ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάπτυξη Ιστοσελίδας/E-shop μουσικών ειδών

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

του

Δημητριάδης Κωνσταντίνος

(ΑΕΜ: 2574)

Επιβλέπων : Βέργαδος Δημήτριος

Καστοριά Νοέμβριος 2024

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάπτυξη Ιστοσελίδας/E-shop μουσικών ειδών

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

του

Δημητριάδης Κωνσταντίνος

(ΑΕΜ: 2574)

Επιβλέπων : Βέργαδος Δημήτριος

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 04/11/2024

Καστοριά Νοέμβριος 2024

Ευχαριστίες

Θα ήθελα να εκφράσω τις ευχαριστίες μου προς όλους του καθηγητές που συνάντησα στην φοιτητική μου πορεία, αλλά ιδιαίτερα τον κ. Δημήτριο Βέργαδο ο οποίος σαν επιβλέπων καθηγητής ήταν δίπλα μου σε οτιδήποτε απορία και βοήθεια χρειαζόμουν για την ολοκλήρωση της πτυχιακής μου εργασίας.

Επίσης θα ήθελα να ευχαριστήσω τους συμφοιτητές μου Παναγιώτη Εγιούπι και Χαράλαμπο Ζευγολάτη, οι οποίοι μου χάρισαν ωραίες αναμνήσεις στην πόλη της Καστοριάς και μία δυνατή φιλία η οποία ακόμα και την αυτή την στιγμή κρατάει,

Τέλος θα ήθελα να ευχαριστήσω από τα βάθη της καρδιάς μου, την οικογένεια μου για την στήριξη που μου παρείχαν την διάρκεια των σπουδών μου.

Περίληψη

Η παρούσα πτυχιακή εργασία επικεντρώνεται στην ανάπτυξη μιας ιστοσελίδας, εστιάζοντας συγκεκριμένα στη δημιουργία ενός ηλεκτρονικού καταστήματος (e-shop) για μουσικά είδη. Το εν λόγω e-shop προσφέρει μια εκτενή γκάμα μουσικών οργάνων, ενισχυτών, χορδών και αξεσουάρ, εξυπηρετώντας έτσι τους χρήστες που επιθυμούν αγορές στον τομέα αυτόν. Με τη χρήση σύγχρονων τεχνολογιών διαδικτύου, έχει επιτευχθεί η δημιουργία και του front-end με χρήση HTML5, CSS3, JavaScript,jQuery καθώς και του back-end με χρήση Node.js, Express.js και MySQL. Τα κομμάτια κώδικα που έχουν αναπτυχθεί εξυπηρετούν τόσο την πλοήγηση των χρηστών στο e-shop όσο και την εξυπηρέτηση των αγορών τους, καθώς και τη διαδικασία δημιουργίας λογαριασμού. Στη διαδικασία ανάπτυξης της ιστοσελίδας, λαμβάνονται υπόψη οι αρχές του σχεδιασμού και της ανάπτυξης μιας επιτυχημένης ιστοσελίδας/e-shop. Η σχεδίαση περιλαμβάνει την εξέταση στατικών και δυναμικών στοιχείων της ιστοσελίδας, καθώς και την εφαρμογή των αρχών της ασφάλειας και προστασίας δεδομένων. Επίσης, δίνεται έμφαση στη βελτίωση της εμπειρίας του χρήστη και στον σχεδιασμό της διεπαφής του χρήστη (UX/UI), με στόχο τη δημιουργία μιας ευχάριστης και λειτουργικής περιήγησης στο e-shop.

Επιπλέον, εξετάζονται οι τάσεις και οι καινοτομίες στον σχεδιασμό ιστοσελίδων για κινητές συσκευές, λαμβάνοντας υπόψη τη σημασία της φιλικότητας προς το χρήστη και την προσαρμοστικότητα σε διάφορες οθόνες και συσκευές.

Λέξεις Κλειδιά: Ιστοσελίδα ,E-shop, Μουσικά είδη, Ηλεκτρονικό κατάστημα,Κατηγορίες προϊόντων, Διαχείριση περιεχομένου, Ανάλυση τεχνολογιών,Front-end, Back-end,

Ασφάλεια δεδομένων ,Εμπειρία χρήστη, Σύγχρονες τεχνολογίες, Προσαρμοστική σχεδίαση για διάφορες οθόνες (Responsive Design), Σχεδιασμός για κινητές συσκευές

Abstract

The present thesis focuses on developing a website, specifically targeting the creation of an e-shop for musical items. The aforementioned e-shop offers an extensive range of musical instruments, amplifiers, strings, and accessories, catering to users interested in purchasing within this domain. By utilizing modern web technologies, both the front-end and back-end have been developed using HTML5, CSS3, JavaScript, jQuery, Node.js, Express.js, and MySQL. The pieces of code developed serve the navigation of users within the e-shop, their purchases, as well as the account creation process.

Throughout the website development process, principles of designing and building a successful website/e-shop are taken into consideration. The design encompasses the examination of static and dynamic elements of the website, along with the implementation of principles of security and data protection. Moreover, emphasis is placed on improving the user experience and designing the user interface (UX/UI) with the aim of creating a pleasant and functional browsing experience on the e-shop.

Additionally, trends and innovations in website design for mobile devices are explored, considering the importance of user-friendliness and adaptability to various screens and devices.

Keywords: Website, E-shop, Musical items, Online store Product categories, Content management, Technology analysis Front-end, Back-end, Data security, User experience, Modern technologies, Responsive Design, Mobile Design.

Πίνακας Περιεχομένων

1 ΣΧΕΔΙΑΣΗ ΚΑΙ ΑΝΑΠΤΥΞΗ ΙΣΤΟΣΕΛΙΔΑΣ/E-SHOP

Το κεφάλαιο της Σχεδίασης και Ανάπτυξης Ιστοσελίδας/E-Shop αποτελεί ένα κρίσιμο στάδιο στη διαδικασία δημιουργίας ενός ηλεκτρονικού καταστήματος ή μιας ιστοσελίδας με σκοπό την παρουσίαση περιεχομένου, πληροφοριών ή υπηρεσιών στο διαδίκτυο. Η επιτυχής υλοποίηση απαιτεί στρατηγικό σχεδιασμό και ανάπτυξη, λαμβάνοντας υπόψη τις ανάγκες του κοινού και τις τεχνολογικές απαιτήσεις.

Στο πλαίσιο αυτού του κεφαλαίου, θα εξετάσουμε τις βασικές αρχές της σχεδίασης και ανάπτυξης ιστοσελίδων και e-shop, αναλύοντας τις τεχνολογίες web που χρησιμοποιούνται. Μέσω της πλήρους ανάλυσης αυτών των τεχνολογιών, θα κατανοήσουμε καλύτερα την παρουσίαση της ιστοσελίδας (front-end) καθώς και τον λειτουργικό κώδικα που εκτελείται πίσω από τις σκηνές (back-end).

Θα διερευνήσουμε τις σύγχρονες τεχνολογίες web που χρησιμοποιούνται στην ανάπτυξη ιστοσελίδων και e-shop. Αυτές περιλαμβάνουν τις γλώσσες προγραμματισμού HTML, CSS και JavaScript. Η χρήση τεχνικών της JavaScript, όπως το AJAX (Asynchronous JavaScript and XML), εξυπηρετεί πολλά σημεία στον κώδικα, επιτρέποντας ασύγχρονη επικοινωνία με τον server χωρίς ανανέωση της σελίδας, βελτιώνοντας την εμπειρία του χρήστη.

Επιπλέον, χρησιμοποιήσαμε το Node.js ως τον κύριο server, με το module του Express, για την ανάπτυξη του back-end μέρους της ιστοσελίδας. Χρησιμοποιούμε επίσης το XAMPP για τη δημιουργία βάσης δεδομένων MySQL και για τον τοπικό host της ιστοσελίδας, επιτρέποντας την ανάπτυξη και τον έλεγχο του site πριν την αναρτησή του στο διαδίκτυο. Επιπλέον, χρησιμοποιούμε το framework Bootstrap για τη δημιουργία ανταποκριτικών και ευέλικτων ιστοσελίδων, προσφέροντας υψηλή απόδοση και ανταπόκριση στις ανάγκες των χρηστών.

Μέσω αυτής της ανάλυσης, αποκτούμε μια πιο ολοκληρωμένη κατανόηση του τρόπου λειτουργίας της διαδικασίας σχεδίασης και ανάπτυξης ιστοσελίδων και e-shop, καθώς και τον τρόπο σύνδεσης των τεχνολογιών για τη δημιουργία ενός λειτουργικού και ελκυστικού διαδικτυακού περιβάλλοντος.

1.1 Τεχνολογίες Web Front-End Development

1.1.1 HTML (HyperText Markup Language):

Η HTML αποτελείται από στοιχεία που περιλαμβάνουν ετικέτες, οι οποίες τοποθετούνται ανάμεσα σε σύμβολα < και > (<tag>), μέσα στο περιεχόμενο μιας ιστοσελίδας. Συνήθως, οι HTML ετικέτες έρχονται σε ζεύγη, με την πρώτη να καλείται ετικέτα έναρξης και τη δεύτερη ετικέτα λήξης . Στο διάστημα μεταξύ των ετικετών, οι δημιουργοί ιστοσελίδων μπορούν να ενσωματώσουν κείμενο, εικόνες, πίνακες και άλλα στοιχεία. Οι browsers έχουν ως στόχο να αναγνωρίζουν τη μορφή HTML και να τις μετατρέπουν σε ιστοσελίδες που είναι προσβάσιμες σε χρήστες. Οι ετικέτες HTML δεν εμφανίζονται στον χρήστη, αλλά βοηθούν τον περιηγητή να οργανώσει την εμφάνιση του περιεχομένου της σελίδας. Όλοι οι ιστότοποι βασίζονται σε HTML στοιχεία. Μέσω της HTML, είναι δυνατή η προσθήκη εικόνων και άλλων αντικειμένων μέσα στη σελίδα, ενώ παρέχεται και η δυνατότητα εμφάνισης διαδραστικών φορμών. Επιτρέπει τη δημιουργία δομημένων εγγράφων, συνδυάζοντας το περιεχόμενο με τον κώδικα μορφοποίησης, προσδιορίζοντας σημαντικά στοιχεία όπως οι κεφαλίδες, οι παράγραφοι, οι λίστες, οι σύνδεσμοι και οι παραθέσεις. Μπορούν να προστεθούν γλώσσες σεναρίων όπως η JavaScript, που δίνουν δυναμική συμπεριφορά στις ιστοσελίδες, καθιστώντας τες διαδραστικές αντί για στατικές. Πλέον η πιο διαδεδομένη έκδοση είναι η HTML5 που είναι η πιο σύγχρονη μορφή της HTML.[1]

1.1.2 CSS (Cascading Style Sheets) & Bootstrap:

Η CSS (Cascading Style Sheets) αποτελεί μια γλώσσα προγραμματισμού που εντάσσεται στην κατηγορία των γλωσσών στυλ και χρησιμοποιείται για τον καθορισμό της εμφάνισης εγγράφων που έχουν δημιουργηθεί με γλώσσες σήμανσης. Η CSS διαδραματίζει κρίσιμο ρόλο στη μορφοποίηση και παρουσίαση ιστοσελίδων που είναι γραμμένες σε HTML ή XHTML, προσφέροντας τη δυνατότητα ελέγχου της οπτικής τους διάταξης. Σκοπός της CSS είναι η διαχείριση του στυλ μιας ιστοσελίδας, επιτρέποντας τη ρύθμιση χαρακτηριστικών όπως τα χρώματα, η στοίχιση και άλλες μορφοποιήσεις, προσφέροντας περισσότερες δυνατότητες σε σύγκριση με την απλή HTML. Η πιο πρόσφατη έκδοση είναι η CSS3, ενώ υπάρχουν και άλλες εκδόσεις όπως η SASS (Syntactically Awesome Style Sheet) και η SCSS (Sassy CSS).[2]

Το Bootstrap είναι ένα δωρεάν λογισκμικό, ανοικτού κώδικα πλαίσιο ανάπτυξης front-end για τη δημιουργία ιστοσελίδων και εφαρμογών ιστού. Διαθέτει στον χρήστη πολλές επιλογές γραφικών στοιχείων όπως τα navbars,footers,γραφικά πλαίσια αλλά και scripts όπως slideshows,dropdowns κ.α. τα οποία καλούνται μέσω της βιβλιοθήκης του bootstrap.

Ως framework, το Bootstrap περιλαμβάνει τα βασικά για την ανταπόκριση της ανάπτυξης ιστοσελίδων, ώστε οι προγραμματιστές να χρειάζεται μόνο να εισάγουν τον κώδικα σε ένα προκαθορισμένο σύστημα πλέγματος. Το πλαίσιο Bootstrap είναι χτισμένο πάνω σε Hypertext Markup Language (HTML), Cascading Style Sheets (CSS) και JavaScript. Οι web

[1] https://el.wikipedia.org/wiki/HTML
[2] https://el.wikipedia.org/wiki/CSS

προγραμματιστές που χρησιμοποιούν το Bootstrap μπορούν να κατασκευάσουν ιστοσελίδες πολύ πιο γρήγορα χωρίς να ανησυχούν για βασικές εντολές και λειτουργίες.[3]

1.1.3 JavaScript:

Η JavaScript (JS) αποτελεί μια διερμηνευόμενη γλώσσα προγραμματισμού που έχει σχεδιαστεί για υπολογιστές. Δημιουργήθηκε με σκοπό τη χρήση της στους web browsers, επιτρέποντας στα client-side σενάρια να αλληλεπιδρούν με τους χρήστες, να μεταφέρουν δεδομένα ασύγχρονα και να τροποποιούν δυναμικά το περιεχόμενο των ιστοσελίδων. Η JavaScript είναι μια γλώσσα σεναρίων που βασίζεται σε πρωτότυπα, χαρακτηρίζεται από δυναμικότητα και αδύναμη τυποποίηση, ενώ οι συναρτήσεις της αντιμετωπίζονται ως αντικείμενα πρώτης τάξης. Η σύνταξή της έχει επηρεαστεί από τη γλώσσα C. Αν και η JavaScript δανείζεται ορισμένες ονομασίες και συντακτικά στοιχεία από τη γλώσσα Java, οι δύο γλώσσες δεν είναι συγγενείς και διαφέρουν σημαντικά στη σημασιολογία τους Εκτός από τις ιστοσελίδες, η JavaScript χρησιμοποιείται επίσης σε άλλες εφαρμογές, όπως στα έγγραφα PDF, σε εξειδικευμένους περιηγητές ιστού και σε μικρές εφαρμογές επιφάνειας εργασίας (widgets). Οι σύγχρονες πλατφόρμες όπως το Node.js έχουν κάνει τη JavaScript ιδιαίτερα δημοφιλή και για την server-side ανάπτυξη εφαρμογών.[4]

1.1.3.1 Τύποι και εργαλεία της JavaScript:

Η JavaScript αποτελεί ένα από τα κύρια εργαλεία στον κόσμο του web development, προσφέροντας μια πληθώρα προγραμματιστικών δυνατοτήτων τόσο στο front-end όσο και στο back-end σημείου του κώδικα. Πέραν της πηγαίας της μορφής για τη δημιουργία λειτουργικοτήτων, υπάρχουν και διάφοροι τύποι και εργαλεία που ενισχύουν την αποτελεσματικότητα και την ευελιξία του κώδικα.

- **DOM (Document Object Model)** : Το DOM είναι ένα μοντέλο που αναπαριστά τη σελίδα HTML ως δέντρο αντικειμένων, επιτρέποντας στους προγραμματιστές να αλλάζουν δυναμικά τη δομή, το περιεχόμενο και τις στυλ της σελίδας.[5]
- **JSON (JavaScript Object Notation):** Το JSON είναι ένας ελαφρύς τρόπος ανταλλαγής δεδομένων μεταξύ του πελάτη και του διακομιστή. Είναι εύκολος στην ανάγνωση και την εγγραφή από ανθρώπους και μηχανές.[6]

[3] https://getbootstrap.com/
[4] https://el.wikipedia.org/wiki/JavaScript
[5] https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction
[6] https://www.json.org/json-en.html

- **AJAX (Asynchronous JavaScript and XML):** Η τεχνική AJAX επιτρέπει την ασύγχρονη ανταλλαγή δεδομένων με τον διακομιστή χωρίς την ανάγκη ανανέωσης της σελίδας. Αυτό οδηγεί σε μια καλύτερη εμπειρία χρήστη και αποδοτικότερες εφαρμογές.[7]
- **jQuery:** Το jQuery είναι μια δημοφιλής βιβλιοθήκη JavaScript που απλοποιεί τη διαχείριση του DOM, την ανάπτυξη εφέ και τις αιτήσεις AJAX, καθιστώντας την ανάπτυξη ιστοσελίδων πιο εύκολη και γρήγορη.[8]
- **Ασύγχρονο JavaScript (Asynchronous JavaScript):** Η ασύγχρονη φύση της JavaScript επιτρέπει την εκτέλεση κώδικα χωρίς να αναστέλλεται η εκτέλεση του υπόλοιπου κώδικα. Αυτό είναι ιδιαίτερα χρήσιμο για την ανταλλαγή δεδομένων, τη διαχείριση συμβάντων και άλλες εργασίες που απαιτούν ασυγχρόνιση. Μερικές τεχνικές των ασύγρονων JavaScript είναι τα callbacks,promises,async/await, οι οποίες είναι αρκετά χρήσιμες όταν πρέπει να χειρίστουμε rquests και responses ή ακομα και μυνήματα που έχουμε ορίσει απο τον server μας.[9]

Ο συνδυασμός των παραπάνω τύπων και εργαλείων εξυπηρετούν στην δημιουργία δυναμικών και αποτελεσματικών ιστοσελίδων καθώς και εφαρμογές που ανταποκρίνονται στις απαιτήσεις των χρηστών και των εφαρμογών τους. Η κατανόηση και η επιδεξιότητα στη χρήση αυτών των εργαλείων είναι ουσιώδης για κάθε προγραμματιστή που ασχολείται με το web development.

1.2 Τεχνολογίες Web Back-End Development

1.2.1 Node.js/Express.js:

Το Node.js είναι μια πλατφόρμα για την ανάπτυξη λογισμικού, κυρίως για διακομιστές, που βασίζεται στη JavaScript. Σκοπός του είναι να διευκολύνει τη δημιουργία διαδικτυακών εφαρμογών που μπορούν να επεκταθούν. Σε αντίθεση με άλλες σύγχρονες πλατφόρμες ανάπτυξης εφαρμογών δικτύων, μια διεργασία του Node δεν βασίζεται σε ένα μοντέλο ασύγχρονης εισόδου/εξόδου. Στη σημερινή εποχή, η καθυστέρηση και η απόδοση είναι σημαντικοί δείκτες για τη λειτουργία των διακομιστών στο διαδίκτυο. Το Node.js, βασισμένο σε ένα περιβάλλον εκτέλεσης JavaScript, επιτυγχάνει χαμηλή καθυστέρηση και

[7] https://www.freecodecamp.org/news/ajax-tutorial/
[8] https://api.jquery.com/
[9] https://www.freecodecamp.org/news/asynchronism-in-javascript/

υψηλή απόδοση, αποφεύγοντας τη σπατάλη χρόνου και πόρων κατά την αναμονή για αιτήματα εισόδου/εξόδου (I/O).[10]

Το Express.js, που μερικές φορές αναφέρεται και ως "Express", είναι ένα μινιμαλιστικό, γρήγορο και παρόμοιο με το Sinatra πλαίσιο υποστήριξης Node.js που παρέχει ισχυρές δυνατότητες και εργαλεία για την ανάπτυξη κλιμακούμενων εφαρμογών υποστήριξης. Σας παρέχει το σύστημα δρομολόγησης και απλοποιημένες δυνατότητες για να επεκτείνετε το πλαίσιο αναπτύσσοντας πιο ισχυρά στοιχεία και εξαρτήματα ανάλογα με τις περιπτώσεις χρήσης της εφαρμογής σας.

Το πλαίσιο παρέχει ένα σύνολο εργαλείων για εφαρμογές Ιστού, αιτήματα και απαντήσεις HTTP, δρομολόγηση και ενδιάμεσο λογισμικό για τη δημιουργία και την ανάπτυξη εφαρμογών μεγάλης κλίμακας, έτοιμες για επιχειρήσεις.

Παρέχει επίσης ένα εργαλείο διεπαφής γραμμής εντολών (CLI) που ονομάζεται Node Package Manager (NPM), όπου οι προγραμματιστές μπορούν να προμηθεύονται τα ανεπτυγμένα πακέτα. Αναγκάζει επίσης τους προγραμματιστές να ακολουθήσουν την αρχή Don't Repeat Yourself (DRY).

Η αρχή DRY στοχεύει στη μείωση της επανάληψης των μοτίβων λογισμικού, στην αντικατάστασή της με αφαιρέσεις ή στη χρήση κανονικοποιήσεων δεδομένων για την αποφυγή πλεονασμού.[11]

1.2.2 XAMPP (Apache & MySQL):

Το XAMPP είναι ένα λογισμικό ανοιχτού κώδικα ελεύθερης χρήσης, που μπορεί να τρέξει σε διάφορα λειτουργικά συστήματα. Περιλαμβάνει τον εξυπηρετητή ιστοσελίδων Apache, τη βάση δεδομένων MySQL και έναν διερμηνέα για τις γλώσσες προγραμματισμού PHP και Perl.

Το όνομα XAMPP είναι ένα ακρωνύμιο που αντιπροσωπεύει τα εξής:

- X: «Cross-platform» (αναφέρεται στη δυνατότητα εκτέλεσης σε διαφορετικά λειτουργικά συστήματα)
- Apache
- MySQL
- PHP
- Perl

Το XAMPP, ως ελεύθερο λογισμικό, περιλαμβάνει έναν εξυπηρετητή ιστοσελίδων που μπορεί να διαχειριστεί δυναμικές ιστοσελίδες βασισμένες σε PHP και MySQL. Είναι συμβατό με πολλά λειτουργικά συστήματα, όπως Windows, Linux και Mac OS X, και

[10] https://en.wikipedia.org/wiki/Node.js
[11] https://en.wikipedia.org/wiki/Express.js https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs

χρησιμοποιείται ευρέως για τη σχεδίαση και ανάπτυξη ιστοσελίδων με τεχνολογίες όπως PHP, JSP και Servlets.

Κατά την εγκατάσταση, το XAMPP απαιτεί μόνο ένα πρόγραμμα συμπίεσης αρχείων, όπως zip, tar, 7z ή exe. Παρέχει επίσης τη δυνατότητα αναβάθμισης των κύριων συνιστωσών του, όπως του εξυπηρετητή Apache, της βάσης MySQL, και των γλωσσών PHP και Perl. Επιπλέον, περιλαμβάνει το OpenSSL και το phpMyAdmin για διαχείριση των βάσεων δεδομένων.[12]

1.2.3 APIs:

Το API (Application Programming Interface) είναι η διεπαφή που επιτρέπει την ανταλλαγή δεδομένων ανάμεσα σε μια εφαρμογή ή πλατφόρμα και έναν διακομιστή, επιστρέφοντας τις επιθυμητές απαντήσεις και αποτελέσματα. Παράδειγμα της λειτουργίας ενός API είναι οι πλατφόρμες που δρουν ως μεσολαβητές για τα e-shops, όπως το Skroutz. Ο χρήστης εισάγει τα δεδομένα αναζήτησης στην πλατφόρμα, η οποία στη συνέχεια επικοινωνεί με τις ιστοσελίδες των επιχειρήσεων για να δώσει τις απαραίτητες πληροφορίες. Ένα άλλο παράδειγμα χρήσης API είναι το Amazon Web Services (AWS).

Ο χρήστης μπαίνει στην πλατφόρμα και καταχωρεί τα δεδομένα της αναζήτησής που επιθυμεί.

Στη συνέχεια, η πλατφόρμα “επικοινωνεί” με την ιστοσελίδα της κάθε επιχείρησης ώστε να δώσει πίσω τις πληροφορίες που χρειάζεται κάποιος.

Επίσης, ένα άλλο γνωστό παράδειγμα του πώς χρησιμοποιείται το API, αποτελεί και το Amazon Web Services (AWS).

Τα βασικά πλεονεκτήματα των API είναι τα εξής:

- **Ταχύτερη Ανάπτυξη**
- **Άμεση Βελτίωση**
- **Έλεγχος Δεδομένων**
- **Ενσωμάτωση**
- **Επεκτασιμότητα**

Υπάρχουν τέσσερις βασικοί τύποι API:

- **Open API**: Είναι δημόσιο και προσβάσιμο σε όλους μέσω του πρωτοκόλλου HTTP, χωρίς περιορισμούς στην πρόσβαση.
- **Partner API**: Απαιτεί ειδικές άδειες και διαπιστευτήρια για την πρόσβαση, συνήθως μέσω μιας διαδικασίας ενσωμάτωσης.

[12] https://el.wikipedia.org/wiki/XAMPP

- **Internal API**: Δημιουργείται για εσωτερική χρήση εντός μιας εταιρείας και δεν είναι διαθέσιμο στο ευρύ κοινό, με σκοπό τη βελτίωση της παραγωγικότητας των ομάδων ανάπτυξης.
- **Composite API**: Συνδυάζει διάφορα API, επιτρέποντας στους προγραμματιστές να αποκτούν πρόσβαση σε πολλές υπηρεσίες με μία μόνο κλήση.[13]

Όλες οι παραπάνω γλώσσες και τεχνολογίες χρησιμοποιήθηκαν με τον κατάλληλο προγραμματισμό για να επιτευχθεί το πρακτικό κομμάτι της παρούσας πτυχιακής εργασίας. Υπάρχουν πολλοί τρόποι για την ανάπτυξη μιας ιστιοσελίδας, αλλά ύστερα από προσωπική έρευνα, θεωρήθηκαν κατάλληλες οι συγκεκριμένες για να επιτευχθούν λειτουργείες όπως η δημιουργία του server, η παρουσίαση και το στυλ των σελίδων, η επικοινωνία του κώδικα με τις βάσεις δεδομένων κ.α.

2 ΣΤΑΤΙΚΕΣ ΚΑΙ ΔΥΝΑΜΙΚΕΣ ΙΣΤΟΣΕΛΙΔΕΣ

Σε αυτό το κεφάλαιο θα πραγματοποιηθεί ανάλυση μεταξύ των στατικών και δυναμικών ιστοσελίδων, με έμφαση στη χρησιμότητά, τη δομή και τις διαφορές τους. Θα εξεταστούν οι τεχνικές που χρησιμοποιούνται για την ανάπτυξή τους, καθώς και οι πρακτικές που συντελούν στη βελτιστοποίηση της λειτουργικότητάς τους και τη βελτίωση της εμπειρίας του χρήστη. Μέσω αυτής της ανάλυσης, θα διαπιστωθεί η σημασία και η αποτελεσματικότητα κάθε τύπου ιστοσελίδας στον σύγχρονο διαδικτυακό χώρο.

2.1 Στατικές Ιστοσελίδες

Μια στατική ιστοσελίδα είναι ένας τύπος ιστοσελίδας όπου το περιεχόμενο παραδίδεται στον χρήστη ακριβώς όπως είναι αποθηκευμένο στον διακομιστή, σε αντίθεση με τις δυναμικές ιστοσελίδες που δημιουργούν το περιεχόμενο μέσω εφαρμογών που εκτελούνται στον διακομιστή.

Σε στατικές ιστοσελίδες, το περιεχόμενο εμφανίζεται ίδιο για όλους τους χρήστες και αποθηκεύεται στο σύστημα αρχείων του διακομιστή. Συνήθως είναι σε μορφή HTML και μεταδίδεται μέσω του πρωτοκόλλου HTTP.

Πλεονεκτήματα:

[13] https://bigblue.academy/gr/ti-einai-to-api

- Μια στατική σελίδας δεν απαιτεί γνώσεις προγραμματισμού.
- Μπορεί εύκολα να αντιγραφεί και να φιλοξενηθεί σε πολλούς διακομιστές.
- Δεν απαιτείται εξειδικευμένο λογισμικό στον διακομιστή για τη δημοσίευση στατικών σελίδων.
- Μια στατική σελίδα μπορεί να προβληθεί από αποθηκευτικά μέσα, όπως CD-ROM ή USB, χωρίς την ανάγκη ενός διακομιστή με εξειδικευμένο λογισμικό. Επίσης, συνήθως είναι πιο ασφαλής.
- Οι στατικές ιστοσελίδες έχουν χαμηλότερο κόστος.
- Λόγω της απλότητάς τους, οι στατικές ιστοσελίδες φορτώνουν γρήγορα.
- Εύκολη αποκατάσταση σε περίπτωση σφαλμάτων
- Δεν απαιτούνται σύνθετα λογισμικά για τη λειτουργία τους.
- Η φιλοξενία τους είναι οικονομική, αφού χρειάζονται λιγότερους πόρους συγκριτικά με τις δυναμικές ιστοσελίδες.

Μειονεκτήματα:

- Η αλληλεπίδραση με τον χρήστη είναι περιορισμένη.
- Η διαχείριση μεγάλου αριθμού στατικών ιστοσελίδων είναι δύσκολη χωρίς αυτοματοποιημένα εργαλεία.
- Περιορισμένες δυνατότητες σχεδίασης και εμφάνισης.
- Δεν προσφέρουν διαδραστικές εμπειρίες ή εξατομικευμένο περιεχόμενο στους επισκέπτες.
- Η ενημέρωση και η αλλαγή του περιεχομένου ή της δομής απαιτεί πολύ χρόνο και μπορεί να είναι δαπανηρή.
- Δυσκολία στην επέκταση και στην προσθήκη νέων λειτουργιών
- Υστερούν σε τομείς ψηφιακού μάρκετινγκ και προώθησης.

2.2 Δυναμικές ιστοσελίδες

Μια δυναμική ιστοσελίδα είναι μια ιστοσελίδα που δημιουργείται και προσαρμόζεται σε πραγματικό χρόνο κατά την πρόσβαση ή την αλληλεπίδραση του χρήστη με τον διακομιστή ιστοσελίδων. Στις δυναμικές ιστοσελίδες, το περιεχόμενο παράγεται με βάση ένα σενάριο εντολών, το οποίο μπορεί να εκτελείται είτε στον πελάτη είτε στον διακομιστή ή και στους δύο.

Η σελίδα συνήθως περιλαμβάνει σενάριο εντολών που επιτρέπει την αλληλεπίδραση με τον χρήστη μέσω πληκτρολογίου ή ποντικιού. Το περιεχόμενο της ιστοσελίδας διαμορφώνεται δυναμικά στον τοπικό υπολογιστή μετά την εκτέλεση του σεναρίου εντολών που έχει ληφθεί από τον απομακρυσμένο διακομιστή.

Η τεχνολογία AJAX χρησιμοποιεί σενάρια εντολών τόσο στον πελάτη όσο και στον απομακρυσμένο διακομιστή για τη δημιουργία δυναμικών ιστοσελίδων. Με τη χρήση AJAX, γίνεται ανταλλαγή δεδομένων μεταξύ του υπολογιστή του πελάτη και του διακομιστή, και η σελίδα προσαρμόζεται τοπικά στον πελάτη. Το πλεονέκτημα αυτής της τεχνολογίας είναι η μείωση του φόρτου στον διακομιστή, καθώς αποστέλλεται μόνο το περιεχόμενο που χρειάζεται. Ένα παράδειγμα μιας εφαρμογής που χρησιμοποιεί AJAX είναι οι χάρτες της Google.

Πλεονεκτήματα:

- Παρέχουν πολύ περισσότερες δυνατότητες και λειτουργίες
- Είναι διαδραστικές και μπορούν να προσαρμόζουν το περιεχόμενο ανάλογα με τις ανάγκες του χρήστη.
- Οι δυναμικές ιστοσελίδες δίνουν περισσότερες δυνατότητες σχεδίασης και η εμφάνιση τους είναι πιο ελκυστική.
- Απλή διαχείριση του περιεχομένου τους ακόμη και για άτομα με βασικές γνώσεις υπολογιστών και διαδικτύου.
- Πολλοί χρήστες μπορούν να επεξεργάζονται την ιστοσελίδα ταυτόχρονα.
- Ευκολότερες στο να πραγματοποιηθούν μεγάλες αλλαγές στον σχεδιασμό και να προστεθούν νέες λειτουργίες.
- Παρέχουν περισσότερες δυνατότητες για digital marketing, προσελκύοντας νέους επισκέπτες.

Μειονεκτήματα:

- Ενδέχεται να έχουν χαμηλότερη ταχύτητα φόρτωσης καθώς είναι πιο περίπλοκες και μπορεί να περιέχουν πολλές λειτουργίες ή να χρειαστεί περισσότερο χρόνο για να φορτώσει το περιεχόμενο.
- Το κόστος ανάπτυξης τους μπορεί να είναι υψηλότερο ανάλογα με το μέγεθος και τις δυνατότητές της.
- Το κόστος φιλοξενίας είναι αυξημένο λόγω των μεγαλύτερων απαιτήσεων σε πόρους.

Ύστερα από την ανάλυση των πλεονεκτημάτων και των μειονεκτημάτων μεταξύ των στατικών και δυναμικών ιστοσελίδων, εστιάσαμε στη χρησιμότητά τους, τη δομή τους και τις διαφορές μεταξύ τους. Κάθε είδος ιστοσελίδας παρουσιάζεται με τα πλεονεκτήματα και τα μειονεκτήματά του, ενισχύοντας την κατανόηση τους και τον τρόπο λειτουργία τους.

Συγκεκριμένα, οι στατικές ιστοσελίδες είναι πιο απλές στη δημιουργία και στη φιλοξενία, παρέχοντας σταθερό περιεχόμενο και χαμηλό κόστος. Ωστόσο, υστερούν σε διαδραστικότητα και δυνατότητες προσαρμογής στις ανάγκες του χρήστη.

Από την άλλη πλευρά, οι δυναμικές ιστοσελίδες προσφέρουν πολλαπλές λειτουργίες και δυνατότητες προσαρμογής, καθιστώντας τις πιο εντυπωσιακές και διαδραστικές. Ωστόσο, αυτές οι λειτουργίες ενδέχεται να οδηγήσουν σε μεγαλύτερο κόστος και περίπλοκη διαχείριση.

Συνολικά, η επιλογή μεταξύ στατικών και δυναμικών ιστοσελίδων εξαρτάται από τις ανάγκες και τους στόχους του κάθε ιδιοκτήτη ιστοσελίδας. Και οι δύο τύποι ιστοσελίδων έχουν τα πλεονεκτήματά τους και μπορούν να αποδειχθούν αποτελεσματικοί, ανάλογα με την κατάλληλη εφαρμογή και την καλή διαχείριση τους. Η μεγαλύτερη μερίδα των ιστοσελίδων που χρησιμοποιούνται καθημερινά από τους χρήστες είναι δυναμικές καθώς πλέον οι βάσεις δεδομένων είναι χρήσιμες για τις ιστοσελίδες, άρα η στατική μορφή ιστοσελίδας αδυνατεί σε τέτοιου είδους ανταποκρίσεις ως προς τους χρήστες.[14]

3 ΑΣΦΑΛΕΙΑ ΚΑΙ ΠΡΟΣΤΑΣΙΑ ΔΕΔΟΜΕΝΩΝ

Η ασφάλεια ενός e-shop αποτελεί κρίσιμη πτυχή και απαιτεί την εφαρμογή πολλαπλών μέτρων προστασίας. Καταρχάς, η χρήση πρωτοκόλλου HTTPS είναι απαραίτητη για την κρυπτογράφηση των δεδομένων μεταξύ του χρήστη και του διακομιστή. Επιπλέον, η διαχείριση των προσβάσεων με ισχυρούς κωδικούς και η συνεχής ενημέρωση του λογισμικού είναι θεμελιώδεις προϋποθέσεις. Η προστασία από κακόβουλο λογισμικό, η ασφαλής μετάδοση των πληρωμών και η ανάλυση δεδομένων ασφαλείας αποτελούν επίσης σημαντικές πτυχές. Επιπλέον, η δημοσίευση μιας σαφούς πολιτικής απορρήτου και η αντιμετώπιση πιθανών ευπαθειών στον κώδικα απαιτούνται. Όλα αυτά τα μέτρα συνθέτουν ένα ολοκληρωμένο σύστημα ασφάλειας που διασφαλίζει την προστασία του e-shop και των δεδομένων των χρηστών. Παρακάτω, θα αναλυθούν περισσότερο οι τεχνικές ασφαλείας και ο τρόπος με τον οποίο ενσωματώνονται, εγκαθίστανται και εκτελούνται μέσα στον κώδικα.

3.1 Πρωτόκολλο HTTPS(Hypertext Transfer Protocol Secure):

Το HTTPS (Hypertext Transfer Protocol Secure) αποτελεί μία τεχνολογία που διασφαλίζει την ασφαλή σύνδεση μέσω HTTP, με κρυπτογραφημένη ανταλλαγή δεδομένων. Όταν ένας σύνδεσμος (URL) ξεκινά με το https, αυτό σημαίνει ότι χρησιμοποιείται το πρωτόκολλο HTTP, η σύνδεση πραγματοποιείται μέσω της θύρας 443 αντί της θύρας 80 με

[14] https://iservices.gr/blog/statikes-dinamikes-istoselides/

κρυπτογραφημένα δεδομένα. Δεν προσφέρουν όλα τα πρωτόκολλα HTTPS το ίδιο επίπεδο ασφάλειας και αποτελεσματικότητας, με συνέπειες στην κατάταξη των ιστοσελίδων στις μηχανές αναζήτησης. Το σύστημα αυτό αναπτύχθηκε αρχικά από την Netscape Communications Corporation για την αυθεντικοποίηση χρηστών και την προστασία της επικοινωνίας σε ιστότοπους, ενώ σήμερα χρησιμοποιείται ευρέως σε ιστοσελίδες όπου απαιτείται υψηλότερο επίπεδο ασφαλείας, όπως σε περιπτώσεις μεταφοράς ευαίσθητων πληροφοριών (π.χ. αριθμοί πιστωτικών καρτών, κωδικοί πρόσβασης).

Παρόλο που συχνά θεωρείται ξεχωριστό πρωτόκολλο, το HTTPS είναι στην πραγματικότητα ένας συνδυασμός του HTTP με τις δυνατότητες κρυπτογράφησης που προσφέρει το Secure Sockets Layer (SSL). Η κρυπτογράφηση προστατεύει τα δεδομένα από υποκλοπές και επιθέσεις τύπου man-in-the-middle, αν και η αποτελεσματικότητα εξαρτάται από τη σωστή εφαρμογή των μεθόδων ασφάλειας και την ποιότητα των αλγορίθμων κρυπτογράφησης. Η ασφάλεια και η κατάταξη μιας ιστοσελίδας στις μηχανές αναζήτησης μπορεί να ποικίλλει ανάλογα με το πρωτόκολλο HTTPS που χρησιμοποιείται.

Για την ενεργοποίηση του HTTPS σε έναν διακομιστή, ο διαχειριστής πρέπει να αποκτήσει ένα πιστοποιητικό δημόσιου κλειδιού, το οποίο σε περιβάλλον UNIX μπορεί να παραχθεί μέσω του OpenSSL. Το πιστοποιητικό αυτό πρέπει να υπογραφεί από μια αρχή πιστοποίησης (Certificate Authority), η οποία διασφαλίζει τη νομιμότητα του εκδότη και την εγκυρότητα του πιστοποιητικού. Οι χρήστες, μέσω της υπογραφής αυτής, μπορούν να επαληθεύσουν την αυθεντικότητα του πιστοποιητικού και να διασφαλίσουν ότι δεν έχει υποστεί πλαστογράφηση από κακόβουλους χρήστες.

Το HTTPS χρησιμοποιείται κυρίως για την ασφαλή μεταφορά ευαίσθητων πληροφοριών. Το επίπεδο προστασίας εξαρτάται από την ορθή υλοποίηση της διαδικασίας ασφαλείας και την ισχύ των αλγορίθμων κρυπτογράφησης που εφαρμόζονται.[15]

3.2 Κύριες Απειλές Ιστοσελίδων και Προστασία Δεδομένων:

Κάθε ιστοσελίδα,κάθε e-shop,γενικώς οτιδήποτε ιστότοπος δίνει την δυνατότητα στον χρήστη να δημιουργήσει έναν λογαριασμό στα μέσα αυτά,πρέπει να διασφαλίζει την ασφάλεια των πληροφοριών με τις οποίες εγγράφονται.Οι πληροφορίες που μπορεί να δώσει μπορεί να είναι ονοματεπώνυμο,email,τόπος διαμονής,οδό,πληρωμές γενικώς προσωπικά στοιχεία,άρα τα site στα οποία εμπιστεύονται αυτες τις πληροφορίες πρέπει να έιναι ορατές αποκλειστικά στον χρήστη(δεδομένου ότι οι admins που διαχείριζονται τα δεδομένα έχουν την ευθύνη για οποιαδήποτε διαρροή).Ωστόσο υπάρχουν εξωτερικοί χρήστες οι οποίοι προσπαθούν να "χτυπήσουν" τις σελίδες με κακόβουλα λογισμικά με σκοπό είτε στην διάλυση της σελίδα ακόμα και στην περισυλλογή δεδομένων των χρηστών.Τα

[15] https://el.wikipedia.org/wiki/HTTPS

κύρια αίτια που μπορεί να οδηγήσουν σε καταστροφικά αποτελέσματα είναι ότι δεν υπάρχουν τα κατάλληλα και σύγχρονα πρωτόκολλα ασφαλείας είτε ότι ο κώδικας έχει ευπαθή σημεία.Με τον όρο ευπαθή σημεία στον κώδικα ενοούμε ότι δεν υπαρχόυν οι κατάλληλες ρυθμίσεις πχ User's authentication,στατικά σημεία,κοινή έκθεση πόρων και κρίσιμων σημείων του κώδικα κ.α.

3.2.1 Κύρια Σημεία που Απειλούνται οι Ιστοσελίδες/e-shops

Εν κατακλείδι, το παραπάνω κείμενο αναφέρεται στη σημασία της ασφάλειας δεδομένων σε ιστοσελίδες, e-shops και γενικά σε ιστότοπους που συλλέγουν προσωπικές πληροφορίες των χρηστών. Αναλύοντας το κείμενο, μπορούμε να επισημάνουμε τα ακόλουθα σημεία:

- **Ανάγκη Προστασίας Δεδομένων:** Οι ιστοσελίδες που συλλέγουν προσωπικές πληροφορίες χρηστών οφείλουν να διασφαλίζουν την ασφάλεια αυτών των δεδομένων.
- **Πιθανές Πληροφορίες Χρηστών:** Οι πληροφορίες που μπορεί να δώσει ένας χρήστης σε ιστότοπους μπορεί να είναι πολλές και ευαίσθητες, όπως ονοματεπώνυμο, email, διεύθυνση κατοικίας, διεύθυνση πληρωμής κλπ.
- **Ευθύνη Διαχειριστών:** Οι διαχειριστές των ιστοσελίδων έχουν την ευθύνη να προστατεύουν τα δεδομένα των χρηστών και να αποτρέπουν τυχόν διαρροές πληροφοριών.
- **Κίνδυνοι Εξωτερικών Χρηστών:** Υπάρχουν κακόβουλοι χρήστες που προσπαθούν να εισβάλλουν σε ιστότοπους και να αποκτήσουν πρόσβαση σε ευαίσθητα δεδομένα, με σκοπό την παράνομη χρήση τους.
- **Κύρια Αίτια Κινδύνου:** Τα κύρια αίτια που μπορούν να οδηγήσουν σε καταστροφικά αποτελέσματα είναι η έλλειψη κατάλληλων πρωτοκόλλων ασφαλείας και η ύπαρξη ευπαθών σημείων στον κώδικα του ιστότοπου.
- **Ευπαθή Σημεία στον Κώδικα:** Τα ευπαθή σημεία στον κώδικα μπορεί να περιλαμβάνουν απουσία κατάλληλης διαδικασίας ελέγχου ταυτότητας χρηστών, στατικά σημεία που διευκολύνουν επιθέσεις, κοινή έκθεση πόρων και άλλα

3.2.2 Τρόποι Προστασίας Δεδομένων

Παραπάνω αναφέρθηκαν οι κυριότεροι κίνδυνοι των ιστοσελίδων, στην συνέχεια θα εξεταστούν οι τρόποι αντιμετώπισης τους

- **Υλοποίηση Πολιτικών Ασφαλείας Δεδομένων:** Ανάπτυξη και εφαρμογή πολιτικών ασφαλείας που να καλύπτουν τη συλλογή, την αποθήκευση και την

επεξεργασία προσωπικών δεδομένων, συμπεριλαμβανομένων πολιτικών πρόσβασης και χρήσης δεδομένων.

- **Κρυπτογράφηση Δεδομένων:** Κρυπτογράφηση των δεδομένων για την προστασία των προσωπικών δεδομένων κατά τη μετάδοση και την αποθήκευσή τους στον διακομιστή.
- **Εκπαίδευση Προσωπικού:** Εκπαίδευση του προσωπικού σχετικά με τις βέλτιστες πρακτικές ασφαλείας και τις διαδικασίες που πρέπει να ακολουθούν για την προστασία των δεδομένων των χρηστών.
- **Ενημέρωση Κωδικής Βάσης:** Ενημέρωση του κώδικα της ιστοσελίδας για την αντιμετώπιση ευπαθών σημείων και την εφαρμογή κατάλληλων πρωτοκόλλων ασφαλείας.
- **Παρακολούθηση και Αντιμετώπιση Κινδύνων:** Παρακολούθηση των δραστηριοτήτων που αφορούν την ασφάλεια των δεδομένων και άμεση αντίδραση σε ενδεχόμενες απειλές ή παραβιάσεις.
- **Ενίσχυση Αυθεντικοποίησης Χρηστών:** Εφαρμογή αποτελεσματικών μηχανισμών αυθεντικοποίησης για τους χρήστες, όπως διπλή επαλήθευση και αναγνώριση παραβάσεων σε πραγματικό χρόνο

4 ΒΕΛΤΙΩΣΗ ΕΜΠΕΙΡΙΑΣ ΧΡΗΣΤΗ ΚΑΙ ΣΧΕΔΙΑΣΜΟΣ ΔΙΕΠΑΦΗΣ ΧΡΗΣΤΗ (UX/UI) ΣΕ ΙΣΤΟΣΕΛΙΔΕΣ

Το UI/UX design (User Interface/User Experience) αναφέρεται σε δύο έννοιες που χρησιμοποιούνται από ειδικούς στον τομέα της πληροφορικής και του ψηφιακού σχεδιασμού προϊόντων. Αφορούν τον σχεδιασμό και τη διαμόρφωση ψηφιακών προϊόντων και υπηρεσιών, όπως ιστοσελίδες, εφαρμογές και ηλεκτρονικά καταστήματα, με στόχο να ικανοποιούν τις ανάγκες των χρηστών. Ο βασικός στόχος ενός UI/UX designer είναι να εξασφαλίσει την εύκολη και αποτελεσματική πλοήγηση του χρήστη στις εφαρμογές ή τις ιστοσελίδες που δημιουργεί.

Ο UI/UX designer στον τομέα της πληροφορικής και του ψηφιακού σχεδιασμού προϊόντων έχει ως κύριο στόχο τη δημιουργία εφαρμογών και ιστοσελίδων που προσφέρουν μια ευχάριστη και λειτουργική εμπειρία στους χρήστες. Οι βασικοί στόχοι ενός UI/UX designer περιλαμβάνουν:

- **Ευχρηστία (Usability):** Η δημιουργία ενός προϊόντος που είναι εύκολο στη χρήση και κατανοητό για τους χρήστες.

- **Ελκυστικότητα του Τελικού Προϊόντος:** Η ανάπτυξη μιας ελκυστικής και εντυπωσιακής εμπειρίας χρήστη που θα κεντρίζει το ενδιαφέρον του κοινού.[16]

4.1 UI(User Interface)

Το User Interface (UI) αναφέρεται σε κάθε είδους οπτικό στοιχείο με το οποίο αλληλεπιδρά ο χρήστης, εστιάζοντας στη γραφική διάταξη και την αισθητική του. Ο όρος UI σχετίζεται με τον σχεδιασμό της διεπαφής ανάμεσα στον χρήστη και το προϊόν. Οι αρμοδιότητες ενός UI designer περιλαμβάνουν τη δημιουργία προσχεδίων, την εφαρμογή σχεδιαστικών μεθόδων και την ανάπτυξη μιας διεπαφής χρήστη που να είναι λειτουργική και οπτικά ελκυστική.

4.2 UX(User Experience)

Το User Experience (UX) αναφέρεται στην εμπειρία που έχει ο χρήστης κατά την αλληλεπίδρασή του με ένα website ή μία εφαρμογή. Η τελική μορφή του προϊόντος διαμορφώνεται με βάση τη συμπεριφορά και τα συναισθήματα του χρήστη, ώστε να ανταποκρίνεται στις επιθυμίες του. Το UX επικεντρώνεται στη δημιουργία μιας ευχάριστης, ολοκληρωμένης και ενδιαφέρουσας εμπειρίας για τον χρήστη. Οι στόχοι της UX περιλαμβάνουν τη βελτίωση της αισθητικής του προϊόντος, την αύξηση της ελκυστικότητάς του, καθώς και την ενίσχυση της αποτελεσματικότητας και της ικανοποίησης του χρήστη.

4.3 Σημαντικότητα του UI/UX

Το UI/UX αποτελούν ουσιώδεις παράγοντες στη δημιουργία μιας ιστοσελίδας καθώς διαμορφώνουν την εμπειρία του χρήστη και τον τρόπο με τον οποίο αλληλεπιδρά με το περιεχόμενο. Το UI (User Interface) αναφέρεται στη σχεδιαστική διάταξη και την εμφάνιση της ιστοσελίδας, ενώ το UX (User Experience) αφορά την ευχρηστία και τη συνολική εμπειρία που προσφέρει η ιστοσελίδα. Μέσω του καλού σχεδιασμού UI/UX, μια ιστοσελίδα μπορεί να παρέχει μια ευχάριστη, λειτουργική και αποτελεσματική εμπειρία στον χρήστη, βοηθώντας έτσι στην επίτευξη των στόχων της ιστοσελίδας και στην αύξηση της αποδοτικότητάς της. Υπάρχουν μερικοί τρόποι για να εξετάσουμε αλλά και να βελτιώσουμε το UI/UX

[16] https://en.wikipedia.org/wiki/User_experience_design

- **Ανάλυση Πελατών και Ερευνα Χρηστών**: Μια βασική πτυχή του UX/UI design είναι η κατανόηση των αναγκών και των προτιμήσεων των χρηστών. Μεθόδοι για την πραγματοποίηση έρευνας χρηστών περιλαμβάνουν συνεντεύξεις, έρευνα αγοράς, και καταγραφή συμπεριφορών. Η έρευνα αυτή επηρεάζει τον σχεδιασμό της διεπαφής καθώς παρέχει εμβάθυνση στις ανάγκες του κοινού.
- **Σχεδιασμός Χρήστη:** Ο σχεδιασμός χρήστη (user-centered design) επικεντρώνεται στις ανάγκες του χρήστη. Μέθοδοι περιλαμβάνουν τα πρωτότυπα χρήστη, τις δοκιμές χρήστη και την προτυποποίηση διεπαφών, βοηθώντας στη βελτίωση της εμπειρίας του χρήστη.
- **Ανάπτυξη Ανταγωνιστικών Πλεονεκτημάτων:** Ο καλός σχεδιασμός της διεπαφής και η βελτίωση της εμπειρίας του χρήστη αποτελούν ανταγωνιστικό πλεονέκτημα για μια ιστοσελίδα ή ένα e-shop, αυξάνοντας την ικανοποίηση των χρηστών και την πιθανότητα επιτυχίας τους.
- **Ανάλυση Σχεδιασμού Ιστοσελίδων:** Η ανάλυση αυτή παρουσιάζει παραδείγματα ιστοσελίδων και e-shops, αναδεικνύοντας τον τρόπο με τον οποίο ο σχεδιασμός της διεπαφής και η εμπειρία του χρήστη επηρεάζουν την απόδοση και την επιτυχία τους.
- **Νέες Τάσεις και Τεχνολογίες:** Η αναζήτηση νέων τάσεων στον σχεδιασμό διεπαφής και την εμπειρία του χρήστη, καθώς και η εφαρμογή νέων τεχνολογιών όπως η AI και η AR/VR, συνεισφέρει στη βελτίωση της εμπειρίας του χρήστη και στην ανάπτυξη ανταγωνιστικών πλεονεκτημάτων.

4.4 Μεθοδολογίες Σχεδιασμού UI/UX και Ανάλυση Χρήσης:

Οι μεθοδολογίες σχεδιασμού UI/UX και η ανάλυση χρήσης αποτελούν ζωτικά εργαλεία για τη βελτίωση της εμπειρίας του χρήστη και την ανάπτυξη αποτελεσματικών ψηφιακών προϊόντων. Ας εξετάσουμε πιο αναλυτικά κάθε πτυχή.

4.4.1 Μεθοδολογίες Σχεδιασμού UI/UX:

- **Design Thinking:** Αυτή η μέθοδος επικεντρώνεται στην κατανόηση των αναγκών του χρήστη και στη δημιουργία λύσεων που να ανταποκρίνονται σε αυτές. Οι φάσεις της περιλαμβάνουν τη συλλογή δεδομένων, την ανάλυση, τη δημιουργία προτύπων και τον πειραματισμό με λύσεις.
- **Human-Centered Design:** Στο επίκεντρο της μεθοδολογίας αυτής είναι οι ανάγκες και οι επιθυμίες των χρηστών. Οι σχεδιαστές εστιάζουν στη δημιουργία προϊόντων που να προσφέρουν απτή αξία στους χρήστες.
- **Lean UX:** Αυτή η μέθοδος επιδιώκει να μειώσει το χρόνο ανάπτυξης προϊόντων και να βελτιώσει τη συνεργασία εντός των ομάδων. Οι σχεδιαστές επικεντρώνονται στην εξέλιξη των ιδεών μέσα από τον πειραματισμό.

- **Agile UX:** Συνδυάζει τις αρχές της ανάπτυξης Agile με την εμπειρία του χρήστη. Η διαδικασία ανάπτυξης διαιρείται σε σύντομες επαναλήψεις που επιτρέπουν την ταχύτερη ανταπόκριση στα αιτήματα των χρηστών.

4.4.2 Ανάλυση Χρήσης:

- **A/B Testing:** Αυτή η τεχνική συγκρίνει δύο ή περισσότερες εκδοχές ενός στοιχείου, όπως μια ιστοσελίδα ή μια εφαρμογή, για να διαπιστωθεί ποια παρουσιάζει καλύτερη απόδοση ή αντίδραση από τους χρήστες.

- **Usage Analytics:** Η συγκέντρωση και επεξεργασία δεδομένων σχετικά με τη χρήση μιας εφαρμογής ή ιστοσελίδας. Αυτή η διαδικασία παρέχει εισαγωγές για τον τρόπο με τον οποίο οι χρήστες συνδιαλέγονται/Interact με το προϊόν και πού χρειάζεται να γίνουν βελτιώσεις.

Ο συνδυασμός αυτών των μεθοδολογιών και εργαλείων επιτρέπει στις ομάδες ανάπτυξης και σχεδιασμού να κατανοήσουν καλύτερα τις ανάγκες των χρηστών και να δημιουργήσουν προϊόντα που προσφέρουν μια πλούσια, ευχάριστη και αποτελεσματική εμπειρία.

5 ΤΑΣΕΙΣ ΚΑΙ ΚΑΙΝΟΤΟΜΙΕΣ ΣΤΟΝ ΣΧΕΔΙΑΣΜΟ ΙΣΤΟΣΕΛΙΔΩΝ ΓΙΑ ΚΙΝΗΤΕΣ ΣΥΣΚΕΥΕΣ

5.1 Σχεδιασμός για Κινητές Συσκευές

Μεγάλη σημασία στην ανάπτυξη ιστοσελίδων και e-shop, είναι και η παρουσία τους σε κινητές συσκευές, καθώς και τα κινητά πλέον μπορούν να παρέχουν υπηρεσίες όπως διαδικτυακές αγορές, εστιάζουμε στις αγορές λόγω του θέματος της εργασίας. Άρα ο προγραμματιστής πρέπει να φροντίσει έτσι ώστε το περιβάλλον της σελίδας να είναι φιλικό και σε κινητές συσκευές, ή όπως ονομάζετε **responsive web design**.

Ο όρος responsive web design αναφέρεται σε μια ιστοσελίδα που προσαρμόζεται πλήρως στις διαφορετικές συσκευές που χρησιμοποιούνται για την πρόσβαση, όπως υπολογιστές και κινητά τηλέφωνα. Αυτό σημαίνει ότι μια ιστοσελίδα με responsive design προσαρμόζει αυτόματα το πλάτος της, καθώς και τα στοιχεία της (όπως εικόνες, γραφήματα, κείμενο κ.λπ.), για να βελτιώσει την εμπειρία του χρήστη ανάλογα με τη συσκευή που χρησιμοποιεί.

Για να επιτευχθεί αυτό, υπάρχουν προγραμματιστικά εργαλεία τα οποία μας βοηθούν να το επιτύχουμε αυτό. Ένα πολύ γνωστό εργαλείο είναι το framework Bootstrap (το οποίο έχει αναφερθεί και σε προηγούμενο κεφάλαιο), το οποίο μας παρέχει μία “βιβλιοθήκη” με γραφήματα, πλαίσια και πολλά άλλα, τα οποία βοηθούν σε μεγάλο βαθμό στην δημιουργία responsive σελίδων, επίσης είναι δωρεάν.

Για να επιτύχει ένας προγραμματιστής την ορθή λειτουργία των πλαισίων του Bootstrap, απαιτείται προσωπική τεχνογνωσία σε διάφορα επίπεδα. Κατ' αρχήν, η κατανόηση της CSS3 είναι ζωτικής σημασίας, καθώς αναλαμβάνει την ευθύνη για το στυλ της ιστοσελίδας. Αυτό περιλαμβάνει την επιλογή χρωμάτων, την τοποθέτηση πλαισίων και κειμένων, καθώς και τη διαχείριση της διάταξης του περιεχομένου.

Επιπλέον, η γνώση της JavaScript είναι εξίσου σημαντική, καθώς πολλά στοιχεία του Bootstrap περιλαμβάνουν διάφορες λειτουργίες. Αυτές περιλαμβάνουν dropdown μενού, hover effects , ενέργειες με κλικ (onclick actions) και πολλά άλλα. Η ικανότητα να διαχειριστεί ο προγραμματιστής αυτές τις λειτουργίες μέσω JavaScript είναι απαραίτητη για την ομαλή λειτουργία και τη δημιουργία δυναμικών εφαρμογών.

5.1.1 Εξέλιξη του Responsive Design:

Το responsive design έχει εξελιχθεί σημαντικά από την εμφάνισή του. Στις αρχές, οι ιστοσελίδες συχνά είχαν έναν απλό σχεδιασμό για σταθερές οθόνες υπολογιστών και δεν ήταν προσαρμοσμένες για κινητές συσκευές. Με την αύξηση της χρήσης κινητών συσκευών, οι ιστοσελίδες άρχισαν να υιοθετούν ανταποκριτικούς σχεδιασμούς, χρησιμοποιώντας τεχνικές όπως οι media queries για να προσαρμόζονται σε διάφορες οθόνες.

Σήμερα, η τάση στον σχεδιασμό για κινητές συσκευές είναι περισσότερο συγκεκριμένη και εξελίσσεται συνεχώς. Αυτό περιλαμβάνει τη χρήση προοδευτικών web εφαρμογών (PWAs), οι οποίες προσφέρουν εμπειρίες όπως εφαρμογές ακόμα και μέσω περιηγητών. Επίσης, ο σχεδιασμός με βάση τον χρήστη (user-centered design) γίνεται ολοένα και πιο διαδεδομένος, εστιάζοντας στην ευχρηστία και την ευκολία πλοήγησης σε μικρές οθόνες.

5.1.2 Βελτιστοποίηση για SEO(Search Engine Optimization) σε Κινητές Συσκευές:

Η βελτιστοποίηση για μηχανές αναζήτησης σε κινητές συσκευές απαιτεί διαφορετική προσέγγιση σε σύγκριση με τον σχεδιασμό για σταθερές οθόνες. Κάποια στοιχεία που θα πρέπει να ληφθούν υπόψη περιλαμβάνουν:

- **Ταχύτητα φόρτωσης:** Ο χρόνος φόρτωσης είναι κρίσιμος για την SEO σε κινητές συσκευές. Οι ιστοσελίδες πρέπει να είναι γρήγορες και αποτελεσματικές για να διατηρήσουν την αναζητησιμότητά τους.
- **Προσαρμοσμένο περιεχόμενο:** Ο σχεδιασμός πρέπει να είναι φιλικός προς το χρήστη και να παρέχει περιεχόμενο που είναι εύκολο να διαβαστεί και να κατανοηθεί σε μικρές οθόνες.
- **Επικέντρωση στην τοπική SEO:** Η τοπική SEO είναι σημαντική για κινητές αναζητήσεις, καθώς πολλοί χρήστες κινητών αναζητούν πληροφορίες και υπηρεσίες κοντά τους.

5.2 Νέες Τάσεις Και Τεχνολογίες

Στον σχεδιασμό ιστοσελίδων για κινητές συσκευές, οι τελευταίες τάσεις επικεντρώνονται στη βελτίωση της εμπειρίας του χρήστη και την αύξηση της αποτελεσματικότητας των ιστοσελίδων. Δύο από τις σημαντικές τεχνολογίες που έχουν επικεντρωθεί σε αυτόν τον τομέα είναι η AMP (Accelerated Mobile Pages) και η τεχνολογία των PWA (Progressive Web Apps).

5.2.1 AMP(Aceelerated Mobile Pages - Προφορτωμένες Σελίδες για Κινητά)

Η AMP (Accelerated Mobile Pages), αν και αρχικά ήταν ακρώνυμο για "Προφορτωμένες Σελίδες για Κινητά", έχει εξελιχθεί σε αυτόνομο όρο. Πρόκειται για ένα ανοικτού κώδικα πρότυπο της Google, που έχει σχεδιαστεί αποκλειστικά για τη βελτίωση της ταχύτητας φόρτωσης ιστοσελίδων σε κινητές συσκευές και λειτουργεί ως αντίπαλος του προτύπου Instant Articles.

Η τεχνολογία AMP σκοπεύει στην άμεση φόρτωση των σελίδων, παρέχοντας μια ταχεία και ομαλή εμπειρία περιήγησης για τους χρήστες. Οι σελίδες αυτές σχεδιάζονται έτσι ώστε να φορτώνουν σχεδόν ακαριαία, κάνοντάς τες ιδανικές για συνθήκες όπου η σύνδεση στο διαδίκτυο είναι αργή ή ασύρματη.

Η τεχνολογία AMP περιορίζει τη χρήση HTML, CSS και JavaScript, ενώ δίνει την δυνατότητα για προσωρινή αποθήκευση του περιεχομένου σε κεντρικούς διακομιστές. Αυτή η προσέγγιση επιτρέπει την γρήγορη πρόσβαση στο περιεχόμενο, εξασφαλίζοντας ότι οι χρήστες μπορούν να περιηγηθούν αποτελεσματικά στις σελίδες ανεξαρτήτως της ταχύτητας της σύνδεσης ή της απόδοσης της συσκευής τους.

5.2.2 PWA (Progressive Web Apps)

Μια Progressive Web App (PWA) είναι μια εφαρμογή λογισμικού που διατίθεται μέσω του διαδικτύου και αξιοποιεί τεχνολογίες ιστού όπως HTML, CSS, JavaScript και WebAssembly. Η βασική της λειτουργία είναι να λειτουργεί σε οποιαδήποτε πλατφόρμα με έναν περιηγητή που συμμορφώνεται με τα πρότυπα.

Οι προοδευτικές web εφαρμογές δεν απαιτούν ξεχωριστό συσκευασμό ή διανομή. Οι προγραμματιστές μπορούν να δημοσιεύουν τις εφαρμογές τους online και να διασφαλίσουν ότι πληρούν τις βασικές απαιτήσεις εγκατάστασης. Η δημοσίευση σε ψηφιακά καταστήματα όπως το Apple App Store ή το Google Play είναι προαιρετική.

Από το 2021, οι PWA είναι διαθέσιμοι σε περιηγητές όπως ο Google Chrome, ο Apple Safari, ο Firefox για Android(όχι τον περιηγητή του στο desktop) και ο Microsoft Edge.

Οι τεχνικές προϋποθέσεις για μια ιστοσελίδα να θεωρείται PWA περιλαμβάνουν την παροχή μέσω HTTPS για ιδιωτικότητα και ασφάλεια, τη χρήση service workers για προγραμματιζόμενα caches περιεχομένου και την αναφορά σε ένα web app manifest που περιέχει βασικές ιδιότητες όπως το όνομα, το URL έναρξης και το εικονίδιο της εφαρμογής.

Τα τεχνικά κριτήρια βάσης για μια ιστοσελίδα να θεωρείται προοδευτική web εφαρμογή και επομένως να είναι ικανή να εγκατασταθεί από τους περιηγητές περιγράφηκαν από τον Russell σε μια ακόλουθη ανάρτηση και έχουν ενημερωθεί από τότε:

- Προέλευση από ασφαλές προέλευση. Να παρέχεται μέσω TLS και να μην έχει ενεργό ανάμεικτο περιεχόμενο. Οι προοδευτικές web εφαρμογές πρέπει να παρέχονται μέσω HTTPS για να εξασφαλιστεί η ιδιωτικότητα, η ασφάλεια του χρήστη και η αυθεντικότητα του περιεχομένου.
- Εγγραφή ενός service worker με έναν fetch handler. Οι προοδευτικές web εφαρμογές πρέπει να χρησιμοποιούν service workers για τη δημιουργία προγραμματιζόμενων caches περιεχομένου. Αυτή η λειτουργία επιτρέπει στις σελίδες να είναι προσβάσιμες ακόμα και όταν δεν υπάρχει σύνδεση ή όταν η ποιότητα του δικτύου είναι χαμηλή.
- Αναφορά σε ένα web app manifest. Το μανιφέστο πρέπει να περιέχει τουλάχιστον τις πέντε βασικές ιδιότητες: name ή short_name, start_url, και display (με τιμή standalone, fullscreen ή minimal-ui), και εικονίδια (με εκδόσεις 192px και 512px). Οι πληροφορίες που περιέχονται στο μανιφέστο καθιστούν τις προοδευτικές web εφαρμογές εύκολα κοινοποιήσιμες μέσω ενός URL, εντοπίσιμες από μηχανές αναζήτησης, και ανακουφίζουν από περίπλοκες διαδικασίες εγκατάστασης.

Συμπεράσματα

Με βάση τα κεφάλαια που αναπτύχθηκαν, προκύπτει ότι η δημιουργία μιας ιστοσελίδας, ειδικότερα ενός e-shop στην προκειμένη περίπτωση, απαιτεί πέραν της τεχνογνωσίας στο προγραμματιστικό κομμάτι για τις βασικές ρυθμίσεις και λειτουργίες του κώδικα, την ανάπτυξη στιλιστικών τεχνικών για την αισθητική παρουσίαση της σελίδας, τη δημιουργία σεναρίων ασφαλείας δεδομένων και προσωπικών πληροφοριών, καθώς και την προσαρμογή της εφαρμογής για κινητές συσκευές. Τα εν λόγω κεφάλαια επικεντρώνονται σε θεμελιώδη θέματα του σύγχρονου web development.

Προσωπικώς, η πρακτική εφαρμογή της εργασίας με βοήθησε να κατανοήσω τον τρόπο αλληλεπίδρασης των τεχνολογιών και των προγραμμάτων μεταξύ τους, προκειμένου να επικοινωνούν αποτελεσματικά και να παρουσιάζουν στον χρήστη το αναμενόμενο περιεχόμενο σε ένα e-shop. Κατά τη διάρκεια της δημιουργίας της εργασίας, απέκτησα εξοικείωση με τα εργαλεία που επέλεξα να χρησιμοποιήσω, κατάλαβα τη λογική πίσω από το web development και την εφάρμοσα στον κώδικα, σχεδίασα σωστά την απεικόνιση και την παρουσίαση των σελίδων, και κυρίως επιδίωξα τη διεύρυνση των γνώσεων μου στον τομέα. Σαφώς και το γραπτό κομμάτι της εργασίας με βοήθησε στο να κατανοήσω σε προφορικό κομμάτι όλες τις τεχνολογίες και τεχνικές που χρησιμοποιήθηκαν αλλά και να μπορέσω να τις παρουσιάσω με όσο περισσότερη ανάλυση σε κοινό είτε που γνωρίζει το αντικείμενο είτε όχι.

Όσον αφορά τα μελλοντικά σχέδια της πτυχιακής μου εργασίας, θα προσπαθήσω να πειραματιστώ και με περισσότερες τεχνολογίες, είτε με διαφορετικά στιλιστικά εργαλεία, είτε με πιο εξελιγμένες τεχνολογίες και frameworks (React.js, Vue.js , Tailwind CSS και άλλα), αλλά σίγουρα θα την χρησιμοποιήσω με σκοπούς εύρεσης εργασίας πάνω στον τομέα του web development.

Βιβλιογραφία

[1] Wikipedia. HTML. Ανακτήθηκε από https://el.wikipedia.org/wiki/HTML
[2] Wikipedia. CSS. Ανακτήθηκε από https://el.wikipedia.org/wiki/CSS
[3] Bootstrap. The most popular HTML, CSS, and JS library in the world. Ανακτήθηκε από https://getbootstrap.com/
[4] Wikipedia. JavaScript. Ανακτήθηκε από https://el.wikipedia.org/wiki/JavaScript
[5] Mozilla Developer Network. Introduction to the DOM. Ανακτήθηκε από https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction
[6] JSON. Introducing JSON. Ανακτήθηκε από https://www.json.org/json-en.html
[7] FreeCodeCamp. AJAX tutorial: Beginner's guide to learn AJAX programming with examples. Ανακτήθηκε από https://www.freecodecamp.org/news/ajax-tutorial/

[8] jQuery Foundation. jQuery API Documentation. Ανακτήθηκε από https://api.jquery.com/
[9] FreeCodeCamp. Asynchronism in JavaScript. Ανακτήθηκε από https://www.freecodecamp.org/news/asynchronism-in-javascript/
[10] Wikipedia. Node.js. Ανακτήθηκε από https://en.wikipedia.org/wiki/Node.js
[11] Wikipedia. Express.js. Ανακτήθηκε από https://en.wikipedia.org/wiki/Express.js & https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs
[12] Wikipedia. XAMPP. Ανακτήθηκε από https://el.wikipedia.org/wiki/XAMPP
[13] Big Blue Data Academy. (2023). Τι Είναι το API και Πώς Λειτουργεί; Ανακτήθηκε από https://bigblue.academy/gr/ti-einai-to-api
[14] iServices. (2022). Δυναμικές Ιστοσελίδες vs Στατικές Ιστοσελίδες: Ποια η Διαφορά. Ανακτήθηκε από https://iservices.gr/blog/statikes-dinamikes-istoselides/
[15] Wikipedia. HTTPS. Ανακτήθηκε από https://el.wikipedia.org/wiki/HTTPS
[16] Wikipedia. UI/UX. Ανακτήθηκε από https://en.wikipedia.org/wiki/User_experience_design

ΠΑΡΑΡΤΗΜΑ ΚΩΔΙΚΑ ΚΑΙ ΑΝΑΛΥΣΗ ΤΟΥ

Στο συγκεκριμένο κεφάλαιο θα παρουσιαστεί το πρακτικό μέρος της εργασίας, με πιο αναλυτική επεξήγηση των τεχνολογιών που χρησιμοποιήθηκαν και του τρόπου με τον οποίο αλληλοεπιδρούν για να επιτευχθεί η παρουσίαση και η λειτουργικότητα της ιστοσελίδας.

Αρχικά, θα γίνει ανάλυση της επικοινωνίας μεταξύ του front-end (παρουσίαση) και του back-end (υποσύστημα), καθώς αυτή αποτελεί τον θεμέλιο λίθο για τη σωστή λειτουργία της ιστοσελίδας. Η επικοινωνία αυτή απαιτεί τη χρήση ενός διακομιστή (server), ο οποίος εξασφαλίζει την αλληλεπίδραση των δύο αυτών μερών. Στη συγκεκριμένη εργασία, χρησιμοποιήθηκε το Node.js, ένα περιβάλλον εκτέλεσης της JavaScript, καθώς και το Express.js, ένα framework του Node.js που διευκολύνει τη δημιουργία web εφαρμογών. Για περισσότερες πληροφορίες σχετικά με τις τεχνολογίες αυτές, μπορείτε να ανατρέξετε στα κεφάλαια 1.1.3 και 1.2.1.

Επιπλέον, θα γίνει απεικόνιση των δραστηριοτήτων των χρηστών της ιστοσελίδας μέσω διαγραμμάτων UML, προκειμένου να αναλυθούν οι ροές εργασιών και οι αλληλεπιδράσεις των χρηστών με το σύστημα.

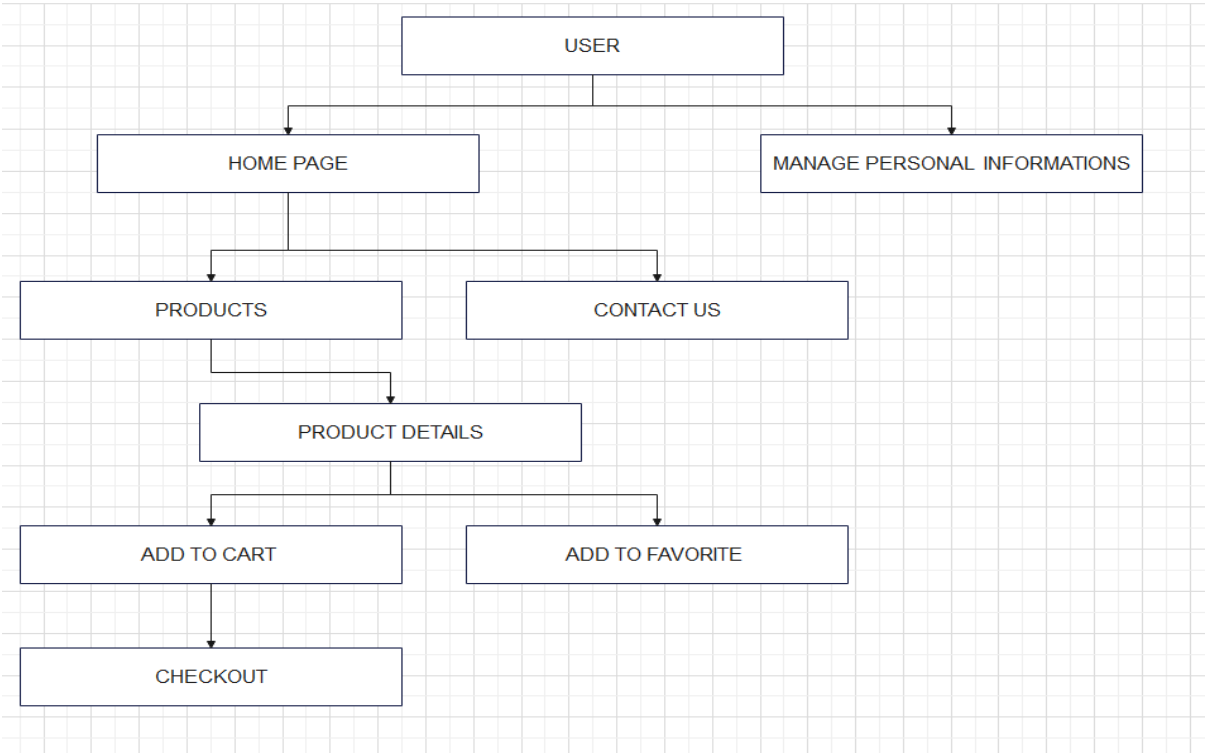
Σχεδιασμός της ιστοσελίδας

Στον προγραμματισμό, ένας τρόπος αναπαράστασης των λειτουργιών των προγραμμάτων μπορεί να γίνει και μέσω των UML διαγραμμάτων. Στην συγκεκριμένη εργασία λόγω της ύπαρξης χρηστών και του διαχειριστή μπορεί να γίνει ανάλυση των ενεργειών μέσω των διαγραμμάτων. Υπάρχουν 2 είδη, το case και το class diagram.

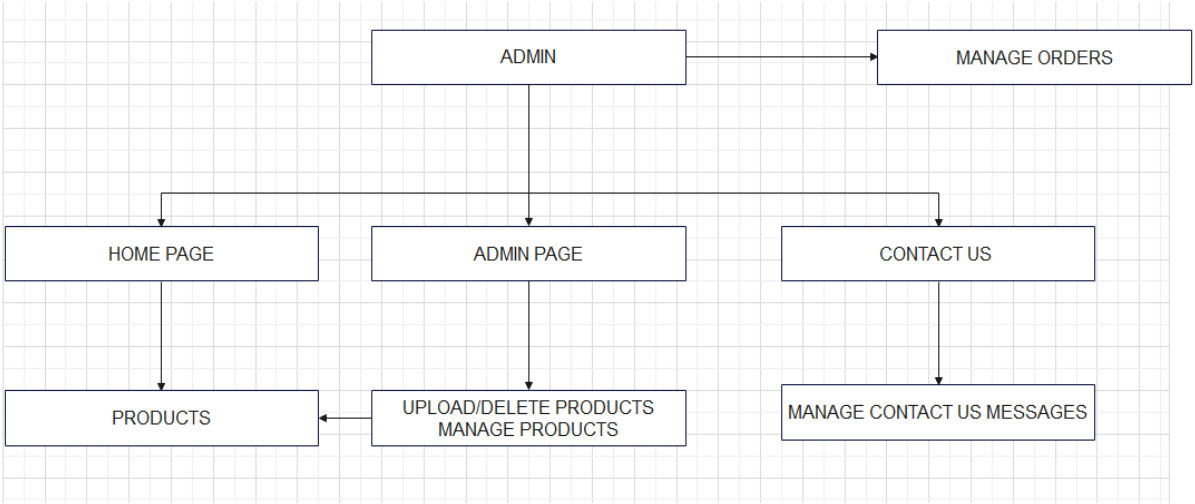
Τα Use Case διαγράμματα χρησιμοποιούνται για να αναπαραστήσουν τις αλληλεπιδράσεις των χρηστών μεταξύ του συστήματος. Κάθε "use case" περιγράφει μια συγκεκριμένη λειτουργία ή μια ενέργεια που μπορεί να πραγματοποιηθεί στο σύστημα. Αποτυπώνονται οι ενέργειες και οι συσχετίσεις των ενεργειών των χρηστών και του διαχειριστή .

Τα Class διαγράμματα χρησιμοποιούνται για να αναπαραστήσουν τη δομή του συστήματος σε επίπεδο κλάσεων και τις σχέσεις μεταξύ τους. Οι κλάσεις αντιπροσωπεύουν αντικείμενα με συγκεκριμένα χαρακτηριστικά και μεθόδους.

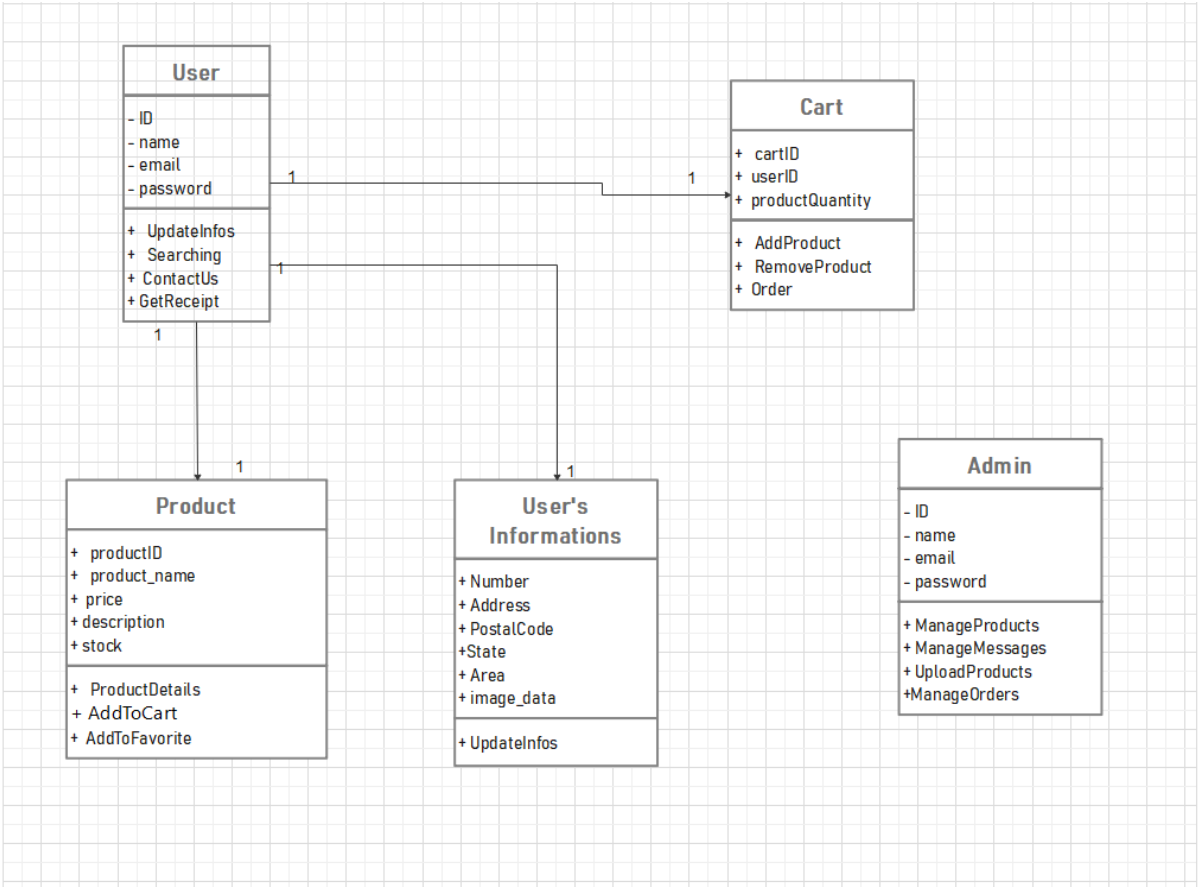
Ωστόσο και τα 2 διαγράμματα είναι εξίσου σημαντικά διότι μπορούν να προσφέρουν μια ολοκληρωμένη εικόνα για τον σχεδιασμό και την ανάπτυξη ενός συστήματος με χρήστες και διαχειριστές. Τα use case είναι χρήσιμα για την κατανόηση των λειτουργιών που πρέπει να παρέχει το σύστημα και τις αλληλεπιδράσεις με τους χρήστες, ενώ τα Class διαγράμματα είναι κρίσιμα για την κατανόηση της δομής του συστήματος και των σχέσεων μεταξύ των διαφορετικών κλάσεων



Διάγραμμα case του user



Διάγραμμα case του admin



Διάγραμμα Class

Το πρόγραμμα το οποίο χρησιμοποιήθηκε για την δημιουργία των προγραμμάτων είναι το Wondershare EdrawMax το οποίο παρέχει δωρεάν μερικές από τις λειτουργίες του. (https://www.edrawmax.com/)

Ανάπτυξη της ιστοσελίδας

Αρχικά ένα από τα πιο σημαντικά σημεία σε έναν κώδικα, στην δημιουργία μίας εφαρμογής είναι να δηλωθούν οι βιβλιοθήκες που θα χρησιμοποιηθούν για την ανάπτυξη της. Τα εικονιζόμενα είναι κάποιες βασικές βιβλιοθήκες οι οποίες θα αναλυθούν.

Ξεκινώντας με το πιο βασικό το οποίο είναι και ένα από τα πιο διαδεδομένα και χρήσιμα εργαλεία στο περιβάλλον του node.js είναι το framework του, express.js

```
const express = require('express');
const cookieParser = require('cookie-parser');
const session = require('express-session');
const bodyParser = require('body-parser');
const mysql = require('mysql2');
const fs = require('fs');
const path = require('path');
const multer = require('multer');
const paypal = require('paypal-rest-sdk');
const nodemailer = require('nodemailer');
```

Const express = require('express')

Με αυτή την εντολή ενσωματώνουμαι το module του express μέσα στον server μας το οποίο μας διευκολύνει στην δημιουργία σερβερικού κώδικα(server-side code) και να καλούμε εφαρμογές APIs, να δημιουργούμε routes και middleware sessions. Τέλος δηλώνουμε το express στον κώδικα μας μέσω του **const app = express()**;

Const session = require (express-session')

Όπως αναφέρθηκε και παραπάνω το express μας βοηθάει να δημιουργούμε και sessions. Η χρήση τους μας διευκολύνει στο να διαχειρίζομαστε τους χρήστες της σελίδας μας ως μοναδικούς βάση του ID τους και τα αιτήματα τους μέσω τους server μας. Η εντολή αυτή λοιπόν μας δίνει την δυνατότητα να δημιουργούμε τα sessions μέσω του express. Με την εντολή bodyParser.urlencode({ extended : true}); το πρόγραμμα μπορεί να εκτελέσει αιτήματα μέσω κωδικοποιημένων URL (πχ αιτήματα μέσω του τοπικού url localhost:4000/) και με την παράμετρο true επιτρέπουμε την ανάλυση σύνθετων δεδομένων μέσω της βιβλιοθήκης "querystring"(qs) όπως πίνακες(arrays), αντικείμενα(Objects). Τέλος η εντολή **app.use(bodyParser.json)** επιτρέπει την ανάλυση αρχείων json.

Const cookieParser= require(cookie-parser)

Το cookie-parser είναι ένα middleware για την Express, το οποίο αναλύει (parse) τα cookies που συνοδεύουν τα εισερχόμενα αιτήματα HTTP. Ενεργοποιείται με την εντολή **app.use(cookieParser());** Αυτό το middleware επιτρέπει στην εφαρμογή να διαβάζει τα cookies από τα αιτήματα του χρήστη, καθιστώντας τα διαθέσιμα μέσω του req.cookies.

```
app.use(session({
    secret: 'thisismysecretkey',
    saveUninitialized: true,
    cookie: { maxAge: 1000 * 60 * 60 * 24 },
    resave: false,
}));
```

Παραπάνω απεικονίζεται ενα κομμάτι κώδικα το οποίο σχετίζεται με το middleware cookie-parser, αναλυτικότερα γραμμή secret διασφαλίζει τα cookies έτσι ώστε να αποτρέπει την παραποίηση τους απο τρίτους, η γραμμή saveUnitialized καθορίζει αν αποθηκευτεί μία συνεδρία στο session store και ορίζοντας το ως true τις αποθηκεύει ακόμα και αν έχουν υποστεί τροποποίηση, με το maxAge ορίζεται ο χρόνος “ζωής” των cookies συγκεκριμένα στον κώδικα αυτό είναι για μία μερα και το resave ορίζει αν χρειάζεται να γίνει ξανά η αποθήκευσει μιας συνεδρείας και με την παράμετρο false αποτρέπεται αυτό εκτός αν έχει γίνει κάποια τροποποίηση.

Const bodyParser= require(body-parser):

Το body-parser είναι ένα middleware για το Express το οποίο αναλύει τα σώματα των αιτημάτων(πχ αιτήματα μέσω forms η αιτήματα αντικειμένων JSON) μέσω του req.body(request body).

Const mysql = require(‘mysql2’):

Ένας τρόπος για την αποθήκευση και την διαχείριση δεδομένα είναι η δημιουργία βάσεων δεδομένων, ωστόσο στο περιβάλλον του node.js η εντολή αυτή μας δίνει την δυνατότητα να αλληλεπιδρούν τα προγράμματα μας με τις MySql βάσεις δεδομένων. Μέσω των routes μπορούμε να διαβάσουμε(get), να διαβάσουμε(post) και να διαγράψουμε(delete) δεδομένα δίχως να εμπλακούμαι στο περιβάλλον της MySql αλλά απο τις σελίδες μας.

```
const db = mysql.createConnection({
    host: 'localhost',
    user: 'root',
```

```
    password: '',
    database: 'myform',
});
const dbProducts = mysql.createConnection({
    host: 'localhost',
    user: 'root',
    password: '',
    database: 'test',
  });


db.connect((err) => {
    if (err) {
        console.error('Error connecting to MySQL:', err);
        return;
    }
    console.log('Connected to MySQL');
});


dbProducts.connect((err) => {
    if (err) {
        console.error('Error connecting to the database:', err);
        throw err;
    }
    console.log('Connected to the database');
});
```

Const path = require('path'):

Είναι ένα βασικό node.js module , το οποίο επιτρέπει τη διαχείριση και τη διασύνδεση διαφόρων αρχείων και φακέλων

```
app.get('/admin', admin, (req, res) => {
    res.sendFile(path.join(__dirname, 'admin.html'));
});
```

Στο παραπάνω παράδειγμα μέσω του route get('admin') ζητάμε να στείλει στον client το αρχείο με το όνομα admin.html, με την χρήση του path ουσιαστικά ζητάμε να εισέλθει(join) στο __dirname(μεταβλητή για να εντοπίζει τα τοπικά αρχεία και φακέλους) και να εντοπίσει το αρχείο που ζητείται

Const fs = require('fs):

Το fs είναι μια ενσωματωμένη βιβλιοθήκη του Node.js που επιτρέπει την αλληλεπίδραση με το σύστημα αρχείων. Παρέχει μια πληθώρα μεθόδων για τη δημιουργία, την ανάγνωση, την ενημέρωση και τη διαγραφή αρχείων, καθώς και για τη διαχείριση καταλόγων. Μέσω του fs, οι προγραμματιστές μπορούν να διαχειρίζονται αρχεία ασύγχρονα, εξασφαλίζοντας υψηλή απόδοση και αποφυγή μπλοκαρίσματος του κύριου νήματος εκτέλεσης. Η χρήση του fs είναι κρίσιμη για την ανάπτυξη διαφόρων εφαρμογών που απαιτούν διαχείριση δεδομένων σε τοπικό ή απομακρυσμένο σύστημα αρχείων.

```
app.post('/save_receipt', (req, res) => {
    const { receiptContent, orderId } = req.body;
    const receiptFilePath = path.join(__dirname, 'receipts',
`ORDER_${orderId}.xhtml`);

    fs.writeFile(receiptFilePath, receiptContent, (err) => {
        if (err) {
            console.error('Error saving receipt:', err);
            return res.status(500).send('Failed to save receipt');
        }
        res.status(200).send('Receipt saved successfully');
    });
})

app.get('/display_receipt', (req, res) => {
    res.sendFile(path.join(__dirname, 'public', 'receipts.html'));
});
```

Const multer = require('multer'):

Ένα επίσης σημαντικό και χρήσιμο εργαλείο το οποίο επιτρέπει να “ανεβάζουμε” αρχεία μέσω του server και να αποθηκεύονται δεδομένα τοπικά αλλά και σε βάσεις δεδομένων.

```
const storage = multer.memoryStorage();
const upload = multer({ storage });

app.post('/upload', upload.single('image'), (req, res) => {
    if (!req.file) {
      return res.status(400).json({ error: 'No file uploaded.' });
    }


    if (!req.session.user || !req.session.user.id) {
```

```
      return res.status(401).json({ error: 'Unauthorized. Please log in.'
});
    }

    const userId = req.session.user.id;
    const imageBuffer = req.file.buffer;


    const query = 'UPDATE users SET image_data = ? WHERE id = ?';


    db.query(query, [imageBuffer, userId], (err, results) => {
      if (err) {
        console.error('Error updating image:', err);
        return res.status(500).json({ error: 'Internal Server Error',
details: err.message });
      } else {
        console.log('Image updated in the users table.');
        res.status(200).json({ message: 'Image updated in the users
table.' });
      }
    });
  });
```

Μέσω του παραδείγματος, με την χρήση του multer μπορούμε να δημιουργήσουμε μια μηχανή η οποία αποθηκεύει τα δεδομένα στην μνήμη και όχι τοπικά, το οποίο επιτρέπει την επεξεργασία των αρχείων άμμεσα δίχως να αποθηκευτούν στον server και μέσω του route app.post('/upload') "ανεβαίνει" ένα αρχείο image στην βάση δεδομένων βάση του id του χρήστη.

```
<form action="/uploadProduct" method="POST" enctype="multipart/form-data">
```

Το multer επίσης χρησιμοποιείται και στα hmtl αρχεία με το enctype="multipart/form-data" με το οποίο δίνει την δυνατότητα στον χρήστη να "ανεβάζει" και πληροφορίες μέσω τα form

Const nodemailer = require('nodemailer');

Για την υλοποίηση της αποστολής email στην πτυχιακή μου εργασία, επέλεξα τη χρήση της βιβλιοθήκης nodemailer. Το nodemailer είναι ένα δημοφιλές εργαλείο για τον χειρισμό και την αποστολή ηλεκτρονικών μηνυμάτων μέσω Node.js. Η επιλογή του έγινε λόγω της ευκολίας στη χρήση του, της ευρείας υποστήριξης πρωτοκόλλων όπως το SMTP και της δυνατότητας ενσωμάτωσής του σε διάφορες εφαρμογές web. Μέσω του nodemailer, κατάφερα να δημιουργήσω λειτουργίες αποστολής email για ειδοποιήσεις χρηστών,

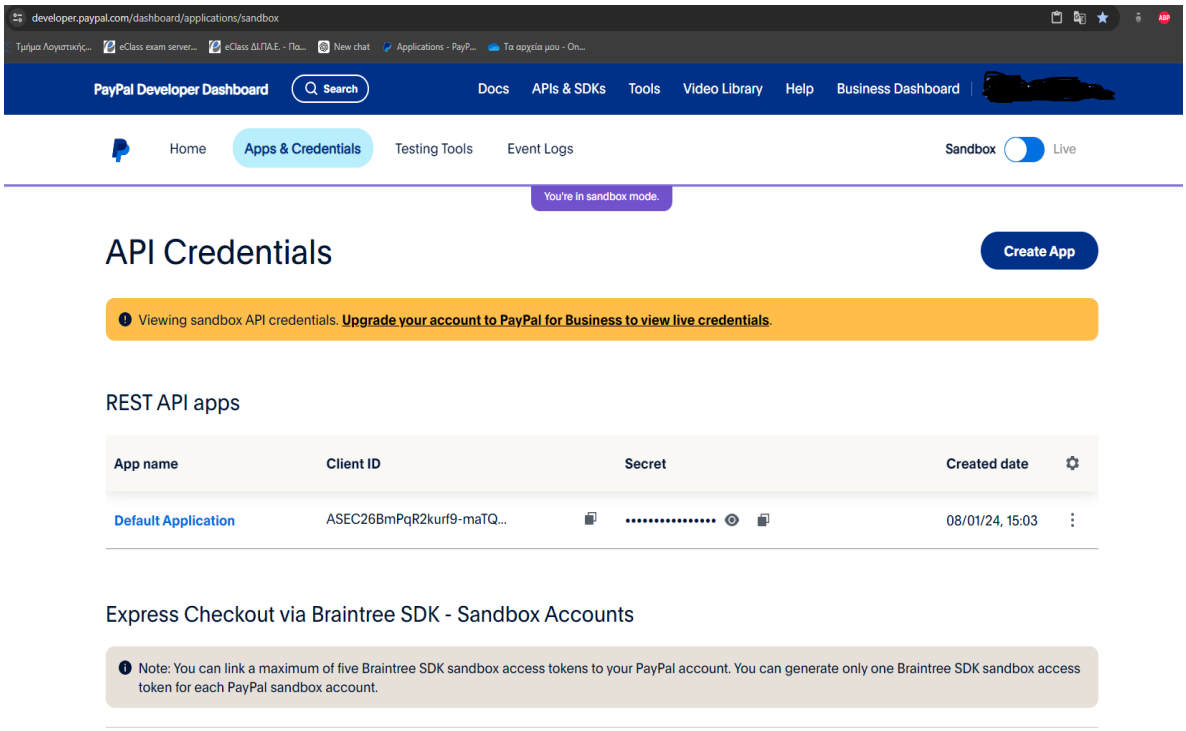
επιβεβαιώσεις εγγραφών και άλλες αυτοματοποιημένες επικοινωνίες, προσφέροντας έτσι μια ολοκληρωμένη και επαγγελματική εμπειρία χρήστη.

```
const transporter = nodemailer.createTransport({
    host: "sandbox.smtp.mailtrap.io",
    port: 2525,
    auth: {
      user: "3ca0fcd76313ba",
      pass: "4a0d5055fc2ef3"
    }
  });

  function localEmail(userData, receiptContent) {
    const mailOptions = {
        from: 'musika-eshop@gmail.com',
        to: userData.email,
        subject: 'Your Order Receipt',
        html: receiptContent
    };

    transporter.sendMail(mailOptions, function(error, info) {
        if (error) {
            console.error('Error sending email:', error);
        } else {
            console.log('Email sent:', info.response);
        }
    });
}

app.post('/send-receipt', (req, res) => {
    const { userData, receiptContent } = req.body;

    localEmail(userData, receiptContent);

    res.sendStatus(200);
});
```

Λόγω του ότι η εργασία “τρέχει” σε τοπικό περιβάλλον, χρησιμοποιήθηκε η βιβλιοθήκη nodemailer για να δημιουργηθεί η επικοινωνία του site με το δηλωμένο email του χρήστη. Το nodemailer είναι ένα ισχυρό εργαλείο για την αποστολή ηλεκτρονικών μηνυμάτων μέσω Node.js, το οποίο επιτρέπει την εύκολη και γρήγορη διαμόρφωση SMTP επικοινωνίας. Επιπλέον, για την ανάπτυξη και τις δοκιμές, έγινε χρήση της πλατφόρμας Mailtrap (https://mailtrap.io/), η οποία προσφέρει δωρεάν υπηρεσίες δημιουργίας δοκιμαστικών SMTP διακομιστών. Η Mailtrap επιτρέπει την ασφαλή δοκιμή και επιβεβαίωση των email χωρίς να απαιτεί την αποστολή πραγματικών μηνυμάτων σε πραγματικούς χρήστες, διασφαλίζοντας έτσι την ορθότητα και λειτουργικότητα του συστήματος επικοινωνίας.

Const paypal = require('paypal-rest-dk'):

Σε μια ιστοσελίδα e-shop, για την διαχείριση των πληρωμών μια έμπιστη και έγκυρη διαδικασία, είναι μέσω της paypal που με την εντολή αυτή και τον κατάλληλο προγραμματισμό της ο χρήστης μπορεί να πραγματοποιεί αγορές. Μέσω της επίσημης σελίδας της Paypal δίνεται η δυνατότητα δημιουργίας ενός test acoount και test store. Έπειτα πρέπει να δηλωθούν τα στοιχεία του acoount αυτού στον server μας και να προγραμματιστεί έτσι ώστε να πραγματοποιούνται επιτυχημένα οι αγορές.



Περιβάλλον δημιουργίας test λογαριασμού paypal (https://developer.paypal.com)

```
paypal.configure({
    mode: 'sandbox',
    client_id: 'ASEC26BmPqR2kurf9-maTQLknJzy9EF4m-Sxo-
A4lAx6x9ZZ8qyP_gwQynaqm3-cVT-EO0t4fImMLksk',
    client_secret: 'ECb997KQno5oWneatav32ZqByD9-4_sRRNgtFQH0-
vkLBnr4oPmNCoCQWs_Ydgdf4qhBziNVoVKvPrTk',
  });
```

```
<script src="https://www.paypal.com/sdk/js?client-id=ASEC26BmPqR2kurf9-
maTQLknJzy9EF4m-Sxo-A4lAx6x9ZZ8qyP_gwQynaqm3-cVT-EO0t4fImMLksk"></script>
```

Το sandbox είναι το test περιβάλλον της paypal για δοκιμαστικές αγορές, υπάρχει και το live αλλά η συγκεκριμένη εργασία εκτελείται τοπικά. Ωστόσο πολύ βασικό είναι να δηλωθεί σε τι περιβάλλον τρέχει η εφαρμογή και το client_id και client_secret. Πολύ βασικό επίσης είναι να δηλωθεί και το client_id στο client side του κώδικα μας , πιο συγκεκριμένα στην html.

Η κλήση της εφαρμογής της paypal γίνεται μέσω του απώτερου κώδικα που ουσιαστικά έχει προγραμματιστεί να "διαβάζει" το id, τα ονόματα των προϊόντων , την τιμή τους, την ποσότητα τους και το currency του νομίσματος. Εφόσον έχει ελέγξει τις πληροφορίες των προϊόντων προχωράει στην τελική τιμή totalPrice και μέσω του paypal.payment.create δημιουργεί το transaction.
Μέσω αυτού του link (https://codeculturepro.medium.com/simple-5-steps-for-your-paypal-payment-integration-in-node-js-400a9dbc8c4d) αποτυπώνονται αναλυτικά τα βήματα προγραμματισμού της εφαρμογής.

```
app.get('/paypal', authenticateUser, async (req, res) => {
    const userId = req.session.user ? req.session.user.id : null;
    const userCart = await getUserCart(userId);
    const totalPrice = TotalPrice(userCart);

    const createPaymentJson = {
        intent: 'sale',
        payer: {
            payment_method: 'paypal',
        },
        redirect_urls: {
            return_url: 'http://localhost:4000/success',
            cancel_url: 'http://localhost:4000/cancel',
        },
        transactions: [{
            item_list: {
                items: userCart.map(item => ({
                    name: item.product_name,
                    sku: item.productId,
                    price: (item.price || 0).toFixed(2),
                    currency: 'USD',
                    quantity: item.quantity,
                })),
            },
            amount: {
                currency: 'USD',
                total: totalPrice.toFixed(2),
            },
            description: 'Your purchase from My Store',
        }],
    };
```

```
    paypal.payment.create(createPaymentJson, (error, payment) => {
        if (error) {
            console.error('Error creating PayPal payment:', error);
            res.status(500).json({ error: 'Internal Server Error' });
        } else {
            const approvalUrl = payment.links.find(link => link.rel ===
'approval_url').href;
            res.redirect(approvalUrl);
        }
    });
});

app.route('/success')
    .get(async (req, res) => {
        try {
            const userId = req.session.user ? req.session.user.id : null;
            if (!userId) {
                return res.status(401).json({ error: 'Unauthorized' });
            }

            const userCart = await getUserCart(userId);
            const totalPrice = TotalPrice(userCart).toFixed(2);

        } catch (error) {
            console.error('Error in /success GET route:', error);
            res.status(500).json({ error: 'Internal Server Error' });
        }
    })
    .post(async (req, res) => {
        try {
            const userId = req.session.user ? req.session.user.id : null;
            if (!userId) {
                return res.status(401).json({ error: 'Unauthorized' });
            }

            const userCart = await getUserCart(userId);

            for (const item of userCart) {
                await productStock(item.productId, item.quantity);
            }

            await updateUserCart(userId, []);


        } catch (error) {
            console.error('Error in /success POST route:', error);
            res.status(500).json({ error: 'Internal Server Error' });
        }
    });
```

Στην συνέχεια, προγραμματίστηκαν οι λειτουργείες οι οποίες είναι απαραίτητες έπειτα επιτυχημένων αγορών(πχ να αδειάζει το καλάθι του χρήστη έπειτα από την αγορά και να μειώνεται το stock των προιόντων)

```
paypal.Buttons({
  createOrder: function(data, actions) {
        return actions.order.create({
            purchase_units: [{
                amount: {
                    currency_code: 'USD',
                    value: totalPriceValue.grandTotal.toFixed(2)
                }
            }]
        });
    },
    onApprove: function(data, actions) {
        return actions.order.capture().then(async function(order) {
            const userData =
JSON.parse(localStorage.getItem('checkoutData'));
            const productDetailsList = await
Promise.all(cartItems.map(item => fetchProductDetails(item.productId)));

            let productsHTML = productDetailsList.map((product, index) =>
`
                <p><strong>Product Name:</strong>
${product.product_name}</p>
                <p><strong>Product Price:</strong> ${product.price}</p>
                <p><strong>Quantity:</strong>
${cartItems[index].quantity}</p>
            `).join('');

            const total = await totalPrice(cartItems,
selectedCourierService);

            const orderId = Math.random().toString(36).substr(2,
9).toUpperCase();

            const currentDateTime = new Date();
            const formattedDate = currentDateTime.toLocaleDateString('en-
US', { year: 'numeric', month: 'long', day: 'numeric' });
            const formattedTime = currentDateTime.toLocaleTimeString('en-
US', { hour: '2-digit', minute: '2-digit', second: '2-digit' });

            fetch('/receipt.html').then(response =>
response.text()).then(template => {
                let receiptContent = template
                    .replace('{{firstName}}', userData.firstName)
                    .replace('{{lastName}}', userData.lastName)
```

```
                    .replace('{{address}}', userData.address)
                    .replace('{{email}}', userData.email)
                    .replace('{{zipCode}}', userData.zipCode)
                    .replace('{{orderId}}', orderId)
                    .replace('{{productsHTML}}', productsHTML)
                    .replace('{{courierFee}}',
total.courierFee.toFixed(2))
                    .replace('{{grandTotal}}',
total.grandTotal.toFixed(2))
                    .replace('{{formattedDate}}', formattedDate)
                    .replace('{{formattedTime}}', formattedTime);

                fetch('/save_receipt', {
                    method: 'POST',
                    headers: {
                        'Content-Type': 'application/json',
                    },
                    body: JSON.stringify({
                        receiptContent: receiptContent,
                        orderId: orderId
                    }),
                }).then(function(response) {
                    if (response.ok) {
                        console.log('Receipt saved successfully');
                    } else {
                        console.error('Failed to save receipt');
                    }
                }).catch(function(error) {
                    console.error('Error saving receipt:', error);
                });

                fetch('/send-receipt', {
                    method: 'POST',
                    headers: {
                        'Content-Type': 'application/json',
                    },
                    body: JSON.stringify({
                        userData: userData,
                        receiptContent: receiptContent
                    }),
                }).then(function(response) {
                    if (response.ok) {
                        console.log('Receipt email sent successfully');
                    } else {
                        console.error('Failed to send receipt email');
                    }
                }).catch(function(error) {
                    console.error('Error sending receipt email:', error);
                });
```

```
                fetch('/success', {
                    method: 'POST',
                    headers: {
                        'Content-Type': 'application/json',
                    },
                    body: JSON.stringify({...order, orderId}),
                })
                .then(function(response) {
                    if (response.ok) {
                        console.log('Cart cleared successfully');
                    } else {
                        console.error('Failed to clear cart');
                    }
                })
                .catch(function(error) {
                    console.error('Error clearing cart:', error);
                });
            });
            window.location.reload();
            localStorage.setItem('purchaseSuccess', 'true');
            });
    }
}).render('#paypal-button-container');
```

Τέλος χρειάζεται προγραμματισμός και στο client side του προγράμματος, συγκεκριμένα ο κώδικας επικοινωνεί με το περιβάλλον της paypal μέσω του server και ενημερώνει τον user για την επιτυχημένη αγορά του. Ωστόσο υπάρχουν και extra ρυθμίσει όπως να δημιουργεί μια σελίδα σαν απόδειξη. Επίσης δημιουργούνται και αυτόματα τα κουμπιά που εκτελούν τις λειτουργίες της εφαρμογής μέσω της εντολής .render('#paypal-button-container');.

Όσα αναφέρθηκαν περί των βιβλιοθηκών της εργασία, το Node.js δημιουργεί αυτόματα 2 αρχεία μορφής JSON, το package & package-lock. Το package είναι αρχείο διαμόρφωσης της εργασίας το οποίο περιέχει της πληροφορίες της όπως το όνομα του έργου, τα scripts που μπορούν και τις βιβλιοθήκες που χρησιμοποιούνται στο Node.js αρχείο, ενω το package-lock περιέχει πληροφορίες για τις ακριβείς εκδόσεις όλων των πακέτων των βιβλιοθηκών(*ονομάζονται dependecies) διασφαλίζοντας ότι έργο μπορεί να εκτελεί τις λειτουργίες όπως έχουν δηλωθεί στο package αρχείο. Ένα παράδειγμα, όταν χρησιμοποιούμαι την εντολή npm install για να προσθέσουμε κάποιο dependecy στον κώδικα, αυτόματα εγγράφεται και στα 2 αρχεία έτσι ώστε να λειτουργεί σωστά μέσα στο πρόγραμμα.

```
{
  "name": "ptixiaki",
  "version": "1.0.0",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node server.js"
```

```
  },
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "@types/express": "^4.17.17"
  },
  "dependencies": {
    "cookie-parser": "^1.4.6",
    "ejs": "^3.1.9",
    "express": "^4.18.2",
    "express-session": "^1.17.3",
    "jimp": "^0.22.10",
    "multer": "^1.4.5-lts.1",
    "mysql2": "^3.6.1",
    "nodemailer": "^6.9.14",
    "passport": "^0.7.0",
    "passport-local": "^1.0.0",
    "paypal-rest-sdk": "^1.8.1",
    "stripe": "^14.11.0"
  },
  "keywords": [],
  "description": ""
}
```

Package.json της εργασίας

```
{
  "name": "ptixiaki",
  "version": "1.0.0",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node server.js"
  },
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "@types/express": "^4.17.17"
  },
  "dependencies": {
    "cookie-parser": "^1.4.6",
    "ejs": "^3.1.9",
    "express": "^4.18.2",
    "express-session": "^1.17.3",
    "jimp": "^0.22.10",
    "multer": "^1.4.5-lts.1",
    "mysql2": "^3.6.1",
    "nodemailer": "^6.9.14",
```

```
    "passport": "^0.7.0",
    "passport-local": "^1.0.0",
    "paypal-rest-sdk": "^1.8.1",
    "stripe": "^14.11.0"
  },
  "keywords": [],
  "description": ""
}
```

Μέρος από το Package-lock.json της εργασίας

Ένα ακόμα middleware το οποίο χρησιμοποιήθηκε για την διασφάλιση των χρηστών, ουσιαστικά τα κομμάτια κώδικα που απεικονίζονται ελέγχει αν υπάρχει το session.user(δηλαδή αν υπάρχει user συνδεδεμένος στην σελίδα). Αν υπάρχει με την μέθοδο next() το πρόγραμμα του δίνει το δικαίωμα πρόσβασης ως user. Στο 2ο κομμάτι κώδικα το πρόγραμμα ελέγχει τον seesion.user έτσι ώστε να επιστρέψει το ID του , αλλιώς το επιστρέφει ως κενό και δεν μπορεί να του δωθεί πρόσβαση.

```
const authenticateUser = (req, res, next) => {
    if (req.session.user) {
        next();
    } else {
        res.status(401).redirect('Please log in.');
    }
};
```

Όταν δημιουργείται μία σελίδα η οποία διαχειρίζεται χρήστες, πρέπει να υπάρχει και ο διαχειριστής της, ωστόσο δημιουργήθηκε μία λειτουργία για να αναγνωρίζει τον user με το ID = 1 ως admin και εν συνεχεία να του δίνει τα δικαιώματα που έχουν οριστεί για αυτόν τον χρήστη.

```
const admin = (req, res, next) => {
    const userId = req.session.user ? req.session.user.id : null;
    if (userId === 1) {
        next();
    } else {
        res.status(401).send('User is not the admin');
    }
};

app.get('/admin', admin, (req, res) => {
    res.sendFile(path.join(__dirname, 'admin.html'));
});
```

Με την εντολή app.listen ορίζουμε την θύρα την οποία “τρέχει” ο server στον local host

```
app.listen(4000, () => {
    console.log('Server running on port 4000');
});
```

Διαδικασίες εγκατάστασης

Ακολουθούν τα βήματα για την εγκατάσταση και εκτέλεση της εφαρμογής Node.js/Express σε τοπικό υπολογιστή, ώστε να λειτουργεί μέσω **localhost**. Για να εγκατασταθεί η εφαρμογή, απαιτείται η ύπαρξη ορισμένων προγραμμάτων στο σύστημα.

- **Node.js & npm, h** εφαρμογή βασίζεται στο Node.js, επομένως είναι απαραίτητη η εγκατάσταση του Node.js μαζί με το npm (Node Package Manager). Αυτά χρησιμοποιούνται για την εκτέλεση της εφαρμογής και τη διαχείριση των εξαρτήσεων της.

Σε περιβάλλον **Ubuntu/Debian**, η εγκατάσταση γίνεται με

- sudo apt update
- sudo apt install nodejs npm

Σε περιβάλλον **Windows**, επισκεφθείτε τον επίσημο ιστότοπο του Node.js και κατεβάστε την τελευταία έκδοση. Ακολουθήστε τον οδηγό εγκατάστασης που παρέχεται.

- **Git**: Εάν ο κώδικας της εφαρμογής φιλοξενείται σε κάποιο αποθετήριο (repository) όπως το GitHub, απαιτείται η εγκατάσταση του Git για την κλωνοποίηση του αποθετηρίου.

Σε περιβάλλον **Ubuntu/Debian**, η εγκατάσταση του Git γίνεται με την εξής εντολή:

- sudo apt install git

Αφού εγκατασταθεί το Git, χρησιμοποιήστε την ακόλουθη εντολή για να κλωνοποιήσετε το αποθετήριο της εφαρμογής στον τοπικό υπολογιστή:

- git clone https://github.com/username/repository-name.git
- cd repository-name

Η παραπάνω εντολή θα κατεβάσει τον κώδικα της εφαρμογής και θα σας τοποθετήσει στον κατάλογο της εφαρμογής.

Αφού γίνει η κλωνοποίηση του αποθετηρίου, είναι απαραίτητη η εγκατάσταση των εξαρτήσεων της εφαρμογής. Αυτό επιτυγχάνεται μέσω της ακόλουθης εντολής:

- npm install

Η εντολή αυτή διαβάζει το αρχείο package.json και εγκαθιστά όλα τα απαραίτητα πακέτα που χρειάζεται η εφαρμογή για να λειτουργήσει.
Σε περίπτωση που η εφαρμογή χρησιμοποιεί περιβαλλοντικές μεταβλητές (όπως πληροφορίες για βάση δεδομένων ή κλειδιά API), αυτές θα πρέπει να διαμορφωθούν σωστά. Για τον σκοπό αυτό, δημιουργούμε ένα αρχείο .env στον κατάλογο της εφαρμογής με την εξής εντολή:

- touch .env

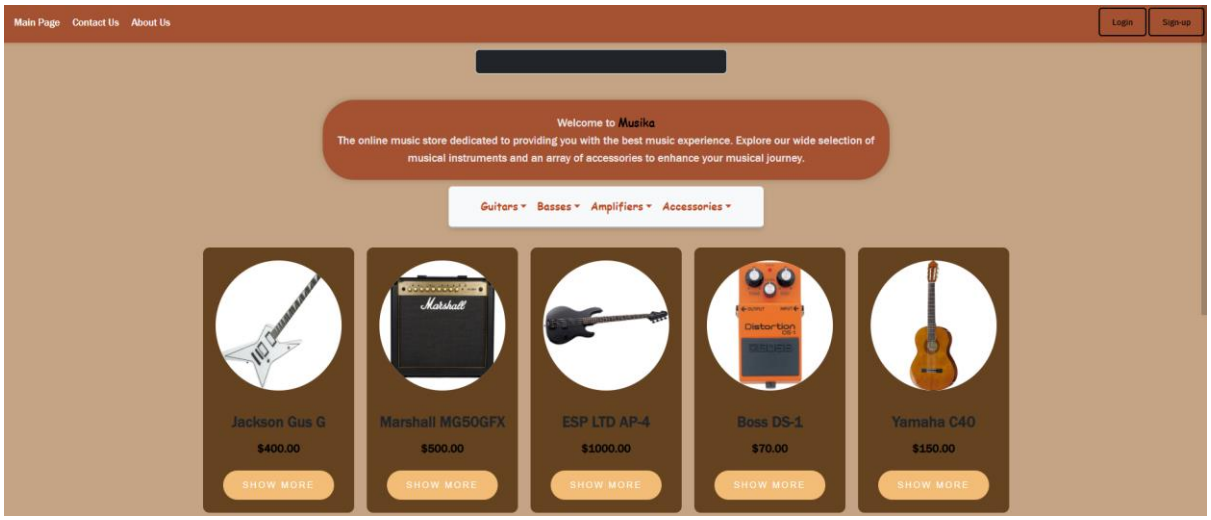
Αφού ολοκληρωθούν τα παραπάνω βήματα, μπορείτε να εκκινήσετε την εφαρμογή τοπικά χρησιμοποιώντας την εξής εντολή:

- npm start

Τέλος για την εκκίνηση της εφαρμογής πληκτρολογούμε στον web browse **http://localhost:(PORT)**

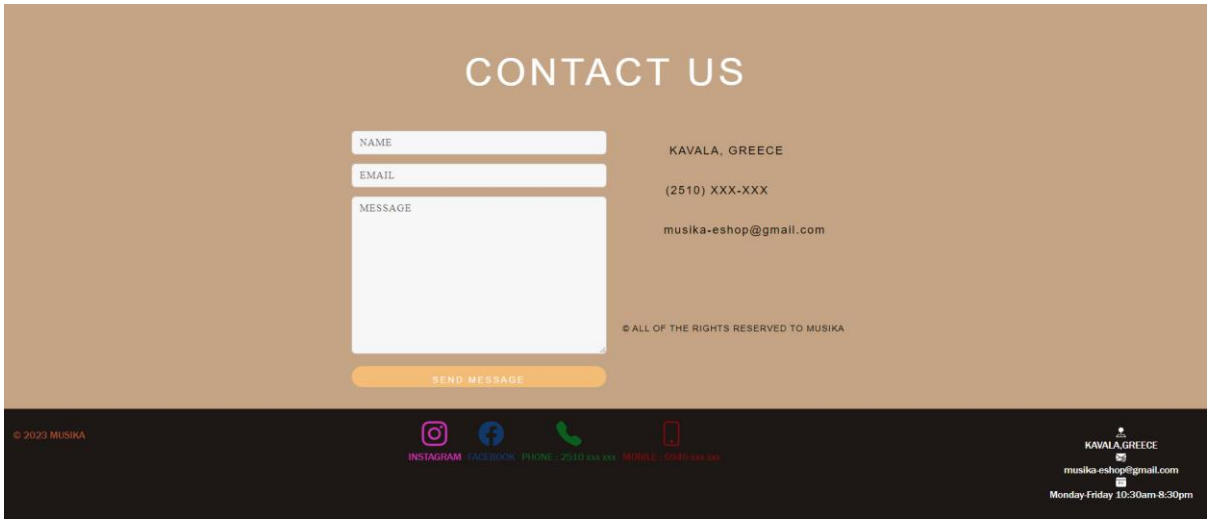
Εκτέλεση της ιστοσελίδας

Η αποτελεσματική λειτουργία μιας ιστοσελίδας δεν περιορίζεται αποκλειστικά στην ορθότητα του κώδικα, αλλά περιλαμβάνει και την συνολική εμπειρία του χρήστη (UX), τη διάταξη, το στυλ και τη λειτουργικότητα της σελίδας. Οι κύριες λειτουργίες, όπως η πλοήγηση, η ασφάλεια και η προσαρμοστικότητα, θα πρέπει να είναι ευανάγνωστες και φιλικές προς τον χρήστη. Η διάταξη πρέπει να εξασφαλίζει μια σαφή ιεραρχία περιεχομένου και ευκολία στην πλοήγηση. Η αισθητική θα πρέπει να περιλαμβάνει προσεκτικά επιλεγμένα χρώματα, ευανάγνωστη τυπογραφία και σχετικές εικόνες, ενώ η ταχύτητα φόρτωσης, η διαδραστικότητα και η προσβασιμότητα είναι κρίσιμες για μια ευχάριστη εμπειρία χρήστη.

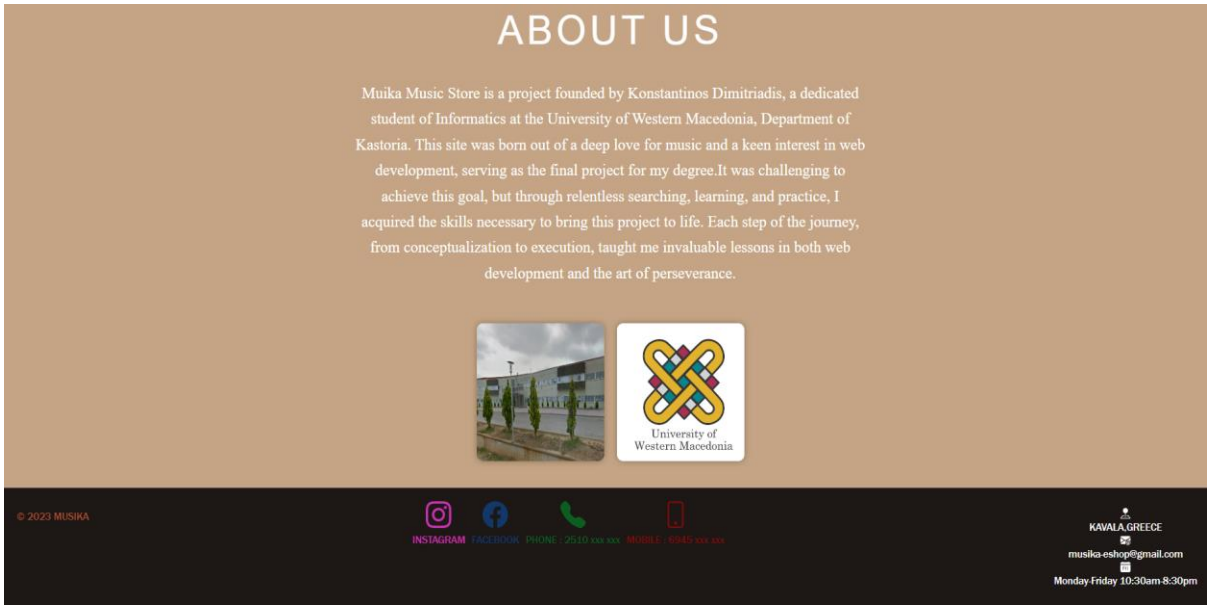


Αρχική σελίδα της εργασίας pt1.

Αρχική σελίδα της εργασίας pt2.

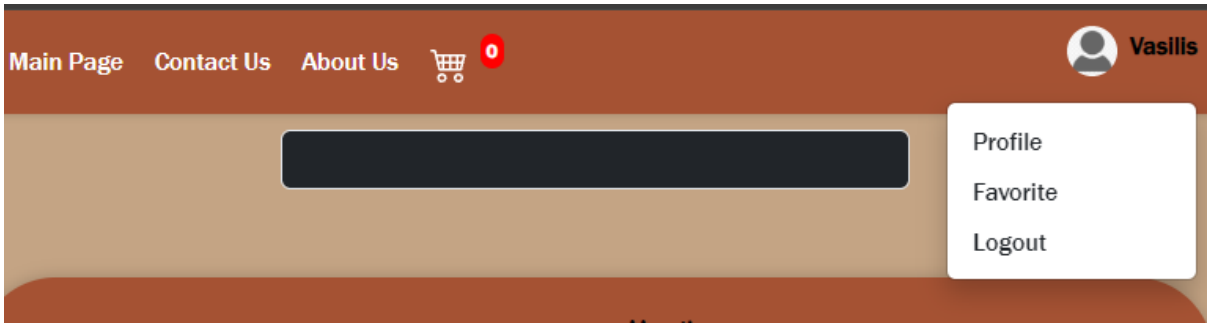


Contact Us σελίδα



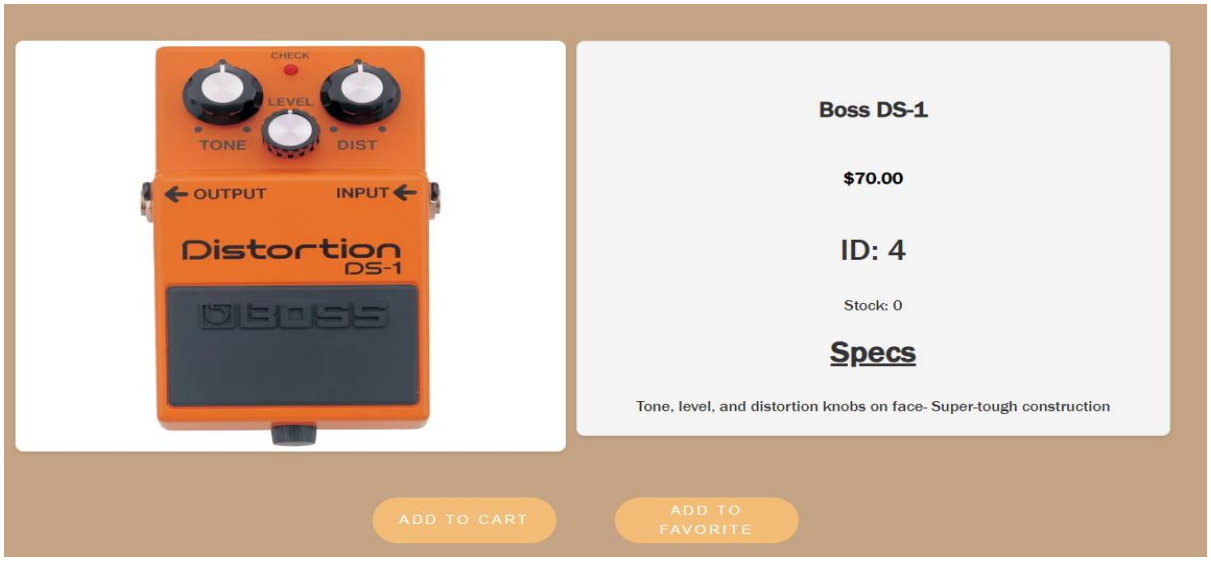
About Us σελίδα

Το πρώτο σημείο που βλέπει ο χρήστης στην ιστοσελίδα είναι η αρχική σελίδα όπου απεικονίζονται κάποια από τα προϊόντα που διατίθενται, η αναζήτηση προϊόντος, το logo της σελίδας, ένα slideshow απεικόνισης και τέλος το footer με την επωνυμία, τα social media, καθώς και την περιοχή, το email και τα ωράρια λειτουργίας. Οι επιλογές που δίνονται σε έναν μη εγγεγραμμένο χρήστη είναι να η σελίδα Contact Us και η About Us, καθώς μπορεί να εγγραφεί ή να συνδεθεί στην σελίδα.

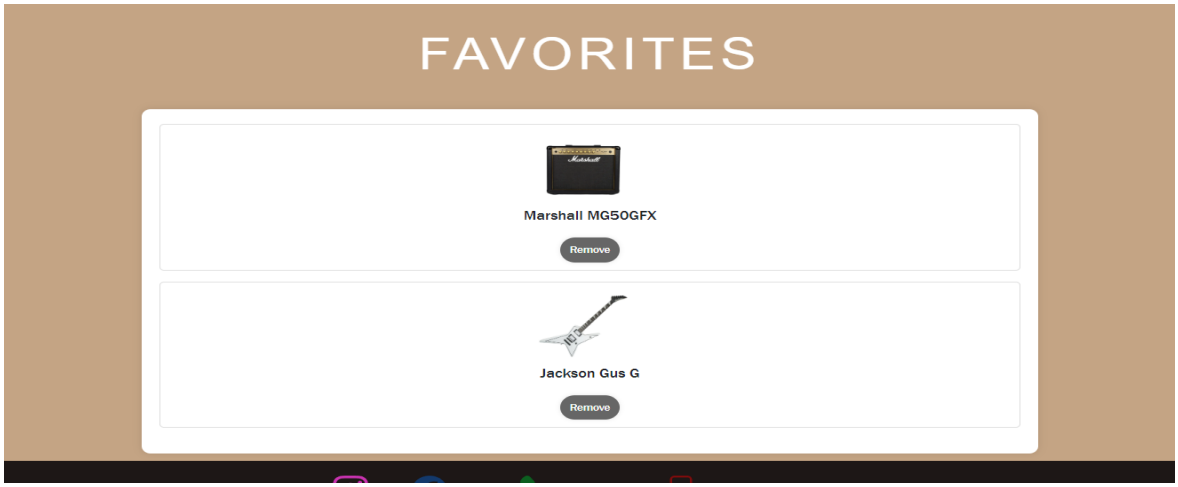


Αρχική σελίδα συνδεδεμένου χρήστη.

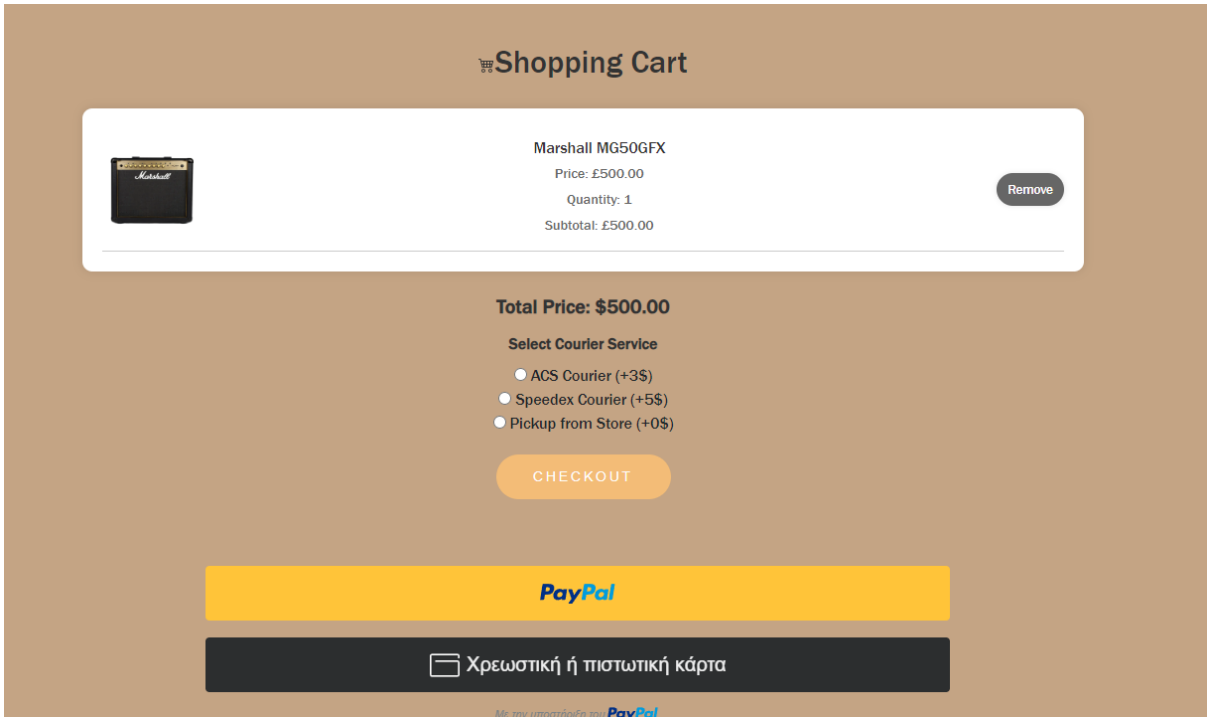
Όταν ένας χρήστης συνδεθεί λαμβάνει ένα μοναδικό id μέσω της καταγραφής του στη βάσης δεδομένων καθώς εγγράφεται με το email του, το όνομα και τον προσωπικό του κωδικό. Οι δυνατότητες που έχει ένας εγγεγραμμένος χρήστης είναι να προσθέτει προϊόντα στο καλάθι του, να τα προσθέτει στα αγαπημένα του και να πραγματοποιεί αγορές και να διαχειρίζεται τις πληροφορίες του προφίλ του.



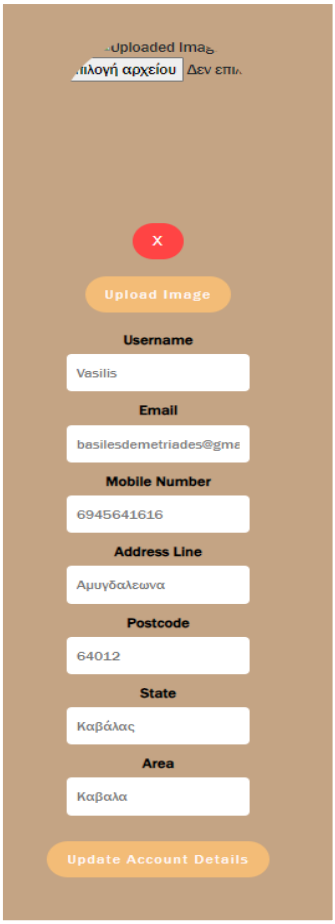
Σελίδα παρουσίασης προϊόντος.



Σελίδα αγαπημένων προϊόντων του χρήστη.

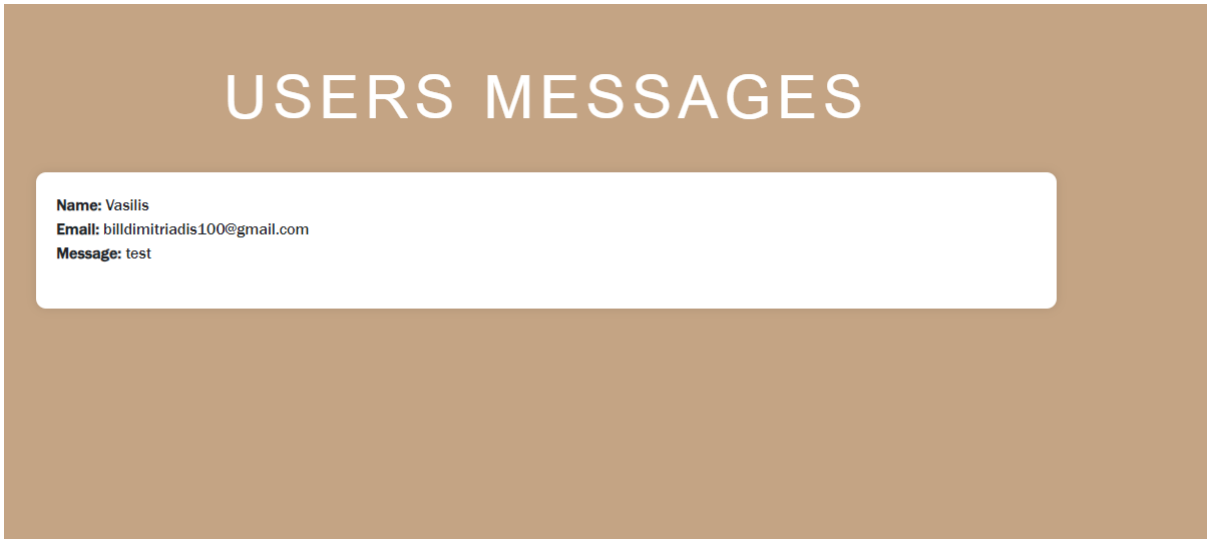


Καλάθι του χρήστη.

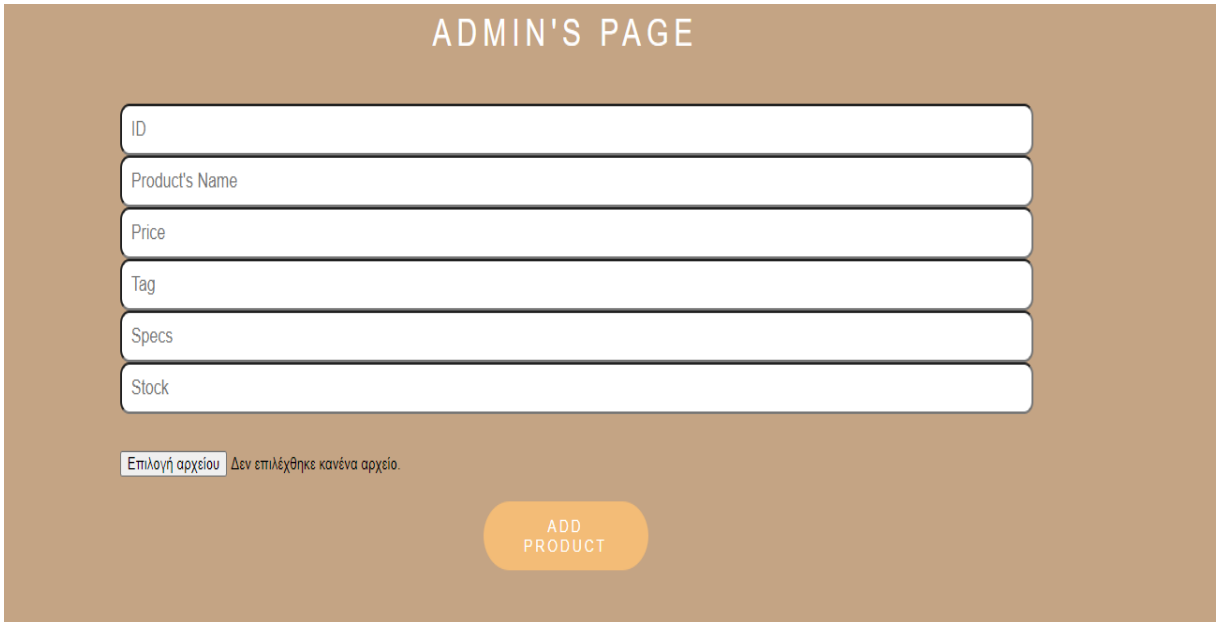


Σελίδα Προφίλ του χρήστη.

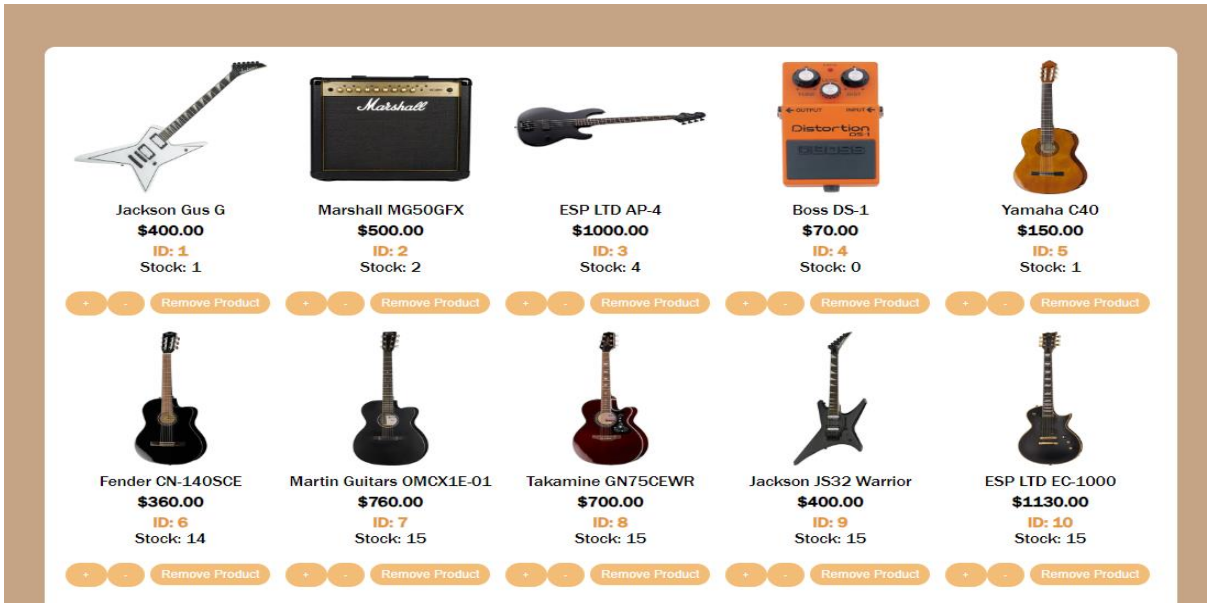
Πέραν των χρηστών, υπάρχει και η σελίδα του διαχειριστή(admin) του οποίου του δίνονται δραστηριότητες όπως να ελέγχει τα μηνύματα των χρηστών μέσω της Contact Us σελίδας, να προσθέτει νέα προϊόντα και να τα διαχειρίζεται καθώς και να επεξεργάζεται τις παραγγελίες των πελατών.



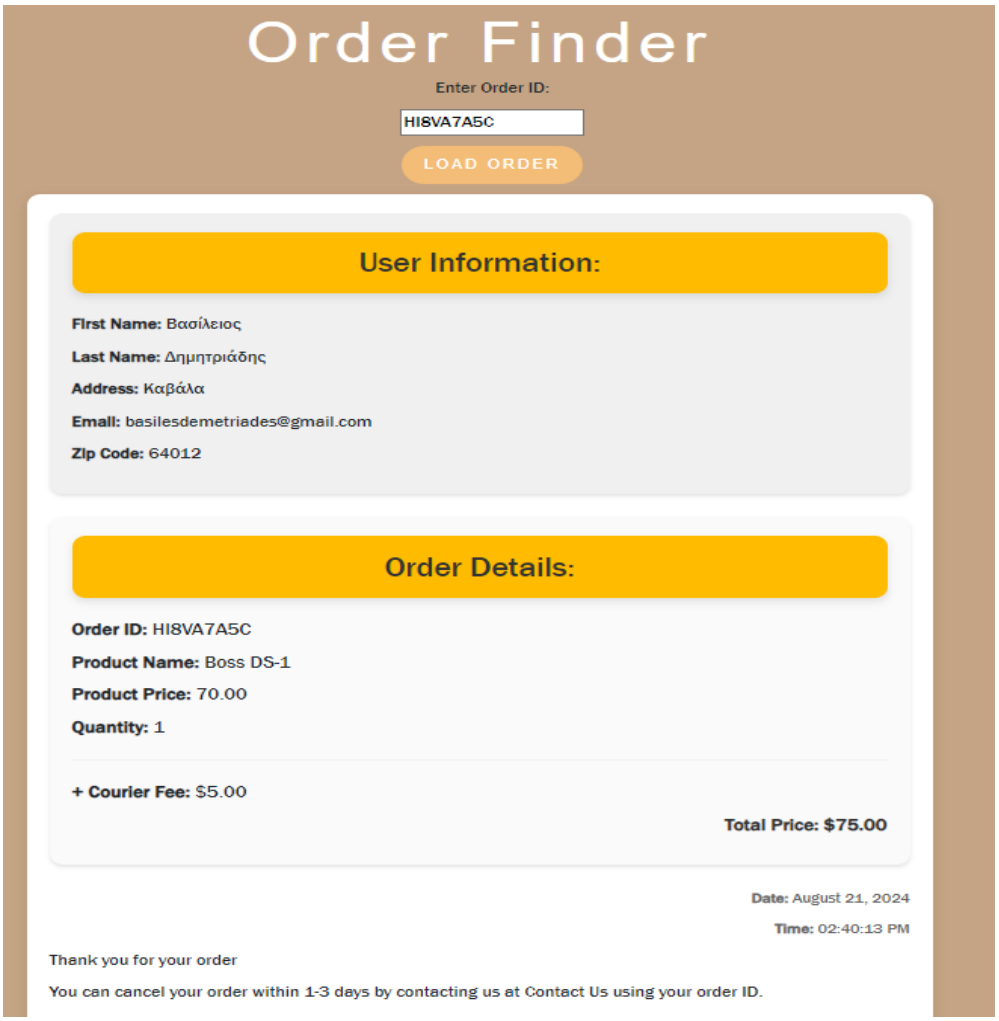
Σελίδα αποτύπωσης μηνυμάτων των χρηστών(ορατή μόνο στον admin)



Σελίδα προθήκης νέου προϊόντος



Σελίδα διαχείρισης προϊόντων



Σελίδα διαχείρισης παραγγελιών

Ο ολοκληρωμένος κώδικας της εργασίας βρίσκεται στο github στον σύνδεσμο https://github.com/Havocus/ptixiaki , όπου μπορείτε να δείτε αναλυτικά την σύνταξη του.