Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Σχεδιασμός και ανάπτυξη ηλεκτρονικού καταστήματος με την χρήση καινοτόμων τεχνολογιών** **Designing and developing an e-commerce store using innovative technologies**
Ονοματεπώνυμο Φοιτητή	**ΑΡΗΣ ΛΑΖΑΡΙΔΗΣ**
Πατρώνυμο	**ΕΥΑΓΓΕΛΟΣ**
Αριθμός Μητρώου	**ΜΠΠΛ/ 20042**
Επιβλέπων	**ΑΛΕΠΗΣ ΕΥΘΥΜΙΟΣ, Αν. Καθηγητής**

Ημερομηνία Παράδοσης **Φεβρουάριος 2024**

Τριμελής Εξεταστική Επιτροπή

Αλέπης Ευθύμιος Αναπληρωτής Καθηγητής	Βίρβου Μαρία Καθηγήτρια	Πατσάκης Κωνσταντίνος Αναπληρωτής Καθηγητής

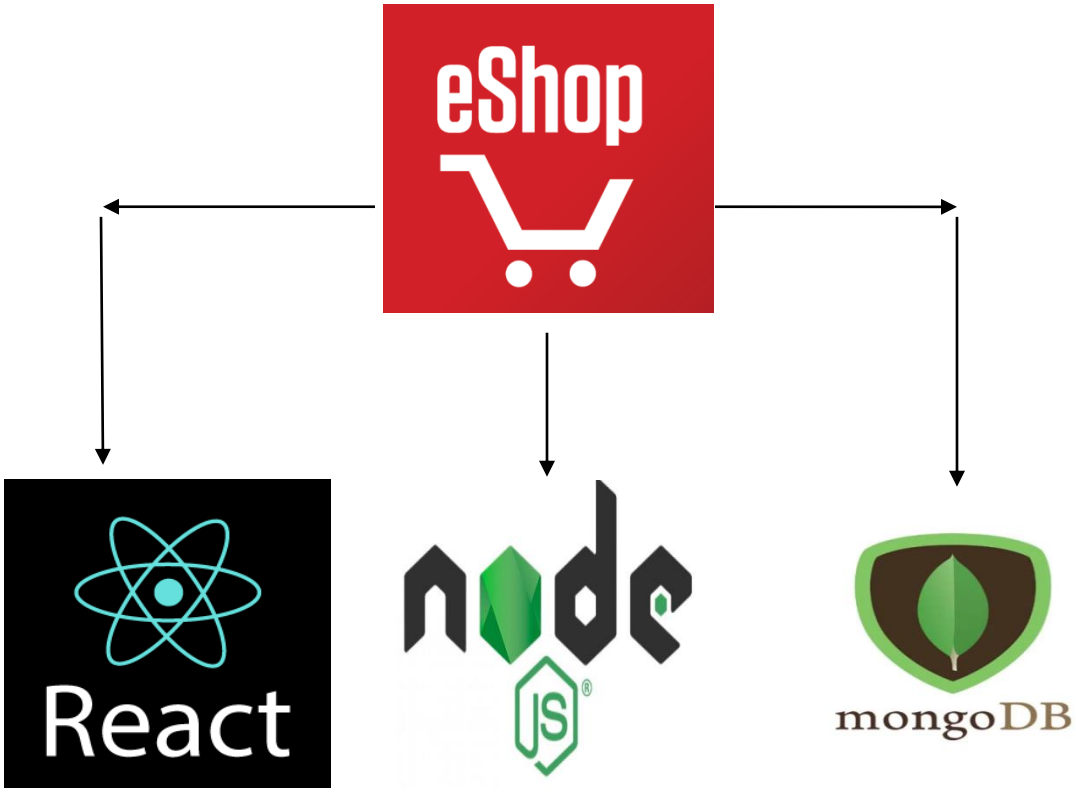
Περιεχόμενα

Περίληψη

Η παρούσα διπλωματική εργασία έχει ως κύριο στόχο την ανάπτυξη ενός ηλεκτρονικού καταστήματος (eshop) με τη χρήση προηγμένων τεχνολογιών. Παρουσιάζονται οι τεχνολογικές επιλογές που εφαρμόστηκαν και η σημασία τους στη δημιουργία μιας αξιόπιστης και αποδοτικής εφαρμογής. Επισημαίνεται η αρχιτεκτονική της εφαρμογής , οι βασικές της λειτουργίες, και περιγράφεται ο τρόπος υλοποίησής της. Επιπλέον, δίνεται έμφαση στην παρουσίαση των καινοτόμων τεχνολογιών που χρησιμοποιήθηκαν και στη συνεισφορά τους στον τομέα του ηλεκτρονικού εμπορίου.

Abstract

The primary goal of this thesis is to develop an electronic store (e-shop) employing advanced technologies. It presents the implemented technological choices and their significance in creating a dependable and efficient application. The thesis emphasizes the application's architecture, outlining its fundamental functions and describing the implementation approach. Furthermore, it focuses on showcasing the innovative technologies utilized and their impact on the e-commerce domain.



Εισαγωγή

Η δημιουργία ενός eshop είναι μια πρόκληση, καθώς απαιτεί την ενσωμάτωση πολλών διαφορετικών απαιτήσεων, όπως η προβολή των διαθέσιμων προιότων, η επικοινωνία, η διαχείριση των παραγγελιών και η διαχείριση των πληρωμών.

Η επιλογή των τεχνολογιών που θα χρησιμοποιηθούν για την ανάπτυξη του eshop παίζουν σημαντικό ρόλο στην αποδοτικότητα του και τον τρόπο διαχείρισης του.

Η χρήση καινοτόμων τεχνολογιών όπως αυτών που θα χρησιμοποιηθούν για την ανάπτυξη του eshop της παρούσας διπλωματικής εργασίας (REACT, NODE.JS, MONGO-DB, REDUX) επιτρέπουν την υλοποίηση μιας αξιόπιστης και αποδοτικής εφαρμογής, και μπορούν να ανταποκριθούν στις απαιτήσεις του σύγχρονου ηλεκτρονικού εμπορίου.

Στο πλαίσιο αυτής της εργασίας θα επισημάνουμε τη σημασία της χρήσης αυτών των τεχνολογιών στην δημιουργία ενός εμπορικού ιστοτόπου, και θα παρουσιάσουμε τις διαφορετικές δυνατότητες που παρέχουν. Επίσης θα εξετάσουμε την αρχιτεκτονική της εφαρμογής που αναπτύχθηκε και θα εξηγήσουμε τον τρόπο που υλοποιήθηκε.

Στόχος της διπλωματικής εργασίας είναι να προσφέρει μια καλή κατανόηση των τεχνολογιών αυτών και του πώς μπορεί να δημιουργηθεί μια λειτουργική και αποτελεσματική εφαρμογή καθώς και να λειτουργήσει ως πηγή έμπνευσης για τους φοιτητές που επιθυμούν να εμβαθύνουν στη χρήση αυτών.

Ανάλυση και Σχεδιασμός

- **Προβολή προϊόντων**
  Οι χρήστες βλέπουν τα διαθέσιμα προϊόντα και τις πληροφορίες αυτών όπως περιγραφή, εικόνα, τιμή και διαθεσιμότητα
- **Προσθήκη προϊόντων στο καλάθι**
  Οι χρήστες μπορούν να προσθέσουν τα προϊόντα στο καλάθι αγορών επιλέγοντας την ποσότητα που επιθυμούν
- **Ολοκλήρωση αγοράς**
  Οι εγγεγραμμένοι χρήστες μπορούν να προβούν στην ολοκλήρωση της αγοράς βάση των στοιχείων που έχουν καταχωρήσει κατά την εγγραφή τους στο σύστημα
- **Εγγραφή νέου χρήστη**
  Οι χρήστες θα μπορούν να δημιουργήσουν έναν λογαριασμό στην πλατφόρμα δίνοντας τις απαραίτητες πληροφοριές που χρειάζονται για να προβούν στην παραγγελία των προϊόντων.
- **Σύνδεση χρήστη**
  Ο χρήστης για να μπορεί να προχωρήσει στην ολοκλήρωση της αγοράς του, θα πρέπει να κάνει login στην εφαρμογή, με το username και password που δημιούργησε κατά την εγγραφή του

Βάση της παραπάνω ανάλυσης πρέπει να δημιουργήσουμε μια διαδικτυακή εφαρμογή που θα αφορά ένα ηλεκτρονικό κατάστημα. Η εφαρμογή θα πρεπει να επιτρέπει στους χρήστες να περιηγηθούν στα διαθέσιμα προϊόντα, να προσθέσουν προϊόντα στο καλάθι αγορών τους, να ολοκληρώσουν την παραγγελία τους καθώς και να μπορούν να συνδεθούν/εγγραφούν στην πλατφόρμα για να παρακολουθούν την εξέλιξη της παραγγελίας τους,

Κατόπιν της ανάλυσης και της καταγραφής των απαιτήσεων της εφαρμογής, θα πρέπει να αξιολογήσουμε τον τρόπο που θα υλοποιηθούν με την χρήση των τεχνολογιών που έχουμε επιλέξει.

Σχεδιασμός Αρχιτεκτονικής

Για την δημιουργία τους ηλεκτρονικού καταστήματος μας, θα πρέπει να οργανώσουμε την αρχιτεκτονική του συστήματος σε 3 κύρια επίπεδα :

1. Front-End
2. Back-End
3. Βάση Δεδομένων

Front-end

Για την ανάπτυξη του Front-End επιλέξαμε την **React** η οποία είναι μια δημοφιλής βιβλιοθήκη της Javascript που χρησιμοποιείται κυρίως για την ανάπτυξη διεπαφών χρήστη (User Interface) σε ιστοσελίδες και εφαρμογές. Η React αναπτύχθηκε από το Facebook και είναι ανοικτού κώδικα, κάτι που την καθιστά προσιτή και δημοφιλή στην κοινότητα των προγραματιστών.

Ορισμένα από τα κύρια χαρακτηριστικά και πλεονεκτήματα της React :

- **Components (Συστατικά)**
  Στη React το UI αναπαρίσταται με τα components. Τα components είναι μικρά και επαναχρησιμοποιήσιμα κομμάτια κώδικα που επιτρέπουν στον προγραμματιστή τον σχεδιασμό της διεπαφής χρήστη σε ανεξάρτητα κομμάτια, καθιστώντας την συντήρηση και την ανάπτυξη νέων λειτουργιών πολύ πιο εύκολη.

- **Virtual Dom (Εικονικό DOM**)
  Η React χρησιμοποιεί το Virtual DOM (Document Object Model) για να παρακολουθεί τις αλλαγές στα δεδομένα και να ενημερώνει την διεπαφή χρήστη αποδοτικά. Ενημερώνει μόνο τα αλλαγμένα στοιχεία και διαχειρίζεται αποτελεσματικά τις αναδημιουργίες διεπαφών. Επιτυγχάνει με αυτόν τον τρόπο υψηλές επιδόσεις και γρήγορη αποδόση ιστοσελίδων.

- **Μονόδρομη ροή δεδομένων**
  Τα δεδομένα μεταφέρονται από πάνω προς τα κάτω στην ιεραρχία των συστατικών (components) εφαρμόζοντας την αρχή της μονόδρομης ροής δεδομένων. Αυτό καθιστά τον κώδικα εύκολο στην ανάγνωση και την συντήρηση του, καθώς και διευκολύνει την παρακολούθηση των αλλαγών στην κατάσταση της εφαρμογής

- **Μεγάλη κοινότητα και ισχυρή υποστήριξη**

Διαθέτει μια μεγάλη κοινότητα προγραμματιστών και ενεργή υποστήριξη από το Facebook και άλλες εταιρείες. Αυτό σημαίνει ότι υπάρχει πάντα πρόσβαση σε ενημερώσεις, νέες τεχνολογίες και λύσεις σε προβλήματα

Σε συνδιασμό με τη χρήση άλλων τεχνολογιών, όπως το REDUX που θα χρησιμοποιήσουμε για την διαχείριση του state και του store της εφαρμογής, η React μπορεί να χρησιμοποιηθεί για την δημιουργία πολύπλοκων εφαρμογών.

Ένας από τους κύριους λόγους που κάποιος οδηγείται στο να χρησιμοποιήσει το REACT.JS έναντι άλλων βιβλιοθηκών/framework της Javascript (πχ. Angular) είναι η απλότητα του. Γνωρίζοντας Html, Css και Javascript, είναι πολύ εύκολο ένας προγραμματιστής να χρησιμοποιήσει την συγκεκριμένη βιβλιοθήκη και να έχει τα οφέλη από την χρήση της όπως αναφέραμε παραπάνω.

Ένα συστατικό (component) στο React είναι μία function (Function Component) η οποία επιστρέφει HTML . Το React χρησιμοποιεί το JSX (Javascript XML) το οποίο μας επιτρέπει να γράφουμε HTML μέσα σε ένα React component. Το React component κάνει render το JSX , και με τον τρόπο αυτό σχηματίζεται η διεπαφή χρήστη .

Επίσης μπορούμε να δώσουμε style στην διεπαφή χρήστη, χρησιμοποιώντας CSS στα React elements εντός του JSX.

Ένα απλό παράδειγμα ενός React Component στο οποίο μπορούμε να δούμε πως ένας προγραμματιστής γνωρίζοντας Html,Css και Javascript μπορεί εύκολα να κατανοήσει και να χρησιμοποιήσει το React στο Front End κομμάτι της εφαρμογής του, είναι το παρακάτω :

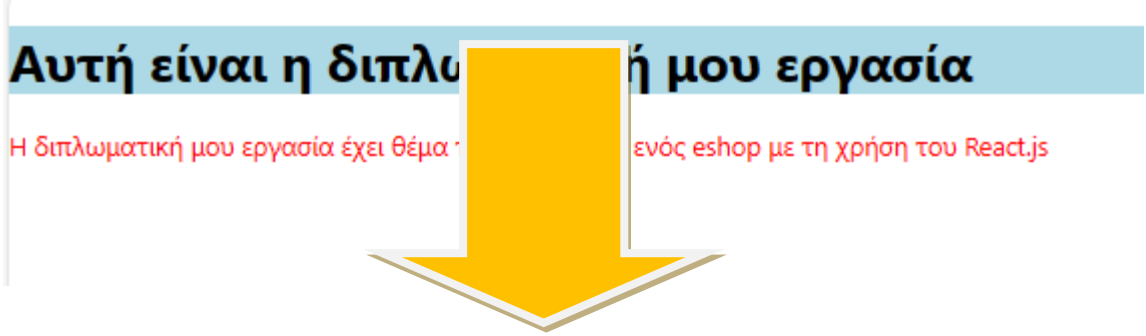
1.

```
ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById("root")
);

reportWebVitals();
```

ReactDOM.Render : Η συγκεκριμένη μέθοδος χρησιμοποιείται για να κάνει render ένα React element σε έναν συγκεκριμένο κόμβο του DOM . Στην συγκεκριμένη περίπτωση το <App/> είναι το κύριο component της εφαρμογής και επιστρέφεται στο στοιχείο με id = “root”

2.

```
import "./App.css";
import React from "react";

function App() {
  return (
    <div>
      <h1 style={{ backgroundColor: "lightblue" }}>
        Αυτή είναι η διπλωματική μου εργασία
      </h1>
      <p style={{ color: "red" }}>
        Η διπλωματική μου εργασία έχει θέμα την υλοποίηση ενός eshop με τη χρήση
        του React.js
      </p>
    </div>
  );
}

export default App;
```

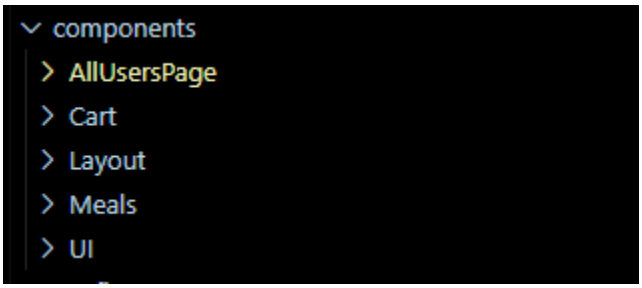


Στην παραπάνω φωτογραφία κώδικα βλέπουμε ότι το component <App/> , είναι μια Javascript function η οποία επιστρέφει HTML και CSS (μέσω του JSX).

Οι σελίδες που αναπτύξαμε στην εφαρμογή μας :

1. Αρχική σελίδα
2. Κατάλογος προϊόντων
3. Καλάθι αγορών
4. Σελίδα λογαριασμού χρήστη (LogIn / SignIn)
5. Σελίδα επικοινωνίας και πληροφοριών της επιχείρησης

Στην παρακάτω φωτογραφία βλέπουμε πως έχουμε χωρίσει σε φακέλους , τα συστατικά – components τα οποία καλύπτουν τις παραπάνω ανάγκες της εφαρμογής.



Back-end

Για την υλοποίηση του Back-end χρησιμοποιήσαμε το Node.js το οποίο είναι ένα περιβάλλον εκτέλεσης Javascript που βασίζεται στη μηχανή Javascript V8 της Google. Έχει δημιουργηθεί για να επιτρέπει την εκτέλεση της Javascript από την πλευρά του διακομιστή (server-side) αντί να περιορίζεται στην πλευρά του περιηγητή (browser-side) . Το Node.js είναι δημοφιλές για την δημιουργία διακομιστών, web εφαρμογών, API . Είναι επίσης ιδανικό για ανάπτυξη real-time εφαρμογών, όπως τα chat-applications.

Παρέχει την απαραίτητη υποδομή για να καλύψει τις λειτουργικές ανάγκες μιας εφαρμογής σε πολλούς τομείς.

Για παράδειγμα :

1. Χρήστες (authentication , authorization)
2. Προϊόντα (CRUD operations, pagination, filtering)
3. Κατηγορίες (CRUD operations, hierarchy)
4. Καλάθι αγορών (CRUD operations, order processing)
5. Πληρωμές (integration with payment gateway)
6. API : Restful API για προϊόντα, κατηγορίες, χρήστες, καλάθια αγορών , παραγγελίες και πληρωμές

Χρησιμοποιήθηκε επίσης το Express.Js το οποίο είναι το πιο γνωστό Node Framework που χρησιμοποιείται για την δημιουργία διακομιστών στην πλατφόρμα Node.Js. Είναι ένα ευέλικτο και εύκολο στην χρήση framework που επιτρέπει τη δημιουργία δυναμικών web εφαρμογών και API (Application programming Interfaces).

```
const express = require("express");

const app = express();
app.use(express.json());
app.use(cors()); // Enable CORS for all routes

const port = 3001;
const uri =
  "mongodb+srv://admin:admin@cluster0.duarj.mongodb.net/food-orders?retryWrites=true&w=majority";


app.get("/productslist", async (req, res) => {
  try {
    const products = await Product.find({});
    console.log("Products from the database: ", products);
    res.send(products);
  } catch (err) {
    console.error("Error retrieving products: ", err);
    res.status(500).json({ error: "Internal server error" });
  }
});
```

Βάση δεδομένων

Χρησιμοποιήσαμε την Mongo-Db , η οποία είναι ένα ανοικτού κώδικα σύστημα διαχείρισης βάσεων δεδομένων που χρησιμοποιεί μια μοντέρνα προσέγγιση για την αποθήκευση και ανάκτηση δεδομένων. Χρησιμοποιεί μια μορφή τύπου Json για την αποθήκευση των δεδομένων όπως θα δούμε στο παράδειγμα μας .

Η βάση χρησιμοποιήθηκε για την αποθήκευση/ανάκτηση :

1. Χρηστών
2. Προϊόντων

Για την αλληλεπίδραση της βάσης (MongoDb) με το περιβάλλον Node.js , χρησιμοποιήθηκε το Mongoose το οποίο είναι ένα framework της Javascript.

Χρησιμοποιήσαμε το mongoose για να ορίσουμε το schema (productSchema) και τους τύπους των δεδομένων για κάθε πεδίο του.

Ορίσαμε το μοντέλο (Product) στο οποίο αντιστοιχεί το συγκεκριμένο schema με την χρήση του “mongoose.model”.

Με την χρήση των σχημάτων (schemas) και των μοντέλων (models) , μπορούμε να καθορίσουμε το πώς θα εμφανίζονται, θα αποθηκεύονται και θα χρησιμοποιούνται τα δεδομένα της βάσης δεδομένων.

Παρακάτω θα δούμε ένα σχετικό παράδειγμα σχετικά με τον ορισμό του σχήματος και του μοντέλου που χρησιμοποιήθηκε.

Ανάκτηση Προϊόντων από τη Βάση Δεδομένων με React και Node.js

Ανάλυση Λειτουργίας

Ας δούμε το πώς η εφαρμογή μας ανακτά τα προϊόντα από την βάση δεδομένων , χρησιμοποιώντας την React στο front-end και την Node.js στο back-end με την χρήση της βάσης δεδομένων MongoDB .

1. Front-End (React)

```
export const fetchProducts = () => {
  return (dispatch) => {
    axios
      .get("http://localhost:3001/productslist")
      .then((response) => {
        const products = response.data;
        dispatch(fetchProductsRequest(products));
      })
      .catch((err) => {
        const errorMsg = err.message;
      });
  };
```

Σε αυτό το τμήμα του κώδικα πραγμοτοποιείται ένα αίτημα (GET request) στον server στην διεύθυνση locallhost:3001/productlist χρησιμοποιώντας το Axios . (Το Axios είναι ένα πακέτο το οποίο το χρησιμοποιούμε για τη δημιουργία HTTP αιτημάτων όπως στο παραπάνω παράδειγμα). Όταν λαμβάνουμε την απάντηση από τον server , αποθηκεύουμε τα προϊόντα στην μεταβλητή

products , και στην συνέχεια καλούμε το action “fetchProductsRequest” το οποίο ενημερώνει το redux store της εφαρμογής μας.

Η αρχική κατάσταση του store της εφαρμογής μας είναι :

```
const INITIAL_STATE = {
  products: [], // {id,title,descr,price,img,qty}
  cart: [], // {id,title,descr,price,img,qty}
  currentItem: null,
  page: PAGES.HomePage,
};
```

Έπειτα από την κλήση για την λήψη των διαθέσιμων προϊόντων , ενημερώνουμε κατάληλα το store.

```
const shopReducer = (state = INITIAL_STATE, action) => {
  switch (action.type) {
    case actionTypes.FETCH_PRODUCTS_REQUEST:
      return {
        ...state,
        products: action.payload,
      };
```

Για την διαχείρηση του store της εφαρμογής χρησιμοποιούμε το REDUX.

State Management (Redux)

Το Redux είναι μια open-source βιβλιοθήκη της Javascript και χρησιμοποιείται για την διαχείριση της κατάστασης της εφαρμογής. Το χρησιμοποιούμε ώστε η κατάσταση της εφαρμογής να είναι προβλέψιμη και ευκολότερη στην ανάπτυξη και την συντήρηση.

ΒΑΣΙΚΕΣ ΑΡΧΕΣ

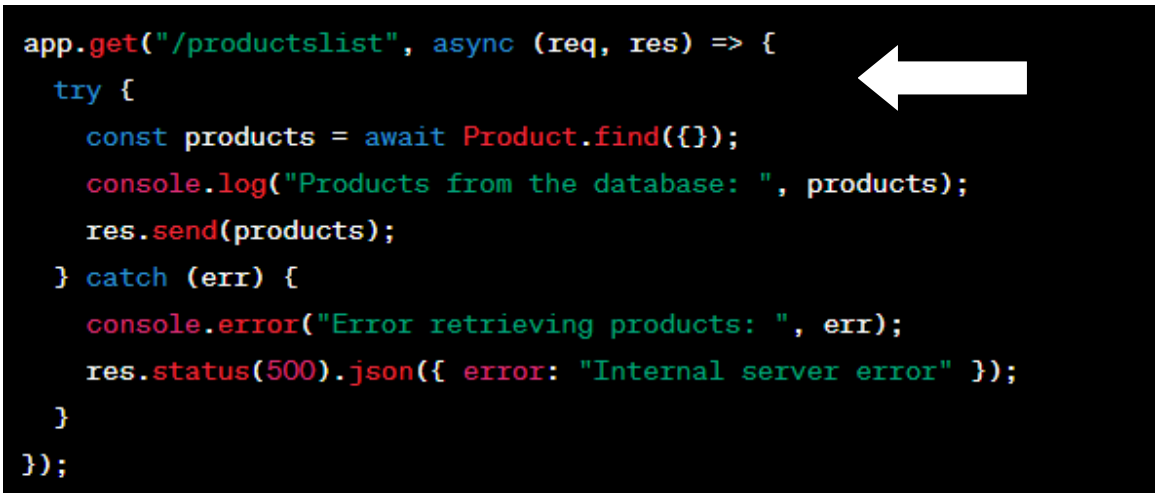
1. Το state ολόκληρης της εφαρμογής αποθηκεύεται σε ένα object το οποίο είναι το store της εφαρμογής που αναπτύσσουμε
2. Το state της εφαρμογής αλλάζει μόνο μέσω των actions, αντικείμενα τα οποία περιέχουν ένα type που περιγράφει το είδος της ενέργειας. Αυτό σημαίνει ότι το state της εφαρμογής δεν μπορεί να αλλάξει απευθείας από τα components.
3. Οι αλλαγές στο state της εφαρμογής πραγματοποιείται μέσω των reducers. Οι reducers λαμβάνουν ως όρισμα το προηγούμενο state της εφαρμογής και το είδος της ενέργειας (action type), και επιστρέφουν το ανανεωμένο state.

Στο παράδειγμα της παραπάνω φωτογραφίας :

1. Store της εφαρμογής -> INITIAL_STATE
2. Ένα από τα action types είναι το FETCH_PRODUCTS_REQUEST όπου έγινε dispatch από το component για να ενημερώσει κατάλληλα το state
3. Ο reducer που ενημερώνει το state της εφαρμογής ανάλογα με το type των actions που λαμβάνει -> shopReducer

Με αυτόν τον τρόπο ενημερώθηκε το state της εφαρμογής, μετά την επιτυχή λήψη των διαθέσιμων προϊόντων μέσω του HTTP request.

2. Back-End (Node.js)



```
app.get("/productslist", async (req, res) => {
  try {
    const products = await Product.find({});
    console.log("Products from the database: ", products);
    res.send(products);
  } catch (err) {
    console.error("Error retrieving products: ", err);
    res.status(500).json({ error: "Internal server error" });
  }
});
```

Στο back-end μέρος της εφαρμογής χρησιμοποιούμε το Express και το Mongoose για την διαχείρηση των προϊόντων.

Έχει δημιουργηθεί το route “/productlist” το οποίο χρησιμοποιούμε για να ανακτήσουμε όλα τα προϊόντα από την βάση δεδομένων.

Στην παρακάτω φωτογραφία βλέπουμε την δημιουργία του μοντέλου Product.

Το Product δημιουργείται με βάση το “productSchema” χρησιμοποιώντας τη μέθοδο “mongoose.model()” .

Στο productSchema καθορίσαμε τη δομή των δεδομένων που αποθηκεύονται στην βάση δεδομένων MongoDB.

```
const mongoose = require("mongoose");
const Schema = mongoose.Schema;

const productSchema = new Schema({
  name: {
    type: String,
    required: true,
    unique: true,
  },
  description: {
    type: String,
  },
  price: {
    type: Number,
    required: true,
    validate(value) {
      if (value < 0) {
        throw new Error("Price must be positive number");
      }
    },
  },
  qty: {
    type: Number,
    default: 0,
    validate(value) {
      if (value < 0) {
        throw new Error("Quantity must be positive number");
      }
    },
  },
  image: {
    type: String,
    required: true,
  },
});

const Product = mongoose.model("product", productSchema);

module.exports = Product;
```

Μέσω του Get request (στο route “/productlist”) τα διαθέσιμα προϊόντα επιστρέφονται ως απάντηση στον client, και με την σειρά του το front-end είναι υπεύθυνο για την παρουσίαση τους.

```
function Meals() {
  return (
    <>
      <MealsSummary />
      <AvailableMeals />
    </>
  );
}
```

Μέσω της παραπάνω συνάρτησης-component Meals() , επιστρέφονται 2 components , το MealsSummary() και το AvailableMeals() .

Meals Summary

Αποτελεί μια παρουσίαση της εταιρίας και των διαθέσιμων προϊόντων της

```
function MealsSummary() {
  return (
    <section className={classes.summary}>
      <h2>Delicious Food , Delivered To You</h2>
      <p>
        Choose your favorite meal from our broad selection of available meals
        and enjoy a delicious lunch or dinner at home.
      </p>
      <p>
        {" "}
        All our meals are cooked with high-quality ingredients, just-in-time and
        of course by experienced chefs!
      </p>
    </section>
  );
}
```

Available Meals

Όπως είδαμε παραπάνω , κατόπιν λήψης των διαθέσιμων προίοντων απο την βάση Δεδομένων , το front-end είναι υπευθυνο για την παρουσίαση τους στον τελικό χρήστη.

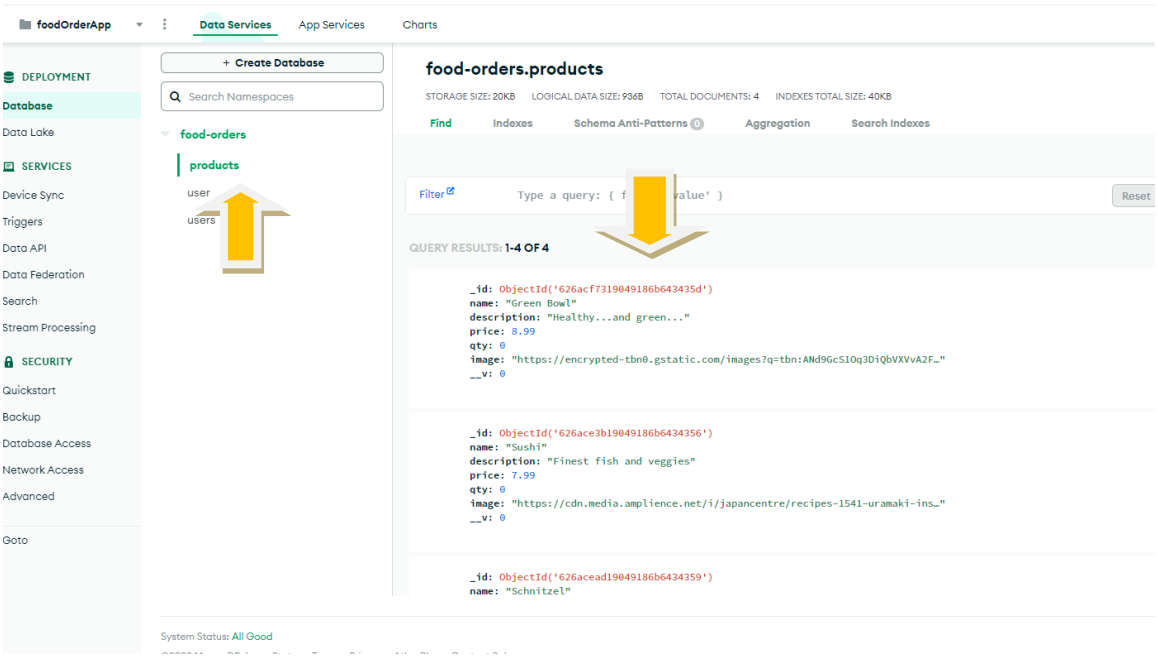
Χρησιμοποιώντας το REDUX, έχουμε ενημερωμένο στο store της εφαρμογής μας και μπορούμε να έχουμε απευθείας πρόσβαση σε αυτό, είτε λαμβάνοντας δεδομένα είτε αλλάζοντας τα μέσω των actions.

Η συγκεκριμένη συνάρτηση λαμβάνει τα διαθέσιμα προϊόντα από το redux store (μέσω του mapStateToProps) , τα οποία λαμβάνουμε από το GET request (fetchProducts) όπως είδαμε παραπάνω , και τα προβάλλει με την μορφή λίστας στον χρήστη.

```
function AvailableMeals({ productsData, fetchProducts }) {
  useEffect(() => {
    fetchProducts();
  }, []);

  const mealsList = productsData.products.map((meal) => (
    <MealItem
      key={meal._id}
      id={meal._id}
      name={meal.name}
      description={meal.description}
      price={meal.price}
      image={meal.image}
    />
  ));

  return (
    <section className={classes.meals}>
      <ul>{mealsList}</ul>
    </section>
  );
}

const mapStateToProps = (state) => {
  return {
    productsData: state.shop,
  };
};

const mapDispatchToProps = (dispatch) => {
  return {
    fetchProducts: () => dispatch(fetchProducts()),
  };
};
```

ΔΙΑΧΕΙΡΗΣΗ ΧΡΗΣΤΩΝ & ΠΡΟΙΟΝΤΩΝ

Βάση Δεδομένων MongoDb



Στην βάση δεδομένων έχουμε δημιουργήσει ένα μοντέλο για τα διαθέσιμα προϊόντα.

{

"**_id**":{"$oid":"626acf7319049186b643435d"},

"**name**":"Green Bowl",

"**description**":"Healthy...and green...",

"**price**":{"$numberDouble":"8.99"},

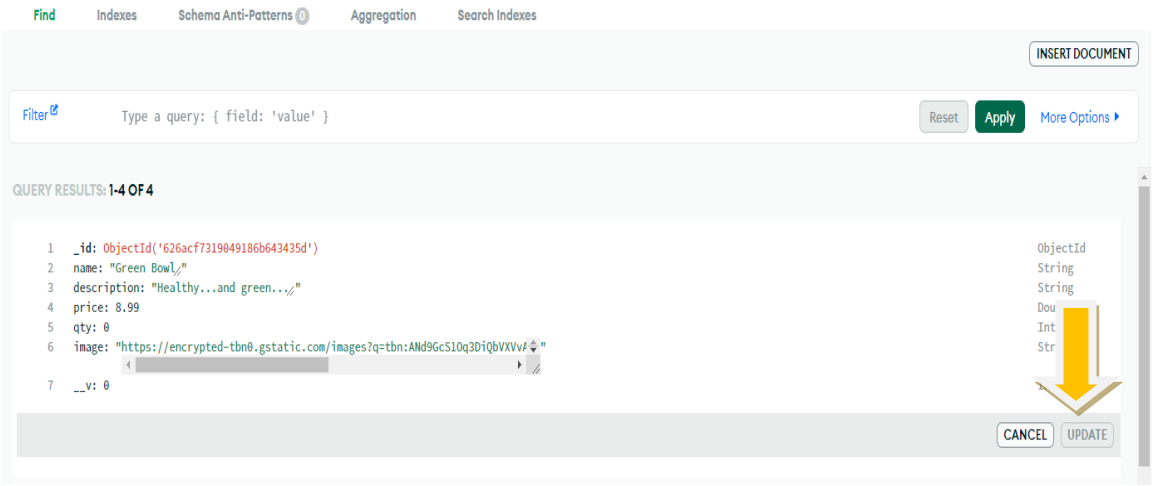
"**qty**":{"$numberInt":"0"},

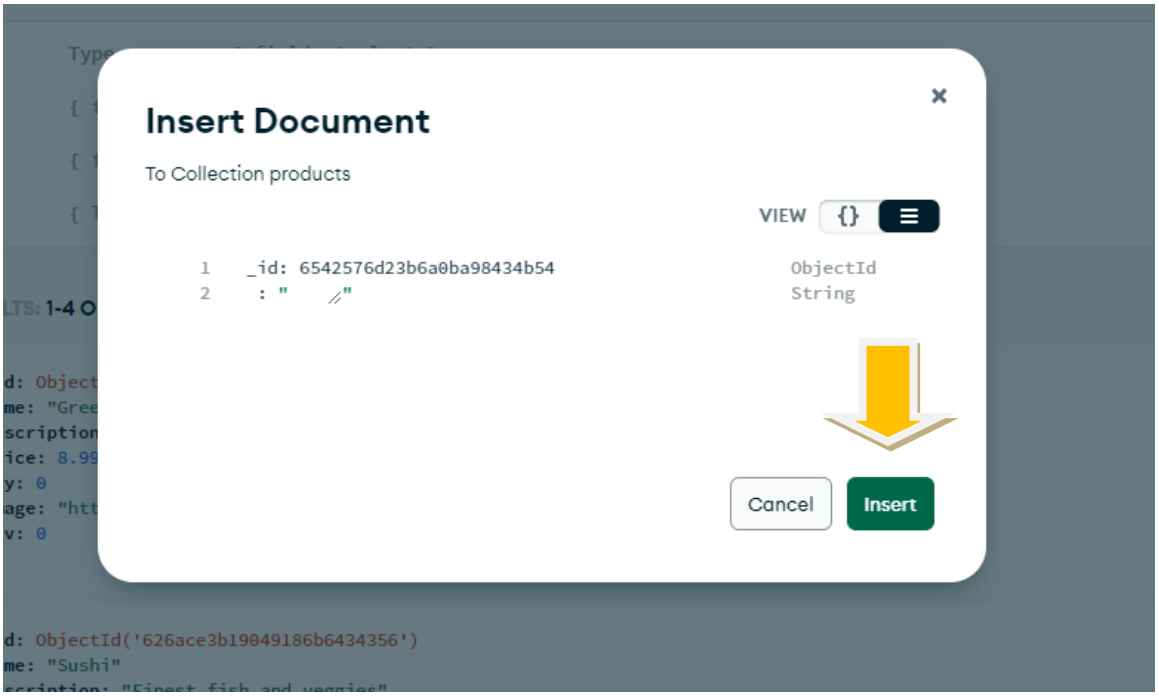
"**image**":"https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcS1Oq3DiQbVXVvA2FFkOmJORy-i7z8E2lR9DW6QC8LXcdXu714a_B2u8H2I9B4iVqK2TxI&usqp=CAU",

"__v":{"$numberInt":"0"}

}

Έχουμε την δυνατότητα εφόσον συνδεθούμε στην βάση, να δούμε τα διαθέσιμα προϊόντα , να τα επεξεργαστούμε καθώς και να προσθέσουμε νέο προιον .





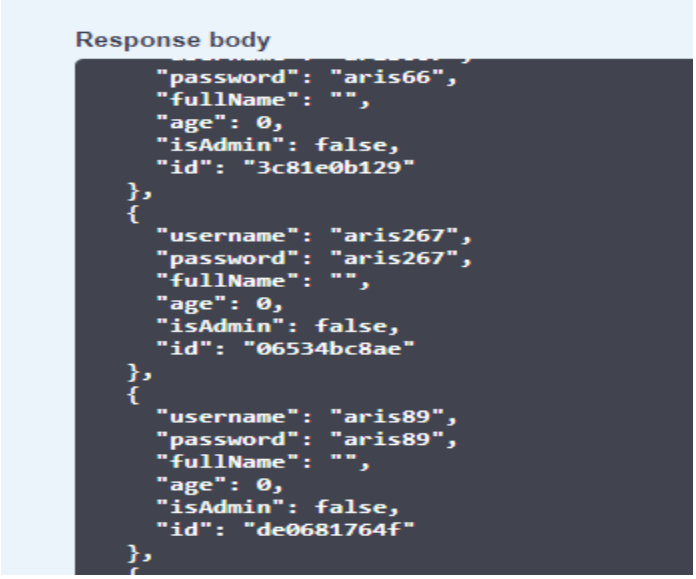
Δημιουργία και Χρήση Εξωτερικού API

Για την διαχείριση των χρηστών της εφαρμογής δημιουργήσαμε ένα API (Application Programming Interface) μέσω της εφαρμογής stackprint.io που μας παρέχει όλες τις απαραίτητες λειτουργίες για τη διαχείριση των χρηστών.

Αυτό το API διαθέτει τις παρακάτω λειτουργίες :

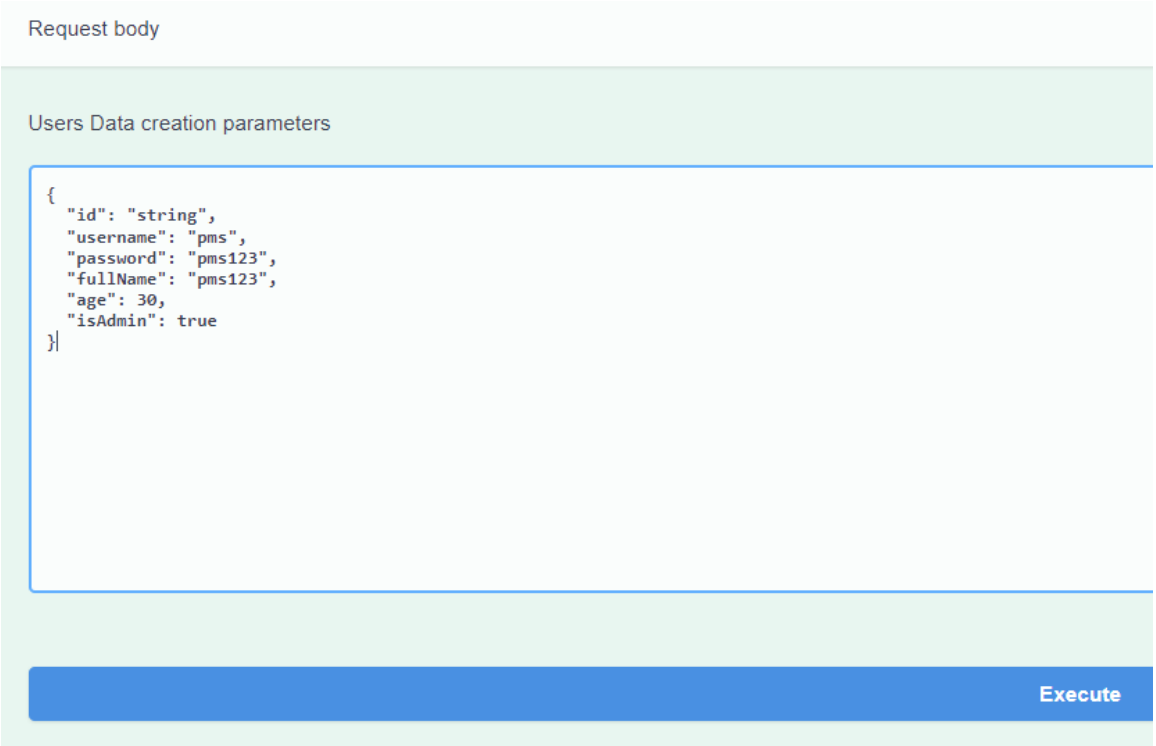
1. **GET /users-data:**

   Αυτή η λειτουργία επιτρέπει την ανάκτηση μιας λίστας με τα δεδομένα όλων των χρηστών.



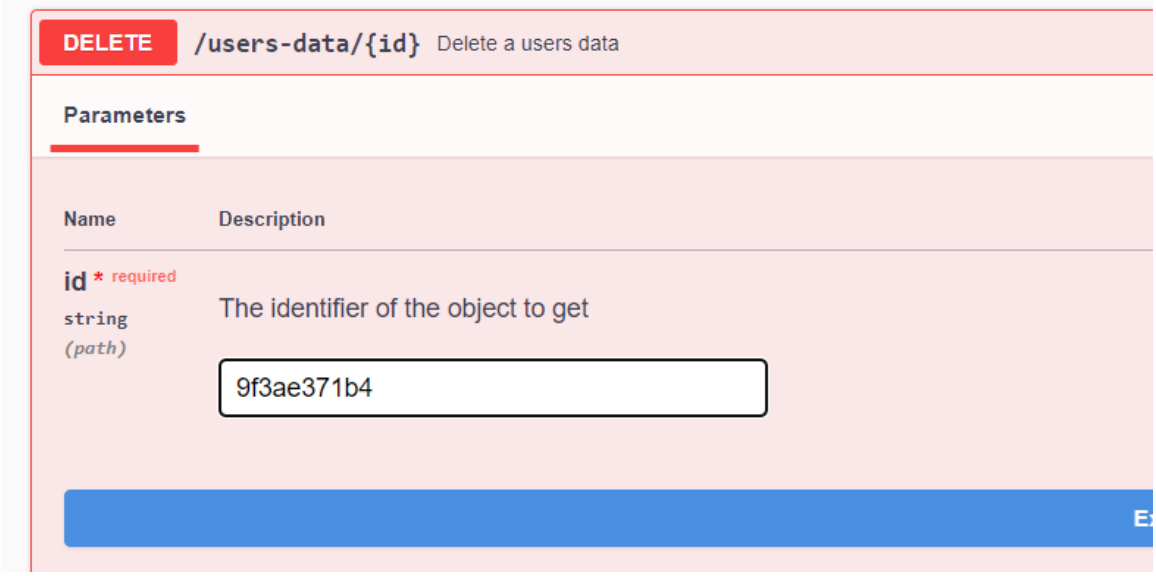
2. POST /users-data:

Με αυτήν την λειτουργία δημιουργούμε νέα δεδομένα χρήστη. Αποστέλλονται τα δεδομένα του νέου χρήστη στον διακομιστή μέσω μιας αίτησης POST.



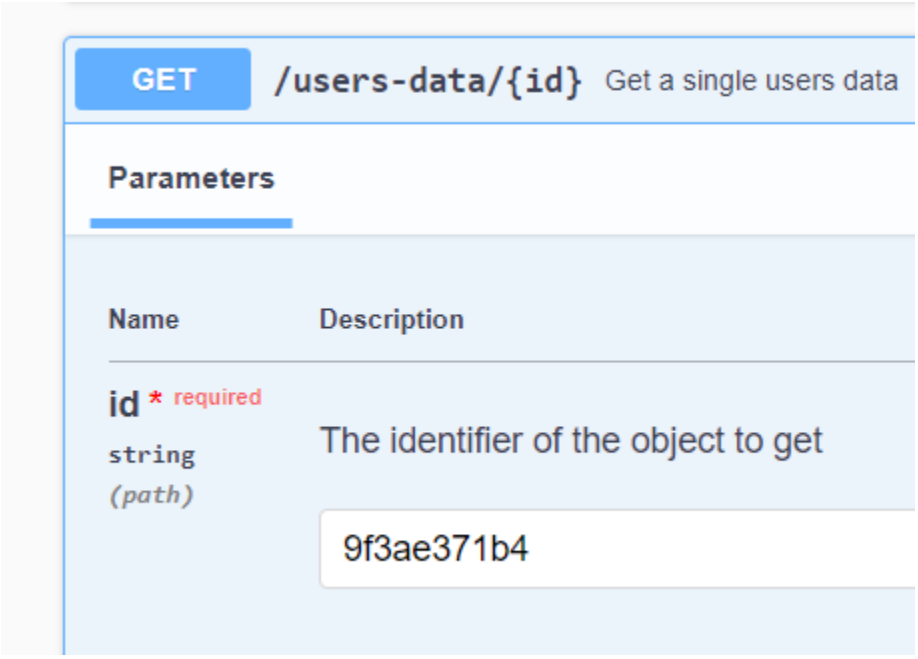
3. DELETE /users-data/{id}:

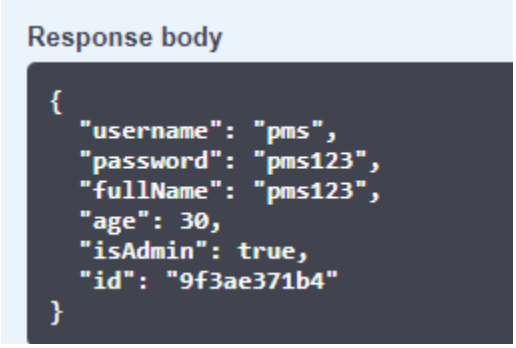
Αυτή η λειτουργία επιτρέπει την διαγραφή των δεδομένων ενός συγκεκριμένου χρήστη βάσει του **id** του χρήστη. Αποστέλλεται μια αίτηση DELETE με το σχετικό **id** στο URL.



4. GET /users-data/{id}:

Με αυτήν την λειτουργία γίνεται ανάκτηση των δεδομένων ενός συγκεκριμένου χρήστη βάσει του **id**. Αποστέλλεται μια αίτηση GET με το σχετικό **id** στο URL.



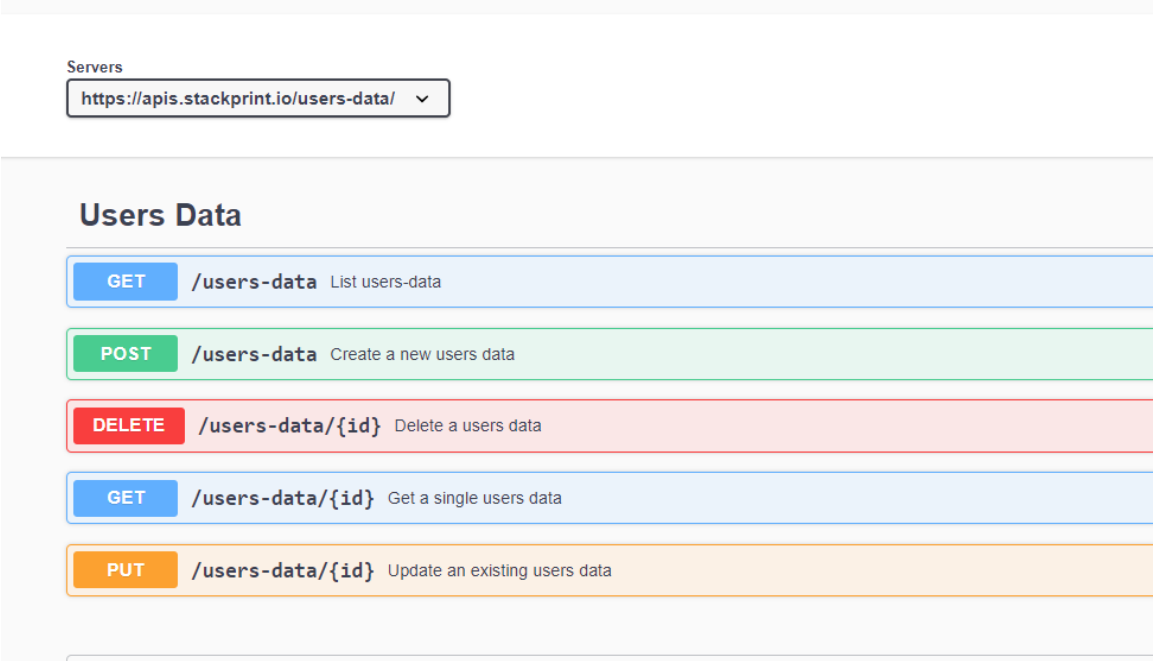


5. PUT /users-data/{id}:

Αυτή η λειτουργία επιτρέπει την ενημέρωση των δεδομένων ενός συγκεκριμένου χρήστη βάσει του **id** του χρήστη. Αποστέλλεται μια αίτηση PUT με το σχετικό **id** στο URL και τα δεδομένα χρήστη ενημερώνονται.

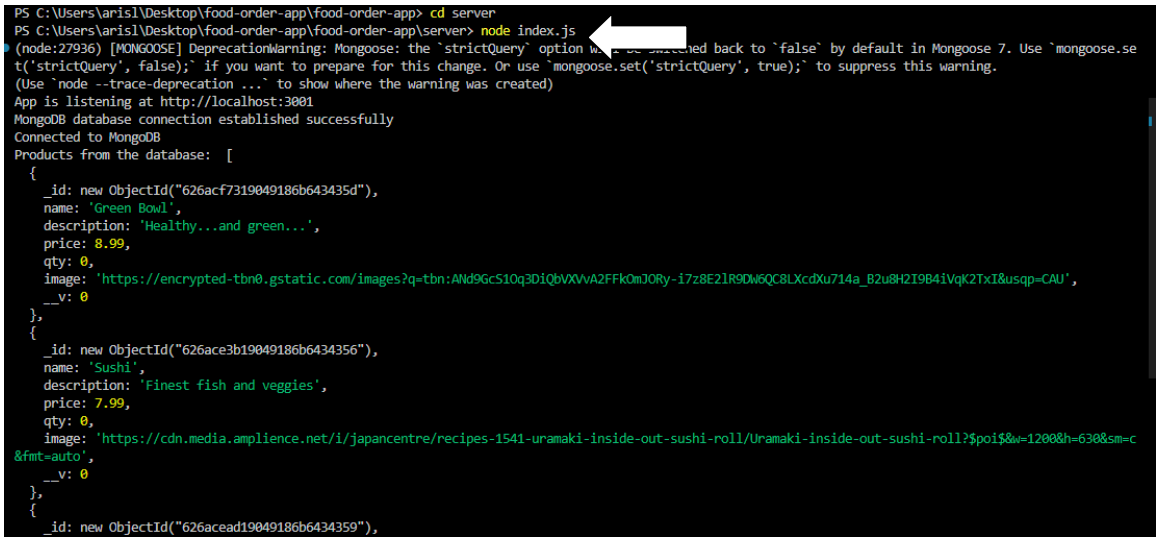
Μέσω αυτού του API , υλοποιήσαμε την φόρμα εγγραφής χρηστών (**POST /users-data)** , και την δυνατότητα σύνδεσης εγγεγραμένου χρήστη , όπου ελέγχουμε μέσω του API αν ο χρήστης υπάρχει . (**GET /users-data/{id}) .**

Η χρήση ενός τέτοιου API αποτελεί μια εναλλακτική λύση, στο να δημιουργήσουμε τη δική μας βάση δεδομένων, καθώς εξοικονομεί χρόνο και πόρους.



ΕΚΚΙΝΗΣΗ ΕΦΑΡΜΟΓΗΣ LOCALLY

Εκκίνηση του server



Τρέχουμε στο τερματικό της εφαρμογής μας (terminal) την εντολή -> node index.js

Η εντολή αυτή τρέχει τον κώδικα που περιέχει το αρχείο index.js (Javascript αρχείο) χρησιμοποιώντας το Node.js runtime environment.

1. **Εκκίνηση του server**
   Ο κώδικας που περιέχει το index.js αρχείο δημιουργεί έναν web server χρησιμοποιώντας το framework Express.js

2. **Σύνδεση με βάση δεδομένων**
   Ο κώδικας χρησιμοποιεί το Mongoose για να συνδεθεί με την MongoDB βάση δεδομένων που δημιουργήσαμε που βρίσκεται στη διεύθυνση που ορίζεται στη μεταβλητή “uri” (Uniform Resource Identifier).

```
const port = 3001;
const uri =
  "mongodb+srv://admin:admin@cluster0.duarj.mongodb.net/food-orders?retryWrites=true&w=majority";
```

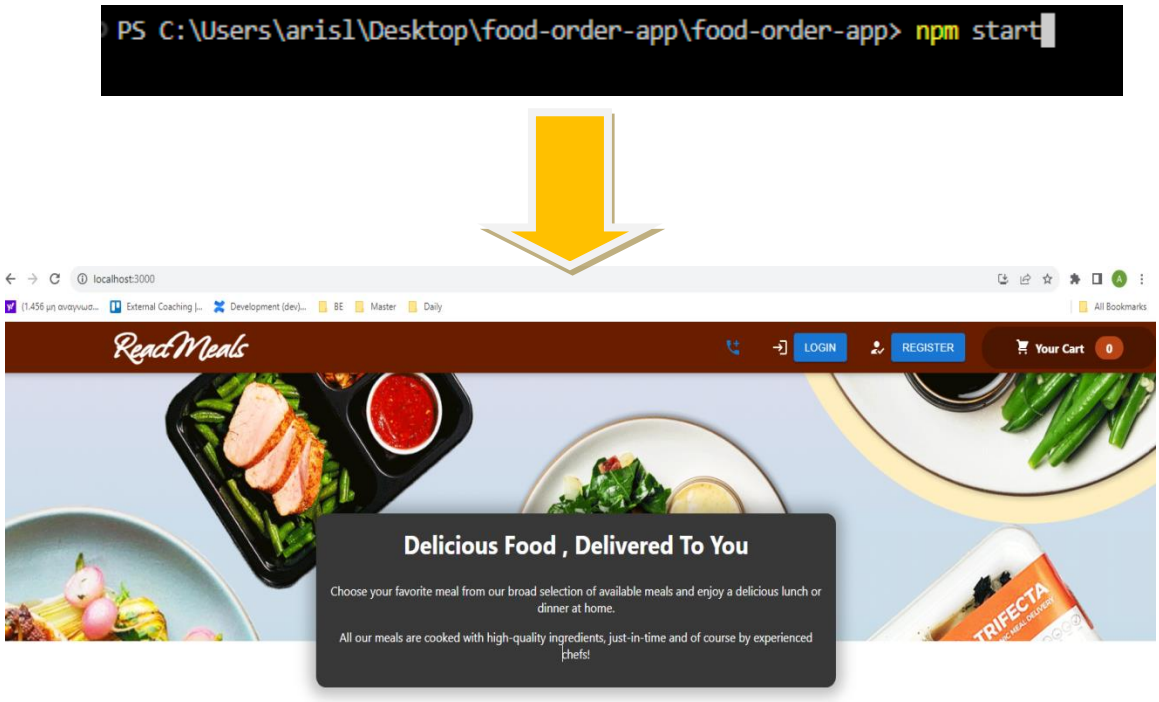
3. **Ρύθμιση Διαδρομών (Routes)**
   Δημιουργούνται διαδρομές όπως το /productslist , /products , /userlist , /user χρησιμοποιώντας τις μεθόδους HTTP GET και POST. Κάθε διαδρομή ανταποκρίνεται στα αιτήματα του client.

4. **Αλληλεπίδραση με τη Βάση Δεδομένων**

Κάθε route αλληλεπιδρά με την βάση δεδομένων και επιστρέφει τα προϊόντα ή καταχωρεί νέα ανάλογα το αίτημα του client (POST / GET)

Όταν εκτελούμε αυτή την εντολή στο τερματικό, ο διακομιστής θα ξεκινήσει να ακούει στην πόρτα 3001 του localhost όπως βλέπουμε στην παραπάνω φωτογραφίας εκτέλεσης της εντολής.

Εκκίνηση του React App μας στο localhost:3000

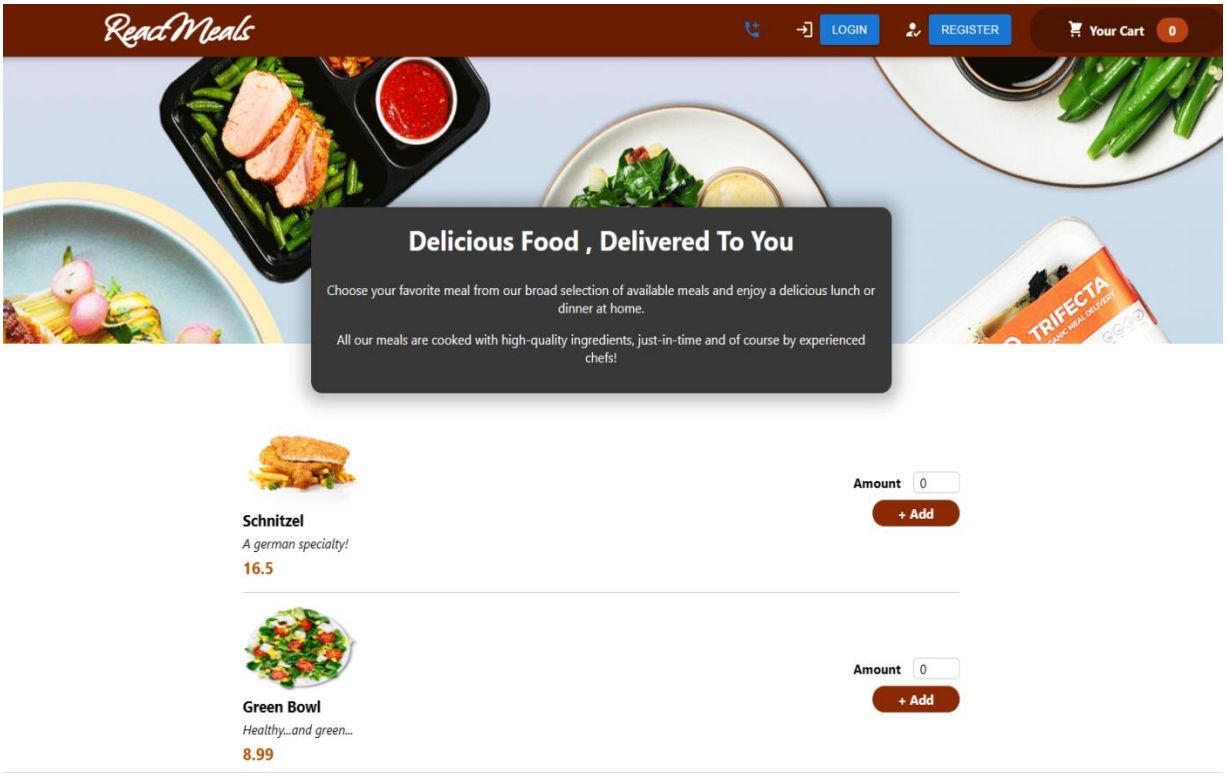


Παρουσίαση του Ηλεκτρονικού Καταστήματος [food-order-app]

[React App (arislazaridis.netlify.app)]

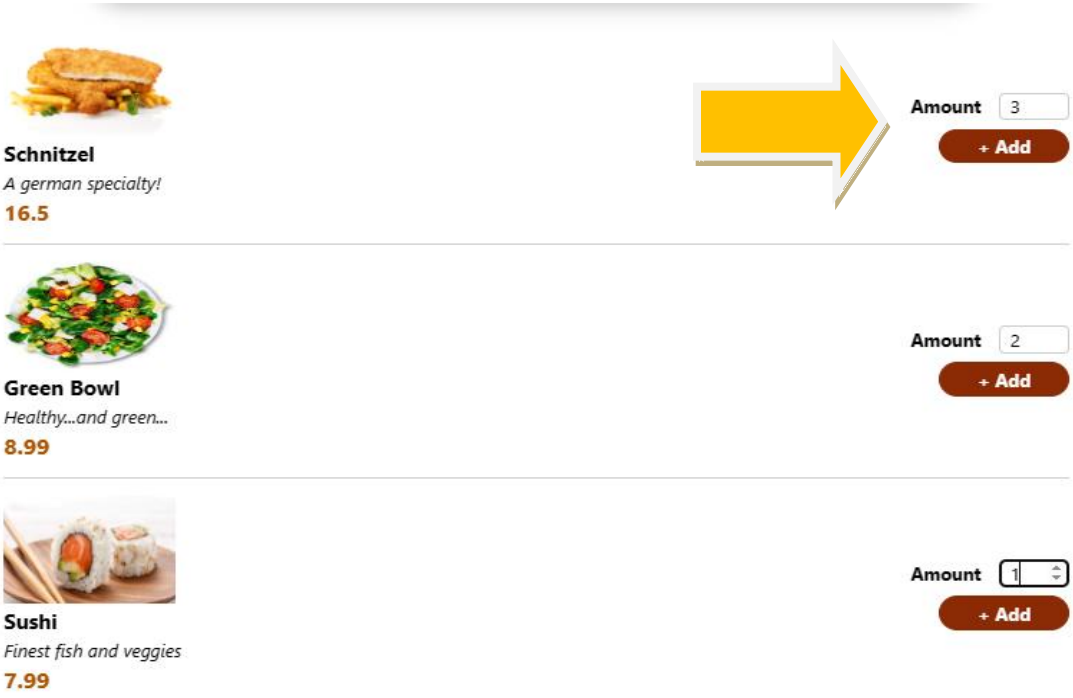
Αρχική σελίδα με τα διαθέσιμα προϊόντα

Παρουσιάζονται τα διαθέσιμα καταχωρημένα προϊόντα , καθώς και μια σύντομη περιγραφή της επιχείρησης



Προσθήκη προϊόντος στο καλάθι αγορών

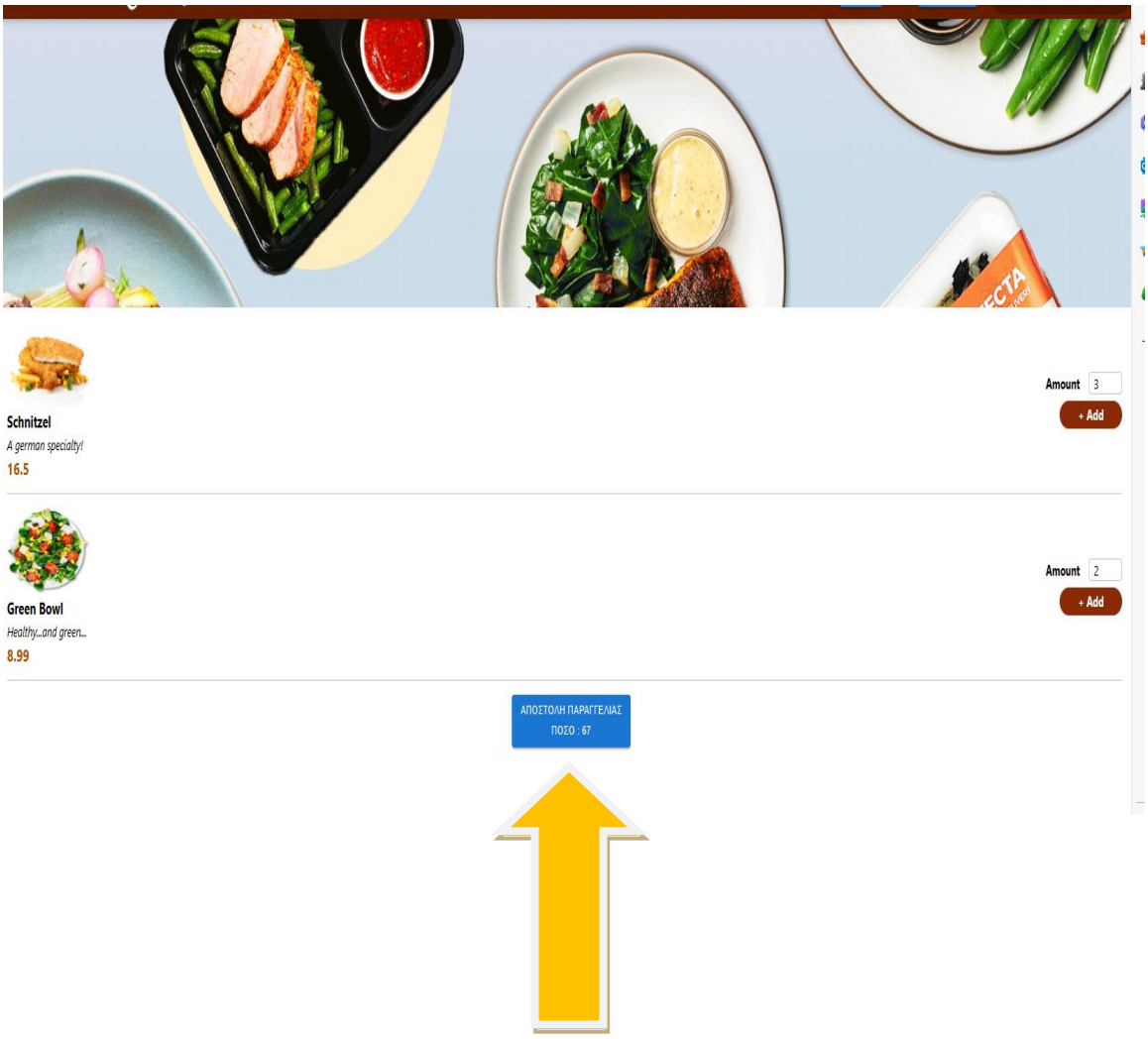
Υπάρχει η δυνατότητα επιλογής της ποσότητας των διαθέσιμων προϊόντων όπου ο χρήστης θέλει να προσθέσει στο καλάθι αγορών του



Η συνολική ποσότητα των προϊόντων που έχει επιλεγεί εμφανίζεται στο παρακάτω εικονίδιο που είναι το καλάθι αγορών :

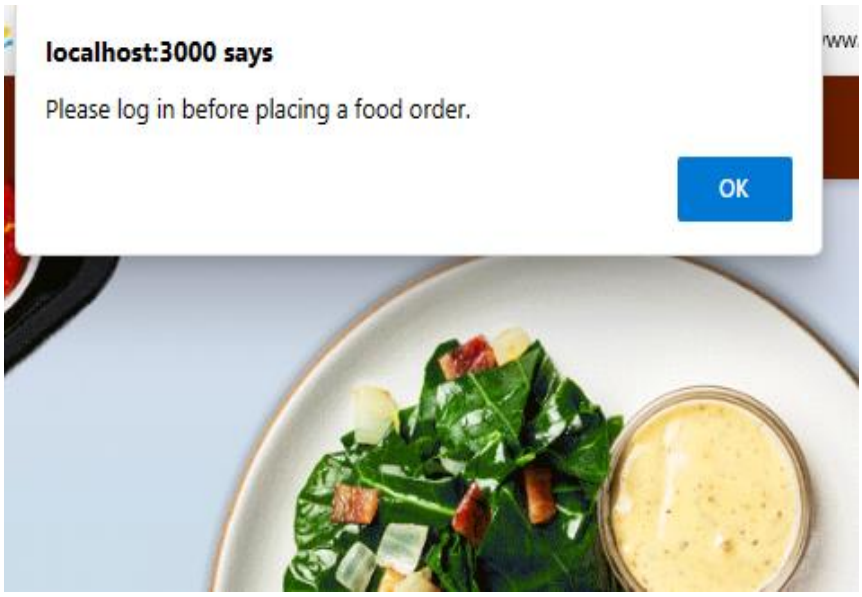
Σελίδα καλαθιού και παραγγελίας

Εδώ ο καταναλωτής βλέπει τα προϊόντα που έχει προσθέσει και μπορεί να προχωρήσει με την αποστολή της παραγγελίας. Αναγράφεται στο κουμπί αποστολής το συνολικό ποσό πληρωμής

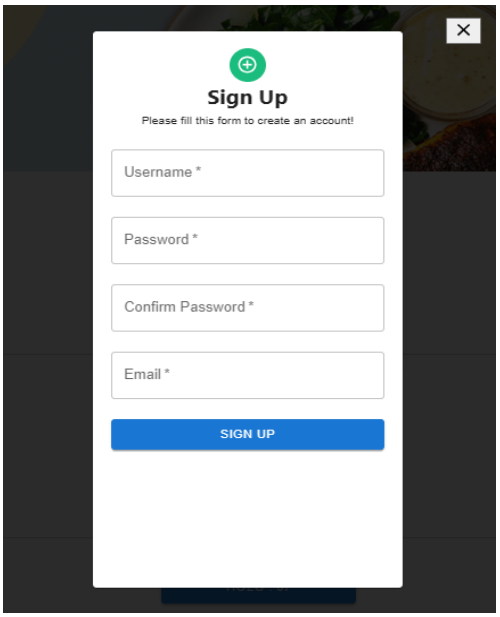


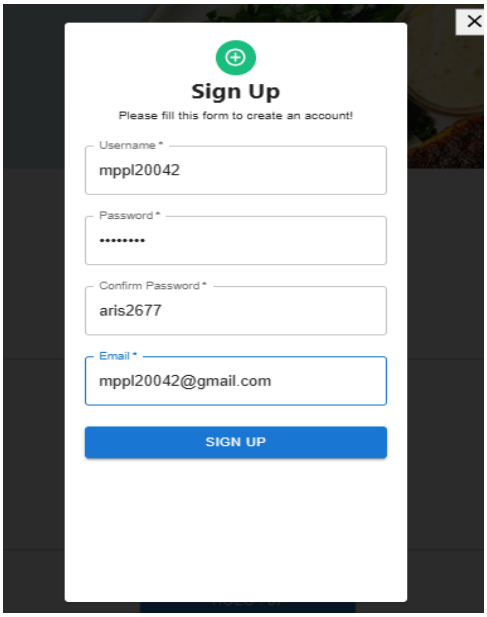
Αποστολή Παραγγελίας

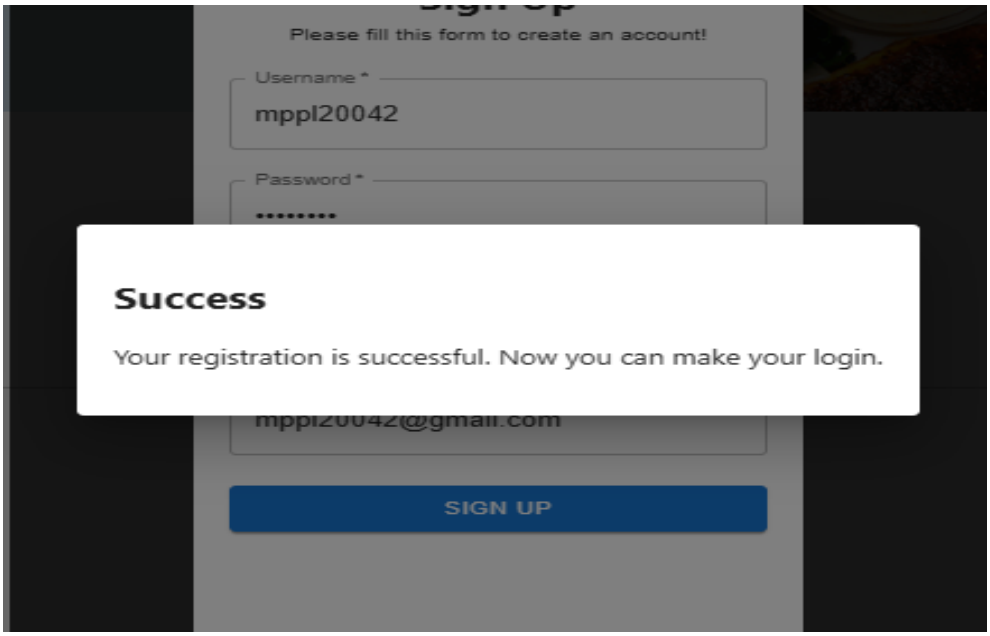
Για να προχωρήσει ο καταναλωτής στην αγορά/αποστολή παραγγελίας πρέπει να είναι εγγεγραμμένος χρήστης στην πλατφόρμα μας και να έχει κάνει login



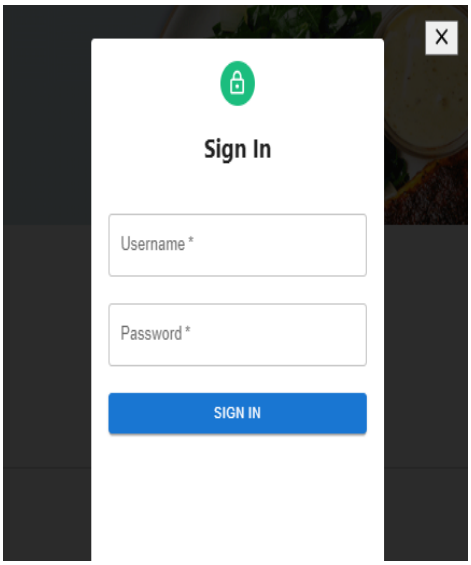
Φόρμα Εγγραφής

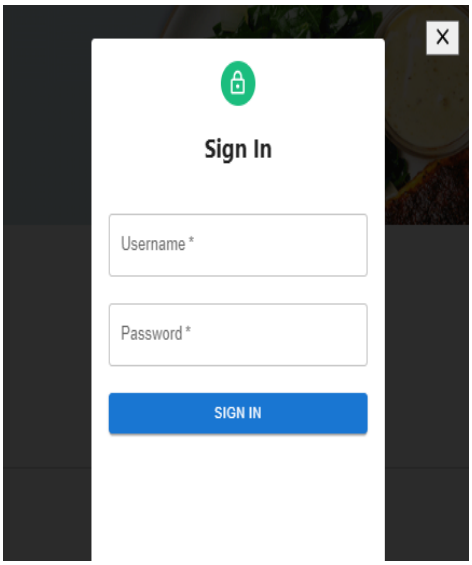


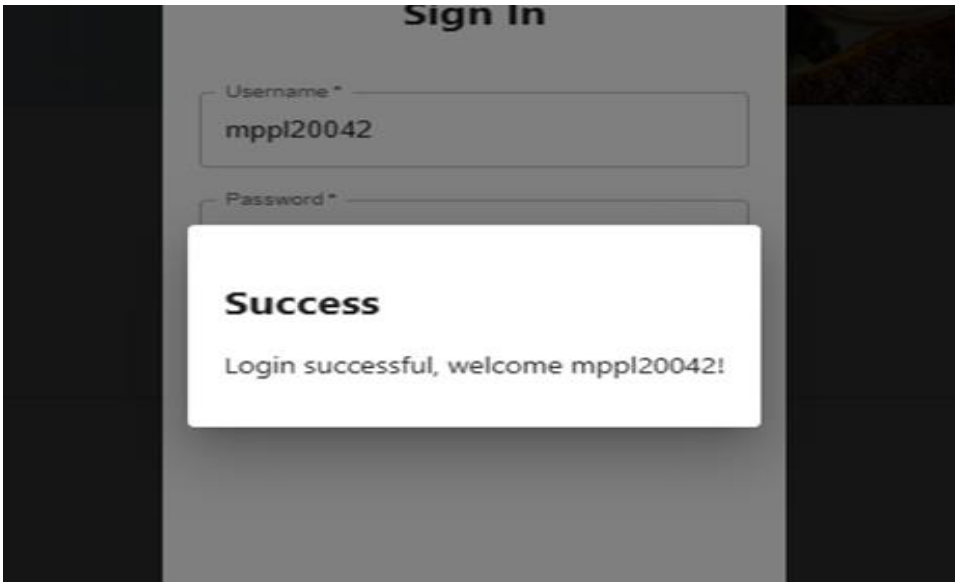




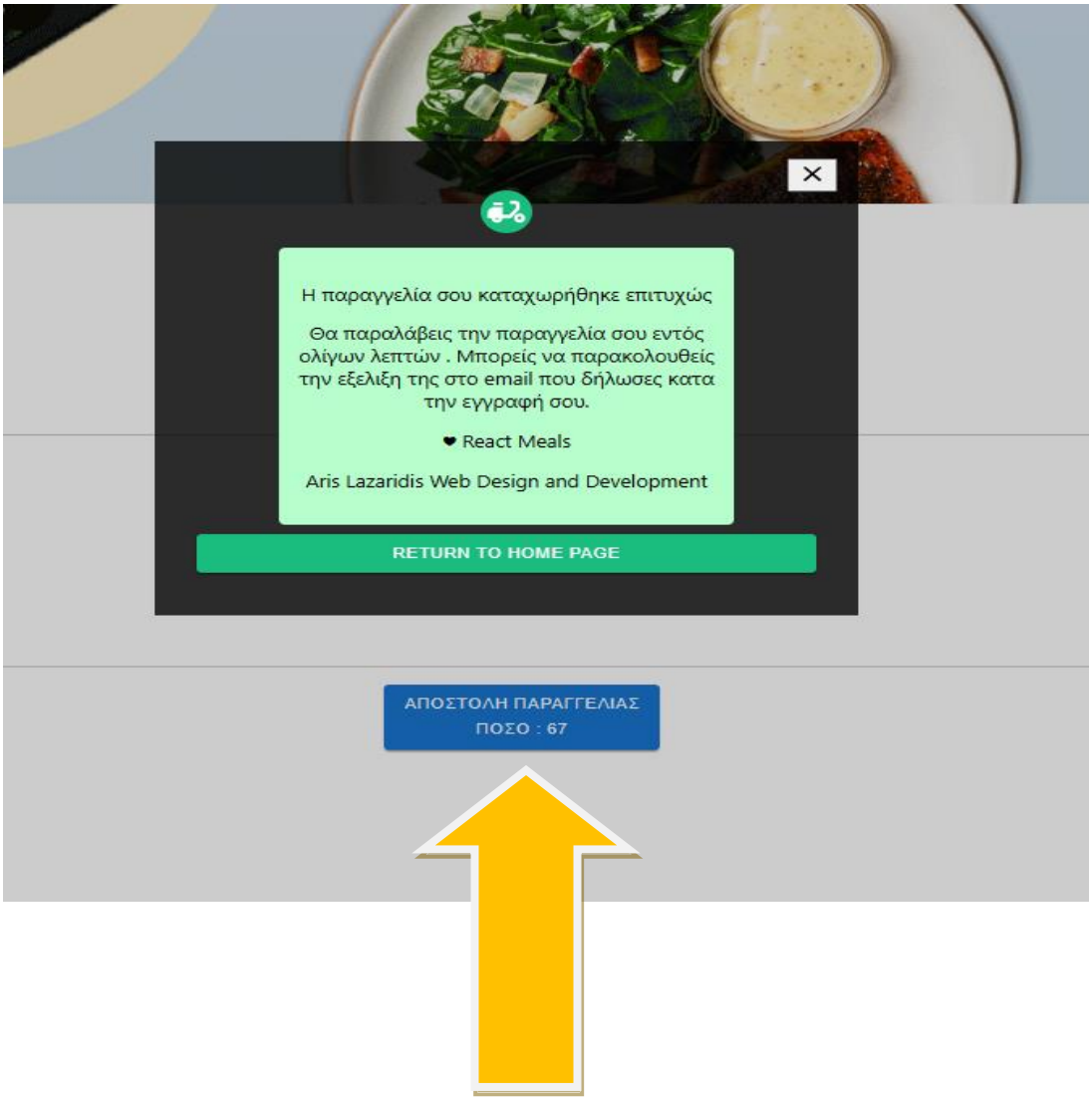
Φόρμα Σύνδεσης στην εφαρμογής







Ολοκλήρωση Παραγγελίας



Συμπεράσματα και μελλοντικές επεκτάσεις

Η παρούσα διπλωματική εργασία επικεντρώθηκε στην ανάπτυξη ενός food-order-app χρησιμοποιώντας προηγμένες τεχνολογίες όπως η React, Node.js ,Mongo.db και Redux.

Σκοπός αυτής της εργασίας ήταν η παρουσίαση ενός παραδείγματος υλοποίησης ενός eshop με έμφαση στην καινοτομία, την αποτελεσματικότητα και την δυνητική του εφαρμογή σε επιχειρησιακό περιβάλλον.

Αυτή η εφαρμογή μπορεί να παραμετροποιηθεί και να αναταποκριθεί στις ανάγκες κάθε επιχείρησης που επιθυμεί να χρησιμοποιήσει μια παρόμοια εφαρμογή για την προώθηση των προϊόντων της.

Ωστόσο υπάρχουν πολλές προοπτικές για μελλοντικές επεκτάσεις και βελτιώσεις στην εφαρμογή. Μερικές από αυτές περιλαμβάνουν :

- Καταχώρηση Στοιχείων Παραγγελίας Πελάτη:

  Προσθήκη της δυνατότητας για τους πελάτες να καταχωρούν τα στοιχεία παραγγελίας, όπως πιστωτική κάρτα και διεύθυνση παράδοσης απευθείας από την εφαρμογή.

- Αναζήτηση Προϊόντων :

  Για την κάλυψη μεγαλύτερης γκάμας προϊόντων, πρέπει να αναπτυχθεί η λειτουργία αναζήτησης, επιτρέποντας στους χρήστες να βρίσκουν το προϊόν που επιθυμούν να αγοράσουν

Συνοψίζοντας, αυτή η εργασία υλοποιήθηκε ως παράδειγμα εφαρμογής που αξιοποιεί τις νεότερες και καινοτόμες τεχνολογίες. Μελλοντικές επεκτάσεις και βελτιώσεις μπορούν να καταστήσουν την εφαρμογή αυτή ακόμη πιο ευέλικτη και αποδοτική ώστε να ανταποκρίνεται στις μεταβαλλόμενες ανάγκες των επιχειρήσεων και των χρηστών.

Βιβλιογραφία

REACT documentation, αναρτήθηκε 5 Δεκεμβριου 2020 απο

https://reactjs.org/docs.

INNOVIEW (2018). 2018 τι είναι το node JS, αναρτήθηκε 27 Απριλίου 2018

https://www.innoview.gr/el/blog/general/545-ti-einai-to-node-js

NPM (2020), The official MongoDB driver for Node.js αναρτήθηκε 11 Νοεμβρίου 2020 από

https://www.npmjs.com/package/mongodb

Why ReactJs is the most favoured Front-end Technology for start-ups αναρτήθηκε 24 Ιουνίου 2022 από

https://www.ranktracker.com/el/blog/why-react-js-is-the-most-favored-front-end-technology-for-startups/

REDUX documentation

https://redux.js.org/

MongoDB documentation

https://www.mongodb.com/what-is-mongodb

Alonistioti, Nancy, Evangelia Aikaterini Tsichrintzi, Konstantina Chrysafiadi, and Efthimios Alepis. 2023. “Requirements for Fuzzy Logic in Personalisation of Fire Emergency Alerts.” In *2023 14th International Conference on Information, Intelligence, Systems & Applications (IISA)*, 1–8. IEEE.

Argyropoulos, Vasileios, Efthimios Alepis, and Constantinos Patsakis. 2022. “Semi-Decentralized File Sharing as a Service.” In *2022 13th International Conference on Information, Intelligence, Systems & Applications (IISA)*, 1–8. IEEE.

Bilika, Domna, Nikoletta Michopoulou, Efthimios Alepis, and Constantinos Patsakis. “Hello Me, Meet the Real Me: Voice Synthesis Attacks on Voice Assistants.” *Computers & Security* 137: 103617.

Douladiris, Anargyros, and Efthimios Alepis. 2023. “Covid-19 New Cases Correlation Analysis: Weather Conditions, Citizen Traffic and Vaccination Statistics Impact in NARX Estimated Regressions in Attica, Greece.” In *2023 14th International Conference on Information, Intelligence, Systems & Applications (IISA)*, 1–7. IEEE.

Giannikis, Athanasios, Efthimios Alepis, and Maria Virvou. 2021. “Crowdsourcing Recognized Image Objects in Mobile Devices Through Machine Learning.” In *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, 560–67. IEEE.

Kapetanios, Constantinos, Theodoros Polyzos, Efthimios Alepis, and Constantinos Patsakis. 2021. “This Is Just Metadata: From No Communication Content to User Profiling, Surveillance and Exploitation.” *Advances in Core Computer Science-Based Technologies: Papers in Honor of Professor Nikolaos Alexandris*, 277–302.

Kontogianni, Aristea, and Efthimios Alepis. 2020. “Smart Tourism: State of the Art and Literature Review for the Last Six Years.” *Array* 6: 100020.

———. 2022. “AI, Blockchain & Cyber Tourism Joining the Smart Tourism Realm.” In *2022 13th International Conference on Information, Intelligence, Systems & Applications (IISA)*, 1–6. IEEE.

———. 2023. “Social Network Data Enabling Smart Tourism.” In *2023 14th International Conference on Information, Intelligence, Systems & Applications (IISA)*, 1–6. IEEE.

Kontogianni, Aristea, Efthimios Alepis, and Constantinos Patsakis. 2022a. “Promoting Smart Tourism Personalised Services via a Combination of Deep Learning Techniques.” *Expert Systems with Applications* 187: 115964.

———. 2022b. “Smart Tourism and Artificial Intelligence: Paving the Way to the Post-Covid-19 Era.” *Advances in Artificial Intelligence-Based Technologies: Selected Papers in Honour of Professor Nikolaos G. Bourbakis—Vol. 1*, 93–109.

Matzavela, Vasiliki, and Efthimios Alepis. 2021. “M-Learning in the COVID-19 Era: Physical Vs Digital Class.” *Education and Information Technologies* 26 (6): 7183–203.

———. 2023. “An Application of Self-Assessment of Students in Mathematics with Intelligent Decision Systems: Questionnaire, Design and Implementation at Digital Education.” *Education and Information Technologies*, 1–16.

Michail, Tselepatiotis, and Efthimios Alepis. 2023. “Design of Real-Time Multiplayer Word Game for the Android Platform Using Firebase and Fuzzy Logic.” In *2023 14th International Conference on Information, Intelligence, Systems & Applications (IISA)*, 1–8. IEEE.

Patsakis, Constantinos, Eugenia Politou, Efthimios Alepis, and Julio Hernandez-Castro. 2023. “Cashing Out Crypto: State of Practice in Ransom Payments.” *International Journal of Information Security*, 1–14.

Politou, Eugenia, Efthimios Alepis, Maria Virvou, and Constantinos Patsakis. 2022. “Privacy and Data Protection Challenges in the Distributed Era.” Springer.

Politou, Eugenia, Efthimios Alepis, Maria Virvou, Constantinos Patsakis, Eugenia Politou, Efthimios Alepis, Maria Virvou, and Constantinos Patsakis. 2022a. “Open Questions and Future Directions.” *Privacy and Data Protection Challenges in the Distributed Era*, 175–80.

———. 2022b. “State-of-the-Art Technological Developments.” *Privacy and Data Protection Challenges in the Distributed Era*, 69–91.

Sigala, Effrosyni, Efthimios Alepis, and Constantinos Patsakis. 2020. “Measuring the Quality of Street Surfaces in Smart Cities Through Smartphone Crowdsensing.” In *2020 11th International Conference on Information, Intelligence, Systems and Applications (IISA*, 1–8. IEEE.

Triantafyllou, Andreas M, George A Tsihrintzis, Maria Virvou, and Efthimios Alepis. 2021. “A Bimodal System for Emotion Recognition via Computer of Known or Unknown Persons in Normal or Fatigue Situations.” In *Advances in Core Computer Science-Based Technologies*, 9–35. Springer, Cham.

Virvou, Maria, Efthimios Alepis, George A Tsihrintzis, and Lakhmi C Jain. 2020. “Machine Learning Paradigms: Advances in Learning Analytics.” *Machine Learning Paradigms: Advances in Learning Analytics*, 1–5.