ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
Τμήμα Πληροφορικής

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Τίτλος Πτυχιακής Εργασίας	Σχεδίαση και ανάπτυξη λογισμικού προώθησης και διαχείρισης πωλήσεων με εξατομικευμένες συστάσεις. Design and development of personalized recommendation eshop.
Ονοματεπώνυμο Φοιτητή	Κοροβέσης Ευάγγελος
Πατρώνυμο	Γεώργιος
Αριθμός Μητρώου	Π20096
Επιβλέπων	Ευάγγελος Σακκόπουλος, Αναπληρωτής Καθηγητής

Ημερομηνία Παράδοσης Ιούνιος 2025

Copyright ©

Επιτελική Σύνοψη

Η παρούσα πτυχιακή εργασία αφορά την ανάπτυξη μιας διαδικτυακής εφαρμογής ηλεκτρονικού εμπορίου (e-shop), η οποία σχεδιάστηκε και υλοποιήθηκε με σκοπό τη διαχείριση και προώθηση ποδοσφαιρικών ειδών για μία φανταστική ομάδα. Η εφαρμογή αναπτύχθηκε με τη χρήση της τεχνολογίας ASP.NET Core MVC, αξιοποιώντας τις δυνατότητες της γλώσσας προγραμματισμού C# και μιας σχεσιακής βάσης δεδομένων SQL, ώστε να διασφαλιστεί η αποδοτική και ασφαλής αποθήκευση των δεδομένων.

Η δημιουργία της εφαρμογής βασίστηκε στην ανάγκη ύπαρξης ενός λειτουργικού και οργανωμένου e-shop, το οποίο επιτρέπει τη διαχείριση προϊόντων, κατηγοριών και ρόλων χρηστών, ενώ ταυτόχρονα προσφέρει εξατομικευμένες προτάσεις αγορών στους επισκέπτες του. Παρόλο που στην αγορά υπάρχουν ήδη πολλές πλατφόρμες ηλεκτρονικού εμπορίου, η παρούσα εφαρμογή διακρίνεται λόγω της προσαρμοσμένης εμπειρίας χρήστη και της δυνατότητας παροχής προσωποποιημένων προτάσεων, βασισμένων στις προτιμήσεις των πελατών.

Κατά τη διαδικασία ανάπτυξης, δόθηκε ιδιαίτερη έμφαση τόσο στη σχεδίαση του μοντέλου δεδομένων όσο και στη λειτουργικότητα της εφαρμογής, ώστε να εξασφαλιστεί η ομαλή αλληλεπίδραση μεταξύ των χρηστών και του συστήματος. Η χρήση του Visual Studio ως κύριου περιβάλλοντος ανάπτυξης και η εφαρμογή του MVC (Model-View-Controller) μοντέλου συνέβαλαν στην ευέλικτη σχεδίαση της πλατφόρμας.

Η εργασία αυτή αποτελεί μια πλήρη μελέτη σχεδίασης, ανάπτυξης και ανάλυσης μιας διαδικτυακής εφαρμογής ηλεκτρονικού εμπορίου, η οποία μπορεί να αποτελέσει βάση για μελλοντικές επεκτάσεις και βελτιώσεις στον τομέα των εξατομικευμένων προτάσεων αγορών.

Ευχαριστίες

Θα ήθελα να εκφράσω τις ειλικρινείς μου ευχαριστίες στον επιβλέποντα καθηγητή μου κ. Ευάγγελο Σακκόπουλο για την πολύτιμη καθοδήγηση, τη στήριξη και τις συμβουλές του καθ' όλη τη διάρκεια της εκπόνησης αυτής της πτυχιακής εργασίας. Η καθοδήγησή του υπήρξε καθοριστική στην ολοκλήρωση του έργου μου.

Επίσης, θα ήθελα να ευχαριστήσω όλους τους καθηγητές του Τμήματος Πληροφορικής του Πανεπιστημίου Πειραιώς για τις γνώσεις που μου παρείχαν κατά τη διάρκεια των σπουδών μου, οι οποίες αποτέλεσαν θεμέλιο λίθο για την ανάπτυξη αυτής της εφαρμογής.

Τέλος, ένα μεγάλο ευχαριστώ στην οικογένειά μου και στους φίλους μου, οι οποίοι με στήριξαν καθ’ όλη τη διάρκεια των σπουδών μου, προσφέροντάς μου τη δύναμη και την ενθάρρυνση που χρειαζόμουν για να φτάσω στην ολοκλήρωση αυτής της προσπάθειας.

Απρίλιος 2025

Κοροβέσης Ευάγγελος

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

Κεφάλαιο 1ο

1 Εισαγωγή

Η παρούσα πτυχιακή εργασία πραγματεύεται την ανάπτυξη ενός ηλεκτρονικού καταστήματος (e-shop) για μια φανταστική ποδοσφαιρική ομάδα. Η εφαρμογή έχει σχεδιαστεί με σκοπό τη διευκόλυνση της πώλησης ποδοσφαιρικών ειδών μέσω μιας διαδικτυακής πλατφόρμας, επιτρέποντας στους χρήστες να περιηγούνται, να επιλέγουν και να αγοράζουν προϊόντα εύκολα και γρήγορα.

1.1 Περιγραφή του υπό μελέτη προβλήματος

Η ανάγκη για την ανάπτυξη αυτής της εφαρμογής προκύπτει από τη διαρκώς αυξανόμενη ζήτηση για διαδικτυακές αγορές αθλητικών ειδών. Ιδιαίτερα οι φίλαθλοι ποδοσφαίρου επιθυμούν να προμηθεύονται προϊόντα της αγαπημένης τους ομάδας μέσω εύχρηστων και λειτουργικών ηλεκτρονικών καταστημάτων. Το πρόβλημα που επιχειρεί να λύσει αυτή η εργασία είναι η έλλειψη μιας προσαρμοσμένης πλατφόρμας που επιτρέπει τη διαχείριση και προώθηση προϊόντων ποδοσφαιρικού περιεχομένου με απλό και αποδοτικό τρόπο.

1.2 Σκοπός και στόχοι της εργασίας

Ο σκοπός της εργασίας είναι η δημιουργία ενός πλήρως λειτουργικού e-shop, το οποίο θα διαθέτει όλα τα απαραίτητα χαρακτηριστικά για την αποτελεσματική διαχείριση των προϊόντων και των χρηστών. Οι βασικοί στόχοι της εργασίας περιλαμβάνουν:

- Ανάπτυξη μιας διαδικτυακής εφαρμογής βασισμένης στο μοντέλο MVC (Model-View-Controller) χρησιμοποιώντας ASP.NET Core.

- Υλοποίηση δυναμικής διαχείρισης προϊόντων, κατηγοριών και χρηστών.

- Δημιουργία ενός συστήματος αγορών που επιτρέπει την προσθήκη προϊόντων στο καλάθι και την ολοκλήρωση των παραγγελιών.

- Υλοποίηση μηχανισμού σύστασης προϊόντων, ο οποίος προτείνει σχετικά είδη με βάση τις προηγούμενες προβολές ή αγορές του χρήστη, ενισχύοντας την εμπειρία πλοήγησης και την πιθανότητα ολοκλήρωσης παραγγελίας.

- Ενσωμάτωση λειτουργιών ασφαλείας, όπως σύστημα αυθεντικοποίησης και ρόλων χρηστών (π.χ. διαχειριστής και πελάτης).

1.3 Βασικοί ορισμοί

Για την κατανόηση της λειτουργικότητας της εφαρμογής, παρουσιάζονται βασικοί όροι που χρησιμοποιούνται:

- E-shop: Διαδικτυακή εφαρμογή που επιτρέπει την πώληση προϊόντων μέσω του διαδικτύου.

- MVC (Model-View-Controller): Αρχιτεκτονικό μοτίβο ανάπτυξης εφαρμογών που διαχωρίζει τη λογική της εφαρμογής από τη διεπαφή χρήστη.

- ASP.NET Core: Framework ανάπτυξης web εφαρμογών της Microsoft, το οποίο επιτρέπει την κατασκευή δυναμικών και επεκτάσιμων συστημάτων.

    - ASP.NET Core Identity: Ενσωματωμένο σύστημα ελέγχου ταυτότητας και διαχείρισης χρηστών που προσφέρει έτοιμες λειτουργίες όπως εγγραφή, σύνδεση, ρόλοι χρηστών και διαχείριση πρόσβασης. Χρησιμοποιήθηκε ως βάση για την υλοποίηση των σελίδων "Login" και "Register" του e-shop (φυσικά και δέχτηκαν επεξεργασία στην συνέχεια) , ενώ συνέβαλε σημαντικά στη δημιουργία του σχήματος της βάσης δεδομένων μας, προσφέροντας έτοιμα models και migration μηχανισμούς που επιτάχυναν σημαντικά την ανάπτυξη.

- Καλάθι αγορών: Δυνατότητα προσθήκης προϊόντων σε μια προσωρινή λίστα πριν την ολοκλήρωση της αγοράς.

- Διαχειριστής (Admin): Ο χρήστης με δικαιώματα διαχείρισης του e-shop, όπως επεξεργασία/διαγραφή προϊόντων, δυνατότητα αναβάθμισης ρόλου ενός απλού χρήστη σε διαχειριστή ή καθολική διαγραφή ενός απλού χρήστη.

- Τελικός χρήστης (Customer): Ο χρήστης που περιηγείται στο e-shop και πραγματοποιεί αγορές.

1.4 Παραδοτέα της εργασίας

Η εργασία περιλαμβάνει τα ακόλουθα παραδοτέα:

1. Το έντυπο κείμενο της πτυχιακής εργασίας, το οποίο περιλαμβάνει την ανάλυση της υλοποίησης, τη μεθοδολογία ανάπτυξης και τα συμπεράσματα.
2. Ο πηγαίος κώδικας της εφαρμογής, υλοποιημένος σε ASP.NET Core MVC με χρήση SQL για τη διαχείριση της βάσης δεδομένων.
3. Η τεκμηρίωση της ανάπτυξης της εφαρμογής, περιλαμβάνοντας τα use case diagrams, τη βάση δεδομένων και τις βασικές αρχιτεκτονικές αποφάσεις.

1.5 Δομή της εργασίας

Η υπόλοιπη εργασία οργανώνεται ως εξής:

- Κεφάλαιο 2: Σχεδίαση της εφαρμογής με UML και παρουσίαση της βάσης δεδομένων.
- Κεφάλαιο 3: Αναλυτική περιγραφή του κώδικα και των βασικών λειτουργιών.
- Κεφάλαιο 4: Εγχειρίδιο χρήσης για απλό χρήστη και διαχειριστή.
- Κεφάλαιο 5: Συμπεράσματα και προτάσεις βελτίωσης.
- Κεφάλαιο 6: Βιβλιογραφικές πηγές.

Κεφάλαιο 2ο

2 Σχεδίαση της Εφαρμογής

Στο παρόν κεφάλαιο παρουσιάζεται η διαδικασία ανάλυσης και σχεδίασης της εφαρμογής ηλεκτρονικού καταστήματος που υλοποιήθηκε στο πλαίσιο αυτής της πτυχιακής εργασίας. Η σωστή και μεθοδική σχεδίαση αποτελεί θεμέλιο λίθο για την επιτυχημένη ανάπτυξη κάθε πληροφοριακού συστήματος, καθώς επιτρέπει την αποσαφήνιση των απαιτήσεων και τη μείωση της πολυπλοκότητας κατά την υλοποίηση. Η διαδικασία αυτή περιλαμβάνει τη δημιουργία διαγραμμάτων, τα οποία βοηθούν στη γραφική απεικόνιση της λειτουργικότητας του συστήματος και διευκολύνουν την κατανόησή του τόσο από την πλευρά του προγραμματιστή όσο και από τρίτους αναγνώστες ή μελλοντικούς συντηρητές της εφαρμογής. Για τον σκοπό αυτό, χρησιμοποιήθηκαν διαγράμματα UML, τα οποία προσφέρουν ένα κοινά αποδεκτό πρότυπο περιγραφής αντικειμενοστραφών συστημάτων. Παράλληλα, ιδιαίτερη βαρύτητα δόθηκε και στον σχεδιασμό της βάσης δεδομένων, καθώς αποτελεί τον βασικό μηχανισμό αποθήκευσης και διαχείρισης των πληροφοριών του συστήματος, όπως προϊόντα, χρήστες, παραγγελίες κ.ά. Χωρίς ένα οργανωμένο και σωστά δομημένο σχήμα βάσης δεδομένων, θα ήταν πρακτικά αδύνατη η εύρυθμη λειτουργία της εφαρμογής.

2.1 Η χρήση των UML διαγραμμάτων

Για την κατανόηση, σχεδίαση και οργάνωση μιας εφαρμογής όπως το ηλεκτρονικό κατάστημα που αναπτύχθηκε στα πλαίσια της παρούσας πτυχιακής εργασίας, κρίθηκε απαραίτητη η χρήση διαγραμμάτων UML. Τα UML (Unified Modeling Language) διαγράμματα προσφέρουν έναν ξεκάθαρο και δομημένο τρόπο αποτύπωσης των λειτουργιών και της ροής της εφαρμογής, βοηθώντας τόσο στη φάση του σχεδιασμού όσο και κατά την υλοποίηση.

Αν και υπάρχουν αρκετοί τύποι UML διαγραμμάτων, επιλέξαμε να επικεντρωθούμε σε δύο που θεωρήσαμε πως αποτυπώνουν με τον πιο κατανοητό τρόπο τις βασικές πτυχές της εφαρμογής μας:

- Διαγράμματα περιπτώσεων χρήσης (Use Case Diagrams): Με αυτά παρουσιάζονται τα βασικά σενάρια αλληλεπίδρασης των χρηστών με την εφαρμογή, όπως η περιήγηση στο κατάστημα, η προσθήκη προϊόντων στο καλάθι, η εγγραφή και η σύνδεση χρηστών, αλλά και οι λειτουργίες διαχείρισης από πλευράς admin.

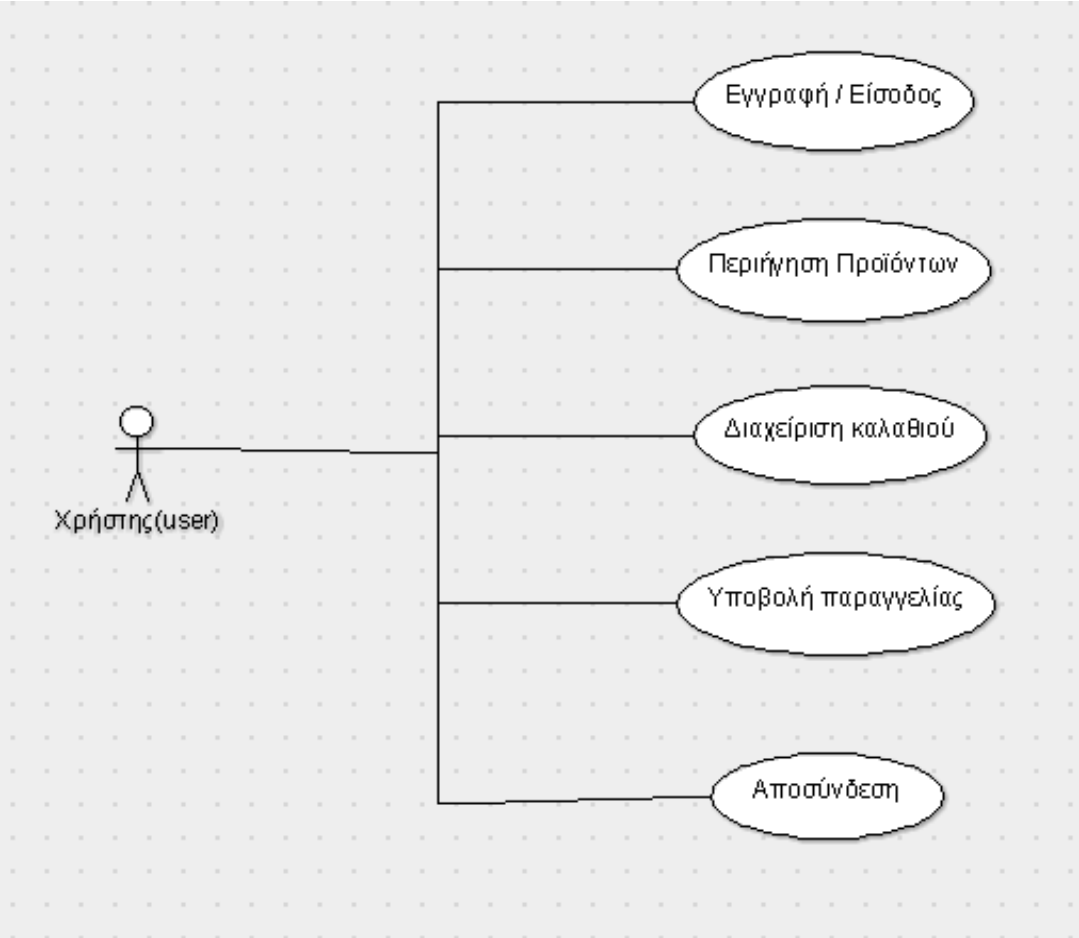
- Διαγράμματα ακολουθίας (Sequence Diagrams): Αποτυπώνουν τη ροή και την αλληλεπίδραση μεταξύ των βασικών αντικειμένων του συστήματος, δείχνοντας πώς εκτελούνται συγκεκριμένες λειτουργίες βήμα προς βήμα. Για παράδειγμα, έχει σχεδιαστεί ένα τέτοιο διάγραμμα που περιγράφει τη διαδικασία ολοκλήρωσης μιας αγοράς από την επιλογή προϊόντων μέχρι την καταχώρηση της παραγγελίας.

Τα διαγράμματα που παρουσιάζονται στο κεφάλαιο αυτό δημιουργήθηκαν με τη βοήθεια του εργαλείου ArgoUML, το οποίο αποτελεί μία δωρεάν και ανοιχτού κώδικα εφαρμογή σχεδίασης διαγραμμάτων. Η επιλογή του συγκεκριμένου εργαλείου έγινε λόγω της απλότητας στη χρήση του, αλλά και της υποστήριξής του για βασικά UML διαγράμματα, όπως τα διαγράμματα περιπτώσεων χρήσης. Κατά τη διάρκεια της ανάπτυξης της εφαρμογής, τα διαγράμματα βοήθησαν σημαντικά στην καλύτερη κατανόηση και απεικόνιση των βασικών λειτουργιών του συστήματος, ενώ αποτέλεσαν και χρήσιμο υλικό τεκμηρίωσης στο πλαίσιο της πτυχιακής εργασίας.

2.1.1 Διαγράμματα Περιπτώσεων Χρήσης (Use Case Diagrams)

Τα διαγράμματα περιπτώσεων χρήσης χρησιμοποιούνται για την αποτύπωση των βασικών λειτουργιών που προσφέρει ένα σύστημα στους χρήστες του. Μέσα από αυτά, παρουσιάζεται με απλό και κατανοητό τρόπο η αλληλεπίδραση των διαφορετικών τύπων χρηστών με την εφαρμογή, εστιάζοντας στο τι μπορεί να κάνει ο κάθε ρόλος και όχι στο πώς υλοποιούνται τεχνικά αυτές οι λειτουργίες. Αποτελούν ένα πολύτιμο εργαλείο κατά τα αρχικά στάδια του σχεδιασμού, καθώς συμβάλλουν στην κατανόηση των απαιτήσεων και στην οργάνωση των βασικών σεναρίων χρήσης του συστήματος.

Στο διάγραμμα περιπτώσεων χρήσης που φαίνεται παρακάτω, βλέπουμε έναν κύριο Actor, οποίος είναι στην συγκεκριμένη περίπτωση ένας απλός χρήστης που επισκέπτεται το E-shop μας καθώς και οι 5 κυριότερες λειτουργίες του χρήστη μέσα στην εφαρμογή. Επομένως, το εν λόγω διάγραμμα που διαφαίνεται παρακάτω είναι ένα case diagram υψηλής αφαίρεσης / υψηλότερου επιπέδου.



Εικόνα 1- Διάγραμμα περιπτώσεων χρήσης απλού user - υψηλού επιπέδου

Περιγραφή των περιπτώσεων χρήσης του απλού χρήστη:

- Εγγραφή / Είσοδος
  Ο χρήστης έχει τη δυνατότητα να δημιουργήσει έναν νέο λογαριασμό (εγγραφή) ή να εισέλθει στον λογαριασμό του (είσοδος), ώστε να μπορεί να αξιοποιήσει τις εξατομικευμένες λειτουργίες του συστήματος.

- Περιήγηση Προϊόντων
  Αφού συνδεθεί ή ακόμα και ως επισκέπτης, ο χρήστης μπορεί να δει τη λίστα των διαθέσιμων προϊόντων στο e-shop. Αυτή η λειτουργία περιλαμβάνει την εμφάνιση προϊόντων με βασικές πληροφορίες όπως εικόνα, τιμή και περιγραφή.

- Διαχείριση καλαθιού

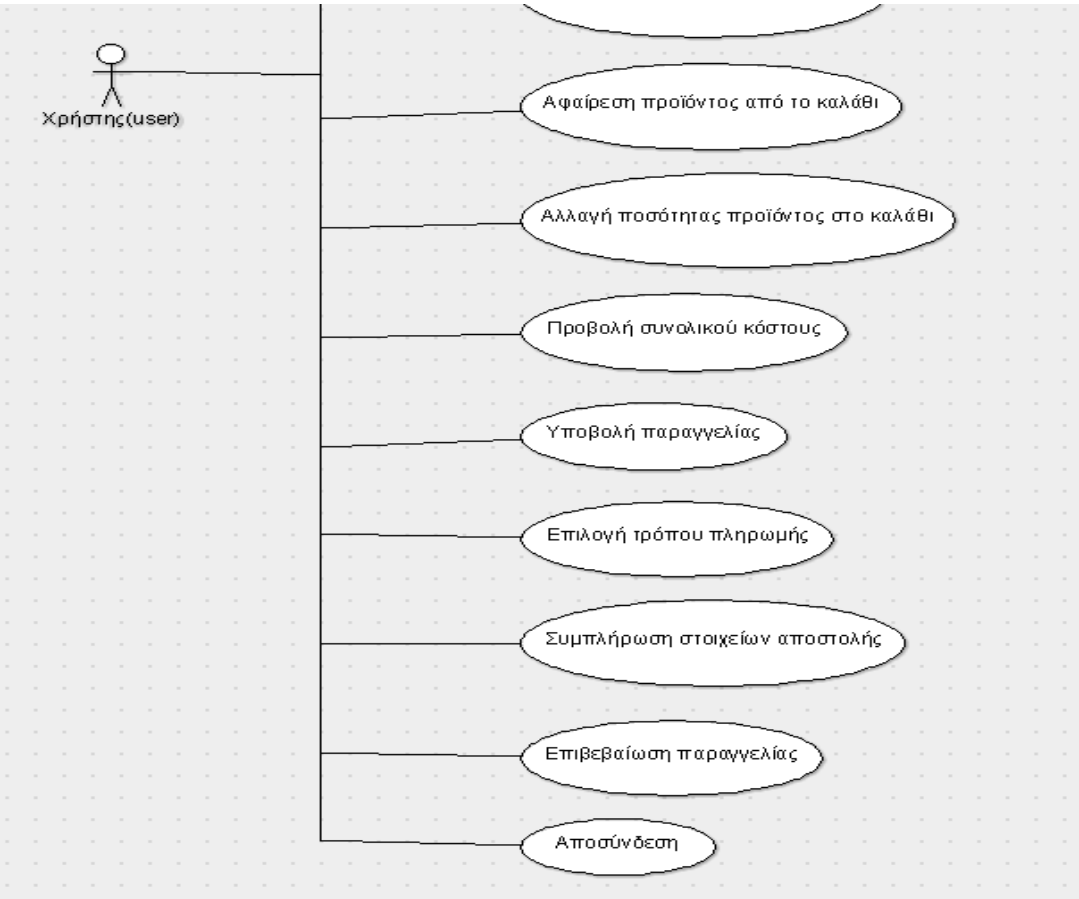
Μέσω αυτής της λειτουργίας, ο χρήστης μπορεί να προσθέτει, αφαιρεί ή επεξεργάζεται προϊόντα στο καλάθι του πριν προχωρήσει σε αγορά.

- Υποβολή παραγγελίας
  Όταν ο χρήστης ολοκληρώσει την επιλογή των προϊόντων του, μπορεί να προχωρήσει στην τελική υποβολή της παραγγελίας, η οποία αποθηκεύεται στο σύστημα μαζί με τα απαραίτητα στοιχεία.

- Αποσύνδεση
  Τέλος, ο χρήστης έχει τη δυνατότητα να αποσυνδεθεί από τον λογαριασμό του, τερματίζοντας έτσι την ενεργή συνεδρία του στο e-shop.

Αυτό το διάγραμμα αποτυπώνει την αρχική γενική εικόνα του τρόπου με τον οποίο ο τελικός χρήστης αλληλεπιδρά με το σύστημα. Στη συνέχεια, κάθε μία από αυτές τις λειτουργίες αναλύεται περαιτέρω με επιμέρους υπο-περιπτώσεις χρήσης, ώστε να γίνει πιο λεπτομερής απεικόνιση της ροής της κάθε ενέργειας:

Εικόνα 2 - Διάγραμμα περιπτώσεων χρήσης απλού user (μέρος 1/2)



Εικόνα 3 - Διάγραμμα περιπτώσεων χρήσης απλού user (μέρος 2/2)

Αναλυτική περιγραφή των περιπτώσεων χρήσης του απλού χρήστη:

- Εγγραφή (με έλεγχο ορθότητας στοιχείων)
  Ο επισκέπτης της ιστοσελίδας έχει τη δυνατότητα να δημιουργήσει νέο λογαριασμό χρήστη. Κατά τη διαδικασία της εγγραφής πραγματοποιούνται έλεγχοι για την εγκυρότητα των στοιχείων που καταχωρούνται, όπως η μορφή του email και η επαλήθευση του κωδικού πρόσβασης.

- Είσοδος (με έλεγχο ορθότητας στοιχείων)
  Ένας ήδη εγγεγραμμένος χρήστης μπορεί να συνδεθεί στο λογαριασμό του χρησιμοποιώντας τα σωστά διαπιστευτήρια (email και κωδικό πρόσβασης). Αν τα στοιχεία είναι λανθασμένα, εμφανίζεται κατάλληλο μήνυμα σφάλματος.

- Περιήγηση προϊόντων
  Ο χρήστης μπορεί να περιηγηθεί ελεύθερα στη λίστα των διαθέσιμων προϊόντων του ηλεκτρονικού καταστήματος, χωρίς να απαιτείται είσοδος στον λογαριασμό του.

- Φιλτράρισμα προϊόντων κατά κατηγορία
  Δίνεται η δυνατότητα στο χρήστη να περιορίσει τα εμφανιζόμενα προϊόντα επιλέγοντας συγκεκριμένη κατηγορία, ώστε να βρει πιο εύκολα αυτό που αναζητά.

- Φιλτράρισμα προϊόντων κατά τιμή
  Επιπλέον, ο χρήστης μπορεί να ορίσει εάν επιθυμεί να αντικρίσει τα προϊόντα κατά αύξουσα ή φθίνουσα τιμή, κάνοντας την αναζήτηση πιο στοχευμένη.

- Προβολή λεπτομερειών προϊόντων
  Κάθε προϊόν συνοδεύεται από αναλυτικές πληροφορίες, όπως σύντομη περιγραφή και τιμή. Ο χρήστης μπορεί να δει αυτές τις λεπτομέρειες επιλέγοντας το προϊόν.

- Σύσταση προϊόντων
  Η εφαρμογή παρέχει στον χρήστη προτάσεις προϊόντων βάσει προηγούμενων επισκέψεων ή επιλεγμένων ειδών, προσφέροντας μια πιο προσωποποιημένη εμπειρία πλοήγησης.

- Διαχείριση καλαθιού
  Ο χρήστης έχει πρόσβαση στο καλάθι του ανά πάσα στιγμή και μπορεί να δει τα προϊόντα που έχει προσθέσει, να αλλάξει τις ποσότητες ή να αφαιρέσει κάποια από αυτά.

- Προσθήκη προϊόντος στο καλάθι
  Κατά την περιήγηση, ο χρήστης μπορεί να προσθέσει ένα προϊόν στο καλάθι του με ένα κλικ.

- Αφαίρεση προϊόντος από το καλάθι
  Εφόσον αλλάξει γνώμη, ο χρήστης μπορεί εύκολα να αφαιρέσει οποιοδήποτε προϊόν από το καλάθι πριν προχωρήσει στην ολοκλήρωση της αγοράς.

- Αλλαγή ποσότητας προϊόντος στο καλάθι
  Υπάρχει η δυνατότητα τροποποίησης της ποσότητας για κάθε προϊόν που βρίσκεται στο καλάθι, ώστε ο χρήστης να αγοράσει τον αριθμό τεμαχίων που επιθυμεί.

- Προβολή συνολικού κόστους
  Σε πραγματικό χρόνο εμφανίζεται το συνολικό κόστος όλων των προϊόντων στο καλάθι.

- Υποβολή παραγγελίας
  Όταν ο χρήστης ολοκληρώσει τις επιλογές του, μπορεί να προχωρήσει στην υποβολή της παραγγελίας.

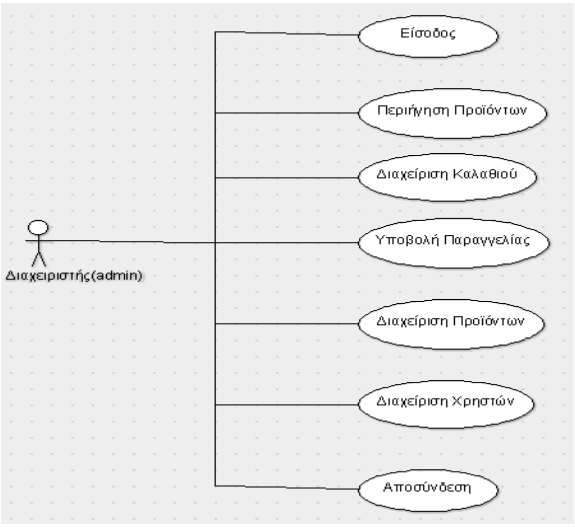
- Επιλογή τρόπου πληρωμής
  Ο χρήστης καλείται να επιλέξει τον τρόπο πληρωμής που τον εξυπηρετεί περισσότερο (π.χ. αντικαταβολή ή άλλος διαθέσιμος τρόπος).

- Συμπλήρωση στοιχείων αποστολής
  Πριν την τελική επιβεβαίωση, ο χρήστης πρέπει να εισάγει τα απαραίτητα στοιχεία αποστολής, όπως ονοματεπώνυμο και διεύθυνση αποστολής.

- Επιβεβαίωση παραγγελίας
  Μετά τη συμπλήρωση όλων των στοιχείων, ο χρήστης επιβεβαιώνει την παραγγελία του και λαμβάνει σχετική ενημέρωση στην οθόνη όπου και θα αντικρίσει μήνυμα εκτιμώμενης ημερομηνίας παράδοσης.

- Αποσύνδεση
  Τέλος, ο χρήστης μπορεί να αποσυνδεθεί από τον λογαριασμό του.

Συνεχίζουμε με το διάγραμμα περιπτώσεων χρήσης του διαχειριστή(admin). Έτσι, βλέπουμε έναν κύριο Actor, οποίος σε αυτήν την περίπτωση είναι ο διαχειριστής του E-shop μας καθώς και τις 7 κυριότερες λειτουργίες του admin μέσα στην εφαρμογή. Επομένως, το εν λόγω διάγραμμα που διαφαίνεται παρακάτω είναι ένα case diagram υψηλής αφαίρεσης / υψηλότερου επιπέδου.



Εικόνα 4 - Διάγραμμα περιπτώσεων χρήσης admin - υψηλού επιπέδου

Περιγραφή των περιπτώσεων χρήσης του διαχειριστή:

- Είσοδος
  Ο διαχειριστής αποκτά πρόσβαση στην πλατφόρμα μέσω μιας διαδικασίας αυθεντικοποίησης, εισάγοντας τα διαπιστευτήριά του.

- Περιήγηση Προϊόντων
  Όπως και ο τελικός χρήστης, ο διαχειριστής μπορεί να βλέπει όλα τα διαθέσιμα προϊόντα του e-shop, είτε για λόγους επίβλεψης είτε για μελλοντική επεξεργασία.

- Διαχείριση Καλαθιού
  Αν και κύρια χρήση του καλαθιού αφορά τον πελάτη, ο διαχειριστής έχει επίσης τη δυνατότητα να προσομοιώσει αγορές, αν αυτό απαιτείται για έλεγχο ή δοκιμές.

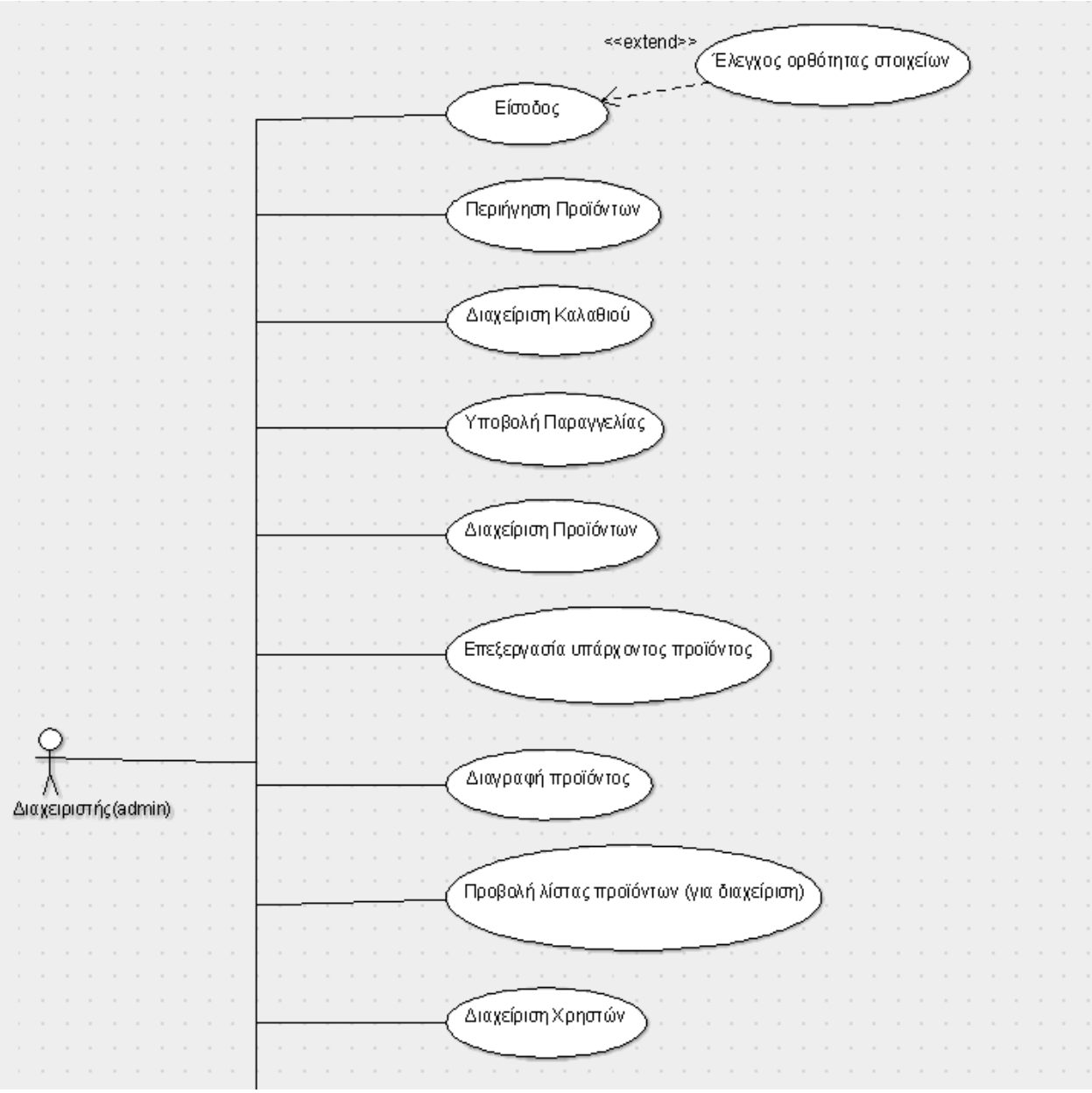
- Υποβολή Παραγγελίας
  Ο διαχειριστής δύναται να υποβάλει παραγγελίες όπως ένας απλός χρήστης – για δοκιμαστικούς ή λειτουργικούς λόγους, εφόσον χρειαστεί.

- Διαχείριση Προϊόντων
  Η βασική αρμοδιότητα του admin είναι να διαχειρίζεται και να επεξεργάζεται πλήρως τον κατάλογο των προϊόντων.

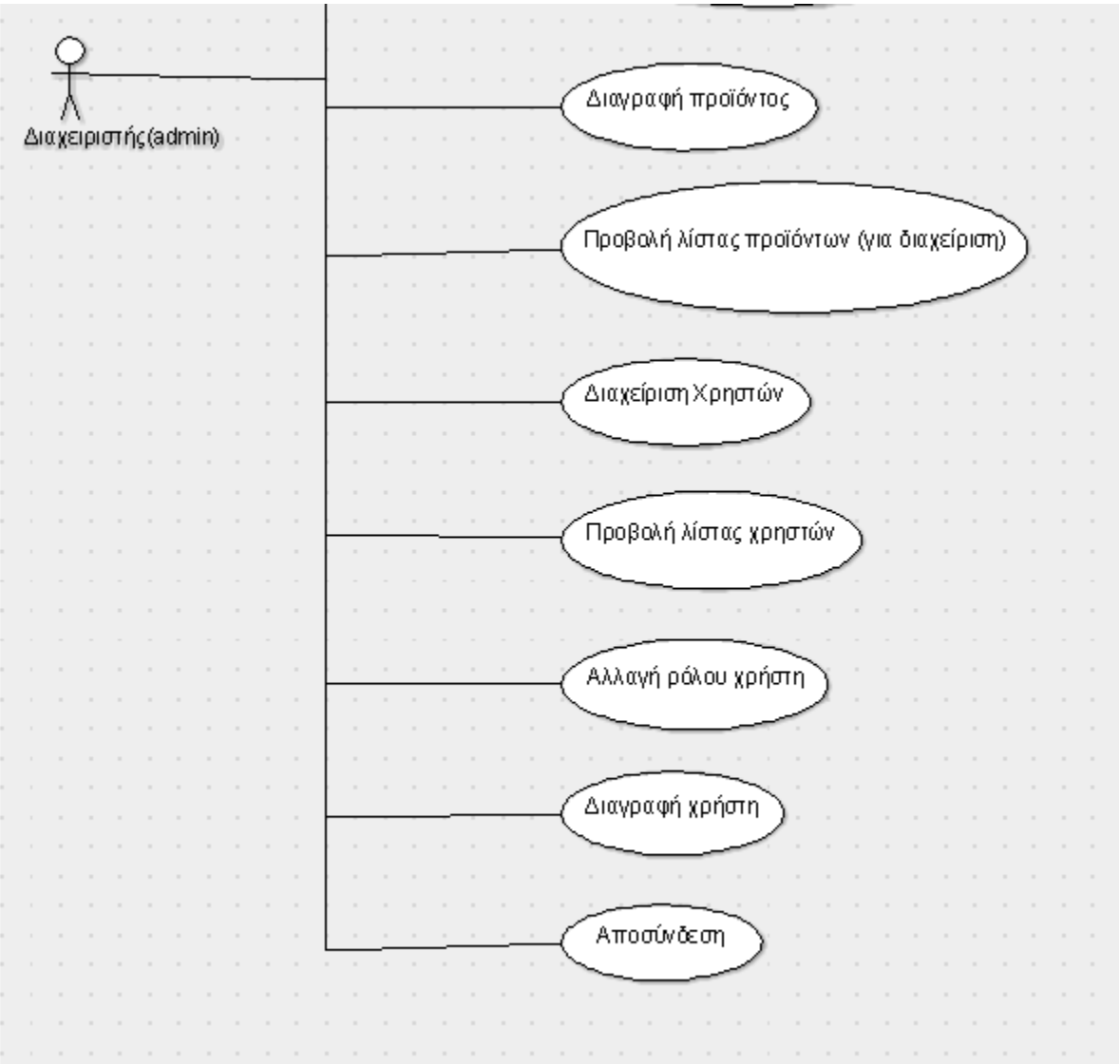
- Διαχείριση Χρηστών
  Ο διαχειριστής μπορεί να έχει πρόσβαση στα προφίλ των εγγεγραμμένων χρηστών, να τους αναβαθμίζει σε ρόλο admin ή και να διαγράφει λογαριασμούς, σε περιπτώσεις καταχρηστικής συμπεριφοράς ή εσφαλμένων εγγραφών.

- Αποσύνδεση
  Τέλος, ο διαχειριστής έχει τη δυνατότητα να αποσυνδεθεί από το σύστημα, ολοκληρώνοντας έτσι με ασφάλεια τη συνεδρία του.

Παρακάτω φαίνεται το αναλυτικό διάγραμμα περιπτώσεων χρήσης του διαχειριστή(admin), έτσι ώστε να συμπεριλάβουμε σχηματικά όλες τις ενέργειες τις οποίες μπορεί να επιτελέσει ο admin του E-shop μέσα στην εφαρμογή:



Εικόνα 5 - Διάγραμμα περιπτώσεων χρήσης admin (μέρος 1/2)



Εικόνα 6 - Διάγραμμα περιπτώσεων χρήσης admin (μέρος 2/2)

Αναλυτική περιγραφή των περιπτώσεων χρήσης διαχειριστή:

- Είσοδος
  Ο διαχειριστής εισάγει τα διαπιστευτήριά του (όνομα χρήστη & κωδικό). Σε περίπτωση που τα στοιχεία είναι έγκυρα, αποκτά πρόσβαση στο περιβάλλον διαχείρισης.

- Περιήγηση Προϊόντων
  Ο διαχειριστής μπορεί να δει τη λίστα των διαθέσιμων προϊόντων του e-shop, όπως και ο τελικός χρήστης.

- Διαχείριση Καλαθιού
  Παρόλο που ο διαχειριστής έχει πρόσβαση στο καλάθι, δεν είναι κύρια λειτουργία του. Μπορεί να προσθέσει ή να αφαιρέσει προϊόντα στο πλαίσιο δοκιμών.

- Υποβολή Παραγγελίας
  Αν ο διαχειριστής επιθυμεί, μπορεί να προσομοιώσει παραγγελίες για έλεγχο της λειτουργίας.

- Διαχείριση Προϊόντων
  Ο διαχειριστής μπορεί να προβεί σε ένα σύνολο ενεργειών, έτσι ώστε να διαχειριστεί τα προϊόντα του E-shop όπως εκείνος επιθυμεί.

- Επεξεργασία υπάρχοντος προϊόντος
  Υπάρχει η δυνατότητα τροποποίησης των χαρακτηριστικών ενός προϊόντος και πιο συγκεκριμένα, δυνατότητα τροποποίησης του ονόματος του προϊόντος, της περιγραφής του, της τιμής του, της κατηγορίας στην οποία ανήκει, καθώς και της εικόνας που διαφαίνεται μέσα από το E-shop για το συγκεκριμένο προϊόν.

- Διαγραφή προϊόντος
  Ο διαχειριστής μπορεί να αφαιρέσει προϊόντα από το e-shop οριστικά.

- Προβολή λίστας προϊόντων (για διαχείριση)
  Ο διαχειριστής βλέπει συγκεντρωμένα όλα τα προϊόντα με δυνατότητα άμεσης ενέργειας (Edit/Delete).

- Διαχείριση Χρηστών
  Ο διαχειριστής του E-shop έχει τη δυνατότητα να προχωρήσει σε ένα σύνολο ενεργειών, έτσι ώστε να διαχειριστεί τους εγγεγραμμένους χρήστες του.

- Προβολή λίστας χρηστών
  Ο admin έχει πρόσβαση σε λίστα με όλους τους εγγεγραμμένους χρήστες.

- Αλλαγή ρόλου χρήστη
  Υπάρχει δυνατότητα να προάγει έναν απλό χρήστη σε admin.

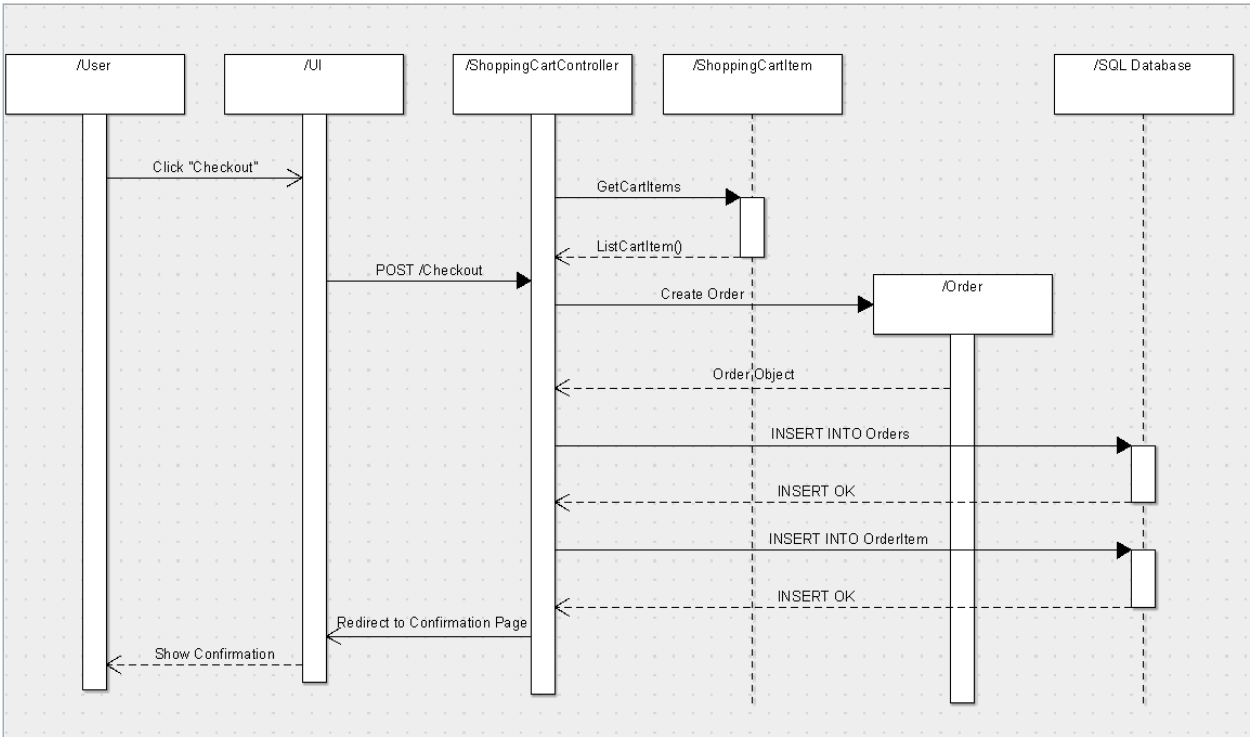
- Διαγραφή χρήστη
  Ο admin μπορεί να διαγράψει οριστικά έναν απλό χρήστη από τη βάση δεδομένων.

- Αποσύνδεση
  Ο διαχειριστής τερματίζει τη συνεδρία του και επιστρέφει στην αρχική σελίδα.

2.1.2 Διαγράμματα ακολουθίας (Sequence diagrams)

Τα διαγράμματα ακολουθίας αποτελούν ένα από τα πιο βασικά εργαλεία στην ανάλυση και σχεδίαση μιας εφαρμογής, καθώς μας επιτρέπουν να κατανοήσουμε και να αποτυπώσουμε με ακρίβεια τη ροή των ενεργειών που πραγματοποιούνται στο σύστημα κατά την εκτέλεση μιας συγκεκριμένης λειτουργίας. Στην περίπτωση της δικής μας εφαρμογής, τα διαγράμματα αυτά απεικονίζουν τη χρονική σειρά με την οποία «επικοινωνούν» μεταξύ τους διάφορα αντικείμενα, όπως controllers, views, και στοιχεία της βάσης δεδομένων, για να εκτελεστούν λειτουργίες όπως η είσοδος χρήστη, η επεξεργασία ενός προϊόντος ή η ολοκλήρωση μιας παραγγελίας. Κάθε τέτοιο αντικείμενο μπορεί να αντιστοιχεί σε μια κλάση του συστήματος ή σε έναν πίνακα της βάσης δεδομένων, ανάλογα με τη λειτουργία που εξετάζουμε. Η σωστή απεικόνιση της χρονικής αλληλουχίας των μηνυμάτων μεταξύ αυτών των αντικειμένων μάς βοηθά να κατανοήσουμε καλύτερα το πώς «ρολάρει» η εφαρμογή σε κάθε σενάριο χρήσης και εντοπίζει πιθανά σημεία που χρήζουν βελτίωσης ή επανεξέτασης κατά την ανάπτυξη.

Το παρόν sequence diagram απεικονίζει τη ροή ενεργειών που εκτελούνται κατά τη διαδικασία υποβολής παραγγελίας από έναν εγγεγραμμένο χρήστη στο ηλεκτρονικό κατάστημα. Το σενάριο ξεκινά με την ενέργεια του χρήστη να επιλέγει την ολοκλήρωση αγοράς (“Checkout”), ακολουθεί η επεξεργασία της παραγγελίας από τον controller της εφαρμογής και ολοκληρώνεται με την αποθήκευση των σχετικών δεδομένων στη βάση δεδομένων και την εμφάνιση της σελίδας επιβεβαίωσης.



Εικόνα 7 - Διάγραμμα ακολουθίας Υποβολής Παραγγελίας απλού χρήστη

Συμμετέχοντα αντικείμενα:

- User: Ο τελικός χρήστης του συστήματος που επιθυμεί να ολοκληρώσει την αγορά των προϊόντων του.

- UI (Browser): Η διεπαφή χρήστη μέσω της οποίας πραγματοποιούνται οι ενέργειες (HTML/CSS/JavaScript views).

- ShoppingCartController: Ο ASP.NET MVC controller που χειρίζεται το αίτημα υποβολής παραγγελίας και ενεργοποιεί τη σχετική επιχειρησιακή λογική.

- ShoppingCartItem: Η λογική αναπαράσταση των αντικειμένων που περιέχονται στο καλάθι του χρήστη.

- Order: Η κλάση μοντέλου που αναπαριστά την παραγγελία που πρόκειται να δημιουργηθεί.

- SQL Database: Το υποσύστημα της βάσης δεδομένων στο οποίο αποθηκεύονται τα δεδομένα της παραγγελίας και των αντίστοιχων στοιχείων των προϊόντων.

Έτσι, με χρονική σειρά, οι ενέργειες εξελίσσονται ως εξής:

1. Ο χρήστης επιλέγει την ενέργεια "Checkout" από τη διεπαφή χρήστη.

2. Το UI αποστέλλει HTTP POST αίτημα στον ShoppingCartController για την υποβολή της παραγγελίας.

3. Ο ShoppingCartController καλεί τα δεδομένα του καλαθιού μέσω της μεθόδου GetCartItems(), και λαμβάνει πίσω μια λίστα αντικειμένων ShoppingCartItem.

4. Στη συνέχεια, δημιουργείται ένα νέο αντικείμενο Order με βάση τα δεδομένα του καλαθιού.

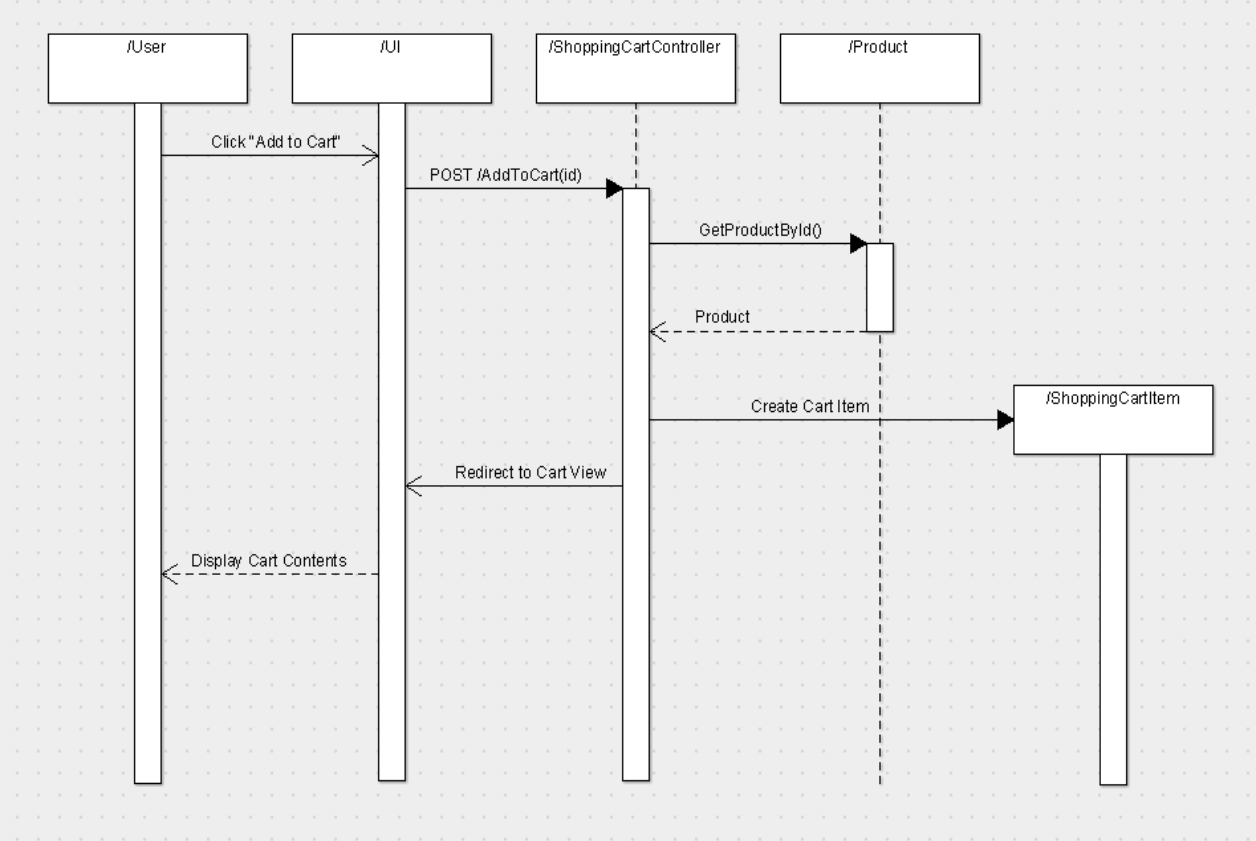
5. Η παραγγελία αποστέλλεται στη βάση δεδομένων μέσω SQL εντολής τύπου INSERT INTO Orders.

6. Ακολουθεί η αποθήκευση των επιμέρους γραμμών της παραγγελίας (OrderItems) στη βάση μέσω ξεχωριστής SQL εντολής INSERT INTO OrderItems.

7. Μόλις ολοκληρωθεί η διαδικασία, ο controller ανακατευθύνει τη διεπαφή στη σελίδα επιβεβαίωσης παραγγελίας.

8. Το UI εμφανίζει στον χρήστη σχετικό μήνυμα επιβεβαίωσης της παραγγελίας.

Παρακάτω παρουσιάζεται και επεξηγείται το Διάγραμμα Ακολουθίας (Sequence Diagram) για την περίπτωση της προσθήκης ενός προϊόντος στο καλάθι από έναν απλό χρήστη.



Εικόνα 8 - Διάγραμμα ακολουθίας Προσθήκης σε Καλάθι απλού χρήστη

Το συγκεκριμένο sequence diagram αποτυπώνει τη λειτουργική ροή που εκτελείται όταν ένας χρήστης επιλέγει να προσθέσει ένα προϊόν στο καλάθι του μέσω της διεπαφής του ηλεκτρονικού καταστήματος. Το διάγραμμα επικεντρώνεται στην αλληλεπίδραση των βασικών συνιστωσών της εφαρμογής κατά την επεξεργασία της εν λόγω ενέργειας.

Συμμετέχοντα αντικείμενα:

- User: Ο επισκέπτης ή εγγεγραμμένος χρήστης του e-shop, ο οποίος αλληλεπιδρά με τη διεπαφή.

- UI (Browser): Η ιστοσελίδα του προϊόντος (π.χ. Product Details View), μέσω της οποίας ο χρήστης ενεργοποιεί την προσθήκη στο καλάθι.

- ShoppingCartController: Ο controller του ASP.NET MVC που χειρίζεται το αίτημα και διαχειρίζεται την επιχειρησιακή ροή.

- Product: Το μοντέλο που αντιστοιχεί στο προϊόν, το οποίο ανακτάται από τη βάση για επιβεβαίωση ύπαρξης και στοιχείων.

- ShoppingCartItem: Η λογική μονάδα που αναπαριστά το αντικείμενο που προστίθεται στο καλάθι αγορών του χρήστη.

Τώρα, με χρονική σειρά, οι ενέργειες εξελίσσονται ως εξής:

1. Ο χρήστης πατάει το κουμπί “Add to Cart” από τη σελίδα ενός προϊόντος.

2. Το UI στέλνει HTTP POST αίτημα προς τον ShoppingCartController, περνώντας ως παράμετρο το id του προϊόντος.

3. Ο ShoppingCartController ανακτά τα στοιχεία του προϊόντος καλώντας τη μέθοδο GetProductById(id) από το μοντέλο Product.

4. Το επιλεγμένο προϊόν επιστρέφεται στον controller, ώστε να επιβεβαιωθεί ότι είναι διαθέσιμο και μπορεί να προστεθεί.

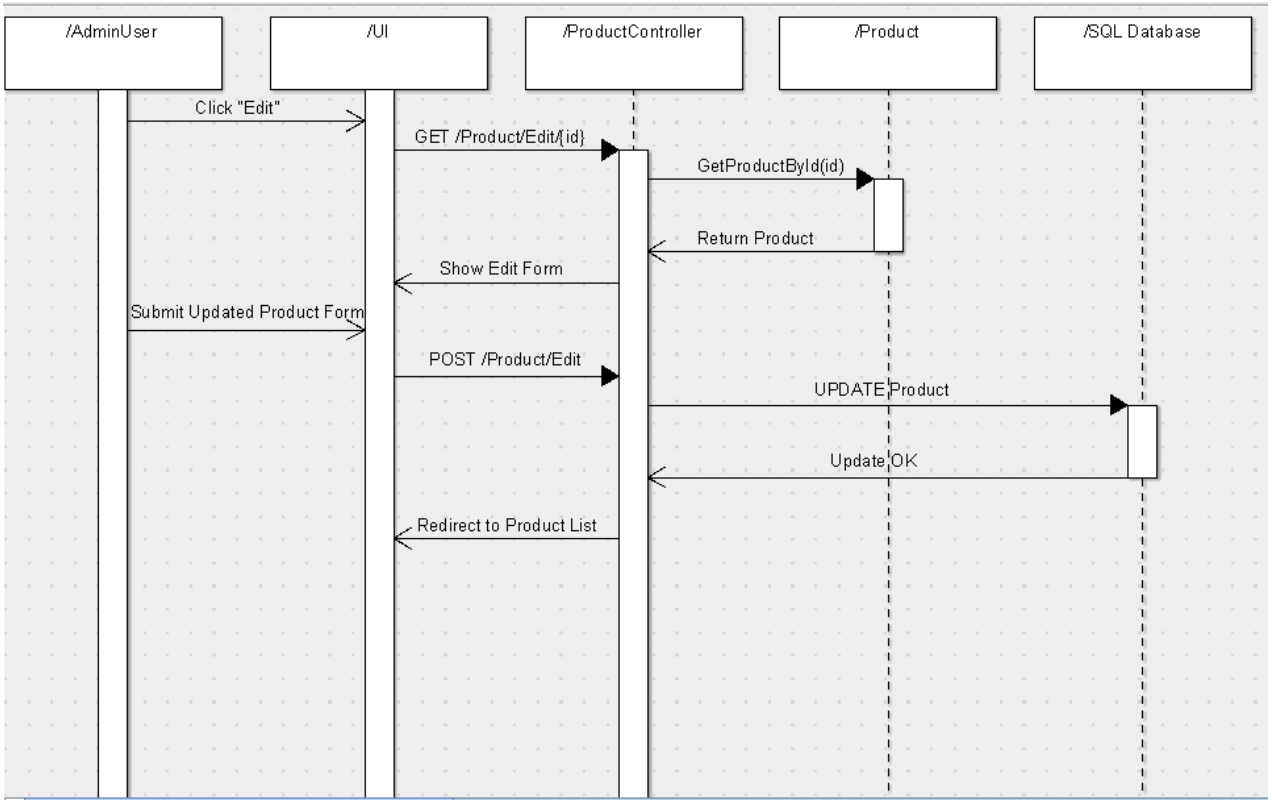
5. Ο controller δημιουργεί ένα νέο αντικείμενο ShoppingCartItem, το οποίο αντιστοιχεί στο προϊόν και προστίθεται στο καλάθι του χρήστη.

6. Ο χρήστης ανακατευθύνεται στη σελίδα του καλαθιού (Cart View) μέσω κατάλληλου redirect.

7. Το UI εμφανίζει την επικαιροποιημένη κατάσταση του καλαθιού με το νέο προϊόν.

Συνεχίζουμε με το Διάγραμμα Ακολουθίας του Διαχειριστή(admin) κατά την Επεξεργασία Υπάρχοντος Προϊόντος.

Πιο συγκεκριμένα, το παρακάτω διάγραμμα ακολουθίας αποτυπώνει τη ροή ενεργειών που εκτελείται κατά την τροποποίηση ενός ήδη υπάρχοντος προϊόντος από τον διαχειριστή της εφαρμογής. Πρόκειται για ένα βασικό σενάριο διαχείρισης περιεχομένου σε ένα ηλεκτρονικό κατάστημα, όπου ο διαχειριστής έχει πρόσβαση σε λειτουργίες επεξεργασίας προϊόντων από το admin περιβάλλον.



Εικόνα 9 - Διάγραμμα ακολουθίας Επεξεργασίας Υπάρχοντος Προϊόντος από Διαχειριστή

Συμμετέχοντα αντικείμενα:

- AdminUser: Ο διαχειριστής του συστήματος, ο οποίος πραγματοποιεί την επεξεργασία ενός προϊόντος μέσω της διεπαφής διαχείρισης.
- UI (User Interface): Η διεπαφή που εμφανίζει το προϊόν προς επεξεργασία, καθώς και τη φόρμα ενημέρωσης των πεδίων του.
- ProductController: Ο controller της εφαρμογής που διαχειρίζεται τη λογική εύρεσης και ενημέρωσης των προϊόντων.
- Product: Το μοντέλο που αντιπροσωπεύει τα προϊόντα του e-shop.
- SQL Database: Η βάση δεδομένων στην οποία είναι αποθηκευμένα τα προϊόντα και καταγράφονται οι αλλαγές.

Επομένως, με χρονική σειρά, οι ενέργειες εξελίσσονται ως εξής:

1. Ο διαχειριστής επιλέγει την ενέργεια "Edit" για κάποιο προϊόν από τη λίστα των διαθέσιμων προϊόντων.

2. Η διεπαφή χρήστη αποστέλλει αίτημα προς τον ProductController, ώστε να ανακτηθούν τα στοιχεία του συγκεκριμένου προϊόντος.

3. Ο controller καλεί το μοντέλο Product για να ανακτήσει το προϊόν με βάση το ID που του έχει δοθεί.

4. Τα στοιχεία του προϊόντος επιστρέφονται στον controller, ο οποίος με τη σειρά του τα προωθεί στη διεπαφή για να εμφανιστούν στη φόρμα επεξεργασίας.

5. Ο διαχειριστής πραγματοποιεί τις επιθυμητές αλλαγές και υποβάλλει τη φόρμα μέσω HTTP POST αιτήματος προς τον controller.

6. Ο ProductController λαμβάνει τα ενημερωμένα δεδομένα και αποστέλλει εντολή προς τη βάση δεδομένων για να πραγματοποιηθεί η αντίστοιχη ενημέρωση (UPDATE) στον σχετικό πίνακα.

7. Η βάση δεδομένων επιβεβαιώνει την επιτυχή εκτέλεση της εντολής.

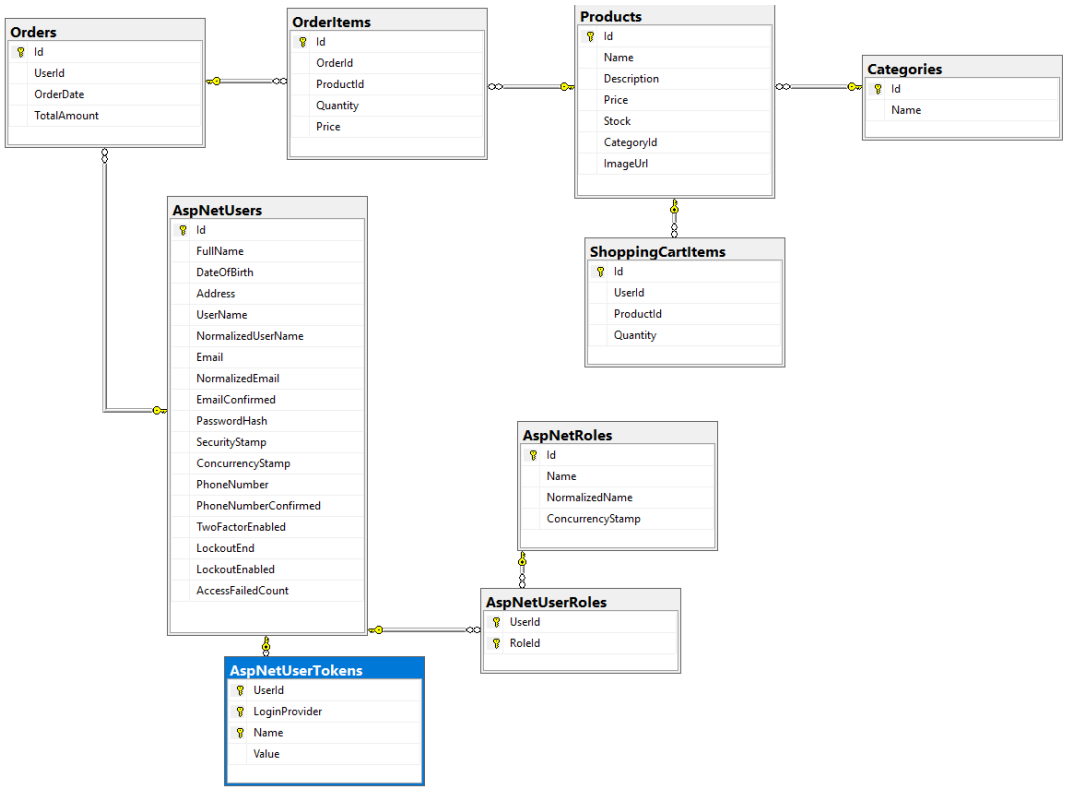
8. Μετά την ολοκλήρωση της διαδικασίας, ο controller ανακατευθύνει τον χρήστη πίσω στη λίστα προϊόντων, ώστε να δει τη νέα ενημερωμένη κατάσταση.

2.2 Η Βάση Δεδομένων

Πριν προχωρήσουμε στην ανάλυση της βάσης δεδομένων της εφαρμογής AITHERAS FC E-shop, κρίνεται σκόπιμο να παρατεθεί ένας σύντομος σχολιασμός σχετικά με τον ρόλο και τη σημασία της. Η βάση δεδομένων αποτελεί τον πυρήνα της κάθε δυναμικής διαδικτυακής εφαρμογής, καθώς εκεί αποθηκεύονται, οργανώνονται και συσχετίζονται όλες οι κρίσιμες πληροφορίες που απαιτούνται για τη λειτουργία του συστήματος. Στην παρούσα εργασία, η βάση δεδομένων φέρει το όνομα MyFootballEshopDB και υλοποιήθηκε μέσω του SQL Server Management Studio, εργαλείο το οποίο προσφέρει ένα ιδιαίτερα φιλικό περιβάλλον για τη δημιουργία, διαχείριση και παραμετροποίηση βάσεων δεδομένων. Η επικοινωνία της βάσης με την υπόλοιπη εφαρμογή επιτυγχάνεται μέσα από το μοντέλο του ASP.NET MVC, εξασφαλίζοντας πλήρη αλληλεπίδραση ανάμεσα στη βάση, την επιχειρησιακή λογική και τη διεπαφή του χρήστη.

2.2.1 Μοντέλο Οντοτήτων Συσχετίσεων (Entity-Relationship Model)

Το παρακάτω ER διάγραμμα αποτυπώνει τη δομή της βάσης δεδομένων που υποστηρίζει τη λειτουργία του διαδικτυακού καταστήματος AITHERAS FC E-Shop. Το σύστημα βασίζεται σε ένα μοντέλο πολλαπλών οντοτήτων και σχέσεων που συνδυάζει custom πίνακες για την επιχειρησιακή λειτουργία του e-shop με τις δομές του ASP.NET Identity Framework για τη διαχείριση χρηστών και ρόλων. Το διάγραμμα απεικονίζει τη σύνδεση μεταξύ προϊόντων, χρηστών, καλαθιών αγορών, παραγγελιών, καθώς και των απαιτούμενων στοιχείων ταυτοποίησης και εξουσιοδότησης χρηστών.



Εικόνα 10 - Μοντέλο οντοτήτων συσχετίσεων της βάσης δεδομένων

Ανάλυση της Δομής της Βάσης Δεδομένων

Η βάση δεδομένων αποτελείται από δύο κύριες κατηγορίες πινάκων:

◇ **<u>Πίνακες που υλοποιήθηκαν εξ ολοκλήρου κατά την ανάπτυξη της εφαρμογής</u>**

- <u>Products</u>
  - ✓ Περιέχει όλα τα προϊόντα που είναι διαθέσιμα στο e-shop.
  - ✓ Πεδία: Id, Name, Description, Price, Stock, CategoryId, ImageUrl
  - ✓ Χρησιμεύει στην αποθήκευση πληροφοριών για τα προϊόντα που διατίθενται προς πώληση.

- <u>Categories</u>
  - ✓ Πεδία: Id, Name
  - ✓ Χρησιμοποιείται για την κατηγοριοποίηση των προϊόντων (π.χ. Kits, Accessories, Shorts).

- <u>Orders</u>
  - ✓ Πεδία: Id, UserId, OrderDate, TotalAmount
  - ✓ Καταγράφει κάθε ολοκληρωμένη παραγγελία ενός χρήστη.

- <u>OrderItems</u>
  - ✓ Πεδία: Id, OrderId, ProductId, Quantity, Price
  - ✓ Περιέχει τα επιμέρους προϊόντα που περιλαμβάνονται σε κάθε παραγγελία.

- <u>ShoppingCartItems</u>
  - ✓ Πεδία: Id, UserId, ProductId, Quantity
  - ✓ Αποθηκεύει προσωρινά τα προϊόντα που έχει προσθέσει ο χρήστης στο καλάθι του, πριν προχωρήσει σε αγορά.

◇ **<u>Πίνακες του ASP.NET Identity Framework</u>**

Οι παρακάτω πίνακες δημιουργούνται αυτόματα μέσω της λειτουργικότητας ASP.NET Core Identity, την οποία αξιοποιήσαμε για την υλοποίηση του συστήματος αυθεντικοποίησης και εξουσιοδότησης χρηστών.

- <u>AspNetUsers</u>
  - ✓ Περιέχει βασικές πληροφορίες των χρηστών όπως UserName, Email, PasswordHash, αλλά και έξτρα πεδία που προσθέσαμε όπως FullName, DateOfBirth, Address.

- AspNetRoles
  - ✓ Ορίζει τους ρόλους χρηστών (π.χ. Admin).

- AspNetUserRoles
  - ✓ Πίνακας συσχέτισης πολλών-προς-πολλούς μεταξύ AspNetUsers και AspNetRoles.

- AspNetUserTokens
  - ✓ Αποθηκεύει tokens για login sessions, χρήσιμο για την υποστήριξη λειτουργιών όπως το "Remember Me".

Περιγραφή της Βάσης Δεδομένων και του Τρόπου Δημιουργίας της

Η ανάπτυξη της βάσης δεδομένων της εφαρμογής βασίστηκε -μεταξύ άλλων- σε τεχνολογίες που προσφέρει το ASP.NET Core και συγκεκριμένα το Identity Framework, το οποίο ενσωματώνει μηχανισμούς για την ασφαλή διαχείριση χρηστών και ρόλων. Οι πίνακες που δημιουργήθηκαν μέσω αυτού του μηχανισμού, όπως οι AspNetUsers, AspNetRoles, AspNetUserRoles, AspNetUserTokens, αποτελούν τον πυρήνα της λειτουργίας πιστοποίησης και εξουσιοδότησης των χρηστών της εφαρμογής.

Η δημιουργία όλων των απαραίτητων πινάκων πραγματοποιήθηκε με τη χρήση του μηχανισμού "migrations" που προσφέρει το Entity Framework. Αρχικά, έγινε η οριστικοποίηση των μοντέλων της εφαρμογής μέσα στο project, και ακολούθως, εκτελέστηκαν οι σχετικές εντολές μέσω του *Package Manager Console* στο Visual Studio, όπως Add-Migration InitialCreate και Update-Database. Μέσω αυτών των εντολών δημιουργήθηκε αυτόματα η δομή της βάσης δεδομένων στον SQL Server, χωρίς να χρειαστεί χειροκίνητη παρέμβαση.

Η σύνδεση μεταξύ των πινάκων της βάσης βασίζεται σε σχέσεις πρωτεύοντων και ξένων κλειδιών. Για παράδειγμα, στον πίνακα Products, κάθε εγγραφή περιέχει ένα CategoryId που σχετίζεται με τον πίνακα Categories, μέσω μιας σχέσης ένα-προς-πολλά (one-to-many). Αυτό επιτρέπει σε κάθε προϊόν να ανήκει σε μία συγκεκριμένη κατηγορία, ενώ παράλληλα μια κατηγορία μπορεί να περιλαμβάνει πολλά προϊόντα.

Αντίστοιχα, η σύνδεση των χρηστών με τους ρόλους γίνεται μέσω του πίνακα AspNetUserRoles, στον οποίο καταγράφεται το ποιοι χρήστες έχουν αποδοθεί σε συγκεκριμένους ρόλους, όπως "Admin". Αυτή η αρχιτεκτονική μάς επέτρεψε να ορίσουμε διακριτές λειτουργίες για κάθε

κατηγορία χρήστη, π.χ. δυνατότητα διαγραφής προϊόντων ή αναβάθμιση ρόλων που είναι διαθέσιμη μόνο στους διαχειριστές.

Η παραπάνω οργάνωση της βάσης σε συνδυασμό με το ASP.NET Identity Framework και την αρμονική συνύπαρξη των βάσεων που αυτό προσφέρει με το custom schema του E-shop, προσφέρει τόσο λειτουργική πληρότητα όσο και ασφάλεια, καλύπτοντας με αξιόπιστο τρόπο τις απαιτήσεις ενός σύγχρονου ηλεκτρονικού καταστήματος.

Κεφάλαιο 3ο

3 Αναλυτική περιγραφή του κώδικα της Εφαρμογής

Στο παρόν κεφάλαιο περιγράφεται αναλυτικά η υλοποίηση της διαδικτυακής εφαρμογής που αναπτύχθηκε στο πλαίσιο της παρούσας πτυχιακής εργασίας. Στόχος είναι η πληρέστερη δυνατή κατανόηση της λειτουργίας του e-shop της ποδοσφαιρικής ομάδας AITHERAS FC, τόσο από τεχνικής όσο και από σχεδιαστικής πλευράς.

Η ανάπτυξη της εφαρμογής βασίστηκε σε σύγχρονες τεχνολογίες web development, ακολουθώντας το πρότυπο MVC (Model-View-Controller), ενώ ιδιαίτερη έμφαση δόθηκε στην καθαρή δομή του κώδικα, τη λειτουργικότητα και την ευκολία στη χρήση από τον τελικό χρήστη. Η σχεδίαση και η υλοποίηση πραγματοποιήθηκαν στο περιβάλλον του Visual Studio με χρήση της πλατφόρμας ASP.NET Core MVC, σε συνδυασμό με SQL Server για την αποθήκευση των δεδομένων. Για την ανάπτυξη του γραφικού περιβάλλοντος αξιοποιήθηκαν HTML, CSS και Bootstrap, εξασφαλίζοντας μοντέρνα και responsive διεπαφή.

3.1 Υλοποίηση φόρμας Εγγραφής(Register form)

Η διαδικασία εγγραφής νέου χρήστη στο σύστημα υλοποιείται μέσω της σελίδας *Register.cshtml* και του αντίστοιχου page model *Register.cshtml.cs*. Ο μηχανισμός βασίζεται στο ASP.NET Core Identity, ένα ολοκληρωμένο σύστημα διαχείρισης χρηστών που παρέχει ενσωματωμένες λειτουργίες για εγγραφή, σύνδεση, ρόλους και ασφάλεια.

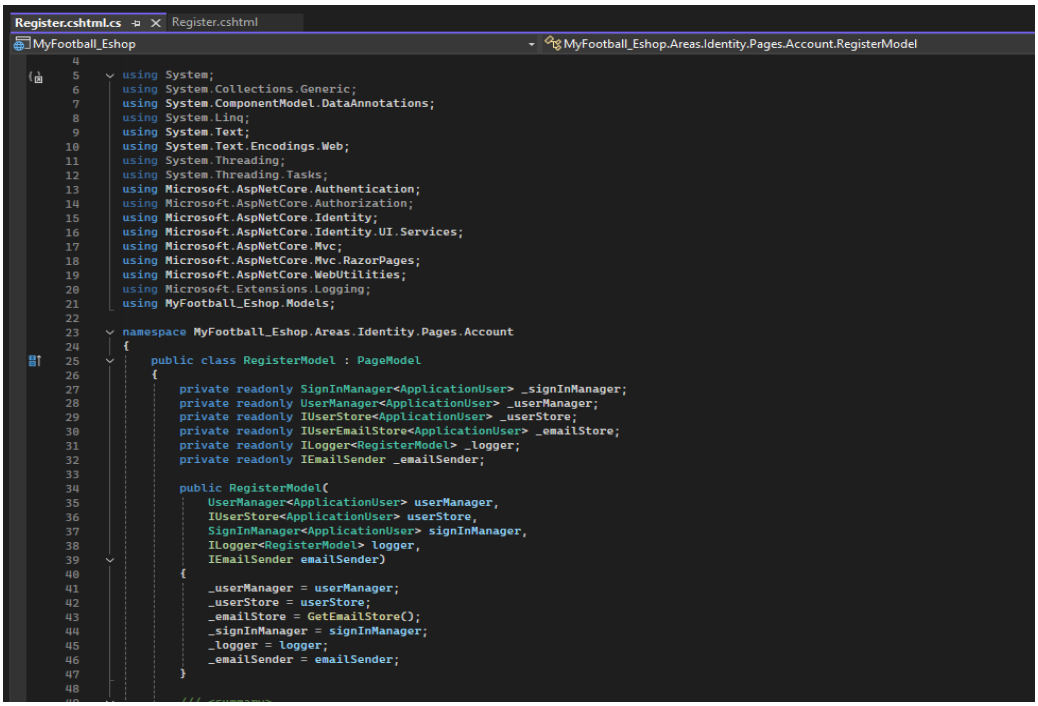
Η σελίδα Register.cshtml αποτελεί το View που εμφανίζεται στον χρήστη κατά την είσοδό του στη σελίδα εγγραφής. Περιλαμβάνει μια φόρμα με πεδία τα οποία είναι προσβάσιμα και επεξεργάσιμα από τον χρήστη. Συγκεκριμένα:

```
<div class="main-container">
    <div class="register-container">
        <h2>Register</h2>
        <form id="registerForm" asp-route-returnUrl="@Model.ReturnUrl" method="post">
            <div asp-validation-summary="ModelOnly" class="text-danger" role="alert"></div>

            <div class="input-group">
                <span class="input-group-text"><i class="fas fa-user"></i></span>
                <input asp-for="Input.FullName" class="form-control" autocomplete="name" placeholder="Full Name" required>
            </div>

            <div class="input-group">
                <span class="input-group-text"><i class="fas fa-envelope"></i></span>
                <input asp-for="Input.Email" class="form-control" autocomplete="username" placeholder="Email" required>
            </div>

            <div class="input-group">
                <span class="input-group-text"><i class="fas fa-lock"></i></span>
                <input asp-for="Input.Password" type="password" class="form-control" autocomplete="new-password" placeholder="Password" required>
            </div>

            <div class="input-group">
                <span class="input-group-text"><i class="fas fa-lock"></i></span>
                <input asp-for="Input.ConfirmPassword" type="password" class="form-control" autocomplete="new-password" placeholder="Confirm Password" required>
            </div>

            <div class="input-group">
                <span class="input-group-text"><i class="fas fa-map-marker-alt"></i></span>
                <input asp-for="Input.Address" class="form-control" autocomplete="street-address" placeholder="Address" required>
            </div>

            <div class="input-group">
                <span class="input-group-text"><i class="fas fa-calendar-alt"></i></span>
                <input asp-for="Input.DateOfBirth" class="form-control" type="date" required>
            </div>

            <button id="registerSubmit" type="submit" class="register-btn">Register</button>
        </form>
    </div>
</div>
```

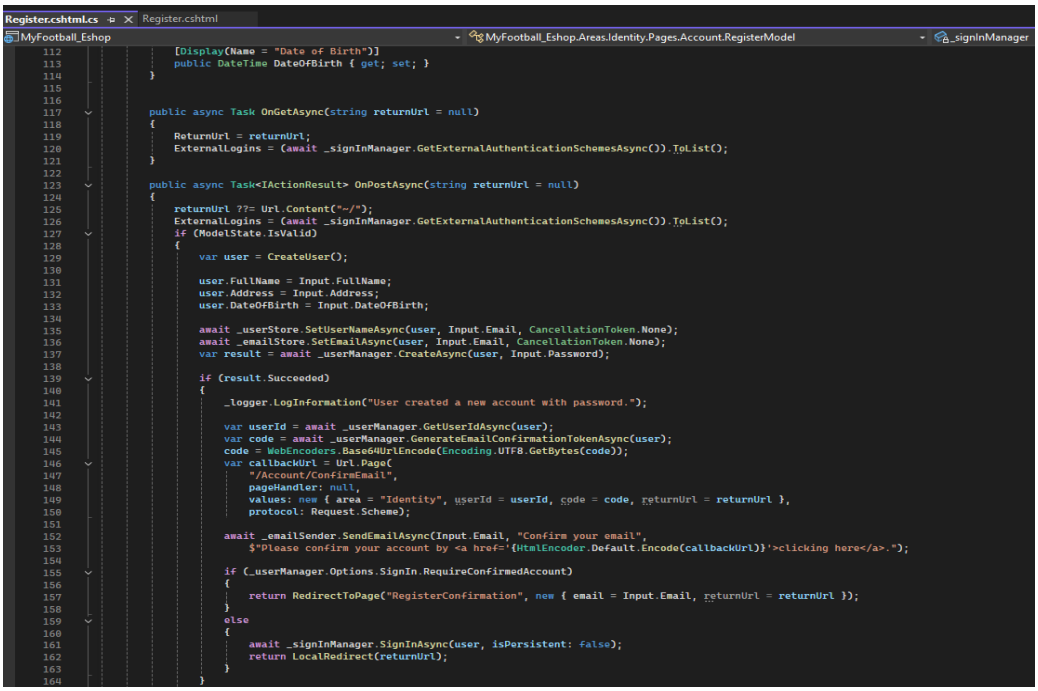
Εικόνα 11 – Register.cshtml

- Full Name: Συμπληρώνεται το ονοματεπώνυμο του χρήστη.
- Email: Εισάγεται η ηλεκτρονική διεύθυνση, η οποία αποτελεί και το βασικό στοιχείο ταυτοποίησης του χρήστη.
- Password / Confirm Password: Ορίζεται και επιβεβαιώνεται ο προσωπικός κωδικός ασφαλείας.
- Address: Βοηθητικό πεδίο που θα φανεί χρήσιμο μετέπειτα με την προσυμπλήρωση διεύθυνσης κατά την ολοκλήρωση της παραγγελίας.
- Ημερομηνία Γέννησης: Χρησιμοποιείται για την ολοκλήρωση του προφίλ του χρήστη.

Το αρχείο *Register.cshtml.cs* περιλαμβάνει τη λογική χειρισμού της εγγραφής. Βασίζεται στο μοντέλο PageModel του Razor Pages και ορίζει τη μέθοδο OnPostAsync(), η οποία καλείται κατά την υποβολή της φόρμας:



Εικόνα 12 – Register.cshtml.cs (1/3)



Εικόνα 13 – Register.cshtml.cs (2/3)

Ενδεικτικά βήματα της μεθόδου:

- Έλεγχος εγκυρότητας μοντέλου (ModelState.IsValid).
- Δημιουργία αντικειμένου χρήστη μέσω της κλάσης ApplicationUser, στην οποία έχουν προστεθεί επιπλέον πεδία (FullName, DateOfBirth, Address).
- Κλήση της μεθόδου UserManager.CreateAsync() για την καταχώρηση του νέου χρήστη στη βάση δεδομένων.

Το ASP.NET Identity διαχειρίζεται τα εξής:

- AspNetUsers πίνακας: Περιλαμβάνει Email, PasswordHash, SecurityStamp, UserName και τα πρόσθετα πεδία που έχουμε ορίσει (FullName, DateOfBirth, Address)
- Πίνακες AspNetRoles και AspNetUserRoles: Διαχειρίζονται τους ρόλους χρηστών "Admin".

Όλη η διαχείριση πραγματοποιείται μέσω Entity Framework Core, και η αρχική δημιουργία των πινάκων γίνεται μέσω migrations, με εντολές όπως:

dotnet ef migrations add InitialCreate

dotnet ef database update

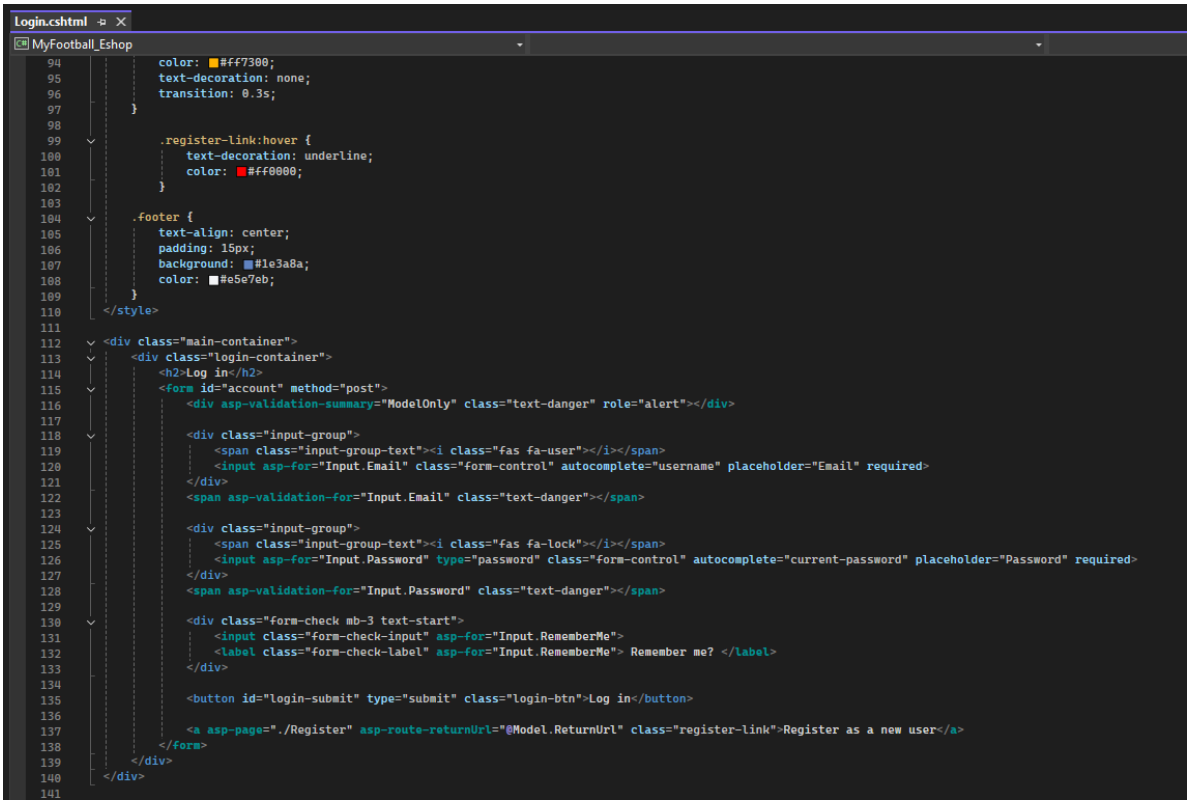
Τέλος, το μοντέλο InputModel στο Register.cshtml.cs περιλαμβάνει Data Annotations για τη βεβαίωση της εγκυρότητας των πεδίων:

```
public class InputModel
{
    /// <summary>
    ///     This API supports the ASP.NET Core Identity default UI infrastructure and is not intended to be used
    ///     directly from your code. This API may change or be removed in future releases.
    /// </summary>
    [Required]
    [EmailAddress]
    [Display(Name = "Email")]
    public string Email { get; set; }

    /// <summary>
    ///     This API supports the ASP.NET Core Identity default UI infrastructure and is not intended to be used
    ///     directly from your code. This API may change or be removed in future releases.
    /// </summary>
    [Required]
    [StringLength(100, ErrorMessage = "The {0} must be at least {2} and at max {1} characters long.", MinimumLength = 6)]
    [DataType(DataType.Password)]
    [Display(Name = "Password")]
    public string Password { get; set; }

    /// <summary>
    ///     This API supports the ASP.NET Core Identity default UI infrastructure and is not intended to be used
    ///     directly from your code. This API may change or be removed in future releases.
    /// </summary>
    [DataType(DataType.Password)]
    [Display(Name = "Confirm password")]
    [Compare("Password", ErrorMessage = "The password and confirmation password do not match.")]
    public string ConfirmPassword { get; set; }

    [Required(ErrorMessage = "Full Name is required.")]
    [Display(Name = "Full Name")]
    public string FullName { get; set; }

    [Required(ErrorMessage = "Address is required.")]
    [Display(Name = "Address")]
    public string Address { get; set; }

    [Required(ErrorMessage = "Date of Birth is required.")]
    [DataType(DataType.Date)]
    [Display(Name = "Date of Birth")]
    public DateTime DateOfBirth { get; set; }
}
```

Εικόνα 14 – Validation πεδίων στη Register.cshtml.cs (3/3)

3.2 Υλοποίηση φόρμας Σύνδεσης(Login form)

Η είσοδος ενός χρήστη στο σύστημα πραγματοποιείται μέσω της σελίδας *Login.cshtml* και του page model *Login.cshtml.cs*. Η υλοποίηση βασίζεται στο ASP.NET Core Identity, το οποίο παρέχει έτοιμους μηχανισμούς για αυθεντικοποίηση και διαχείριση συνεδρίας. Έτσι:



Εικόνα 15 – *Login.cshtml*

Η σελίδα περιλαμβάνει φόρμα με πεδία:

- Email: Το όνομα χρήστη.
- Password: Ο κωδικός πρόσβασης.
- Remember me?: Επιλογή για παραμονή σε σύνδεση.

Το page model (στο Login.cshtml.cs) χειρίζεται την υποβολή της φόρμας μέσω της OnPostAsync():

- Γίνεται έλεγχος εγκυρότητας (ModelState.IsValid).
- Καλείται η μέθοδος SignInManager.PasswordSignInAsync() για να γίνει η είσοδος του χρήστη.

- Αν τα στοιχεία είναι ορθά, ο χρήστης ανακατευθύνεται στην αρχική σελίδα.

```
Login.cshtml.cs  Login.cshtml
MyFootball_Eshop  MyFootball_Eshop.Areas.Identity.Pages.Account.LoginModel  _signInManage
85          public bool RememberMe { get; set; }
86      }
87
88      public async Task OnGetAsync(string returnUrl = null)
89      {
90          if (!string.IsNullOrEmpty(ErrorMessage))
91          {
92              ModelState.AddModelError(string.Empty, ErrorMessage);
93          }
94
95          returnUrl ??= Url.Content("~/");
96
97          // Clear the existing external cookie to ensure a clean login process
98          await HttpContext.SignOutAsync(IdentityConstants.ExternalScheme);
99
100         ExternalLogins = (await _signInManager.GetExternalAuthenticationSchemesAsync()).ToList();
101
102         ReturnUrl = returnUrl;
103     }
104
105     public async Task<IActionResult> OnPostAsync(string returnUrl = null)
106     {
107         returnUrl ??= Url.Content("~/");
108
109         ExternalLogins = (await _signInManager.GetExternalAuthenticationSchemesAsync()).ToList();
110
111         if (ModelState.IsValid)
112         {
113             // This doesn't count login failures towards account lockout
114             // To enable password failures to trigger account lockout, set lockoutOnFailure: true
115             var result = await _signInManager.PasswordSignInAsync(Input.Email, Input.Password, Input.RememberMe, lockoutOnFailure: false);
116             if (result.Succeeded)
117             {
118                 _logger.LogInformation("User logged in.");
119                 return LocalRedirect(returnUrl);
120             }
121             if (result.RequiresTwoFactor)
122             {
123                 return RedirectToPage("./LoginWith2fa", new { ReturnUrl = returnUrl, RememberMe = Input.RememberMe });
124             }
125             if (result.IsLockedOut)
126             {
127                 _logger.LogWarning("User account locked out.");
128                 return RedirectToPage("./Lockout");
129             }
130             else
131             {
132                 ModelState.AddModelError(string.Empty, "Invalid login attempt.");
133                 return Page();
134             }
135         }
```

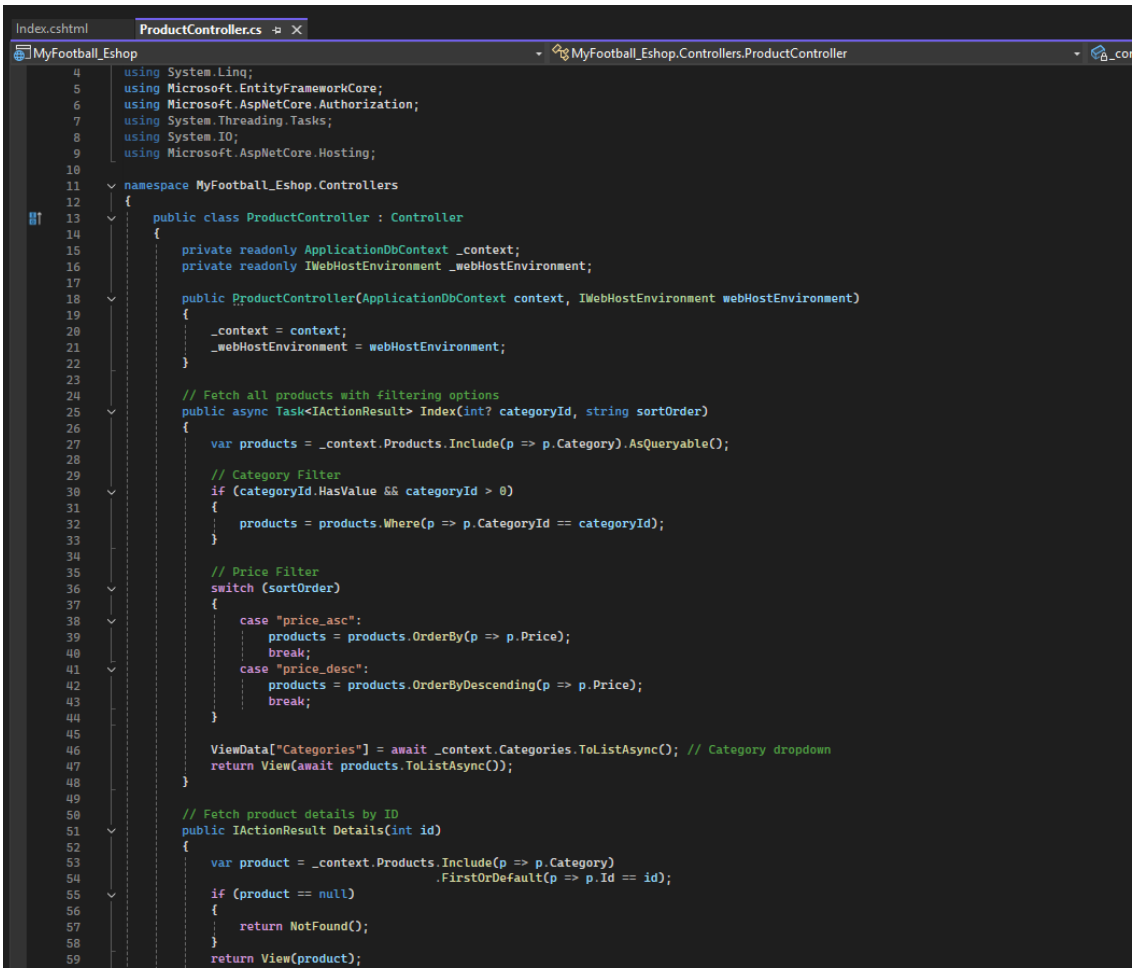
Εικόνα 16 – Login.cshtml.cs (1/2)

```
public class InputModel
{
    /// <summary>
    ///     This API supports the ASP.NET Core Identity default UI infrastructure and is not intended to be used
    ///     directly from your code. This API may change or be removed in future releases.
    /// </summary>
    [Required]
    [EmailAddress]
    public string Email { get; set; }

    /// <summary>
    ///     This API supports the ASP.NET Core Identity default UI infrastructure and is not intended to be used
    ///     directly from your code. This API may change or be removed in future releases.
    /// </summary>
    [Required]
    [DataType(DataType.Password)]
    public string Password { get; set; }
```

Εικόνα 17 – *Validation πεδίων στη Login.cshtml.cs (2/2)*

3.3 Προβολή Προϊόντων και Φίλτρα Αναζήτησης

Η προβολή των προϊόντων πραγματοποιείται μέσω της μεθόδου Index() του *ProductController* και της αντίστοιχης προβολής Views/Product/Index.cshtml. Ο χρήστης μπορεί να δει όλα τα διαθέσιμα προϊόντα και να εφαρμόσει φίλτρα για ευκολότερη πλοήγηση.



```
Index.cshtml | ProductController.cs
MyFootball_Eshop | MyFootball_Eshop.Controllers.ProductController

using System.Linq;
using Microsoft.EntityFrameworkCore;
using Microsoft.AspNetCore.Authorization;
using System.Threading.Tasks;
using System.IO;
using Microsoft.AspNetCore.Hosting;

namespace MyFootball_Eshop.Controllers
{
    public class ProductController : Controller
    {
        private readonly ApplicationDbContext _context;
        private readonly IWebHostEnvironment _webHostEnvironment;

        public ProductController(ApplicationDbContext context, IWebHostEnvironment webHostEnvironment)
        {
            _context = context;
            _webHostEnvironment = webHostEnvironment;
        }

        // Fetch all products with filtering options
        public async Task<IActionResult> Index(int? categoryId, string sortOrder)
        {
            var products = _context.Products.Include(p => p.Category).AsQueryable();

            // Category Filter
            if (categoryId.HasValue && categoryId > 0)
            {
                products = products.Where(p => p.CategoryId == categoryId);
            }

            // Price Filter
            switch (sortOrder)
            {
                case "price_asc":
                    products = products.OrderBy(p => p.Price);
                    break;
                case "price_desc":
                    products = products.OrderByDescending(p => p.Price);
                    break;
            }

            ViewData["Categories"] = await _context.Categories.ToListAsync(); // Category dropdown
            return View(await products.ToListAsync());
        }

        // Fetch product details by ID
        public IActionResult Details(int id)
        {
            var product = _context.Products.Include(p => p.Category)
                                           .FirstOrDefault(p => p.Id == id);
            if (product == null)
            {
                return NotFound();
            }
            return View(product);
```

Εικόνα 18 – ProductController – Index method

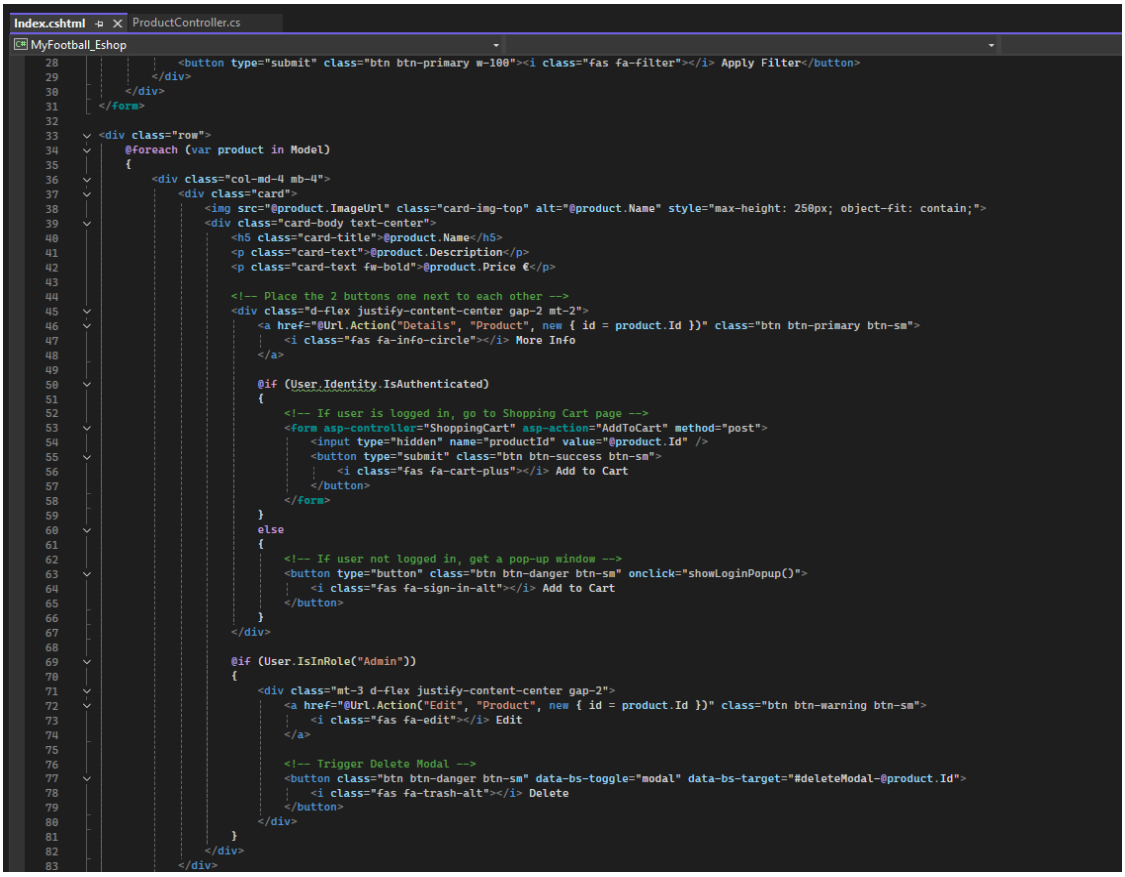
Η Index μέθοδος:

- Ανακτά όλα τα προϊόντα από τη βάση δεδομένων.
- Παρέχει δυναμικά λίστες για τα φίλτρα κατηγορίας και τιμής.
- Επιστρέφει φιλτραρισμένα αποτελέσματα όταν δοθούν παράμετροι από τον χρήστη.

Στο View Index.cshtml, γίνεται χρήση του @model List<Product> και η εμφάνιση των προϊόντων επιτυγχάνεται με επανάληψη foreach.

Κάθε προϊόν εμφανίζει:

- Εικόνα
- Όνομα, σύντομη περιγραφή και τιμή.
- Κουμπί "More Info"
- Κουμπί "Add to Cart", αν ο χρήστης είναι logged-in.
- Κουμπιά "Edit" και "Delete", αν ο χρήστης είναι αναγνωρισμένος ως admin.



Εικόνα 19 – Index.cshtml του Product View (1/2)

Ο χρήστης μπορεί να εφαρμόσει:

- Εμφάνιση προϊόντων ανά κατηγορία.
- Ταξινόμηση κατά τιμή (αύξουσα/φθίνουσα).
- Τα φίλτρα αποστέλλονται με GET method ώστε το URL να είναι καθαρό και εύκολα διαμοιράσιμο.

```
Index.cshtml | ProductController.cs
MyFootball_Eshop
1   @model List<MyFootball_Eshop.Models.Product>
2
3   <h2 class="text-center">Products</h2>
4
5   <form asp-action="Index" asp-controller="Product" method="get" class="mb-4">
6       <div class="row g-2 justify-content-center">
7           <div class="col-md-4">
8               <select name="categoryId" class="form-select">
9                   <option value="">All Categories</option>
10                  @foreach (var category in ViewData["Categories"] as List<MyFootball_Eshop.Models.Category>)
11                  {
12                      <option value="@category.Id" selected="@(category.Id.ToString() == Context.Request.Query["categoryId"])">
13                          @category.Name
14                      </option>
15                  }
16              </select>
17          </div>
18
19          <div class="col-md-4">
20              <select name="sortOrder" class="form-select">
21                  <option value="">Sort by Price</option>
22                  <option value="price_asc" selected="@(Context.Request.Query["sortOrder"] == "price_asc")">Price: Low to High</option>
23                  <option value="price_desc" selected="@(Context.Request.Query["sortOrder"] == "price_desc")">Price: High to Low</option>
24              </select>
25          </div>
26
27          <div class="col-md-2">
28              <button type="submit" class="btn btn-primary w-100"><i class="fas fa-filter"></i> Apply Filter</button>
29          </div>
30      </div>
31  </form>
```

Εικόνα 20 – Index.cshtml του Product View (2/2)

3.4 Προσθήκη Προϊόντων στο καλάθι

Η λειτουργία προσθήκης προϊόντων στο καλάθι και γενικότερα η διαχείριση των αγορών πραγματοποιείται μέσω εγγραφών στη βάση δεδομένων. Οι σχετικές ενέργειες ελέγχονται από τον *ShoppingCartController* και αξιοποιούν τον πίνακα *ShoppingCartItems*:

ShoppingCartController.cs

MyFootball_Eshop MyFootball_Eshop.Controllers.ShoppingCartController

```
31          return View(cartItems);
32      }
33
34      public async Task<IActionResult> AddToCart(int productId)
35      {
36          var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);
37          var existingItem = await _context.ShoppingCartItems
38              .FirstOrDefaultAsync(s => s.UserId == userId && s.ProductId == productId);
39
40          if (existingItem != null)
41          {
42              existingItem.Quantity++;
43          }
44          else
45          {
46              var cartItem = new ShoppingCartItem
47              {
48                  UserId = userId,
49                  ProductId = productId,
50                  Quantity = 1
51              };
52              _context.ShoppingCartItems.Add(cartItem);
53          }
54
55          await _context.SaveChangesAsync();
56          return RedirectToAction("Index");
57      }
58
59      public async Task<IActionResult> IncreaseQuantity(int id)
60      {
61          var cartItem = await _context.ShoppingCartItems.FindAsync(id);
62          if (cartItem != null)
63          {
64              cartItem.Quantity++;
65              await _context.SaveChangesAsync();
66          }
67          return RedirectToAction("Index");
68      }
69
70      public async Task<IActionResult> DecreaseQuantity(int id)
71      {
72          var cartItem = await _context.ShoppingCartItems.FindAsync(id);
73          if (cartItem != null && cartItem.Quantity > 1)
74          {
75              cartItem.Quantity--;
76              await _context.SaveChangesAsync();
77          }
78          else if (cartItem != null)
79          {
80              _context.ShoppingCartItems.Remove(cartItem);
81              await _context.SaveChangesAsync();
82          }
83          return RedirectToAction("Index");
84      }
85
86      public async Task<IActionResult> RemoveFromCart(int id)
```

Εικόνα 21 – ShoppingCartController - AddToCart(1/2)

Η μέθοδος AddToCart:

- Λαμβάνει το productId και εντοπίζει τον συνδεδεμένο χρήστη μέσω ClaimTypes.NameIdentifier.
- Αναζητά αν υπάρχει ήδη το προϊόν στο καλάθι του χρήστη (πίνακας ShoppingCartItems).
- Αν υπάρχει, αυξάνει την ποσότητα. Αν όχι, δημιουργεί νέα εγγραφή.
- Οι αλλαγές αποθηκεύονται μόνιμα στη βάση με SaveChangesAsync().

Το καλάθι αποθηκεύεται στη βάση δεδομένων μέσω του πίνακα ShoppingCartItems. Κάθε εγγραφή περιέχει UserId, ProductId και Quantity, εξασφαλίζοντας ότι κάθε χρήστης διατηρεί ξεχωριστό περιεχόμενο καλαθιού.

Αυτός ο τρόπος αποθήκευσης προσφέρει:

- Προσωποποιημένη εμπειρία ανά χρήστη
- Ανθεκτικότητα σε αποσύνδεση ή αλλαγή σελίδας
- Συγχρονισμό σε πολλαπλές συσκευές ή περιόδους σύνδεσης

Έτσι, η εφαρμογή διατηρεί το καλάθι μόνιμα για κάθε χρήστη.

Αντίστοιχες μέθοδοι υλοποιούνται για:

- Αύξηση/μείωση ποσότητας (IncreaseQuantity/DecreaseQuantity)
- Αφαίρεση εγγραφής από τον πίνακα ShoppingCartItems(RemoveFromCart)

```
57          }
58
59          public async Task<IActionResult> IncreaseQuantity(int id)
60          {
61              var cartItem = await _context.ShoppingCartItems.FindAsync(id);
62              if (cartItem != null)
63              {
64                  cartItem.Quantity++;
65                  await _context.SaveChangesAsync();
66              }
67              return RedirectToAction("Index");
68          }
69
70          public async Task<IActionResult> DecreaseQuantity(int id)
71          {
72              var cartItem = await _context.ShoppingCartItems.FindAsync(id);
73              if (cartItem != null && cartItem.Quantity > 1)
74              {
75                  cartItem.Quantity--;
76                  await _context.SaveChangesAsync();
77              }
78              else if (cartItem != null)
79              {
80                  _context.ShoppingCartItems.Remove(cartItem);
81                  await _context.SaveChangesAsync();
82              }
83              return RedirectToAction("Index");
84          }
85
86          public async Task<IActionResult> RemoveFromCart(int id)
87          {
88              var cartItem = await _context.ShoppingCartItems.FindAsync(id);
89              if (cartItem != null)
90              {
91                  _context.ShoppingCartItems.Remove(cartItem);
92                  await _context.SaveChangesAsync();
93              }
94              return RedirectToAction("Index");
95          }
```

Εικόνα 22 – ShoppingCartController – Basket Quantity(2/2)

Το περιεχόμενο του καλαθιού προβάλλεται μέσω της μεθόδου Index() του ShoppingCartController, η οποία επιστρέφει το View Views/ShoppingCart/Index.cshtml.

Κατά την είσοδο του χρήστη στη σελίδα του καλαθιού:

- Ανακτώνται από τη βάση όλα τα προϊόντα που ανήκουν στον τρέχοντα χρήστη (μέσω UserId).

- Τα δεδομένα αποστέλλονται ως List<ShoppingCartItem> στο View.

Στο View:

- Εμφανίζονται ανά γραμμή: η εικόνα του προϊόντος, η ποσότητα, η τιμή ανά τεμάχιο και η συνολική τιμή.

- Παρέχονται επιλογές για αύξηση, μείωση ή διαγραφή προϊόντος από το καλάθι.

```
Index.cshtml | ShoppingCartController.cs
MyFootball_Eshop
@model List<MyFootball_Eshop.Models.ShoppingCartItem>

@{
    ViewData["Title"] = "Shopping Cart";
}

<h2 class="text-center">🛒 Your Shopping Cart</h2>

@if (Model.Count == 0)
{
    <div class="alert alert-warning text-center">
        Your cart is empty. <a asp-controller="Product" asp-action="Index">Go shopping!</a>
    </div>
}
else
{
    <table class="table table-striped text-center">
        <thead>
            <tr>
                <th>Product</th>
                <th>Quantity</th>
                <th>Price</th>
                <th>Total</th>
                <th>Actions</th>
            </tr>
        </thead>
        <tbody>
            @foreach (var item in Model)
            {
                <tr>
                    <td>
                        <img src="@item.Product.ImageUrl" alt="@item.Product.Name" width="80" height="80" />
                        <br />
                        @item.Product.Name
                    </td>
                    <td>@item.Quantity</td>
                    <td>@item.Product.Price €</td>
                    <td>@(item.Product.Price * item.Quantity) €</td>
                    <td>
                        <a asp-controller="ShoppingCart" asp-action="IncreaseQuantity" asp-route-id="@item.Id" class="btn btn-sm btn-success">+</a>
                        <a asp-controller="ShoppingCart" asp-action="DecreaseQuantity" asp-route-id="@item.Id" class="btn btn-sm btn-warning">-</a>
                        <a asp-controller="ShoppingCart" asp-action="RemoveFromCart" asp-route-id="@item.Id" class="btn btn-sm btn-danger">Remove</a>
                    </td>
                </tr>
            }
        </tbody>
    </table>

    <div class="text-right">
        <h4>Total: <strong>@Model.Sum(item => item.Product.Price * item.Quantity) €</strong></h4>
        <a asp-controller="ShoppingCart" asp-action="Checkout" class="btn btn-primary btn-lg">Checkout</a>
    </div>
}
```

Εικόνα 23 – Views/ShoppingCart/Index.cshtml

3.5 Υποβολή Παραγγελίας(Checkout)

Η υποβολή της παραγγελίας αποτελεί το τελικό και σημαντικότερο βήμα στη ροή μιας αγοράς. Η διαδικασία υλοποιείται μέσω της μεθόδου Checkout στον *ShoppingCartController* και της αντίστοιχης φόρμας στο View *Checkout.cshtml*.

Η διαδικασία Checkout υλοποιείται μέσω δύο μεθόδων (GET και POST) στον ShoppingCartController. Ο στόχος είναι η υποβολή της παραγγελίας και η μεταφορά των προϊόντων από το καλάθι στους πίνακες Orders και OrderItems.

Η GET Checkout:

```
95          }
96
97          public async Task<IActionResult> Checkout()
98          {
99              var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);
100             var user = await _context.Users.FirstOrDefaultAsync(u => u.Id == userId);
101
102             if (user == null)
103             {
104                 return RedirectToAction("Index");
105             }
106
107             var viewModel = new CheckoutViewModel
108             {
109                 FullName = user.FullName ?? "", // Auto fill-in user's credentials
110                 Address = user.Address ?? "",
111                 PaymentMethod = "",
112                 CardNumber = "",
113                 ExpiryDate = "",
114                 CVC = ""
115             };
116
117             return View(viewModel);
118         }
119
```

Εικόνα 24 – GET Checkout

- Φορτώνει τα στοιχεία του συνδεδεμένου χρήστη (FullName, Address) για να προσυμπληρωθεί η φόρμα.

- Επιστρέφει ένα CheckoutViewModel στο view με κενά πεδία πληρωμής.

```
[HttpPost]
public async Task<IActionResult> Checkout(CheckoutViewModel model)
{
    var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);
    if (userId == null)
    {
        return RedirectToAction("Index");
    }

    var user = await _context.Users.FirstOrDefaultAsync(u => u.Id == userId);
    if (user == null)
    {
        return RedirectToAction("Index");
    }

    var cartItems = await _context.ShoppingCartItems
        .Include(s => s.Product)
        .Where(s => s.UserId == userId)
        .ToListAsync();

    if (!cartItems.Any())
    {
        return RedirectToAction("Index");
    }

    // Make sure that fullname and address are always filled-in
    model.FullName = string.IsNullOrEmpty(model.FullName) ? user.FullName : model.FullName;
    model.Address = string.IsNullOrEmpty(model.Address) ? user.Address : model.Address;

    if (string.IsNullOrEmpty(model.PaymentMethod))
    {
        ModelState.AddModelError("PaymentMethod", "Please select a payment method.");
        return View(model);
    }

    if (model.PaymentMethod == "CreditCard")
    {
        if (string.IsNullOrEmpty(model.CardNumber) || model.CardNumber.Length != 16 || !model.CardNumber.All(char.IsDigit))
        {
            ModelState.AddModelError("CardNumber", "Invalid Credit Card Number. It must be 16 digits.");
            return View(model);
        }

        if (string.IsNullOrEmpty(model.ExpiryDate) || !model.ExpiryDate.Contains("/") || model.ExpiryDate.Length != 5)
        {
            ModelState.AddModelError("ExpiryDate", "Invalid Expiry Date format. Use MM/YY.");
            return View(model);
        }

        var expiryParts = model.ExpiryDate.Split('/');
        if (expiryParts.Length != 2 || !int.TryParse(expiryParts[0], out int month) || !int.TryParse("20" + expiryParts[1], out int year))
        {
            ModelState.AddModelError("ExpiryDate", "Invalid Expiry Date format. Use MM/YY.");
            return View(model);
        }
```

Εικόνα 25 – POST Checkout

<u>Η POST Checkout:</u>

- Validation πληρωμής: Αν η μέθοδος είναι "CreditCard", γίνεται έλεγχος σε αριθμό κάρτας, ημερομηνία λήξης και CVC.

- Έλεγχος αποθέματος: Για κάθε προϊόν στο καλάθι, μειώνεται το Stock αν υπάρχει επάρκεια.

- Καταχώρηση παραγγελίας: Δημιουργείται νέα εγγραφή στον πίνακα Orders, ενώ για κάθε προϊόν, δημιουργείται εγγραφή στο OrderItems.

- Καθαρισμός καλαθιού: Διαγράφονται οι σχετικές εγγραφές από τον πίνακα ShoppingCartItems.

- Ανακατεύθυνση σε επιβεβαίωση: Ο χρήστης οδηγείται στη σελίδα OrderConfirmation.

3.6 Επιβεβαίωση Παραγγελίας (OrderConfirmation)

Μετά την επιτυχή υποβολή της παραγγελίας, ο χρήστης ανακατευθύνεται στη μέθοδο OrderConfirmation του *ShoppingCartController*, η οποία εμφανίζει τη σελίδα *OrderConfirmation.cshtml*.

```
        public IActionResult OrderConfirmation()
        {
            ViewBag.DeliveryDate = System.DateTime.Now.AddDays(2).ToString("dd/MM/yyyy"); // Delivery date +2 days
            return View();
        }
    }
}
```

Εικόνα 26 – OrderConfirmation method

OrderConfirmation.cshtml | ShoppingCartController.cs

MyFootball_Eshop

```
1   @{
2       ViewData["Title"] = "Order Confirmation";
3   }
4
5   <div class="container text-center mt-5">
6       <h2 class="text-success">☑ Order Placed Successfully!</h2>
7       <p class="lead">Thank you for your purchase.</p>
8
9       <div class="alert alert-info">
10          📦 Your order will be delivered by: <strong>@ViewBag.DeliveryDate</strong>
11      </div>
12
13      <a href="@Url.Action("Index", "Home")" class="btn btn-primary mt-3">Return to Home</a>
14  </div>
```

Εικόνα 27 – OrderConfirmation.cshtml

Στην σελίδα αυτή:

- Εμφανίζεται μήνυμα επιτυχούς καταχώρησης της παραγγελίας.
- Παρουσιάζεται η εκτιμώμενη ημερομηνία παράδοσης, υπολογισμένη ως δύο ημέρες από την ημερομηνία υποβολής.

Κεφάλαιο 4ο

4 Εγχειρίδιο χρήσης της Εφαρμογής

Στο παρόν κεφάλαιο παρουσιάζονται τρία διακριτά εγχειρίδια χρήσης της διαδικτυακής πλατφόρμας του ηλεκτρονικού καταστήματος της AITHERAS FC, τα οποία απευθύνονται στους βασικούς τύπους χρηστών του συστήματος: τους απλούς χρήστες (επισκέπτες/πελάτες), τους διαχειριστές (admins) αλλά και τους απλούς επισκέπτες που δεν έχουν πραγματοποιήσει (ακόμα) σύνδεση.

Η ύπαρξη αυτών των εγχειριδίων θεωρείται ιδιαίτερα σημαντική, καθώς παρέχουν αναλυτικές οδηγίες για την πλοήγηση και αξιοποίηση των βασικών και προχωρημένων λειτουργιών της εφαρμογής. Ιδιαίτερη μέριμνα δίνεται στην παρουσίαση των δυνατοτήτων κάθε τύπου χρήστη με τρόπο κατανοητό, ακόμη και σε άτομα που ενδεχομένως δεν έχουν προηγούμενη εμπειρία με παρόμοιες διαδικτυακές εφαρμογές.

Παράλληλα, τα εγχειρίδια συμβάλλουν στην καλύτερη κατανόηση της συνολικής λειτουργίας της πλατφόρμας, εστιάζοντας στην πρακτική χρήση και αξιοποίηση των παρεχόμενων εργαλείων. Μέσα από αυτά, εξασφαλίζεται η ορθή χρήση της εφαρμογής, τόσο σε επίπεδο απλών αγορών όσο και σε επίπεδο διαχείρισης προϊόντων και χρηστών από την πλευρά του διαχειριστή.

4.1 Εγκατάσταση της Εφαρμογής

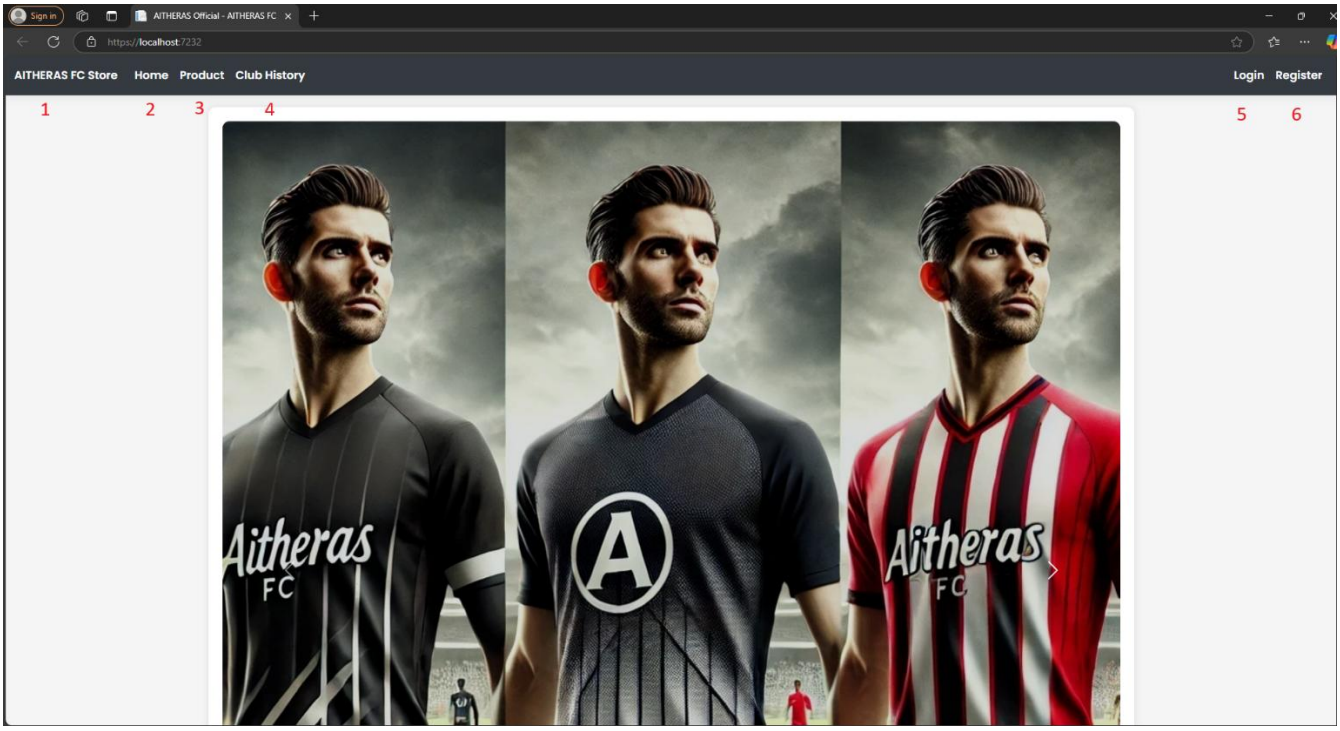
Για τη σωστή εγκατάσταση και λειτουργία της εφαρμογής, ο χρήστης θα πρέπει να διαθέτει το λογισμικό Visual Studio 2022, μέσω του οποίου θα ανοίξει και θα εκτελέσει το project. Το Visual Studio προσφέρει πλήρη υποστήριξη για εφαρμογές που βασίζονται στο ASP.NET Core MVC framework και διευκολύνει σημαντικά την ανάπτυξη αλλά και τη διαχείριση του κώδικα.

Επιπλέον, για τη διαχείριση και σύνδεση της εφαρμογής με τη σχετική βάση δεδομένων, απαιτείται η εγκατάσταση του SQL Server Management Studio 19 (SSMS 19). Στα παραδοτέα της εργασίας περιλαμβάνεται αρχείο SQL με το σχήμα της βάσης δεδομένων καθώς και τα δεδομένα που χρησιμοποιήθηκαν κατά τη διάρκεια της ανάπτυξης. Το αρχείο αυτό μπορεί να εισαχθεί στο SSMS 19, ώστε να δημιουργηθεί το απαραίτητο περιβάλλον για την ορθή λειτουργία της εφαρμογής.

Με την εγκατάσταση των παραπάνω εργαλείων, ο χρήστης είναι σε θέση να εκτελέσει την εφαρμογή τοπικά, να εξετάσει τη λειτουργικότητά της και να αλληλεπιδράσει πλήρως με όλα τα χαρακτηριστικά που έχουν υλοποιηθεί.

4.2 Χρήση της Εφαρμογής από Απλό Χρήστη

Ξεκινώντας την εφαρμογή, ο χρήστης αντικρίζει την αρχική του E-shop, η οποία διαφαίνεται παρακάτω:



Εικόνα 28 - Αρχική σελίδα του E-shop

Σε αυτή την σελίδα, ο χρήστης μπορεί να επιλέξει:

1. AITHERAS FC Store
   Όλοι οι χρήστες, είτε είναι συνδεδεμένοι είτε επισκέπτες, μπορούν να πατήσουν στο όνομα/logo και να επιστρέψουν στην αρχική σελίδα της εφαρμογής.

2. Home
   Ο χρήστης μεταφέρεται στην αρχική σελίδα, όπου προβάλλεται carousel με εικόνες και τα πιο πρόσφατα ή δημοφιλή προϊόντα.

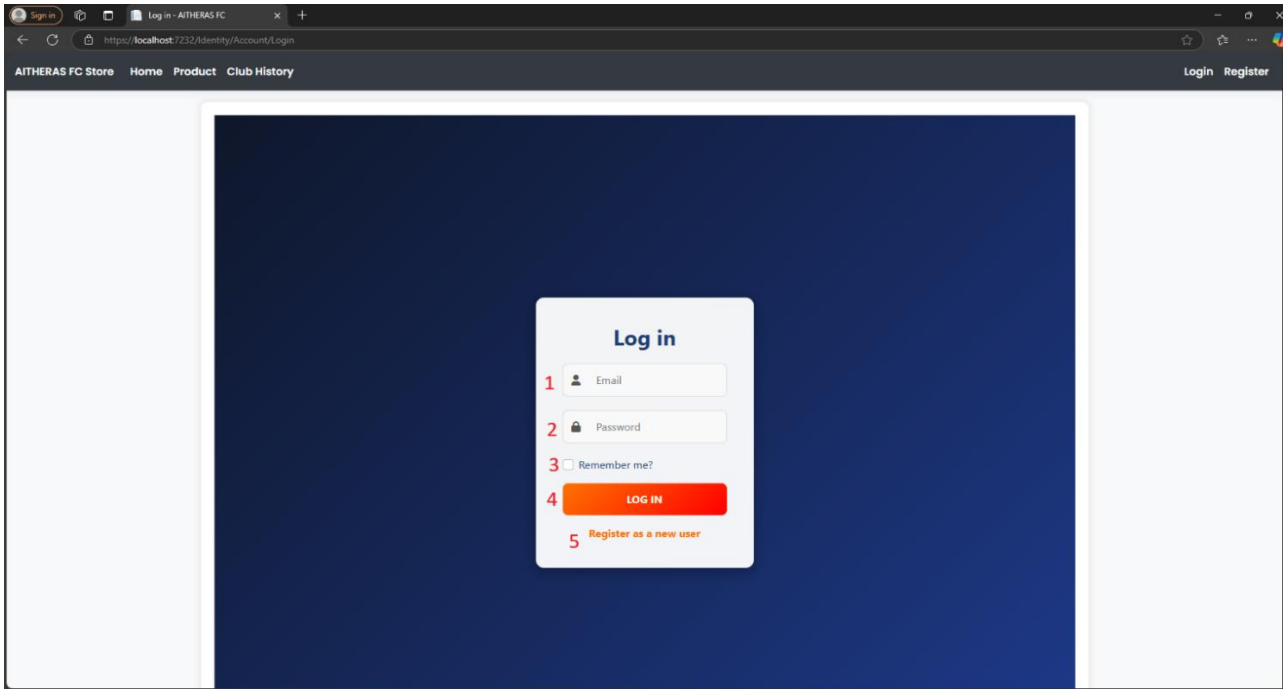
3. Product
   Ο χρήστης μπορεί να περιηγηθεί σε όλα τα διαθέσιμα προϊόντα του e-shop. Επιπλέον, μπορεί να φιλτράρει τα προϊόντα ανά τιμή ή κατηγορία και να δει λεπτομέρειες για κάθε ένα από αυτά.

4. Club History
   Περιλαμβάνει ιστορικές πληροφορίες για την ομάδα, όπως η ημερομηνία ίδρυσης και σημαντικές διακρίσεις. Η σελίδα είναι διαθέσιμη προς όλους τους χρήστες χωρίς απαίτηση σύνδεσης.

5. Login
   Ο χρήστης μπορεί να συνδεθεί χρησιμοποιώντας τα προσωπικά του στοιχεία. Μετά τη σύνδεση, του δίνεται η δυνατότητα να προσθέσει προϊόντα στο καλάθι και να πραγματοποιήσει παραγγελία.

6. Register
   Ο επισκέπτης έχει τη δυνατότητα να δημιουργήσει νέο λογαριασμό χρήστη, ώστε να αποκτήσει πρόσβαση σε επιπλέον λειτουργίες του e-shop.

Συνεχίζουμε με την σελίδα Σύνδεσης – Login form για τους εγγεγραμμένους χρήστες:



Εικόνα 29 – Φόρμα σύνδεσης στο E-shop

Σε αυτή τη σελίδα, έχουμε:

1. Email
   Πεδίο εισαγωγής της ηλεκτρονικής διεύθυνσης του χρήστη.
   Ο χρήστης πρέπει να πληκτρολογήσει το email με το οποίο έχει εγγραφεί στο σύστημα.

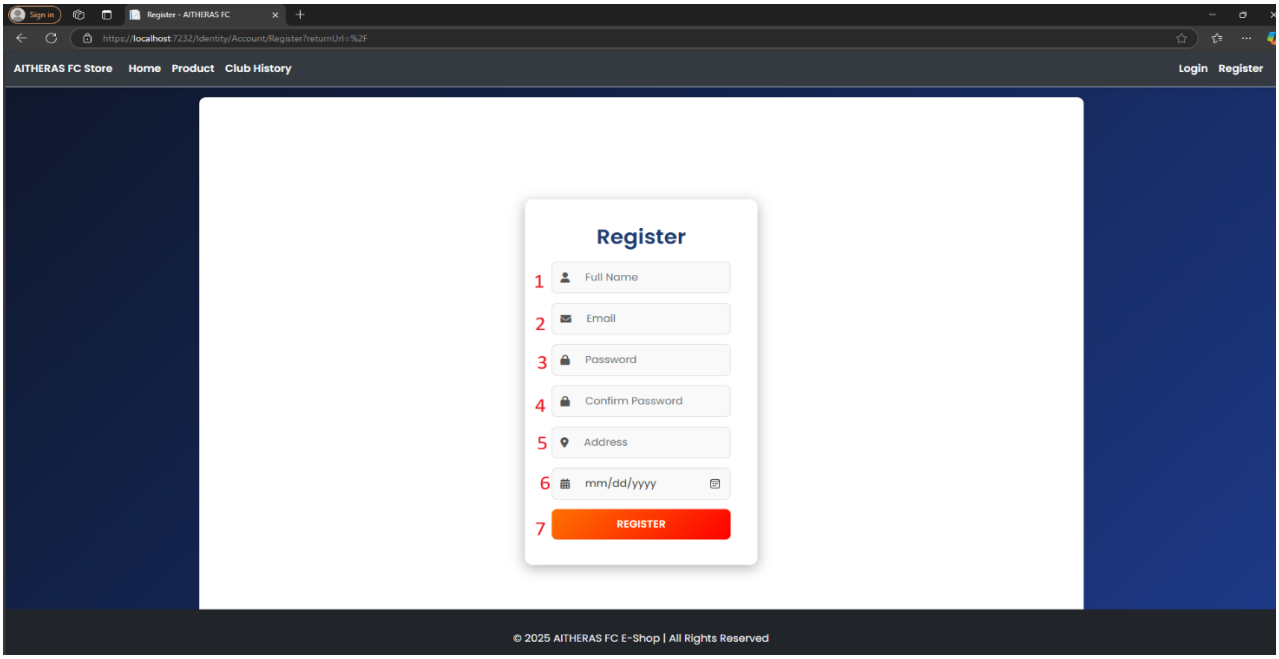
2. Password
   Πεδίο εισαγωγής του προσωπικού κωδικού πρόσβασης.
   Τα στοιχεία που πληκτρολογούνται είναι κρυφά για λόγους ασφαλείας.

3. Remember me?
   Επιλογή που επιτρέπει στο σύστημα να διατηρήσει τον χρήστη συνδεδεμένο για μεγαλύτερο χρονικό διάστημα.
   Χρήσιμο σε προσωπικές συσκευές.

4. LOG IN (κουμπί σύνδεσης)
   Κουμπί που ενεργοποιεί τη διαδικασία ελέγχου των στοιχείων.
   Αν τα στοιχεία είναι σωστά, ο χρήστης οδηγείται στην αρχική σελίδα.

5. Register as a new user
   Σύνδεσμος που οδηγεί σε φόρμα εγγραφής για νέους χρήστες.
   Χρησιμοποιείται όταν κάποιος δεν έχει ακόμη λογαριασμό.

Για όποιον χρήστη δεν έχει εγγραφεί, μεταφέρεται στη φόρμα Εγγραφής:



Εικόνα 30 – Φόρμα εγγραφής στο E-shop

1. Full Name
   Πεδίο εισαγωγής του ονοματεπώνυμου του χρήστη.
   Χρησιμοποιείται για προσωποποιημένη εμφάνιση και διαχείριση προφίλ.

2. Email
   Ηλεκτρονική διεύθυνση που θα συσχετιστεί με τον νέο λογαριασμό.
   Απαραίτητη για την ταυτοποίηση.

3. Password
   Πεδίο για την επιλογή του κωδικού πρόσβασης.
   Τα δεδομένα είναι κρυφά κατά την πληκτρολόγηση για λόγους ασφαλείας.

4. Confirm Password
   Επαναπληκτρολόγηση του κωδικού για επιβεβαίωση.
   Χρησιμοποιείται για να μειωθούν τα λάθη κατά την αρχική δημιουργία του λογαριασμού.

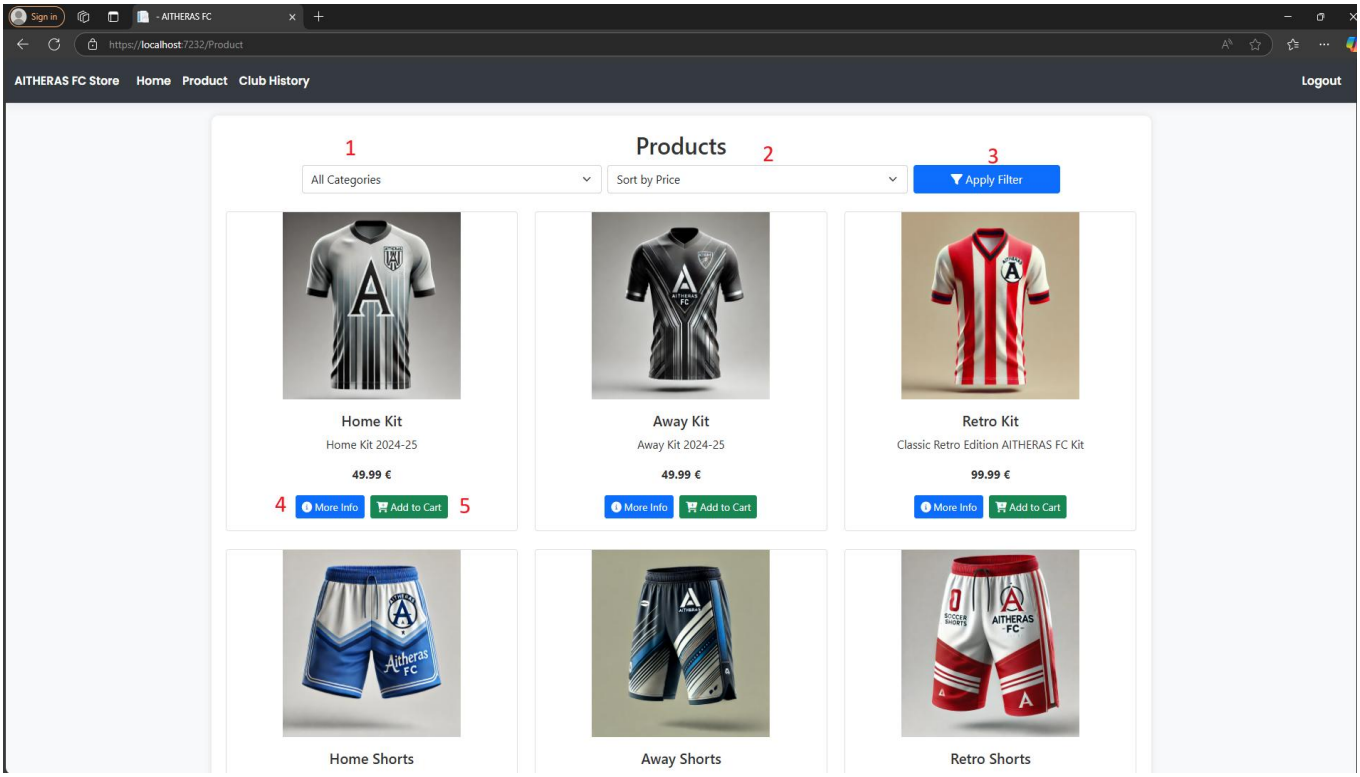
5. Address
   Διεύθυνση κατοικίας του χρήστη.

Χρησιμοποιείται για προσυμπληρωμένη διεύθυνση αποστολής κατά την ολοκλήρωση αγοράς.

6. <u>Ημερομηνία Γέννησης</u>
   Επιλογή ημερομηνίας μέσω date picker.
   Χρησιμοποιείται για τη δημιουργία του προφίλ.

7. <u>REGISTER (κουμπί εγγραφής)</u>
   Κουμπί που ενεργοποιεί τη διαδικασία δημιουργίας νέου λογαριασμού.
   Αν όλα τα πεδία είναι σωστά συμπληρωμένα, ο χρήστης καταχωρείται στο σύστημα και εισέρχεται αυτόματα.

Αφού ο χρήστης καταφέρει να συνδεθεί, μόλις πάει στο Tab με τα προϊόντα, θα αντικρίσει τα παρακάτω:



Εικόνα 31 – Σελίδα προϊόντων του E-shop

1. <u>All Categories (Dropdown φίλτρου κατηγορίας)</u>
   Επιτρέπει στο χρήστη να επιλέξει μία συγκεκριμένη κατηγορία προϊόντων (π.χ. Jerseys, Shorts) για εμφάνιση.
   Χρησιμοποιείται για φιλτράρισμα των αποτελεσμάτων με βάση τον τύπο προϊόντος.

2. Sort by Price (Dropdown ταξινόμησης)
   Δίνει τη δυνατότητα ταξινόμησης των προϊόντων με βάση την τιμή, αυξανόμενα ή φθίνουσα.
   Αναβαθμίζει την εμπειρία πλοήγησης, επιτρέποντας στον χρήστη να εντοπίσει ευκολότερα οικονομικές ή premium επιλογές.

3. Apply Filter (Κουμπί εφαρμογής φίλτρων)
   Εφαρμόζει τα φίλτρα που έχει επιλέξει ο χρήστης στα πεδία κατηγορίας και ταξινόμησης.
   Ανανεώνει τη λίστα των προϊόντων με βάση τις επιθυμητές επιλογές.

4. More Info (Κουμπί λεπτομερειών προϊόντος)
   Οδηγεί στη σελίδα με τις αναλυτικές πληροφορίες του προϊόντος (π.χ. περιγραφή, χαρακτηριστικά, φωτογραφίες).
   Συμβάλλει στη λήψη καλύτερης αγοραστικής απόφασης.

5. Add to Cart (Κουμπί προσθήκης στο καλάθι)
   Προσθέτει το επιλεγμένο προϊόν στο καλάθι αγορών του χρήστη.

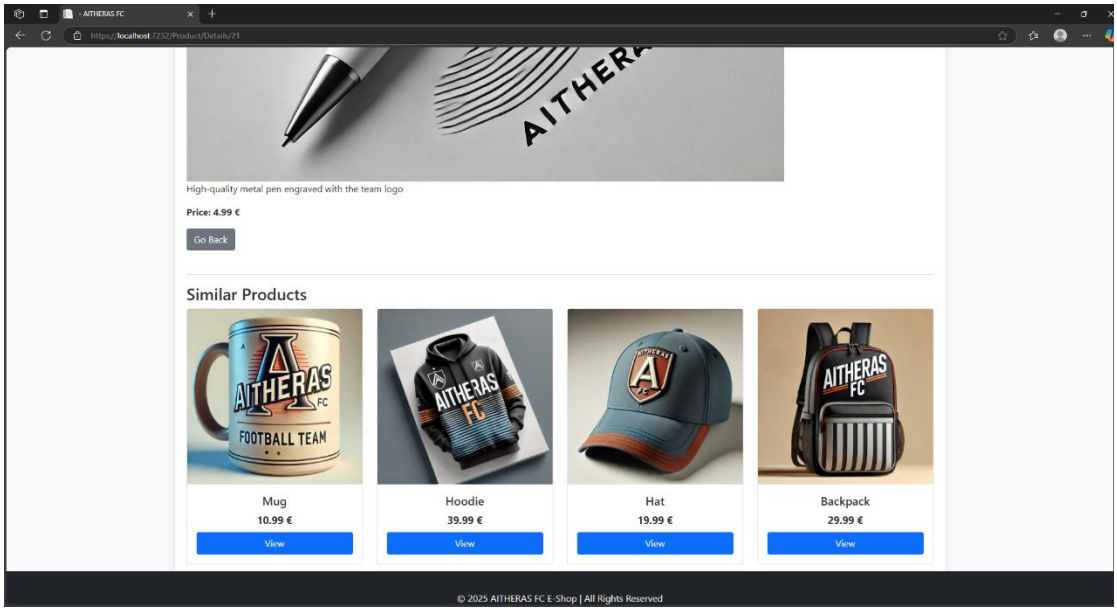
Σε περίπτωση που ο χρήστης κλικάρει το κουμπί «More Info», η εφαρμογή εμφανίζει το προϊόν με μεγέθυνση εικόνας και από κάτω έναν απλό αλλά αποτελεσματικό μηχανισμό σύστασης προϊόντων, ο οποίος λειτουργεί με βάση το ιστορικό περιήγησης του χρήστη μέσω των session IDs. Η υλοποίηση δεν απαιτεί να είναι ο χρήστης συνδεδεμένος στο σύστημα.
Η βασική λογική λειτουργίας είναι η εξής:

1. Κάθε φορά που ο χρήστης επισκέπτεται μία σελίδα προϊόντος, το ID αυτού καταγράφεται στο session.
2. Σε κάθε εμφάνιση πρότασης, δίνεται προτεραιότητα στην εξατομίκευση, δηλαδή προβάλλονται πρώτα τα προϊόντα της ίδιας κατηγορίας που έχει επισκεφτεί ο χρήστης τελευταία.
3. Σε περίπτωση που τα session-based προϊόντα είναι λιγότερα από 4, εμφανίζονται πρώτα τα εξατομικευμένα και στη συνέχεια συμπληρώνονται τυχαία (για αποφυγή στατικότητας) άλλα προϊόντα της ίδιας κατηγορίας.

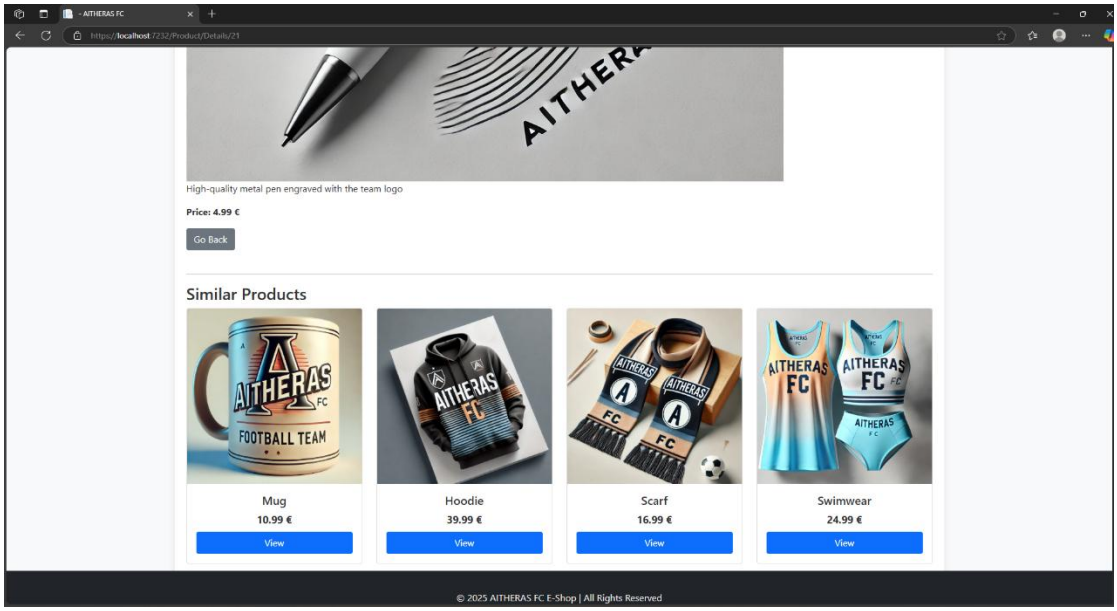
Ακολουθεί ένα παράδειγμα επεξήγησης της παραπάνω λειτουργίας:

- Έστω ότι ένας χρήστης έχει δει δύο προϊόντα της κατηγορίας Accessories, όπως ένα Mug και ένα Hoodie. Στη συνέχεια επισκέπτεται ακόμη ένα προϊόν της ίδιας κατηγορίας, π.χ. ένα Pen. Η εφαρμογή θα προτείνει κατά σειρά πρώτα το Mug και

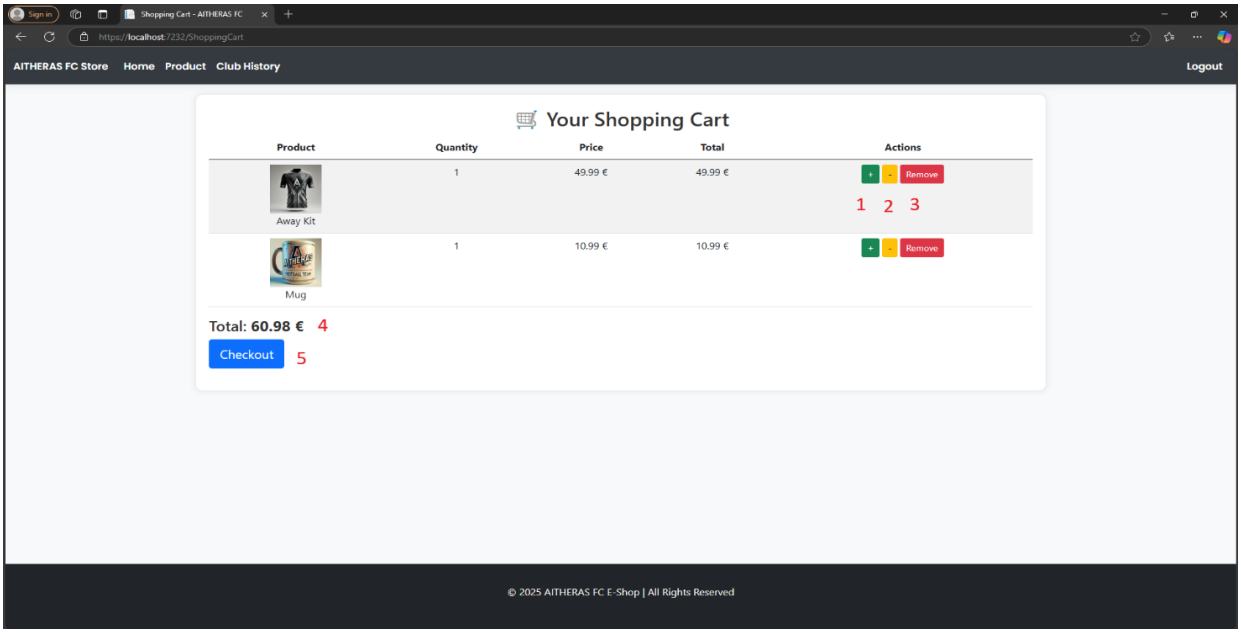
το Hoodie (ως πρόσφατα session-based προϊόντα), ενώ οι υπόλοιπες δύο θέσεις θα συμπληρωθούν με τυχαία προϊόντα της κατηγορίας Accessories.



- Αν ο χρήστης κάνει ανανέωση (refresh) της σελίδας, τα δύο πρώτα προϊόντα θα παραμείνουν σταθερά, ενώ τα υπόλοιπα δύο μπορεί να αλλάξουν, ώστε να διατηρείται μια αίσθηση δυναμικότητας και ποικιλίας στις προτάσεις.



Αφού ο χρήστης έχει δει τα χαρακτηριστικά των προϊόντων που επιθυμεί να αγοράσει και πατήσει την επιλογή «Add to Cart», τότε γίνεται αυτόματο Redirect στη σελίδα του καλαθιού, στην οποία φαίνονται όλα τα επιλεγμένα προς αγορά προϊόντα. Ο χρήστης, δηλαδή, αντικρίζει το παρακάτω:



Εικόνα 32 – Σελίδα καλαθιού

1. <u>Αύξηση Ποσότητας</u>
   Κουμπί που αυξάνει την ποσότητα του συγκεκριμένου προϊόντος στο καλάθι κατά μία μονάδα.
   Η τιμή στη στήλη "Total" ενημερώνεται αυτόματα.

2. <u>Μείωση Ποσότητας</u>
   Κουμπί που μειώνει την ποσότητα του προϊόντος κατά μία μονάδα.
   Εάν η ποσότητα φτάσει στο μηδέν, το προϊόν μπορεί να αφαιρεθεί εντελώς από το καλάθι.

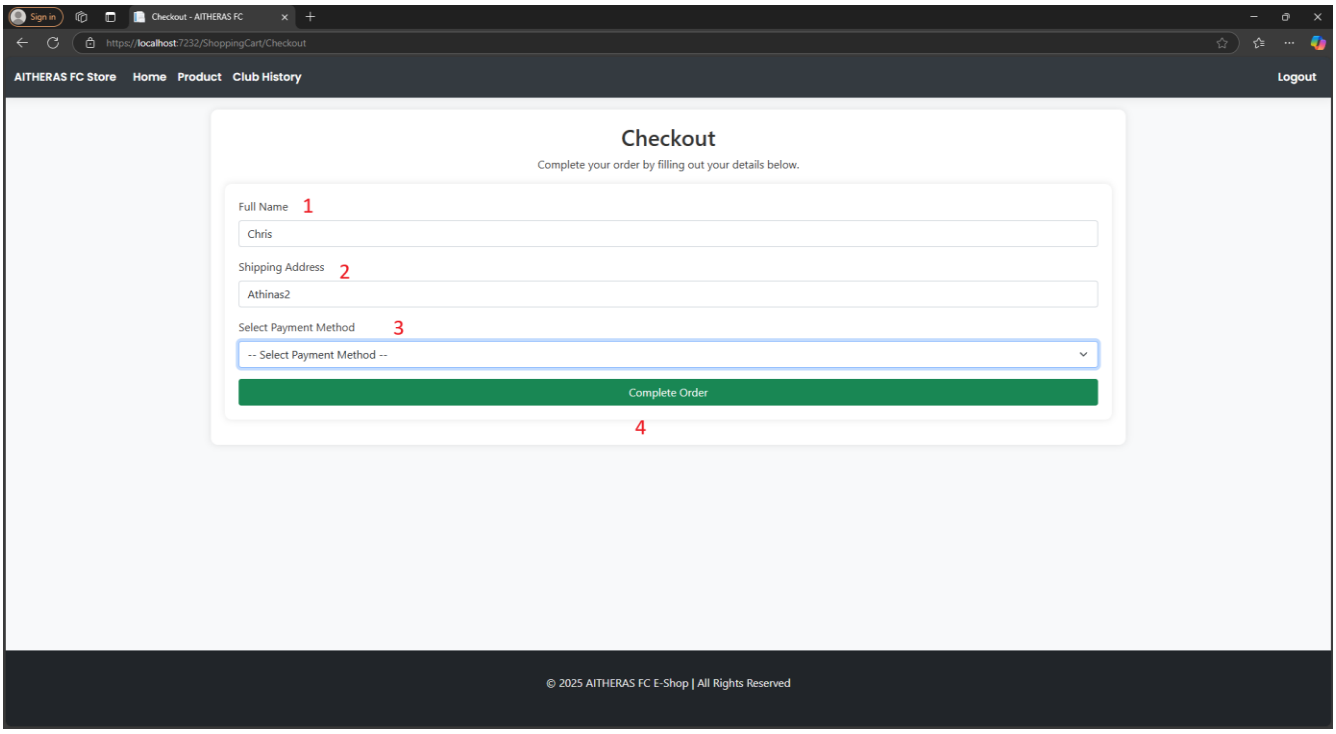
3. <u>Αφαίρεση Προϊόντος</u>
   Διαγράφει πλήρως το συγκεκριμένο προϊόν από το καλάθι.
   Χρήσιμο όταν ο χρήστης αλλάξει γνώμη για την αγορά του.

4. <u>Total</u>
   Εμφανίζει το συνολικό ποσό της παραγγελίας με βάση τα προϊόντα και τις ποσότητές τους.

Το ποσό ενημερώνεται δυναμικά καθώς αλλάζει το περιεχόμενο του καλαθιού.

5. Checkout
   Κουμπί που οδηγεί στη διαδικασία ολοκλήρωσης της παραγγελίας.
   Είναι το τελευταίο βήμα πριν την υποβολή της παραγγελίας.

Έχοντας κλικάρει στο κουμπί «Checkout», ο χρήστης τώρα έχει ανακατευθυνθεί στη φόρμα συμλήρωσης των στοιχείων του, προκειμένου να ολοκληρωθεί η αγορά:



Εικόνα 33 – Σελίδα Checkout (1/2)

1. Full Name
   Το ονοματεπώνυμο του χρήστη.
   Το πεδίο είναι προσυμπληρωμένο με τα στοιχεία που εισήχθησαν κατά την εγγραφή, ώστε να διευκολυνθεί η διαδικασία παραγγελίας.
   Ωστόσο, ο χρήστης έχει τη δυνατότητα να το τροποποιήσει, εφόσον επιθυμεί να πραγματοποιήσει την παραγγελία με διαφορετικά στοιχεία.

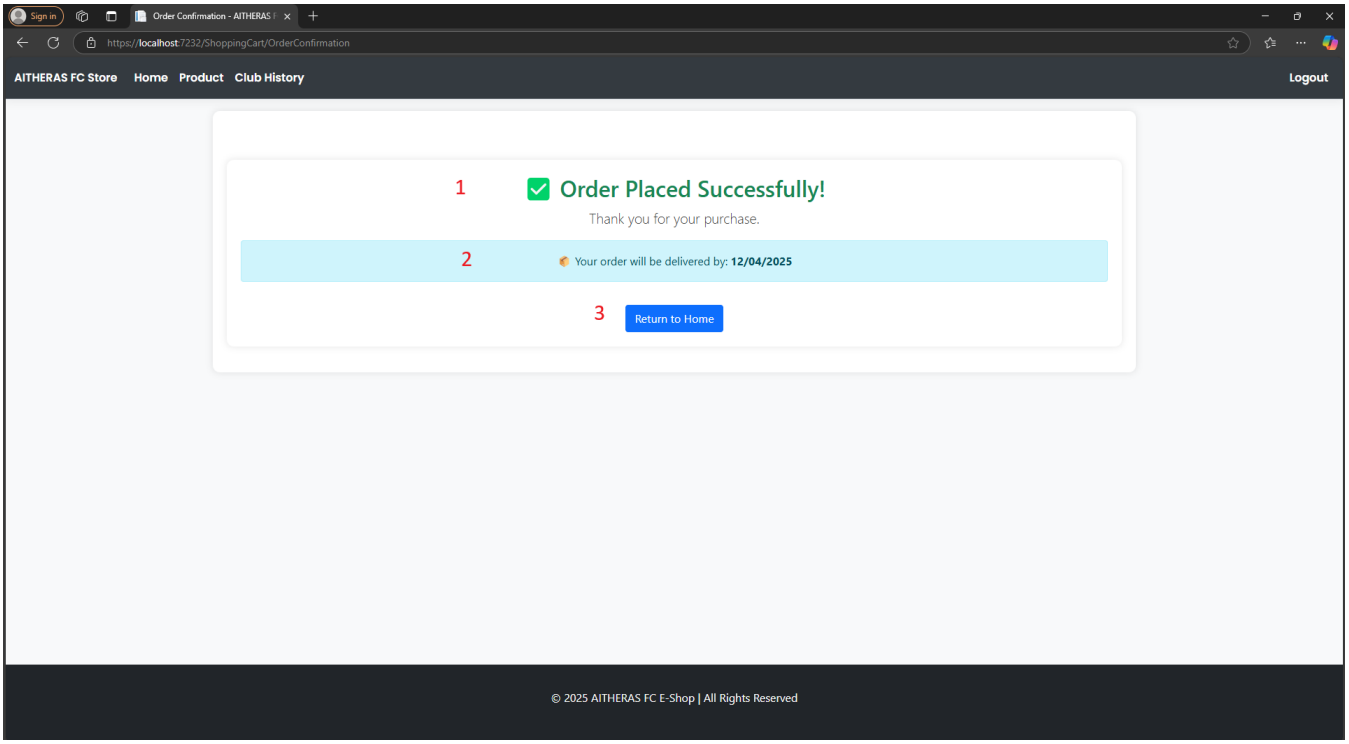
2. Shipping Address
   Η διεύθυνση αποστολής των προϊόντων.

Και αυτό το πεδίο είναι προσυμπληρωμένο από τη φόρμα εγγραφής, αλλά μπορεί να επεξεργαστεί από τον χρήστη για αποστολή σε άλλη τοποθεσία.

3. Select Payment Method
   Επιλογή τρόπου πληρωμής μέσω αναπτυσσόμενου μενού.
   Ο χρήστης μπορεί να επιλέξει ανάμεσα σε δύο διαθέσιμες μεθόδους:
   Credit Card (πληρωμή με κάρτα) ή *Cash on Delivery* (αντικαταβολή κατά την παράδοση).
   Η επιλογή αυτή επηρεάζει τον τρόπο διεκπεραίωσης της παραγγελίας.

4. Complete Order
   Κουμπί επιβεβαίωσης και ολοκλήρωσης της παραγγελίας.
   Με την επιλογή του, το σύστημα καταχωρεί την παραγγελία, αποθηκεύει τα σχετικά στοιχεία και οδηγεί τον χρήστη στη σελίδα επιβεβαίωσης, εφόσον δεν υπάρχουν σφάλματα.

Εφόσον ο χρήστης επιλέξει ως τρόπο πληρωμής την αντικαταβολή (Cash on Delivery) κατά τη διαδικασία checkout, μετά την επιτυχή καταχώρηση της παραγγελίας εμφανίζεται η σελίδα επιβεβαίωσης. Η σελίδα ενημερώνει τον χρήστη ότι η παραγγελία του έχει καταχωρηθεί επιτυχώς και του εμφανίζει την εκτιμώμενη ημερομηνία παράδοσης:



Εικόνα 34 – Σελίδα επιβεβαίωσης Αγοράς

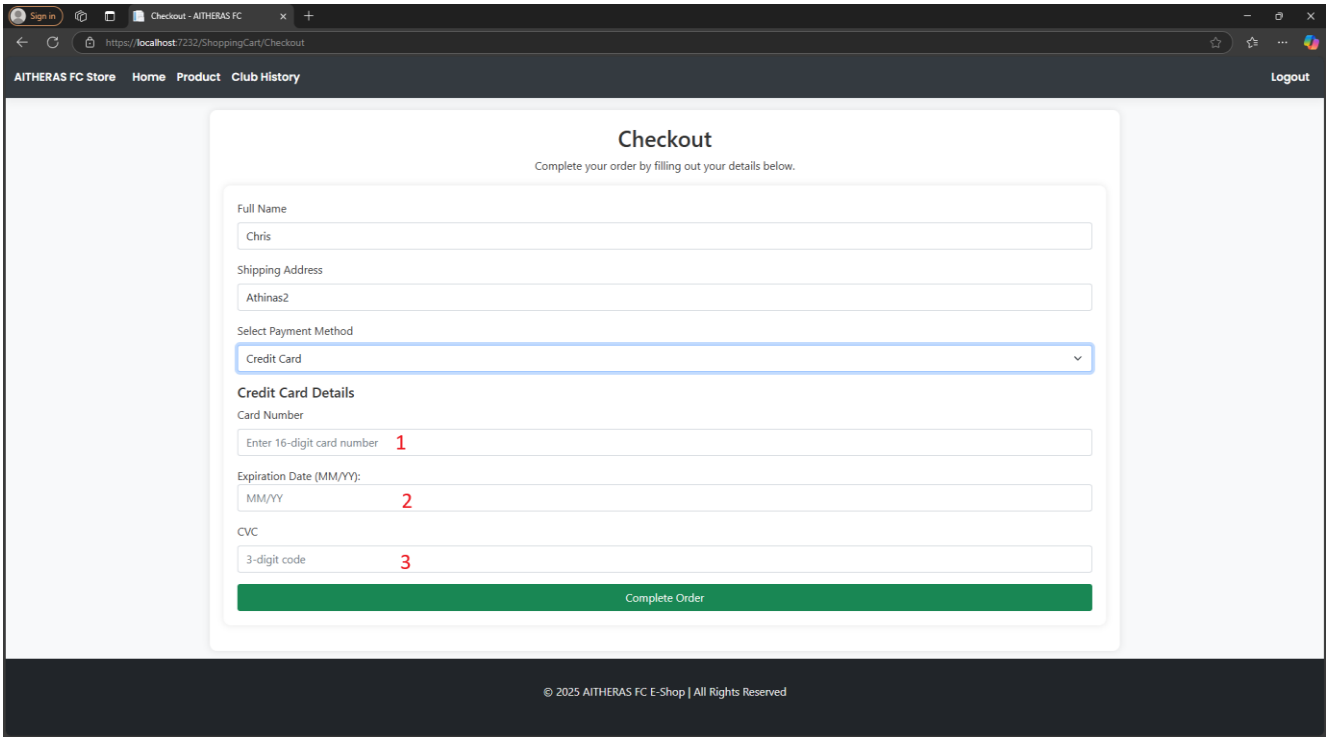
1. Order Placed Successfully!
   Μήνυμα επιτυχούς υποβολής της παραγγελίας.
   Συνοδεύεται από επιβεβαιωτική ένδειξη και ευχαριστήριο μήνυμα προς τον χρήστη.

2. Εκτιμώμενη Ημερομηνία Παράδοσης
   Εμφανίζει την ημερομηνία κατά την οποία αναμένεται να παραδοθεί η παραγγελία.
   Η ημερομηνία υπολογίζεται δυναμικά με βάση την ημέρα υποβολής και πιθανή πολιτική αποστολής του e-shop.

3. Return to Home
   Κουμπί επιστροφής στην αρχική σελίδα του καταστήματος.
   Δίνει τη δυνατότητα στον χρήστη να συνεχίσει την πλοήγησή του ή να ξεκινήσει νέα παραγγελία.

Εάν τώρα ο χρήστης επιλέξει ως μέθοδο πληρωμής την πιστωτική κάρτα, εμφανίζεται δυναμικά μία επιπλέον ενότητα στη σελίδα Checkout, όπου ζητούνται τα στοιχεία της κάρτας.

Η πληρωμή πραγματοποιείται άμεσα με την ολοκλήρωση της παραγγελίας και ο χρήστης μεταφέρεται στην σελίδα επιβεβαίωσης της αγοράς:



Εικόνα 35 – Σελίδα Checkout (2/2)

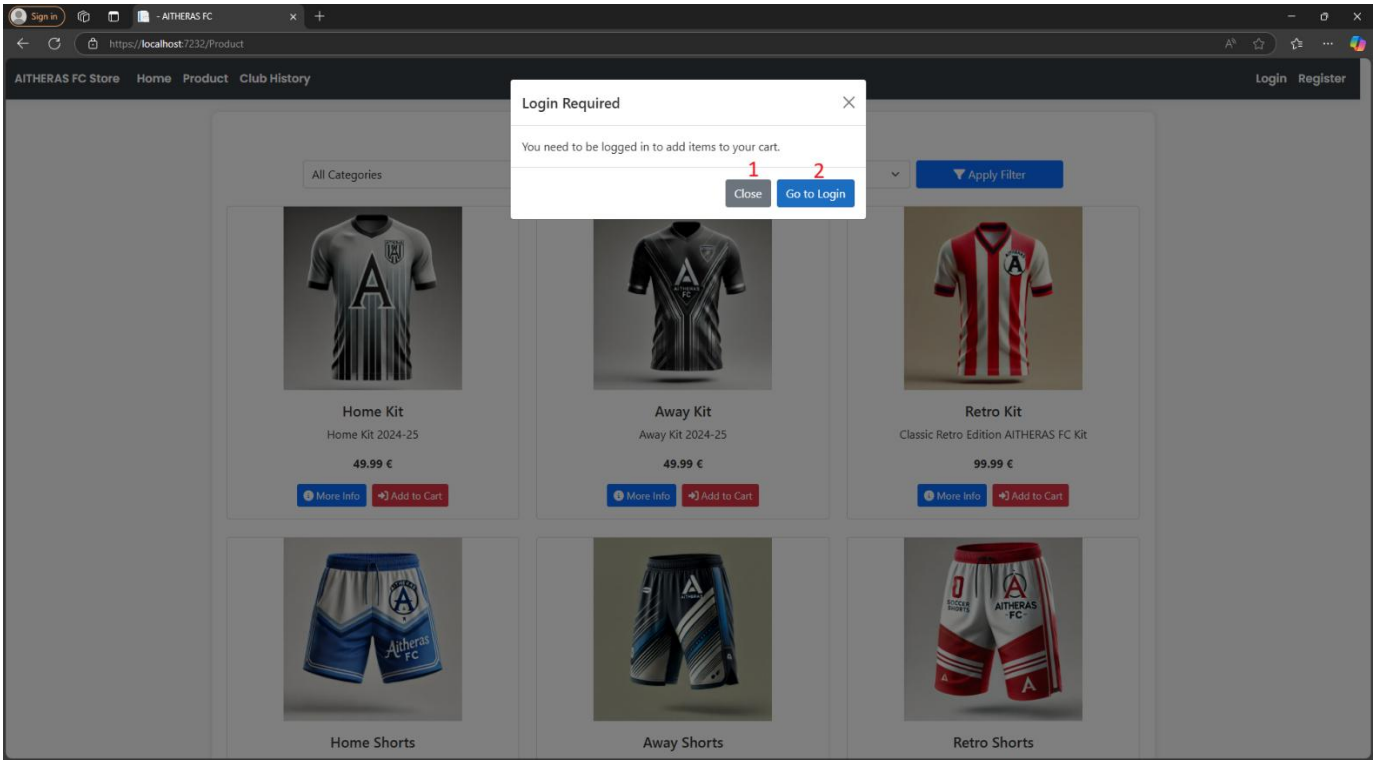
1. Card Number
   Πεδίο εισαγωγής του αριθμού της κάρτας (16 ψηφία, χωρίς κενά).
   Αντιστοιχεί στον βασικό αριθμό που εμφανίζεται στην εμπρόσθια όψη της κάρτας.

2. Expiration Date (MM/YY)
   Εισαγωγή της ημερομηνίας λήξης της κάρτας, σε μορφή Μήνας/Έτος (π.χ. 06/27).
   Χρησιμοποιείται για την επικύρωση της ισχύος της κάρτας.

3. CVC
   Τριψήφιος κωδικός ασφαλείας (Card Verification Code).
   Αναγράφεται συνήθως στο πίσω μέρος της κάρτας και χρησιμοποιείται για επαλήθευση της κάρτας από τον κάτοχό της.

Με αυτό τον τρόπο, ο χρήστης έχει ολοκληρώσει με επιτυχία την αγορά του!

Σε αυτό το σημείο, αξίζει να αναφερθεί, ότι εάν ένας απλός χρήστης, ο οποίος δεν έχει πραγματοποιήσει Login στο E-shop, προσπαθήσει να προσθέσει ένα προϊόν στο καλάθι του, τότε θα αντικρίσει το παρακάτω μήνυμα:

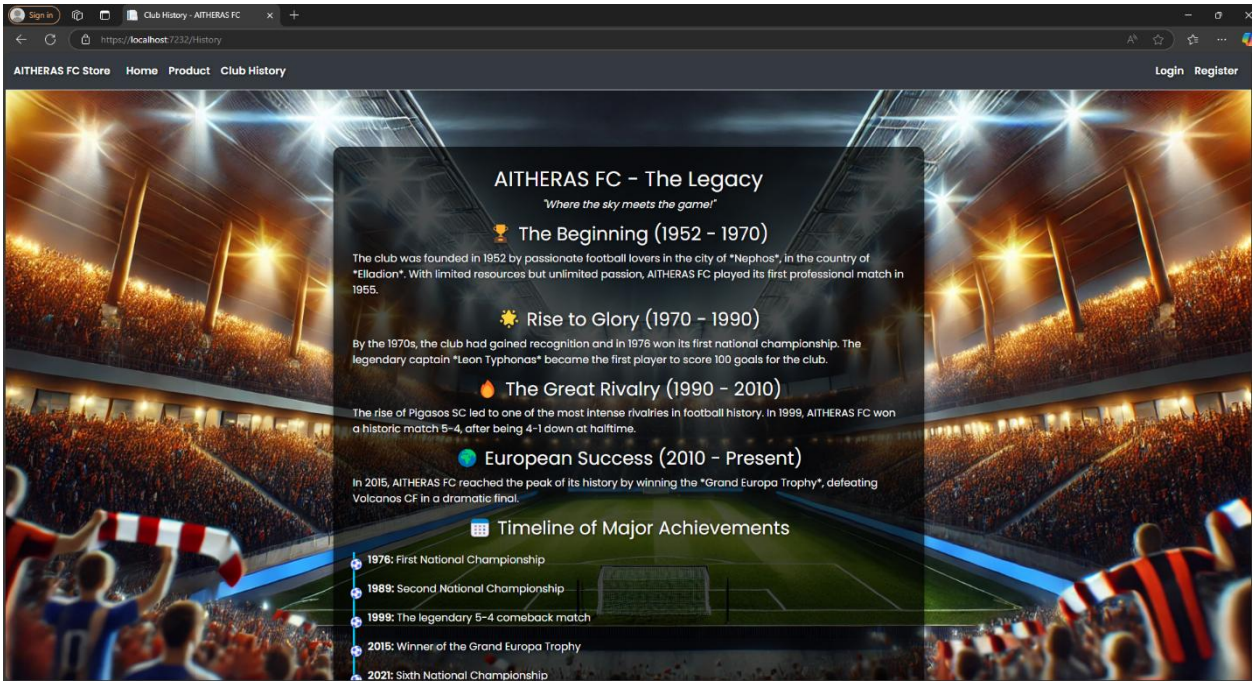


Εικόνα 36 – Σελίδα Επισκέπτη

1. <u>Close</u>
   Κουμπί που κλείνει το παράθυρο ειδοποίησης χωρίς να πραγματοποιηθεί καμία ενέργεια.
   Ο χρήστης παραμένει στη σελίδα των προϊόντων χωρίς να προσθέσει κάτι στο καλάθι.

2. <u>Go to Login</u>
   Κουμπί που μεταφέρει τον χρήστη στη σελίδα Login.
   Αφού πραγματοποιηθεί επιτυχής σύνδεση, μπορεί να επιστρέψει και να προσθέσει προϊόντα στο καλάθι.

Ολοκληρώνοντας το εγχειρίδιο χρήσης της εφαρμογής από τον απλό user, αξίζει να αναφερθεί και η παρακάτω σελίδα, η οποία αποτελεί ένα «κόσμημα» για το E-shop, καθώς πέρα από τα εντυπωσιακά της εφέ, περιέχει και σημαντικές πληροφορίες για την ομάδα:



Εικόνα 37 – Club History tab

Η σελίδα Club History παρουσιάζει τη μακρά και ένδοξη πορεία του συλλόγου AITHERAS FC, από την ίδρυσή του έως και τη σύγχρονη εποχή, με αφηγηματική δομή και ξεχωριστές ενότητες για κάθε δεκαετία.

Η πρόσβαση στη σελίδα αυτή δεν απαιτεί σύνδεση ή κάποιον admin ρόλο. Είναι ελεύθερα προσβάσιμη τόσο σε επισκέπτες όσο και σε εγγεγραμμένους χρήστες, ώστε όλοι να μπορούν να γνωρίσουν το παρελθόν και τις διακρίσεις του συλλόγου.

4.3 Χρήση της Εφαρμογής από Διαχειριστή (admin)

Η πλατφόρμα του AITHERAS FC Store παρέχει ειδικά δικαιώματα σε χρήστες με ρόλο Admin, επιτρέποντας την εποπτεία και διαχείριση κρίσιμων στοιχείων του e-shop, όπως τα προϊόντα και οι χρήστες.

Η πρόσβαση του διαχειριστή στο σύστημα πραγματοποιείται μέσω της ίδιας φόρμας σύνδεσης (Login page) που χρησιμοποιούν και οι απλοί χρήστες. Επίσης, οι βασικές σελίδες, όπως:

- η Αρχική σελίδα (Home page),

- η σελίδα Club History,
- και η ίδια η φόρμα σύνδεσης και εγγραφής,

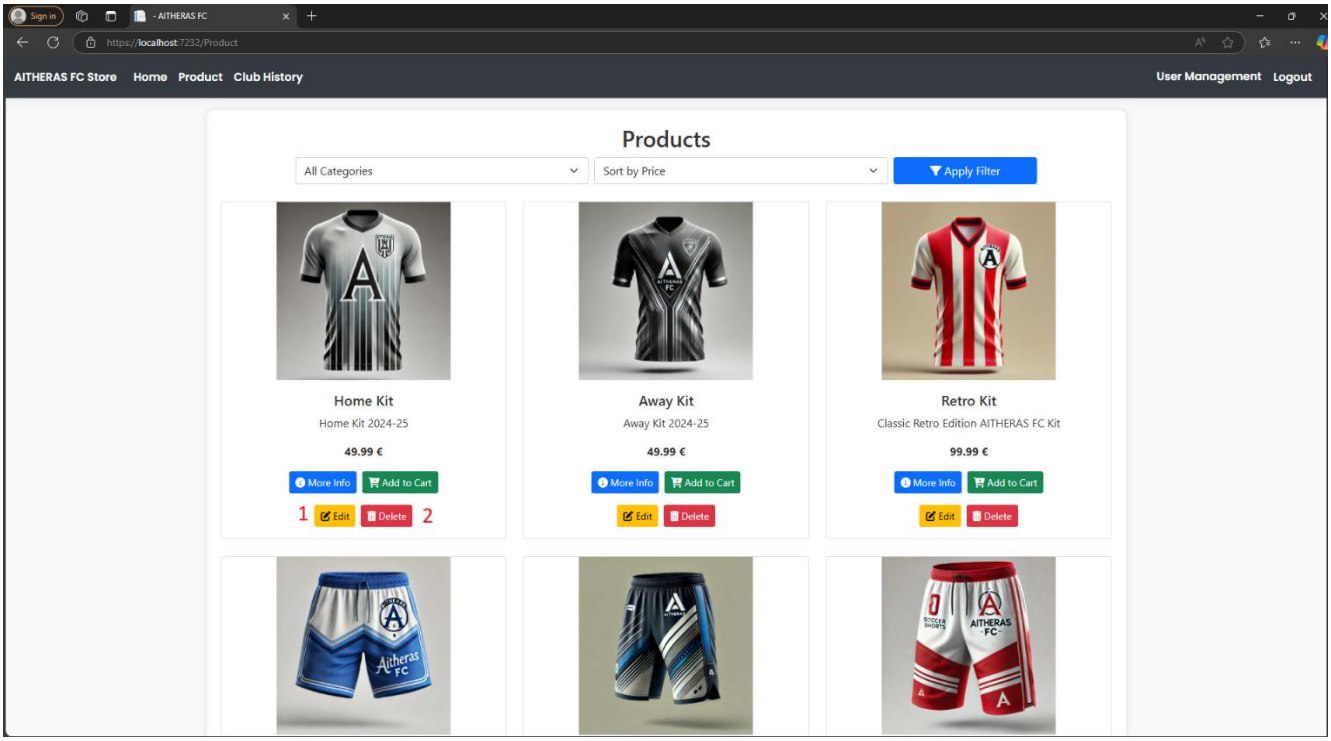
είναι κοινές τόσο για τους απλούς χρήστες όσο και για τους διαχειριστές.

Η βασική διαφοροποίηση του ρόλου του admin εντοπίζεται κυρίως στα εξής σημεία:

- Προβολή και διαχείριση των προϊόντων (Products tab):
  Εκτός από την απλή περιήγηση και προσθήκη προϊόντων στο καλάθι, ο admin διαθέτει επιπλέον λειτουργίες όπως η επεξεργασία και η διαγραφή υπαρχόντων προϊόντων.

- Εμφάνιση πρόσθετου tab “User Management” στο επάνω μενού πλοήγησης:
  Αυτή η ενότητα εμφανίζεται αποκλειστικά σε χρήστες με admin ρόλο και επιτρέπει την εποπτεία και διαχείριση των εγγεγραμμένων χρηστών της πλατφόρμας.

Πάμε να εξετάσουμε την πρώτη διαφοροποίηση, δηλαδή τις διαφορές στο Products tab. Συγκεκριμένα, όταν ο χρήστης συνδεθεί με δικαιώματα διαχειριστή, στη σελίδα προϊόντων εμφανίζονται επιπλέον λειτουργίες διαχείρισης κάτω από κάθε προϊόν. Αυτές οι επιλογές δεν είναι ορατές στους απλούς χρήστες και επιτρέπουν την άμεση τροποποίηση ή αφαίρεση προϊόντων από το σύστημα:

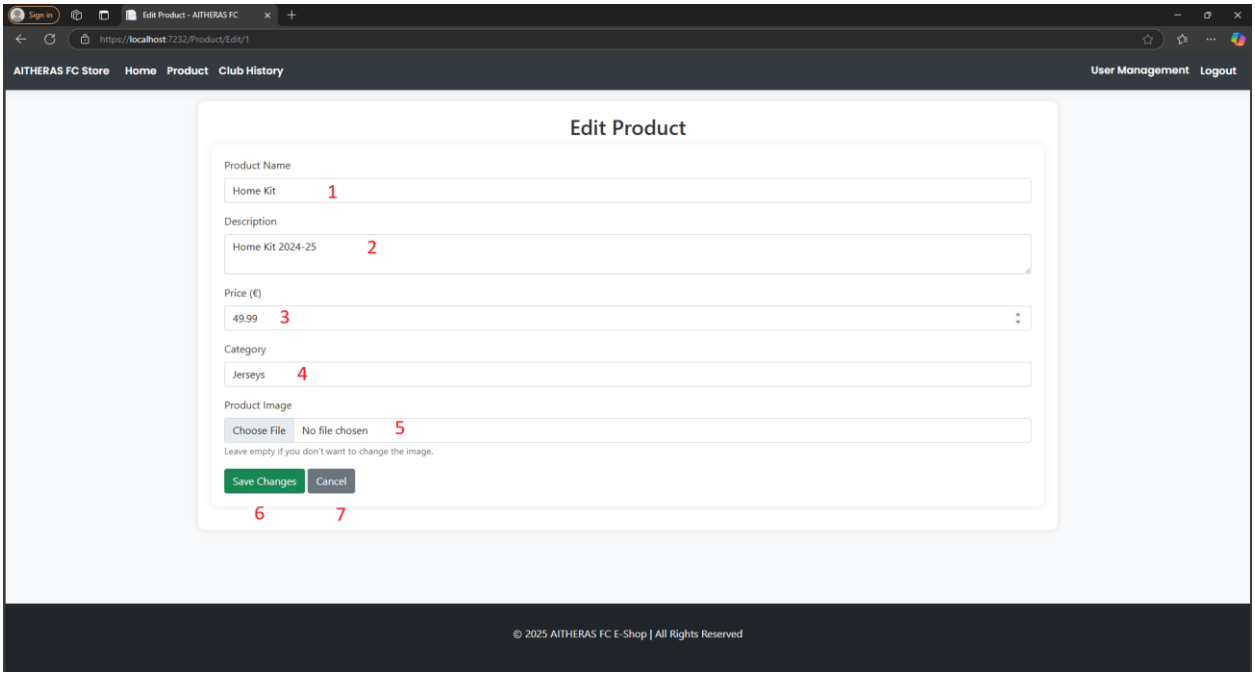


Εικόνα 38 – Product page ως admin

1. Edit
   Το κουμπί Edit ανοίγει τη φόρμα επεξεργασίας του επιλεγμένου προϊόντος.
   Ο διαχειριστής μπορεί να τροποποιήσει πληροφορίες όπως όνομα, περιγραφή, τιμή, κατηγορία ή την εικόνα εμφάνισης του προϊόντος.

2. Delete
   Το κουμπί Delete διαγράφει το συγκεκριμένο προϊόν από τη βάση δεδομένων.

Εφόσον επιλεχθεί το κουμπί Edit, ανοίγει η σελίδα τροποποίησης. Αυτή περιλαμβάνει τα παρακάτω:



Εικόνα 39 – Edit page προϊόντων ως admin

1. Product Name
   Πεδίο τροποποίησης του ονόματος του προϊόντος.
   Θα εμφανίζεται ως τίτλος στη σελίδα προϊόντων και στις λεπτομέρειες του προϊόντος.

2. Description
   Αναλυτική περιγραφή του προϊόντος.
   Εμφανίζεται στη σελίδα "More Info" για κάθε προϊόν και βοηθά στην ενημέρωση του χρήστη.

3. Price (€)
   Τιμή πώλησης του προϊόντος σε ευρώ.
   Πρέπει να περιλαμβάνει δεκαδικά ψηφία, όπου χρειάζεται (π.χ. 49.99).

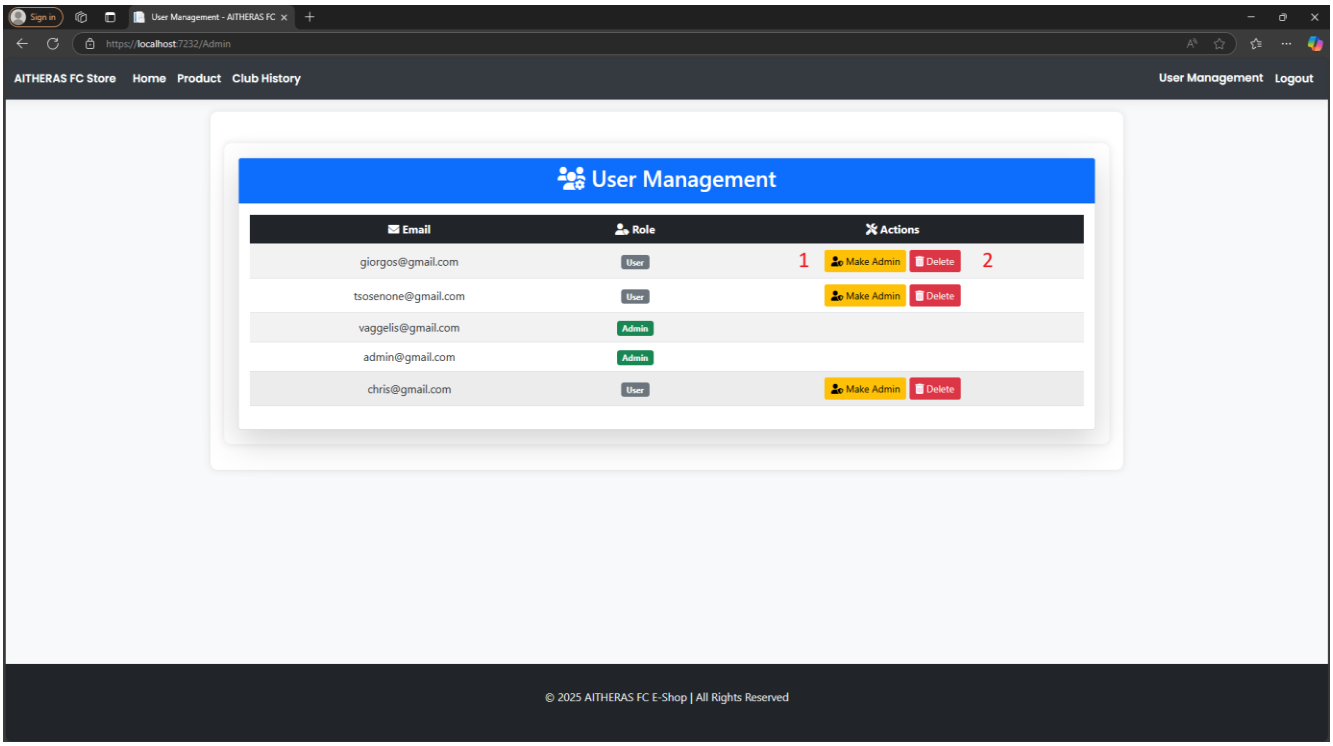
4. Category
   Κατηγορία στην οποία ανήκει το προϊόν (π.χ. Jerseys, Shorts, Accessories).
   Χρησιμοποιείται για φιλτράρισμα και ομαδοποίηση στην προβολή προϊόντων.

5. Product Image
   Επιλογή νέας εικόνας για το προϊόν μέσω αρχείου (Upload).

6. Save Changes
   Πράσινο κουμπί αποθήκευσης των τροποποιήσεων.
   Μόλις πατηθεί, οι αλλαγές καταχωρούνται στη βάση δεδομένων και ο admin επιστρέφει στη σελίδα προϊόντων.

7. Cancel
   Γκρι κουμπί ακύρωσης.
   Ακυρώνει την επεξεργασία και επιστρέφει πίσω χωρίς να γίνουν αλλαγές.

Συνεχίζουμε με το tab το οποίο εμφανίζεται μόνο όταν συνδέεσαι ως admin, το User Management, το οποίο είναι ορατό στο επάνω μενού σε οποιαδήποτε σελίδα και αν βρισκόμαστε:



Εικόνα 40 – User Management page ως admin

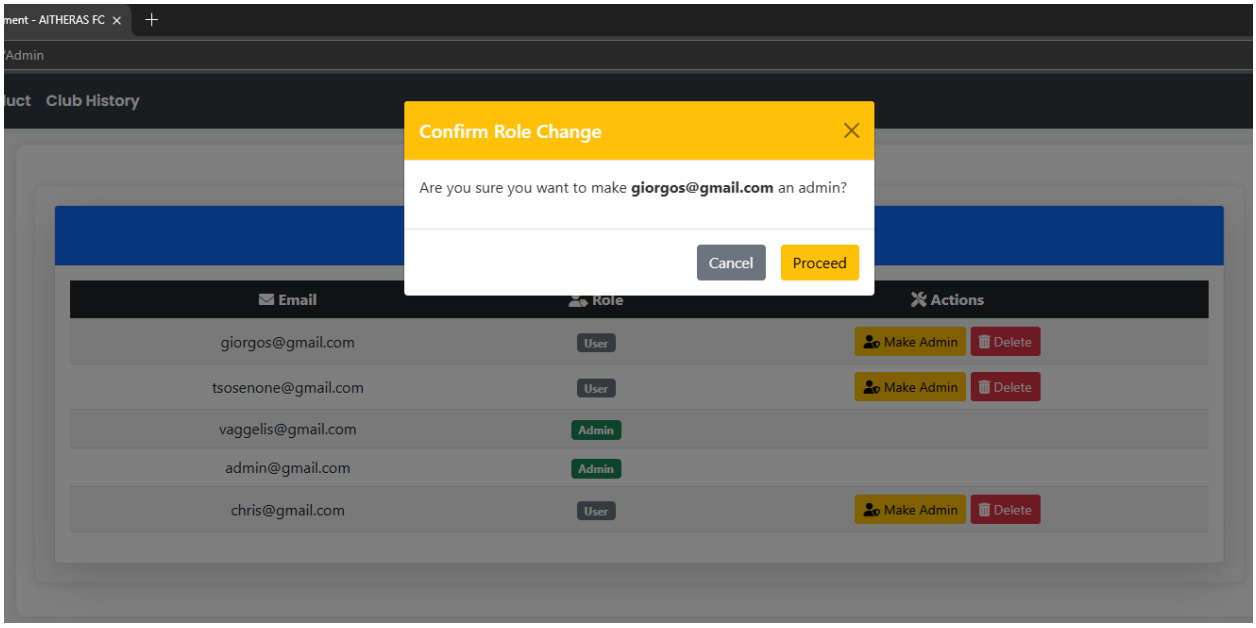
Το tab User Management επιτρέπει στον διαχειριστή του AITHERAS FC Store να έχει πλήρη εικόνα και έλεγχο των εγγεγραμμένων χρηστών του συστήματος.

Η σελίδα προβάλλει σε μορφή πίνακα τα εξής στοιχεία:

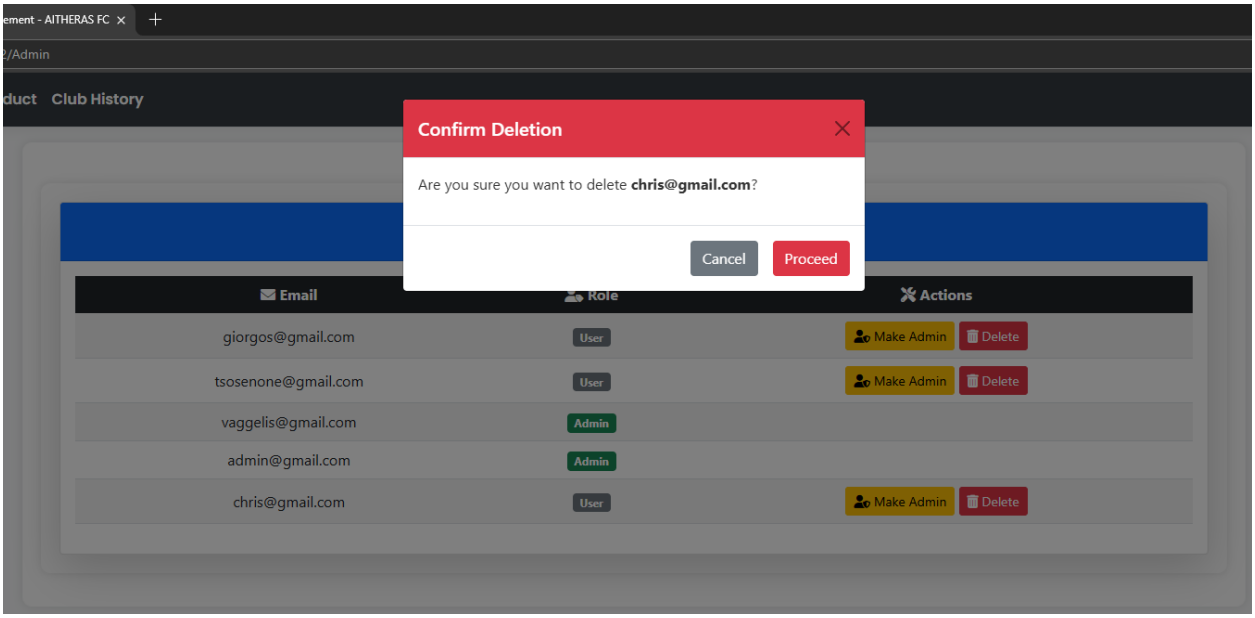
- Email του χρήστη
- Ρόλος του χρήστη (User ή Admin)
- Διαθέσιμες ενέργειες (Actions) για κάθε χρήστη

1. Make Admin
   Κουμπί που προάγει τον χρήστη από απλό χρήστη (User) σε διαχειριστή (Admin). Μόλις πατηθεί,εμφανίζεται μήνυμα επιβεβαίωσης προαγωγής του ρόλου του χρήστη από User σε Admin και σε περίπτωση επιβεβαίωσης ο χρήστης αποκτά πλήρη πρόσβαση στα διαχειριστικά tabs και λειτουργίες.
   Το κουμπί αυτό εμφανίζεται μόνο για χρήστες που δεν είναι ήδη Admin.

2. Delete
   Κουμπί διαγραφής χρήστη από το σύστημα.
   Η ενέργεια αφαιρεί εντελώς τον λογαριασμό του χρήστη, καθώς και την πρόσβασή του στο e-shop. Συνοδεύεται από επιβεβαιωτικό μήνυμα για αποφυγή τυχαίας διαγραφής.



Εικόνα 41 – Επιβεβαίωση αναβάθμισης ρόλου ως admin



Εικόνα 42 – Επιβεβαίωση διαγραφής εγγεγραμμένου χρήστη

Κεφάλαιο 5ο

5 Συμπεράσματα – Βελτιώσεις

Η παρούσα πτυχιακή εργασία είχε ως στόχο την ανάπτυξη μιας πλήρους και λειτουργικής διαδικτυακής εφαρμογής ηλεκτρονικού εμπορίου για λογαριασμό μιας φανταστικής ποδοσφαιρικής ομάδας, με την ονομασία AITHERAS FC. Μέσα από μια οργανωμένη και συστηματική διαδικασία σχεδίασης και υλοποίησης, δημιουργήθηκε ένα e-shop που καλύπτει τόσο τις βασικές όσο και πιο σύνθετες λειτουργίες ενός σύγχρονου ηλεκτρονικού καταστήματος. Από την περιήγηση των χρηστών και την προσθήκη προϊόντων στο καλάθι έως τη διαχείριση προϊόντων και χρηστών από διαχειριστές, η εφαρμογή προσφέρει μια ολοκληρωμένη εμπειρία.

Κατά την ανάπτυξη, χρησιμοποιήθηκαν σύγχρονες τεχνολογίες όπως το ASP.NET Core MVC, το SQL Server για την αποθήκευση των δεδομένων και το Identity framework της Microsoft για τη διαχείριση των χρηστών και των ρόλων τους. Παράλληλα, δόθηκε ιδιαίτερη έμφαση στην εμφάνιση και ευχρηστία της εφαρμογής, με σελίδες που σχεδιάστηκαν ώστε να είναι φιλικές προς τον χρήστη και να εμπνέουν επαγγελματισμό.

Αναμφισβήτητα, η εφαρμογή είναι πλήρως λειτουργική και έτοιμη να παρουσιαστεί ως ένα αξιόλογο αποτέλεσμα πτυχιακής εργασίας. Ωστόσο, όπως κάθε δυναμική εφαρμογή, έτσι και το AITHERAS FC E-Shop αφήνει περιθώρια για μελλοντικές βελτιώσεις. Ενδεικτικά, θα μπορούσε να προστεθεί λειτουργία προβολής των τρεχουσών παραγγελιών για τον απλό χρήστη, καθώς και η δυνατότητα για τον διαχειριστή να βλέπει συγκεντρωτικά όλες τις παραγγελίες που έχουν καταχωρηθεί από τους χρήστες. Τέτοιες λειτουργίες όχι μόνο θα ενίσχυαν την πληρότητα του συστήματος, αλλά θα το προετοίμαζαν και για πιθανή επαγγελματική αξιοποίηση στο μέλλον.

Συνοψίζοντας, η εργασία πέτυχε πλήρως τους στόχους της, αποδεικνύοντας πως με τον σωστό προγραμματισμό, την επιμονή και τη σωστή χρήση εργαλείων, είναι εφικτό να δημιουργηθεί ένα σύγχρονο και αποδοτικό σύστημα e-commerce που να ανταποκρίνεται στις ανάγκες μιας ποδοσφαιρικής ομάδας – και όχι μόνο.

Κεφάλαιο 6ο

6 Βιβλιογραφικές Πηγές

1. Microsoft Visual Studio: https://visualstudio.microsoft.com (Ημερομηνία εισόδου: 15/04/2025)
2. Microsoft Visual Studio 2022: https://visualstudio.microsoft.com/vs/ (Ημερομηνία εισόδου: 15/04/2025)
3. Εισαγωγή στη C#: https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/language-specification/introduction (Ημερομηνία εισόδου: 15/04/2025)
4. Επίσημη τεκμηρίωση για το ASP.NET Core: https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-9.0 (Ημερομηνία εισόδου: 15/04/2025)
5. Επισκόπηση και οδηγίες για το ASP.NET Identity: https://learn.microsoft.com/en-us/aspnet/identity/ (Ημερομηνία εισόδου: 15/04/2025)
6. Εισαγωγικός οδηγός για την ανάπτυξη web εφαρμογών με χρήση του ASP.NET MVC: https://learn.microsoft.com/en-us/aspnet/mvc/overview/getting-started/introduction/getting-started (Ημερομηνία εισόδου: 15/04/2025)
7. Τα Views στο ASP.NET Core MVC: https://learn.microsoft.com/en-us/aspnet/core/mvc/views/overview?view=aspnetcore-3.1 (Ημερομηνία εισόδου: 15/04/2025)
8. Επισκόπηση του Entity Framework Core: https://learn.microsoft.com/en-us/ef/core/ (Ημερομηνία εισόδου: 15/04/2025)
9. Πώς να δημιουργήσετε ένα ηλεκτρονικό κατάστημα ποδοσφαιρικού (φανταστικού) συλλόγου: https://freewebstore.com/sell/football-clubs.html (Ημερομηνία εισόδου: 15/04/2025)
10. Συμβουλές σχεδίασης ιστοσελίδων για αθλητικά e-shop, με έμφαση σε δυναμικά οπτικά στοιχεία, πλοήγηση φιλική προς τον χρήστη: https://digitalagencynetwork.com/best-sports-ecommerce-website-design-tips-with-examples/ (Ημερομηνία εισόδου: 15/04/2025)
11. Inter Official site Boutique Store: https://store.inter.it/it-it/?srsltid=AfmBOooO1GqxdlNDFzJfpLcCk-nLhU4ilvYvg4FiskYGZanqT_ZZWt7M (Ημερομηνία εισόδου: 15/04/2025)
12. Barca Official Store: https://store.fcbarcelona.com/ (Ημερομηνία εισόδου: 15/04/2025)

13. Περιγραφή των διαφορετικών τύπων διαγραμμάτων UML και των χρήσεών τους: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/overview-of-the-14-uml-diagram-types/ (Ημερομηνία εισόδου: 15/04/2025)
14. Οδηγός για τη δημιουργία Use Case Diagrams με παραδείγματα και συμβουλές: https://www.lucidchart.com/pages/uml-use-case-diagram (Ημερομηνία εισόδου: 15/04/2025)
15. Αναλυτικός οδηγός για τη δημιουργία Sequence Diagrams με παραδείγματα και συμβουλές: https://creately.com/guides/sequence-diagram-tutorial/ (Ημερομηνία εισόδου: 15/04/2025)
16. Δημιουργία Διαγραμμάτων στο ArgoUML: https://users.ece.utexas.edu/~miryung/teaching/EE461L-Spring2012/labs/uml.html (Ημερομηνία εισόδου: 15/04/2025)
17. Βήμα προς βήμα οδηγός για τη δημιουργία Use Case διαγραμμάτων στο ArgoUML: https://argouml-tigris-org.github.io/tigris/argouml/tours/bdUseCaseDiagram.html (Ημερομηνία εισόδου: 15/04/2025)
18. SQL Server Management Studio 19: Οδηγίες για τη σύνδεση σε μια βάση δεδομένων, τη δημιουργία πινάκων και την εκτέλεση queries: https://learn.microsoft.com/en-us/ssms/quickstarts/ssms-connect-query-sql-server (Ημερομηνία εισόδου: 15/04/2025)
19. Επισκόπηση των Migration: https://learn.microsoft.com/en-us/ef/core/managing-schemas/migrations/?tabs=dotnet-core-cli (Ημερομηνία εισόδου: 15/04/2025)
20. Οδηγίες για την εφαρμογή των migrations σε βάσεις δεδομένων: https://learn.microsoft.com/en-us/ef/core/managing-schemas/migrations/applying (Ημερομηνία εισόδου: 15/04/2025)