



## EchoBond

Γαβούρας Δημήτριος, Καφαντάρης Κωνσταντίνος, Μωυσής Μωυσής

May 2024

**Προχωρημένη διαχείριση δεδομένων 2023-2024**  
Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών  
Πανεπιστήμιο Θεσσαλίας, Βόλος  
email: {dgavouras, kkafantaris, mmoysis} @ e-ce.uth.gr

### Abstract

Η παρούσα εργασία ασχολείται με την ανάπτυξη μιας ολοκληρωμένης διαδικτυακής εφαρμογής μουσικής, η οποία περιλαμβάνει καλλιτέχνες, άλμπουμ, τραγούδια και τη δυνατότητα κοινωνικής δικτύωσης μεταξύ των χρηστών. Η διαδικασία ανάπτυξης της εφαρμογής ακολούθησε μια σειρά βημάτων, ξεκινώντας από την επιλογή του θέματος και την περιγραφή των προδιαγραφών του, μέχρι την υλοποίηση και την αξιολόγηση της λειτουργικότητάς της.

Το πρώτο βήμα περιλάμβανε την επιλογή του θέματος και τον καθορισμό των απαιτούμενων προδιαγραφών. Ακολούθησε η επιλογή του κατάλληλου Συστήματος NoSQL Βάσης Δεδομένων (ΒΔ) για την αποθήκευση και διαχείριση των δεδομένων της εφαρμογής. Μετά την εγκατάσταση του επιλεγμένου Συστήματος NoSQL ΒΔ, καθορίστηκαν τα χρήσιμα ερωτήματα για την εφαρμογή.

Η δημιουργία δεδομένων πραγματοποιήθηκε μέσω ανεύρεσης έτοιμων δεδομένων από το διαδίκτυο. Στη συνέχεια, σχεδιάστηκε το μοντέλο της ΒΔ και υλοποιήθηκε στο Σύστημα MongoDB (NoSQL), με την επακόλουθη φόρτωση των δεδομένων στη Βάση Δεδομένων. Η εργασία περιλάμβανε επίσης τη δημιουργία κώδικα για λειτουργίες CRUD (Create, Read, Update, Delete) δεδομένων.

Προαριστικά, δημιουργήθηκε διεπαφή χρήστη για τις λειτουργίες CRUD, καθώς και για την εκτέλεση των ερωτημάτων στη ΒΔ. Η υλοποίηση των ερωτημάτων στη ΒΔ και η εκτέλεσή τους αποτέλεσε ένα σημαντικό μέρος της διαδικασίας, με την εξέταση της ορθότητας των αποτελεσμάτων να επιβεβαιώνει την επιτυχή ολοκλήρωση της εφαρμογής.

Η εργασία αυτή αποσκοπεί να αναδείξει τη συνδυαστική δύναμη της τεχνολογίας NoSQL και των κοινωνικών δικτύων, προσφέροντας μια καινοτόμο και διαδραστική πλατφόρμα για τους λάτρεις της μουσικής.

## Contents

1 Εισαγωγή	2
2 Επιλογή βάσης δεδομένων	3
3 Δημιουργία βάσης MongoDB	3
4 Εξήγηση των API's	9
5 Προσδιορισμός χρήσιμων ερωτημάτων για την εφαρμογή	12
6 Υλοποίηση των ερωτημάτων CRUD στη ΒΔ	26
7 Φόρτωση των δεδομένων στη ΒΔ	30
8 Επεξήγηση του User Interface της Εφαρμογής	35
9 Μελλοντικές επεκτάσης της εφαρμογής	45
10 Απεσταλμένα αρχεία	45
11 Βιβλιογραφία	45

## 1 Εισαγωγή

Η εργασία αυτή παρουσιάζει την ανάπτυξη μιας ολοκληρωμένης ιστοσελίδας μουσικής, σχεδιασμένης να προσφέρει μια εμπλουτισμένη μουσική εμπειρία στους χρήστες της. Η πλατφόρμα διαθέτει μια ισχυρή βάση δεδομένων η οποία αποτελείται από καλλιτέχνες, άλμπουμ και τραγούδια, παρέχοντας στους χρήστες εκτενή πρόσβαση σε ποικίλο μουσικό περιεχόμενο. Οι χρήστες μπορούν να εξερευνήσουν προφίλ καλλιτεχνών, πλήρεις δισκογραφίες και μεμονωμένα τραγούδια, ενισχύοντας την ανακάλυψη και την εμπλοκή τους με τη μουσική.

Ένα μοναδικό χαρακτηριστικό της ιστοσελίδας είναι η δυνατότητα κοινωνικής σύνδεσης, επιτρέποντας στους χρήστες να συνδέονται με φίλους, να μοιράζονται τις μουσικές τους προτιμήσεις και να συμμετέχουν σε συζητήσεις. Αυτό το κοινωνικό στοιχείο όχι μόνο ενισχύει την κοινότητα των μουσικόφιλων, αλλά επίσης ενθαρρύνει τη συνεργατική και διαδραστική εμπειρία των χρηστών. Η ενσωμάτωση των κοινωνικών χαρακτηριστικών στοχεύει στη δημιουργία ενός δυναμικού και ελκυστικού περιβάλλοντος όπου η μουσική και η κοινωνική αλληλεπίδραση διασταυρώνονται.

Το έργο δίνει έμφαση στην ευχρηστία πλοήγησης, στις αποδοτικές δυνατότητες αναζήτησης και σε ένα οπτικά ελκυστικό περιβάλλον διεπαφής για να εξασφαλίσει

την καλύτερη δυνατή εμπειρία χρήστη. Η διαδικασία ανάπτυξης περιλαμβανε τη χρήση σύγχρονων τεχνολογιών διαδικτύου για τη δημιουργία μιας ευέλικτης και κλιμακούμενης πλατφόρμας, ικανής να διαχειρίζεται εκτενείς αλληλεπιδράσεις και δεδομένα χρηστών.

Η ιστοσελίδα στοχεύει να γίνει προορισμός αναφοράς για τους λάτρεις της μουσικής, προσφέροντας έναν απρόσκοπτο συνδυασμό μουσικής εξερεύνησης και κοινωνικής αλληλεπίδρασης. Μελλοντικές βελτιώσεις θα επικεντρωθούν στην επέκταση της μουσικής βάσης δεδομένων, στην εισαγωγή εξατομικευμένων προτάσεων και στη περαιτέρω βελτίωση των χαρακτηριστικών κοινωνικής σύνδεσης.

## 2 Επιλογή βάσης δεδομένων

Η βάση δεδομένων που χρησιμοποιήθηκε για την υλοποίηση της ολοκληρωμένης μουσικής πλατφόρμας είναι η MongoDB, η οποία είναι στημένη σε μια online πλατφόρμα στην οποία ο κάθε χρήστης επιλέγει το τρόπο(πακέτο) εγγραφής που επιθυμεί ανάλογα με τις δυνατότητες και τον χώρο που χρειάζεται να του προσφέρει η βάση. Στην προκειμένη περίπτωση έγινε επιλόγη της δωρεάν χρήσης η οποία καλύπτει τις ανάγκες του συγκεκριμένου project. Οι ανάγκες αυτές πιο αναλυτικά είναι η αποθήκευση των στοιχείων του κάθε Αλμπουμ, του κάθε καλλιτέχνη και του κάθε τραγουδιού. Επιπλέον αποθηκεύσεις στη βάση όπως, οι εγγραφές των χρηστών και των στοιχείων τους, οι φιλίες μεταξύ τους και οι συνομιλίες τους ακόμα και οι αναζητήσεις τους, καλύπτονται όλες από το δωρεάν πακέτο που επιλέξαμε.

## 3 Δημιουργία βάσης MongoDB

Για την δημιουργία μιας βάσης στην πλατφόρμα της MongoDB πρέπει να ακολουθηθούν μερικά βήματα όπως φαίνεται στην συνέχεια:

1. Αρχικά συνδεόμαστε στην ιστοσελίδα της MongoDB με τον εξής σύνδεσμο : <https://www.mongodb.com/> και οδηγούμαστε στην αρχική σελίδα όπως φαίνεται στην φωτογραφία παρακάτω(Figure 1) .

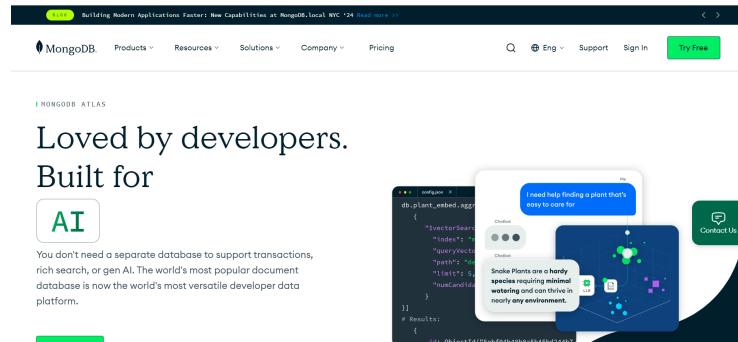


Figure 1: Αρχική σελίδα της MongoDB

- Στη συνέχεια επιλέγουμε το κουμπί **Sing In** και στην σελίδα που μας οδηγεί επιλέγουμε το κουμπί δίπλα στην φράση "Don't have an account? **Sign Up**" , όπως φαίνεται στο Figure 2.



Figure 2: Σελίδα "Sing In"

- Στην σελίδα που οδηγούμαστε, τοποθετούμε τα στοιχεία που ζητούνται και στην συνέχεια αφού σθμπληρώσουμε σωστά όλα τα πεδία πατάμε το κουμπί **Sing Up** στο τέλος της φόρμας, όπως φαίνεται στο Figure 3.

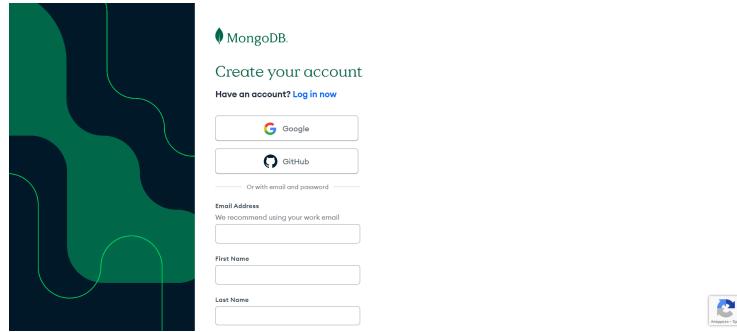


Figure 3: Σελίδα "Sing Up"

4. Ακολουθούμε την διαδικασία επιβεβαίωσης του email που τοποθετήσαμε και έπειτα κάνουμε **Sing In** τοποθετώντας τα στοιχεία σύνδεσής μας, όπως φαίνεται στο Figure 4.

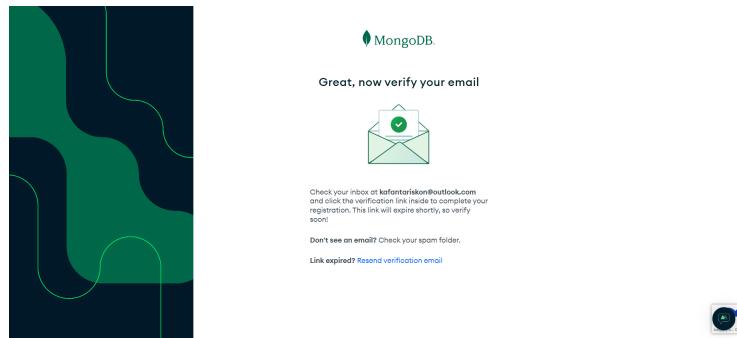


Figure 4: Σελίδα "Verify"

5. Στη συνέχεια, οδηγούμαστε στην κεντρική σελίδα της MongoDB και είμαστε έτοιμοι να δημιουργήσουμε την βάση μας, όπως φαίνεται στο Figure 5.

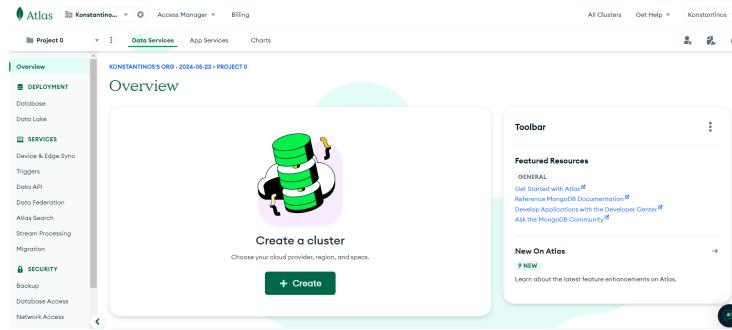


Figure 5: Κεντρική σελίδα της MongoDB

6. Στην κεντρική σελίδα που βρισκόμαστε πατάμε το κουμπί "Create" στο πεδίο "Create a Cluster" και οδηφούμαστε στην σελίδα του Figure 6, όπου επιλέγουμε το δεξιό πεδίο, το οποίο αναφέρεται ως "M0" και τοποθετούμε "cluster name" στο πεδίο όπου αναφέρεται και ζητείται. Έπειτα επιλέγουμε την "AWS" ως provider και κάνουμε κλικ στο κουμπί "create deployment", το οποίο βρίσκεται κάτω δεξιά στη σελίδα..

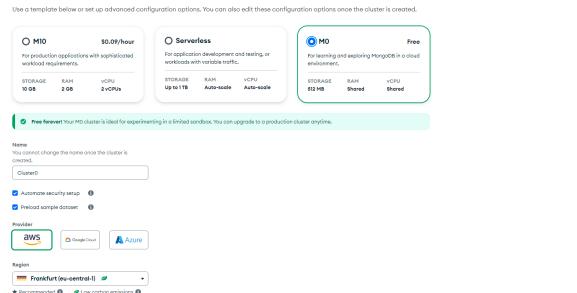


Figure 6: Σελίδα create cluster

7. Για το επόμενο βήμα οδηγούμαστε αυτόματα παλι στην κεντρική σελίδα όπως φαίνεται στο Figure 7, οπου δημιουργούμε τον χρήστη του Cluster.

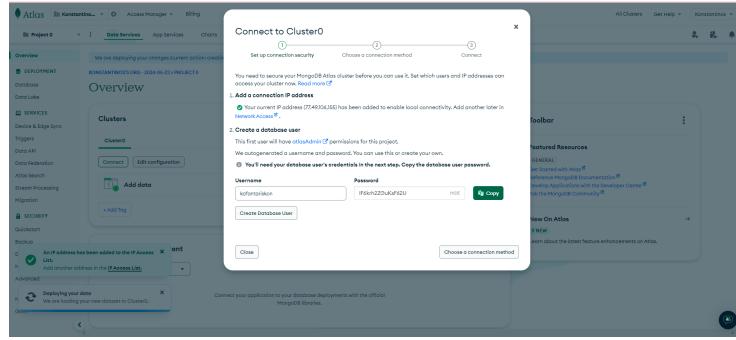


Figure 7: Σελίδα create cluster user

8. Επειτα οδηγούμαστε στο περιεχόμενο του Figure 8, όπου και επιλέγουμε το κουμπί **"Choose a connection method"** κάτω δεξιά στο αναδύομενο παράθυρο.

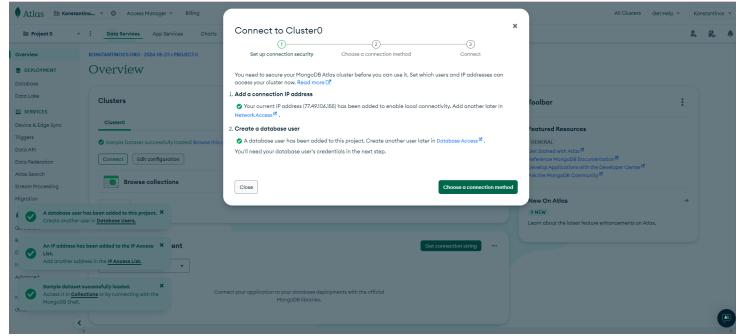


Figure 8: Σελίδα cluster connection method

9. Στη συνέχεια επιλέγουμε την πρώτη επιλογή η οποία αναφέρεται ως "Drivers", όπως φαίνεται στο Figure 9.

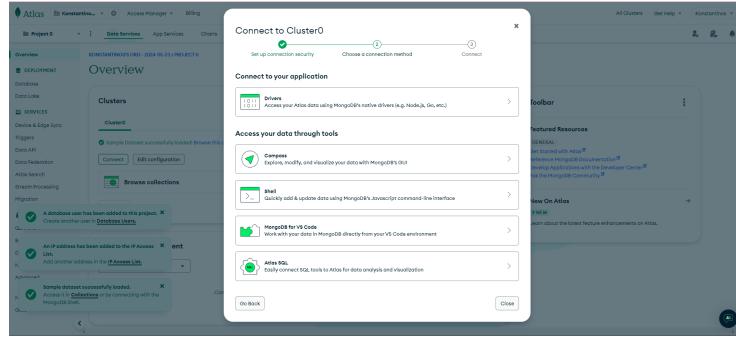


Figure 9: Σελίδα cluster's connection methods

10. Προχωρώντας για να ολοκληρώσουμε την σύνδεση με το Server (Figure 10), πρέπει στο filepath του φωκέλου που θα δημιουργήσουμε για την υλοποίηση αυτού σε γλώσσα Node.js, να εγκαταστήσουμε την MongoDB με την εντολή **npm install mongodb**. Επιπρόσθετα αντιγράφουμε τον connection string που μας δίνει και το τοποθετούμε στο αρχείο database.js του server που δημιουργήσαμε ως "const uri = *your connection string*;" , όπως φαίνεται στο Figure 11.

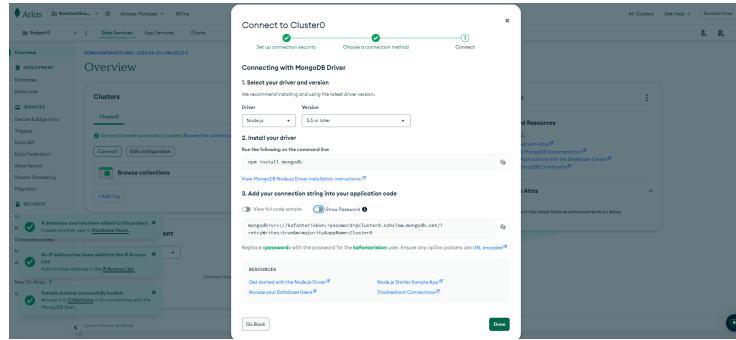


Figure 10: Σελίδα cluster's final steps

```

import { MongoClient } from "mongodb"

const uri = <"your connection string">;
const client = new MongoClient(uri)

export default client

```

Figure 11: MongoDb - server connection

## 4 Εξήγηση των API's

Για να μπορέσουμε να προχωρήσουμε στην επεξήγηση των ερωτημάτων τα οποία έχουν χρησιμοποιηθεί στην εφαρμογή οφείλουμε να βίξουμε μια σύντομη ματιά στα API's που δημιουργήθηκαν για τον σκοπό αυτό.

- **User's API's**

Τα περισσότερα και τα βασικότερα από όλα τα API's έχουν να κάνουν με την λειτουργία των ερωτημάτων του χρήστη. Ο χρήστης είναι η μόνη οντότητα στην βάση δεδομένων για την οποία χρειάζονται κάθε τύπου API , δηλαδή GET,POST,PUT,DELETE και PATCH, όπως φαίνεται στις παρακάτω εικόνες (Figure 14 ,15).

```

// User log in.
router.post("/login", LogIn)

// User register.
router.put("/register", Register)

// Delete user's account.
router.delete("/delete-account", DeleteUser)

// Change user's country.
router.patch("/change-user-data", ChangeUserData)

// User addfriend
router.post("/add-friend", AddFriend)

// User confirm friend request
router.post("/confirm-friend", ConfirmFriend)

// User decline friend request.
router.post("/decline-request", DeclineRequest)

// Get all user's friends.
router.get("/get-friends", GetFriends)

// Get all user's friend requests.
router.get("/get-friend-requests", GetFriendRequests)

router.post("/search-friends", SearchFriends)

```

Figure 12: API User(1)

```

// ...
// Change user's photo.
router.patch("/change-user-image", ChangeProfileImage)

// Get user's data.
router.get("/get-user-info", GetUserInfo)

// ...
// Favourites.

router.put("/add-favourite-album", AddFavouriteAlbum)
router.delete("/remove-favourite-album", RemoveFavouriteAlbum)

router.put("/add-favourite-artist", AddFavouriteArtist)
router.delete("/remove-favourite-artist", RemoveFavouriteArtist)

router.get("/get-favourite-albums", GetFavouriteAlbum)
router.get("/get-favourite-artists", GetFavouriteArtist)

// ...

router.get("/search-recent", SearchRecent)
router.get("/get-recent-songs", GetRecentSongs)

```

Figure 13: API User(2)

- **Αλμπουμ**

Σχετικά με τα άλμπουμ δημιουργήθηκαν μόνο δύο GET API για να ικανοποιούν τα κατάλληλα ερωτήματα, όπως φαίνεται στο Figure 14 παρακάτω.

```

// Get a specific songs from album
router.get("/get-album-songs", GetAlbumSongs)

// get all albums
router.get("/get-all-albums", GetAllAlbums)

```

Figure 14: API αλμπουμ

- **Καλλιτέχνες**

Σχετικά με τους καλλιτέχνες δημιουργήθηκαν μόνο δύο GET API με τον ίδιο τρόπο με αυτά των Αλμπουμ, όπως φαίνεται στο Figure 15 παρακάτω.

```
// Get all artists from the database.
router.get("/get-all-artists", GetAllArtists)

// Get a specific artist's all songs.
router.get("/get-artists-songs", GetArtistssongs)
```

Figure 15: API καλλιτεχνών

- **Τραγούδια**

Αναφορικά με τα τραγούδια δημιουργήθηκαν τρία GET API ,όπως φαίνεται στο Figure 16 παρακάτω.

```
// Get a specific song.
router.get("/get-song", GetSong)

router.get("/get-song-data", GetSongData)

router.get("/get-all-songs", GetAllSongs)
```

Figure 16: API τραγουδιών

- **Αναζήτηση**

Για την αναζήτηση υλοποιήθηκαν δύο APIs , ένα PUT και ένα GET τα οποία θα αναλυθούν περαιτέρω στην συνέχεια (Figure 17).

```
router.get("/get-search", GetSearch)

router.put("/search-items", SearchItems)
```

Figure 17: API αναζήτησης

- **Tokens**

Για την υλοποίηση των token χρειάστηκαν δύο GET API, τα οποία χρησιμοποιούνται σε κάθε προσπάθεια επικοινωνίας μεταξύ Back-end και Front-end (Figure 18).

```
// Check if token exists and is not expired.  
router.get("/check-token", CheckToken)  
  
// Refresh access token.  
router.get("/refresh-token", RefreshToken)
```

Figure 18: API Tokens

- **Συνομιλία**

Για την υλοποίηση της συνομιλίας μεταξύ χρηστών χρειάστηκε ένα POST API, το οποίο φαίνεται στο Figure 19.

```
router.post("/get-conversation-id", GetConversationID)
```

Figure 19: API Συνομιλίας

- **One Time Password(OTP)**

Για το OTP χρειάστηκε ένα POST API (Figure 20).

```
router.post("/send-otp", SendOTP)
```

Figure 20: API OTP

## 5 Προσδιορισμός χρήσιμων ερωτημάτων για την εφαρμογή

Στην παράγραφο αυτή ότι σχολιαστούν τα βασικά ερωτήματα της εφαρμογής μας, τα οποία έχουν να κάνουν με Αναζήτηση τραγουδιών, με αποθήκευση προσφάτων και αγαπημένων και τέλος με την επικοινωνία των χρήστων μεταξύ τους.

### 1. Ερωτήματα των Tokens

Αρχικά αξίζει να αναφερθούν τα ερωτήματα των Tokens, τα οποία είναι το "access Token" και το "refresh token".

Το access token χρησιμοποιείται σε όλα τα requests ώστε να επαληφθεύεται η ταυτότητα του χρήστη. Αυτό ενισχύει την ασφάλεια του server μας από κακόβουλους χρήστες που μπορεί να θέλουν να εισβάλλουν στα δεδομένα του server. Το access token δημιουργείται κατά την είσοδο (log in) του χρήστη μαζί με ένα refresh token το οποίο είναι υπεύθυνο για να κάνει refresh το άλλο token. Αυτά αποθηκεύονται στο local storage του κάθε client. Το καθένα είναι μοναδικό για κάθε χρήστη και στο payload που περιέχουν βρίσκεται το username τους hashed. Το access token δημιουργείται με διάρκεια μίας ώρας ενώ το refresh token με διάρκεια μίας εβδομάδας επιτρέποντας τους χρήστες να παραμένουν συνδεδεμένοι, ακόμα και αν βγαίνουν και ξαναμπαίνουν στο browser. Αυτά προστίθενται στο header του κάθε axios request με αποτέλεσμα να τα λαμβάνει το server. Για την διαχείρισή τους δημιουργήθηκαν δύο συναρτήσεις (με ονόματα CheckToken και RefreshToken).

```
try {
  // Check if token exists.
  let accessToken = req.headers.authorization

  if (!accessToken) {
    return res.status(401).json({ error: "No token" })
  }

  // Check if token is expired.
  accessToken = accessToken.split(' ')[1]
  jwt.verify(accessToken, process.env.ACCESS_TOKEN_SECRET)

  return res.status(200).json( { status: "okay" } )
} catch (error) {
  return res.status(401).json({ error: "Token expired" })
}
```

Figure 21: Access Token (1)

Η συνάρτηση CheckToken ελέγχει αν το access token ενός χρήστη είναι ακόμα έγκυρο, ενώ η συνάρτηση RefreshToken κάνει refresh το access token χρησιμοποιώντας το refresh token του. Στη συνέχεια επιστρέφει το ανανεωμένο access token ώστε ο χρήστης να καταφέρνει να συνδέεται με επιτυχία. Σε περίπτωση αποτυχίας του refresh token (γεγονός που σηματοδοτεί ότι αυτό δεν είναι πλέον έγκυρο) ο χρήστης ανακατευθύνεται στη σελίδα log in ώστε να ξανά εισέλθει στην κύρια σελίδα με τα στοιχεία του.

```

let refreshToken = req.headers.authorization;

if (!refreshToken) [
  return res.status(401).json({ error: "Refresh token not provided" });
]

refreshToken = refreshToken.split(" ")[1];

try {
  // Verify the refresh token
  let payload = jwt.verify(refreshToken, process.env.REFRESH_TOKEN_SECRET);

  // Remove any sensitive or unnecessary fields from the payload
  const { exp, iat, ...newPayload } = payload;

  // Create a new access token with the updated payload
  const accessToken = jwt.sign(newPayload, process.env.ACCESS_TOKEN_SECRET, { expiresIn: '1h' });

  return res.status(200).json({ accessToken: accessToken });
} catch (err) {
  console.error("Error verifying refresh token: ", err.message);
  return res.status(401).json({ error: "Invalid refresh token" });
}

```

Figure 22: Refresh Token (2)

## 2. Ερωτήματα Χρήστη

Μερικά από τα πολλα ερωτήματα του χρήστη που χρησιμοποιούνται είναι η αποθήκευση της αναζήτησης που κάνει ο χάθε χρήστης για να βρεί το αλμπουμ , τον καλλιτέχνη ή το τραγούδι που επιθυμεί, ως αναζητήθηκε πρόσφατα (recently searched) . Στο ερώτημα αυτό αναζητείται με βάση το username από το collection "History" της βάσης οι πρόσφατες αναζητήσεις που έχει κάνει ο χρήστης με την εντολή `.find(username : username)` (Figure 23) και τα δεδομένα επιστρέφονται σε μορφή πίνακα. Έπειτα τα δεδομένα χωρίζονται αναλόγως με τον τύπο που έχει το κάθε ένα (αλμπουμ, καλλιτέχνης ή τραγουδι) και επιστρέφονται σε json μορφή (Figure 24) όπως φαίνεται στη συνέχεια: `res.status(200).json( albums: albums,artists:artists,songs:songs )`;

```
let history = client.db("EchoBond").collection("History");
let alb = client.db("EchoBond").collection("Albums");
let art = client.db("EchoBond").collection("Artists");
let song = client.db("EchoBond").collection("Songs");

const historyData = await history.find({ username: username })
  .sort({ _id: -1 })
  .limit(5)
  .toArray();

if (historyData.length === 0) {
  return res.status(404).json({ message: "No history found." });
}

const albums = [];
const artists = [];
const songs = [];
```

Figure 23: Search Query (1)

```

for (const item of historyData){
  if (item.type === 'album') {
    const albumsDetails = await alb.find({ name: item.title }).toArray();
    Complexity is 4 Everything is cool!
    const processedAlbums = albumsDetails.map(album => {
      if (album.image && album.image.buffer) {
        // Assuming the binary data is in a buffer;
        album.image = `data:image/jpeg;base64,${[album.image.buffer.toString('base64')]}`;
      }
      return album;
    });
    albums.push(...processedAlbums);
  } else if (item.type === 'artist') {
    const artistDetails = await art.find({ name: item.title }).toArray();
    Complexity is 4 Everything is cool!
    const processedArtists = artistDetails.map(artist => {
      if (artist.image && artist.image.buffer) {
        // Assuming the binary data is in a Buffer;
        artist.image = `data:image/jpeg;base64,${[artist.image.buffer.toString('base64')]}`;
      }
      return artist;
    });
    artists.push(...processedArtists);
  }
  else if (item.type === 'song') {
    const songDetails = await song.find({ title: item.title },
      { projection: { title: 1, artist: 1, time: 1, image: 1 } })
      .toArray();
    Complexity is 4 Everything is cool!
    const processedSongs = songDetails.map(songg => {
      if (songg.image && songg.image.buffer) {
        // Assuming the binary data is in a Buffer;
        songg.image = `data:image/jpeg;base64,${[songg.image.buffer.toString('base64')]}`;
      }
      return songg;
    });
    songs.push(...processedSongs);
  }
}
  
```

Figure 24: Search Query (2)

Επίσης σημαντικό είναι να αναφερθεί το Query(ερώτημα) για τα πρόσφατα τραγούδια που έχει ακούσει ο χρήστης, Το Query ζητάει αυτά τα τραγούδια από την βάση για να τοποθετηθούν στην αρχική σελίδα της εφαρμογής έτσι ώστε ο χρήστης να βρίσκει εύκολα κάποιο τραγούδι που άκουσε πρόσφατα(Figure 25) .

```

let token = req.headers.authorization;
if (!token) {
  return res.status(401).json({ error: "Unauthorized access" });
}
token = token.split(' ')[1];
const decode = jwt.verify(token, process.env.ACCESS_TOKEN_SECRET);
let username = decode.username;

const history = client.db("EchoBond").collection("Listened_History");
const song = client.db("EchoBond").collection("Songs");

const historyData = await history.find({ username: username })
  .sort({ _id: -1 })
  .limit(4)
  .toArray();

if (historyData.length === 0) {
  return res.status(404).json({ message: "No history found." });
}

const songPromises = historyData.map(item =>
  song.find({ title: item.title }, {
    projection: {
      title: 1,
      artist: 1,
      album: 1,
      time: 1,
      image: 1
    }
  }).toArray()
)

const songsData = await Promise.all(songPromises);

```

Figure 25: Recent songs Query

Τα τραγούδια αυτά αποθηκεύονται στη βάση μέσω του Socket(θα αναλυθούν αργότερα) την ώρα που ο χρήστης επιλέγει να ακούσει ένα τραγούδι όπως φαίνεται στο Figure 26. Αν ένα τραγούδι που παίζεται αυτή την στιγμή υπάρχει ήδη στα recently listened τότε διαγράφεται και προστίθεται ξανα πρώτο στην σειρά ως πιο πρόσφατη εγγραφή.

```

let history = client.db("EchoBond").collection("Listened_History");

if (history.countDocuments({}) != 0) {
  await history.findOneAndDelete ({title : title, username : username})
}

await history.insertOne({ title : title, username : username})

```

Figure 26: Add recent played song

Στη συνέχεια αναφέρεται το πώς επιστρέφονται τα δεδομένα του χρήστη όταν αυτός επιλέξει να δεί το προφίλ του μέσα στην εφαρμογή (Figure 27). Το κομμάτι **projection:username:1, ...** επιστρέφει μόνο συγκεκριμένα δεδομένα, για να μην επιστρέψει άσκοπα δεδομένα που δεν χρησιμοποιούνται γιατί το ερώτημα.

```

let users = client.db("EchoBond").collection("Users");

const userInfo = await users.findOne(
  { username: username },
  { projection: { username: 1, email: 1, Photo: 1, Country: 1 } }
);

if (userInfo && userInfo.Photo) {
  // Convert the photo buffer to a Base64 string
  userInfo.Photo = `data:image/jpeg;base64,${userInfo.Photo.toString('base64')}`;
}

```

Figure 27: User Data

Άλλα ερωτήματα άξια αναφοράς είναι αυτά σχετικά με την διεπαφή των χρηστών μεταξύ τους, όπως αυτά της προσθήκης φίλου, της αποδοχής του ή της απόρριψης του αντίστοιχα. Ξεκινώντας με αυτό της προσθήκης φίλου (Figure 28) στο οποίο γίνεται μια εισαγωγή στο Collection "Friends" στο οποίο αποθηκεύεται η πληροφορία ως "status:pending" ανάμεσα στα δύο username εως ότου ο χρήστης που δέχεται το αίτημα φιλίας απαντήσει.

3.

```

let token = req.headers.authorization;

if (!token) {
  return res.status(401).json({ error: "Unauthorized access" });
}

token = token.split(' ')[1];

const decode = jwt.verify(token, process.env.ACCESS_TOKEN_SECRET);
const usernameMain = decode.username;

let friends = client.db("EchoBond").collection("Friends");

await friends.insertOne({ usernameMain : usernameMain, usernameSec : usernameSec, status : 'pending', addedOn: new Date() });

if (connectedUsers.has(usernameSec)) {
  io.to(usernameSec).emit('friendRequest', usernameMain)
}

```

Figure 28: Προσθήκη φίλου

Στην περίπτωση τώρα που ο χρήστης, ο οποίος δέχτηκε το αίτημα φιλίας αποφασίσει να το αποδεχτεί, τρέχει ένα ερώτημα, το οποίο, όπως φαίνεται και στο Figure 29, εντωπίζει το αίτημα και μετατρέπει το "status" σε confirmed, το οποίο σημαίνει πως πλέον οι δύο χρήστες εμφανίζονται ως φίλοι στην βάση. Επιπροσθέτως γίνεται μια καινούρια εγγραφή στο Collection της βάσης δεδομένων "Conversations", στο οποίο αποθηκεύονται τα δύο ονόματα και ένα μοναδικό ID, το οποίο χρησιμοποιείται αργότερα για την επικοινωνία τους μέσω socket στο chat.

```

let token = req.headers.authorization;

if (!token) {
  return res.status(401).json({ error: "Unauthorized access" });
}

token = token.split(' ')[1];

const decode = jwt.verify(token, process.env.ACCESS_TOKEN_SECRET);
const usernameMain = decode.username;

let friends = client.db("EchoBond").collection("Friends");

//find the request pending from Sec to Main user then update the status
if (await friends.findOneAndUpdate(
  { usernameMain: usernameSec, usernameSec: usernameMain, status: 'pending' }, // Find data.
  { $set: { status: 'confirmed' } }, // Update data.
  { returnOriginal: false } // Return new updated file.
)) {
  //if update was succesfull , then insert a new conversation in the specific collection.
  let conversation = client.db("EchoBond").collection("Conversations");

  await conversation.insertOne({ username1 : usernameSec, username2 : usernameMain });

  // console.log("All Good")
}

```

Figure 29: Αποδοχή φίλου

Το ίδιο συμβαίνει και με την απόρριψη ενός αιτήματος φιλίας, μόνο που αντί να ανανεώνεται το πεδίο "status" διαγράφεται η εισαγωγή από το Collection "Friends" και δεν δημιουργείται κάποια εισαγωγή στο

Collection "Conversations".

Τελευταίο και πολύ σημαντικό ερώτημα είναι αυτό της εύρεσης του φίλου μέσω λειτουργίας search, όπως φαίνεται στο Figure 30 παραχάτω. Σε αυτό το query αρχικά επιστρέφονται όλοι οι χρήστες με τους οποίου ο χρήστης που κάνει το request είναι φίλος. Στην συνέχεια γίνεται ένα query, το οποίο χρησιμοποιούντας το Collection "Users" εντωπίζει όλους τους χρήστες με τους οποίους υπάρχει εγγραφή στο Collection "Friends" και δεν τους εμφανίζει καθόλου, ενώ εμφανίζει αυτούς που το username τους ταιριάζει (εν μέρει) με το "searchValue" ("τιμή αναζήτησης") και δεν έχουν "σχέση φιλίας" ή "σχέση pending" μεταξύ τους (Figure 30).

```
const users = client.db("EchoBond").collection("Users");
const friends = client.db("EchoBond").collection("Friends");

const friendships = await friends.find({
  $or: [
    { usernameMain: username, status: { $in: ['confirmed', 'pending'] } },
    { usernameSec: username, status: { $in: ['confirmed', 'pending'] } }
  ]
}).toArray();

const friendshipUsernames = [];
Complexity is 3. Everything is cool!
friendships.forEach(friendship => {
  if (friendship.usernameMain !== username) {
    friendshipUsernames.push(friendship.usernameMain);
  }
  if (friendship.usernameSec !== username) {
    friendshipUsernames.push(friendship.usernameSec);
  }
});

const friendData = await users.aggregate([
  {
    $match: {
      username: { $regex: `^${friendUsername}^`, $ne: username, $options: 'i' },
      username: { $ne: username },
    }
  },
  {
    $match: {
      username: { $nin: friendshipUsernames }
    }
  },
  {
    $project: {
      username: 1,
      Photo: 1
    }
  },
  [
    { $limit: 3 }
  ]
]).toArray();
```

Figure 30: Αναζήτηση φίλων

#### 4. Ερωτήματα Αναζήτησης

Τα ερωτήματα αναζήτησης, όπως αναφέρθηκε και παραπάνω είναι τα εξής

δύο:

Το πρώτο είναι η αναζήτηση που κάνει ο χρήστης , ερώτημα το οποίο με την τιμή αναζήτησης που βάζει ως input επιστρέφει τα ”αντικείμενα” τα οποία ταιριάζουν περισσότερο με αυτή χατηγοριοποιημένα σε αλμπουμ , καλλιτέχνες και τραγουδια ξεχωριστά ,όπως φαίνεται στα Figure 31,32 .

```
let albumResults
let artistResults
let songResults
try {
  // Token-based user identification.
  let token = req.headers.authorization;
  if (!token) {
    return res.status(401).json({ error: "Unauthorized access" });
  }

  token = token.split(' ')[1];
  jwt.verify(token, process.env.ACCESS_TOKEN_SECRET)

  let searchValue = req.query.value;

  let album = client.db("EchoBond").collection("Albums");
  let artist = client.db("EchoBond").collection("Artists");
  let song = client.db("EchoBond").collection("Songs");

  albumResults = await album.find({
    $or: [
      { name: new RegExp(searchValue, 'i') }
    ]
  }).toArray();
  Complexity is 4 Everything is cool!
  albumResults = albumResults.map(album => {
    if (album.image && album.image.buffer) [
      album.image = `data:image/jpeg;base64,${album.image.buffer.toString('base64')}`;
    ]
    return album;
  });
}
```

Figure 31: Αναζήτηση(1)

```

artistResults = await artist.find({
  $or: [
    { name: new RegExp(searchValue, 'i') }
  ]
}).toArray();

Complexity is 4 Everything is cool!
artistResults = artistResults.map(artist => {
  if (artist.image && artist.image.buffer) {
    artist.image = `data:image/jpeg;base64,${artist.image.buffer.toString('base64')}`;
  }
  return artist;
});

songResults = await song.find({ title: new RegExp(searchValue, 'i') },
  {projection: { title: 1, artist: 1, time: 1, image: 1 }})
.toArray();

Complexity is 4 Everything is cool!
songResults = songResults.map(song => {
  if (song.image && song.image.buffer) {
    song.image = `data:image/jpeg;base64,${song.image.buffer.toString('base64')}`;
  }
  return song;
});

```

Figure 32: Αναζήτηση(2)

Και το δεύτερο είναι η αποθήκευση των πρόσφατων αναζητήσεων του χρήστη, ερώτημα το οποίο λειτουργεί με την επιλογή οποιουδήποτε αντικειμένου που εμφανίζεται ως αποτέλεσμα μετά την διαδικασία της αναζήτησης. Έτσι χρατίσται ένα πρόσφατο ιστορικό, το οποίο, όπως προαναφέρθηκε εμφανίζεται στον χρήστη κάτα την εισαγωγή του στην αναζήτηση για δικιά του βιοήθεια (Figure 33). Το ερώτημα αυτό είναι παρόμοιο με το ερώτημα "πρόσφατα τραγούδια", το οποίο περιγράφηκε πιο πάνω.

```

let token = req.headers.authorization;
if (!token) {
  return res.status(401).json({ error: "Unauthorized access" });
}

token = token.split(' ')[1];
const decode = jwt.verify(token, process.env.ACCESS_TOKEN_SECRET)
const username = decode.username

let title = req.body.name;
let type = req.body.type;

let history = client.db("EchoBond").collection("History");

if (history.countDocuments({}) != 0) {
  await history.findOneAndDelete ({title : title, type : type, username : username})
}

await history.insertOne({ title : title, type : type, username : username})

```

Figure 33: Εισαγωγή πρόσφατης αναζήτησης

## 5. Ερωτήματα μηνυμάτων

Τα ερωτήματα των μηνυμάτων είναι αυτο της προσθήκης μηνύματος, το οποίο φαίνεται στο Figure 34. Το query αυτο παίρνει τα δεδομένα conversation Id , τον αποστολέα , το περιεχόμενο του μηνύματος και την ημερομηνία αποστολής, γίνεται μια εισαγωγή στο Collection "Messages". Αν ο παραλήπτης έιναι συνδεδεμένος στον λογαριασμό του έρχεται ειδοποίηση(notification) , αλλιώς αποθηκεύεται για να σταλεί το μήνυμα όταν κάνει ο ίδιος log in.

```
const { conversationID, sentFrom, message, sentDate } = details

try {
  let messageCollection = client.db("EchoBond").collection("Messages")

  await messageCollection.insertOne({ conversationID, sentFrom, message, sentDate })

  const receiver = await GetReceiver(conversationID, sentFrom)

  console.log(receiver, connectedUsers)

  if (connectedUsers.has(receiver)) {
    io.to(receiver).emit('newMessage', sentFrom, message)
  }
  else {
    let notifications = client.db("EchoBond").collection("Notifications")

    notifications.insertOne({ username: receiver, sender: sentFrom, message: message })
  }
} catch (error) {
  console.log(error)
}
```

Figure 34: Αποστολή μηνύματος

Τώρα, αναφορικά με το ερώτημα εύρεσης όλων των μηνυμάτων μιας συνομιλίας, το οποίο εκτελεί απλά ένα find query, το οποίο επιστρέφει με βάση το conversation ID όλα τα μήνυματα της συνομιλίας αυτής(Figure 35).

```
try {
  const messagesCollection = client.db("EchoBond").collection("Messages")
  const conversationMessages = await messagesCollection.find({ conversationID }).toArray()

  return conversationMessages
} catch (error) {
  console.error('Failed to retrieve conversation messages:', error)
  return [] // Return an empty array to signify failure
}
```

Figure 35: Επιστροφή των μηνυμάτων μιας συνομιλίας

## 6. Ερωτήματα Αγαπημένων

Τα ερωτήματα αυτά ανήκουν στα ερωτήματα του χρήστη. Ξεκινώντας με το query της προσθήκης ενός αγαπημένου καλλιτέχνη ή ενός αγαπημένου αλμπουμ σε ενα Collection στη βάση το οποίο αποθηκεύει το όνομα του καλλιτέχνη ή τον τίτλο του άλμπουμ , το όνομα χρήστη αλλα και τον τύπο του ”αντικειμένου” , δηλαδή αν είναι αλμπουμ ή καλλιτέχνης(Figure 36)

```
if (!token) {
  return res.status(401).json({ error: "Unauthorized access" })
}

token = token.split(' ')[1]

const decode = jwt.verify(token, process.env.ACCESS_TOKEN_SECRET)
const username = decode.username

//getting the name of the album
const artistName = req.body.name
//fining the correct collection
const favourites = client.db("EchoBond").collection("Favourites")

//creating the item we want to insert
const favourite = {
  username,
  artistName,
  type:"artist"
};

// Insert the favourite into the database
const result = await favourites.insertOne(favourite);
// console.log(` A document was inserted with the ArtistName: ${artistName}`);
res.send({ message: 'Favourite added successfully', _id: result.insertedId });
```

Figure 36: Προσθήκη αγαπημένου

Επιπροσθέτως , παρόμοιο είναι και το ερώτημα της αφαίρεσης ενός αγαπημένου αλμπουμ ή καλλιτέχνη μόνο που χρησιμοποιεί delete query έτσι ώστε να διαγράψει την εγγραφή από το Collection(Figure 37).

```

//getting the name of the artist
const artistName = req.query.name
const favourites = client.db("EchoBond").collection("Favourites")

//creating the item
const favourite = {
  username: username,
  artistName: artistName,
  type: "artist"
};

// Delete the favorite from the database
const result = await favourites.deleteOne(favourite);
// console.log(`A document was deleted with the artistName: ${artistName}`);

res.send({ message: 'Favorite removed successfully', _id: result.insertedId });

```

Figure 37: Αφαίρεση αγαπημένου

Τελευταίο ερώτημα που αφορά τα αγαπημένα είναι το ερώτημα το οποίο επιστρέφει από την βάση όλα τα αγαπημένα αλμπουμ και όλους τους αγαπημένους καλλιτέχνες του χρήστη βρίσκοντάς τα με ένα find query χρησιμοποιώντας το username του (Figure 38).

```

//Finding the client and the right collection
const favourite = client.db("EchoBond").collection("Favourites")
const artistsCollection = client.db("EchoBond").collection("Artists");

//returns all the Favourite Albums from the username
allFavouriteArtists = await favourite.find({username:username,type:"artist"}).toArray();

const favouriteArtistsnames = allFavouriteArtists.map(artist => artist.artistName);
allFavouriteArtists = await artistsCollection.find({ name: { $in: favouriteArtistsnames } }).toArray();

//with this we can use the image(base64) from the database
Complexity is 4 Everything is cool!
allFavouriteArtists = allFavouriteArtists.map(artist => {
  if ([artist.image && artist.image.buffer]) {
    artist.image = `data:image/jpeg;base64,${artist.image.buffer.toString('base64')}`;
  }
  return artist;
});

```

Figure 38: Επιστροφή των αγαπημένων

## 7. Ερωτήματα Ήχου

Το ερώτημα αυτό παίρνει το τραγούδι από την βάση όπου και είναι αποθηκευμένο ως binary και μέσω ενός buffer το μετατρέπει σε μορφή που μπορεί να την επεξεργαστεί η react, έτσι ώστε να καθίσταται δυνατό το άκουσμα του τραγουδιού (Figure 39)

```

const songs = client.db("EchoBond").collection("songs")
const song = await songs.findOne({ title: songTitle },
  {projection: { data: 1 }})
}

const audioData = song.data.buffer; // Get the buffer from the Binary data
const buffer = Buffer.from(audioData)

return buffer

```

Figure 39: Επιστροφή ήχου

## 8. Απλά ερωτήματα GET

Αυτά τα ερωτήματα είναι μερικά όπως η επιστροφή όλων των καλλιτεχμών ή όλην των αλμπουμ από την βάση. Τα ερωτήματα αυτά δεν περιέχουν κάτι περισσότερο από ενα απλό find query το οποίο επιστρέφει ολόκληρο τον πίνακα που ζητείται από την βάση (Figure 40).

```

//finding the client and the right collection
const artists = client.db("EchoBond").collection("Artists")
//returns all the artists
allArtists = await artists.find({}).toArray();
Complexity is 4 Everything is cool
allArtists = allArtists.map(artist => {
  if (artist.image && artist.image.buffer) {
    // Assuming the binary data is in a Buffer; adjust as necessary based on your actual data structure
    artist.image = `data:image/jpeg;base64,${artist.image.buffer.toString('base64')}`;
  }
  return artist;
});

```

Figure 40: Επιστροφή όλων των καλλιτεχνών

## 6 Υλοποίηση των ερωτημάτων CRUD στη ΒΔ

Μερικα σημαντικά ερωτήματα για την λειτουργία της εφαρμογής και για την διευκόλυνση του χρήστη είναι τα ερωτήματα CRUD. Παρακάτω αναφερόμαστε πάνω σε αυτά που χρησιμοποιήθηκαν για τις ανάγκες της εργασίας.

### 1. Δημιουργία Λογαριασμού

Αυτό το request εξασφαλίζει ότι τα δεδομένα του νέου χρήστη εισάγονται με ασφάλεια στη βάση δεδομένων αφού ελεγχθεί η εγκυρότητα του OTP και χρυπτογραφηθεί ο κωδικός πρόσβασης. Όπως φαίνεται και από figure xx, λαμβάνει τα δεδομένα του χρήστη από το αίτημα (username, password, email, insertedOTP) και μετά τον έλεγχο του OTP γίνεται μια απλή εισαγωγή στο Collection Users της mongodb (Figure 41).

```

try {
  if (!(await CheckOTP({ email, insertedOTP }))) {
    res.status(404).json({ Register: "Check OTP error" })
    return
  }

  // Connect in the database.
  await client.connect()

  const users = client.db("EchoBond").collection("Users")

  // Encrypt password.
  let encryptedPassword = await bcrypt.hash(password, 10)

  // Insert new user in the database.
  users.insertOne({ username: username, password: encryptedPassword, email: email })

  res.status(200).json({ Register: "Register OK" })
} catch (error) {
  res.status(500).json({ error: "Register error" })
}

```

Figure 41: Εγγραφή στην εφαρμογή

## 2. Είσοδος στο λογαριασμό

Αυτός το request εξασφαλίζει ότι ο χρήστης μπορεί να συνδεθεί αν τα διαπιστευτήρια είναι σωστά, και δημιουργεί ένα access token και ένα refresh token για τον έλεγχο ταυτότητας και την ανανέωση της συνεδρίας. Πιο αναλυτικά λαμβάνει το όνομα χρήστη και τον κωδικό πρόσβασης από το αίτημα και τον αναζητά στη βάση δεδομένων ενώ ελέγχει και τον κωδικό πρόσβασης (Figure 42).

```

const users = client.db("EchoBond").collection("Users")
const user = await users.findOne({ username: username })

// If there is no user with this username,
// or the password is not correct.
if (!user || !await bcrypt.compare(password, user.password)) {
  res.status(404)
  return
}

// Make a new token.
const payload = {
  username: username
}

let accessToken = jwt.sign(payload, process.env.ACCESS_TOKEN_SECRET, { expiresIn: '1h' })
let refreshToken = jwt.sign(payload, process.env.REFRESH_TOKEN_SECRET, { expiresIn: '7d' })

```

Figure 42: Είσοδος στο λογαριασμό

## 3. Αλλαγή κωδικού πρόσβασης του χρήστη

Στη συνέχεια έχει δημιουργηθεί ένα ερώτημα που διασφαλίζει ότι ο

χρήστης μπορεί να αλλάξει τον κωδικό πρόσβασης του αφού ελεγχθεί η εγκυρότητα του παλιού κωδικού πρόσβασης και κρυπτογραφηθεί ο νέος κωδικός πρόσβασης. Αναλυτικότερα, λαμβάνει τον παλιό και τον νέο κωδικό πρόσβασης από το αίτημα, συνδέεται με τη βάση δεδομένων και ελέγχει αν υπάρχει ο χρήστης και αν ο παλιός κωδικός πρόσβασης είναι σωστός. Εντέλει κρυπτογραφεί τον νέο κωδικό πρόσβασης και ενημερώνει τον κωδικό πρόσβασης του χρήστη στη βάση δεδομένων.

```
const users = client.db("EchoBond").collection("Users");
const user = await users.findOne({ username: username });

if (!user) {
  return res.status(404).json({ error: "User not found" });
}

// Check if the current password is correct
const passwordIsValid = await bcrypt.compare(oldPassword, user.password);
if (!passwordIsValid) {
  return res.status(403).json({ error: "Current password is incorrect" });
}

// Encrypt the new password
const hashedPassword = await bcrypt.hash(newPassword, 10);

// Update the user's password
await users.updateOne(
  { username: username },
  { $set: { password: hashedPassword } }
);
```

Figure 43: Αλλαγή κωδικού πρόσβασης

#### 4. Αλλαγή email του χρήστη

Αυτή είναι μια συνάρτηση για την αλλαγή του email ενός χρήστη. Αυτή η συνάρτηση συνδέεται με τη βάση δεδομένων, ενημερώνει το email του χρήστη στη βάση δεδομένων, και επιστρέφει true αν η ενημέρωση ήταν επιτυχής και false σε περίπτωση σφάλματος ή αν δεν βρέθηκε ο χρήστης. Στην συνέχεια πηγαίνει στο αρχείο που την κάλεσε, που σε συνδυασμό με το changeCountry σε περίπτωση που ο χρήστης έχει αλλάξει χώρα ανανεώνονται συνολικά τα στοιχεία (Figure 44).

```

const users = client.db("EchoBond").collection("Users");

// Update the user's photo in the database
const updateResult = await users.updateOne(
  { username: username },
  { $set: { email: newEmail } }
);

if (updateResult.matchedCount === 0) {
  return false
}

```

Figure 44: Αλλαγή κωδικού πρόσβασης

## 5. Διαγραφή του προφίλ χρήστη

Αυτός ο κώδικας (Figure 45,46) αποτελεί μια συνάρτηση για τη διαγραφή χρήστη από τη βάση δεδομένων, μαζί με όλα τα σχετικά δεδομένα του. Η συνάρτηση χρησιμοποιεί JWT για τον έλεγχο ταυτότητας του χρήστη που επιλύεται τη διαγραφή. Αρχικά, ελέγχει αν υπάρχει το token στο αίτημα και το επαληθεύει. Στη συνέχεια, συνδέεται με τη βάση δεδομένων και διαγράφει τα δεδομένα του χρήστη από διάφορες συλλογές όπως "Users", "Friends", "ListenedHistory", "Favorites", "Conversations" και "History". Για τις συνομιλίες (Conversations), συλλέγει τα IDs των συνομιλιών στις οποίες συμμετείχε ο χρήστης και διαγράφει όλες τις σχετικές εγγραφές από τη συλλογή "Messages". Τέλος, επιστρέφει μια επιτυχημένη απόκριση αν η διαγραφή ολοκληρωθεί χωρίς σφάλματα ή καταγράφει και επιστρέφει το σφάλμα αν υπάρχει κάποιο πρόβλημα κατά τη διαδικασία διαγραφής.

```

let token = req.headers.authorization;

if (!token) {
  return res.status(401).json({ error: "Unauthorized access" });
}
token = token.split(' ')[1];
const decode = jwt.verify(token, process.env.ACCESS_TOKEN_SECRET);
let username = decode.username;

const db = client.db("EchoBond");

// Collections where the user's data might be present
const collections = ["Users", "Friends", "Listened_History", "Favorites", "Conversations", "History"];

```

Figure 45: Διαγραφή λογαριασμού(1)

```

for (const collectionName of collections) {
    const collection = db.collection(collectionName);

    // Remove documents where the username matches
    if (collectionName === "Users") {
        await collection.deleteOne({ username: username });
    } else if (collectionName === "Friends") {
        await collection.deleteMany({
            $or: [
                { usernameMain: username },
                { usernameSec: username }
            ]
        });
    } else if (collectionName === "Conversations") {
        // Find and collect conversation IDs
        const conversations = await collection.find(
            { $or: [{ username1: username }, { username2: username }] },
            { projection: { _id: 1 } }
        ).toArray();

        conversationIds = conversations.map(conversation => conversation._id);

        // Delete conversations where the user is a participant
        await collection.deleteMany(
            { $or: [{ username1: username }, { username2: username }] }
        );
    } else {
        await collection.deleteMany({ username: username });
    }
}

if (conversationIds.length > 0) {
    const mesCollection = db.collection("Messages");

    // Delete all the messages from the conversations of the user
    await mesCollection.deleteMany({ conversationID: { $in: conversationIds } });
}
}

```

Figure 46: Διαγραφή λογαριασμού(2)

## 7 Φόρτωση των δεδομένων στη ΒΔ

Τα δεδομένα τα οποία υπάρχουν στην βάση είναι έτοιμα δεδομένα από το διαδίκτυο, δηλαδή πληροφορίες για τα αλμπουμ και τους καλλιτεχνες, όπως και πληροφορίες για τα τραγούδια. Τα δεδομένα αυτά, περάστηκαν στη βάση χειροκίνητα με τους παρακάτω κώδικες (Figure 41, 42, 43), οι οποίοι τοποθετήθηκαν στο Server. Επιπρόσθετα, υπάρχει και δυνατότητα εισαγωγής των δεδομένων κατευθείαν από την πλατφόρμα της MongoDB.

Αρχικά για το πέρασμα των αλμπουμ τρέχει ένα απλό Insert query, το οποίο παίρνει την φωτογραφία του άλμπουμ από ένα συγκεκριμένο path και την μετατρέπει σε ενα συγκεκριμένο μέγεθος, έτσι ώστε να μήν πιάνει πολύ χώρο στη βάση. Επίσης μειώνεται ελάχιστα η ποιότητα για τον ίδιο σκοπό (Figure 47).

```
// IMPORT ALBUMS
try{
  const file = fs.readFileSync('./photo/Eurovision2024.png')

  const processedImage = await sharp(file)
  .resize(100, 100) // Resize to 100x100 pixels
  .jpeg({ quality: 80 }) // Convert to JPEG with 80% quality
  .toBuffer(); // Convert to a buffer for insertion

  await client.connect()
  const albums = client.db("EchoBond").collection("Albums")
  await albums.insertOne({title:"Eurovision 2024",artist:"Many Artists",year:"2024",style:"Mix",image:processedImage})
}

catch (error){
  console.log(error)
}
```

Figure 47: Εισαγωγή αλμπουμ στη βάση

Με τον ίδιο ακριβώς τρόπο μπορούν να περαστούν στη βάση και τα δεδομένα για τους καλλιτέχνες, με την μόνη αλλαγή στο insert query είναι τα διαφορετικά πεδία (Figure 48).

```
// IMPORT ARTISTS
try{
  const file = fs.readFileSync('./photo/Argyros.png')

  const processedImage1 = await sharp(file)
  .resize(100, 100) // Resize to 100x100 pixels
  .jpeg({ quality: 80 }) // Convert to JPEG with 80% quality
  .toBuffer(); // Convert to a buffer for insertion

  await client.connect()
  const albums = client.db("EchoBond").collection("Artists")
  await albums.insertOne({name:"Argyros Konstantinos",style:"Pop",image:processedImage1})
}

catch (error){
  console.log(error)
}
```

Figure 48: Εισαγωγή καλλιτεχνών στη βάση

Τέλος για την ολοκλήρωση των βασικών δεδομένων και για την λειτουργία της εφαρμογής πρέπει να προστεθούν και τα τραγούδια, τα οποία πέρα από την φωτογραφία έχουν ωσ πεδίο και τον καταγεγραμένο ήχο ο οποίο αργότερα χρησιμοποιείται με audio display για το άκουσμα του τραγουδιού (Figure 49).

```
// IMPORT Songs
try{
  const file = fs.readFileSync('./Photos/Tali.png')

  const processedImage = await sharp(file)
    .resize(100, 100) // Resize to 50x50 pixels
    .jpeg({ quality: 99 }) // Convert to JPEG with 99% quality
    .toBuffer(); // Convert to a buffer for insertion

  const song = fs.readFileSync('./Photos/Fighter.mp3')
  const processedSong = Buffer.from(song)

  // console.log(processedSong)

  await client.connect()
  const songs = client.db("EchoBond").collection("Songs")
  await songs.insertOne({title:"Fighter", artist:"Tali", album:"Eurovision 2024", time:"3:12" ,image:processedImage, data:processedSong})
  console.log("inserted")
}
catch (error){
  console.log(error)
}
```

Figure 49: Εισαγωγή τραγουδιών στη βάση

Στις παρακάτω εικόνες φαίνονται αναλυτικά όλα τα Collection τα οποία έχουν τοποθετηθεί στην βάση, αλλά και ενα παράδειγμα εγγραφών για το καθε ένα.

- **Collections**

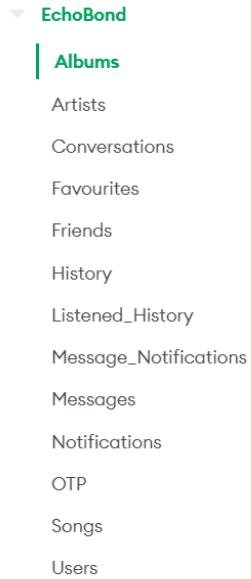


Figure 50: Τα collections

- **Albums**

Οι εγγραφές στο Collection Albums εμφανίζονται ως εξής:

The screenshot shows the Compose interface with the 'Albums' collection selected. The search bar contains the query 'Type a query: { field: 'value' }'. The results are displayed as follows:

```

QUERY RESULTS: 1-2 OF 2

_id: ObjectId('60484350802b46a68034dc1')
name: "Eurovision 2024"
artist: "Many Artists"
year: "2024"
style: "Pop"
image: Binary.createFromBase64('...')/9j/2wBDAAMCAjRCAjAgDAAuMEAwMEQFBQkE8QjQhBwYTDaaMDAaKwIaNDIaIQ48DgSLByQEMuFRUVDA8XG...

```

Figure 51: Εγγραφες σε Albums

• **Artists** Οι εγγραφές στο Collection Artists εμφανίζονται ως εξής:

The screenshot shows the Compose interface with the 'Artists' collection selected. The search bar contains the query 'Type a query: { field: 'value' }'. The results are displayed as follows:

```

QUERY RESULTS: 1-1 OF 1

_id: ObjectId('6640d56b971068952d7f917')
name: "Argyros Konstantinos"
style: "Pop"
image: Binary.createFromBase64('...')/9j/2wDAAVEQBYFBAYGBQYBwYICAKpk3chQDowFvQYGBcUFhYahTsUfGhsJhBvN1Ceg1yrmScp6RlM09o...

```

Figure 52: Εγγραφες σε Artists

• **Conversation**

Οι εγγραφές στο Collection Conversation εμφανίζονται ως εξής:

The screenshot shows the Compose interface with the 'Conversation' collection selected. The search bar contains the query 'Type a query: { field: 'value' }'. The results are displayed as follows:

```

QUERY RESULTS: 1-2 OF 2

_id: ObjectId('6648b869f858ca7d8982bbff')
username1: "dtihagay"
username2: "randronikou"

```

Figure 53: Εγγραφες σε Conversation

• **Friends**

Οι εγγραφές στο Collection Friends εμφανίζονται ως εξής:

Figure 54: Εγγραφες σε Friends

### • History

Οι εγγραφές στο Collection History εμφανίζονται ως εξής:

Figure 55: Εγγραφες σε History

### • Listened History

Οι εγγραφές στο Collection Listened History εμφανίζονται ως εξής:

Figure 56: Εγγραφες σε Listened History

### • Users

Οι εγγραφές στο Collection Users εμφανίζονται ως εξής:



Figure 57: Εγγραφες σε Users

### • OTP

Οι εγγραφές στο Collection OTP εμφανίζονται ως εξής:



Figure 58: Εγγραφες σε OTP

## 8 Επεξήγηση του User Interface της Εφαρμογής

Η κατανόηση του User Interface (UI) μιας εφαρμογής είναι κρίσιμη για τη βέλτιστη χρήση και την αποδοτική πλοήγηση. Το UI της εφαρμογής μας έχει σχεδιαστεί με γνώμονα την ευκολία χρήσης και την άμεση πρόσβαση στις βασικές λειτουργίες. Σε αυτή την ενότητα, θα παρουσιάσουμε μια αναλυτική επεξήγηση κάθε σελίδας της εφαρμογής, επισημαίνοντας τα κύρια στοιχεία και τις λειτουργίες που διαθέτει. Σκοπός μας είναι να παρέχουμε μια ολοκληρωμένη εικόνα του τρόπου με τον οποίο οι χρήστες μπορούν να αξιοποιήσουν στο έπακρο τις δυνατότητες της εφαρμογής.



Figure 59: Αρχική σελίδα

Αρχικά ξεκινώντας από την αρχική σελίδα της ιστοσελίδας (Figure 59) παρατηρείται ότι παρέχεται η δυνατότητα "scroll down", όπου και παρατίθενται μερικά λόγια για την ομάδα που δούλεψε σε αυτό το Project. Επιλέγοντας στην συνέχεια το κουμπί "Sing Up", το οποίο βρίσκεται πάνω δεξιά οδηγούμαστε στην σελίδα της εγγραφής (Figure 60). Στη σελίδα αυτή προχωράμε στην συμπλήρωση των πεδίων της φόρμας που διατίθεται και έπειτα επιλέγουμε το "Sign Up".

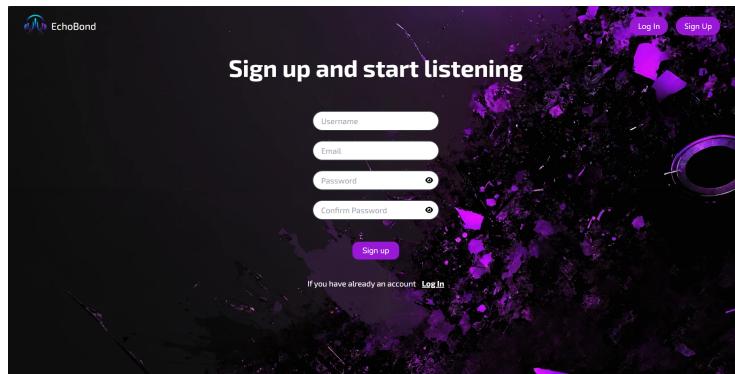


Figure 60: Φόρμα εγγραφής

Στην συνέχεια, οδηγούμαστε στην σελίδα του One Time Password (OTP), όπως φαίνεται στο Figure 61, όπου και πρέπει να συμπληρώσουμε τον 4ψήφιο κωδικό, ο οποίος στάλθηκε στο email που τοποθετήσαμε στην φόρμα στην προηγούμενη σελίδα. Αν για κάποιο λόγο δεν προλάβουμε να συμπληρώσουμε

τον κωδικό OTP μπορούμε να επιλέξουμε την επαναποστολή με το κουμπί "resend OTP". Τέλος πατάμε το "Check OTP" και αν δεν λάβουμε κάποιο μήνυμα λάθους η διαδικασία εγγραφής ολοκληρώνεται και οδηγούμαστε αυτόματα στην σελίδα Log In (Figure 62). Σε αυτό το σημείο άξιο σημείωσης είναι το ότι με την εγγραφή του χρήστη στην βάση ο κωδικός του περνάει από μερικά στάδια encryption μέσω hashing, τα οποία βελτιώνουν την ασφάλεια του λογαριασμού του.

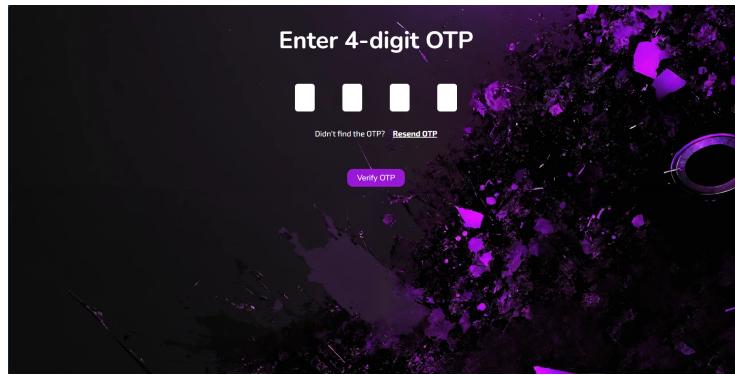


Figure 61: OTPI

Τώρα, ως εγγεγραμένος χρήστης μπορούμε να προχωρήσουμε στην σύνδεση στον λογαριασμό μας τοποθετώντας τα στοιχεία μας στη φόρμα που παρατίθεται στην οθόνη μας και επιλέγομε το "Log In", έτσι ώστε να επιβεβαιωθούν τα στοιχεία μας και έπειτα με αυτόματη ανανέωση, αν όλα πήγαν καλά, οδηγούμαστε στην κεντρική σελίδα της ιστοσελίδας.

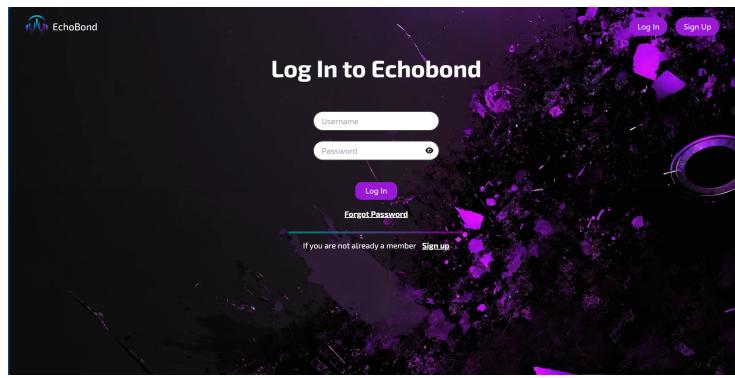


Figure 62: Φόρμα Συνδεσης

Τα πρώτα πρόγματα που παρατηρούνται στην κεντρική σελίδα(Figure 63) είναι αρχίκα πάνω δεξιά το εικονίδιο το χρήστη και στο κεντρικό χώρο της παρατηρούνται δύο ”στήλες”. Η αριστερή στήλη χωρίζεται σε δύο σειρές , όπου στην πρώτη φαίνεται ένα πλάσιο που περιέχει το HomeButton και το SearchButton. Στη δεύτερη σειρά της αριστερής στήλης παρατηρούνται οι φίλοι που αρχικά παρουσιάζεται ενα άδειο πλαίσιο ,διότι ο χρήστης δεν έχει φίλους. Για την δεξιά στήλη παρατηρούμε στην κορυφή τέσσερις επιλογές , αυτές είναι το ”All”, το ”Artists”, το ”Albums” και τέλος το ”Favourites”.

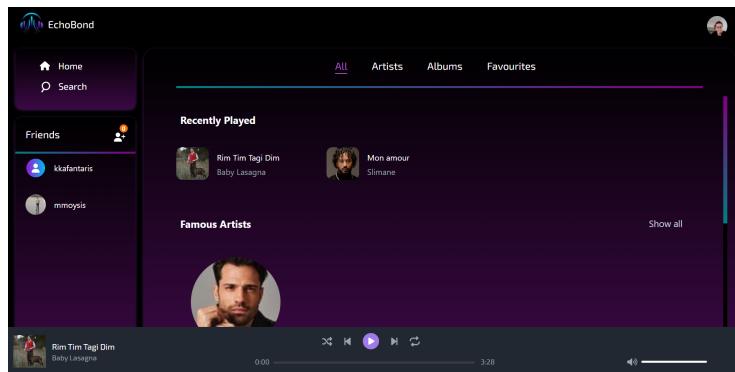


Figure 63: Κεντρική σελίδα

Ας ξεκινήσουμε τώρα με την ανάλυση όλων αυτών που αναφέραμε παραπάνω αρχίζοντας με την λειτουργία του κουμπιού του χρήστη που βρίσκεται πάνω δεξιά στο Navbar.

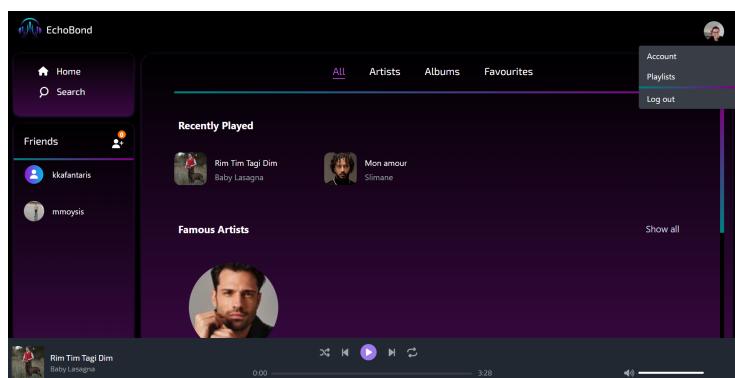


Figure 64: Επιλογες χρήστη

Πατώντας το κουμπί αυτό εμφανίζεται μια λίστα με τρείς επιλογές (Figure 64). Επιλέγοντας log out, ο χρήστης αποσυνδέεται από το προφίλ του και ανακατευθύνεται στην αρχικά σελίδα. Επιλέγοντας το κουμπί Account, ο χρήστης μεταφέρεται στην σελίδα του προφίλ του (Figure 65).

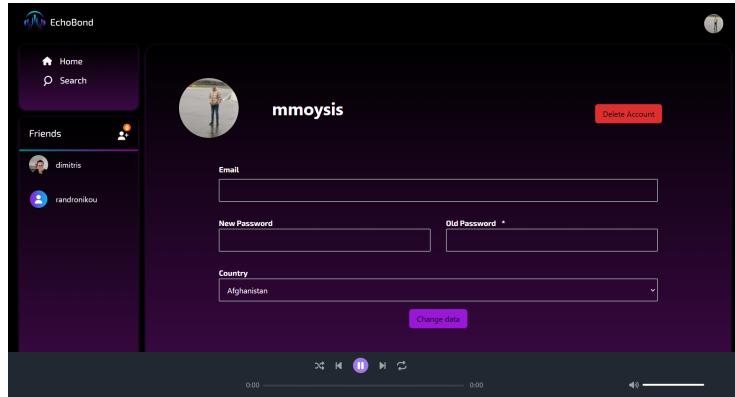


Figure 65: Προφίλ χρήστη

Σε αυτή την σελίδα ο χρήστης έχει την δυνατότητα να αλλάξει τα δικα του στοιχεία συμπληρώνοντας οποιαδήποτε πεδία θέλει και πατώντας το κουμπί "Change data", αναγκαία είναι η εισαγωγή του παλιού κωδικού για να λειτουργήσει η αντικατάσταση των δεδομένων. Επιπλέον μπορεί να αλλάξει και την εικόνα του προφίλ του πατόντας απλά πάνω στο κατάλληλο πεδίο.

Στη συνέχεια μπορούμε να αναφερθούμε στη δεξιά στήλη της κεντρικής σελίδας στην οποία περιέχονται οι 4 επιλογές όπως προαναφέρθηκε. Αρχικά στο home page βρισκόμαστε στην επιλογή "All" όπου εμφανίζονται τα πρόσφατα τραγούδια που άκουσε ο χρήστης, πεδίο που όμως είναι αρχικά άδειο μεχρι να πάξει κάποιο τραγούδι (Figure 63). Κάτω από αυτό εμφανίζονται μερικοί καλλιτέχνες με τυχαία επιλογή από την βάση και στην συνέχεια εμφανίζονται μερικά αμπουμς. Όλα αυτά που προαναφέργηκαν έχουν σημουργηθεί ως ξεχωριστά components με την λειτουργία επιλογής. Σε περίπτωση επιλογής ενός τραγουδιού αυτό παίζει απευθείας αυτόματα. Επιπρόσθια σε επιλογή ενός καλλιτέχνη ή ενός αλμπουμ υπάρχει ανακατεύθυνση στην κατάλληλη σελίδα για το κάθε ενα.

Η σελίδα που ανακατευθύνεται ο χρήστης όταν επιλέγει ενα καλλιτέχνη φαίνεται στο Figure 66, αυτή είναι η σελίδα από εδώ και πέρα που όμως ανακατευθύνεται ο χρήστης από όποια άλλη σελίδα και αν επιλέγει έναν καλλιτέχνη. Πιο αναλυτικά σε αυτή την σελίδα εμφανίζονται οι πληροφορίες του καλλιτέχνη, τα τραγούδια του που υπάρχουν στην βάση με το όνομά του όλα και τα αλμπουμ που είναι δικά του. Πάλι με τον ίδιο τρόπο αν επιλέξουμε κάποιο τραγούδι παίζει απευθείας και με την επιλογή κάποιου Αλμπουμ ανακατευθύνομαστε όπως αναφέρθηκε και

πιο πάνω στην σελίδα των του αλμπουμ.

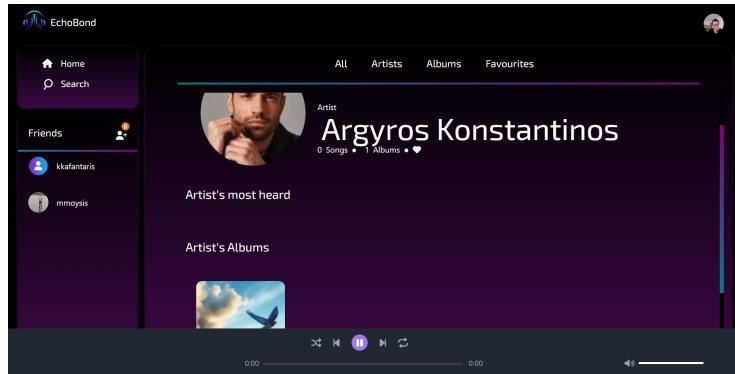


Figure 66: Σελίδα καλλιτέχνη

Από οποιudήποτε και άν επιλέξουμε ένα αλμπουμ θα ανακατευθυνθούμε στην σελίδα του αλμπουμ όπως φαίνεται στο Figure 67. Σε αυτή την σελίδα εμφανίζονται τα δεδομένα του αλμπουμ και έπειτα τα τραγούδια που ανήκουν σε αυτό σύμφωνα με τις πληροφορίες που έχουν περαστεί στην βάση.

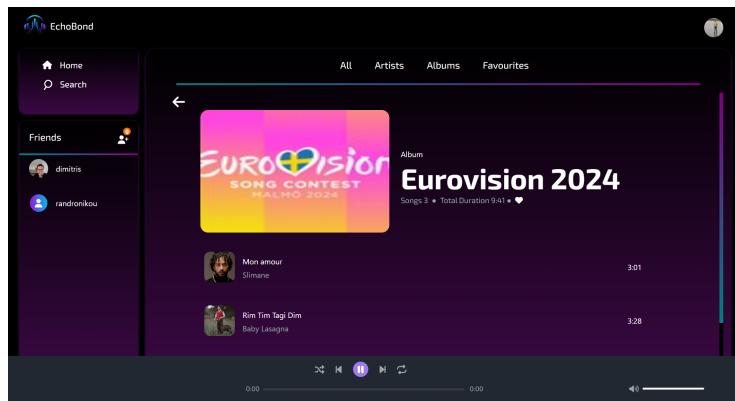


Figure 67: Σελίδα αλμπουμ

Και στις δύο σελίδες αυτές υπάρχει μια "χαρδία" δίπλα από τις πληροφορίες του αλμπουμ ή του καλλιτέχνη και μπορεί ο χρήστης να την επιλέξει για να προσθέσει στα αγαπημένα του κάποιο αλμπουμ ή κάποιον καλλιτέχνη. Η σελίδα των αγαπημένων φαίνεται στο Figure 68 και περιέχει απλά τα αγαπημένα αλμπουμ και καλλιτεχνες που έχει προσθέση ο χρήστης κατα την διάρκεια της

πειήγησής του στην ιστοσελίδα

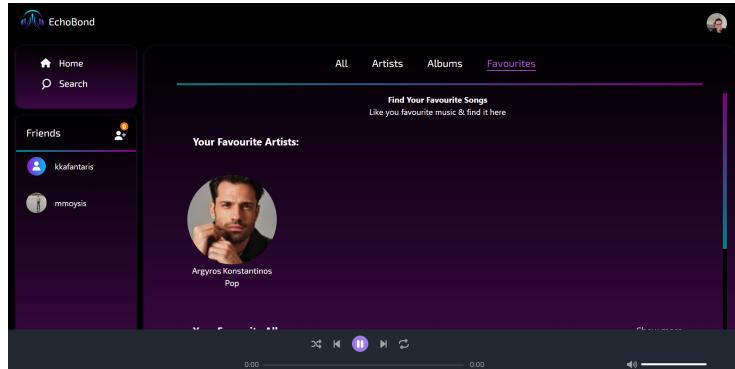


Figure 68: Σελίδα αγαπημενων

Έπειτα μπορούμε να αναφερθούμε στην σελίδα των ”Artists” η οποία εντωπίζει και εμφανίζει όλους του καλλιτέχνες που υπάρχουν στην ιστοσελίδα όπως φαίνεται ενδικτειακά στο Figure 69. Όπου για ακόμα μια φορά αν επιλεχθεί ένας καλλιτέχνης γίνεται ανακατεύθυνση στην σελίδα του καλλιτέχνη.

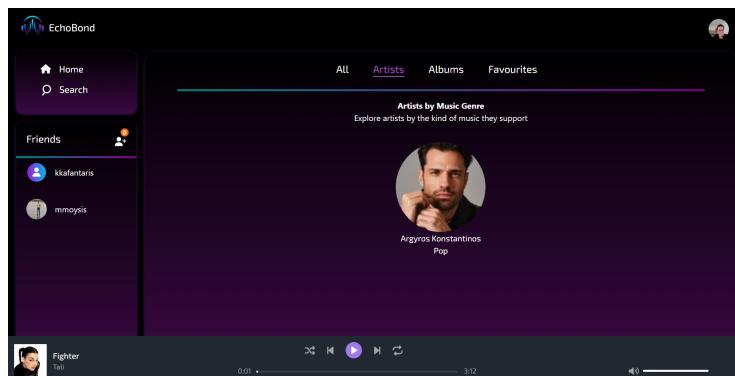


Figure 69: Σελίδα Artists

Στη συνέχεια επιλέγοντας από την κεντρική σελίδα το ”Albums” εμφανίζονται όλα τα διαέσιμα άλμπουμ,όπως φαίνεται στο Figure 70. Με την επιλογή κάποιου οδηγούμαστε στην σελίδα του όπου και περιέχονται τα τραγούδια του .

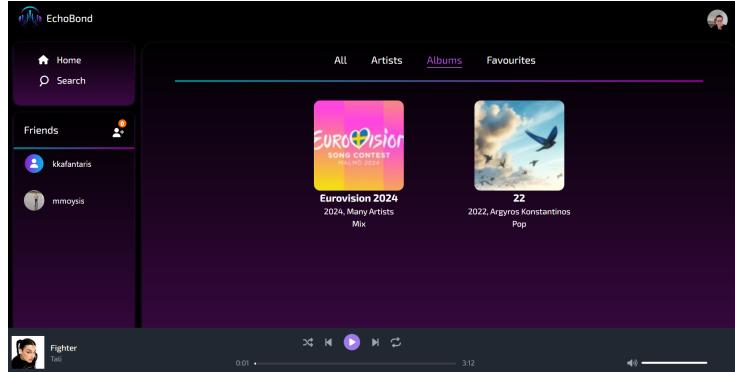


Figure 70: Σελίδα Albums

Τώρα σε περίπτωση που ο χρήστης θέλει να κάνει κάποιο search για να εντοπίσει ένα τραγούδι ή καλλιτέχνη ή κάποιο άλμπουμ που επιθυμεί. Σε αυτή την περίπτωση πατάει στην αρχική οισόνη το κουμπί Search πάνω αριστερά (Figure 63), το οποίο τον οδηγεί στην συγκεκριμένη σελίδα. Αρχικά, εμφανίζεται το search bar για να δώσει ο χρήστης το όνομα από όπου φάχνει καθώς και κάποια στοιχεία από κάτω τα οποία είχε κανεί search στο παρελθόν και είχε πατήσει (Figure 71).

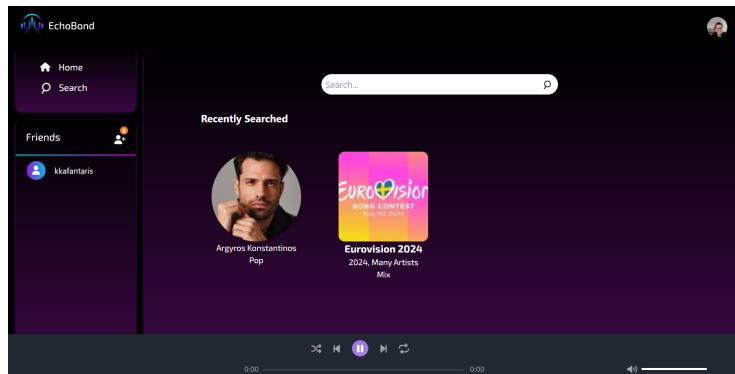


Figure 71: Σελίδα Search recent

Στη συνέχεια, αφού ο χρήστης πατήσει το κουμπί search button από το bar μαζί με το όνομα που αναζητεί, η ιστοσελίδα θα του εμφανίσει όλα τα πιθανά τραγούδια ή ακαλλιτέχνες ή άλμπουμ τα οποία τερίαζουν (εν μέρει ή ακριβώς) με το ζητούμενο (Figure 72). Έτσι μπορεί να πατήσει όποιο αυτός επιθυμεί και σε

περίπτωση που ξαναπάει στη σελίδα του search bar θα του εμφανιστεί σαν προτεινόμενο.

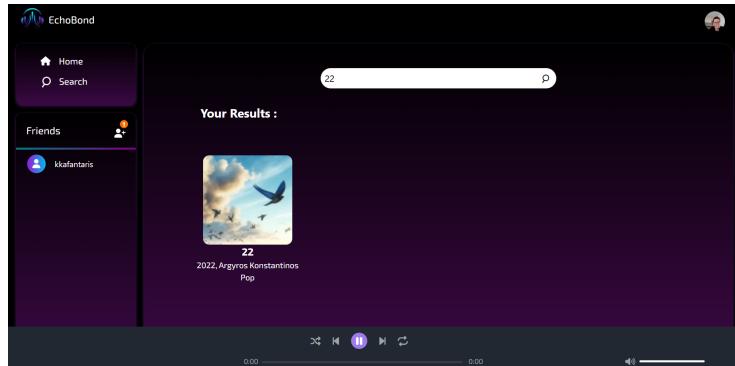


Figure 72: Σελίδα Αναζήτησης

Στην αρχική οθόνη, ο χρήστης μπορεί να εντοπίσει κάτω αριστερά τους φίλους του (Friends). Με αυτούς τους φίλους μπορεί να μιλήσει αποκλειστικά και σε οποιαδήποτε κατάσταση βρίσκεται στην ιστοσελίδα. Τώρα σε περίπτωση που ο χρήστης θέλει να αποκτήσει καινούργιους φίλους θα πρέπει να πατήσει το κουμπί που βρίσκεται πάνω δεξιά στο κουτάχι των φίλων. Στη συνέχεια, θα αλλάξει η εικόνα των φίλων και θα εμφανιστεί ένα search bar (Figure 73), τα αιτήματα από άλλους χρήστες που θέλουν να δημιουργήσουν μια φιλία με τον χρήστη, ενώ από κάτω θα εμφανιστούν κάποιοι πιθανοί φίλοι από την βάση

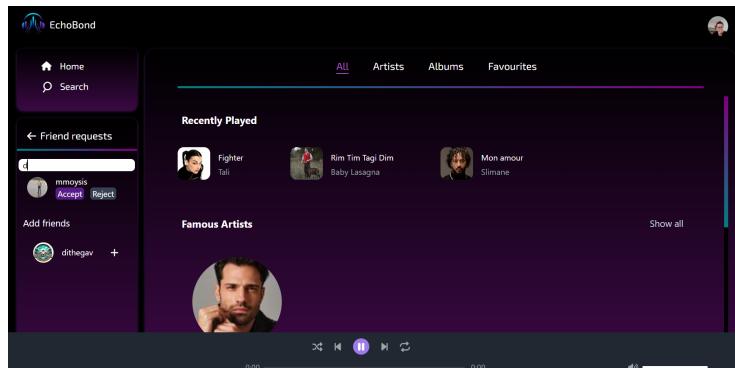


Figure 73: Φίλοι

. Εάν ο χρήστης θέλει να εντοπίσει έναν συγκεκριμένο φίλο μπορεί να αρχίσει να

γράφει στο πεδίο του search που βρίσκεται στο κουτί που αναφέραμε και αυτό θα εμφανίζει αποτελέσματα ζωντανά όσο πληκτρολογεί, χωρίς να πατηθεί κάποιο κουμπί search button. Έτσι ο χρήστης μπορεί να στείλει ένα αίτημα φιλίας στο συγκεκριμένο χρήστη ο οποίος θα ειδοποιηθεί εντός εφαρμογής για το αίτημα και θα περιμένει απάντηση. Το ίδιο και με οποιονδήποτε χρήστη, δίνεται η δυνατότητα να γίνει Accept είτε Reject το αίτημα φιλίας (Figure 73) το οποίο ανανεώνει μετά την επιλογή την λίστα. Σε περίπτωση θετικής απάντησης ο χρήστης μπορεί να γυρίσει πίσω στο κατάλογο των φίλων του και να εντοπίσει την καινολυργία του φίλα. Κάθε χρήστης μπορεί να διαγράψει οποιαδήποτε στιγμή κάποιο φίλο απλά πηγαίνοντας το ποντίκι πάνω στον φίλο που επιθυμεί και επιλέξει το εικονίδιο του ”κάδου απορριμάτων” έτσι ώστε να διαγραφεί ο επιλεγμένος φίλος (Figure 74).

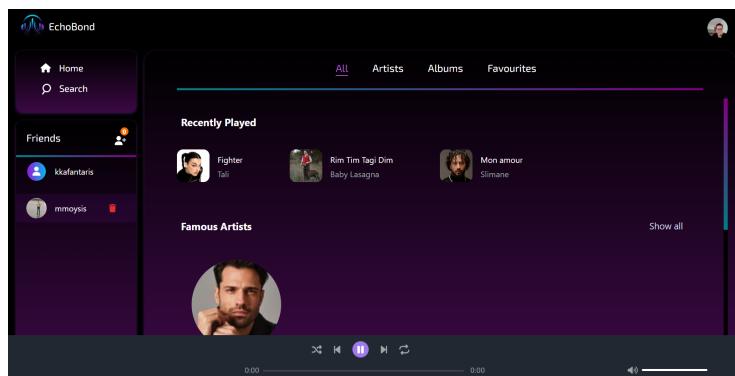


Figure 74: Διαγραφή φίλου

Τώρα, όσον αναφορά το chatting μεταξύ φίλων, ο χρήστης ανά πάσα στιγμή μπορεί να πατήσει το συγκεκριμένο φίλο και να εμφανιστεί στη κεντρική σελίδα τα μηνύματα που έχουν ανταλλάξει νωρίτερα ή άδειο και αναμένει καινούργια μηνύματα από τους δύο (Figure 75). Για να στείλει μήνυμα ο χρήστης πατάει κατω στην σελίδα το search bar που εμφανίζεται γράφει το μήνυμά του και πατάει στο σύμβολο ακριβώς δεξιά, αποστολή. Ο φίλος του θα ειδοποιηθεί για το μήνυμα εντός εφαρμογής για να απαντήσει (το ίδιο ισχύει προφανώς και για τον ίδιο τον χρήστη).

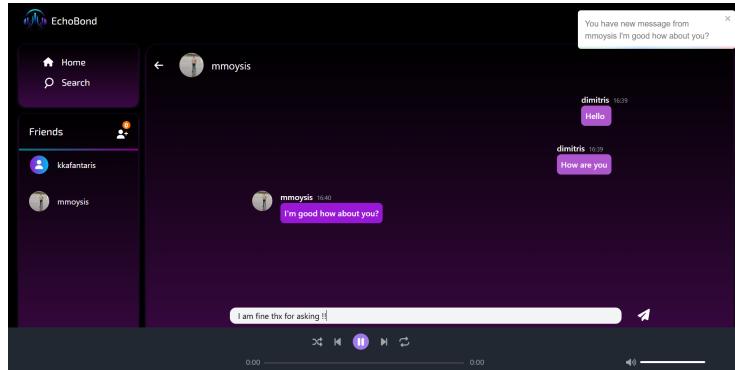


Figure 75: Συνομιλία φίλων

## 9 Μελλοντικές επεκτάσης της εφαρμογής

Μια από τις βασικές μελλοντικές επεκτάσης της εφαρμογής μας πιστεύουμε οτι θα είναι το κοινό άκουσμα τραγουδιών μεταξύ φίλων, το οποίο έχει ως σκοπό να έρθουν πιο κοντά και οι φίλοι που είναι μακριά μέσω της μουσικής.

## 10 Απεσταλμένα αρχεία

Τα αρχεία που στέλνουμε βρίσκονται σε δύο φακέλους. Ο πρώτος είναι ο φάκελος Client που περιέχει όλα τα αρχεία που έχουν να κάνουν με το Front-end και ο δεύτερος είναι ο φάκελος Server ο οποίος περιέχει όλα τα αρχεία σχετικά με το server της εφαρμογής.

## 11 Βιβλιογραφία

- <https://www.udemy.com/>
- <https://www.coursera.org/>
- <https://stackoverflow.com/questions/16423150/socket-io-subscribe-to-multiple-channels>
- <https://kb.objectrocket.com/mongo-db/how-to-store-audio-files-in-mongodb-396>
- <https://stackoverflow.com/questions/55230048/passing-audio-from-mongodb-to-audio-tag>
- <https://nodejs.org/api/net.html#class-netsocket>

- [https://medium.com/@yelee2369/  
node-js-streaming-audio-files-10dd5e8670d0](https://medium.com/@yelee2369/node-js-streaming-audio-files-10dd5e8670d0)
- [https://dev.to/saint\\_vandora/  
how-to-integrate-nodejs-with-nosql-databases-3b26](https://dev.to/saint_vandora/how-to-integrate-nodejs-with-nosql-databases-3b26)