ΠΑΝΕΠΙΣΤΗΜΙΟ
ΙΩΑΝΝΙΝΩΝ

ΣΧΟΛΗ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΕΦΑΡΜΟΓΗ ΓΙΑ ΑΓΡΟΤΙΚΑ ΠΡΟΪΟΝΤΑ – FARMER APP

Δασκαλόπουλος Ηλίας

Επιβλέπων: Φώτιος Βαρτζιώτης

Επίκουρος Καθηγητής

Άρτα, Μάρτιος, 2026

APPLICATION FOR AGRICULTURAL PRODUCTS – FARMER APP

Εγκρίθηκε από τριμελή εξεταστική επιτροπή

Άρτα, 16/03/2026

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Επιβλέπων καθηγητής

   ΦΩΤΙΟΣ ΒΑΡΤΣΙΩΤΗΣ

2. Μέλος επιτροπής

   ΔΗΜΗΤΡΙΟΣ ΛΙΑΡΟΚΑΠΗΣ

3. Μέλος επιτροπής

   ΚΑΤΣΗΣ ΧΡΗΣΤΟΣ

Δήλωση μη λογοκλοπής

Δηλώνω υπεύθυνα και γνωρίζοντας τις κυρώσεις του Ν. 2121/1993 περί Πνευματικής Ιδιοκτησίας, ότι η παρούσα μεταπτυχιακή εργασία είναι εξ ολοκλήρου αποτέλεσμα δικής μου ερευνητικής εργασίας, δεν αποτελεί προϊόν αντιγραφής ούτε προέρχεται από ανάθεση σε τρίτους. Όλες οι πηγές που χρησιμοποιήθηκαν (κάθε είδους, μορφής και προέλευσης) για τη συγγραφή της περιλαμβάνονται στη βιβλιογραφία.

Δασκαλόπουλος, Ηλίας

Υπογραφή

Περίληψη

Η παρούσα πτυχιακή εργασία αφορά τη σχεδίαση και την υλοποίηση μιας δυναμικής διαδικτυακής εφαρμογής αγγελιών με την ονομασία FarmerApp, η οποία αποσκοπεί στη διευκόλυνση των συναλλαγών στον πρωτογενή τομέα. Στόχος της εφαρμογής είναι η δημιουργία ενός σύγχρονου ψηφιακού περιβάλλοντος όπου παραγωγοί και αγρότες μπορούν να προβάλλουν προϊόντα, μηχανήματα και σοδειές, ενώ οι ενδιαφερόμενοι αγοραστές έχουν τη δυνατότητα να αναζητούν και να εντοπίζουν αγροτικά αγαθά βάσει κατηγορίας και γεωγραφικής περιοχής.

Για την ανάπτυξη του συστήματος επιλέχθηκε το τεχνολογικό σχήμα MEN stack (MongoDB, Express.js, Node.js), το οποίο εξασφαλίζει υψηλή απόδοση, επεκτασιμότητα και ταχύτητα απόκρισης. Η αρχιτεκτονική βασίστηκε στο πρότυπο Model-View-Controller (MVC), επιτυγχάνοντας τον σαφή διαχωρισμό της επιχειρησιακής λογικής από τη διεπαφή χρήστη. Ιδιαίτερη έμφαση δόθηκε στην ασφάλεια των δεδομένων μέσω κρυπτογράφησης κωδικών πρόσβασης με χρήση Bcrypt, καθώς και στη διαχείριση αρχείων πολυμέσων μέσω Multer.

Η μεθοδολογία που ακολουθήθηκε περιλάμβανε την ανάλυση απαιτήσεων, τον σχεδιασμό διαγραμμάτων UML και της βάσης δεδομένων, την υλοποίηση των επιμέρους λειτουργιών και τον έλεγχο ορθής λειτουργίας της εφαρμογής. Το τελικό αποτέλεσμα είναι μια σύγχρονη, αποκρινόμενη (responsive) εφαρμογή με σκοτεινό θέμα (dark mode), η οποία προσφέρει φιλική εμπειρία χρήστη σε κάθε είδους συσκευή. Η εργασία αναδεικνύει τον ρόλο των

σύγχρονων τεχνολογιών ιστού στον εκσυγχρονισμό των παραδοσιακών μεθόδων αγοροπωλησίας στον αγροτικό τομέα.

Λέξεις-κλειδιά: διαδικτυακή εφαρμογή, αγροτικές αγγελίες, MEN stack, αρχιτεκτονική MVC, ψηφιακό marketplace.

Abstract

This thesis focuses on the design and implementation of a dynamic web-based classifieds application named FarmerApp, aiming to facilitate transactions within the primary sector. The main objective of the platform is to create a modern digital environment where producers and farmers can promote their products, machinery, and crops, while prospective buyers can easily search and locate agricultural goods based on category and geographical region.

The system was developed using the MEN stack (MongoDB, Express.js, Node.js), which ensures high performance, scalability, and responsiveness. The application architecture follows the Model-View-Controller (MVC) pattern, achieving a clear separation between business logic and user interface. Particular emphasis was placed on data security through password encryption using Bcrypt, as well as on multimedia file management through Multer.

The methodology included requirements analysis, UML diagram design, database schema modeling, system implementation, and functional testing. The final outcome is a modern, responsive web application featuring a dark mode theme, providing a user-friendly experience across various devices. This thesis highlights the contribution of modern web technologies to the modernization of traditional buying and selling practices in the agricultural sector.

Keywords: web application, agricultural classifieds, MEN stack, MVC architecture, digital marketplace

Περιεχόμενα

Κεφάλαιο 1ο: Εισαγωγή

Στη σύγχρονη εποχή, ο ψηφιακός μετασχηματισμός επηρεάζει κάθε πτυχή της οικονομικής δραστηριότητας, με τον αγροτικό τομέα να μην αποτελεί εξαίρεση. Η ανάγκη για άμεση επικοινωνία μεταξύ παραγωγών και αγοραστών, η εξάλειψη των περιττών μεσαζόντων και η ανάγκη για διαφάνεια στις τιμές των αγροτικών προϊόντων και μηχανημάτων, οδήγησαν στη δημιουργία εξειδικευμένων ψηφιακών πλατφορμών.

Το **FarmerApp** γεννήθηκε ως μια λύση που γεφυρώνει το χάσμα μεταξύ της παραδοσιακής γεωργίας και της σύγχρονης ηλεκτρονικής αγοράς (e-marketplace). Πρόκειται για μια εφαρμογή που επιτρέπει στον αγρότη, τον κτηνοτρόφο και τον επαγγελματία του κλάδου να διαχειρίζεται τις αγγελίες του με ευκολία, χρησιμοποιώντας σύγχρονα εργαλεία ιστού.

1.1 Αντικείμενο της Πτυχιακής Εργασίας

Αντικείμενο της παρούσας πτυχιακής εργασίας αποτελεί η μελέτη, ο σχεδιασμός και η υλοποίηση μιας ολοκληρωμένης (Full-stack) δυναμικής διαδικτυακής πλατφόρμας, με την ονομασία **"FarmerApp"**. Η εφαρμογή αυτή λειτουργεί ως ένας ψηφιακός κόμβος αγοραπωλησιών, ο οποίος εξειδικεύεται στον πρωτογενή τομέα, επιτρέποντας στους χρήστες την καταχώριση και αναζήτηση αγγελιών που αφορούν αγροτικά προϊόντα, γεωργικό εξοπλισμό, κτηνοτροφικό κεφάλαιο και αγροτεμάχια.

Η ανάπτυξη της εφαρμογής βασίστηκε στην αρχιτεκτονική προσέγγιση **MVC (Model-View-Controller)**, η οποία επιλέχθηκε για να διασφαλίσει τον πλήρη διαχωρισμό της επιχειρηματικής λογικής (Business Logic) από το περιβάλλον διεπαφής του χρήστη (User Interface). Το τεχνολογικό υπόβαθρο της υλοποίησης στηρίζεται στο οικοσύστημα της **JavaScript**, χρησιμοποιώντας το περιβάλλον εκτέλεσης **Node.js** και το framework **Express.js** για την πλευρά του διακομιστή (Server-side).Πιο συγκεκριμένα, το αντικείμενο της εργασίας αναλύεται στους εξής άξονες:

Α. Σχεδιασμός και Διαχείριση Μη Σχεσιακής Βάσης Δεδομένων Κεντρικό πυλώνα της εργασίας αποτελεί η δημιουργία και η παραμετροποίηση μιας δυναμικής βάσης δεδομένων με τη χρήση του **MongoDB**. Σε αντίθεση με τις παραδοσιακές σχεσιακές βάσεις (SQL), η επιλογή μιας NoSQL βάσης έγινε με γνώμονα την ευελιξία στη δομή των δεδομένων (Schema flexibility), η οποία είναι απαραίτητη σε μια εφαρμογή αγγελιών όπου οι κατηγορίες προϊόντων (π.χ. από τρακτέρ μέχρι σπόρους) έχουν διαφορετικά χαρακτηριστικά. Μέσω του **Mongoose (ODM)**, υλοποιήθηκαν μοντέλα δεδομένων για τους χρήστες και τα προϊόντα, διασφαλίζοντας την ακεραιότητα και την ταχύτητα στην ανάκτηση της πληροφορίας.

Β. Ανάπτυξη Μηχανισμού Αυθεντικοποίησης και Προστασίας Δεδομένων Μια κρίσιμη πτυχή του αντικειμένου είναι η διασφάλιση των προσωπικών δεδομένων των χρηστών. Στο πλαίσιο αυτό, αναπτύχθηκε ένα πλήρες σύστημα εγγραφής (Sign-up) και σύνδεσης (Login) χρηστών. Η τεκμηρίωση της ασφάλειας βασίζεται στη χρήση της βιβλιοθήκης **Bcrypt** για την

κρυπτογράφηση (hashing) των κωδικών πρόσβασης πριν την αποθήκευσή τους στη βάση, καθώς και στη διαχείριση συνεδριών (**Sessions**) μέσω του express-session, ώστε η πλοήγηση του χρήστη να είναι εξατομικευμένη και ασφαλής.

Γ. Διαχείριση Πολυμέσων και Σύστημα Ανάρτησης Αρχείων Δεδομένου ότι η οπτική πληροφορία είναι ζωτικής σημασίας για την επιτυχία μιας αγγελίας, το αντικείμενο περιλαμβάνει την υλοποίηση ενός υποσυστήματος διαχείρισης αρχείων. Με τη χρήση του middleware **Multer**, η εφαρμογή επιτρέπει στους χρήστες να μεταφορτώνουν (upload) εικόνες σε πραγματικό χρόνο. Η εργασία πραγματεύεται τις προκλήσεις της τοπικής αποθήκευσης αρχείων, της μετονομασίας τους για την αποφυγή διενέξεων και της σύνδεσης των μεταδεδομένων της εικόνας με την αντίστοιχη εγγραφή στη βάση δεδομένων.

Δ. Σχεδιασμός Διεπαφής (UI/UX) και Δυναμικό Rendering Τέλος, το αντικείμενο ολοκληρώνεται με την κατασκευή ενός σύγχρονου και φιλικού περιβάλλοντος εργασίας. Η χρήση του **Tailwind CSS** επέτρεψε τη δημιουργία ενός "responsive" σχεδιασμού που προσαρμόζεται σε κάθε είδους οθόνη (Desktop, Tablet, Mobile). Η παραγωγή του περιεχομένου γίνεται δυναμικά μέσω της μηχανής προτύπων **EJS (Embedded JavaScript)**, η οποία επιτρέπει την ενσωμάτωση λογικής προγραμματισμού απευθείας στα αρχεία HTML, προσφέροντας έτσι μια ζωντανή και διαδραστική εμπειρία στον τελικό χρήστη.

1.2 Σκοπός και Στόχοι

Ο θεμελιώδης **σκοπός** της παρούσας πτυχιακής εργασίας είναι η δημιουργία ενός σύγχρονου ψηφιακού εργαλείου που θα λειτουργήσει ως καταλύτης για τον εκσυγχρονισμό των εμπορικών συναλλαγών στον αγροτικό τομέα. Η εφαρμογή στοχεύει στην κατάργηση των γεωγραφικών περιορισμών και των χρονικών καθυστερήσεων που συχνά αντιμετωπίζουν οι παραγωγοί κατά τη διάθεση των προϊόντων τους ή την αναζήτηση εξοπλισμού. Μέσω του FarmerApp, επιδιώκεται η δημιουργία μιας "ανοιχτής αγοράς" (Open Marketplace), όπου η πληροφορία διαχέεται ελεύθερα, ενισχύοντας τον υγιή ανταγωνισμό και την τοπική οικονομία.

Επιπλέον, σκοπός της εργασίας είναι η επίδειξη των δυνατοτήτων των σύγχρονων τεχνολογιών Web (JavaScript full-stack) στην επίλυση πραγματικών προβλημάτων της καθημερινότητας, προσφέροντας μια λύση που είναι ταυτόχρονα στιβαρή τεχνικά και απλή στη χρήση.

Για την επιτυχή εκπλήρωση του παραπάνω σκοπού, τέθηκαν οι εξής επιμέρους **στόχοι**:

Α. Λειτουργικοί Στόχοι (Functional Goals)

- **Απλοποίηση της Καταχώρισης:** Η παροχή ενός εύχρηστου μηχανισμού όπου ο αγρότης, ακόμα και με περιορισμένες ψηφιακές δεξιότητες, θα μπορεί σε ελάχιστο χρόνο να αναρτήσει μια αγγελία συνοδευόμενη από φωτογραφίες και τεχνικά χαρακτηριστικά.

- **Αποδοτική Κατηγοριοποίηση και Αναζήτηση:** Η οργάνωση των δεδομένων σε σαφείς κατηγορίες (Μηχανήματα, Ζώα, Σοδειά, Χωράφια κ.λπ.), ώστε ο υποψήφιος αγοραστής να εντοπίζει άμεσα αυτό που αναζητά μέσω έξυπνων φίλτρων.
- **Άμεση Επικοινωνία:** Η διευκόλυνση της αλληλεπίδρασης μεταξύ πωλητή και αγοραστή μέσω της άμεσης προβολής στοιχείων επικοινωνίας και τοποθεσίας, προωθώντας τη διαφάνεια και την εμπιστοσύνη στις συναλλαγές.

Β. Τεχνικοί Στόχοι (Technical Goals)

- **Υψηλή Διαθεσιμότητα και Ταχύτητα:** Η χρήση του ασύγχρονου μοντέλου του Node.js εξασφαλίζει ότι η εφαρμογή μπορεί να εξυπηρετεί ταυτόχρονα πολλούς χρήστες με ελάχιστη καθυστέρηση (low latency), κάτι κρίσιμο για μια πλατφόρμα με μεγάλο όγκο δεδομένων και εικόνων.
- **Ασφάλεια και Αξιοπιστία:** Ο σχεδιασμός ενός συστήματος όπου τα δεδομένα των χρηστών προστατεύονται από κακόβουλες ενέργειες. Αυτό περιλαμβάνει την ασφαλή αποθήκευση κωδικών πρόσβασης και τον έλεγχο των δικαιωμάτων (Authorization), διασφαλίζοντας ότι μόνο ο ιδιοκτήτης μιας αγγελίας μπορεί να την τροποποιήσει ή να τη διαγράψει.
- **Επεκτασιμότητα (Scalability):** Η αρχιτεκτονική της εφαρμογής και η χρήση της MongoDB στοχεύουν στο να μπορεί το σύστημα να δεχθεί χιλιάδες νέες εγγραφές προϊόντων χωρίς να απαιτείται ριζικός ανασχεδιασμός της βάσης δεδομένων.

Γ. Στόχοι Διεπαφής και Εμπειρίας Χρήστη (UI/UX Goals)

- **Προσαρμοστικότητα (Responsiveness):** Δεδομένου ότι ο αγρότης βρίσκεται συχνά στο πεδίο εργασίας, βασικός στόχος είναι η εφαρμογή να λειτουργεί άψογα σε κινητές συσκευές (Smartphones), προσφέροντας την ίδια λειτουργικότητα με την έκδοση για υπολογιστές.
- **Σύγχρονη Αισθητική:** Η χρήση του Tailwind CSS στοχεύει στη δημιουργία ενός περιβάλλοντος που εμπνέει επαγγελματισμό και σοβαρότητα, ενθαρρύνοντας τους χρήστες να εμπιστευτούν την πλατφόρμα για τις συναλλαγές τους.

1.3 Δομή της Εργασίας

Η παρούσα πτυχιακή εργασία διαρθρώνεται σε πέντε κύρια κεφάλαια, καθένα από τα οποία πραγματεύεται μια συγκεκριμένη πτυχή της ανάλυσης, του σχεδιασμού και της υλοποίησης της εφαρμογής **FarmerApp**. Η δομή ακολουθεί μια λογική αλληλουχία, ξεκινώντας από τη θεωρητική θεμελίωση και καταλήγοντας στην πρακτική εφαρμογή και τα συμπεράσματα.

Αναλυτικότερα:

- **Κεφάλαιο 1 - Εισαγωγή:** Παρουσιάζεται το γενικό πλαίσιο της εργασίας, ορίζεται το αντικείμενο μελέτης και αναλύονται ο σκοπός και οι επιμέρους στόχοι που τέθηκαν κατά την έναρξη του έργου.
- **Κεφάλαιο 2 - Μελέτη Τεχνολογιών και Βιβλιογραφική Επισκόπηση:** Πραγματοποιείται μια εκτενής αναφορά στις σύγχρονες τεχνολογίες ανάπτυξης διαδικτυακών εφαρμογών. Συγκρίνονται οι διάφορες αρχιτεκτονικές προσέγγισης και αιτιολογείται η επιλογή του Node.js, της MongoDB και των λοιπών εργαλείων που συνθέτουν το τεχνολογικό stack της εφαρμογής.
- **Κεφάλαιο 3 - Μεθοδολογία Υλοποίησης και Σχεδιασμός Συστήματος:** Αποτελεί το τεχνικό υπόβαθρο της εργασίας. Περιλαμβάνει την ανάλυση της αρχιτεκτονικής MVC, τον σχεδιασμό των μοντέλων δεδομένων και την παρουσίαση της λειτουργικότητας μέσω διαγραμμάτων UML (Use Case, Activity Diagrams), προσφέροντας μια εποπτική εικόνα της ροής εργασιών του συστήματος.
- **Κεφάλαιο 4 - Εγχειρίδιο Χρήσης και Παρουσίαση Εφαρμογής:** Στο κεφάλαιο αυτό γίνεται η επίδειξη του τελικού προϊόντος. Μέσα από πλούσιο φωτογραφικό υλικό (screenshots) και οδηγούς βήμα προς βήμα, περιγράφονται οι λειτουργίες της εγγραφής, της σύνδεσης και της διαχείρισης αγγελιών, ενώ δίνονται οδηγίες για την εγκατάσταση και παραμετροποίηση της εφαρμογής σε τοπικό περιβάλλον.
- **Κεφάλαιο 5 - Συμπεράσματα και Μελλοντικές Επεκτάσεις:** Η εργασία ολοκληρώνεται με την αξιολόγηση των αποτελεσμάτων. Συζητούνται οι προκλήσεις που αντιμετωπίστηκαν κατά την ανάπτυξη, τα οφέλη που προέκυψαν από τη χρήση των συγκεκριμένων τεχνολογιών και προτείνονται μελλοντικές βελτιώσεις για την περαιτέρω εξέλιξη της πλατφόρμας.

Τέλος, η εργασία συνοδεύεται από την απαραίτητη **Βιβλιογραφία**, καθώς και **Παραρτήματα** στα οποία παρατίθενται καίρια τμήματα του πηγαίου κώδικα για τον έλεγχο και την επαλήθευση της υλοποίησης.

Κεφάλαιο 2ο: Μελέτη Υπαρχουσών Τεχνολογιών & Επιλογή Stack

Η ραγδαία εξέλιξη του παγκόσμιου ιστού (World Wide Web) έχει οδηγήσει στη δημιουργία ενός πληθώρας εργαλείων και γλωσσών προγραμματισμού, καθιστώντας την επιλογή του κατάλληλου τεχνολογικού συνδυασμού (stack) μια σύνθετη αλλά καθοριστική διαδικασία. Για την ανάπτυξη του **FarmerApp**, η επιλογή των τεχνολογιών δεν βασίστηκε μόνο στη δημοτικότητά τους, αλλά κυρίως στην ικανότητά τους να ανταποκρίνονται στις απαιτήσεις για υψηλή ταχύτητα, επεκτασιμότητα και ασφάλεια.

Στο παρόν κεφάλαιο, πραγματοποιείται μια εκτενής επισκόπηση των σύγχρονων τεχνολογιών Web, διαχωρίζοντας τις λειτουργίες που εκτελούνται στην πλευρά του πελάτη (Client-side) από εκείνες στην πλευρά του διακομιστή (Server-side). Επιπλέον, εξετάζονται και συγκρίνονται τα δημοφιλέστερα frameworks ανάπτυξης, καθώς και τα διαφορετικά μοντέλα οργάνωσης δεδομένων (Σχεσιακές έναντι Μη-Σχεσιακών βάσεων). Η ανάλυση αυτή ολοκληρώνεται με την πλήρη αιτιολόγηση της επιλογής του **MEN stack** (MongoDB, Express, Node.js), αναδεικνύοντας τα πλεονεκτήματα που προσφέρει η χρήση μιας ενιαίας γλώσσας προγραμματισμού, της JavaScript, σε όλο το εύρος της εφαρμογής.

2.1 Επισκόπηση Τεχνολογιών Web (Client-side vs Server-side)

Η αρχιτεκτονική του σύγχρονου παγκόσμιου ιστού βασίζεται στο μοντέλο Πελάτη-Εξυπηρετητή (Client-Server) **[1]**. Σε αυτό το μοντέλο, οι αρμοδιότητες χωρίζονται σε δύο διακριτά επίπεδα: το Front-end (Client-side), το οποίο αφορά όλα όσα βλέπει και με τα οποία αλληλεπιδρά ο χρήστης στον περιηγητή του, και το Back-end (Server-side), το οποίο αποτελεί τον «εγκέφαλο» της εφαρμογής, διαχειρίζεται τη βάση δεδομένων και εκτελεί τη λογική των λειτουργιών. Η κατανόηση της ισορροπίας μεταξύ αυτών των δύο επιπέδων είναι απαραίτητη για τον σχεδιασμό εφαρμογών που είναι ταχύτατες, ασφαλείς και προσφέρουν υψηλή εμπειρία χρήσης.

2.1.1 Client-Side

Ο όρος **Client-side** αναφέρεται στο σύνολο των διεργασιών, της λογικής και της παρουσίασης που εκτελούνται τοπικά στην τερματική συσκευή του χρήστη μέσω του φυλλομετρητή (Web Browser). Στη σύγχρονη αρχιτεκτονική του διαδικτύου, η πλευρά του πελάτη δεν αποτελεί πλέον έναν απλό δέκτη στατικών εγγράφων, αλλά ένα δυναμικό περιβάλλον εκτέλεσης κώδικα που καθορίζει την ταχύτητα απόκρισης και την αισθητική αρτιότητα της εφαρμογής **[2]**. Ο πρωταρχικός στόχος του Client-side προγραμματισμού στο **FarmerApp** είναι η οπτικοποίηση των δεδομένων που αποστέλλονται από τον διακομιστή και η διαχείριση της αλληλεπίδρασης (interactivity) με τον χρήστη με τρόπο άμεσο και αποδοτικό.

Για την ολοκληρωμένη υλοποίηση της διεπαφής του FarmerApp, επιστρατεύτηκαν οι εξής τεχνολογικοί πυλώνες, αναλυόμενοι παραγραφικά:

- **Δομή και Σκελετός με HTML (HyperText Markup Language):** Η HTML5 χρησιμοποιήθηκε για να ορίσει τη σημασιολογική δομή (semantic structure) των σελίδων. Κάθε στοιχείο της εφαρμογής, από τις φόρμες εγγραφής χρηστών έως τις κάρτες παρουσίασης προϊόντων στο index.ejs, οργανώνεται σε ιεραρχικά στοιχεία (tags). Η σωστή δόμηση της HTML είναι κρίσιμη, καθώς επιτρέπει στα υπόλοιπα επίπεδα (CSS και JavaScript) να "αγκιστρωθούν" στον κώδικα και να εφαρμόσουν μορφοποίηση ή λειτουργικότητα.
- **Αισθητική και Σχεδιασμός με CSS & Tailwind CSS:** Η μορφοποίηση της εφαρμογής βασίστηκε στο **Tailwind CSS**, ένα utility-first framework που επιτρέπει την ταχύτατη σχεδίαση απευθείας μέσα στα αρχεία προτύπων. Μέσω αυτού, επιτεύχθηκε η δημιουργία ενός πλήρως αποκρινόμενου (responsive) περιβάλλοντος. Αυτό σημαίνει ότι ο κώδικας περιλαμβάνει ειδικούς κανόνες (media queries) που αναγνωρίζουν τις διαστάσεις της οθόνης, εξασφαλίζοντας ότι ένας αγρότης που χρησιμοποιεί το FarmerApp από το χωράφι μέσω smartphone θα έχει την ίδια λειτουργική εμπειρία με έναν χρήστη σε σταθερό υπολογιστή.
- **Δυναμική Αλληλεπίδραση με Client-side JavaScript:** Η JavaScript αποτελεί τη μοναδική γλώσσα προγραμματισμού που εκτελείται εγγενώς στους περιηγητές. Στην πλευρά του πελάτη, η χρήση της επικεντρώνεται στη βελτίωση της εμπειρίας χωρίς την ανάγκη συνεχών ανανεώσεων της σελίδας (page reloads). Ειδικότερα, χρησιμοποιείται για τον έλεγχο εγκυρότητας των δεδομένων στις φόρμες (client-side validation) —όπως ο έλεγχος αν ένα email έχει τη σωστή μορφή— και για τη διαχείριση συμβάντων (event handling), προσφέροντας μια ρευστή αίσθηση στην πλοήγηση.
- **Δυναμική Παραγωγή Προτύπων με EJS (Embedded JavaScript):** Παρόλο που το EJS εκτελείται τεχνικά στον διακομιστή για να "χτίσει" τη σελίδα, ο ρόλος του είναι καθοριστικός για το τι θα φτάσει τελικά στον Client. Επιτρέπει την ενσωμάτωση λογικής (όπως loops για την εμφάνιση πολλαπλών προϊόντων ή conditionals για την εμφάνιση του ονόματος του συνδεδεμένου χρήστη) μέσα στην HTML. Το αποτέλεσμα είναι ένας "ζωντανός" κώδικας που προσαρμόζεται στα δεδομένα που ανακτώνται από τη βάση δεδομένων σε κάθε αίτημα.

Η μεταφορά μέρους της επεξεργασίας στην πλευρά του πελάτη αποτελεί μια στρατηγική επιλογή που μειώνει δραστικά τον φόρτο εργασίας του διακομιστή (Server Load). Με την εκτέλεση τοπικών λειτουργιών, όπως η εμφάνιση αναδυόμενων μενού ή η προσωρινή αποθήκευση στοιχείων διεπαφής, η εφαρμογή επιτυγχάνει ταχύτερους χρόνους απόκρισης, παρέχοντας μια σύγχρονη και επαγγελματική αίσθηση στον τελικό χρήστη.

2.1.2 Server Side

Ο όρος **Server-side** αναφέρεται στο σύνολο των διεργασιών και της υπολογιστικής λογικής που λαμβάνουν χώρα στον απομακρυσμένο διακομιστή (server), αποτελώντας το υπόβαθρο που υποστηρίζει τη λειτουργία κάθε σύγχρονης διαδικτυακής εφαρμογής. Ενώ η πλευρά του πελάτη επικεντρώνεται στην οπτική παρουσίαση, η πλευρά του διακομιστή αποτελεί τον «εγκέφαλο» του συστήματος, αναλαμβάνοντας τη διαχείριση της πληροφορίας, την επιβολή κανόνων ασφαλείας και την εκτέλεση της επιχειρηματικής λογικής (Business Logic) **[3]**. Η σημασία του Server-side προγραμματισμού στο **FarmerApp** είναι καθοριστική, καθώς διασφαλίζει ότι οι ευαίσθητες λειτουργίες εκτελούνται σε ένα ελεγχόμενο περιβάλλον, μακριά από την άμεση πρόσβαση του τελικού χρήστη.

Στο πλαίσιο της εφαρμογής FarmerApp, οι κύριες λειτουργίες που επιτελούνται στον διακομιστή αναλύονται παραγραφικά ως εξής:

- **Επεξεργασία Αιτημάτων (Request Handling):** Ο διακομιστής, μέσω του framework Express.js, λειτουργεί ως ένας αδιάλειπτος δέκτης αιτημάτων HTTP. Όταν ένας χρήστης αλληλεπιδρά με την εφαρμογή (π.χ. επιλέγοντας την κατηγορία «Μηχανήματα»), ο διακομιστής αναλύει το αίτημα, αναγνωρίζει τις παραμέτρους (όπως το query string για το φιλτράρισμα) και δρομολογεί την εκτέλεση στον κατάλληλο ελεγκτή (controller) για να προετοιμάσει την απάντηση.
- **Διαχείριση Βάσης Δεδομένων και Ακεραιότητα:** Η επικοινωνία με τη **MongoDB** πραγματοποιείται αποκλειστικά από τον διακομιστή μέσω του Mongoose ODM. Αυτή η αρχιτεκτονική επιλογή είναι ζωτικής σημασίας για την ασφάλεια, καθώς η πλευρά του πελάτη δεν διαθέτει ποτέ απευθείας πρόσβαση στα διαπιστευτήρια ή τη δομή της βάσης. Ο διακομιστής είναι υπεύθυνος για την αποθήκευση των αγγελιών, την ανάκτηση των δεδομένων με βάση τα φίλτρα του χρήστη και τη διασφάλιση ότι οι αλλαγές (updates) γίνονται σύμφωνα με τους ορισμένους κανόνες.
- **Αυθεντικοποίηση και Έλεγχος Πρόσβασης:** Ο διακομιστής αναλαμβάνει την κρίσιμη αποστολή της διαχείρισης των συνεδριών (**Sessions**). Μέσω αυτών, ελέγχει αν ο χρήστης που ζητά να διαγράψει μια αγγελία είναι πράγματι ο δημιουργός της. Αυτή η διαδικασία επαλήθευσης δικαιωμάτων (Authorization) εκτελείται στο back-end, εμποδίζοντας οποιαδήποτε κακόβουλη προσπάθεια παράκαμψης των περιορισμών της διεπαφής.
- **Δυναμική Παραγωγή Περιεχομένου (Server-Side Rendering):** Χρησιμοποιώντας το **Node.js** σε συνδυασμό με τη μηχανή προτύπων **EJS**, ο διακομιστής δεν αποστέλλει απλώς στατικά αρχεία, αλλά «χτίζει» τη σελίδα HTML σε πραγματικό χρόνο. Ενσωματώνει τα δεδομένα που ανακτήθηκαν από τη βάση (π.χ. τίτλο, τιμή, εικόνα προϊόντος) μέσα στον κώδικα HTML πριν αυτός καταλήξει στον περιηγητή του χρήστη, προσφέροντας μια πλήρως ενημερωμένη και εξατομικευμένη σελίδα.

- **Προηγμένη Ασφάλεια και Προστασία:** Η πλευρά του διακομιστή αποτελεί το πρώτο και κύριο επίπεδο άμυνας της εφαρμογής. Εδώ υλοποιείται η κρυπτογράφηση των κωδικών πρόσβασης με τη χρήση του αλγορίθμου **Bcrypt**, καθώς και ο καθαρισμός των εισερχόμενων δεδομένων (sanitization) από τις φόρμες. Με τον τρόπο αυτό, ο διακομιστής αποτρέπει επιθέσεις τύπου SQL Injection (ή NoSQL Injection στην προκειμένη περίπτωση) και Cross-Site Scripting (XSS).

Η ισχύς του Server-side προγραμματισμού έγκειται στην ικανότητά του για κεντρική διαχείριση και συντονισμό. Διασφαλίζει ότι όλοι οι χρήστες της πλατφόρμας FarmerApp έχουν πρόσβαση στην ίδια, έγκυρη και συγχρονισμένη πληροφορία, ενώ παράλληλα διατηρείται ένα υψηλό επίπεδο ασφάλειας και σταθερότητας που θα ήταν αδύνατο να επιτευχθεί αν η λογική της εφαρμογής βασιζόταν αποκλειστικά στην πλευρά του πελάτη.

2.2 Σύγκριση Framworks: Node.js vs PHP vs Python (Django/Flask)

Με την εξέλιξη του Web 2.0 και την ανάγκη για ολοένα και πιο διαδραστικές εφαρμογές, αναπτύχθηκαν διαφορετικά οικοσυστήματα προγραμματισμού, το καθένα με τη δική του φιλοσοφία και αρχιτεκτονική. Στο πλαίσιο της ανάπτυξης του **FarmerApp**, κρίθηκε απαραίτητο να εξεταστούν οι κυρίαρχες λύσεις της αγοράς, ώστε η τελική επιλογή να προσφέρει τη βέλτιστη ισορροπία μεταξύ απόδοσης και ευελιξίας.

Η σύγκριση επικεντρώνεται σε τρεις βασικούς πυλώνες: το **Node.js**, το οποίο φέρνει τη JavaScript στον διακομιστή, την παραδοσιακή αλλά εξαιρετικά διαδεδομένη **PHP**, και την **Python** με τα ισχυρά frameworks Django και Flask. Η ανάλυση που ακολουθεί δεν εστιάζει μόνο στα τεχνικά χαρακτηριστικά της κάθε γλώσσας, αλλά και στην υποστήριξη που παρέχουν μέσω των βιβλιοθηκών τους, την ευκολία διασύνδεσης με βάσεις δεδομένων και τον τρόπο με τον οποίο διαχειρίζονται τα ταυτόχρονα αιτήματα των χρηστών. Μέσα από αυτή τη διαδικασία αξιολόγησης, αναδεικνύονται οι λόγοι για τους οποίους ορισμένες τεχνολογίες υπερτερούν σε περιβάλλοντα που απαιτούν επεξεργασία δεδομένων σε πραγματικό χρόνο (real-time data processing), όπως είναι μια πλατφόρμα αγγελιών.

2.2.1 Node.js

Το **Node.js** δεν αποτελεί μια ακόμη γλώσσα προγραμματισμού, αλλά ένα περιβάλλον εκτέλεσης (runtime environment) ανοιχτού κώδικα που επιτρέπει την εκτέλεση της JavaScript στην πλευρά του διακομιστή (Server-side). Βασίζεται στην πανίσχυρη μηχανή **V8** της Google (την ίδια που χρησιμοποιεί ο περιηγητής Chrome), η οποία μετατρέπει τον κώδικα JavaScript απευθείας σε γλώσσα μηχανής, προσφέροντας εξαιρετικά υψηλές επιδόσεις **[4]**. Η υιοθέτηση του Node.js για την ανάπτυξη του **FarmerApp** υπήρξε στρατηγικής σημασίας, καθώς επιτρέπει τη δημιουργία γρήγορων και κλιμακούμενων εφαρμογών δικτύου.

Τα κύρια χαρακτηριστικά και πλεονεκτήματα του Node.js αναλύονται παραγραφικά ως εξής:

- **Μοντέλο Event-Driven και Non-blocking I/O:** Σε αντίθεση με τις παραδοσιακές τεχνολογίες που δημιουργούν ένα νέο «νήμα» (thread) για κάθε αίτημα χρήστη, το Node.js λειτουργεί σε ένα μόνο νήμα χρησιμοποιώντας έναν βρόχο συμβάντων (Event

Loop). Αυτό σημαίνει ότι μπορεί να διαχειρίζεται χιλιάδες ταυτόχρονες συνδέσεις χωρίς να «παγώνει» την εκτέλεση του κώδικα περιμένοντας τη βάση δεδομένων ή την ανάγνωση αρχείων. Για μια εφαρμογή όπως το FarmerApp, όπου πολλοί χρήστες μπορεί να αναζητούν προϊόντα ταυτόχρονα, αυτή η αρχιτεκτονική διασφαλίζει ελάχιστους χρόνους αναμονής.

- **Ενιαία Γλώσσα Προγραμματισμού (Unified JavaScript):** Ένα από τα μεγαλύτερα οφέλη είναι η δυνατότητα χρήσης της JavaScript τόσο στο Front-end όσο και στο Back-end. Αυτό μειώνει την πολυπλοκότητα της ανάπτυξης, καθώς ο προγραμματιστής χρησιμοποιεί την ίδια σύνταξη, τις ίδιες δομές δεδομένων (JSON) και τις ίδιες βιβλιοθήκες σε όλο το εύρος του συστήματος, διευκολύνοντας τη συντήρηση και την επέκταση του κώδικα.
- **Το Οικοσύστημα NPM (Node Package Manager):** Το Node.js συνοδεύεται από το NPM, τη μεγαλύτερη αποθήκη λογισμικού στον κόσμο. Μέσω αυτού, ενσωματώθηκαν στην εφαρμογή κρίσιμα εργαλεία όπως το **Express.js** για τη δρομολόγηση, το **Mongoose** για τη σύνδεση με τη MongoDB και το **Bcrypt** για την ασφάλεια. Η δυνατότητα επαναχρησιμοποίησης έτοιμων, δοκιμασμένων πακέτων επιταχύνει σημαντικά τη διαδικασία ανάπτυξης.
- **Διαχείριση Δεδομένων σε Πραγματικό Χρόνο:** Λόγω της ελαφριάς και γρήγορης φύσης του, το Node.js είναι ιδανικό για εφαρμογές που απαιτούν συχνή ανανέωση δεδομένων. Στο FarmerApp, αυτό μεταφράζεται σε άμεση ενημέρωση των αγγελιών και γρήγορη φόρτωση εικόνων, προσφέροντας μια εμπειρία χρήσης που προσομοιάζει σε εφαρμογή επιφάνειας εργασίας (desktop-like experience).

Συνοψίζοντας, το Node.js επιλέχθηκε ως η βάση της εφαρμογής διότι συνδυάζει την ταχύτητα εκτέλεσης με την ευελιξία της JavaScript, επιτρέποντας στο FarmerApp να είναι αποδοτικό, ασφαλές και έτοιμο να διαχειριστεί μεγάλο όγκο πληροφοριών χωρίς να θυσιάζει την εμπειρία του χρήστη.

2.2.2 PHP

Η **PHP** είναι μία από τις παλαιότερες και πιο ευρέως χρησιμοποιούμενες γλώσσες προγραμματισμού στην πλευρά του διακομιστή (Server-side). Δημιουργήθηκε το 1994 και έκτοτε έχει αποτελέσει τη βάση για μερικές από τις μεγαλύτερες πλατφόρμες παγκοσμίως, όπως το WordPress και το Facebook. Πρόκειται για μια γλώσσα που σχεδιάστηκε εξαρχής για την ανάπτυξη ιστοσελίδων, προσφέροντας εξαιρετική ευκολία στη διασύνδεση με σχεσιακές βάσεις δεδομένων, όπως η MySQL **[5]**. Στο πλαίσιο της μελέτης για το **FarmerApp**, η PHP εξετάστηκε ως μια εναλλακτική λύση λόγω της τεράστιας κοινότητας υποστήριξης και της ευκολίας εύρεσης έτοιμων λύσεων φιλοξενίας (hosting).

Η λειτουργία και η θέση της PHP στο σύγχρονο οικοσύστημα αναλύονται παραγραφικά:

- **Μοντέλο Εκτέλεσης (Synchronous/Multi-threaded):** Σε αντίθεση με το Node.js, η PHP ακολουθεί παραδοσιακά ένα σύγχρονο μοντέλο εκτέλεσης. Για κάθε αίτημα (request) που δέχεται ο διακομιστής, η PHP δημιουργεί μια νέα διεργασία ή νήμα. Παρόλο που αυτό κάνει τον προγραμματισμό πιο απλό και προβλέψιμο, μπορεί να οδηγήσει σε υψηλή κατανάλωση πόρων όταν η εφαρμογή δέχεται μεγάλο αριθμό ταυτόχρονων χρηστών. Για το FarmerApp, κρίθηκε ότι το μη-μπλοκαρισμένο μοντέλο του Node.js προσφέρει καλύτερη διαχείριση πόρων για μελλοντική επέκταση.
- **Ενσωμάτωση στον Ιστό και Οικοσύστημα:** Η PHP διακρίνεται για την ικανότητά της να ενσωματώνεται απευθείας μέσα στον κώδικα HTML, γεγονός που την έκανε εξαιρετικά δημοφιλή για γρήγορα σενάρια ανάπτυξης (scripting). Διαθέτει το σύστημα διαχείρισης πακέτων **Composer**, το οποίο, αντίστοιχα με το NPM, προσφέρει πρόσβαση σε χιλιάδες βιβλιοθήκες. Ωστόσο, η ανάγκη για εκμάθηση μιας διαφορετικής σύνταξης (PHP στο back-end και JavaScript στο front-end) αυξάνει το χρόνο ανάπτυξης σε σχέση με τη χρήση μιας ενιαίας γλώσσας.
- **Frameworks και Σύγχρονη Ανάπτυξη:** Η γλώσσα έχει εξελιχθεί σημαντικά με την έλευση σύγχρονων frameworks όπως το **Laravel** και το **Symfony**, τα οποία εισήγαγαν το πρότυπο MVC και βελτίωσαν την ασφάλεια και τη δομή του κώδικα. Παρόλα αυτά, η PHP παραμένει άρρηκτα συνδεδεμένη με το μοντέλο «αίτημα-απάντηση» (request-response), καθιστώντας την λιγότερο αποδοτική από το Node.js σε εφαρμογές που απαιτούν αμφίδρομη επικοινωνία σε πραγματικό χρόνο (real-time features).
- **Συμβατότητα και Φιλοξενία:** Ένα από τα ισχυρότερα πλεονεκτήματα της PHP είναι η καθολική συμβατότητά της. Σχεδόν κάθε κοινόχρηστος διακομιστής (shared hosting) υποστηρίζει PHP χωρίς επιπλέον παραμετροποίηση. Αν και αυτό αποτελεί σημαντικό πλεονέκτημα για απλές ιστοσελίδες, για το FarmerApp προτιμήθηκε η ευελιξία και ο έλεγχος που προσφέρει το περιβάλλον του Node.js, παρά την ανάγκη για πιο εξειδικευμένο setup.

Συμπερασματικά, η PHP παραμένει μια ισχυρή και αξιόπιστη επιλογή για την ανάπτυξη διαδικτυακών εφαρμογών. Ωστόσο, για τις ανάγκες της παρούσας πτυχιακής, κρίθηκε ότι υστερεί έναντι του Node.js ως προς την ταχύτητα επεξεργασίας ταυτόχρονων αιτημάτων και την ομοιογένεια της γλώσσας προγραμματισμού σε όλο το stack.

2.2.3 Python (Django/Flask)

Η **Python** αποτελεί μια υψηλού επιπέδου γλώσσα προγραμματισμού, η οποία διακρίνεται για την αναγνωσιμότητά της και την απλότητα στη σύνταξη, καθιστώντας την ιδανική για ταχεία ανάπτυξη (rapid prototyping). Στον προγραμματισμό της πλευράς του διακομιστή, η Python παρέχει στιβαρές λύσεις που καλύπτουν από απλές μικρο-υπηρεσίες έως πολύπλοκες εταιρικές εφαρμογές. Η επιλογή της Python εξετάστηκε για το **FarmerApp** λόγω των ισχυρών εργαλείων που διαθέτει για την επεξεργασία δεδομένων, γεγονός που θα μπορούσε να φανεί

χρήσιμο σε μελλοντικές προσθήκες, όπως η ανάλυση τιμών αγοράς ή η πρόβλεψη σοδειάς **[6]**.

Οι δύο βασικές επιλογές που προσφέρει η Python στον ιστό αναλύονται παραγραφικά:

- **Django (The "Batteries-Included" Framework):** Το Django ακολουθεί τη φιλοσοφία του «όλα συμπεριλαμβάνονται». Προσφέρει έτοιμα υποσυστήματα για τη διαχείριση χρηστών, το panel διαχείρισης (Admin Interface) και την προστασία από κοινές επιθέσεις (CSRF, SQL Injection). Αν και η χρήση του Django θα επιτάχυνε την υλοποίηση του back-end, κρίθηκε ότι η δομή του είναι αρκετά «βαριά» και περιοριστική (opinionated) για τις ανάγκες του FarmerApp, το οποίο απαιτεί μεγαλύτερη ευελιξία στη διαχείριση των μη-σχεσιακών δεδομένων της MongoDB.
- **Flask (The Micro-framework):** Το Flask αποτελεί την αντίθετη προσέγγιση, προσφέροντας μόνο τα απολύτως απαραίτητα εργαλεία για τη λειτουργία ενός διακομιστή. Ο προγραμματιστής έχει την πλήρη ελευθερία να επιλέξει τις βιβλιοθήκες που θα χρησιμοποιήσει για τη βάση δεδομένων ή την αυθεντικοποίηση. Παρόλο που η ελαφριά του φύση προσομοιάζει στην ευελιξία του Express.js, η ανάγκη για χειροκίνητη σύνδεση πολλών διαφορετικών εξαρτημάτων αυξάνει την πιθανότητα σφαλμάτων κατά την ανάπτυξη.
- **Απόδοση και Διαχείριση Ταυτόχρονων Αιτημάτων:** Η Python, όπως και η PHP, αντιμετωπίζει προκλήσεις στη διαχείριση ταυτόχρονων αιτημάτων (concurrency) λόγω του Global Interpreter Lock (GIL). Παρόλο που υπάρχουν σύγχρονες λύσεις (όπως το ASGI), το Node.js παραμένει πιο αποδοτικό στη διαχείριση μεγάλου αριθμού I/O λειτουργιών (όπως το ανέβασμα εικόνων στο FarmerApp) λόγω της εγγενούς μη-μπλοκαρισμένης φύσης του.
- **Οικοσύστημα και Βιβλιοθήκες:** Το οικοσύστημα της Python είναι απαράμιλλο σε τομείς όπως η μηχανική μάθηση και η στατιστική ανάλυση. Ωστόσο, για μια εφαρμογή τύπου marketplace, το οικοσύστημα του Node.js (NPM) προσφέρει πιο εξειδικευμένα εργαλεία για την ανάπτυξη web εφαρμογών σε πραγματικό χρόνο, επιτρέποντας την απρόσκοπτη μετακίνηση δεδομένων JSON μεταξύ του διακομιστή και της διεπαφής.

Συνολικά, η Python αποτελεί μια εξαιρετική επιλογή για εφαρμογές που απαιτούν βαριά επεξεργασία δεδομένων ή χρήση αλγορίθμων τεχνητής νοημοσύνης. Ωστόσο, για την υλοποίηση του FarmerApp, κρίθηκε ότι η JavaScript (μέσω του Node.js) προσφέρει μεγαλύτερη ομοιογένεια και καλύτερη απόδοση στις καθημερινές λειτουργίες μιας πλατφόρμας αγγελιών.

2.3 Συστήματα Βάσεων Δεδομένων: Σχέσιακές (SQL) vs Μη-Σχεσιακές (NoSQL/MongoDB)

Κάθε σύγχρονη διαδικτυακή εφαρμογή, όπως το **FarmerApp**, βασίζεται σε έναν μηχανισμό μόνιμης αποθήκευσης δεδομένων (Persistence Layer). Η επιλογή ανάμεσα σε μια Σχεσιακή Βάση Δεδομένων (Relational Database - SQL) και μια Μη-Σχεσιακή (Non-Relational - NoSQL) αποτελεί μια από τις σημαντικότερες αποφάσεις κατά τον σχεδιασμό της αρχιτεκτονικής ενός συστήματος. Η απόφαση αυτή δεν αφορά μόνο την τεχνική υλοποίηση, αλλά επηρεάζει άμεσα τον τρόπο με τον οποίο μοντελοποιούνται οι οντότητες της πραγματικότητας (όπως οι χρήστες και οι αγγελίες) μέσα στον ψηφιακό κόσμο.

Στην ανάλυση που ακολουθεί, συγκρίνονται τα δύο επικρατούντα μοντέλα βάσεων δεδομένων. Από τη μία πλευρά, οι **SQL** βάσεις (όπως η MySQL και η PostgreSQL) προσφέρουν αυστηρή δομή και εγγυήσεις ακεραιότητας μέσω πινάκων και σχέσεων. Από την άλλη πλευρά, οι **NoSQL** βάσεις (με κύριο εκπρόσωπο τη MongoDB) εισάγουν μια πιο ευέλικτη προσέγγιση, βασισμένη σε έγγραφα (documents), η οποία ταιριάζει απόλυτα με τη δυναμική φύση των σύγχρονων εφαρμογών ιστού. Η κατανόηση των διαφορών τους είναι απαραίτητη για να δικαιολογηθεί η επιλογή της **MongoDB** ως της ιδανικής λύσης για τις ανάγκες του FarmerApp, όπου η ποικιλομορφία των αγροτικών προϊόντων απαιτεί ένα σχήμα δεδομένων που μπορεί να προσαρμόζεται εύκολα.

2.3.1 SQL

Οι σχεσιακές βάσεις δεδομένων βασίζονται στο μαθηματικό μοντέλο των σχέσεων και χρησιμοποιούν τη γλώσσα **SQL** για τη διαχείριση και την ανάκτηση των δεδομένων. Η θεμελιώδης αρχή τους είναι η οργάνωση της πληροφορίας σε αυστηρά ορισμένους πίνακες (tables), οι οποίοι αποτελούνται από γραμμές (records) και στήλες (fields). Κάθε πίνακας διαθέτει ένα προκαθορισμένο σχήμα (schema), το οποίο επιβάλλει συγκεκριμένους τύπους δεδομένων για κάθε στήλη, διασφαλίζοντας έτσι την ομοιογένεια και την εγκυρότητα της πληροφορίας **[7]**.

Τα κύρια χαρακτηριστικά των SQL βάσεων δεδομένων αναλύονται παραγραφικά ως εξής:

- **Αυστηρή Δομή και Σχήμα (Fixed Schema):** Πριν από την εισαγωγή οποιουδήποτε δεδομένου, πρέπει να έχει οριστεί πλήρως η δομή των πινάκων. Οποιαδήποτε αλλαγή στις ανάγκες της εφαρμογής (π.χ. προσθήκη ενός νέου χαρακτηριστικού σε ένα προϊόν) απαιτεί την τροποποίηση του σχήματος της βάσης (schema migration), μια διαδικασία που μπορεί να είναι χρονοβόρα και σύνθετη σε μεγάλης κλίμακας συστήματα.
- **Σχέσεις και Κανονικοποίηση (Normalization):** Η SQL βασίζεται στη σύνδεση πινάκων μέσω κοινών κλειδιών (Primary & Foreign Keys). Η διαδικασία της κανονικοποίησης στοχεύει στην ελαχιστοποίηση της επανάληψης των δεδομένων (data redundancy). Για παράδειγμα, τα στοιχεία ενός πωλητή αποθηκεύονται σε έναν πίνακα και συνδέονται με τον πίνακα των προϊόντων μέσω ενός αναγνωριστικού, αντί να επαναλαμβάνονται σε κάθε αγγελία.

- **Ιδιότητες ACID:** Οι σχεσιακές βάσεις δεδομένων διακρίνονται για την υποστήριξη των ιδιοτήτων ACID (Atomicity, Consistency, Isolation, Durability). Αυτό εγγυάται ότι οι συναλλαγές (transactions) εκτελούνται με απόλυτη αξιοπιστία, γεγονός που τις καθιστά ιδανικές για τραπεζικά συστήματα ή εφαρμογές όπου η ακεραιότητα των δεδομένων είναι η απόλυτη προτεραιότητα.
- **Προκλήσεις στην Επεκτασιμότητα:** Παρόλο που οι SQL βάσεις είναι εξαιρετικά ισχυρές, παρουσιάζουν δυσκολίες στην οριζόντια επεκτασιμότητα (Horizontal Scaling). Η διασπορά μιας σχεσιακής βάσης σε πολλούς διακομιστές είναι τεχνικά δύσκολη λόγω των περίπλοκων σχέσεων και των αναγκών για συγχρονισμό, οδηγώντας συχνά στη λύση της κατακόρυφης επεκτασιμότητας (Vertical Scaling), δηλαδή στην αγορά ισχυρότερου υλικού.

Για το **FarmerApp**, η χρήση μιας SQL βάσης εξετάστηκε αλλά κρίθηκε λιγότερο αποδοτική σε σχέση με τις NoSQL λύσεις. Η ανάγκη για αποθήκευση ποικιλόμορφων προϊόντων με διαφορετικά χαρακτηριστικά (π.χ. ένα τρακτέρ έχει ίππους, ενώ μια σοδειά σιταριού έχει βάρος και υγρασία) θα απαιτούσε πολλούς πίνακες και περίπλοκα JOIN ερωτήματα, αυξάνοντας την πολυπλοκότητα του κώδικα και μειώνοντας την ταχύτητα απόκρισης.

2.3.2 NoSQL

Οι μη-σχεσιακές βάσεις δεδομένων, γνωστές ως **NoSQL**, αποτελούν μια εναλλακτική προσέγγιση στη διαχείριση δεδομένων, η οποία απομακρύνεται από το παραδοσιακό μοντέλο των πινάκων και των αυστηρών σχέσεων. Σχεδιάστηκαν για να προσφέρουν οριζόντια επεκτασιμότητα (Horizontal Scalability) και τη δυνατότητα διαχείρισης μη δομημένων ή ημι-δομημένων δεδομένων. Η πιο διαδεδομένη κατηγορία τους, στην οποία ανήκει και η **MongoDB** που χρησιμοποιήθηκε στο **FarmerApp**, είναι οι βάσεις δεδομένων προσανατολισμένες σε έγγραφα (Document-oriented databases). Σε αυτό το μοντέλο, τα δεδομένα αποθηκεύονται σε έγγραφα μορφής **JSON** ή **BSON**, επιτρέποντας την ομαδοποίηση όλων των σχετικών πληροφοριών σε μία ενιαία οντότητα **[8]**.

Τα κύρια χαρακτηριστικά των NoSQL βάσεων δεδομένων αναλύονται παραγραφικά ως εξής:

- **Δυναμικό Σχήμα (Dynamic Schema):** Το σημαντικότερο πλεονέκτημα των NoSQL βάσεων είναι η απουσία ενός προκαθορισμένου και αυστηρού σχήματος. Κάθε έγγραφο (document) μπορεί να έχει διαφορετική δομή από τα υπόλοιπα στην ίδια συλλογή (collection). Για το FarmerApp, αυτό είναι κρίσιμο, καθώς επιτρέπει σε μια αγγελία για ένα τρακτέρ να περιλαμβάνει πεδία όπως "ιπποδύναμη", ενώ μια αγγελία για σιτάρι να περιλαμβάνει "ποσοστό υγρασίας", χωρίς να απαιτείται ανασχεδιασμός της βάσης.
- **Υψηλή Απόδοση και Ταχύτητα:** Λόγω της αποθήκευσης των δεδομένων σε ενιαία έγγραφα, αποφεύγεται η ανάγκη για περίπλοκα και δαπανηρά (σε πόρους) ερωτήματα σύνδεσης πινάκων (Joins). Η πληροφορία ανακτάται ως ένα αυτοτελές

αντικείμενο, γεγονός που μειώνει δραστικά τον χρόνο απόκρισης του συστήματος, ειδικά σε εφαρμογές με μεγάλη κίνηση.

- **Οριζόντια Επεκτασιμότητα (Sharding):** Οι NoSQL βάσεις είναι σχεδιασμένες να επεκτείνονται εύκολα προσθέτοντας περισσότερους διακομιστές στο δίκτυο. Μέσω της τεχνικής του sharding, τα δεδομένα κατανέμονται σε διαφορετικούς κόμβους, επιτρέποντας στην εφαρμογή να διαχειρίζεται απεριόριστο όγκο πληροφοριών και χρηστών χωρίς πτώση της απόδοσης.
- **Ευελιξία στην Ανάπτυξη:** Η χρήση μορφής δεδομένων JSON επιτρέπει την άμεση αντιστοίχιση των δεδομένων της βάσης με τα αντικείμενα της JavaScript στον κώδικα (Node.js). Αυτή η ομοιογένεια εξαλείφει την ανάγκη για περίπλοκα επίπεδα μετάφρασης δεδομένων, επιταχύνοντας τον κύκλο ανάπτυξης και κάνοντας τον κώδικα πιο καθαρό και εύκολα συντηρήσιμο.

Η επιλογή της **MongoDB** για το FarmerApp βασίστηκε στην ανάγκη για μια βάση δεδομένων που "μεγαλώνει" μαζί με την εφαρμογή. Η ευελιξία των εγγράφων επιτρέπει την εύκολη προσθήκη νέων κατηγοριών προϊόντων στο μέλλον, ενώ η ταχύτητα ανάκτησης διασφαλίζει ότι ο χρήστης θα λαμβάνει τα αποτελέσματα των αναζητήσεών του ακαριαία, βελτιώνοντας τη συνολική εμπειρία χρήσης της πλατφόρμας.

2.4 Αιτιολόγηση Επιλογής: Γιατί επιλέχθηκε το MERN/MEN stack (Node.js, Express, MongoDB, EJS)

Η τελική απόφαση για τη χρησιμοποίηση του **MEN stack** (παραλλαγή του MERN όπου το React αντικαθίσταται από τη μηχανή προτύπων **EJS** για ταχύτερο Server-Side Rendering) βασίστηκε στην ανάγκη για ένα ομοιογενές, ευέλικτο και υψηλής απόδοσης τεχνολογικό υπόβαθρο. Η επιλογή αυτή δεν ήταν τυχαία, αλλά προέκυψε από τη διαπίστωση ότι ο συνδυασμός αυτών των τεχνολογιών επιτρέπει την ανάπτυξη μιας πλήρους εφαρμογής χρησιμοποιώντας αποκλειστικά τη γλώσσα **JavaScript** σε όλα τα επίπεδα (Full-stack JavaScript). Αυτή η ομοιογένεια μειώνει δραστικά τον χρόνο ανάπτυξης, καθώς η ροή των δεδομένων από τη βάση (MongoDB) στον διακομιστή (Node/Express) και τελικά στη διεπαφή (EJS) γίνεται χωρίς την ανάγκη μετασχηματισμού των δομών από μια γλώσσα σε άλλη.

Οι βασικοί λόγοι της επιλογής αναλύονται παραγραφικά:

- **Ομοιογένεια Δεδομένων (JSON/BSON):** Το FarmerApp βασίζεται στην ανταλλαγή πληροφοριών αγγελιών που έχουν δυναμική φύση. Η MongoDB αποθηκεύει τα δεδομένα σε μορφή BSON (binary JSON), η οποία είναι άμεσα συμβατή με τα αντικείμενα JavaScript του Node.js. Αυτό σημαίνει ότι μια αγγελία που ανακτάται από τη βάση μπορεί να περάσει απευθείας στα αρχεία EJS και να προβληθεί στον χρήστη χωρίς περίπλοκες διαδικασίες parsing, διατηρώντας την ακεραιότητα και την ταχύτητα του συστήματος.

- **Ταχύτητα και Αποδοτικότητα με το Express.js:** Η επιλογή του Express.js ως framework του διακομιστή έγινε λόγω της μινιμαλιστικής του φύσης. Προσφέρει στον προγραμματιστή τον πλήρη έλεγχο των δρομολογήσεων (routing) και των middlewares (όπως το Multer για τις φωτογραφίες), επιτρέποντας στο FarmerApp να παραμένει ελαφρύ και γρήγορο, χωρίς τις περιττές λειτουργίες που ενδεχομένως να επέβαλαν πιο "βαριά" frameworks όπως το Django.
- **Αμεσότητα του Server-Side Rendering (EJS):** Σε αντίθεση με το React (που χρησιμοποιείται στο MERN), το **EJS** επιλέχθηκε γιατί επιτρέπει στον διακομιστή να παραδίδει έτοιμο κώδικα HTML στον περιηγητή. Αυτό είναι ιδιαίτερα σημαντικό για την πτυχιακή εργασία, καθώς διευκολύνει το SEO (Search Engine Optimization) – ώστε οι αγγελίες των αγροτών να βρίσκονται ευκολότερα από τις μηχανές αναζήτησης – και εξασφαλίζει ότι η εφαρμογή θα φορτώνει γρήγορα ακόμη και σε συσκευές με περιορισμένη επεξεργαστική ισχύ ή αργή σύνδεση στο διαδίκτυο.
- **Κλιμακωτή Ανάπτυξη (Scalability):** Το συγκεκριμένο stack επιτρέπει στην εφαρμογή να ξεκινήσει ως ένα MVP (Minimum Viable Product) και να επεκταθεί εύκολα. Η μη-σχεσιακή φύση της βάσης επιτρέπει την προσθήκη νέων πεδίων στις αγγελίες (π.χ. σύστημα αξιολόγησης ή GPS coordinates) ανά πάσα στιγμή, ενώ το Node.js εγγυάται ότι η πλατφόρμα θα μπορεί να υποστηρίξει εκατοντάδες ταυτόχρονους χρήστες κατά την περίοδο αιχμής των αγροτικών συναλλαγών.

Συνοψίζοντας, το MEN stack προσφέρει στο FarmerApp μια στιβαρή αρχιτεκτονική που συνδυάζει την ταχύτητα της JavaScript, την ευελιξία της NoSQL βάσης και την απλότητα των EJS templates. Το αποτέλεσμα είναι μια εφαρμογή που ανταποκρίνεται στις σύγχρονες απαιτήσεις του ιστού, παραμένοντας ταυτόχρονα εύκολα συντηρήσιμη και επεκτάσιμη.

Κεφάλαιο 3ο: Μεθοδολογία Υλοποίησης & Σχεδιασμός (UML)

Η επιτυχής ανάπτυξη μιας διαδικτυακής εφαρμογής απαιτεί πέρα από τη συγγραφή κώδικα, έναν προσεκτικό αρχιτεκτονικό σχεδιασμό και μια ξεκάθαρη μεθοδολογία υλοποίησης. Στο παρόν κεφάλαιο αναλύεται η στρατηγική που ακολουθήθηκε για τη δόμηση του **FarmerApp**, εστιάζοντας στον τρόπο με τον οποίο οι απαιτήσεις των χρηστών μεταφράστηκαν σε τεχνικές προδιαγραφές.

Η παρουσίαση ξεκινά με την ανάλυση του προτύπου **Model-View-Controller (MVC)**, το οποίο καθορίζει την οργάνωση του πηγαίου κώδικα και τη ροή των δεδομένων. Στη συνέχεια, μέσω της γλώσσας μοντελοποίησης **UML**, οπτικοποιούνται οι λειτουργίες του συστήματος και οι αλληλεπιδράσεις των χρηστών. Ιδιαίτερη έμφαση δίνεται στον σχεδιασμό της βάσης δεδομένων και των σχημάτων (schemas), καθώς και στον τρόπο διαχείρισης των αρχείων πολυμέσων, στοιχεία που αποτελούν τη ραχοκοκαλιά της πλατφόρμας αγγελιών. Η μεθοδολογική αυτή προσέγγιση διασφαλίζει ότι η εφαρμογή δεν είναι μόνο λειτουργική, αλλά και επεκτάσιμη, ασφαλής και εύκολα συντηρήσιμη.

3.1 Αρχιτεκτονική Συστήματος: Ανάλυση του προτύπου Model-View-Controller (MVC)

Η αρχιτεκτονική του **FarmerApp** βασίζεται στο πρότυπο **Model-View-Controller (MVC)**, ένα από τα πιο διαδεδομένα πρότυπα σχεδίασης λογισμικού για την ανάπτυξη διαδικτυακών εφαρμογών. Η κύρια φιλοσοφία του MVC είναι ο διαχωρισμός των ευθυνών (Separation of Concerns), όπου η εφαρμογή χωρίζεται σε τρία διακριτά επίπεδα που αλληλεπιδρούν μεταξύ τους με συγκεκριμένους κανόνες. Αυτή η προσέγγιση επιτρέπει στον προγραμματιστή να τροποποιεί ένα τμήμα της εφαρμογής (π.χ. τη σχεδίαση της διεπαφής) χωρίς να επηρεάζει τη λογική της βάσης δεδομένων.

Η εφαρμογή του προτύπου MVC στο FarmerApp αναλύεται παραγραφικά ως εξής:

- **Το Μοντέλο (Model - /models):** Το επίπεδο αυτό αποτελεί την αναπαράσταση των δεδομένων και των κανόνων που τα διέπουν. Στο FarmerApp, τα Models (όπως το User.js και το Product.js) ορίζουν τα σχήματα (schemas) της MongoDB μέσω του Mongoose. Είναι υπεύθυνα για την επικοινωνία με τη βάση δεδομένων, την ανάκτηση πληροφοριών και την επιβολή περιορισμών, όπως το αν ένα email χρήστη είναι μοναδικό ή αν μια τιμή προϊόντος είναι υποχρεωτική.
- **Η Προβολή (View - /views):** Το View αφορά το επίπεδο παρουσίασης, δηλαδή ό,τι βλέπει ο τελικός χρήστης στον περιηγητή του. Στην εργασία αυτή, τα Views υλοποιούνται με τη χρήση της μηχανής προτύπων **EJS**. Τα αρχεία EJS λαμβάνουν δεδομένα από τον Controller και τα μετατρέπουν δυναμικά σε HTML κώδικα. Για παράδειγμα, το αρχείο showItem.ejs είναι μια "μήτρα" που γεμίζει αυτόματα με τις πληροφορίες της εκάστοτε αγγελίας που ζητά ο χρήστης.

- **Ο Ελεγκτής (Controller/Routes - /routes):** Ο Controller λειτουργεί ως ο ενδιάμεσος κρίκος που συντονίζει τη ροή των δεδομένων. Στην υλοποίησή μας, ο ρόλος αυτός επιτελείται από τα αρχεία στα routes (π.χ. productRoutes.js). Όταν ένας χρήστης κάνει ένα αίτημα (π.χ. κλικ στο "Περισσότερα"), ο Controller λαμβάνει το αίτημα, ζητά τα δεδομένα από το Model, και στη συνέχεια επιλέγει το σωστό View για να τα προβάλει. Επίσης, διαχειρίζεται τη λογική των λειτουργιών, όπως ο έλεγχος αν ένας κωδικός πρόσβασης είναι σωστός κατά τη σύνδεση.

Η Ροή Εργασίας (Workflow) στο FarmerApp:

1. Ο χρήστης στέλνει ένα αίτημα μέσω του browser (π.χ. αναζήτηση στην κατηγορία "Ζώα").
2. Το **Route (Controller)** λαμβάνει το αίτημα και καλεί το **Product Model**.
3. Το **Model** εκτελεί το ερώτημα στη **MongoDB** και επιστρέφει τη λίστα με τα ζώα στον Controller.
4. Ο **Controller** στέλνει αυτή τη λίστα στο **index.ejs (View)**.
5. Το **View** παράγει την τελική σελίδα και την επιστρέφει στον χρήστη.

Η υιοθέτηση της αρχιτεκτονικής MVC στο FarmerApp προσέφερε στην ανάπτυξη εξαιρετική ευελιξία. Επέτρεψε την ανεξάρτητη δοκιμή των λειτουργιών και κατέστησε την εφαρμογή εύκολα επεκτάσιμη, καθώς η προσθήκη νέων δυνατοτήτων (όπως ένα σύστημα σχολίων) απαιτεί απλώς τη δημιουργία ενός νέου Model, ενός View και της αντίστοιχης διαδρομής (Route), χωρίς να διαταράσσεται η υπάρχουσα δομή.

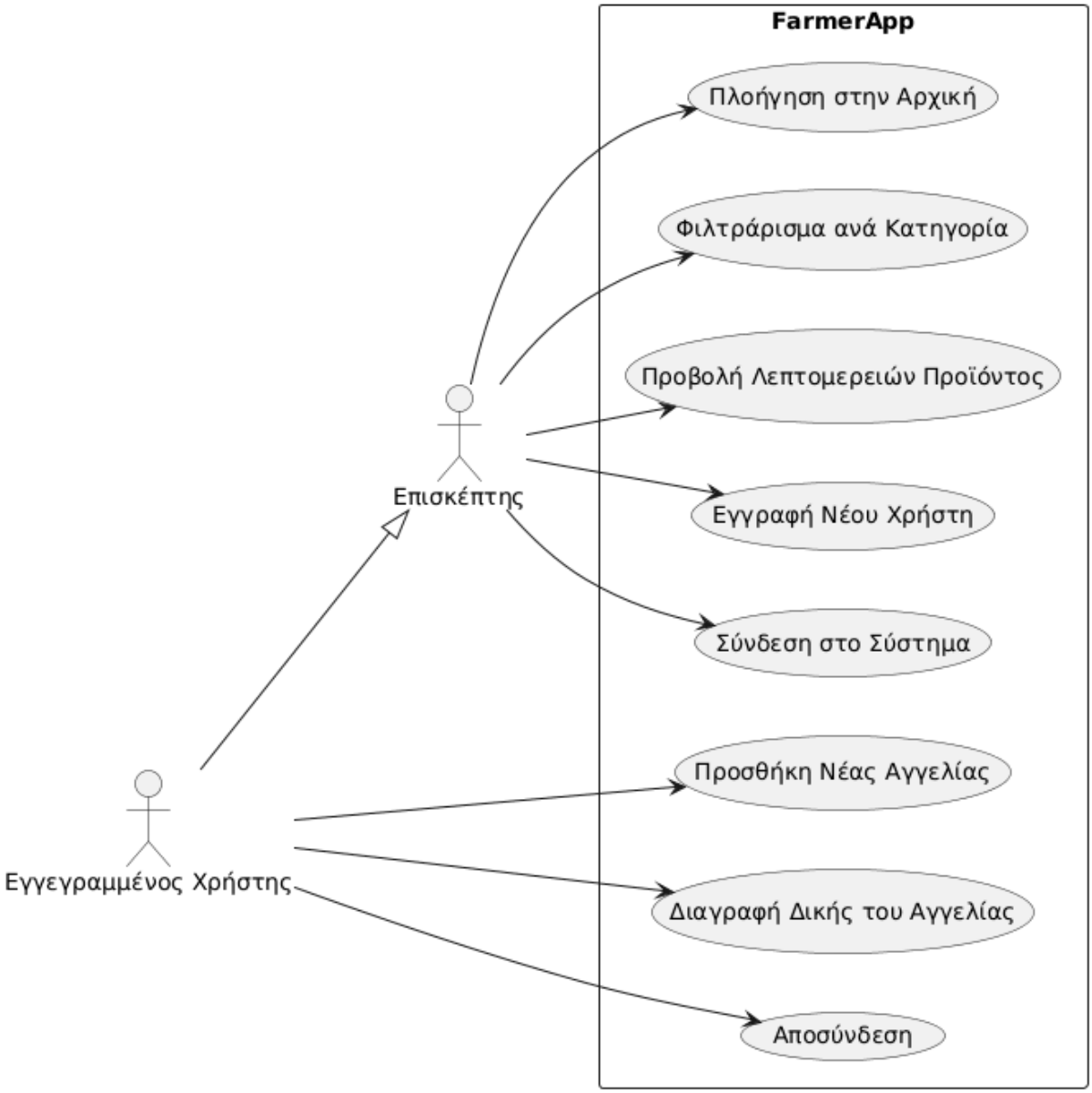
3.2 Διαγράμματα UML

Τα διαγράμματα UML αποτελούν το διεθνές πρότυπο για την οπτικοποίηση, τον προσδιορισμό και την τεκμηρίωση των στοιχείων ενός συστήματος λογισμικού. Στο πλαίσιο του **FarmerApp**, η μοντελοποίηση επικεντρώνεται σε δύο βασικούς άξονες: τις δυνατότητες που παρέχονται στους διαφορετικούς τύπους χρηστών και τη διαδοχή των ενεργειών που απαιτούνται για την ολοκλήρωση μιας λειτουργίας.

Μέσω των διαγραμμάτων αυτών, επιτυγχάνεται η αποσαφήνιση των ορίων του συστήματος και η ανάλυση των αλληλεπιδράσεων μεταξύ του χρήστη και της πλατφόρμας. Τα διαγράμματα που ακολουθούν (Use Case και Activity Diagrams) προσφέρουν μια σφαιρική εικόνα της λειτουργικότητας, διευκολύνοντας τόσο τη φάση της υλοποίησης όσο και τη μελλοντική συντήρηση ή επέκταση της εφαρμογής από άλλους προγραμματιστές.

3.2.1 Use Case Diagram: Τι μπορεί να κάνει ο Χρήστης (π.χ. Εγγραφή, Προσθήκη Προϊόντος)

Το Διάγραμμα Περιπτώσεων Χρήσης απεικονίζει τις αλληλεπιδράσεις των χρηστών (**Actors**) με το σύστημα (**FarmerApp**). Στην εφαρμογή μας διακρίνουμε δύο βασικούς ρόλους: τον **Επισκέπτη** (Guest) και τον **Εγγεγραμμένο Χρήστη** (Registered User/Seller).



Ανάλυση των Περιπτώσεων Χρήσης (Για το κείμενο της πτυχιακής)

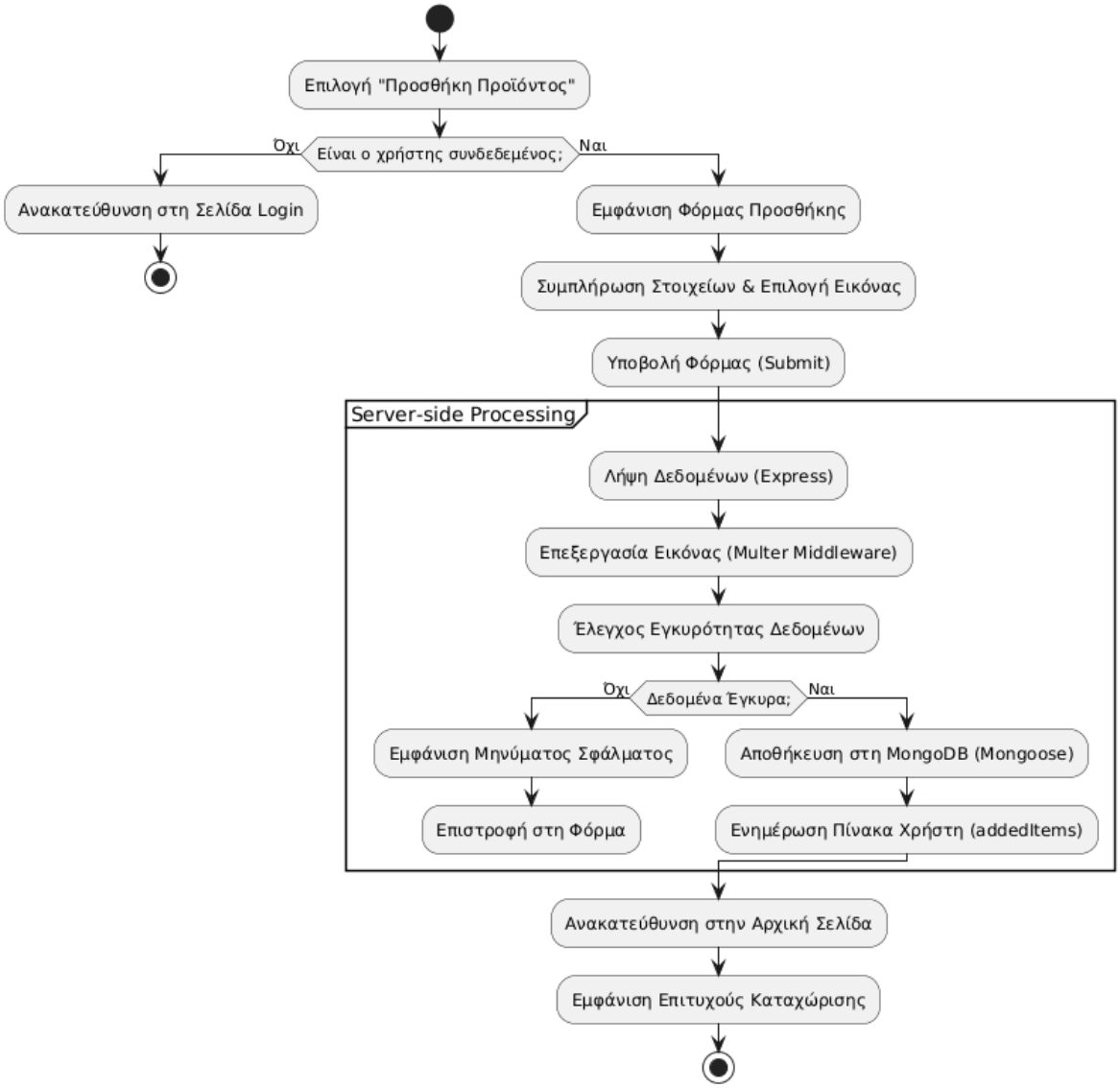
Στο διάγραμμα αυτό, οι λειτουργίες αναλύονται παραγραφικά ως εξής:

- **Ο Επισκέπτης (Guest):** Αποτελεί τον ανώνυμο χρήστη που εισέρχεται στην πλατφόρμα. Έχει το δικαίωμα να περιηγηθεί στην αρχική σελίδα, να εφαρμόσει φίλτρα κατηγοριών (π.χ. "Μηχανήματα" ή "Ζώα") και να δει τις πλήρεις λεπτομέρειες κάθε αγγελίας, συμπεριλαμβανομένων των στοιχείων επικοινωνίας του πωλητή. Επίσης, μπορεί να εκκινήσει τη διαδικασία εγγραφής ή σύνδεσης.
- **Ο Εγγεγραμμένος Χρήστης (User):** Κληρονομεί όλες τις δυνατότητες του επισκέπτη, αλλά διαθέτει επιπλέον προνόμια που απαιτούν αυθεντικοποίηση. Μετά την επιτυχή σύνδεση (Session-based login), αποκτά πρόσβαση στη φόρμα "Προσθήκη Προϊόντος", όπου μπορεί να ανεβάσει περιγραφές και φωτογραφίες.

- **Αποσύνδεση (Logout):** Μια απαραίτητη λειτουργία που καταστρέφει το session του χρήστη, προστατεύοντας τον λογαριασμό του από μη εξουσιοδοτημένη πρόσβαση.

3.2.2 Activity Diagram: Η ροή της διαδικασίας αγοροπωλησίας

Το Διάγραμμα Δραστηριότητας αποτυπώνει τη ροή εργασίας (workflow) από τη στιγμή που ο χρήστης επιθυμεί να καταχωρίσει ένα προϊόν μέχρι την οριστική αποθήκευση της αγγελίας στη βάση δεδομένων. Αναδεικνύει τα σημεία απόφασης (π.χ. αν ο χρήστης είναι συνδεδεμένος ή αν η φόρμα είναι έγκυρη) και τις ενέργειες που εκτελούνται παράλληλα ή διαδοχικά.



Ανάλυση της Ροής Δραστηριοτήτων (Για το κείμενο της πτυχιακής)

Η διαδικασία που περιγράφεται στο διάγραμμα αναλύεται παραγραφικά ως εξής:

- **Έλεγχος Αυθεντικοποίησης:** Η ροή ξεκινά με μια πύλη απόφασης (decision node) που ελέγχει την κατάσταση του session του χρήστη. Εάν ο χρήστης δεν είναι συνδεδεμένος, το σύστημα διακόπτει τη διαδικασία και τον προτρέπει να ταυτοποιηθεί, διασφαλίζοντας την ασφάλεια της πλατφόρμας.
- **Αλληλεπίδραση με τη Φόρμα:** Μετά την είσοδο, ο χρήστης αλληλεπιδρά με το UI (View), συμπληρώνοντας τα πεδία και επιλέγοντας το αρχείο της εικόνας. Η ενέργεια αυτή καταλήγει σε ένα αίτημα POST προς τον διακομιστή.
- **Επεξεργασία στο Back-end:** Στο επίπεδο του διακομιστή, εκτελούνται ταυτόχρονα δύο διεργασίες: η λήψη των κειμενικών δεδομένων και η επεξεργασία του αρχείου μέσω του middleware **Multer**, το οποίο μετονομάζει και αποθηκεύει την εικόνα στον φάκελο public/uploads.
- **Επικύρωση και Αποθήκευση:** Το σύστημα πραγματοποιεί έναν τελικό έλεγχο εγκυρότητας. Εφόσον τα δεδομένα πληρούν τις προδιαγραφές του Schema, το Model επικοινωνεί με τη MongoDB για τη μόνιμη αποθήκευση. Παράλληλα, ενημερώνεται το προφίλ του χρήστη, συνδέοντας το ID του νέου προϊόντος με τον λογαριασμό του.
- **Ολοκλήρωση:** Η διαδικασία τερματίζεται με την ανακατεύθυνση (redirect) του χρήστη, προσφέροντάς του οπτική επιβεβαίωση ότι η αγγελία του είναι πλέον ενεργή και ορατή στην κοινότητα.

3.3 Σχεδιασμός Βάσης Δεδομένων: Περιγραφή των Schemas (User.js, Product.js)

Ο σχεδιασμός της βάσης δεδομένων στο **FarmerApp** βασίζεται στην έγγραφο-κεντρική (document-oriented) προσέγγιση της MongoDB. Για την οργάνωση των δεδομένων χρησιμοποιήθηκε η βιβλιοθήκη **Mongoose (ODM)**, η οποία επιτρέπει τον ορισμό αυστηρών σχημάτων (**Schemas**) πάνω από την εγγενώς ευέλικτη δομή της MongoDB. Αυτό διασφαλίζει ότι κάθε έγγραφο που αποθηκεύεται ακολουθεί συγκεκριμένους κανόνες και τύπους δεδομένων.

Η αρχιτεκτονική της βάσης χωρίζεται σε δύο βασικές συλλογές (collections), οι οποίες περιγράφονται αναλυτικά παρακάτω:

3.3.1 Το Σχήμα Χρήστη (User Schema)

Το αρχείο User.js καθορίζει την ταυτότητα των μελών της πλατφόρμας. Τα κύρια χαρακτηριστικά του περιλαμβάνουν:

- **Ταυτοποίηση**: Πεδία όπως το name, το email (το οποίο ορίζεται ως unique για την αποφυγή διπλότυπων λογαριασμών) και το password.

- **Προφίλ**: Πληροφορίες όπως profilePic, contact και γεωγραφικά στοιχεία (state, district, village) για την ενίσχυση της τοπικότητας των συναλλαγών.
- **Συσχετίσεις (References)**: Το σχήμα περιλαμβάνει τα πεδία addedItems και cart, τα οποία είναι πίνακες από ObjectIds που παραπέμπουν (ref) στη συλλογή των Προϊόντων. Αυτή η προσέγγιση επιτρέπει τη σύνδεση των χρηστών με τις αγγελίες τους χωρίς να διπλασιάζεται η πληροφορία στη βάση.

3.3.2 Το Σχήμα Προϊόντος (Product Schema)

Το αρχείο Product.js είναι ο πυρήνας της πληροφορίας του marketplace. Η δομή του σχεδιάστηκε για να καλύπτει ένα ευρύ φάσμα αγροτικών αναγκών:

- **Βασικά Στοιχεία**: Πεδία για title, description, price και category. Η κατηγορία χρησιμοποιεί το γνώρισμα enum, περιορίζοντας τις επιλογές σε συγκεκριμένες τιμές (π.χ. machine, crop, animal), διασφαλίζοντας την ορθή λειτουργία των φίλτρων αναζήτησης.
- **Μετρήσιμα Μεγέθη**: Πεδία όπως quantity και measureUnit επιτρέπουν την ακριβή περιγραφή της προσφοράς (π.χ. "10 τόννοι σιτάρι").
- **Ενσωματωμένη Πληροφορία Πωλητή (Denormalization)**: Στο πεδίο seller, αποθηκεύεται το ID του χρήστη αλλά και βασικές πληροφορίες όπως το όνομα και το τηλέφωνο επικοινωνίας. Αυτή η τεχνική "αποκανονικοποίησης" (denormalization) επιλέχθηκε για να επιταχύνει την προβολή των αγγελιών, καθώς η εφαρμογή δεν χρειάζεται να κάνει επιπλέον αναζήτηση στη βάση για να βρει τα στοιχεία επικοινωνίας του πωλητή κατά την προβολή του προϊόντος.

3.4 Διαχείριση Αρχείων: Η λειτουργία του Multer middleware

Η διαχείριση των αρχείων στην εφαρμογή **FarmerApp** υλοποιείται μέσω του middleware **Multer**, το οποίο είναι ένα πακέτο του Node.js ειδικά σχεδιασμένο για τη διαχείριση δεδομένων μορφής multipart/form-data. Η χρήση του είναι απαραίτητη για τη μεταφόρτωση (upload) εικόνων, καθώς τα τυπικά σώματα αιτημάτων HTTP (request bodies) δεν μπορούν να επεξεργαστούν αρχεία χωρίς την κατάλληλη ενδιάμεση επεξεργασία.

Η διαδικασία διαχείρισης αρχείων αναλύεται παραγραφικά στους εξής άξονες:

- **Ρύθμιση Αποθήκευσης (Storage Configuration):** Στο αρχείο multer.js, χρησιμοποιείται η μέθοδος diskStorage για τον πλήρη έλεγχο της αποθήκευσης. Ορίζεται ο προορισμός (destination), ο οποίος είναι ο φάκελος public/uploads, και ο τρόπος ονομασίας του αρχείου (filename). Για την αποφυγή διενέξεων (file conflicts), κάθε αρχείο μετονομάζεται αυτόματα χρησιμοποιώντας τη χρονική σήμανση της στιγμής του upload (Date.now()) σε συνδυασμό με την αρχική του επέκταση.
- **Αυτοματοποιημένη Διαχείριση Φακέλων:** Για τη διασφάλιση της σταθερότητας του συστήματος, ο κώδικας περιλαμβάνει έλεγχο ύπαρξης του καταλόγου αποθήκευσης μέσω της βιβλιοθήκης fs (file system). Εάν ο φάκελος uploads δεν υπάρχει, η εφαρμογή τον δημιουργεί αυτόματα κατά την εκκίνηση, αποτρέποντας πιθανά σφάλματα κατά την πρώτη προσπάθεια μεταφόρτωσης.

- **Ενσωμάτωση στη Ροή Δεδομένων:** Το Multer λειτουργεί ως "φίλτρο" πριν το αίτημα φτάσει στον τελικό ελεγκτή (Route Controller). Όταν ένας χρήστης υποβάλλει μια αγγελία ή αλλάζει τη φωτογραφία προφίλ του, το Multer αναλαμβάνει να αποθηκεύσει το αρχείο στον δίσκο και στη συνέχεια "περνάει" το όνομα του αρχείου στο αντικείμενο req.file. Με αυτόν τον τρόπο, ο Controller μπορεί να αποθηκεύσει στη MongoDB μόνο το σχετικό μονοπάτι (path) της εικόνας ως κείμενο, διατηρώντας τη βάση δεδομένων ελαφριά.
- **Ασφάλεια και Προστασία:** Η χρήση του Multer επιτρέπει τον έλεγχο των εισερχόμενων αρχείων. Παρόλο που στην παρούσα φάση η εφαρμογή δέχεται βασικούς τύπους εικόνων, η αρχιτεκτονική επιτρέπει την εύκολη προσθήκη φίλτρων (file filters) για τον περιορισμό του μεγέθους των αρχείων ή του τύπου τους (π.χ. μόνο .jpg, .png), προστατεύοντας τον διακομιστή από κακόβουλα αρχεία ή υπερφόρτωση του αποθηκευτικού χώρου.

Η επιλογή της τοπικής αποθήκευσης αρχείων στο φάκελο public της εφαρμογής καθιστά την πρόσβαση σε αυτά εξαιρετικά γρήγορη, καθώς εξυπηρετούνται ως στατικά αρχεία (static assets) απευθείας από τον Express.js. Αυτή η προσέγγιση είναι ιδανική για το μέγεθος και τις ανάγκες της παρούσας πτυχιακής εργασίας, προσφέροντας μια αξιόπιστη και εύκολα υλοποιήσιμη λύση διαχείρισης περιεχομένου.

Κεφάλαιο 4ο: Εγχειρίδιο Χρήσης & Περιγραφή Εφαρμογής

Μετά την ανάλυση της αρχιτεκτονικής και του σχεδιασμού των δεδομένων, το παρόν κεφάλαιο εστιάζει στην πρακτική υπόσταση της εφαρμογής **FarmerApp**. Σκοπός του είναι να προσφέρει μια πλήρη εικόνα του λειτουργικού συστήματος, λειτουργώντας ταυτόχρονα ως οδηγός πλοήγησης για τον τελικό χρήστη και ως τεχνικό εγχειρίδιο για τον προγραμματιστή.

Η παρουσίαση ξεκινά με την περιγραφή του περιβάλλοντος ανάπτυξης και των εργαλείων που χρησιμοποιήθηκαν για τη συγγραφή και τον έλεγχο του κώδικα. Στη συνέχεια, ακολουθεί μια λεπτομερής περιήγηση στις κύριες λειτουργίες της πλατφόρμας —από την αρχική σελίδα και το σύστημα φιλτραρίσματος έως τις διαδικασίες εγγραφής και ανάρτησης αγγελιών— αναδεικνύοντας τον τρόπο με τον οποίο το UI (User Interface) αλληλεπιδρά με το Back-end. Τέλος, παρατίθενται οι απαραίτητες οδηγίες εγκατάστασης και παραμετροποίησης, ώστε η εφαρμογή να μπορεί να μεταφερθεί και να λειτουργήσει σε οποιοδήποτε τοπικό περιβάλλον ανάπτυξης. Η ενότητα αυτή αποτελεί την απόδειξη της ορθής υλοποίησης των στόχων που τέθηκαν στην εισαγωγή της παρούσας εργασίας.

4.1 Περιβάλλον Ανάπτυξης: (VS Code, MongoDB Atlas, Postman)

Για την υλοποίηση της εφαρμογής χρησιμοποιήθηκε ένα σύγχρονο και ευρέως διαδεδομένο σύνολο εργαλείων (Development Tools), τα οποία προσφέρουν σταθερότητα και υψηλή παραγωγικότητα. Το περιβάλλον ανάπτυξης δεν περιορίστηκε μόνο στη συγγραφή του κώδικα, αλλά επεκτάθηκε στη διαχείριση της βάσης δεδομένων στο υπολογιστικό νέφος (Cloud) και στον εξονυχιστικό έλεγχο των διαδρομών (routes) της εφαρμογής.

Τα βασικά εργαλεία που χρησιμοποιήθηκαν αναλύονται παραγραφικά ως εξής:

- **Visual Studio Code (VS Code):** Αποτέλεσε τον κύριο επεξεργαστή κώδικα (Code Editor). Επιλέχθηκε λόγω της ελαφριάς του φύσης και της πληθώρας επεκτάσεων (extensions) που διαθέτει για το οικοσύστημα της JavaScript και του Node.js. Εργαλεία όπως το ενσωματωμένο τερματικό (terminal), η αυτόματη συμπλήρωση κώδικα (IntelliSense) και ο έλεγχος σφαλμάτων σε πραγματικό χρόνο, διευκόλυναν σημαντικά τη δόμηση της αρχιτεκτονικής MVC και τη διαχείριση των αρχείων του project.
- **MongoDB Atlas:** Για τη φιλοξενία της βάσης δεδομένων επιλέχθηκε η υπηρεσία MongoDB Atlas, η οποία αποτελεί μια πλήρως διαχειριζόμενη Cloud λύση (Database as a Service - DBaaS). Μέσω του Atlas, η βάση δεδομένων του FarmerApp είναι προσβάσιμη από οποιοδήποτε περιβάλλον, προσφέροντας παράλληλα εργαλεία παρακολούθησης (monitoring), αυτόματα αντίγραφα ασφαλείας και υψηλή κλιμακωτότητα. Η σύνδεση της εφαρμογής με το Cloud cluster έγινε μέσω του URI σύνδεσης που αποθηκεύτηκε με ασφάλεια στο αρχείο περιβάλλοντος .env.
- **Postman:** Χρησιμοποιήθηκε ως το βασικό εργαλείο για τον έλεγχο και την τεκμηρίωση των API endpoints της εφαρμογής. Πριν από την ενσωμάτωση της λογικής στα EJS views, το Postman επέτρεψε την αποστολή αιτημάτων HTTP (GET,

POST, DELETE) προς τον διακομιστή, επαληθεύοντας ότι οι διαδρομές (routes) επιστρέφουν τα σωστά δεδομένα και ότι η επικοινωνία με τη MongoDB λειτουργεί απρόσκοπτα.

- **Node Package Manager (NPM):** Αποτέλεσε το εργαλείο διαχείρισης των εξωτερικών βιβλιοθηκών. Μέσω του αρχείου package.json, το NPM επέτρεψε την εγκατάσταση και τον συντονισμό όλων των απαραίτητων πακέτων (Express, Mongoose, Multer, Bcrypt, κ.α.), διασφαλίζοντας ότι όλες οι εξαρτήσεις (dependencies) της εφαρμογής είναι ενημερωμένες και συμβατές μεταξύ τους.

Η συνδυαστική χρήση αυτών των εργαλείων δημιούργησε ένα στιβαρό περιβάλλον ανάπτυξης, επιτρέποντας τη γρήγορη μετάβαση από το στάδιο του σχεδιασμού στην πλήρη λειτουργική υλοποίηση, διατηρώντας παράλληλα τον κώδικα οργανωμένο και απαλλαγμένο από σφάλματα.

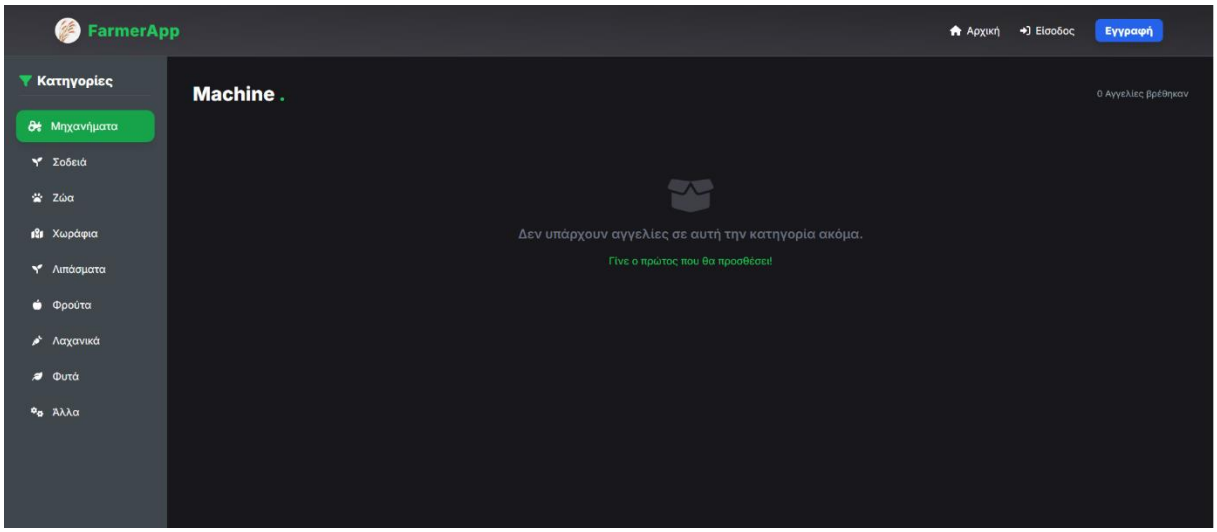
4.2 Παρουσίαση Λειτουργιών

Η παρουσίαση των λειτουργιών αποτελεί το επιστέγασμα της αναπτυξιακής διαδικασίας, καθώς επιδεικνύει τον τρόπο με τον οποίο οι απαιτήσεις των χρηστών μετατράπηκαν σε διαδραστικές σελίδες και αυτοματοποιημένες διεργασίες. Η εφαρμογή έχει σχεδιαστεί με γνώμονα την **ευχρηστία (Usability)** και την **αμεσότητα**, επιτρέποντας στον αγρότη να πλοηγείται σε έναν όγκο πληροφοριών χωρίς περιττή πολυπλοκότητα.

Στις υποενότητες που ακολουθούν, αναλύονται τα κύρια τμήματα του FarmerApp, συνοδευόμενα από στιγμιότυπα οθόνης (screenshots) που αποτυπώνουν την τελική μορφή της πλατφόρμας. Η ανάλυση καλύπτει τη ροή εργασίας από την πρώτη επαφή του επισκέπτη με την αρχική σελίδα, τη διαδικασία ταυτοποίησης μέσω του συστήματος εγγραφής, έως και την ολοκληρωμένη διαχείριση μιας αγγελίας. Μέσα από αυτή την περιήγηση, γίνεται φανερή η συνέργεια μεταξύ του **Tailwind CSS** (για την αισθητική αρτιότητα) και του **Node.js/Express** (για τη λειτουργική υποστήριξη), προσφέροντας μια συνολική εικόνα της δυναμικής φύσης της εφαρμογής.

4.2.1 Αρχική Σελίδα

Η Αρχική Σελίδα του **FarmerApp** αποτελεί το κεντρικό σημείο αλληλεπίδρασης του χρήστη με το περιεχόμενο της πλατφόρμας. Σχεδιάστηκε με γνώμονα τη μινιμαλιστική αισθητική και την άμεση πρόσβαση στην πληροφορία, χρησιμοποιώντας ένα σκοτεινό θέμα (Dark Mode) που προσδίδει σύγχρονο χαρακτήρα και ξεκουράζει τον χρήστη κατά την περιήγηση.



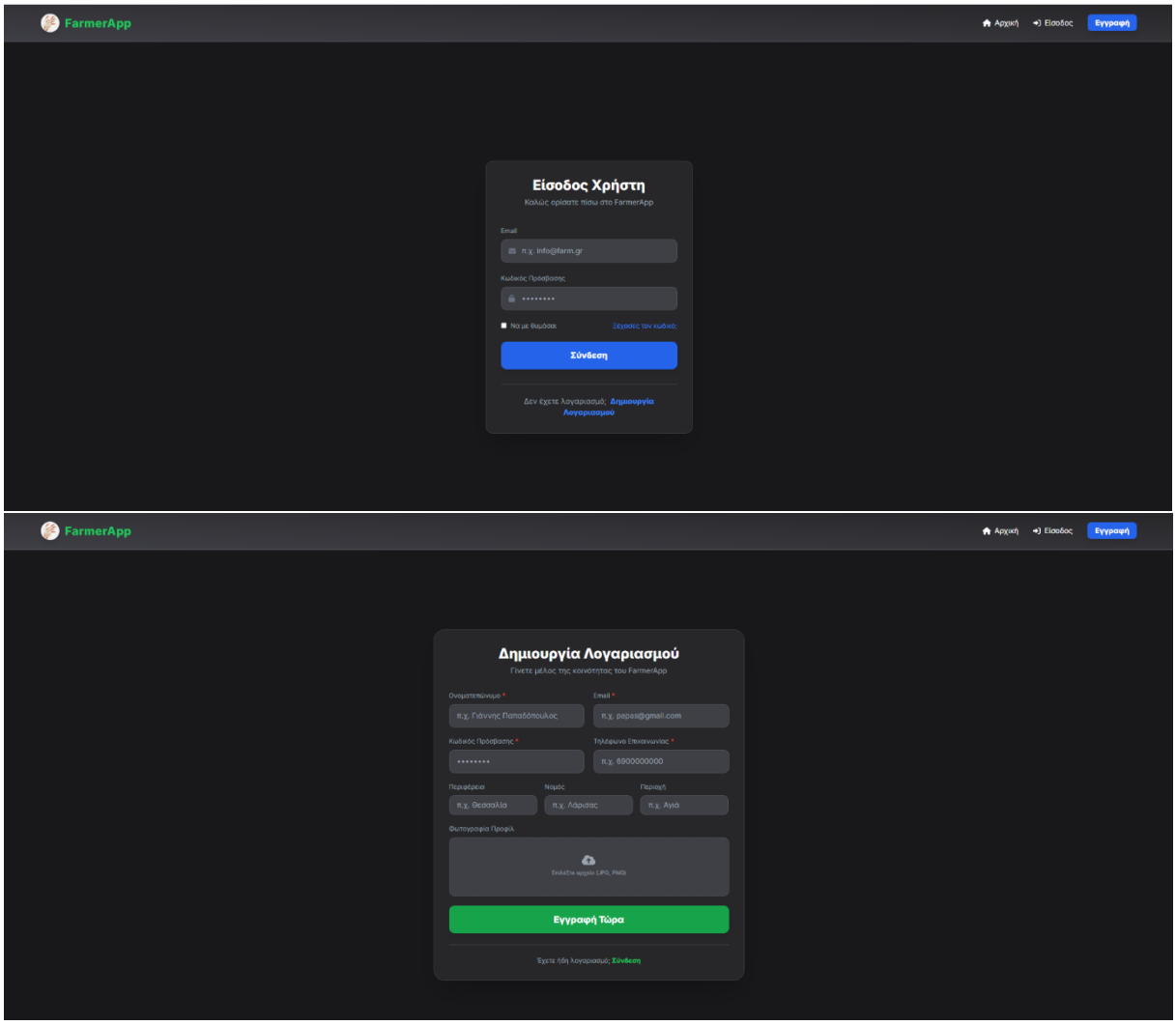
Σχολιασμός Λειτουργικών Στοιχείων:

- **Κεντρική Μπάρα Πλοήγησης (Navbar):** Στο επάνω μέρος της σελίδας βρίσκεται η μπάρα πλοήγησης, η οποία περιλαμβάνει το λογότυπο της εφαρμογής και τα βασικά κουμπιά ελέγχου. Η χρήση έντονου μπλε χρώματος για το κουμπί «Εγγραφή» λειτουργεί ως **Call-to-Action (CTA)**, κατευθύνοντας τους νέους χρήστες στη δημιουργία λογαριασμού.
- **Πλευρικό Μενού Κατηγοριών (Sidebar):** Στην αριστερή πλευρά ενσωματώνεται ένα κάθετο μενού κατηγοριών. Κάθε κατηγορία (π.χ. Μηχανήματα, Ζώα, Σοδειά) συνοδεύεται από ένα αντιπροσωπευτικό εικονίδιο της βιβλιοθήκης *Font Awesome*, διευκολύνοντας την οπτική αναγνώριση. Η επιλεγμένη κατηγορία («Μηχανήματα» στο στιγμιότυπο) επισημαίνεται με πράσινο χρώμα, προσφέροντας στον χρήστη άμεση πληροφόρηση για το πού βρίσκεται (Visual Feedback).
- **Περιοχή Προβολής Περιεχομένου:** Το κεντρικό τμήμα της σελίδας καταλαμβάνει ο χώρος προβολής των αγγελιών. Σε περίπτωση που μια κατηγορία είναι κενή, η εφαρμογή εμφανίζει ένα φιλικό μήνυμα («Δεν υπάρχουν αγγελίες...») μαζί με ένα link παρότρυνσης για προσθήκη νέας αγγελίας. Αυτό εξασφαλίζει ότι ο χρήστης δεν θα βρεθεί ποτέ σε μια "νεκρή" σελίδα χωρίς οδηγίες.
- **Δυναμικό Φιλτράρισμα:** Η εναλλαγή μεταξύ των κατηγοριών γίνεται μέσω παραμέτρων στο URL (π.χ. /?category=machine), οι οποίες επεξεργάζονται στον διακομιστή (Server-side) για την ταχύτατη ανάκτηση των αντίστοιχων δεδομένων από τη MongoDB.

Η χρήση του **Tailwind CSS** επέτρεψε την κατασκευή μιας διεπαφής που είναι πλήρως αποκρινόμενη (responsive). Όπως φαίνεται στο στιγμιότυπο, η διάταξη των στοιχείων διατηρεί την ισορροπία της, εξασφαλίζοντας ότι η εμπειρία πλοήγησης παραμένει σταθερή ανεξάρτητα από τη συσκευή πρόσβασης.

4.2.2 Σύστημα Εγγραφής και Αυθεντικοποίησης (Bcrypt hashing)

Το σύστημα αυθεντικοποίησης αποτελεί την πύλη εισόδου των χρηστών στο **FarmerApp** και έχει σχεδιαστεί με κύριο άξονα την ασφάλεια των προσωπικών δεδομένων και την ευκολία χρήσης. Η διαδικασία χωρίζεται σε δύο βασικές φάσεις: τη δημιουργία λογαριασμού (Register) και την ταυτοποίηση (Login).



Σχολιασμός Λειτουργικών Στοιχείων:

- **Φόρμα Εγγραφής (Register):** Η σελίδα εγγραφής συλλέγει όλα τα απαραίτητα στοιχεία για τη δημιουργία ενός πλήρους αγροτικού προφίλ. Πέρα από τα βασικά (Όνομα, Email, Κωδικός), περιλαμβάνει πεδία για γεωγραφικό προσδιορισμό (Περιφέρεια, Νομός, Περιοχή), γεγονός που επιτρέπει στο σύστημα να κατηγοριοποιεί τις αγγελίες τοπικά. Η χρήση "placeholders" (π.χ. π.χ. Θεσσαλία) καθοδηγεί τον χρήστη στη σωστή συμπλήρωση των πεδίων.
- **Μεταφόρτωση Φωτογραφίας Προφίλ:** Στο κάτω μέρος της φόρμας εγγραφής, ενσωματώνεται ένα διακριτό πεδίο "Drag & Drop" για τη φωτογραφία προφίλ. Η λειτουργία αυτή υποστηρίζεται από το **Multer middleware** που αναλύθηκε στο

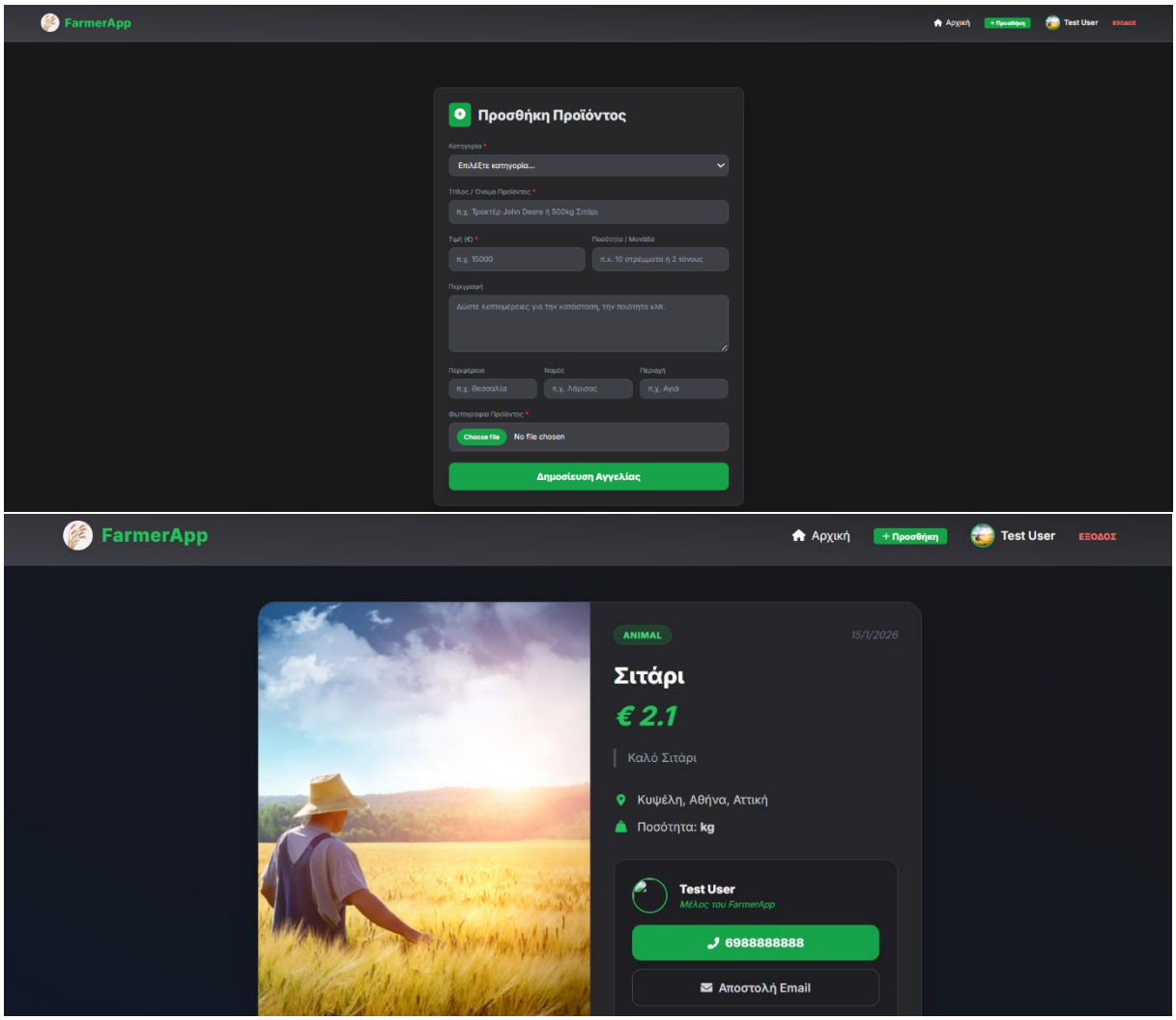
Κεφάλαιο 3, επιτρέποντας στους χρήστες να εξατομικεύσουν την παρουσία τους στην πλατφόρμα.

- **Ασφάλεια μέσω Bcrypt Hashing:** Το πιο κρίσιμο στοιχείο που δεν είναι ορατό στο UI αλλά εκτελείται στο Back-end είναι η κρυπτογράφηση. Οι κωδικοί πρόσβασης δεν αποθηκεύονται ποτέ σε μορφή απλού κειμένου στη MongoDB. Χρησιμοποιείται η βιβλιοθήκη **Bcrypt**, η οποία εφαρμόζει έναν αλγόριθμο κατακερματισμού (hashing) με την προσθήκη "salt". Αυτό διασφαλίζει ότι, ακόμη και σε περίπτωση διαρροής της βάσης δεδομένων, οι πραγματικοί κωδικοί των χρηστών παραμένουν μη αναγνώσιμοι.
- **Σελίδα Εισόδου (Login):** Η σελίδα σύνδεσης ακολουθεί την ίδια αισθητική, προσφέροντας μια απλή διεπαφή με δύο πεδία. Το σύστημα χρησιμοποιεί **Sessions** για να διατηρεί τον χρήστη συνδεδεμένο κατά την πλοήγησή του, επιτρέποντάς του να προσθέτει αγγελίες χωρίς να χρειάζεται συνεχή ταυτοποίηση.
- **Validation & Feedback:** Και οι δύο φόρμες διαθέτουν ελέγχους εγκυρότητας. Εάν ένας χρήστης προσπαθήσει να εγγραφεί με email που ήδη υπάρχει ή δώσει λάθος κωδικό, ο διακομιστής επιστρέφει κατάλληλα μηνύματα σφάλματος, καθοδηγώντας τον χρήστη στην ορθή ενέργεια.

Η υλοποίηση αυτή εγγυάται ότι το FarmerApp τηρεί τις βασικές αρχές ασφάλειας ιστού, προστατεύοντας την κοινότητα των αγροτών από κακόβουλες ενέργειες και διασφαλίζοντας την ιδιωτικότητα των επικοινωνιών τους.

4.2.3 Φόρμα Προσθήκης και Προβολή Προϊόντος

Η καρδιά του **FarmerApp** χτυπά στη δυνατότητα των χρηστών να αλληλεπιδρούν με το περιεχόμενο, είτε δημιουργώντας νέες αγγελίες είτε αναζητώντας λεπτομέρειες για υπάρχοντα αγροτικά προϊόντα. Η σχεδίαση αυτών των σελίδων ακολουθεί τη λογική της "εστιασμένης προσοχής", όπου ο χρήστης βλέπει μόνο τις απαραίτητες πληροφορίες χωρίς περισπασμούς.



Σχολιασμός Λειτουργικών Στοιχείων:

- **Φόρμα Προσθήκης Προϊόντος (addProduct.png):** Η σελίδα αυτή είναι προσβάσιμη μόνο σε συνδεδεμένους χρήστες, όπως υποδηλώνει η παρουσία του ονόματος "Test User" και του κουμπιού "Έξοδος" στο Navbar. Η φόρμα περιλαμβάνει έξυπνα πεδία εισαγωγής:
    - **Dropdown Κατηγορίας:** Διασφαλίζει ότι το προϊόν θα ταξινομηθεί σωστά στη βάση δεδομένων.
    - **Τιμή και Ποσότητα:** Διακριτά πεδία για την οικονομική και ποσοτική περιγραφή, επιτρέποντας στο σύστημα να εμφανίζει το σύμβολο του ευρώ (€) αυτόματα.
    - **Τοπικότητα:** Η προσθήκη Περιφέρειας, Νομού και Περιοχής ενισχύει τον τοπικό χαρακτήρα των συναλλαγών, επιτρέποντας σε έναν αγοραστή να γνωρίζει την ακριβή προέλευση του προϊόντος.
    - **Upload Φωτογραφίας:** Το κουμπί "Choose file" ενεργοποιεί το Multer middleware, το οποίο επεξεργάζεται την εικόνα πριν την αποθήκευση.

- **Σελίδα Προβολής Προϊόντος (viewProduct.png):** Μόλις το προϊόν δημοσιευθεί, ο χρήστης οδηγείται στη δυναμική σελίδα προβολής. Εδώ παρατηρούμε:
    - **Οπτική Κυριαρχία:** Η φωτογραφία του προϊόντος καταλαμβάνει το αριστερό τμήμα, προσφέροντας άμεση οπτική επαφή.
    - **Δυναμικά Δεδομένα:** Η τιμή εμφανίζεται με έντονο πράσινο χρώμα και μεγάλο μέγεθος για έμφαση. Η ημερομηνία (π.χ. 15/1/2026) παράγεται αυτόματα από το πεδίο createdAt του Schema.
    - **Στοιχεία Επικοινωνίας:** Η κάρτα του πωλητή ("Test User") περιλαμβάνει άμεσα προσβάσιμα κουμπιά για τηλεφωνική κλήση και αποστολή email. Αυτό μειώνει τα βήματα που χρειάζεται να κάνει ένας ενδιαφερόμενος αγοραστής, αυξάνοντας τις πιθανότητες επιτυχούς συναλλαγής.
    - **Badge Κατηγορίας:** Η ετικέτα "ANIMAL" στο πάνω μέρος λειτουργεί ως οπτική επιβεβαίωση της ταξινόμησης.

Η συνέργεια αυτών των δύο σελίδων επιτρέπει στο FarmerApp να λειτουργεί ως ένα πλήρες marketplace. Η πληροφορία ρέει απρόσκοπτα από τη φόρμα εισαγωγής στη βάση δεδομένων και από εκεί στην τελική "κάρτα" προϊόντος, διατηρώντας την αισθητική ομοιομορφία του dark mode και την ταχύτητα απόκρισης του **EJS rendering**.

4.3 Οδηγίες Εγκατάστασης: (npm install, setup .env, npm start)

Η εγκατάσταση και η σωστή παραμετροποίηση του **FarmerApp** είναι μια απλή αλλά κρίσιμη διαδικασία, χάρη στη χρήση του **NPM (Node Package Manager)**. Ακολουθώντας τα παρακάτω βήματα, η εφαρμογή μπορεί να τεθεί σε λειτουργία σε οποιοδήποτε τοπικό περιβάλλον (Localhost).

Βήμα 1: Εγκατάσταση Εξαρτήσεων (npm install) Αφού γίνει λήψη του πηγαίου κώδικα, ο χρήστης πρέπει να ανοίξει το τερματικό στον φάκελο του project και να εκτελέσει την εντολή: npm install Η εντολή αυτή διαβάζει το αρχείο package.json και κατεβάζει αυτόματα όλες τις απαραίτητες βιβλιοθήκες (Express, Mongoose, Multer, Bcrypt κ.λπ.) στον φάκελο node_modules.

Βήμα 2: Ρύθμιση Μεταβλητών Περιβάλλοντος (.env) Για λόγους ασφαλείας, ευαίσθητα δεδομένα όπως το **URL σύνδεσης της MongoDB Atlas** και το **Session Secret** δεν περιλαμβάνονται στον κώδικα. Ο χρήστης πρέπει να δημιουργήσει ένα αρχείο με το όνομα .env στη ρίζα του project και να ορίσει τις απαραίτητες τιμές, όπως: MONGO_URI=mongodb+srv://your_username:password@cluster... PORT=3000

Βήμα 3: Εκκίνηση της Εφαρμογής (node index.js) Μετά την ολοκλήρωση των παραπάνω, η εφαρμογή τίθεται σε λειτουργία με την εντολή: node index.js (Εναλλακτικά, αν έχει ρυθμιστεί το script στο package.json, μπορεί να χρησιμοποιηθεί η εντολή npm start).

Μόλις εμφανιστεί το μήνυμα επιτυχούς σύνδεσης στο τερματικό, το FarmerApp είναι προσβάσιμο μέσω του περιηγητή στη διεύθυνση http://localhost:3000.

Κεφάλαιο 5ο: Συμπεράσματα & Μελλοντικές Επεκτάσεις

Με την ολοκλήρωση της παρουσίασης του **FarmerApp**, καθίσταται σαφές ότι η δημιουργία μιας σύγχρονης διαδικτυακής πλατφόρμας αγγελιών απαιτεί έναν συνδυασμό τεχνικής κατάρτισης, στρατηγικής επιλογής εργαλείων και ανθρωποκεντρικού σχεδιασμού. Στο τελευταίο αυτό κεφάλαιο, πραγματοποιείται μια συνολική ανασκόπηση του έργου, καταγράφοντας τα στάδια της υλοποίησης και τις τεχνικές προκλήσεις που αντιμετωπίστηκαν.

Παράλληλα, αναλύονται τα συμπεράσματα που προέκυψαν από τη χρήση του **MEN stack**, οι προσωπικές γνώσεις που αποκτήθηκαν κατά τη διάρκεια της ανάπτυξης και οι τρέχοντες περιορισμοί του συστήματος. Η ενότητα κλείνει με μια σειρά από προτάσεις για μελλοντικές επεκτάσεις, οι οποίες θα μπορούσαν να μετατρέψουν το FarmerApp από μια ακαδημαϊκή εργασία σε μια ολοκληρωμένη εμπορική πλατφόρμα, ενισχύοντας περαιτέρω την υποστήριξη του πρωτογενούς τομέα μέσω της τεχνολογίας.

5.1 Σύνοψη Υλοποίησης

Η υλοποίηση του **FarmerApp** ολοκληρώθηκε με επιτυχία, καταλήγοντας σε μια λειτουργική και σταθερή διαδικτυακή πλατφόρμα. Η πορεία της ανάπτυξης ακολούθησε μια αυστηρά δομημένη σειρά σταδίων, ξεκινώντας από την ανάλυση των αναγκών του αγροτικού τομέα και καταλήγοντας στη συγγραφή κώδικα και τον έλεγχο των λειτουργιών. Η εφαρμογή καταφέρνει να καλύψει το κενό στην άμεση επικοινωνία μεταξύ παραγωγών και αγοραστών, προσφέροντας ένα περιβάλλον όπου η πληροφορία διακινείται με διαφάνεια και ταχύτητα.

Η σύνοψη της υλοποίησης μπορεί να κατηγοριοποιηθεί στους εξής άξονες:

- **Αρχιτεκτονική Πληρωμένης Γνώσης:** Η υιοθέτηση του προτύπου **MVC** επέτρεψε την οργάνωση του κώδικα σε διακριτά τμήματα, διευκολύνοντας την επίλυση σφαλμάτων και τη σταδιακή προσθήκη λειτουργιών χωρίς να διαταραχθεί η συνολική δομή.
- **Λειτουργική Αρτιότητα:** Η εφαρμογή διαθέτει πλέον ένα πλήρες σύστημα διαχείρισης χρηστών και αγγελιών. Οι χρήστες μπορούν να εγγράφονται με ασφάλεια, να ανεβάζουν φωτογραφίες και λεπτομέρειες των προϊόντων τους και να πλοηγούνται σε κατηγοριοποιημένο περιεχόμενο, καλύπτοντας τον βασικό κύκλο ζωής ενός ηλεκτρονικού marketplace.
- **Τεχνολογική Συνοχή:** Η χρήση του **MEN stack** αποδείχθηκε ορθή επιλογή, καθώς η JavaScript χρησιμοποιήθηκε ως κοινή γλώσσα επικοινωνίας σε όλο το φάσμα της εφαρμογής. Η ενσωμάτωση της **MongoDB Atlas** εξασφάλισε ότι τα δεδομένα είναι αποθηκευμένα με ασφάλεια στο υπολογιστικό νέφος, καθιστώντας την εφαρμογή προσβάσιμη και κλιμακώσιμη.
- **Διεπαφή Χρήστη (UI):** Μέσω του **Tailwind CSS**, επιτεύχθηκε ένας σύγχρονος και αποκρινόμενος σχεδιασμός (responsive design). Το "Dark Mode" και η καθαρή ιεραρχία των στοιχείων προσφέρουν μια επαγγελματική εμπειρία χρήσης, η οποία προσαρμόζεται αυτόματα σε κινητές συσκευές και υπολογιστές.

Συνολικά, η υλοποίηση του FarmerApp απέδειξε ότι η χρήση σύγχρονων εργαλείων ανοιχτού κώδικα επιτρέπει τη δημιουργία ισχυρών ψηφιακών εργαλείων με χαμηλό κόστος αλλά υψηλή παραγωγικότητα. Το τελικό προϊόν δεν είναι απλώς ένας κώδικας, αλλά μια ολοκληρωμένη λύση που ανταποκρίνεται στις απαιτήσεις της σύγχρονης αγροτικής αγοράς.

5.2 Κέρδοι απο την Υλοποίηση

Η διαδικασία σχεδιασμού και ανάπτυξης του FarmerApp αποτέλεσε μια σημαντική μαθησιακή εμπειρία, καθώς επέτρεψε τη μετάβαση από τη θεωρητική γνώση στην πρακτική εφαρμογή σύνθετων εννοιών του σύγχρονου προγραμματισμού. Μέσα από την επίλυση πραγματικών προκλήσεων, αποκτήθηκαν δεξιότητες που αποτελούν τα θεμέλια για κάθε επαγγελματία Full-stack προγραμματιστή.

Τα κύρια οφέλη από την υλοποίηση αναλύονται παραγραφικά:

- **Εξοικείωση με την Ασύγχρονη JavaScript (Asynchronous JS):** Μία από τις σημαντικότερες κατακτήσεις ήταν η κατανόηση του ασύγχρονου μοντέλου εκτέλεσης της JavaScript. Η χρήση των async/await και των Promises για την επικοινωνία με τη βάση δεδομένων και τη διαχείριση των αιτημάτων, επέτρεψε τη δημιουργία ενός συστήματος που δεν "παγώνει" κατά την εκτέλεση βαριών διεργασιών, όπως το ανέβασμα εικόνων. Η γνώση αυτή είναι κρίσιμη για την ανάπτυξη εφαρμογών υψηλής απόδοσης.
- **Διαχείριση Μη-Σχεσιακών Βάσεων Δεδομένων (NoSQL):** Η εργασία με τη **MongoDB** και το **Mongoose** προσέφερε μια νέα οπτική στη μοντελοποίηση δεδομένων. Σε αντίθεση με τις παραδοσιακές SQL βάσεις, η ευελιξία των εγγράφων (documents) και η ικανότητα δυναμικής αλλαγής των σχημάτων (schemas) δίδαξαν πώς να σχεδιάζονται συστήματα που μπορούν να εξελίσσονται χωρίς την ανάγκη περίπλοκων migrations.
- **Κατανόηση της Λογικής των Middlewares:** Η χρήση του **Express.js** ανέδειξε τη σημασία των Middlewares (όπως το **Multer** για αρχεία ή το **Bcrypt** για την ασφάλεια). Η ικανότητα να παρεμβάλλονται μικρά, αυτόνομα τμήματα κώδικα στη ροή ενός αιτήματος για να εκτελέσουν ελέγχους, κρυπτογράφηση ή αποθήκευση, αποτέλεσε το "κλειδί" για τη δόμηση ενός καθαρού και επεκτάσιμου κώδικα.
- **Ολιστική Προσέγγιση Full-stack:** Η ανάπτυξη ολόκληρου του stack (Database, Server, View Engine) βοήθησε στην κατανόηση του κύκλου ζωής ενός αιτήματος HTTP από την αρχή μέχρι το τέλος. Η διασύνδεση των EJS προτύπων με τα δεδομένα του διακομιστή και η εφαρμογή στυλ μέσω Tailwind CSS, πρόσφερε μια ολοκληρωμένη εικόνα του πώς οι τεχνικές επιλογές στο Back-end επηρεάζουν άμεσα την ευχρηστία στο Front-end.

Συνολικά, η υλοποίηση του FarmerApp δεν βελτίωσε μόνο τις προγραμματιστικές μου ικανότητες, αλλά ενίσχυσε και την αναλυτική μου σκέψη στην επίλυση προβλημάτων (debugging), στη διαχείριση του χρόνου ανάπτυξης και στον αρχιτεκτονικό σχεδιασμό λογισμικού.

5.3 Πλεονεκτήματα & Περιορισμοί της Εφαρμογής

Η ανάλυση των πλεονεκτημάτων και των περιορισμών επιτρέπει την κατανόηση της τρέχουσας κατάστασης του FarmerApp και θέτει τις βάσεις για την περαιτέρω εξέλιξή του. Η εφαρμογή καταφέρνει να προσφέρει μια σταθερή λύση για τις βασικές ανάγκες ενός αγροτικού marketplace, διατηρώντας παράλληλα μια ελαφριά και αποδοτική αρχιτεκτονική.

Τα κύρια χαρακτηριστικά αναλύονται παραγραφικά ως εξής:

- **Πλεονεκτήματα:**
    - **Υψηλή Απόδοση και Ταχύτητα:** Χάρη στο **Node.js** και το **Server-Side Rendering (EJS)**, η εφαρμογή ανταποκρίνεται ακαριαία στα αιτήματα των χρηστών, προσφέροντας μια απρόσκοπτη εμπειρία πλοήγησης.
    - **Ευελιξία Δεδομένων:** Η χρήση της **MongoDB** επιτρέπει την καταχώριση προϊόντων με διαφορετικά χαρακτηριστικά χωρίς περιορισμούς, καθιστώντας την πλατφόρμα ιδανική για την ποικιλομορφία του αγροτικού τομέα.
    - **Ασφάλεια και Αξιοπιστία:** Η κρυπτογράφηση των κωδικών με **Bcrypt** και η διαχείριση των συνεδριών (Sessions) διασφαλίζουν ότι τα δεδομένα των χρηστών είναι προστατευμένα από μη εξουσιοδοτημένη πρόσβαση.
    - **Σύγχρονο και Responsive UI:** Ο σχεδιασμός με **Tailwind CSS** εξασφαλίζει ότι η εφαρμογή είναι αισθητικά άρτια (Dark Mode) και λειτουργεί άψογα σε κινητά τηλέφωνα, κάτι που είναι κρίσιμο για τους αγρότες που εργάζονται στο πεδίο.
- **Περιορισμοί:**
    - **Έλλειψη Αμφίδρομης Επικοινωνίας:** Στην παρούσα φάση, η επικοινωνία αγοραστή-πωλητή περιορίζεται στα στατικά στοιχεία (τηλέφωνο, email) και δεν υπάρχει ενσωματωμένο σύστημα ζωντανής συνομιλίας (Real-time Chat).
    - **Στατική Διαχείριση Τοποθεσίας:** Η τοποθεσία των προϊόντων εισάγεται ως κείμενο και δεν υπάρχει οπτικοποίηση σε χάρτη ή αυτόματος εντοπισμός μέσω GPS, γεγονός που θα διευκόλυνε την εύρεση κοντινών αγγελιών.
    - **Απουσία Ηλεκτρονικών Πληρωμών:** Η εφαρμογή λειτουργεί ως πλατφόρμα αγγελιών και όχι ως πλήρες e-shop, καθώς δεν υποστηρίζει την ολοκλήρωση συναλλαγών μέσω πιστωτικών καρτών ή άλλων ψηφιακών πορτοφολιών.
    - **Περιορισμένη Διαχείριση Πολυμέσων:** Υποστηρίζεται η μεταφόρτωση μίας μόνο φωτογραφίας ανά αγγελία, περιορίζοντας τη δυνατότητα του πωλητή να δείξει το προϊόν από διαφορετικές οπτικές γωνίες.

Παρά τους περιορισμούς αυτούς, το FarmerApp αποτελεί μια στιβαρή βάση (MVP - Minimum Viable Product), η οποία επιτυγχάνει τον πρωταρχικό της στόχο: τη δημιουργία μιας αξιόπιστης γέφυρας επικοινωνίας στον πρωτογενή τομέα.

5.4 Προτάσεις για Μελλοντικές Επεκτάσεις

Το **FarmerApp**, στην παρούσα μορφή του, αποτελεί μια ολοκληρωμένη και λειτουργική βάση. Ωστόσο, η φύση του **MEN stack** και της **Node.js** επιτρέπει την ενσωμάτωση προηγμένων λειτουργιών που μπορούν να αναβαθμίσουν την εφαρμογή σε μια πλήρη ψηφιακή αγορά για τον πρωτογενή τομέα. Οι προτάσεις για μελλοντική ανάπτυξη εστιάζουν στην ενίσχυση της αλληλεπίδρασης, της αξιοπιστίας των συναλλαγών και της γεωγραφικής στόχευσης.

Οι κυριότερες προτάσεις για μελλοντικές επεκτάσεις αναλύονται παραγραφικά:

- **Ενσωμάτωση Συστήματος Ζωντανής Συνομιλίας (Real-time Chat):** Μια από τις σημαντικότερες προσθήκες θα ήταν η υλοποίηση ενός εσωτερικού συστήματος μηνυμάτων μεταξύ αγοραστών και πωλητών. Χρησιμοποιώντας την τεχνολογία **WebSockets** (μέσω της βιβλιοθήκης **Socket.io**), η επικοινωνία θα μπορούσε να γίνεται σε πραγματικό χρόνο, καταργώντας την ανάγκη για εξωτερικά μέσα (όπως τηλέφωνο ή email) και διατηρώντας τον χρήστη μέσα στο περιβάλλον της εφαρμογής.
- **Γεωγραφική Απεικόνιση με Google Maps API:** Η ενσωμάτωση υπηρεσιών χαρτών θα επέτρεπε την οπτικοποίηση της θέσης των προϊόντων. Οι χρήστες θα μπορούσαν να αναζητούν αγγελίες σε μια συγκεκριμένη ακτίνα από την τοποθεσία τους, διευκολύνοντας τα logistics και μειώνοντας το κόστος μεταφοράς των αγροτικών προϊόντων και μηχανημάτων.
- **Σύστημα Ηλεκτρονικών Πληρωμών (Payment Gateway):** Η μετάβαση από μια πλατφόρμα αγγελιών σε ένα πλήρες ηλεκτρονικό κατάστημα (e-shop) απαιτεί την ενσωμάτωση πυλών πληρωμής, όπως το **Stripe** ή το **PayPal**. Αυτό θα επέτρεπε την ασφαλή δέσμευση χρημάτων και την ολοκλήρωση της αγοραπωλησίας ψηφιακά, προσφέροντας εγγυήσεις και στους δύο συμβαλλόμενους.
- **Προηγμένη Διαχείριση Πολυμέσων και Πολλαπλές Εικόνες:** Η επέκταση του συστήματος ώστε να επιτρέπει τη μεταφόρτωση πολλαπλών φωτογραφιών ανά αγγελία, καθώς και η χρήση cloud υπηρεσιών αποθήκευσης (όπως το **Cloudinary** ή το **Amazon S3**), θα βελτίωνε την απόδοση της εφαρμογής και την ποιότητα της παρουσίασης των προϊόντων.
- **Σύστημα Αξιολόγησης και Κριτικών (Rating System):** Η προσθήκη ενός συστήματος αξιολόγησης πωλητών θα ενίσχυε το αίσθημα εμπιστοσύνης στην κοινότητα. Οι αγοραστές θα μπορούσαν να βαθμολογούν την ποιότητα των προϊόντων και την αξιοπιστία του παραγωγού, δημιουργώντας ένα αυτο-ρυθμιζόμενο και αξιοκρατικό περιβάλλον συναλλαγών.

Οι παραπάνω επεκτάσεις θα μπορούσαν να μετατρέψουν το FarmerApp σε ένα ισχυρό εργαλείο στα χέρια του Έλληνα αγρότη, συνδυάζοντας την ευκολία χρήσης με τις πλέον σύγχρονες τεχνολογίες του διαδικτύου.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Mozilla Developer Network (MDN). Client-server overview. Διαθέσιμο σε: https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Introduction

[2] Flanagan, D. (2020). JavaScript: The Definitive Guide: Master the World's Most-Used Programming Language. O'Reilly Media.

[3] Nixon, R. (2021). Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5. O'Reilly Media. (Σημείωση: Παρόλο που αναφέρει PHP, το κεφάλαιο για το Client-Server μοντέλο είναι κλασική αναφορά για τον διαχωρισμό των ευθυνών).

[4] Cantelon, M., Harter, M., Holowaychuk, T., & Rajlich, N. (2014). Node.js in Action. Manning Publications. (Ή εναλλακτικά: Teixeira, P. (2012). Professional Node.js: Building Fast, Scalable Network Sites. John Wiley & Sons.).

[5] Lerdorf, R., Tatroe, K., & MacIntyre, P. (2006). Programming PHP. O'Reilly Media. (Ή εναλλακτικά μια πιο σύγχρονη έκδοση: Nixon, R. (2021). Learning PHP, MySQL & JavaScript. O'Reilly Media.).

[6] Percival, H., & Elman, G. (2020). Test-Driven Development with Python: Obey the Testing Goat: Using Django, Selenium, and JavaScript. O'Reilly Media. (Ή εναλλακτικά: Grinberg, M. (2018). Flask Web Development: Developing Web Applications with Python. O'Reilly Media.).

[7] Silberschatz, A., Korth, H. F., & Sudarshan, S. (2019). Database System Concepts. McGraw-Hill Education. (Μια από τις πλέον κλασικές αναφορές για τη θεωρία των βάσεων δεδομένων).

[8] Banker, K., Bakkum, P., Verch, S., Douglas, D., & Hawkins, T. (2016). MongoDB in Action: Covers MongoDB version 3.0. Manning Publications. (Μια εξαιρετική πηγή για την κατανόηση της εγγράφου-κεντρικής προσέγγισης των NoSQL βάσεων).