

ΙΟΝΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ



Πτυχιακή Εργασία

«Ανάπτυξη εφαρμογής Web Player με τη χρήση APIs
Online υπηρεσιών streaming»

Νταρλαγιάννης Δημήτριος

Επιβλέπων: Μιχαήλ Στεφανιδάκης

Κέρκυρα Σεπτέμβριος 2023



«Ανάπτυξη εφαρμογής Web Player με τη χρήση APIs Online υπηρεσιών streaming»

Δημήτριος Νταρλαγιάννης

Τριμελής Επιτροπή

Μιχαήλ Στεφανιδάκης
Αναπληρωτής Καθηγητής

Τμήμα Πληροφορικής,
Ιόνιο Πανεπιστήμιο

Δημήτριος Ρίγγας
ΕΔΙΠ

Τμήμα Πληροφορικής,
Ιόνιο Πανεπιστήμιο

Χριστόφορος Νταντογιάν
Επίκουρος Καθηγητής

Τμήμα Πληροφορικής,
Ιόνιο Πανεπιστήμιο

Περίληψη

Το αντικείμενο της παρούσας πτυχιακής εργασίας επικεντρώνεται στην ανάπτυξη ενός Web Player για αναπαραγωγή τραγουδιών, χρησιμοποιώντας το API μιας streaming πλατφόρμας. Η εφαρμογή αυτή έχει σκοπό να προσφέρει μια χρήσιμη και ευχάριστη εμπειρία ακρόασης μουσικής στους χρήστες, παρέχοντας πρόσβαση σε μια τεράστια βιβλιοθήκη τραγουδιών. Με τη χρήση του API, πραγματοποιείται αναζήτηση, ανάκτηση και αναπαραγωγή τραγουδιών ανάλογα με τις προτιμήσεις του χρήστη.

Ο χρήστης έχει τη δυνατότητα να αναζητήσει τραγούδια βάσει των καλλιτεχνών, των άλμπουμ ή των ειδών μουσικής που τον ενδιαφέρουν. Επιπλέον, η εφαρμογή παρέχει λειτουργίες όπως ο χρήστης να μπορεί να κάνει “μου αρέσει” σε τραγούδια, παρέχει το ιστορικό των τελευταίων είκοσι τραγουδιών που άκουσε ο χρήστης και έναν δείκτη που δείχνει πόσο πολύ ακούνε οι χρήστες του Spotify κάθε τραγούδι εκείνη την περίοδο. Ακόμα, ο χρήστης μπορεί να αυξομειώσει την ένταση, να επιλέξει από ποια συσκευή να γίνει αναπαραγωγή και να αλλάξει το σημείο του τραγουδιού.

Κατά την ανάπτυξη της εφαρμογής, έχουν ληφθεί υπόψη οι βέλτιστες πρακτικές για τη σχεδίαση και την ανάπτυξη Web εφαρμογών. Η διεπαφή είναι φιλική προς τον χρήστη και προσαρμόζεται σε διάφορες συσκευές, όπως σταθερούς και φορητούς υπολογιστές, smartphones και tablets.

Επίσης γίνεται ανάλυση την αρχιτεκτονικής της εφαρμογής, των τεχνολογιών που χρησιμοποιήθηκαν για την ανάπτυξη (όπως HTML, Tailwind CSS και Next.js) αλλά και των προκλήσεων που προέκυψαν κατά τη διάρκεια της ανάπτυξης.

Συνολικά, η εργασία αποτελεί μια συνεκτική και λεπτομερή μελέτη για την ανάπτυξη μιας Web Player εφαρμογής που εκμεταλλεύεται το API μιας streaming πλατφόρμας για την αναπαραγωγή τραγουδιών.

Abstract

The subject of this thesis focuses on the development of a Web Player for playing songs, using the API of a streaming platform. This application aims to provide a useful and enjoyable music listening experience to users by providing access to a huge library of songs. By using the API, songs are searched, retrieved and played according to the user's preferences.

The user has the ability to search for songs based on the artists, albums or genres of music they are interested in. In addition, the app provides features such as the user being able to “like” songs, providing a history of the last twenty songs that the user has listened to, and an indicator that shows how much Spotify users are listening to each song during that period. Also, the user can increase or decrease the volume, choose from which device to play and change the point of the song.

Best practices for web application design and development have been considered while developing the application. The interface is user-friendly and adapts to various devices, such as desktop and laptop computers, smartphones and tablets. The architecture of the application, the technologies used for development (such as HTML, tailwind CSS and Next.js) and the challenges that arose during development are also analyzed.

Overall, the work is a coherent and detailed study of the development of a web player application that uses the API of a streaming platform to play songs.

Περιεχόμενα

Περίληψη	i
Abstract	ii
Περιεχόμενα	iii
Κατάλογος Εικόνων	iv
Κατάλογος Αποσπασμάτων Κώδικα	v
1. Εισαγωγή	1
1.1. Κίνητρο	1
1.2. Σκοπός	1
1.3. Συνεισφορά Εργασίας	2
1.4. Διάρθρωση Εργασίας	2
2. Περιγραφή Εργαλείων	3
2.1. Spotify	3
2.2. Εργαλεία που Χρησιμοποιήθηκαν	3
2.2.1. HTML	4
2.2.2. CSS	4
2.2.3. JavaScript	5
2.2.4. React	6
2.2.5. Next.js	7
2.2.6. Tailwind CSS	8
2.2.7. Visual Studio Code	8
2.2.8. NextAuth.js	9
2.2.9. Headless UI	10
2.2.10. Recoil	10
2.2.11. Heroicons	11
2.2.12. Node.js	12
2.2.13. NPM.js	13
2.2.14. GitHub	14

2.2.15. Spotify for Developers	15
3. Αρχιτεκτονική	17
3.1. Μοντέλο Ανάπτυξης	17
3.2. Αυθεντικοποίηση και Σύνδεση	19
3.3. Atoms	24
3.4. Components της Εφαρμογής	25
3.4.1. Search	26
3.4.2. Recently Played	27
3.4.3. Player	28
3.4.4. Track	29
3.4.5. Poster	30
3.4.6. Body	31
3.4.7. Dashboard	34
3.4.8. Right	36
4. Deployment	38
4.1. Vercel	38
4.2. Διαδικασία Deployment	39
5. Επίλογος	42
5.1. Συμπεράσματα	42
5.2. Μελλοντικές Επεκτάσεις	42
Βιβλιογραφικές Αναφορές	44

Κατάλογος Εικόνων

2.1	Λογότυπο της HTML	4
2.2	Λογότυπο της CSS	5
2.3	Λογότυπο της JavaScript	6
2.4	Λογότυπο της React	7
2.5	Λογότυπο του Next.js	7
2.6	Λογότυπο του tailwindcss	8
2.7	Λογότυπο του Visual Studio Code	9
2.8	Λογότυπο της NextAuth.js	10
2.9	Λογότυπο της Headless UI	10
2.10	Λογότυπο της Recoil.js	11
2.11	Λογότυπο της Heroicons	12
2.12	Λογότυπο του Node.js	13
2.13	Λογότυπο του NPM.js	14
2.14	Λογότυπο της GitHub	15
2.15	Λογότυπο του Spotify for Developers	16
3.1	Σελίδα σύνδεσης στην εφαρμογή	23
3.2	Σελίδα σύνδεσης στην εφαρμογή	23
3.3	Loader της εφαρμογής	24
3.4	Search component της εφαρμογής	26
3.5	RecentlyPlayed component της εφαρμογής	27
3.6	Player component της εφαρμογής	28
3.7	Track component της εφαρμογής	29
3.8	Poster component της εφαρμογής	30
3.9	Body component της εφαρμογής	31
3.10	Dashboard component της εφαρμογής	34
3.11	Right component της εφαρμογής	36
4.1	Λογότυπο της Vercel	38
4.2	Dashboard της Vercel (1)	39

4.3	Dashboard της Vercel (2)	39
4.4	Dashboard της Vercel (3)	40
4.5	Dashboard της Vercel (4)	40
4.6	Dashboard της Vercel (5)	41

Κατάλογος Αποσπασμάτων Κώδικα

3.1	spotify.js	19
3.2	[...nextauth].js	20
3.3	[...nextauth].js	22
3.4	login.js	24
3.5	playerAtom.js	25
3.6	Search component	26
3.7	RecentlyPlayed component	27
3.8	Player component	28
3.9	Track component	29
3.10	Poster component	30
3.11	Body component	32
3.12	Body component	33
3.13	Dashboard component	35
3.14	Right component	36

Κεφάλαιο 1

Εισαγωγή

Στην ψηφιακή εποχή, η ροή μουσικής έχει αλλάξει τον τρόπο που καταναλώνουμε και απολαμβάνουμε τη μουσική. Με την εκθετική ανάπτυξη των πλατφορμών ροής, όπως το Spotify, οι χρήστες έχουν πλέον πρόσβαση σε μια εκτενή βιβλιοθήκη τραγουδιών από διάφορα είδη, καλλιτέχνες και εποχές. Αυτή η προσβασιμότητα, σε συνδυασμό με την πρόοδο των τεχνολογιών ιστού, έχει οδηγήσει στην εμφάνιση καινοτόμων διαδικτυακών εφαρμογών που στοχεύουν στη βελτίωση της εμπειρίας ροής μουσικής.

Το Spotify API [1] είναι μια ισχυρή διεπαφή που παρέχει στους προγραμματιστές πρόσβαση σε ένα ευρύ φάσμα λειτουργιών, όπως η αναζήτηση κομματιών, η δημιουργία λιστών αναπαραγωγής, η ανάκτηση πληροφοριών καλλιτέχνη, ακόμη και ο έλεγχος της αναπαραγωγής. Αξιοποιώντας τις δυνατότητες του Spotify API, η εφαρμογή στοχεύει να προσφέρει ένα ολοκληρωμένο σύνολο λειτουργιών που συναγωνίζονται τις παραδοσιακές πλατφόρμες ροής μουσικής.

1.1 Κίνητρο

Το κίνητρο πίσω από αυτήν την έρευνα πηγάζει από την επιθυμία να δημιουργηθεί ένα πρόγραμμα αναπαραγωγής μουσικής ανεξάρτητης πλατφόρμας, βασισμένο σε πρόγραμμα περιήγησης, το οποίο επιτρέπει στους χρήστες να έχουν πρόσβαση στα αγαπημένα τους κομμάτια από οπουδήποτε, χωρίς να χρειάζονται εφαρμογές συγκεκριμένης πλατφόρμας. Χρησιμοποιώντας το Spotify API, μπορούμε να αξιοποιήσουμε την τεράστια μουσική βιβλιοθήκη και τις δυνατότητες εξατομίκευσης που προσφέρει το Spotify, ενώ ενσωματώνουμε πρόσθετες λειτουργίες για τη βελτίωση της εμπειρίας χρήστη.

1.2 Σκοπός

Ο σκοπός της παρούσας πτυχιακής είναι η ανάπτυξη μιας εφαρμογής Web Player που εκμεταλλεύεται το Spotify API, με στόχο να βελτιώσει την εμπειρία ακρόασης μουσικής. Το Spotify API είναι ένα ισχυρό εργαλείο που παρέχει στους προγραμματιστές πρόσβαση σε μια ευρεία γκάμα λειτουργιών, όπως αναζήτηση τραγουδιών, δημιουργία λιστών αναπαραγωγής, ανάκτηση πληροφοριών καλλιτέχνη και ακόμα και έλεγχο αναπαραγωγής. Αξιοποιώντας τις δυνατότητες του Spotify API, η εφαρμογή Web Player που προτείνουμε αποσκοπεί να προσφέρει έναν πλήρη σετ χαρακτηριστικών που ανταγωνίζεται τις παραδοσιακές πλατφόρμες μουσικής αναπαραγωγής.

1.3 Συνεισφορά Εργασίας

Η εφαρμογή του Web Player που υλοποιήθηκε στο πλαίσιο της εργασίας προσφέρει μία ακόμα επιλογή που δίνει έμφαση στην πρόταση νέων τραγουδιών στον καθένα που χρησιμοποιεί το Spotify ώστε να ακούει μουσική του. Στην εργασία αναλύονται όλες οι νέες τεχνολογίες που χρησιμοποιήθηκαν στη διαδικασία της ανάπτυξης αλλά και με ποιον τρόπο αξιοποιήθηκαν.

1.4 Διάρθρωση Εργασίας

Στο παρόν κείμενο αναλύεται σε κεφάλαια όλη η διαδικασία ανάπτυξης της εφαρμογής, τα εργαλεία που χρησιμοποιήθηκαν, η αρχιτεκτονική και τέλος η διαδικασία του Deployment της εφαρμογής.

Στο κεφάλαιο 2 επομένως, γίνεται λεπτομερής ανάλυση των γλωσσών προγραμματισμού, των βιβλιοθηκών και των frameworks που χρησιμοποιήθηκαν κατά την ανάπτυξη αλλά και τι ακριβώς είναι το Spotify στου οποίου το API βασίζεται και η εφαρμογή του Web Player που υλοποιήθηκε.

Το κεφάλαιο 3 περιέχει την αρχιτεκτονική της εφαρμογής περιλαμβάνοντας το μοντέλο ανάπτυξης που χρησιμοποιήθηκε αλλά και αποσπάσματα κώδικα και εικόνες από τα components, τα atoms και τη διαδικασία σύνδεσης και αυθεντικοποίησης στην εφαρμογή.

Το κεφάλαιο 4 εστιάζει στο Deployment της εφαρμογής, περιγράφοντας αρχικά το περιβάλλον που χρησιμοποιήθηκε και στη συνέχεια τη διαδικασία του Deployment αναλύοντας κάθε βήμα που ακολουθήθηκε.

Τέλος, το κεφάλαιο 5 αποτελείται από τα συμπεράσματα που προέκυψαν από όλη τη διαδικασία ανάπτυξης, από τα προβλήματα που αντιμετωπίστηκαν και από μελλοντικές επεκτάσεις που θα μπορούσαν να προστεθούν στην εφαρμογή.

Κεφάλαιο 2

Περιγραφή Εργαλείων

Σε αυτό το κεφάλαιο της εργασίας γίνεται η αναλυτική περιγραφή των εργαλείων που χρησιμοποιήθηκαν για την ανάπτυξη του Web Player που χρησιμοποιεί το Spotify API. Συγκεκριμένα, περιγράφονται οι γλώσσες, οι βιβλιοθήκες και τα frameworks που χρησιμοποιήθηκαν για την ολοκλήρωση της εργασίας.

2.1 Spotify

Το Spotify είναι μια δημοφιλής υπηρεσία ψηφιακής ροής μουσικής που παρέχει στους χρήστες πρόσβαση σε μια τεράστια βιβλιοθήκη μουσικών κομματιών, άλμπουμ, podcast και άλλου περιεχομένου ήχου από διάφορους καλλιτέχνες και είδη. Επιτρέπει στους χρήστες να ανακαλύψουν, να ακούσουν και να δημιουργήσουν λίστες αναπαραγωγής με τα αγαπημένα τους τραγούδια. Το Spotify προσφέρει δωρεάν και premium επιλογές συνδρομής.

Οι χρήστες μπορούν να αναζητήσουν συγκεκριμένα τραγούδια, άλμπουμ ή καλλιτέχνες και να τα μεταδώσουν κατά παραγγελία. Το Spotify δημιουργεί εξατομικευμένες λίστες αναπαραγωγής για χρήστες με βάση τις συνήθειες ακρόασής τους, όπως "Discover Weekly", "Release Radar" και "Daily Mixes". Επίσης, επιμελείται λίστες αναπαραγωγής με βάση διαφορετικά θέματα, διαθέσεις και είδη, όπως "Κορυφαίες επιτυχίες", "Chill Vibes" και "Workout Jams". Οι χρήστες μπορούν να δημιουργήσουν τις δικές τους λίστες αναπαραγωγής προσθέτοντας τραγούδια από τον τεράστιο κατάλογο του Spotify. Προσφέρει επίσης ένα ευρύ φάσμα podcast για διάφορα θέματα, καθιστώντας το μια πλατφόρμα τόσο για μουσική όσο και για περιεχόμενο προφορικού λόγου.

Οι χρήστες μπορούν να ακολουθούν τους φίλους τους και να βλέπουν τι ακούνε, να μοιράζονται λίστες αναπαραγωγής και να συνεργάζονται σε λίστες αναπαραγωγής μαζί. Κάθε καλλιτέχνης έχει τη δική του σελίδα στο Spotify, η οποία περιλαμβάνει τη δισκογραφία, τα βιογραφικά του στοιχεία και το σχετικό περιεχόμενο. Το Spotify είναι διαθέσιμο σε διάφορες πλατφόρμες, όπως smartphone (iOS και Android), tablet, επιτραπέζιους υπολογιστές, έξυπνες τηλεοράσεις και άλλες συσκευές. Ενώ η δωρεάν έκδοση του Spotify περιλαμβάνει διαφημίσεις και περιορισμούς σε ορισμένες λειτουργίες, η premium συνδρομή καταργεί τις διαφημίσεις, επιτρέπει την ακρόαση εκτός σύνδεσης και παρέχει υψηλότερη ποιότητα ήχου.

2.2 Εργαλεία που Χρησιμοποιήθηκαν

Για την κάλυψη των αναγκών της εργασίας και ολοκλήρωση ενός πλήρως λειτουργικού Web Player όπου ο χρήστης θα μπορεί να αναζητεί οποιοδήποτε τραγούδι επιθυμεί και να το ελέγχει πλήρως, όπως την ένταση, το σημείο του τραγουδιού, να το προσθέσει στα αγαπημένα αλλά και την συσκευή από την οποία γίνεται η αναπαραγωγή του τραγουδιού,

έγινε χρήση διάφορων εργαλείων όπως γλώσσες προγραμματισμού, βιβλιοθήκες, frameworks, περιβάλλον ανάπτυξης και άλλων λογισμικών.

2.2.1 HTML

Η HTML (HyperText Markup Language) είναι μια γλώσσα σήμανσης που χρησιμοποιείται για τη δημιουργία και τον σχεδιασμό ιστοσελίδων. Είναι η κύρια γλώσσα που χρησιμοποιείται για να δομήσει το περιεχόμενο και να παρέχει μορφοποίηση και διάταξη στις ιστοσελίδες που βλέπουμε στον ιστό. Η HTML χρησιμοποιεί διάφορες ετικέτες (tags) που περιβάλλουν το περιεχόμενο και καθορίζουν την λειτουργία του. Κάθε ετικέτα έχει μια συγκεκριμένη σημασία και χρησιμοποιείται για να περιγράψει τον τύπο των δεδομένων που παρουσιάζονται στην ιστοσελίδα. Οι ετικέτες HTML περιλαμβάνουν τίτλους, κείμενο, εικόνες, συνδέσμους (links), φόρμες (forms) για τη συλλογή δεδομένων, πίνακες και πολλά άλλα.

Η HTML διευκολύνει την πρόσβαση και την ανάγνωση των πληροφοριών από διαφορετικές συσκευές και πλατφόρμες, καθώς παρέχει μια δομημένη μέθοδο για την παρουσίαση του περιεχομένου. Επίσης, συνδέεται συχνά με άλλες τεχνολογίες, όπως την CSS (Cascading Style Sheets) για την καλύτερη μορφοποίηση και τη διαχείριση της εμφάνισης του περιεχομένου και την JavaScript για την προσθήκη διαδραστικότητας στις ιστοσελίδες. Είναι ένα απαραίτητο εργαλείο για τους προγραμματιστές ιστού και διαδραματίζει καθοριστικό ρόλο στη δημιουργία και την παρουσίαση πληροφοριών στον κόσμο του διαδικτύου. Χάρη στην HTML, οι πληροφορίες μπορούν να παρουσιαστούν κατανοητά και διαδραστικά σε μια ποικιλία συσκευών και περιγητών [2].



Εικόνα 2.1: Λογότυπο της HTML [2]

2.2.2 CSS

Η CSS (Cascading Style Sheets) είναι μια γλώσσα στυλ που χρησιμοποιείται για τον σχεδιασμό και τη μορφοποίηση ιστοσελίδων που δημιουργούνται με τη χρήση HTML ή άλλων μεταγενέστερων γλωσσών σήμανσης. Επιτρέπει τον έλεγχο και την αλλαγή της εμφάνισης του περιεχομένου σε ιστοσελίδες, όπως το χρώμα, τη γραμματοσειρά, το διάστιχο, τα περιθώρια, τις διαστάσεις των εικόνων και πολλά άλλα. Η CSS λειτουργεί ως ένα επίπεδο στυλ που ξεχωρίζει από το περιεχόμενο της ιστοσελίδας που έχει δημιουργηθεί με τη χρήση HTML. Αυτός ο διαχωρισμός του περιεχομένου και του στυλ παρέχει πολλά πλεονεκτήματα, όπως τη δυνατότητα αλλαγής της εμφάνισης των ιστοσελίδων χωρίς να χρειάζεται να αλλάξει το περιεχόμενό τους, καθώς και την ευκολία συντήρησης και αναβάθμισης του στυλ σε ολόκληρο τον ιστότοπο.

Η CSS χρησιμοποιείται επίσης για να διαχειριστεί τη διάταξη των στοιχείων στη σελίδα, να δημιουργήσει εφέ κίνησης και μεταβολής, και να προσαρμόσει την εμφάνιση της ιστοσελίδας σε διάφορες συσκευές και οθόνες (responsive design). Η χρήση της βελτιώνει την εμπειρία του χρήστη, καθώς επιτρέπει στους σχεδιαστές και προγραμματιστές να δημιουργούν όμορφες και λειτουργικές ιστοσελίδες, και να ελέγχουν τον τρόπο με τον οποίο το περιεχόμενο παρουσιάζεται στους χρήστες [3].



Εικόνα 2.2: Λογότυπο της CSS [3]

2.2.3 JavaScript

Η ανάπτυξη του Web Player έγινε κυρίως με τη χρήση της γλώσσας προγραμματισμού JavaScript. Η JavaScript είναι μια διαδραστική και πολυδιάστατη προγραμματιστική γλώσσα που χρησιμοποιείται κυρίως για την ανάπτυξη δυναμικών και διαδραστικών ιστοσελίδων. Αποτελεί την πιο δημοφιλή γλώσσα προγραμματισμού στον κόσμο στο Front-end Web development και συχνά χρησιμοποιείται για την προσθήκη λειτουργικότητας, διαδραστικότητας και εφέ στις ιστοσελίδες.

Οι βασικοί τύποι εφαρμογών που δημιουργούνται με τη χρήση της JavaScript είναι:

- **Ιστοσελίδες:** Χρησιμοποιείται για την προσθήκη διαδραστικότητας στις ιστοσελίδες. Μπορεί να αλλάξει το περιεχόμενο, να δημιουργήσει animations, να επικοινωνεί με τους χρήστες μέσω φορμών και πολλά άλλα.
- **Web Application:** Η JavaScript είναι η βάση για τη δημιουργία δυναμικών Web Applications. Χρησιμοποιείται για την επεξεργασία δεδομένων, τη διαχείριση λογικής και την αλληλεπίδραση με τους χρήστες.
- **Εφαρμογές για κινητά:** Μέσω του framework React Native, ο κώδικας JavaScript μπορεί να χρησιμοποιηθεί για την ανάπτυξη εφαρμογών για κινητά που τρέχουν σε διάφορα λειτουργικά συστήματα, όπως iOS και Android.
- **Ανάπτυξη παιχνιδιών:** Υπάρχουν και παιχνίδια που χρησιμοποιούν JavaScript για την δημιουργία παιχνιδιών στον περιηγητή.

Η JavaScript δίνει τη δυνατότητα στους προγραμματιστές να δημιουργούν ευέλικτες και επαναχρησιμοποιήσιμες λειτουργίες (functions) και μπορεί να χρησιμοποιηθεί με την HTML και CSS για να δημιουργηθούν πλούσιες και εντυπωσιακές εμπειρίες χρήστη (UX) στον κόσμο του ιστού και των εφαρμογών διαδικτύου [4].



Εικόνα 2.3: Λογότυπο της JavaScript [4]

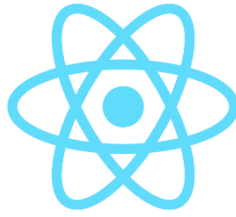
2.2.4 React

Για την υλοποίηση του Web Player χρησιμοποιήθηκε η React. Η React είναι μια βιβλιοθήκη της JavaScript ανοιχτού κώδικα που αναπτύχθηκε από το Facebook. Χρησιμοποιείται για τη δημιουργία διεπαφής χρήστη (UI) και Front-end components για εφαρμογές ιστού. Η React επιτρέπει στους προγραμματιστές να δημιουργούν επαναχρησιμοποιήσιμα UI components και να διαχειρίζονται αποτελεσματικά την κατάσταση αυτών των components, με αποτέλεσμα μια πιο οργανωμένη και επεκτάσιμη βάση κώδικα.

Η αρχιτεκτονική της React βασίζεται στα components. Το UI αναλύεται σε μικρά και αυτοτελή components τα οποία μπορούν να επαναχρησιμοποιηθούν, να συνδυαστούν και να ενσωματωθούν για τη δημιουργία πολύπλοκων UI. Επίσης, η React χρησιμοποιεί ένα εικονικό DOM (Document Object Model), το οποίο είναι ένα ελαφρύ αντίγραφο του πραγματικού DOM. Όταν υπάρχουν αλλαγές σε ένα component, η React δημιουργεί ένα νέο εικονικό DOM, το συγκρίνει με το προηγούμενο και ενημερώνει μόνο τα απαραίτητα μέρη του πραγματικού DOM, ελαχιστοποιώντας τον αριθμό των άμεσων χειρισμών DOM βελτιώνοντας έτσι την απόδοση. Ακόμα, η React χρησιμοποιεί δηλωτική σύνταξη όπου οι προγραμματιστές καθορίζουν το πως θα πρέπει να είναι το UI με βάση την τρέχουσα κατάσταση και η React φροντίζει για την ενημέρωση του DOM ανάλογα.

Η React επίσης χρησιμοποιεί JavaScript XML (JSX), το οποίο είναι μια συντακτική επέκταση που επιτρέπει τους προγραμματιστές να γράφουν τύπου HTML κώδικα απευθείας στην JavaScript. Το JSX διευκολύνει τον καθορισμό της δομής των components και βελτιώνει την αναγνωσιμότητα του κώδικα. Επιπρόσθετα, η React ακολουθεί μια ροή δεδομένων μονής κατεύθυνσης, όπου τα δεδομένα ρέουν από τα parent components στα child components, κάτι που διευκολύνει την παρακολούθηση αλλαγών στην κατάσταση της εφαρμογής και την κατανόηση του τρόπου με τον οποίο χρησιμοποιούνται τα δεδομένα. Η React ακόμα επιτρέπει στα components να έχουν μία κατάσταση, η οποία είναι μεταβλητά δεδομένα που μπορούν να επηρεάσουν την εμφάνιση του στοιχείου. Η διαχείριση της κατάστασης μπορεί να γίνει χρησιμοποιώντας τοπική κατάσταση component ή με εξωτερικές βιβλιοθήκες όπως την Redux.

Η React διαθέτει ένα τεράστιο οικοσύστημα με πολυάριθμες βιβλιοθήκες και εργαλεία που επεκτείνουν τις δυνατότητές της. Υπάρχουν βιβλιοθήκες για δρομολόγηση, διαχείριση κατάστασης, UI components και πολλά άλλα, διευκολύνοντας τους προγραμματιστές να δημιουργούν πολύπλοκες εφαρμογές. Η δημοτικότητα της React πηγάζει από την αποτελεσματικότητά της, την απόδοσή της και την ικανότητά της να δημιουργεί εξαιρετικά διαδραστικά και δυναμικά UIs. Χρησιμοποιείται ευρέως από προγραμματιστές ιστού και είναι μια από τις πιο δημοφιλείς βιβλιοθήκες Front-end για τη δημιουργία σύγχρονων εφαρμογών ιστού [5].



Εικόνα 2.4: Λογότυπο της React [5]

2.2.5 Next.js

Το Next.js είναι ένα δημοφιλές framework ανάπτυξης ανοιχτού κώδικα για την React που χρησιμοποιείται για τη δημιουργία ευέλικτων και γρήγορων Εφαρμογών Μίας Σελίδας (SPA) [17] και στατικών ιστοτόπων. Βασίζεται στην τεχνολογία των React Components και παρέχει πολλές πρόσθετες δυνατότητες για την ευκολία και την αποτελεσματικότητα των προγραμματιστών κατά την ανάπτυξη της ιστοσελίδας. Χρησιμοποιεί την ίδια σύνταξη και βασικές αρχές με την React, καθιστώντας το εύκολο στην κατανόηση και την ενσωμάτωση για προγραμματιστές που είναι εξοικειωμένοι με την React. Επίσης, το Next.js επιτρέπει τα δυναμικά δεδομένα να κάνουν pre-render στον διακομιστή πριν από την παράδοση της σελίδας στον πελάτη. Αυτό επιτρέπει ταχύτερους χρόνους φόρτωσης και βελτιώνει την εμπειρία του χρήστη. Ακόμα, μπορεί να κάνει pre-render στατικό περιεχόμενο κατά το χρόνο δημιουργίας και χωρίζει αυτόματα τον κώδικα σε μικρότερα κομμάτια προσφέροντας ακόμη ταχύτερη φόρτωση σελίδων και μειωμένους χρόνους απόκρισης.

Επίσης, το Next.js έχει την δυνατότητα δρομολόγησης η οποία φορτώνει αυτόματα τις απαιτούμενες σελίδες καθώς οι χρήστες περιηγούνται στον ιστότοπο. Διαθέτει HMR (Hot Module Replacement), κάτι που επιτρέπει στους προγραμματιστές να βλέπουν τις αλλαγές που κάνουν σε πραγματικό χρόνο χωρίς να απαιτείται ανανέωση σελίδας. Συνολικά, το Next.js είναι ένα πολύ ισχυρό framework που απλοποιεί την ανάπτυξη σε React και προσφέρει ένα ευρύ φάσμα δυνατοτήτων για τη δημιουργία εφαρμογών ιστού και στατικών ιστοτόπων. Έχει κερδίσει σημαντική δημοτικότητα στην κοινότητα ανάπτυξης ιστού λόγω της απλότητας, της απόδοσης και της ευκολίας χρήσης του [6].



Εικόνα 2.5: Λογότυπο του Next.js [6]

2.2.6 Tailwind CSS

Το Tailwind CSS είναι ένα πολύ προσαρμόσιμο και utility-first framework το οποίο απλοποιεί τη διαδικασία διαμόρφωσης εφαρμογών ιστού. Σε αντίθεση με τα παραδοσιακά frameworks της CSS που έρχονται με προσχεδιασμένα στοιχεία, το Tailwind παρέχει ένα σύνολο από κατηγορίες βοηθητικών προγραμμάτων που μπορούν να συνδυαστούν ώστε να μορφοποιήσουν HTML στοιχεία. Επιτρέπει δηλαδή στους προγραμματιστές να δημιουργούν γρήγορα, σύγχρονα και ανταποκρινόμενα UIs. Είναι ένα εξαιρετικά προσαρμόσιμο framework, επιτρέποντας στους προγραμματιστές να το διαμορφώσουν με βάση τις συγκεκριμένες απαιτήσεις σχεδιασμού του έργου τους. Χρησιμοποιώντας το αρχείο διαμόρφωσης του Tailwind, οι προγραμματιστές μπορούν να ενεργοποιήσουν ή να απενεργοποιήσουν τις κατηγορίες βοηθητικών προγραμμάτων, να προσθέσουν νέες ή να προσαρμόσουν τα υπάρχοντα στυλ.

Επειδή είναι utility-first framework οι προγραμματιστές μπορούν εύκολα να συνδυάσουν και να ταιριάξουν διάφορες κλάσεις ώστε να δημιουργήσουν μοναδικά σχέδια. Αυτή η αρθρωτή προσέγγιση βοηθάει στην επαναχρησιμοποίηση και στη διατήρηση μιας πιο οργανωμένης βάσης κώδικα. Σε αντίθεση με άλλα frameworks που έρχονται με προκαθορισμένα θέματα, αυτό δεν προσθέτει κάποιο θέμα στο έργο, αλλά παρέχει ένα σημείο εκκίνησης από το οποίο ο καθένας μπορεί να δημιουργήσει το δικό του σχέδιο από την αρχή. Ακόμα, δεδομένου ότι το Tailwind περιλαμβάνει μόνο τις κατηγορίες βοηθητικών κλάσεων που χρησιμοποιούνται πραγματικά στο έργο, το CSS αρχείο που προκύπτει είναι συνήθως μικρότερο σε σύγκριση με αυτά από τα παραδοσιακά frameworks, έχοντας ως αποτέλεσμα καλύτερη απόδοση.

Τέλος, το Tailwind είναι μια εξαιρετική επιλογή για προγραμματιστές που θέλουν να έχουν περισσότερο έλεγχο στη σχεδιάσή τους, αξιοποιώντας παράλληλα ένα σύνολο έτοιμων προς χρήση κλάσεων για να επιταχύνουν την ανάπτυξη. Είναι ιδιαίτερα δημοφιλές μεταξύ των σύγχρονων προγραμματιστών ιστού που αναζητούν μια περισσότερο βασισμένη στη χρησιμότητα προσέγγιση για το στυλ των εφαρμογών τους, έχει μια μεγάλη και ενεργή κοινότητα και υπάρχουν διάφορες προσθήκες, επεκτάσεις και ενσωματώσεις τρίτων που είναι διαθέσιμες για την ενίσχυση της λειτουργικότητας του framework [7].



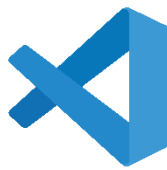
Εικόνα 2.6: Λογότυπο του tailwindcss [7]

2.2.7 Visual Studio Code

Για τη συγγραφή του κώδικα χρησιμοποιήθηκε το ολοκληρωμένο περιβάλλον ανάπτυξης (IDE), Visual Studio Code (VS Code). Το VS Code είναι ένα δημοφιλές και δωρεάν περιβάλλον ανάπτυξης που έχει αναπτυχθεί από τη Microsoft. Πρόκειται για ένα ανοιχτού κώδικα λογισμικό, που στοχεύει στην υποστήριξη προγραμματιστών σε διάφορες πλατφόρμες και γλώσσες προγραμματισμού. Το VS Code προσφέρει πληθώρα εργαλείων για τη δημιουργία και αποσφαλμάτωση κώδικα, κάνοντάς το μια δυνατή επιλογή για προγραμματιστές σε διάφορους τομείς. Ξεχωρίζει για την απλότητά του και την γρήγορη απόκρισή του, κάτι που το καθιστά ιδανικό για προγραμματιστές που εργάζονται σε

υπολογιστές με περιορισμένους πόρους. Επίσης, υποστηρίζει τις τρεις κύριες πλατφόρμες Windows, macOS και Linux, επιτρέποντας στους προγραμματιστές να εργάζονται στο λειτουργικό σύστημα της επιλογής τους. Υποστηρίζει πολλές γλώσσες προγραμματισμού και παρέχει προηγμένο IntelliSense, προβλέποντας αυτόματα τις μεθόδους, τις μεταβλητές και τα πρότυπα κώδικα κατά την πληκτρολόγηση. Ακόμα, επιτρέπει την εγκατάσταση επεκτάσεων που προσθέτουν νέες λειτουργίες και δυνατότητες, προσαρμόζοντας το περιβάλλον ανάπτυξης σύμφωνα με τις ανάγκες κάθε προγραμματιστή.

Συνολικά, το Visual Studio Code αποτελεί ένα εξαιρετικά ευέλικτο και ισχυρό περιβάλλον ανάπτυξης που καλύπτει τις ανάγκες προγραμματιστών σε διάφορους τομείς και παρέχει τα απαραίτητα εργαλεία για τη δημιουργία υψηλής ποιότητας λογισμικού [8].



Εικόνα 2.7: Λογότυπο του Visual Studio Code [8]

2.2.8 NextAuth.js

Η NextAuth.js είναι μία βιβλιοθήκη ανοιχτού κώδικα που χρησιμοποιείται για την αυθεντικοποίηση και στην παρούσα εργασία χρησιμοποιήθηκε για την αυθεντικοποίηση του προφίλ του χρήστη στο Spotify. Παρέχει ένα σύνολο εργαλείων και utilities για την απλοποίηση της διαδικασίας προσθήκης αυθεντικοποίησης και διαχείρισης ταυτότητας σε εφαρμογές ιστού που έχουν δημιουργηθεί με την Next.js. Η NextAuth.js υποστηρίζει ένα ευρύ φάσμα παρόχων αυθεντικοποίησης, συμπεριλαμβανομένων δημοφιλών πλατφορμών μέσω κοινωνικής δικτύωσης (όπως Google, Facebook, Twitter, GitHub, Spotify, κ.λπ.), παρόχων ταυτότητας (όπως Auth0, Okta και άλλοι), καθώς και παραδοσιακές μεθόδους αυθεντικοποίησης μέσω email και κωδικού πρόσβασης. Έχει σχεδιαστεί για να λειτουργεί απρόσκοπτα με εφαρμογές Next.js, καθιστώντας εύκολη τη ρύθμιση και τη διαμόρφωση της αυθεντικοποίησης σε ένα Next.js έργο.

Επίσης, η NextAuth.js ελέγχει τη διαχείριση περιόδων σύνδεσης, συμπεριλαμβανομένης της διαχείρισης περιόδου σύνδεσης του χρήστη, τη δημιουργία tokens για ασφαλή περίοδο σύνδεσης και το χειρισμό των session cookies. Επιτρέπει στους προγραμματιστές να προσαρμόζουν τη ροή της αυθεντικοποίησης και να ενσωματώνουν και άλλους παρόχους ή συστήματα αυθεντικοποίησης. Παρέχει ενσωματωμένη υποστήριξη για τη διαχείριση των λογαριασμών των χρηστών, συμπεριλαμβάνοντας λειτουργίες όπως δημιουργία λογαριασμού, επαναφορά κωδικού πρόσβασης και επαλήθευση email.

Τέλος, το NextAuth.js είναι ένα πολύ καλά συντηρημένο και αναπτυγμένο ενεργά framework, καθιστώντας το μια ασφαλή και αξιόπιστη επιλογή για τον χειρισμό της αυθεντικοποίησης και είναι μια δημοφιλής επιλογή για προγραμματιστές που θέλουν να προσθέσουν αυθεντικοποίηση και διαχείριση ταυτότητας στις Next.js εφαρμογές τους χωρίς να χρειάζεται να δημιουργήσουν ολόκληρο το σύστημα αυθεντικοποίησης από την αρχή. Απλοποιεί έτσι την

διαδικασία, επιτρέποντας στους προγραμματιστές να επικεντρωθούν στη δημιουργία άλλων βασικών χαρακτηριστικών [9].



Εικόνα 2.8: Λογότυπο της NextAuth.js [9]

2.2.9 Headless UI

Η Headless UI (User Interface) είναι μια βιβλιοθήκη ανοιχτού κώδικα της React που παρέχει στοιχεία διεπαφής χωρίς στυλ, προσβάσιμα και πλήρως προσαρμόσιμα για τη δημιουργία UI σε React εφαρμογές. Ακολουθεί το "ακέφαλο" μοτίβο διεπαφής χρήστη (UI), που σημαίνει ότι προσφέρει τη βασική λειτουργικότητα ενός UI στοιχείου χωρίς να επιβάλλει συγκεκριμένα οπτικά στυλ ή απόψεις σχετικά με τη σχεδίαση. Αυτό δίνει στους προγραμματιστές τον πλήρη έλεγχο της εμφάνισης και της συμπεριφοράς των στοιχείων, επιτρέποντάς τους να τα ενσωματώνουν απρόσκοπτα στα έργα τους. Η βιβλιοθήκη έχει σχεδιαστεί για να δίνει προτεραιότητα στην προσβασιμότητα, διευκολύνοντας τους προγραμματιστές να δημιουργούν περιεκτικά UI που μπορούν να χρησιμοποιηθούν από όλους τους χρήστες.

Η Headless UI είναι μία πολύ ελαφριά βιβλιοθήκη και τα components της έχουν ελάχιστο αντίκτυπο στο συνολικό μέγεθος του πακέτου μιας εφαρμογής και παρέχει σαφή και ολοκληρωμένη τεκμηρίωση, διευκολύνοντας τους προγραμματιστές να κατανοήσουν πώς να χρησιμοποιούν και να προσαρμόζουν τα components. Τέλος, είναι ιδιαίτερα χρήσιμη για έργα όπου οι απαιτήσεις σχεδιασμού είναι συγκεκριμένες και όπου οι προγραμματιστές θέλουν να διατηρήσουν τον πλήρη έλεγχο της οπτικής παρουσίασης του UI και να μειώσουν τον επαναλαμβανόμενο κώδικα που χρειάζεται να γράψουν [10].



Εικόνα 2.9: Λογότυπο της Headless UI [10]

2.2.10 Recoil

Η Recoil είναι μία βιβλιοθήκη διαχείρισης κατάστασης (state management) για React εφαρμογές και είναι ανοιχτού κώδικα. Αναπτύχθηκε και αυτήν από το Facebook και έχει σχεδιαστεί για χρήση ειδικά με React εφαρμογές, καθιστώντας το μια φυσική επιλογή για προγραμματιστές που κατασκευάζουν πολύπλοκα και μεγάλης κλίμακας projects σε React. Παρέχει έναν ευέλικτο και διαισθητικό τρόπο διαχείρισης κατάστασης των React components και επιτρέπει στους προγραμματιστές να δημιουργήσουν μια καθολική κατάσταση που μπορεί να μοιραστεί σε διαφορετικά μέρη μιας εφαρμογής. Είναι εμπνευσμένο από την έννοια των

atoms, των selectors και του reactive προγραμματισμού, κάτι που κάνει τη διαχείριση κατάστασης πιο προβλέψιμη και διατηρήσιμη.

Τα atoms είναι μονάδες κατάστασης στη Recoil.js. Αντιπροσωπεύουν μεμονωμένα κομμάτια κατάστασης που μπορούν να διαβαστούν και να ενημερωθούν από components. Κάθε atom είναι ένα κομμάτι κατάστασης που μπορεί να μοιραστεί μεταξύ των components. Για να οριστεί ένα atom χρειάζεται ένα μοναδικό κλειδί και μια προεπιλεγμένη τιμή. Μετά τον ορισμό, μπορεί να χρησιμοποιηθεί αυτό το atom για κοινή χρήση κατάστασης σε κάθε component που θέλει ο προγραμματιστής.

Οι selectors αντιπροσωπεύουν μια συνάρτηση ή μία συμπληρωματική (derived) κατάσταση στη Recoil.js. Μπορούν να παρομοιαστούν με μία idempotent ή καθαρή λειτουργία (pure function) που επιστρέφει πάντα την ίδια τιμή για ένα δεδομένο σύνολο εξαρτημένων τιμών.

Ο reactive προγραμματισμός σημαίνει ότι τα components αντιδρούν αυτόματα σε αλλαγές της κατάστασης στην οποία ανήκουν. Όταν αλλάζει η τιμή ενός atom ή ενός selector, τα εξαρτώμενα components κάνουν re-render με τα ενημερωμένα δεδομένα.

Η Recoil.js υποστηρίζει ασύγχρονες ενημερώσεις κατάστασης, επιτρέποντας στους προγραμματιστές να διαχειρίζονται ασύγχρονες ροές δεδομένων κατά την ενημέρωση της κατάστασης. Επίσης στοχεύει στην ελαχιστοποίηση του boilerplate κώδικα, καθιστώντας ευκολότερο και πιο αποτελεσματικό για τους προγραμματιστές τη διαχείριση της κατάστασης στις εφαρμογές. Είναι χτισμένη με TypeScript και παρέχει πρώτης τάξεως υποστήριξη για type checking, καθιστώντας το ασφαλέστερο και πιο αξιόπιστο στη χρήση του σε TypeScript έργα. Ακόμα, συνοδεύεται από ένα πακέτο εργαλείων για προγραμματιστές που παρέχει πληροφορίες για την τρέχουσα κατάσταση και βοηθά τους προγραμματιστές να εντοπίσουν σφάλματα και να βελτιστοποιήσουν τη διαχείριση κατάστασης των εφαρμογών.

Τέλος, είναι μια ισχυρή και επεκτάσιμη λύση διαχείρισης κατάστασης για React εφαρμογές. Προσφέρει ένα απλό API, απρόσκοπτη ενοποίηση με τη React και αποτελεσματική διαχείριση της global κατάστασης μίας εφαρμογής. Είναι ιδιαίτερα κατάλληλη για έργα με πολύπλοκες απαιτήσεις κατάστασης ή όταν η διαχείριση κατάστασης σε πολλά components γίνεται δύσκολη με τη χρήση άλλων λύσεων διαχείρισης κατάστασης [11].



Εικόνα 2.10: Λογότυπο της Recoil.js [11]

2.2.11 Heroicons

Τα icons που χρησιμοποιήθηκαν στην εργασία ήταν από τη Heroicons. Η Heroicons είναι μια βιβλιοθήκη ανοιχτού κώδικα που παρέχει μια συλλογή από προσαρμόσιμα, δωρεάν στη χρήση εικονίδια SVG για React εφαρμογές. Αυτά τα εικονίδια έχουν σχεδιαστεί από την ομάδα Heroicons και μπορούν να χρησιμοποιηθούν για τη βελτίωση του UI και τη βελτίωση της οπτικής ελκυστικότητας των έργων. Το σετ εικονιδίων Heroicons διατίθεται σε δύο στυλ: με περίγραμμα και συμπαγές. Τα εικονίδια περιγράμματος έχουν απλό και καθαρό σχεδιασμό,

ενώ τα συμπαγή εικονίδια έχουν πιο τολμηρή και γεμάτη εμφάνιση. Η βιβλιοθήκη προσφέρει ένα ευρύ φάσμα εικονιδίων, καλύπτοντας διάφορες κατηγορίες, όπως κοινά στοιχεία UI, βέλη, εικονίδια μέσων κοινωνικής δικτύωσης και άλλα. Τα εικονίδια είναι κλιμακούμενα και ανεξάρτητα από την ανάλυση. Αυτό σημαίνει ότι τα εικονίδια θα φαίνονται καθαρά και ευκρινή σε οποιοδήποτε μέγεθος ή ανάλυση οθόνης. Μπορούν εύκολα να προσαρμοστούν ως προς το μέγεθος και το χρώμα χρησιμοποιώντας τα `standard props` της React και CSS. Έχουν σχεδιαστεί με γνώμονα την προσβασιμότητα, έχουν σωστές ετικέτες ARIA και μπορούν να χρησιμοποιηθούν για τη βελτίωση της προσβασιμότητας του UI. Η βιβλιοθήκη υποστηρίζει `tree shaking`, πράγμα που σημαίνει ότι μόνο τα εικονίδια που χρησιμοποιούνται στην εφαρμογή θα συμπεριληφθούν, μειώνοντας το συνολικό μέγεθος του πακέτου.

Χρησιμοποιώντας τη Heroicons, οι προγραμματιστές μπορούν να προσθέσουν γρήγορα υψηλής ποιότητας, προσαρμόσιμα εικονίδια στις React εφαρμογές τους χωρίς να χρειάζεται να δημιουργήσουν ή να αναζητήσουν εικονίδια αλλού. Η βιβλιοθήκη είναι μια δημοφιλής επιλογή μεταξύ των προγραμματιστών σε React λόγω της ευκολίας χρήσης της και της εκτεταμένης συλλογής εικονιδίων που προσφέρει [12].



Εικόνα 2.11: Λογότυπο της Heroicons [12]

2.2.12 Node.js

Το Node.js είναι ένα ανοιχτού κώδικα, πολλαπλών πλατφορμών, περιβάλλον εκτέλεσης της JavaScript που βασίζεται στη μηχανή V8 JavaScript του Chrome. Επιτρέπει στους προγραμματιστές να εκτελούν κώδικα JavaScript εκτός ενός προγράμματος περιήγησης ιστού, επιτρέποντας τη δημιουργία σεναρίων από την πλευρά του διακομιστή και την ανάπτυξη κλιμακούμενων εφαρμογών δικτύου. Το Node.js δημιουργήθηκε από τον Ryan Dahl και κυκλοφόρησε για πρώτη φορά το 2009. Χρησιμοποιεί ένα event-driven, non-blocking I/O μοντέλο. Αυτό σημαίνει ότι μπορεί να χειριστεί αποτελεσματικά πολλές ταυτόχρονες συνδέσεις χωρίς να εμποδίζει την εκτέλεση του κώδικα, καθιστώντας το κατάλληλο για εφαρμογές με υψηλές I/O λειτουργίες, όπως εφαρμογές συνομιλίας σε πραγματικό χρόνο, υπηρεσίες ροής και πολλά άλλα. Εφόσον το Node.js χρησιμοποιεί JavaScript, οι προγραμματιστές μπορούν να χρησιμοποιήσουν την ίδια γλώσσα προγραμματισμού για την ανάπτυξη τόσο από την πλευρά του πελάτη (πρόγραμμα περιήγησης) όσο και από την πλευρά του διακομιστή (Node.js), γεγονός που μπορεί να απλοποιήσει τη διαδικασία ανάπτυξης. Συνοδεύεται από NPM, έναν ισχυρό διαχειριστή πακέτων που επιτρέπει στους προγραμματιστές να εγκαθιστούν, να διαχειρίζονται και να μοιράζονται επαναχρησιμοποιήσιμες μονάδες κώδικα και βιβλιοθήκες. Το τεράστιο αποθετήριο NPM παρέχει χιλιάδες πακέτα που μπορούν να ενσωματωθούν σε έργα Node.js για να επεκτείνουν τη λειτουργικότητά τους.

Το Node.js είναι ελαφρύ και έχει σχετικά μικρό αποτύπωμα σε σύγκριση με τους παραδοσιακούς διακομιστές ιστού, καθιστώντας το αποτελεσματικό και γρήγορο. Η ικανότητά του να χειρίζεται ταυτόχρονες συνδέσεις βελτιώνει περαιτέρω την απόδοσή του. Έχει

σχεδιαστεί για να είναι επεκτάσιμο, καθιστώντας το κατάλληλο για τη δημιουργία εφαρμογών που μπορούν να χειριστούν μεγάλο αριθμό χρηστών και συνδέσεων ταυτόχρονα. Έχει μια ζωντανή και ενεργή κοινότητα, που συμβάλλει συνεχώς στην ανάπτυξη και την ανάπτυξή του. Αυτό είχε ως αποτέλεσμα ένα πλούσιο οικοσύστημα μονάδων και εργαλείων που είναι διαθέσιμα για προγραμματιστές. Το Node.js χρησιμοποιείται συνήθως για τη δημιουργία διαφόρων τύπων εφαρμογών, όπως διακομιστές ιστού, API (Application Programming Interfaces), εφαρμογές σε πραγματικό χρόνο (π.χ. εφαρμογές συνομιλίας), υπηρεσίες ροής, εργαλεία γραμμής εντολών και άλλα. Η δημοτικότητά του έχει αυξηθεί σημαντικά λόγω των μοναδικών δυνατοτήτων και της ευελιξίας του για την κατασκευή σύγχρονων διαδικτυακών εφαρμογών [13].



Εικόνα 2.12: Λογότυπο του Node.js [13]

2.2.13 NPM.js

Το NPM (Node Package Manager) είναι ο διαχειριστής πακέτων και διαδικτυακός χώρος αποθήκευσης για τα πακέτα και τις λειτουργικές μονάδες Node.js που χρησιμοποιήθηκαν κατά την ανάπτυξη. Το NPM συνοδεύεται από το Node.js, επομένως εγκαθίσταται αυτόματα με το Node.js. Χρησιμεύει ως κεντρικό μητρώο για χιλιάδες επαναχρησιμοποιήσιμες βιβλιοθήκες ανοιχτού κώδικα και εργαλεία που μπορούν να χρησιμοποιήσουν οι προγραμματιστές για να βελτιώσουν τα έργα τους και απλοποιεί τη διαδικασία εγκατάστασης, διαχείρισης και ενημέρωσης αυτών των πακέτων. Επίσης υποστηρίζει τη διαχείριση εκδόσεων για πακέτα. Οι προγραμματιστές μπορούν να καθορίσουν την έκδοση ενός πακέτου που θέλουν να χρησιμοποιήσουν στο έργο τους. Αυτό διασφαλίζει ότι το έργο παραμένει σταθερό και συνεπές, ακόμη και αν το πακέτο ενημερωθεί στο μέλλον. Ακόμα, παρακολουθεί τις εξαρτήσεις μεταξύ των πακέτων. Όταν εγκαθίσταται ένα πακέτο που βασίζεται σε άλλα πακέτα, το NPM εγκαθιστά και διαχειρίζεται αυτόματα αυτές τις εξαρτήσεις.

Το NPM ακολουθεί τη σημασιολογική έκδοση, η οποία αποτελείται από τρεις αριθμούς που χωρίζονται με τελείες. Οι αριθμοί έκδοσης υποδεικνύουν τη συμβατότητα του πακέτου με το έργο. Με την τήρηση της σημασιολογικής έκδοσης, οι προγραμματιστές μπορούν να ελέγχουν πότε θα ενημερώνουν τα πακέτα χωρίς να εισάγουν αλλαγές που μπορούν να δημιουργήσουν κάποιο πρόβλημα συμβατότητας. Οι προγραμματιστές μπορούν να δημοσιεύσουν τα δικά τους πακέτα στο μητρώο NPM. Αυτό διευκολύνει τους άλλους να ανακαλύψουν και να χρησιμοποιήσουν τον κώδικά τους. Το NPM έχει διαδραματίσει καθοριστικό ρόλο στην ανάπτυξη και τη δημοτικότητα του Node.js, καθώς επέτρεψε στην κοινότητα να μοιράζεται και να διανέμει εύκολα κώδικα. Έχει γίνει ένα απαραίτητο εργαλείο για τους προγραμματιστές που χρησιμοποιούν Node.js και χρησιμοποιείται ευρέως σε διάφορα έργα ανάπτυξης ιστού [14].



Εικόνα 2.13: Λογότυπο του NPM.js [14]

2.2.14 GitHub

Το GitHub είναι μια διαδικτυακή πλατφόρμα που είναι κομβική για την ανάπτυξη λογισμικού και τη συνεργασία χρησιμοποιώντας το σύστημα ελέγχου έκδοσης Git. Παρέχει εργαλεία και δυνατότητες που επιτρέπουν στους προγραμματιστές να συνεργάζονται, να διαχειρίζονται αποθετήρια κώδικα, να παρακολουθούν αλλαγές και να συνεργάζονται για την ανάπτυξη και συντήρηση κώδικα. Χρησιμοποιεί το Git, ένα κατακεντρωμένο σύστημα ελέγχου έκδοσης, για να παρακολουθεί τις αλλαγές στον πηγαίο κώδικα με την πάροδο του χρόνου. Αυτό επιτρέπει σε πολλούς προγραμματιστές να εργάζονται στην ίδια βάση κώδικα χωρίς να παρεμβαίνουν ο ένας στη δουλειά του άλλου. Επίσης, επιτρέπει στους προγραμματιστές να φιλοξενούν τα αποθετήρια κώδικά τους στο διαδίκτυο, καθιστώντας εύκολη την κοινή χρήση κώδικα με τον υπόλοιπο κόσμο. Κάθε αποθετήριο είναι μια συλλογή αρχείων και φακέλων που αντιπροσωπεύουν ένα έργο λογισμικού.

Πολλοί προγραμματιστές μπορούν να συνεργαστούν σε ένα έργο κάνοντας αλλαγές στα δικά τους κλαδιά κώδικα και στη συνέχεια συγχωνεύοντας αυτές τις αλλαγές σε έναν κύριο κλαδί το οποίο συχνά ονομάζεται "master" ή "main" κλαδί. Ακόμα, το GitHub παρέχει εργαλεία για τη διαχείριση και την επίλυση συγκρούσεων που μπορεί να προκύψουν κατά τη συγχώνευση των αλλαγών. Διαθέτει ένα σύστημα παρακολούθησης προβλημάτων που επιτρέπει στους προγραμματιστές να δημιουργούν και να διαχειρίζονται εργασίες, σφάλματα, αιτήματα λειτουργιών και άλλα ζητήματα που σχετίζονται με ένα έργο. Αυτά τα ζητήματα μπορούν να ανατεθούν, να επισημανθούν και να ιεραρχηθούν, καθιστώντας ευκολότερη τη διαχείριση της διαδικασίας ανάπτυξης. Όταν ένας προγραμματιστής θέλει να συνεισφέρει αλλαγές σε ένα έργο, δημιουργεί ένα αίτημα έλξης. Αυτό είναι ένα αίτημα για συγχώνευση των αλλαγών από τον κλάδο του στον κύριο κλάδο. Άλλοι συνεισφέροντες μπορούν να ελέγξουν τις αλλαγές, να τις συζητήσουν και να προτείνουν τροποποιήσεις πριν από τη συγχώνευση των αλλαγών. Διευκολύνει την επιθεώρηση κώδικα επιτρέποντας στους προγραμματιστές να σχολιάζουν συγκεκριμένες γραμμές κώδικα μέσα σε ένα αίτημα έλξης, κάτι που βοηθά στη διασφάλιση της ποιότητας και της ορθότητας του κώδικα πριν από τη συγχώνευσή του.

Το GitHub ενσωματώνεται με διάφορα εργαλεία CI/CD, επιτρέποντας στους προγραμματιστές να αυτοματοποιούν τη δοκιμή, τη δημιουργία και την ανάπτυξη του κώδικά τους διασφαλίζοντας ότι οι αλλαγές κώδικα ελέγχονται διεξοδικά και αναπτύσσονται ομαλά. Τα αποθετήρια στο GitHub μπορούν επίσης να περιέχουν wiki και τεκμηρίωση, καθιστώντας εύκολη την παροχή οδηγιών, επεξηγήσεων και οδηγιών για το έργο. Παρέχει μια πλατφόρμα για τους προγραμματιστές να ανακαλύπτουν, να διαχωρίζουν και να συνεισφέρουν σε έργα ανοιχτού κώδικα και ενθαρρύνει τη συνεργασία εντός της παγκόσμιας κοινότητας ανάπτυξης λογισμικού. Έχει παίξει καθοριστικό ρόλο στον κόσμο της ανάπτυξης λογισμικού ανοιχτού κώδικα και έχει γίνει ένα ουσιαστικό εργαλείο για μεμονωμένους προγραμματιστές, μικρές ομάδες και μεγάλους οργανισμούς και χρησιμοποιείται ευρέως για τη διαχείριση και τη συνεργασία σε έργα, τόσο ανοιχτού κώδικα όσο και ιδιόκτητα [15].



Εικόνα 2.14: Λογότυπο της GitHub [15]

2.2.15 Spotify for Developers

Το "Spotify for Developers" είναι μια πλατφόρμα που παρέχεται από το Spotify, τη δημοφιλή υπηρεσία ροής μουσικής, η οποία προσφέρει εργαλεία, πόρους και API (Application Programming Interfaces) στους προγραμματιστές για τη δημιουργία εφαρμογών και ενσωματώσεων που αλληλεπιδρούν με την υπηρεσία Spotify. Αυτή η πλατφόρμα επιτρέπει στους προγραμματιστές να δημιουργούν εφαρμογές και υπηρεσίες που σχετίζονται με τη μουσική που ενισχύουν και επεκτείνουν τη λειτουργικότητα του οικοσυστήματος του Spotify. Παρέχει ένα σύνολο από API που επιτρέπουν στους προγραμματιστές να έχουν πρόσβαση σε διάφορες πτυχές του καταλόγου μουσικής του Spotify, σε λίστες αναπαραγωγής χρηστών, δεδομένα χρήστη και πολλά άλλα. Αυτά τα API επιτρέπουν στους προγραμματιστές να δημιουργούν εφαρμογές που μπορούν να αναζητήσουν κομμάτια, να ανακτήσουν έργα τέχνης των άλμπουμ, να δημιουργήσουν λίστες αναπαραγωγής και να εκτελέσουν διάφορες ενέργειες που σχετίζονται με μουσικό περιεχόμενο. Επίσης, προσφέρει SDK (Software development kit) για διαφορετικές πλατφόρμες, όπως iOS, Android τα οποία απλοποιούν τη διαδικασία ενσωμάτωσης λειτουργιών του Spotify σε εφαρμογές. Αυτά τα SDK παρέχουν προκατασκευασμένα στοιχεία και εργαλεία που μπορούν να χρησιμοποιήσουν οι προγραμματιστές για να προσθέσουν εύκολα την αναπαραγωγή Spotify, την αυθεντικοποίηση και άλλες λειτουργίες στις εφαρμογές τους. Το Spotify Playback SDK επιτρέπει στους προγραμματιστές να ενσωματώνουν τις δυνατότητες αναπαραγωγής μουσικής του Spotify στις εφαρμογές τους. Αυτό επιτρέπει στους χρήστες να ακούν μουσική του Spotify απευθείας μέσα σε εφαρμογές τρίτων παρέχοντας παράλληλα στοιχεία ελέγχου και λειτουργίες αναπαραγωγής.

Το Spotify for Developers περιλαμβάνει εργαλεία αυθεντικοποίησης και εξουσιοδότησης, επιτρέποντας στους προγραμματιστές να συνδέουν με ασφάλεια τις εφαρμογές τους με τους λογαριασμούς Spotify των χρηστών. Το Spotify Web API επιτρέπει στους προγραμματιστές να έχουν πρόσβαση σε ένα ευρύ φάσμα των μουσικών δεδομένων του Spotify, συμπεριλαμβανομένων κομματιών, άλμπουμ, καλλιτεχνών, λιστών αναπαραγωγής και πληροφοριών χρήστη. Αυτό το API επιτρέπει στους προγραμματιστές να δημιουργούν εφαρμογές που μπορούν να αλληλεπιδράσουν με τον κατάλογο του Spotify και τις βιβλιοθήκες χρηστών. Ακόμα, Ο πίνακας ελέγχου προγραμματιστών παρέχει μια κεντρική τοποθεσία για τους προγραμματιστές να διαχειρίζονται τις εφαρμογές Spotify, να αποκτούν API κλειδιά και να παρακολουθούν τη χρήση των εφαρμογών.

Το Spotify παρέχει ολοκληρωμένη τεκμηρίωση και οδηγούς εκμάθησης για να βοηθήσει τους προγραμματιστές να κατανοήσουν και να χρησιμοποιήσουν αποτελεσματικά τα διαθέσιμα API

και SDK. Οι προγραμματιστές που χρησιμοποιούν την πλατφόρμα Spotify for Developers μπορούν να αλληλοεπιδράσουν με την κοινότητα, να κάνουν ερωτήσεις και να αναζητήσουν βοήθεια μέσω φόρουμ και καναλιών υποστήριξης. Μέσω αυτού, οι προγραμματιστές μπορούν να δημιουργήσουν μια μεγάλη ποικιλία εφαρμογών και ενσωματώσεων, όπως εφαρμογές προτάσεων μουσικής, γεννήτριες λιστών αναπαραγωγής, εργαλεία ανάλυσης μουσικής και πολλά άλλα. Αυτές οι εφαρμογές μπορούν να βελτιώσουν την εμπειρία του Spotify για τους χρήστες και να συμβάλουν στο συνολικό οικοσύστημα λογισμικού και υπηρεσιών που σχετίζονται με τη μουσική [16].



Εικόνα 2.15: Λογότυπο του Spotify for Developers [16]

Κεφάλαιο 3

Αρχιτεκτονική

Η αρχιτεκτονική μίας εφαρμογής web αναφέρεται στη δομή και τον τρόπο με τον οποίο είναι οργανωμένη η εφαρμογή αυτή, συμπεριλαμβανομένων των συνιστωσών της και των σχέσεων μεταξύ τους. Σκοπός της αρχιτεκτονικής είναι να παρέχει μια βάση για την ανάπτυξη, τη συντήρηση και την επέκταση της web εφαρμογής με τρόπο που να είναι αποδοτικός, ασφαλής και εύκολος στη διαχείριση. Υπάρχουν διάφορες αρχιτεκτονικές που μπορεί να χρησιμοποιηθούν για την ανάπτυξη μίας web εφαρμογής, ανάλογα με τις ανάγκες του έργου, τη κλίμακα του, τη διαθεσιμότητα πόρων και άλλους παράγοντες.

Για το έργο αυτό επιλέχθηκε η αρχιτεκτονική SPA (Single Page Application) [17]. Η αρχιτεκτονική αυτήν αναφέρεται σε έναν τύπο web εφαρμογής όπου όλο το περιεχόμενο φορτώνεται σε μια μόνο σελίδα, και η αλληλεπίδραση με τον διακομιστή γίνεται δυναμικά, χωρίς την ανάγκη για νέες φορτώσεις σελίδων. Αυτό επιτυγχάνεται συνήθως με τη χρήση τεχνολογιών όπως το AJAX (Asynchronous JavaScript and XML), και το JSON (JavaScript Object Notation).

Όλο το περιεχόμενο της εφαρμογής (HTML, CSS, JavaScript) φορτώνεται αρχικά κατά την εκκίνηση της εφαρμογής. Αυτό συμπεριλαμβάνει τα πάντα, από την αρχική διάταξη (layout) μέχρι τα δεδομένα που θα εμφανιστούν. Οι αλληλεπιδράσεις του χρήστη (π.χ., κλικ σε κουμπιά ή σύρσιμο) προκαλούν δυναμικές αλλαγές στο περιεχόμενο της σελίδας χωρίς την ανάγκη για ανανέωση ολόκληρης της σελίδας. Οι SPA συνήθως υποστηρίζουν δυνατότητες δρομολόγησης, που επιτρέπουν στον χρήστη να πλοηγείται σε διάφορες "σελίδες" μέσα στην εφαρμογή, αν και αυτές δεν αντιστοιχούν σε πραγματικές HTML σελίδες. Οι δρομολογητές (routers) διαχειρίζονται τη δρομολόγηση και την εμφάνιση του αντίστοιχου περιεχομένου.

Για την ανάκτηση δεδομένων από τον διακομιστή, οι SPA χρησιμοποιούν συχνά ασύγχρονα αιτήματα AJAX. Αυτό επιτρέπει τη φόρτωση δεδομένων χωρίς την ανάγκη για ανανέωση της σελίδας. Οι SPA διατηρούν την κατάσταση της εφαρμογής στον πελάτη, επιτρέποντας στο χρήστη να συνεχίσει από τη θέση όπου σταμάτησε, χωρίς την ανάγκη να ξαναφορτώσει τη σελίδα. Η αρχιτεκτονική SPA είναι δημοφιλής για διάφορους λόγους, συμπεριλαμβανομένης της βελτιωμένης εμπειρίας χρήστη και της αποτελεσματικότερης αξιοποίησης του δικτύου. Ωστόσο, απαιτεί προσεκτικό σχεδιασμό και διαχείριση κατάστασης για να αποφευχθούν προβλήματα απόδοσης και ασφάλειας.

3.1 Μοντέλο Ανάπτυξης

Το μοντέλο ανάπτυξης που χρησιμοποιήθηκε για την υλοποίηση του Web Player είναι το Μοντέλο Καταρράκτη (Waterfall Model) [18] καθώς θεωρήθηκε το καταλληλότερο για την συγκεκριμένη περίπτωση. Το μοντέλο αυτό είναι ένα παραδοσιακό και γραμμικό μοντέλο ανάπτυξης λογισμικού το οποίο όμως χρησιμοποιείται και στην ανάπτυξη ιστού (Web

Development). Συνήθως χρησιμοποιείται για την ανάπτυξη μεγάλων και πολύπλοκων έργων, όπου οι απαιτήσεις είναι σχετικά σταθερές και γνωστές από την αρχή. Το μοντέλο ανάπτυξης καταρράκτη ακολουθεί μια σειρά σταδίων, τα οποία προχωρούν με γραμμική σειρά από την αρχή μέχρι το τέλος [19]. Τα βασικά στάδια περιλαμβάνουν:

1. Συγκέντρωση απαιτήσεων: Το πρώτο βήμα είναι η συγκέντρωση και η τεκμηρίωση όλων των απαιτήσεων για την εφαρμογή Ιστού (Web Application). Αυτό περιλαμβάνει την κατανόηση των αναγκών, των χαρακτηριστικών, των λειτουργιών και των προτιμήσεων σχεδιασμού.
2. Σχεδιασμός Συστήματος: Με βάση τις απαιτήσεις που συγκεντρώθηκαν, δημιουργείται ένας λεπτομερής σχεδιασμός συστήματος. Αυτό περιλαμβάνει το σχεδιασμό της αρχιτεκτονικής του ιστοτόπου, τη διεπαφή χρήστη, τη ροή πλοήγησης και οποιεσδήποτε άλλες τεχνικές προδιαγραφές απαιτούνται για την ανάπτυξη.
3. Υλοποίηση: Μετά τον σχεδιασμό του συστήματος, μπορεί να ξεκινήσει η κωδικοποίηση της διαδικτυακής εφαρμογής. Αυτό περιλαμβάνει τη σύνταξη του κώδικα HTML, CSS και JavaScript, την ανάπτυξη λειτουργιών από την πλευρά του διακομιστή και την εφαρμογή πρόσθετων λειτουργιών ή API.
4. Δοκιμή: Μόλις ολοκληρωθεί η υλοποίηση, διενεργείται ενδελεχής δοκιμή για να διασφαλιστεί ότι η εφαρμογή Ιστού λειτουργεί όπως αναμένεται. Αυτό περιλαμβάνει τόσο λειτουργικές δοκιμές (έλεγχος εάν όλες οι λειτουργίες λειτουργούν σωστά) όσο και μη λειτουργικές δοκιμές (επαλήθευση απόδοσης, ασφάλειας και χρηστικότητας).
5. Ανάπτυξη: Μετά από επιτυχή δοκιμή, η εφαρμογή Ιστού αναπτύσσεται σε περιβάλλον παραγωγής, καθιστώντας την προσβάσιμη στους χρήστες. Αυτό περιλαμβάνει τη ρύθμιση διακομιστών, τη διαμόρφωση του περιβάλλοντος φιλοξενίας και τη διασφάλιση της σωστής ανάπτυξης της εφαρμογής.
6. Συντήρηση: Μόλις αναπτυχθεί η εφαρμογή Ιστού, απαιτείται συνεχής συντήρηση και υποστήριξη. Αυτό περιλαμβάνει την αντιμετώπιση σφαλμάτων, την εκτέλεση ενημερώσεων, την προσθήκη νέων λειτουργιών ή βελτίωση των ήδη υπάρχοντων με βάση τα σχόλια των χρηστών και τη διασφάλιση της ασφάλειας και της ενημέρωσης του ιστότοπου.

Ένα χαρακτηριστικό του μοντέλου ανάπτυξης καταρράκτη είναι ότι δεν παρέχει μεγάλη ευελιξία για αλλαγές κατά τη διάρκεια της ανάπτυξης. Κάθε στάδιο ολοκληρώνεται πριν από την έναρξη του επόμενου, και οποιαδήποτε αλλαγή απαιτεί συνήθως επανεκτέλεση των προηγούμενων σταδίων. Παρόλα αυτά, το μοντέλο σχεδίασης καταρράκτη εξακολουθεί να χρησιμοποιείται σε ορισμένα έργα όπου οι απαιτήσεις είναι γνωστές εκ των προτέρων και η ευελιξία δεν είναι το πιο σημαντικό κριτήριο.

3.2 Αυθεντικοποίηση και Σύνδεση

```

import SpotifyWebApi from "spotify-web-api-node";

const scopes = [
  "user-read-email",
  "playlist-read-private",
  "playlist-read-collaborative",
  "user-read-email",
  "streaming",
  "user-read-private",
  "user-library-read",
  "user-top-read",
  "user-library-modify",
  "user-read-playback-state",
  "user-modify-playback-state",
  "user-read-currently-playing",
  "user-read-recently-played",
  "user-follow-read",
].join(",");

const params = {
  scope: scopes,
};

const queryParamsString = new URLSearchParams(params);

const LOGIN_URL =
  "https://accounts.spotify.com/authorize?" + queryParamsString.toString();

const spotifyApi = new SpotifyWebApi({
  clientId: process.env.SPOTIFY_CLIENT_ID,
  clientSecret: process.env.SPOTIFY_CLIENT_SECRET,
});

export default spotifyApi;

export { LOGIN_URL };

```

Απόσπασμα κώδικα 3.1: *spotify.js*

Το παραπάνω απόσπασμα κώδικα αποτελεί ένα JavaScript module το οποίο αλληλεπιδρά με το Spotify API χρησιμοποιώντας τη βιβλιοθήκη 'spotify-web-api-node'. Ο πίνακας 'scopes' περιέχει ένα σύνολο από άδειες πεδίων τις οποίες η εφαρμογή ζητά από τον χρήστη. Τα πεδία (scopes) που αναφέρονται στον πίνακα αντιστοιχούν σε διάφορα δικαιώματα, όπως πρόσβαση σε λίστες αναπαραγωγής, στη βιβλιοθήκη και στο ιστορικό του χρήστη και άλλα.

Το αντικείμενο 'params' περιέχει τις παραμέτρους ερωτήματος που θα χρησιμοποιηθούν στο URL της αυθεντικοποίησης. Σε αυτήν την περίπτωση, είναι η παράμετρος πεδίου που έχει οριστεί στην ενωμένη λίστα πεδίων.

Το `'queryString'` δημιουργείται χρησιμοποιώντας τον κατασκευαστή `'URLSearchParams'` και το αντικείμενο `'params'`. Αυτό θα κωδικοποιήσει τις παραμέτρους σε μια συμβολοσειρά φιλική προς τη διεύθυνση URL.

Η σταθερά `'LOGIN_URL'` δημιουργείται συνδυάζοντας το τελικό σημείο αυθεντικοποίησης του Spotify (<https://accounts.spotify.com/authorize>) με τη συμβολοσειρά παραμέτρων ερωτήματος. Αυτή η διεύθυνση URL είναι όπου θα κατευθύνονται οι χρήστες προκειμένου να παραχωρήσουν άδεια στην εφαρμογή.

Στη συνέχεια δημιουργείται ένα instance της κλάσης `'SpotifyWebApi'` χρησιμοποιώντας τον κατασκευαστή `'SpotifyWebApi'`. Ως επιλογές διαμόρφωσης παίρνει το `'clientId'` και το `'clientSecret'`, οι τιμές αυτές διαβάζονται από μεταβλητές περιβάλλοντος `'process.env.SPOTIFY_CLIENT_ID'` και `'process.env.SPOTIFY_CLIENT_SECRET'`, οι οποίες θα πρέπει να οριστούν στο περιβάλλον της εφαρμογής για να διατηρούνται ασφαλείς οι ευαίσθητες πληροφορίες.

```
async function refreshAccessToken(token) {
  try {
    const url =
      "https://accounts.spotify.com/api/token?" +
      new URLSearchParams({
        client_id: process.env.SPOTIFY_CLIENT_ID,
        client_secret: process.env.SPOTIFY_CLIENT_SECRET,
        grant_type: "refresh_token",
        refresh_token: token.refreshToken,
      });

    const response = await fetch(url, {
      headers: {
        "Content-Type": "application/x-www-form-urlencoded",
      },
      method: "POST",
    });

    const refreshedTokens = await response.json();

    if (!response.ok) {
      throw refreshedTokens;
    }

    return {
      ...token,
      accessToken: refreshedTokens.access_token,
      accessTokenExpires: Date.now() + refreshedTokens.expires_in * 1000,
      refreshToken: refreshedTokens.refresh_token ?? token.refreshToken, // Fall back to old refresh token
    };
  } catch (error) {
    console.log(error);

    return {
      ...token,
      error: "RefreshAccessTokenError",
    };
  }
}
```

Απόσπασμα κώδικα 3.2: [...nextauth].js

Η συνάρτηση `'refreshAccessToken'` παίρνει ένα token ως όρισμα και επιχειρεί να ανανεώσει το token πρόσβασης χρησιμοποιώντας τον μηχανισμό ανανέωσης token του Spotify API. Η συνάρτηση ξεκινά με την κατασκευή μίας διεύθυνσης URL για το τελικό σημείο (endpoint) του token του Spotify API χρησιμοποιώντας το παρεχόμενο `'refreshToken'`. Ο κατασκευαστής `'URLSearchParams'` χρησιμοποιείται για τη δημιουργία των παραμέτρων ερωτήματος (query) για τη διεύθυνση URL, συμπεριλαμβανομένου του `'client_id'`, του `'client_secret'`, του `'grant_type'` και του `'refresh_token'`.

Η συνάρτηση χρησιμοποιεί τη συνάρτηση ανάκτησης (fetch function) για να κάνει ένα αίτημα POST στο URL που κατασκευάστηκε όπου αποστέλλεται και το πραγματικό αίτημα ανανέωσης token στο Spotify API. Στη συνέχεια, η συνάρτηση αναμένει την απάντηση και διαβάζει το σώμα απόκρισης ως JSON χρησιμοποιώντας την `'response.json'`, το οποίο πρέπει να περιέχει τα ανανεωμένα tokens που επιστρέφονται από το Spotify API.

Έπειτα, γίνεται έλεγχος εάν η κατάσταση απόκρισης δεν είναι εντάξει `'!response.ok'`. Εάν η κατάσταση απόκρισης υποδεικνύει σφάλμα, εμφανίζει σφάλμα χρησιμοποιώντας τα δεδομένα του `'refreshedTokens'`. Αυτός ο χειρισμός σφαλμάτων είναι χρήσιμος για τον χειρισμό περιπτώσεων όπου το ίδιο το αίτημα ανανέωσης token αποτυγχάνει.

Εάν η απόκριση είναι επιτυχής, η συνάρτηση κατασκευάζει ένα νέο αντικείμενο token με το ενημερωμένο token πρόσβασης και τον χρόνο λήξης του token πρόσβασης και λαμβάνεται από το `'refreshedTokens.access_token'` και ο χρόνος λήξης υπολογίζεται χρησιμοποιώντας την τρέχουσα ώρα συν την τιμή του `'refreshedTokens.expires_in'` σε δευτερόλεπτα.

Το `'refreshToken'` ενημερώνεται στο νέο token ανανέωσης που ελήφθη, εάν είναι διαθέσιμο. Εάν το token ανανέωσης δεν επιστραφεί στην απόκριση, δηλαδή το `'refreshedTokens.refresh_token'` είναι ψευδές, επιστρέφει στο υπάρχον token ανανέωσης που είναι αποθηκευμένο στο αντικείμενο token.

Εάν παρουσιαστεί σφάλμα κατά τη διαδικασία ανανέωσης του token, για παράδειγμα, λόγω προβλημάτων δικτύου ή μη έγκυρου token ανανέωσης, η συνάρτηση εντοπίζει το σφάλμα, το καταγράφει και επιστρέφει το αρχικό αντικείμενο token με μια πρόσθετη ιδιότητα `'error'` που έχει οριστεί σε `'RefreshAccessTokenError'`.

```

export default NextAuth({
  // Configure one or more authentication providers
  providers: [
    SpotifyProvider({
      clientId: process.env.SPOTIFY_CLIENT_ID,
      clientSecret: process.env.SPOTIFY_CLIENT_SECRET,
      authorization:
        "https://accounts.spotify.com/authorize?scope=user-read-email,playlist-read-private",
    }),
    // ...add more providers here
  ],
  secret: process.env.JWT_SECRET,
  pages: {
    signIn: "login",
  },
  callbacks: {
    async jwt({ token, user, account }) {
      // Initial sign in
      if (account && user) {
        return {
          accessToken: account.access_token,
          accessTokenExpires: Date.now() + account.expires_in * 1000,
          refreshToken: account.refresh_token,
          user,
        };
      }

      // Return previous token if the access token has not expired yet
      if (Date.now() < token.accessTokenExpires) {
        return token;
      }

      // Access token has expired, try to update it
      return refreshAccessToken(token);
    },
    async session({ session, token }) {
      session.user = token.user;
      session.accessToken = token.accessToken;
      session.error = token.error;

      return session;
    },
  },
});

```

Απόσπασμα κώδικα 3.3: [...nextauth].js

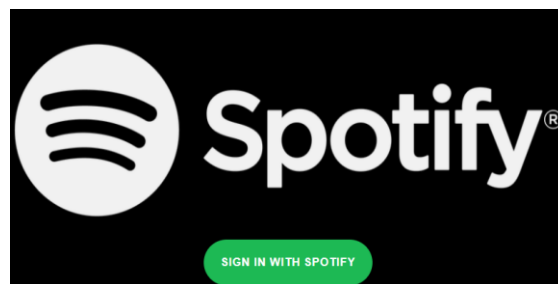
Το ‘SpotifyProvider’ έχει ρυθμιστεί ως πάροχος αυθεντικοποίησης και καθορίζει το ‘clientId’, το ‘clientSecret’ και το URL αυθεντικοποίησης. Αυτός ο πάροχος επιτρέπει στους χρήστες να συνδεθούν με τους λογαριασμούς τους στο Spotify.

Στο ‘secret’ έχει οριστεί το ‘process.env.JWT_SECRET’ και είναι το μυστικό που χρησιμοποιείται για την υπογραφή JSON Web Tokens (JWT) για τη διαχείριση συνεδριών.

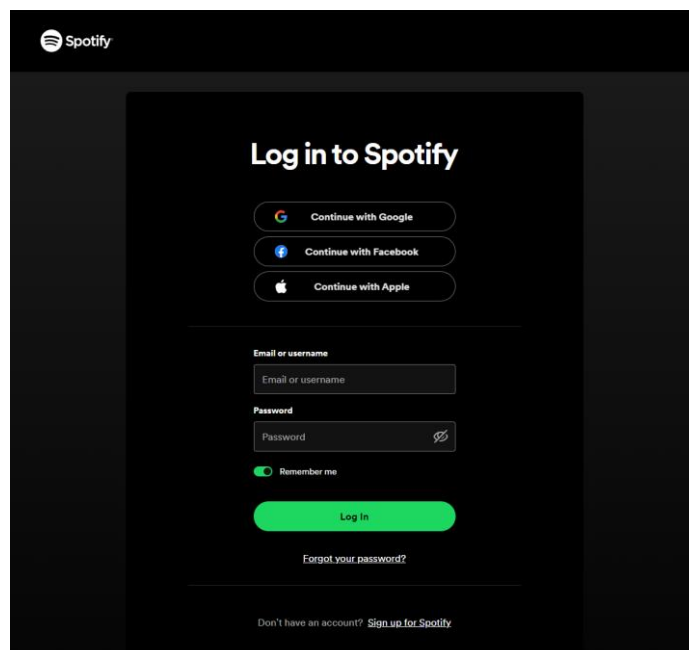
Το ‘pages’ property χρησιμοποιείται για την προσαρμογή των σελίδων αυθεντικοποίησης. Συγκεκριμένα, ορίζει τη σελίδα σύνδεσης σε ‘login’. Αυτό σημαίνει ότι όταν οι χρήστες πρέπει να συνδεθούν, θα κατευθύνονται στη σελίδα ‘login’.

Η επανάκληση (callback) ‘jwt’ χρησιμοποιείται για τον χειρισμό των JWT (JSON Web Tokens) που δημιουργούνται κατά τη διαδικασία αυθεντικοποίησης. Όταν πραγματοποιείται μια αρχική είσοδος παρέχονται και ο λογαριασμός και ο χρήστης, επιστρέφει ένα αντικείμενο που περιλαμβάνει το ‘accessToken’, το ‘accessTokenExpires’, το ‘refreshToken’ και ο χρήστης. Αυτό ουσιαστικά ρυθμίζει τη συνεδρία του χρήστη με τα tokens πρόσβασης στο Spotify. Εάν το token πρόσβασης δεν έχει λήξει ακόμη, απλώς επιστρέφει το υπάρχον token. Εάν το token πρόσβασης έχει λήξει, επιχειρεί να το ανανεώσει καλώντας τη συνάρτηση refreshAccessToken.

Η επανάκληση ‘session’ χρησιμοποιείται για την προσαρμογή του αντικειμένου συνεδρίας που χρησιμοποιεί το NextAuth. Προσθέτει ιδιότητες στο αντικείμενο συνεδρίας, όπως χρήστη, ‘accessToken’ και σφάλμα με βάση τις τιμές στο αντικείμενο token. Συνοπτικά, αυτή η διαμόρφωση ρυθμίζει το Spotify ως πάροχο αυθεντικοποίησης, ορίζει προσαρμοσμένες σελίδες για αυθεντικοποίηση και καθορίζει επανακλήσεις για τη διαχείριση της δημιουργίας JWT και της διαχείρισης συνεδριών. Οι επανακλήσεις διασφαλίζουν ότι η διαχείριση των tokens πρόσβασης γίνεται σωστά και το αντικείμενο συνεδρίας συμπληρώνεται με πληροφορίες χρήστη και κατάσταση σφάλματος.



Εικόνα 3.1: Σελίδα σύνδεσης στην εφαρμογή



Εικόνα 3.2: Σελίδα σύνδεσης στην εφαρμογή

Αυτήν είναι η σελίδα σύνδεσης στην εφαρμογή, στην οποία ο χρήστης συνδέεται με τον λογαριασμό του στο Spotify.

```
function Login({ providers }) {
  const { data: session } = useSession();
  const router = useRouter();

  useEffect(() => {
    if (session) {
      router.push("/");
    }
  }, [session]);

  if (session) return <Loader />;
}
```

Απόσπασμα κώδικα 3.4: *login.js*

Το άγκιστρο ‘useSession’ χρησιμοποιείται για τη λήψη των δεδομένων συνεδρίας του χρήστη και το άγκιστρο ‘useRouter’ επιτρέπει την αλλαγή στις διαδρομές μέσα στα Client components.

Το άγκιστρο ‘useEffect’ χρησιμοποιείται για την αυτόματη ανακατεύθυνση του χρήστη στην αρχική σελίδα ("/") εάν είναι ήδη συνδεδεμένος, δηλαδή έχει ενεργή περίοδο σύνδεσης. Εάν υπάρχει ενεργή περίοδος σύνδεσης τότε κάνει render το component Loader το οποίο θα εμφανίζεται στην οθόνη του χρήστη κατά τη διαδικασία επικύρωσης της συνεδρίας.



Εικόνα 3.6: *Loader της εφαρμογής*

3.3 Atoms

```
import { atom } from "recoil";

export const playState = atom({
  key: "playState",
  default: false,
});

export const playingTrackState = atom({
  key: "playingTrackState",
  default: "",
});
```

Απόσπασμα κώδικα 3.5: *playerAtom.js*

Τα Recoil Atoms είναι μονάδες κατάστασης όπου τα στοιχεία μπορούν να εγγραφούν και να χρησιμοποιηθούν για τη διαχείριση και την κοινή χρήση κατάστασης σε όλη την εφαρμογή.

Το 'playState' atom αντιπροσωπεύει την κατάσταση του εάν ένα κομμάτι παίζει αυτήν τη στιγμή ή όχι. Χρησιμοποιεί τη συνάρτηση atom από το Recoil, το οποίο παίρνει ένα αντικείμενο με ιδιότητες για τη διαμόρφωση του atom. Η προεπιλεγμένη τιμή του ατόμου σε αυτήν την περίπτωση είναι false, υποδεικνύοντας ότι κανένα κομμάτι δεν παίζει αρχικά.

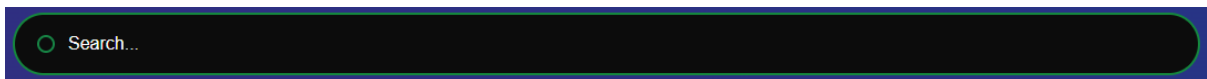
Το 'playingTrackState' atom αντιπροσωπεύει την κατάσταση του URI του τρέχοντος κομματιού (Uniform Resource Identifier). Χρησιμοποιεί τη συνάρτηση atom με παρόμοιες ιδιότητες με το playState atom. Η προεπιλεγμένη τιμή του atom σε αυτήν την περίπτωση είναι μια κενή συμβολοσειρά, που υποδεικνύει ότι δεν αναπαράγεται κάποιο κομμάτι στην αρχή.

Αυτά τα άτομα μπορούν να χρησιμοποιηθούν για τη διαχείριση και την κοινή χρήση της κατάστασης που σχετίζεται με την αναπαραγωγή κομματιού και τις πληροφορίες του κομματιού που αναπαράγεται κάθε στιγμή σε διάφορα components της εφαρμογής. Τα στοιχεία μπορούν να εγγραφούν σε αυτά τα atoms χρησιμοποιώντας Recoil άγκιστρα, όπως 'useRecoilValue' και 'useRecoilState' για ανάγνωση και ενημέρωση της κατάστασης αντίστοιχα.

3.4 Components της Εφαρμογής

Μια εφαρμογή είναι ένα πρόγραμμα που έχει σχεδιαστεί για να εκτελεί συγκεκριμένες εργασίες ή λειτουργίες για τους χρήστες. Τα components μιας εφαρμογής μπορεί να διαφέρουν ανάλογα με την πολυπλοκότητα και τον σκοπό της, αλλά γενικά, μια εφαρμογή αποτελείται από πολλά βασικά στοιχεία. Αυτά τα components μπορεί να διαφέρουν ανάλογα με τον τύπο της εφαρμογής (π.χ. εφαρμογή για κινητά, εφαρμογή για υπολογιστές, εφαρμογή ιστού) και τις συγκεκριμένες απαιτήσεις της. Οι προγραμματιστές σχεδιάζουν και ενσωματώνουν προσεκτικά αυτά τα components για να δημιουργήσουν λειτουργικές και φιλικές προς το χρήστη εφαρμογές.

3.4.1 Search



Εικόνα 3.8: Search component της εφαρμογής

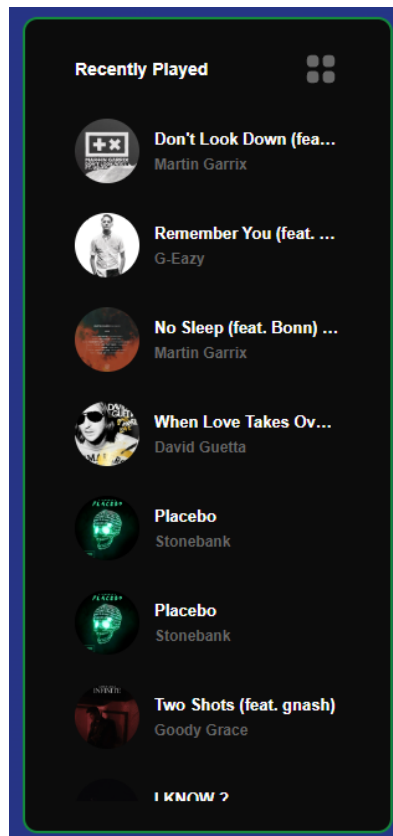
```
function Search({ search, setSearch }) {
  return (
    <div className="max-w-[1150px] bg-[#0c0c0c] rounded-full overflow-hidden border-2 border-[#15883e] p-1.5 px-5 pr-8 flex items-center mx-4">
      <div className="h-4 w-4 rounded-full border-2 flex-shrink-0 animate-pulse border-[#20CB67]" />
      <input
        type="text"
        value={search}
        onChange={(e) => setSearch(e.target.value)}
        className="bg-[#0c0c0c] text-white border-none lg:w-full focus:ring-0 outline-none placeholder-[#FAFAFA] text-s"
        placeholder="Search..."
      />
    </div>
  );
}

export default Search;
```

Απόσπασμα κώδικα 3.6: Search component

Το Search Component είναι η μπάρα αναζήτησης της εφαρμογής και δέχεται δύο properties το 'search' και το 'setSearch' τα οποία χρησιμοποιούνται για τον έλεγχο της τιμής του πεδίου εισαγωγής και την ενημέρωση του όταν ο χρήστης πληκτρολογεί. Το 'value = {search}' δεσμεύει την τιμή του πεδίου εισαγωγής με το 'search' καθιστώντας την είσοδο ελεγχόμενη. Το 'onChange={(e) => setSearch(e.target.value)}' ρυθμίζει έναν χειριστή συμβάντων (event handler) ο οποίος ενημερώνει το 'search' property όποτε αλλάζει η τιμή εισόδου.

3.4.2 Recently Played



Εικόνα 3.10: *RecentlyPlayed* component της εφαρμογής

Αυτό είναι από το *Recently Played* component το οποίο δείχνει τα 20 τελευταία τραγούδια που έχει ακούσει ο χρήστης με αυτόν τον λογαριασμό.

```
function RecentlyPlayed({ track, chooseTrack }) {
  const [play, setPlay] = useRecoilState(playState);
  const [playingTrack, setPlayingTrack] = useRecoilState(playingTrackState);

  const handlePlay = () => {
    chooseTrack(track);

    if (track.uri === playingTrack.uri) {
      setPlay(!play);
    }
  };
}
```

Απόσπασμα κώδικα 3.7: *RecentlyPlayed* component

Το component εισάγει το άγκιστρο ‘useRecoilState’ από το Recoil και δύο Recoil atoms ‘playingTrackState’ και ‘playState’ από το playerAtom. Αυτά τα atoms διατηρούν την κατάσταση που σχετίζεται με το τρέχον κομμάτι που παίζει και την κατάσταση αναπαραγωγής/παύσης.

Η συνάρτηση ‘handlePlay’ καλείται όταν ο χρήστης κάνει κλικ στο κουμπί αναπαραγωγής/παύσης, τότε καλείται η συνάρτηση ‘chooseTrack’ με παράμετρο ένα κομμάτι (track). Ελέγχει εάν το URI του κομματιού στο οποίο έγινε κλικ ταιριάζει με το URI του

κομματιού που αναπαράγεται εκείνη τη στιγμή και, εάν ταιριάζουν, αλλάζει την κατάσταση αναπαραγωγής είτε σε παύση ή είτε σε αναπαραγωγή του κομματιού.

3.4.3 Player



Εικόνα 3.12: Player component της εφαρμογής

Στον Player της εφαρμογής ο χρήστης στα αριστερά βλέπει πληροφορίες για το τραγούδι που παίζει κάθε στιγμή όπως το όνομα του τραγουδιού, τον καλλιτέχνη και μπορεί να κάνει like στο τραγούδι. Στο κέντρο υπάρχουν οι επιλογές του play/pause, να επιλέξει ο χρήστης το προηγούμενο ή το επόμενο τραγούδι και επίσης μπορεί να δει και να αλλάξει το σημείο του τραγουδιού. Στα δεξιά, υπάρχει ο έλεγχος της έντασης και η επιλογή από ποια συσκευή να αναπαράγεται το τραγούδι.

```
function Player({ accessToken, trackUri }) {
  const [play, setPlay] = useRecoilState(playState);
  const [playingTrack, setPlayingTrack] = useRecoilState(playingTrackState);

  useEffect(() => {
    if (trackUri) {
      setPlay(true);
    }
  }, [trackUri]);

  if (!accessToken) return null;
}
```

Απόσπασμα κώδικα 3.8: Player component

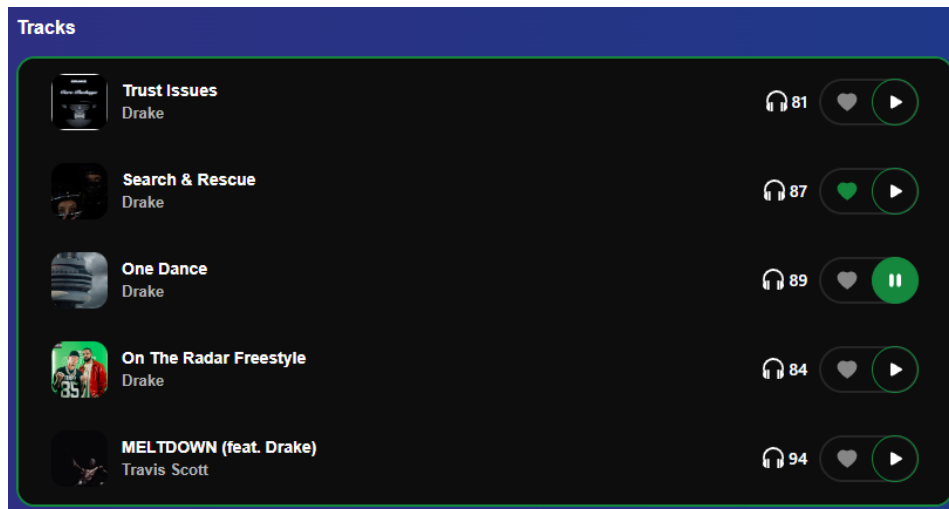
Το component αυτό εισάγει τα Recoil atoms ‘playState’ και ‘playingTrackState’, το SpotifyPlayer από το react-spotify-web-playback [20] και τα άγκιστρα ‘useRecoilState’ και ‘useEffect’ από τη React.

Το component χρησιμοποιεί το ‘useRecoilState’ άγκιστρο για τη διαχείριση της κατάστασης του ‘play’ και ‘playingTrack’ από το playerAtom.

Το άγκιστρο ‘useEffect’ χρησιμοποιείται για την αυτόματη έναρξη της αναπαραγωγής του κομματιού όταν αλλάζει το trackUri. Εάν παρέχεται ένα trackUri, ορίζει την κατάσταση αναπαραγωγής σε true, το οποίο ξεκινά την αναπαραγωγή.

Επίσης, αν δεν παρέχεται accessToken, το component επιστρέφει null, δηλαδή δεν κάνει render ο player.

3.4.4 Track



Εικόνα 3.14: Track component της εφαρμογής

Το component αυτό ονομάζεται Track και σε αυτό εμφανίζεται στα αριστερά η εικόνα του εξωφύλλου του άλμπουμ, ο τίτλος του κομματιού και του καλλιτέχνη. Στα δεξιά εμφανίζεται η δημοτικότητα του κομματιού, ένα εικονίδιο καρδιάς το οποίο δηλώνει αν ο το τραγούδι αρέσει στον χρήστη. Όταν ο χρήστης πατήσει το εικονίδιο, αυτό γίνεται πράσινο και με αυτόν τον τρόπο δηλώνει ότι του αρέσει. Επίσης, εμφανίζει ένα κουμπί αναπαραγωγής/παύσης που εμφανίζεται με βάση εάν το κομμάτι είναι αυτό που παίζεται εκείνη τη στιγμή και εάν είναι σε κατάσταση αναπαραγωγής. Το κουμπί εναλλάσσεται μεταξύ των εικονιδίων αναπαραγωγής και παύσης

```
function Track({ track, chooseTrack }) {
  const [hasLiked, setHasLiked] = useState(false);
  const [play, setPlay] = useRecoilState(playState);
  const [playingTrack, setPlayingTrack] = useRecoilState(playingTrackState);

  const handlePlay = () => {
    chooseTrack(track);

    if (track.uri === playingTrack.uri) {
      setPlay(!play);
    }
  };
}
```

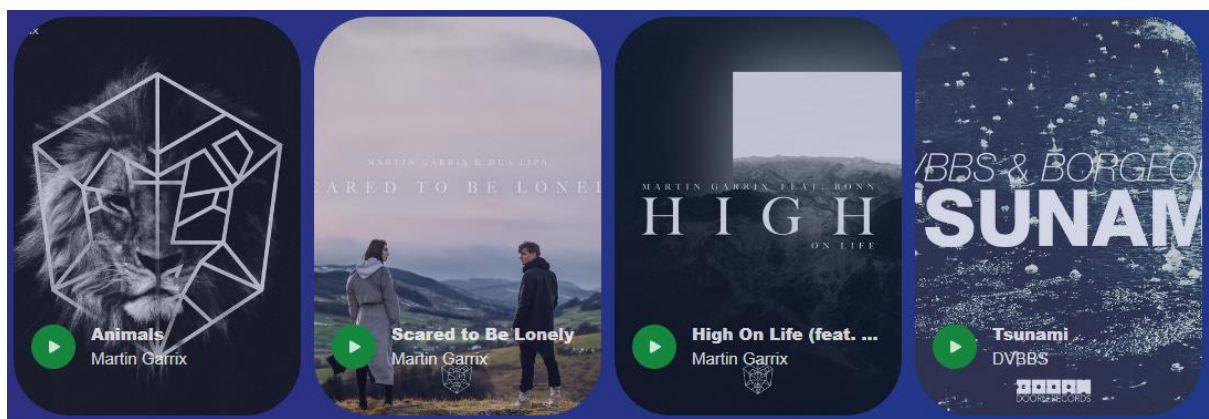
Απόσπασμα κώδικα 3.9: Track component

Τα άγκιστρα useState και useRecoilState εισάγονται από την React για τη διαχείριση της κατάστασης του component. Τα 'playingTrackState' και 'playState' atoms εισάγονται για τη διαχείριση πληροφοριών αναπαραγωγής και κομματιού. Το component παίρνει δύο props, το 'track' που είναι οι πληροφορίες του τραγουδιού και το 'chooseTrack' που είναι μία συνάρτηση που χειρίζεται την επιλογή των τραγουδιών.

Το component χρησιμοποιεί το άγκιστρο ‘useState’ για τη διαχείριση της κατάστασης του ‘hasLiked’, το οποίο αντιπροσωπεύει εάν το κομμάτι άρεσε στον χρήστη. Χρησιμοποιεί επίσης το άγκιστρο ‘useRecoilState’ για τη διαχείριση της κατάστασης του ‘play’ και ‘playingTrack’ από το playerAtom.

Η συνάρτηση ‘handlePlay’ καλείται όταν ο χρήστης κάνει κλικ στο κουμπί αναπαραγωγής/παύσης. Τότε καλείται η συνάρτηση ‘chooseTrack’ με παράμετρο ένα κομμάτι (track). Ελέγχεται εάν το URI του κομματιού στο οποίο έγινε κλικ ταιριάζει με το URI του κομματιού που αναπαράγεται εκείνη τη στιγμή και, εάν ταιριάζουν, αλλάζει την κατάσταση αναπαραγωγής είτε σε παύση ή είτε σε αναπαραγωγή του κομματιού.

3.4.5 Poster



Εικόνα 3.16: Poster component της εφαρμογής

Το component αυτό ονομάζεται Poster και όταν δεν υπάρχει κάτι στην αναζήτηση τότε δείχνει τα εξώφυλλα από τέσσερα προτεινόμενα άλμπουμ τραγουδιστών στα οποία μπορεί να κάνει κλικ ο χρήστης, τα ονόματά τους και τα ονόματα των καλλιτεχνών, ενώ όταν αρχίζει να πληκτρολογεί ο χρήστης στην αναζήτηση, το component αλλάζει δυναμικά και δείχνει τα πιο σχετικά κομμάτια σχετικά με την αναζήτηση. Επίσης, εμφανίζεται ένα κουμπί αναπαραγωγής/παύσης στο κάτω μέρος της αφίσας. Το κουμπί εναλλάσσεται μεταξύ των εικονιδίων αναπαραγωγής και παύσης με βάση την κατάσταση αναπαραγωγής του κομματιού.

```
function Poster({ track, chooseTrack }) {
  const [play, setPlay] = useRecoilState(playState);
  const [playingTrack, setPlayingTrack] = useRecoilState(playingTrackState);

  const handlePlay = () => {
    chooseTrack(track);

    if (track.uri === playingTrack.uri) {
      setPlay(!play);
    }
  };
}
```

Απόσπασμα κώδικα 3.10: Poster component

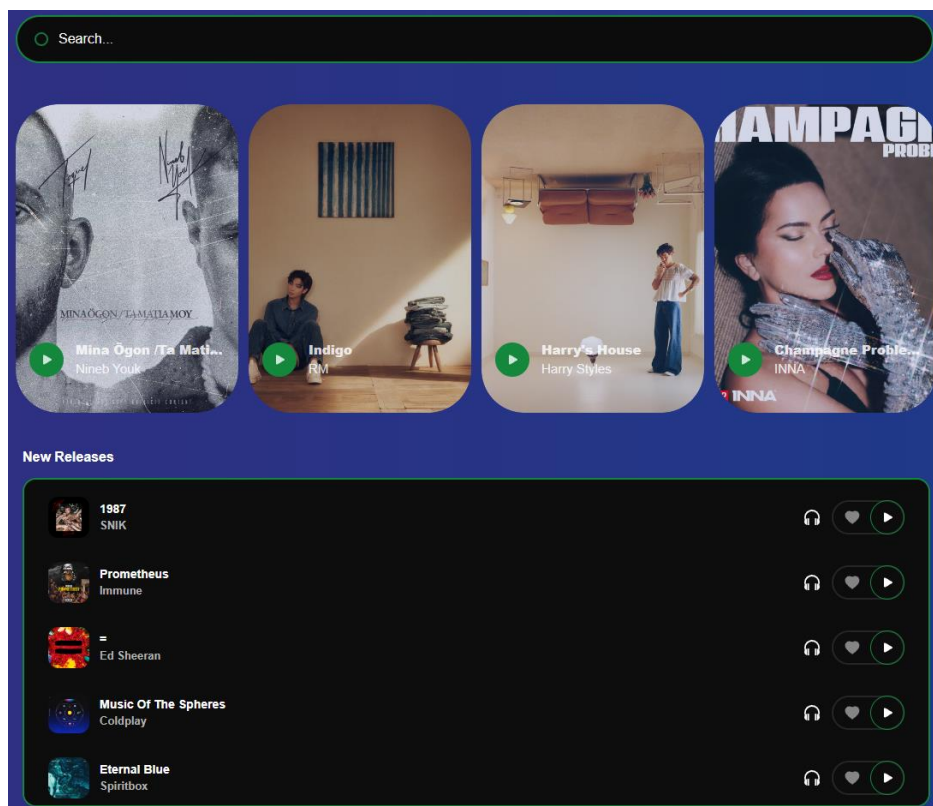
Το στοιχείο χρησιμοποιεί το άγκιστρο `useRecoilState` για τη διαχείριση της κατάστασης του `‘play’` και `‘playingTrack’` από το `playerAtom`.

Η συνάρτηση `‘handlePlay’` καλείται όταν ο χρήστης κάνει κλικ στο κουμπί αναπαραγωγής/παύσης, τότε καλείται η συνάρτηση `‘chooseTrack’` με παράμετρο ένα κομμάτι (`track`). Ελέγχει εάν το URI του κομματιού στο οποίο έγινε κλικ ταιριάζει με το URI του κομματιού που αναπαράγεται εκείνη τη στιγμή, εάν ταιριάζουν, αλλάζει την κατάσταση αναπαραγωγής είτε σε παύση ή είτε σε αναπαραγωγή του κομματιού.

Το component λαμβάνει δύο props το `‘track’` και το `‘chooseTrack’`. Μέσα στη συνάρτηση του component αρχικοποιούνται οι καταστάσεις `‘play’` και `‘playingTrack’` χρησιμοποιώντας το `‘useRecoilState’` άγκιστρο και τα Recoil atoms.

Η συνάρτηση `‘handlePlay’` έχει οριστεί για να χειρίζεται την αναπαραγωγή/παύση όταν ο χρήστης κάνει κλικ στο poster ενός τραγουδιού ή άλμπουμ. Όταν ο χρήστης κλικάρει, καλείται η συνάρτηση `‘chooseTrack’` με όρισμα το τρέχον κομμάτι και ενημερώνει την κατάσταση για να ορίσει το επιλεγμένο κομμάτι ως το τρέχον κομμάτι που παίζει. Στη συνέχεια ελέγχεται αν το URI του κομματιού στο οποίο έγινε κλικ `‘track.uri’` ταιριάζει με το URI του τρέχοντος κομματιού `‘playingTrack.uri’`. Εάν ταιριάζουν, αλλάζει την κατάσταση αναπαραγωγής χρησιμοποιώντας το `‘setPlay(!play)’` για εναλλαγή μεταξύ των καταστάσεων αναπαραγωγής και παύσης.

3.4.6 Body



Εικόνα 3.18: Body component της εφαρμογής

Το component Body περιλαμβάνει μια γραμμή αναζήτησης που χρησιμοποιεί το Search component, από κάτω εμφανίζει ένα πλέγμα από εξώφυλλα άλμπουμ που είτε είναι αποτελέσματα αναζήτησης είτε νέες κυκλοφορίες και χρησιμοποιεί το Poster component. Τέλος, εμφανίζει μια λίστα κομματιών τα οποία και αυτά χωρίζονται σε δύο κατηγορίες ανάλογα με το αν είναι αποτελέσματα αναζήτησης του χρήστη ή νέες κυκλοφορίες και χρησιμοποιεί το Track component.

```
function Body({ spotifyApi, chooseTrack }) {  
  const { data: session } = useSession();  
  const { accessToken } = session;  
  const [search, setSearch] = useState("");  
  const [searchResults, setSearchResults] = useState([]);  
  const [newReleases, setNewReleases] = useState([]);  
  
  useEffect(() => {  
    if (!accessToken) return;  
    spotifyApi.setAccessToken(accessToken);  
  }, [accessToken]);  
}
```

Απόσπασμα κώδικα 3.11: Body component

Χρησιμοποιεί το ‘useSession’ άγκιστρο για να λάβει τα δεδομένα περιόδου σύνδεσης του χρήστη, συμπεριλαμβανομένου του ‘accessToken’ το οποίο παρέχει την εξουσιοδότηση για το Spotify API. Επίσης χρησιμοποιεί το ‘useState’ άγκιστρο για διαχειριστεί την αναζήτηση, τον πίνακα αποτελεσμάτων αναζήτησης (κομμάτια που ταιριάζουν με την αναζήτηση) και τον πίνακα newReleases (κομμάτια που κυκλοφόρησαν πρόσφατα).

Το πρώτο ‘useEffect’ άγκιστρο ορίζει το token πρόσβασης του Spotify όταν αυτό αλλάζει, δηλαδή όταν ο χρήστης συνδέεται ή αποσυνδέεται.

```

useEffect(() => {
  if (!search) return setSearchResults([]);
  if (!accessToken) return;

  let cancel = false;

  spotifyApi.searchTracks(search).then((res) => {
    if (cancel) return;
    setSearchResults(
      res.body.tracks.items.map((track) => {
        return {
          id: track.id,
          artist: track.artists[0].name,
          title: track.name,
          uri: track.uri,
          albumUrl: track.album.images[0].url,
          popularity: track.popularity,
        };
      })
    );
  });

  return () => (cancel = true);
}, [search, accessToken]);

// New Releases...
useEffect(() => {
  if (!accessToken) return;

  spotifyApi.getNewReleases().then((res) => {
    setNewReleases(
      res.body.albums.items.map((track) => {
        return {
          id: track.id,
          artist: track.artists[0].name,
          title: track.name,
          uri: track.uri,
          albumUrl: track.images[0].url,
        };
      })
    );
  });
}, [accessToken]);

```

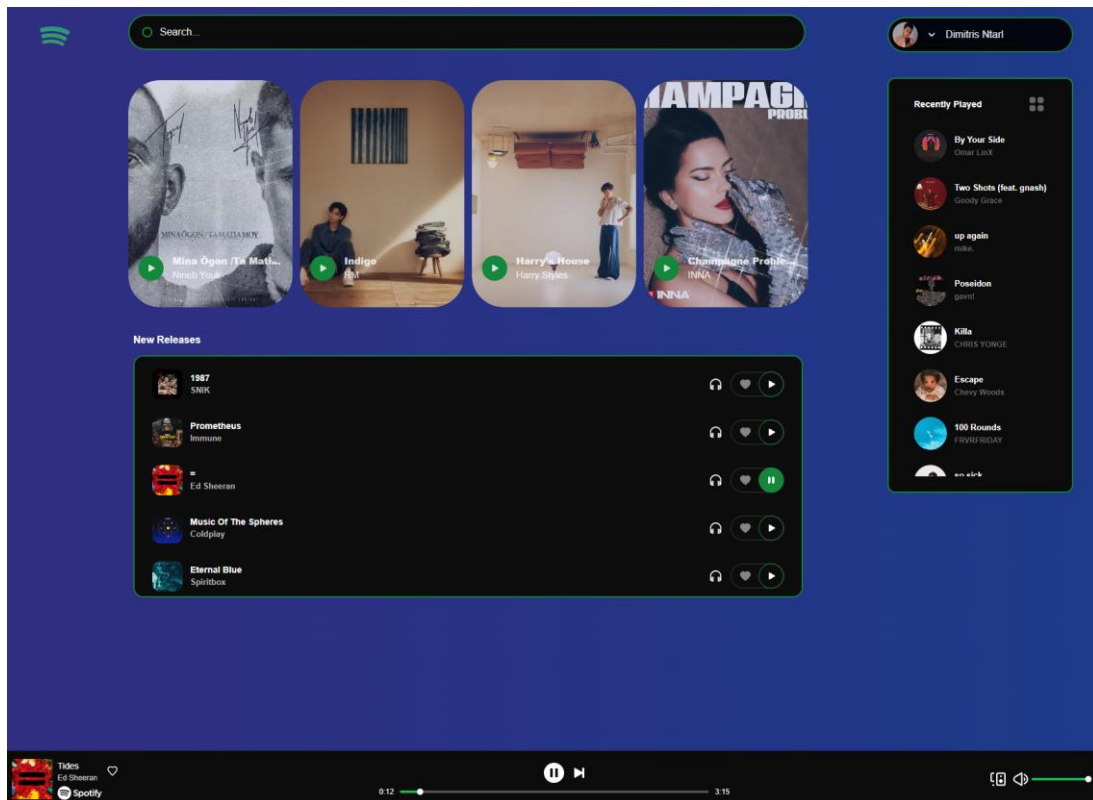
Απόσπασμα κώδικα 3.12: *Body component*

Το δεύτερο άγκιστρο ‘useEffect’ διαχειρίζεται τη λειτουργία αναζήτησης, εκτελεί μια Spotify API αναζήτηση για κομμάτια με βάση την αναζήτηση του χρήστη. Τα αποτελέσματα αναζήτησης μετατρέπονται σε μια σειρά αντικειμένων, συμπεριλαμβανομένων πληροφοριών όπως ID, καλλιτέχνης, τίτλος, URI, διεύθυνση URL άλμπουμ και δημοτικότητα του κομματιού. Επίσης, επιστρέφει μια συνάρτηση εκκαθάρισης για να αποτρέψει την κατάσταση ενημέρωσης εάν το στοιχείο αποπροσαρτηθεί πριν ολοκληρωθεί η κλήση API.

Το τρίτο άγκιστρο ‘useEffect’ ανακτά νέα δεδομένα που κυκλοφορούν από το Spotify API και τα μετατρέπει σε μια σειρά αντικειμένων, παρόμοια με τα αποτελέσματα αναζήτησης.

Συνολικά, το component Body συνδυάζει τη διαχείριση περιόδων σύνδεσης χρήστη, τη λειτουργικότητα αναζήτησης και την εμφάνιση κομματιών για να παρέχει μια ολοκληρωμένη διεπαφή αναπαραγωγής μουσικής. Το στοιχείο εμφανίζει αποτελέσματα αναζήτησης, νέες κυκλοφορίες και μια λίστα κομματιών με βάση την αλληλεπίδραση των χρηστών.

3.4.7 Dashboard



Εικόνα 3.21: Dashboard component της εφαρμογής

Το component αυτό είναι το Dashboard και είναι υπεύθυνο για το rendering του κύριου περιεχομένου της εφαρμογής συμπεριλαμβανομένου του body, του player και του δεξιού πάνελ. Το στοιχείο εισάγει διάφορες εξαρτήσεις και components απαραίτητα για τη λειτουργικότητά του, όπως το `playingTrackState` από το Recoil atom, το `SpotifyWebApi` από τη βιβλιοθήκη `spotify-web-api-node`, το `useSession` από `next-auth/react` και τα React άγκιστρα `useEffect` και `useState`.

```

const spotifyApi = new SpotifyWebApi({
  clientId: process.env.SPOTIFY_CLIENT_ID,
});

function Dashboard() {
  const { data: session } = useSession();
  const { accessToken } = session;

  const [playingTrack, setPlayingTrack] = useRecoilState(playingTrackState);
  const [showPlayer, setShowPlayer] = useState(false);

  useEffect(() => {
    setShowPlayer(true);
  }, []);

  const chooseTrack = (track) => {
    setPlayingTrack(track);
  };
}

```

Απόσπασμα κώδικα 3.13: *Dashboard component*

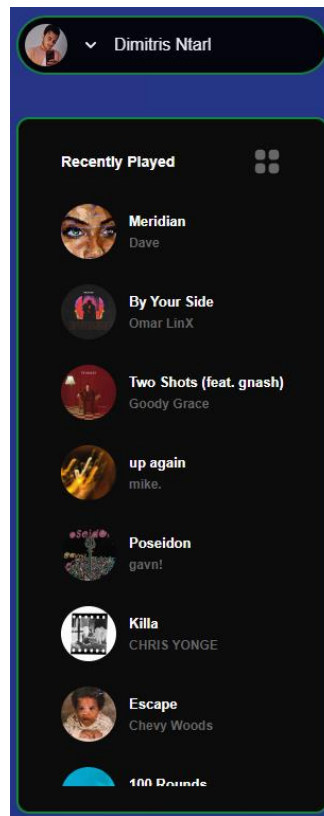
Δημιουργείται ένα instance του `SpotifyWebApi` χρησιμοποιώντας το `SPOTIFY_CLIENT_ID` από τις μεταβλητές περιβάλλοντος. Αυτό το instance θα χρησιμοποιηθεί για την αλληλεπίδραση με το Spotify Web API.

Το `useSession` άγκιστρο χρησιμοποιείται για τη λήψη των δεδομένων περιόδου σύνδεσης χρήστη, τα οποία περιλαμβάνουν το `accessToken` που απαιτείται για την αυθεντικοποίηση αιτημάτων στο Spotify API.

Το άγκιστρο `useRecoilState` χρησιμοποιείται για τη διαχείριση της κατάστασης του τρέχοντος κομματιού που αναπαράγεται και μια σημαία boolean `showPlayer`, η οποία καθορίζει εάν ο player πρέπει να εμφανίζεται. Το άγκιστρο `useEffect` χρησιμοποιείται για να οριστεί η αρχική τιμή του `showPlayer` σε `true` κατά την προσάρτηση του στοιχείου, δηλαδή ο player θα εμφανίζεται από προεπιλογή.

Η συνάρτηση `chooseTrack` λαμβάνει ως παράμετρο ένα κομμάτι (`track`) και ορίζει το `track` ως τρέχον κομμάτι που παίζει χρησιμοποιώντας την συνάρτηση `setPlayingTrack`

3.4.8 Right

Εικόνα 3.23: *Right component της εφαρμογής*

Το component Right αποτελεί την δεξιά μεριά της εφαρμογής και περιλαμβάνει πληροφορίες για το προφίλ του χρήστη, τη δυνατότητα να κάνει αποσύνδεση και μία λίστα κομματιών που παίχτηκαν πρόσφατα, τα οποία είναι το Recently Played component

```
function Right({ chooseTrack, spotifyApi }) {
  const { data: session } = useSession();
  const { accessToken } = session;
  const [recentlyPlayed, setRecentlyPlayed] = useState([]);

  useEffect(() => {
    if (!accessToken) return;

    spotifyApi.getMyRecentlyPlayedTracks({ limit: 20 }).then((res) => {
      setRecentlyPlayed(
        res.body.items.map(({ track }) => {
          return {
            id: track.id,
            artist: track.artists[0].name,
            title: track.name,
            uri: track.uri,
            albumUrl: track.album.images[0].url,
          };
        })
      );
    });
  }, [accessToken]);
}
```

Απόσπασμα κώδικα 3.14: *Right component*

Το component χρησιμοποιεί το άγκιστρο ‘useSession’ για πρόσβαση στα δεδομένα περιόδου σύνδεσης του χρήστη, συμπεριλαμβανομένου του ονόματος και της εικόνας προφίλ του και επίσης εξάγεται το accessToken του χρήστη από τα δεδομένα της περιόδου σύνδεσης.

Το άγκιστρο ‘useEffect’ χρησιμοποιείται για τη λήψη των κομματιών που έπαιξε πρόσφατα ο χρήστης χρησιμοποιώντας το Spotify API. Τα κομμάτια ανακτώνται χρησιμοποιώντας το accessToken του χρήστη και στη συνέχεια, αποθηκεύονται σε μια μορφή που μπορεί να χρησιμοποιηθούν από το RecentlyPlayed component.

Κεφάλαιο 4

Deployment

Το Deployment αναφέρεται στη διαδικασία της εγκατάστασης και της λειτουργίας της εφαρμογής σε ένα περιβάλλον, ώστε να είναι διαθέσιμο στους χρήστες. Το Deployment είναι ένα σημαντικό και κρίσιμο κομμάτι στην ανάπτυξη μιας εφαρμογής, καθώς επιτρέπει τη χρήση της από τους τελικούς χρήστες. Επιπλέον, μπορεί να υπάρχουν αυτοματοποιημένες διαδικασίες για το Deployment που εξασφαλίζουν την αξιοπιστία και τη σταθερότητα. Τέτοιες αυτοματοποιημένες διαδικασίες ονομάζονται Continuous Deployment, όπου η εφαρμογή ενημερώνεται και διανέμεται στον Server παραγωγής όσο πιο συχνά γίνεται, εξασφαλίζοντας έτσι τις ταχύτερες βελτιώσεις και διορθώσεις.

Μια αποτελεσματική διαδικασία Deployment είναι σημαντική για την επιτυχημένη λειτουργία μιας εφαρμογής και την εξασφάλιση της υψηλής απόδοσης και αξιοπιστίας του συστήματος. Επίσης, επιτρέπει στις ομάδες ανάπτυξης να παρέχουν νέες λειτουργίες και βελτιώσεις στους χρήστες όσο το δυνατόν ταχύτερα και με μειωμένο κίνδυνο για τις υπηρεσίες που παρέχουν. Για να επιτύχει ένα αποτελεσματικό Deployment, συνήθως γίνεται χρήση εργαλείων όπως τα συστήματα διαχείρισης κώδικα (π.χ. Git, GitHub), οι αυτοματοποιημένες πλατφόρμες CI/CD (Continuous Integration/Continuous Deployment) και υπηρεσίες cloud για την ευκολία και την ασφάλεια της ανάπτυξης και του Deployment.

4.1 Vercel

Η Vercel είναι μια πλατφόρμα cloud που ειδικεύεται στο να επιτρέπει στους προγραμματιστές να αναπτύσσουν και να φιλοξενούν Web εφαρμογές, ιστοτόπους και λειτουργίες χωρίς διακομιστή με ευκολία. Παρέχει μια απρόσκοπτη ροή εργασίας για προγραμματιστές που ασχολούνται με το Front-end κομμάτι, καθιστώντας εύκολη τη λήψη του codebase και την ανάπτυξή του σε ένα ζωντανό περιβάλλον.

Η πλατφόρμα είναι γνωστή για τις δυνατότητες φιλοξενίας στατικής ιστοσελίδας. Μπορεί να εξυπηρετήσει στατικά στοιχεία όπως αρχεία HTML, CSS και JavaScript με χαμηλό χρόνο αδράνειας και υψηλή απόδοση. Αυτό είναι ιδιαίτερα χρήσιμο για εφαρμογές μιας σελίδας και στατικών ιστοσελίδων. Η Vercel επιτρέπει στους προγραμματιστές να δημιουργούν λειτουργίες χωρίς διακομιστή που μπορούν να εκτελεστούν από την πλευρά του διακομιστή. Αυτές οι λειτουργίες επιτρέπουν στους προγραμματιστές να εκφορτώνουν ορισμένες εργασίες από την πλευρά του πελάτη, παρέχοντας βελτιωμένη ασφάλεια και επεκτασιμότητα. Επίσης, υποστηρίζει συνεχή ανάπτυξη, πράγμα που σημαίνει ότι κάθε φορά που προωθούνται αλλαγές στο χώρο αποθήκευσης του κώδικα (π.χ. GitHub, GitLab), η Vercel θα δημιουργήσει και θα αναπτύξει αυτόματα την ενημερωμένη εφαρμογή.

Γίνεται χρήση ενός παγκοσμίου CDN (Content Delivery Network) για τη διανομή περιεχομένου και τη διασφάλιση της γρήγορης φόρτωσης της εφαρμογής για τους χρήστες σε όλο τον κόσμο. Ακόμα, οι προγραμματιστές μπορούν εύκολα να συσχετίσουν

προσαρμοσμένα domains με τις εφαρμογές τους που φιλοξενούνται στη πλατφόρμα και παρέχονται αυτόματα πιστοποιητικά SSL τα οποία διαχειρίζονται την ασφάλεια των συνδέσεων. Επίσης, παρέχει εργαλεία συνεργασίας και δυνατότητα συνεργασίας σε ομάδες, επιτρέποντας σε πολλούς προγραμματιστές να συνεργάζονται απρόσκοπτα στο ίδιο έργο αλλά προσφέρει και ενσωματωμένες δυνατότητες ανάλυσης και παρακολούθησης, παρέχοντας πληροφορίες για την απόδοση και τη χρήση της εφαρμογής.

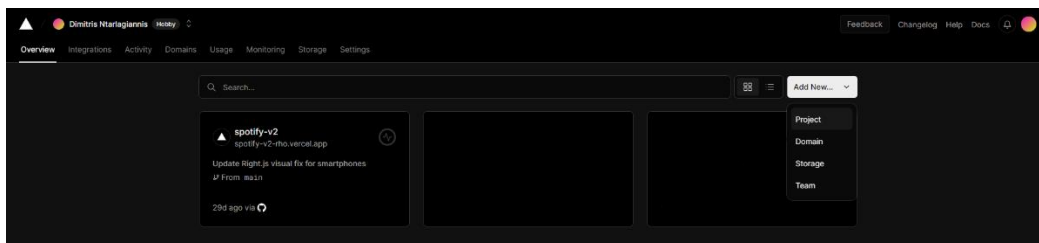
Τέλος, η Vercel είναι μια δημοφιλής επιλογή μεταξύ Front-end προγραμματιστών λόγω της ευκολίας χρήσης, της ταχύτητας και των φιλικών προς τους προγραμματιστές λειτουργιών. Υποστηρίζει διάφορα frameworks και τεχνολογίες, καθιστώντας το ευέλικτο και συμβατό με ένα ευρύ φάσμα έργων [21].



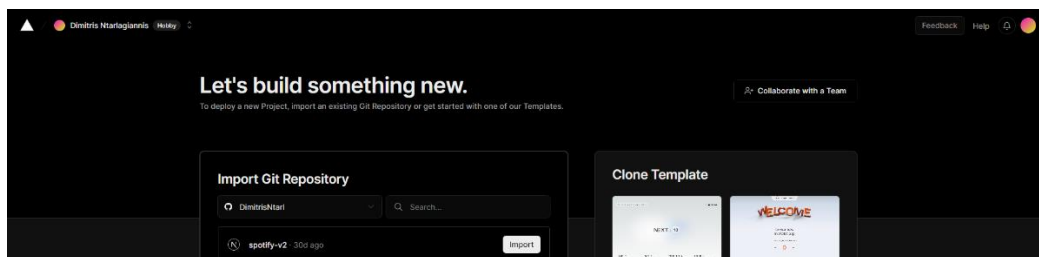
Εικόνα 4.1: Λογότυπο της Vercel [21]

4.2 Διαδικασία Deployment

Η διαδικασία του Deployment ενός έργου με την χρήση Vercel είναι πολύ απλή. Αρχικά, από την στιγμή που ο κώδικας της εργασίας είναι αποθηκευμένος στο GitHub, γίνεται σύνδεση του λογαριασμού του GitHub με αυτόν της Vercel. Έπειτα, μπορεί να γίνει εισαγωγή ενός έργου από το GitHub στη Vercel διαλέγοντας απλά το κατάλληλο αποθετήριο.

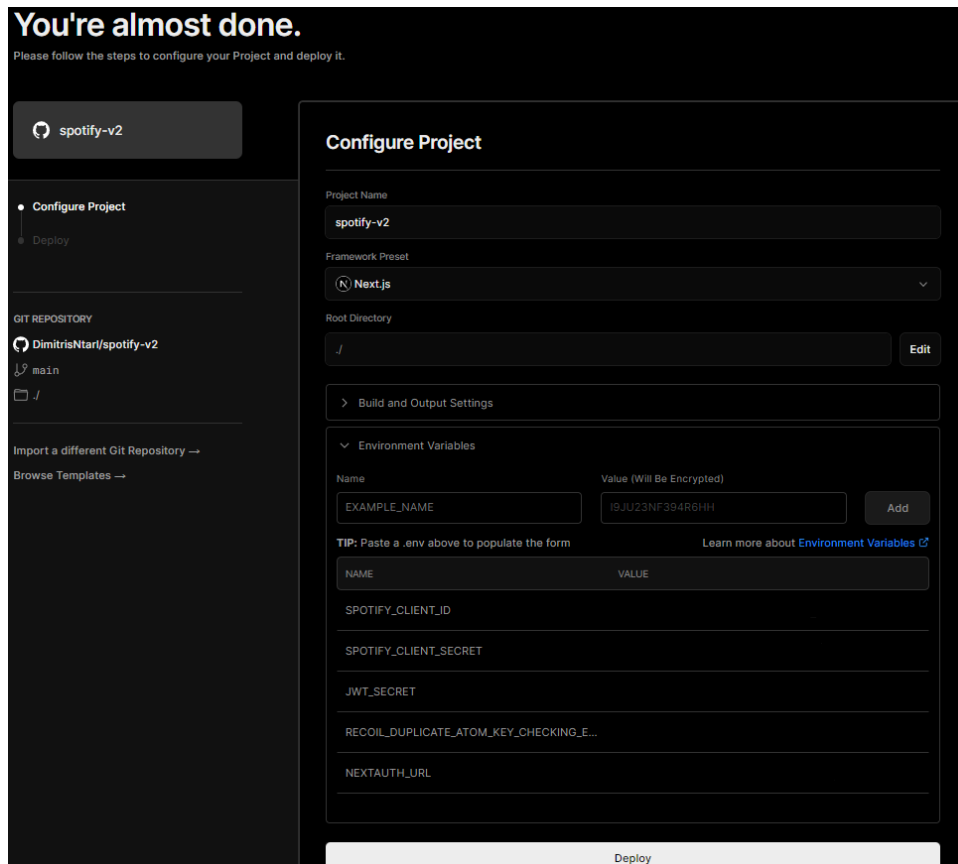


Εικόνα 4.2: Dashboard της Vercel (1)



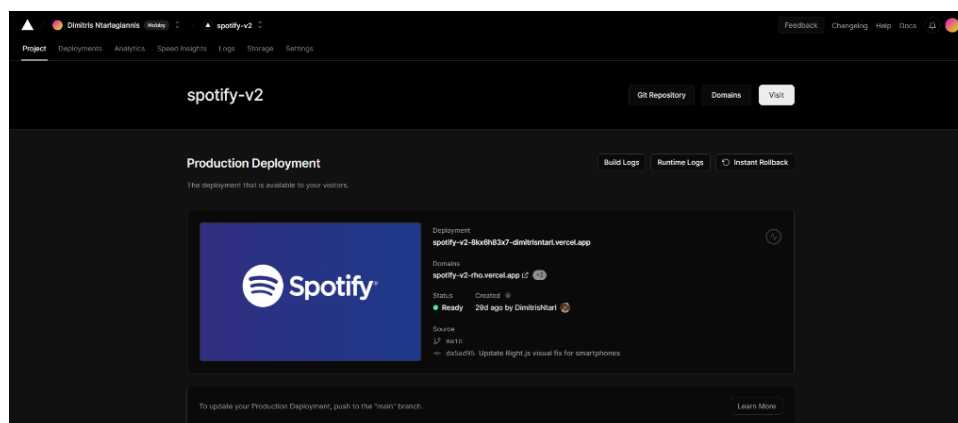
Εικόνα 4.3: Dashboard της Vercel (2)

Στην συνέχεια γίνεται η προσθήκη του ονόματος του έργου, η επιλογή του Framework με το οποίο έγινε η υλοποίηση αλλά και η προσθήκη των μεταβλητών περιβάλλοντος, οι οποίες είναι αναγκαίες διότι χωρίς αυτές δεν θα δουλεύει η αυθεντικοποίηση.



Εικόνα 4.4: Dashboard της Vercel (3)

Μετά την προσθήκη των μεταβλητών το έργο είναι έτοιμο για Deployment και μένει να πατηθεί το κουμπί *Deploy* που φαίνεται στην παραπάνω εικόνα για να ξεκινήσει η διαδικασία. Μόλις ολοκληρωθεί το Deployment εμφανίζεται το Dashboard του έργου.

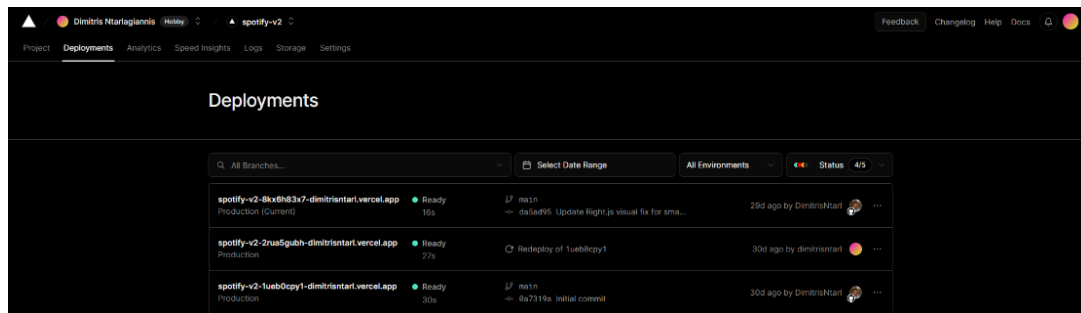


Εικόνα 4.5: Dashboard της Vercel (4)

Εκεί υπάρχουν πολλές και σημαντικές λειτουργίες για τον προγραμματιστή ώστε να παρακολουθεί και να διαχειρίζεται το έργο. Αρχικά, κάθε φορά που προωθείται μία αλλαγή στο αποθετήριο του έργου στο GitHub, η Vercel κάνει αυτόματα ξανά το Deployment του έργου με τις καινούργιες αλλαγές κρατώντας όμως κάθε Deployment που έχει γίνει σε

ΚΕΦΑΛΑΙΟ 4: DEPLOYMENT

περίπτωση που ο προγραμματιστής θέλει να επιστρέψει σε μια παλιότερη έκδοση, να συγκρίνει τις αλλαγές ή οτιδήποτε άλλο.



Εικόνα 4.6: Dashboard της Vercel (5)

Από τις ρυθμίσεις ο προγραμματιστής μπορεί να αλλάξει πάρα πολλά πράγματα σχετικά με το Deployment του έργου, όπως την έκδοση της Node.js που θα χρησιμοποιηθεί, να προσθέσει ένα δικό του domain καθώς η Vercel παρέχει ένα τυχαίο domain, να αλλάξει το κλαδί παραγωγής του αποθετηρίου από το GitHub και πολλά άλλα.

Κεφάλαιο 5

Επίλογος

Στο κεφάλαιο αυτό, το οποίο είναι και το τελευταίο, γίνεται η περιγραφή των συμπερασμάτων από την ολοκλήρωση της εργασίας, δηλαδή το σχεδιασμό και την ανάπτυξη ενός Web Player με τη χρήση του API του Spotify. Επίσης, αναφέρονται και κάποιες λειτουργίες οι οποίες θα μπορούσαν να προστεθούν μελλοντικά ώστε η εφαρμογή να καλύπτει τις ανάγκες περισσότερων χρηστών.

5.1 Συμπεράσματα

Ο βασικός στόχος της εργασίας ήταν η ανάπτυξη ενός Web Player που χρησιμοποιεί το Spotify API. Αυτός ο στόχος επετεύχθη με επιτυχία και το τελικό προϊόν λειτουργεί όπως αναμενόταν. Το API ενσωματώθηκε με επιτυχία στην εφαρμογή, κάτι που επέτρεψε στους χρήστες να έχουν πρόσβαση σε δεδομένα και λειτουργίες που παρέχει το Spotify αλλά και στον λογαριασμό τους σε αυτό.

Η εφαρμογή παρέχει τις βασικές λειτουργίες που αναμένονται από έναν Web Player, όπως αναπαραγωγή, παύση, έλεγχο της έντασης και αναζήτηση τραγουδιών. Επιπλέον, προστέθηκε η δυνατότητα ο χρήστης να κάνει “μου αρέσει” στα τραγούδια που θέλει αλλά και να βλέπει το ιστορικό των είκοσι τελευταίων τραγουδιών που άκουσε ο χρήστης. Δόθηκε μεγάλη έμφαση στο User Interface και στο User Experience καθώς παίζουν κρίσιμο ρόλο στην ικανοποίηση των χρηστών και στην επιτυχία της εφαρμογής. Η ανάπτυξη της εφαρμογής απαιτούσε τη χρήση διάφορων τεχνολογιών και εργαλείων, όπως HTML, CSS, JavaScript και εννοείται το Spotify API, κάτι που οδήγησε και στην εξοικείωση μου με τις τεχνολογίες αυτές.

Κατά την ανάπτυξη της εφαρμογής, αντιμετωπίστηκαν διάφορες προκλήσεις όπως η σωστή λειτουργία της αυθεντικοποίησης του λογαριασμού του χρήστη και η εμφάνιση των δεδομένων του λογαριασμού του. Επίσης, δύσκολη ήταν και η σωστή τοποθέτηση των components ώστε να χωράνε όλα στην οθόνη του χρήστη ομοιόμορφα, χωρίς δηλαδή να φεύγει κάποιο από τα όρια της οθόνης ή το ένα να επικαλύπτει το άλλο. Γι’ αυτόν τον λόγο κάποια components αλλάζουν μορφή ανάλογα με το μέγεθος του παραθύρου του προγράμματος περιήγησης.

5.2 Μελλοντικές Επεκτάσεις

Παρόλο που η εφαρμογή είναι πλήρως λειτουργική με αρκετά χαρακτηριστικά, υπάρχει χώρος για μελλοντικές προσθήκες και βελτιώσεις οι οποίες δεν έγιναν λόγω χρονικών περιορισμών ή προβλημάτων που προέκυψαν κατά την ανάπτυξη. Αρχικά, μία λειτουργία που λείπει είναι η δυνατότητα του χρήστη να έχει πρόσβαση στις λίστες αναπαραγωγής του ή να μπορεί να δημιουργεί και να τις επεξεργάζεται. Μία τέτοια προσθήκη θα έκανε την εφαρμογή ακόμα πιο ολοκληρωμένη και χρήσιμη.

Επίσης, μία ακόμα προσθήκη θα μπορούσε να είναι τα προφίλ των καλλιτεχνών όπου εκεί θα εμφανίζονται όλα τα τραγούδια και λίστες αναπαραγωγής τους αλλά και πιο γενικές πληροφορίες για τους ίδιους και οι χρήστες θα έχουν την δυνατότητα να ακολουθήσουν αυτά τα προφίλ ώστε να ενημερώνονται για τυχόν νέες κυκλοφορίες άλμπουμ ή τραγουδιών.

Βιβλιογραφικές Αναφορές

- 1 Maleshkova, M., Pedrinaci, C., & Domingue, J. (2010). Investigating Web APIs on the World Wide Web. IEEE. 10.1109/ECOWS.2010.9
- 2 MDN contributors. (2023, July 18). HTML: HyperText Markup Language. MDN Web Docs. <https://developer.mozilla.org/en-US/docs/Web/HTML>
- 3 MDN contributors. (2023, July 22). CSS: Cascading Style Sheets. MDN Web Docs. <https://developer.mozilla.org/en-US/docs/Web/CSS>
- 4 MDN contributors. (2023, July 8). JavaScript. MDN Web Docs. <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
- 5 Abramov, D. & Nabors, R. (2023, March 16). Introducing react.dev. <https://react.dev/blog/2023/03/16/introducing-react-dev>
- 6 Next.js. (n.d.). Introduction. Retrieved July, 2023, from <https://nextjs.org/docs>
- 7 Tailwindcss. (n.d.). Get started with Tailwind CSS. Retrieved July, 2023, from <https://tailwindcss.com/docs/installation>
- 8 Visual Studio Code. (n.d.). Getting Started. Retrieved July, 2023, from <https://code.visualstudio.com/docs>
- 9 Orbán, B. (2023, August 11). Introduction. NextAuth.js. <https://next-auth.js.org/getting-started/introduction>
- 10 Headless UI. (n.d.). Documentation. Retrieved July, 2023, from <https://headlessui.com/>
- 11 Recoil. (n.d.). Documentation. Retrieved July, 2023, from <https://recoiljs.org/docs/introduction/getting-started>
- 12 Heroicons. (n.d.). Documentation. Retrieved July, 2023, from <https://github.com/tailwindlabs/heroicons>
- 13 Node.js. (n.d.). Documentation. Retrieved July, 2023, from <https://nodejs.org/api/>

- 14 NPM. (n.d.). Documentation. Retrieved July, 2023, from <https://docs.npmjs.com/about-npm>
- 15 GitHub. (n.d.). GitHub Docs. Retrieved July, 2023, from <https://docs.github.com/en>
- 16 Spotify for Developers. (n.d.). Web API. Retrieved July, 2023, from <https://developer.spotify.com/documentation/web-api>
- 17 Lawson, K. (n.d.). What Is a Single Page Application. Bloomreach. Retrieved July, 2023, from <https://www.bloomreach.com/en/blog/2018/what-is-a-single-page-application>
- 18 Adenowo, A.A.A., & Adenowo, B.A. (2013, July). Software Engineering Methodologies: A Review of the Waterfall Model and Object-Oriented Approach. International Journal of Scientific & Engineering Research, 4(7), 427-434
- 19 tutorialspoint. (n.d.). SDLC – Waterfall Model. Retrieved July, 2023, from https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm
- 20 Barbara, J & Adams, G (2019) react-spotify-web-playback (Version 0.13.4) [Source code] <https://github.com/gilbarbara/react-spotify-web-playback>
- 21 Vercel. (n.d.). Vercel Documentation. Retrieved July, 2023, from <https://vercel.com/docs>