποίες έχουν ως στόχο να κάνουν το διαδίκτυο πιο γρήγορο και ασφαλές, αλλά συγχρόνως να προσφέρουν και στους προγραμματιστές μια πιο φιλική και εύχρηστη εμπειρία κατά την διαδικασία ανάπτυξης των εφαρμογών.

Μία από τις προτάσεις βελτίωσης της εφαρμογής είναι η χρήση της TypeScript αντί της απλής JavaScript, για την συγγραφή του κώδικα. Η TypeScript είναι μία επέκταση της JavaScript και δίνει την δυνατότητα της δήλωσης του τύπου των μεταβλητών. Με την απλή JavaScript, κάθε μεταβλητή μπορεί να είναι οποιουδήποτε τύπου, γεγονός που την κάνει ευέλικτη, αλλά παράλληλα κάνει τον κώδικα απρόβλεπτο και δυσανάγνωστο, πράγμα που γίνεται ιδιαίτερα αντιληπτό όσο περισσότερο μεγαλώνει η εφαρμογή.

Με την χρήση της TypeScript, είναι ευκολότερο για τους προγραμματιστές να κατανοήσουν την λειτουργία μίας μεθόδου ή ενός κομματιού κώδικα, γεγονός που γίνεται ιδιαίτερα εμφανές όταν η εφαρμογή έχει γραφτεί από πολλαπλούς προγραμματιστές. Η TypeScript ανιχνεύει πιθανά σφάλματα στον κώδικα, όταν υπάρχουν μεταβλητές με λάθος τύπο, πριν αυτός εκτελεστεί, ενώ παράλληλα παρέχει προτάσεις για αυτόματη συμπλήρωση κώδικα, για παράδειγμα όταν γίνεται προσπάθεια πρόσβασης σε κάποια ιδιότητα ενός αντικειμένου.

Η αρχική σχεδίαση και ανάπτυξη μίας εφαρμογής είναι μόνο ένα μικρό κομμάτι του κύκλου ζωής της, αφού οι διαδικτυακές εφαρμογές ενημερώνονται συνεχώς, προσθέτοντας νέες λειτουργίες και επεκτείνοντας τις ήδη υπάρχουσες. Μία λειτουργία που θα βελτίωνε την εμπειρία του χρήστη της εφαρμογής, θα ήταν η δυνατότητα να ανοίξει λογαριασμό κάνοντας εγγραφή στην εφαρμογή. Η εφαρμογή αποθηκεύει τα αγαπημένα και το watchlist του χρήστη στο local storage του περιηγητή, δηλαδή τοπικά στην συσκευή του χρήστη. Προσθέτοντας την δυνατότητα εγγραφής χρηστών, θα ήταν εφικτό να αποθηκεύονται τα δεδομένα αυτά στο cloud, κι έτσι οι χρήστες θα είχαν πρόσβαση σε αυτά από οποιαδήποτε συσκευή συνδεθούν.

Καθώς η πολυπλοκότητα της εφαρμογής αυξάνεται και αναπτύσσονται νέες λειτουργίες, οι οποίες μπορεί να αλληλοεπιδρούν με πολλά κομμάτια της εφαρμογής, θα ήταν μία καλή ιδέα η χρήση μίας βιβλιοθήκης που θα βοηθούσε στο state management, όπως για παράδειγμα η Redux. Η Redux είναι μία βιβλιοθήκη που εξυπηρετεί στην διαχείριση του state μιας εφαρμογής, παρέχοντας την δυνατότητα στα components να μοιράζονται πληροφορίες με ευκολία.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Internet Sites

[1] Web application [Online]. Διαθέσιμο: https://en.wikipedia.org/wiki/Web_application

[2] Frontend vs Backend [Online]. Διαθέσιμο: https://academind.com/tutorials/frontend-vs-backend

[3] Seven Reasons Why a Website's Front-End And Back-End Should Be Kept Separate [Online]. Διαθέσιμο: www.forbes.com/sites/forbestechcouncil/2018/07/19/seven-reasons-why-a-websites-front-end-and-back-end-should-be-kept-separate

[4] HTML element structure [Online]. Διαθέσιμο: https://commons.wikimedia.org/wiki/File:HTML_element_structure.svg

[5] HTML basics [Online]. Διαθέσιμο: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics

[6] The Science of Semantic HTML [Online]. Διαθέσιμο: https://medium.com/geekculture/the-science-of-semantic-html-c66fda24f105

[7] Pseudo-elements [Online]. Διαθέσιμο: https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-elements

[8] Desktop vs Mobile vs Tablet Market Share Worldwide [Online]. Διαθέσιμο: https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/worldwide/#monthly-201208-202203

[9] JavaScript [Online]. Διαθέσιμο: https://en.wikipedia.org/wiki/JavaScript

[10] Brendan Eich [Online]. Διαθέσιμο: https://en.wikipedia.org/wiki/Brendan_Eich

[11] JavaScript HTML DOM [Online]. Διαθέσιμο: https://www.w3schools.com/js/js_htmldom.asp

[12] Introducing JSON [Online]. Διαθέσιμο: https://www.json.org/json-en.html

[13] Introducing JSX [Online]. Διαθέσιμο: https://reactjs.org/docs/introducing-jsx.html

[14] Writing Markup with JSX [Online]. Διαθέσιμο: https://beta.reactjs.org/learn/writing-markup-with-jsx#the-rules-of-jsx

[15] Top 10 Websites Built on React.js [Online]. Διαθέσιμο: https://www.clariontech.com/blog/top-10-websites-built-on-react.js

[16] Next.js [Online]. Διαθέσιμο: https://en.wikipedia.org/wiki/Next.js

[17] Next.js Pages [Online]. Διαθέσιμο: https://nextjs.org/docs/basic-features/pages

[18] How Next.js Works [Online]. Διαθέσιμο: https://nextjs.org/learn/foundations/how-nextjs-works/cdns-and-edge

[19] Image Component and Image Optimization [Online]. Διαθέσιμο: https://nextjs.org/docs/basic-features/image-optimization

[20] What is Jamstack? [Online]. Διαθέσιμο: https://jamstack.org/what-is-jamstack/

[21] New to Jamstack? Everything You Need to Know to Get Started [Online]. Διαθέσιμο: https://snipcart.com/blog/jamstack

[22] Why Jamstack? [Online]. Διαθέσιμο: https://jamstack.org/why-jamstack/

[23] Jamming into the Mainstream: Jamstack Community Survey 2021 [Online]. Διαθέσιμο: https://jamstack.org/survey/2021/

[24] A/B testing [Online]. Διαθέσιμο: https://en.wikipedia.org/wiki/A/B_testing

[25] Color Psychology: The Emotional Effects of Colors [Online]. Διαθέσιμο: http://www.arttherapyblog.com/online/color-psychology-psychologica-effects-of-colors

[26] What is Visual Hierarchy? Here's How to Create Attention-Grabbing Content Using these 5 Key Principles. [Online]. Διαθέσιμο: https://cemoh.com/blog/attention-grabbing-content-using-visual-hierarchy/

[27] User Experience Design [Online]. Διαθέσιμο: http://semanticstudios.com/user_experience_design/

[28] ACCESSIBILITY [Online]. Διαθέσιμο: https://www.w3.org/standards/webdesign/accessibility

[29] Responsive Web Design [Online]. Διαθέσιμο: https://alistapart.com/article/responsive-web-design/

[30] Responsive design [Online]. Διαθέσιμο: https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Responsive_Design

[31] Geocoding API [Online]. Διαθέσιμο: https://developers.google.com/maps/documentation/geocoding/overview