ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πρόγραμμα Μεταπτυχιακών Σπουδών

«ΠΜΣ ΠΛΗΡΟΦΟΡΙΚΗ»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**SmartOps ERP – Πρωτότυπο Μοντέλο** **Υλοποιημένο σε ASP.NET Core MVC (.NET 8 Technology) και Entity Framework** **SmartOps ERP – Prototype Model** **Implemented in ASP.NET Core MVC (.NET 8 Technology) and Entity Framework**
Ονοματεπώνυμο Φοιτητή	**Γεροβασίλης Κωνσταντίνος**
Πατρώνυμο	**Νικόλαος**
Αριθμός Μητρώου	**ΜΠΠΛ21008**
Επιβλέπων	**Ευθύμιος Αλέπης, Καθηγητής**

Ημερομηνία Παράδοσης **Δεκέμβριος 2025**

Τριμελής Εξεταστική Επιτροπή

Ευθύμιος Αλέπης Καθηγητής	Μαρία Βίρβου Καθηγήτρια	Διονύσιος Σωτηρόπουλος Αναπληρωτής Καθηγητής
(Υπογραφή)	(Υπογραφή)	(Υπογραφή)

Περιεχόμενα

ΠΕΡΙΛΗΨΗ (Ελληνικά)

Η παρούσα μεταπτυχιακή διατριβή παρουσιάζει την ανάπτυξη του **SmartOps**, ενός πρωτοτύπου mini ERP συστήματος που υλοποιήθηκε ως web εφαρμογή με χρήση ASP.NET Core MVC, Entity Framework Core και SQL Server Express. Στόχος της εργασίας ήταν η δημιουργία ενός απλού, ευέλικτου και επεκτάσιμου πληροφοριακού συστήματος που καλύπτει βασικές επιχειρησιακές λειτουργίες, όπως η διαχείριση πελατών, προμηθευτών, ειδών, υπηρεσιών και παραστατικών.

Η αρχιτεκτονική του SmartOps βασίζεται σε modular σχεδίαση, επιτρέποντας τη μελλοντική επέκταση του συστήματος με πρόσθετες λειτουργίες, όπως το MyData, αποθήκη, αναφορές και AI αυτοματισμούς. Στο πλαίσιο της εργασίας αναπτύχθηκαν επίσης μηχανισμοί validation, επιχειρησιακοί κανόνες (business logic), αυτόματη αρίθμηση παραστατικών, dashboard στατιστικών και μηχανισμοί ασφαλούς διαχείρισης χρηστών.

Το SmartOps ERP αποδεικνύει ότι ένα σύγχρονο πληροφοριακό σύστημα μπορεί να παραμείνει λειτουργικό και αποδοτικό, ακόμη και όταν σχεδιάζεται με έμφαση στην απλότητα. Μέσα από την υλοποίηση του συστήματος αναδεικνύονται η σημασία ενός καθαρού μοντέλου δεδομένων, η αξία του MVC προτύπου, καθώς και οι προκλήσεις που προκύπτουν κατά την ανάπτυξη πραγματικών πληροφοριακών εφαρμογών.

Το έργο αποτελεί μια ολοκληρωμένη βάση για μελλοντική έρευνα και ανάπτυξη συστημάτων ERP με προηγμένες δυνατότητες αυτοματοποίησης και τεχνητής νοημοσύνης.

ABSTRACT (English)

This Master's thesis presents the development of **SmartOps**, a prototype mini ERP system implemented as a web application using ASP.NET Core MVC, Entity Framework Core, and SQL Server Express. The main objective of this project was to design and implement a simple, flexible, and extensible management system that supports essential business operations, including customer management, supplier tracking, product and service handling, and invoice creation.

The architecture of SmartOps follows a modular approach, allowing seamless future expansion with additional modules such as MyData integration, warehouse management, reporting tools, and AI-driven automation. As part of the implementation, the system incorporates validation mechanisms, business logic enforcement, automated invoice numbering, a statistical dashboard, and secure user authentication and management.

SmartOps demonstrates that a modern ERP system can remain efficient, functional, and user-friendly even when designed with simplicity as a core principle. The development process highlights the importance of clean data modeling, the benefits of the MVC pattern, and the practical challenges encountered in real-world information system implementation.

This project provides a solid foundation for further research and development of ERP platforms enriched with intelligent automation, predictive analytics, and artificial intelligence capabilities.

1.Εισαγωγή

1.1 Σκοπός και Συνεισφορά του έργου

Η ταχύτατη τεχνολογική πρόοδος και οι αυξανόμενες απαιτήσεις της αγοράς έχουν καταστήσει αναγκαία τη διαρκή αναβάθμιση των συστημάτων διαχείρισης επιχειρησιακών πόρων. Στο πλαίσιο αυτής της διπλωματικής εργασίας, εξετάζεται η υλοποίηση ενός πρωτοτύπου ERP συστήματος, το οποίο επιδιώκει να συνδυάσει λειτουργικότητα, ευελιξία και καινοτομία, αποτελώντας ένα σημαντικό εργαλείο για την υποστήριξη των σύγχρονων επιχειρήσεων. Μέσα από τη μελέτη και την ανάπτυξη του έργου, αναλύονται τα κύρια χαρακτηριστικά, οι προκλήσεις και οι προοπτικές εξέλιξης ενός τέτοιου συστήματος, με ιδιαίτερη έμφαση στη συμβολή του στη βελτίωση των επιχειρησιακών διαδικασιών και στην αύξηση της ανταγωνιστικότητας.

1.2 Τι είναι το ERP

Το ακρωνύμιο **ERP** προέρχεται από τα αρχικά των λέξεων **Enterprise Resource Planning**, το οποίο μπορεί να αποδοθεί στα ελληνικά ως «σύστημα επιχειρησιακού σχεδιασμού». Στο internet υπάρχουν αμέτρητες αναλύσεις και επιστημονικά άρθρα για το ERP. Ωστόσο, ο επικρατέστερος ίσως ορισμός είναι πως το ERP είναι ένα σύστημα λογισμικού, το οποίο είναι επιφορτισμένο με το να διαχειρίζεται όλες τις λειτουργίες της επιχείρησης, με απώτερο σκοπό την αύξηση του business performance. Αυτό επιτυγχάνεται με την ορθολογικότερη οργάνωση και αξιοποίηση δεδομένων και πόρων της επιχείρησης (ανθρώπινου δυναμικού, υλικού, οικονομικών πόρων κ.λπ.) *[Πηγή: www.Softone.gr]*

1.3 ERP: Ιστορική αναδρομή

Από τη δημιουργία του, τη δεκαετία του ’90, όπου το ERP ξεκίνησε εισάγοντας λειτουργίες λογιστηρίου και διαχείρισης ανθρωπίνων πόρων/μισθοδοσίας σε ένα MRP (Manufacturing Resource Planning), έχει εξελιχθεί τεχνολογικά με ταχύτατους ρυθμούς. Έτσι, τα σύγχρονα ERP έχουν πλέον ενσωματώσει κι άλλες δυνατότητες, όπως διαχείριση πελατειακών σχέσεων (CRM), διαχείριση αποθήκης (warehouse management system - WMS), ηλεκτρονική ανταλλαγή δεδομένων (EDI), ακόμα και συστήματα διαχείριση ποιότητας (Integrated Quality Management – IQM).

Πολλοί συγχέουν τα συστήματα ERP με τα εμπορικά πακέτα λογισμικού. Ωστόσο, ένα ERP είναι κάτι πολύ παραπάνω από μια λογιστική διαχείριση.

Η διαφορά με αυτά είναι ότι ένα ERP επεξεργάζεται και αξιοποιεί όλα τα δεδομένα της επιχείρησης, υποστηρίζοντας όλες τις λειτουργίες της, που μπορεί να εκτείνονται από τις οικονομικές υπηρεσίες και τη διοίκηση ανθρώπινων πόρων, μέχρι την εφοδιαστική αλυσίδα και τις σχέσεις με τους πελάτες.

Όλα τα παραπάνω γίνονται εφικτά χάρη στην ενιαία βάση δεδομένων στην οποία καταχωρούνται όλες οι συναλλαγές και τα δεδομένα της επιχείρησης, τα οποία παρακολουθούνται σε πραγματικό χρόνο (ή σχεδόν πραγματικό κάποιες φορές). [Πηγή: www.Softone.gr]

1.4 Οφέλη ενός ERP

Μια λύση ERP βοηθάει μία επιχείρηση να:

- Αυξήσει τα έσοδά της
- Βελτιώσει την αποτελεσματικότητά της, ενοποιώντας διάφορες εργασίες (Finance & Accounting, Manufacturing, Supply Chain, Customer Relationship Management, Human Resources, BI/Reporting)

- Αυξήσει την παραγωγικότητα των ανθρώπων της, αυτοματοποιώντας τις διαδικασίες της
- Μειώσει τα λειτουργικά της έξοδα
- Αυξήσει την ασφάλεια των επιχειρησιακών της δεδομένων
- Αντλήσει και να αξιοποιήσει πληροφορίες από διαφορετικά τμήματα της εταιρείας

[Πηγή: www.Softone.gr]

1.5 Τι περιλαμβάνει ένα σύστημα ERP

Στην ουσία ένα σύστημα ERP, αποτελείται από επιμέρους **ενσωματωμένες εφαρμογές (γνωστές ως modules)**, τις οποίες η επιχείρηση αξιοποιεί για να συγκεντρώνει, να αποθηκεύει, να διαχειρίζεται και σε τελική ανάλυση, να ερμηνεύει δεδομένα για πολλές (ιδανικά όλες) από τις δραστηριότητές της. Μερικά από τα modules ενός συστήματος ERP είναι:

- Εμπορική Διαχείριση
- Χρηματοοικονομική Διαχείριση
- Έσοδα-Έξοδα
- Γενική Λογιστική
- Διαχείριση Παγίων
- Διαχείριση Αποθήκης
- Διαχείριση Ανθρώπινων Πόρων (HRM)
- Μισθοδοσία
- Διαχείριση Διανομής
- Διαχείριση Εφοδιαστικής Αλυσίδας
- Διαχείριση Πελατειακών Σχέσεων (CRM)
- Διαχείριση Λιανικής
- Προϊοντικός Σχεδιασμός
- Διαχείριση Παραγωγής
- Διαχείριση Υπηρεσιών
- Χρηματοοικονομική Λογιστική
- Πωλήσεις
- Marketing
- Τιμολόγηση

[Πηγή: www.Softone.gr]

1.6 Χαρακτηριστικά ERP

Όταν μια επιχείρηση έχει αναγνωρίσει τις γενικές και ιδιαίτερες ανάγκες της, και εξετάζει τις επιλογές της για μια λύση ERP, εκτός από το κόστος της λύσης και το κατά πόσο αυτή καλύπτει τις παραπάνω ανάγκες, έχει σημασία να εξετάσει και τα χαρακτηριστικά της ίδιας της λύσης, όπως και των επιπλέον υπηρεσιών που προσφέρει η εταιρεία ανάπτυξης ERP.

Έτσι, θα πρέπει να συγκριθούν τα επιμέρους χαρακτηριστικά των ERP προγραμμάτων, καθώς και των εταιρειών ανάπτυξής τους, ώστε οι διαδικασίες μετάβασης, εγκατάστασης, εκμάθησης και τελικά χρήσης της εφαρμογής, να ολοκληρωθούν έγκαιρα και χωρίς προβλήματα.

Πιο συγκεκριμένα, τα χαρακτηριστικά και οι επιπλέον υπηρεσίες που πρέπει να εξεταστούν είναι:

- **Έκδοση εφαρμογής**: Καλύπτονται οι ανάγκες της επιχείρησης από τα έτοιμα πακέτα μιας έκδοσης ERP;
- **Χρόνος υλοποίησης**: Πόσο θα διαρκέσει η ολοκλήρωση και παράδοση της λύσης;
- **Δυνατότητα μεταφοράς δεδομένων**: Τι έτοιμες επιλογές παρέχονται για data migration;
- **Συχνότητα αναβάθμισης**: Πόσο συχνά αναβαθμίζεται το ERP ώστε να συμπεριλαμβάνει καίριες διορθώσεις και να καλύπτει/προσθέτει νέες δυνατότητες;
- **Τύπος εγκατάστασης και deployment options**: Υπάρχει η δυνατότητα επιλογής μεταξύ SaaS (Software as a Service) και on-premise (in-house) μοντέλου;
- **Security**: Τι ασφάλεια παρέχει όσον αφορά τα επιχειρησιακά δεδομένα και πληροφορίες;
- **Ταχύτητα**: Πόσο γρήγορα «φορτώνει» τις πληροφορίες/δεδομένα που χρειάζεται η επιχείρηση;
- **Scalability**: Πόσο έτοιμο είναι το σύστημα ERP ώστε να ενσωματώσει τυχόν απαιτούμενες αλλαγές, καθώς η επιχείρηση θα αναπτύσσεται και επεκτείνεται;
- **Adaptability**: Υπάρχει η δυνατότητα προσαρμογής της λύσης στις ιδιαίτερες ανάγκες της επιχείρησης;
- **Platform**: Πόσο φιλική στο χρήστη, είναι η πλατφόρμα ERP;
- **Training**: Υπάρχει δυνατότητα εκπαίδευσης, αλλά και συνεχούς πληροφόρησης για τις νέες εκδόσεις;

[Πηγή: www.Softone.gr]

2.Περιγραφή του SmartOps

2.1 Τι είναι το SmartOps

Το SmartOps είναι ένα **ERP Prototype** το οποίο είναι της μορφής Web Application με τοπικό server και βάση δεδομένων. Συνεπώς είναι μια τοπική εγκατάσταση με λειτουργίες που καλύπτουν τις μέσες ανάγκες και απαιτήσεις μιας επιχείρησης. Στην σημερινή εποχή υπάρχουν ήδη πολλά ERP της συγκεκριμένης μορφής, οπότε το συγκεκριμένο ίσως θεωρηθεί ακόμη ένα κοινό πρόγραμμα σαν τα υπόλοιπα. Ωστόσο το συγκεκριμένο ERP διαφέρει για την απλότητα του, για την ευκολία του χρήστη να μπορεί να το χειριστεί εύκολα και γρήγορα και επίσης αυτό που θα το κάνει να διαφέρει είναι ότι μελλοντικά θα προστεθεί μία ιδιαίτερη λειτουργία που έχει να κάνει και με έναν τομέα που είναι ιδιαίτερα αμφιλεγόμενος. Φυσικά αναφερόμαστε στον τομέα της τεχνητής νοημοσύνης. Αυτή η ιδιαίτερη λειτουργεία λοιπόν θα είναι ένα ChatBot AI το οποίο θα προσφέρει στον χρήστη την δυνατότητα να ψάχνει πράγματα που θέλει απλά ρωτώντας το συγκεκριμένο Bot.

2.2 Τι σημαίνει SmartOps - από που προκύπτει η ονομασία του

Το όνομα **SmartOps** επιλέχθηκε γιατί συνδυάζει δύο βασικά στοιχεία του συστήματος: τη "νοημοσύνη" (Smart) και τις "επιχειρησιακές λειτουργίες" (Operations). Αντικατοπτρίζει τον σκοπό της εφαρμογής, δηλαδή να είναι κάτι περισσότερο από ένα απλό ERP, προσφέροντας αυτοματισμούς και απλότητα στον χρήστη.

2.3 Σε ποιους απευθύνεται

Το SmartOps ERP Prototype απευθύνεται σε **όλες τις επιχειρήσεις** που θέλουν να καλύπτουν τις βασικές επιχειρησιακές τους ανάγκες και ταυτόχρονα να αποκτήσουν εμπειρία πάνω σε ένα ERP σύστημα μέσω της απλότητας που προσφέρει, της εύκολης προσβασιμότητας, καθώς και της παραμετροποίησης που μπορεί να δεχτεί, αν χρειαστεί. Επίσης δίνει και την δυνατότητα στον χρήστη με την μελλοντική προσθήκη της AI τεχνολογία να εξοικειωθεί και σε αυτό τον τομέα, καθώς το να μάθεις να συνομιλείς με ένα chatbot και να παίρνεις πληροφορίες και δεδομένα μέσω αυτού είναι μια ιδιαίτερη εμπειρία και κάτι το οποίο φαίνεται πως θα ακολουθεί τον τεχνολογικό κόσμο για τα επόμενα χρόνια.

2.4 Οι λειτουργίες του SmartOps

Οι λειτουργίες που προσφέρει το SmartOps είναι οι βασικές λειτουργίες που χρειάζεται ένα ERP για να καλύψει τις επιχειρησιακές ανάγκες μιας οποιαδήποτε επιχείρησης. Συγκεκριμένα αυτές οι λειτουργίες είναι:

- Η εμπορική διαχείριση, η οποία περιλαμβάνει τις πωλήσεις και τις αγορές.
- Οι πελάτες και οι προμηθευτές της επιχείρισης.
- Τα είδη και οι υπηρεσίες της επιχείρησης.

2.5 Ανταγωνιστικά Πλεονεκτήματα του SmartOps ERP

Το **SmartOps ERP Prototype** αναπτύχθηκε με στόχο να συνδυάσει τη λειτουργικότητα των σύγχρονων εμπορικών και λογιστικών συστημάτων με μια πιο απλή, ευέλικτη και

προσαρμόσιμη αρχιτεκτονική. Σε σύγκριση με αντίστοιχα εμπορικά λογισμικά που διατίθενται στην αγορά, το SmartOps παρουσιάζει ορισμένα σημαντικά ανταγωνιστικά πλεονεκτήματα, τα οποία ενισχύουν τόσο την αποδοτικότητα των επιχειρησιακών διαδικασιών όσο και την εμπειρία του τελικού χρήστη.

1. **Απλότητα και εργονομία διεπαφής χρήστη (UI/UX)**
   Η διεπαφή του SmartOps σχεδιάστηκε με γνώμονα τη λειτουργικότητα και την ευκολία χρήσης. Οι βασικές οντότητες (Πελάτες, Προμηθευτές, Προϊόντα, Υπηρεσίες, Τιμολόγια) είναι οργανωμένες με καθαρή δομή, ελάχιστα επίπεδα πλοήγησης και ενιαία αισθητική, μειώνοντας τον χρόνο εκπαίδευσης των χρηστών σε σχέση με πιο πολύπλοκα ERP περιβάλλοντα.
2. **Ανοιχτή και επεκτάσιμη αρχιτεκτονική**
   Το SmartOps έχει υλοποιηθεί σε περιβάλλον **ASP.NET Core MVC** με **Entity Framework Core**, γεγονός που επιτρέπει την εύκολη προσαρμογή και επέκταση του συστήματος. Η modular λογική της εφαρμογής διευκολύνει τη μελλοντική προσθήκη επιμέρους ενοτήτων, όπως διαχείριση αποθήκης ή MyData διασύνδεση, χωρίς ανάγκη ριζικής ανασχεδίασης.
3. **Τοπική βάση δεδομένων με δυνατότητα μελλοντικής μετάβασης στο cloud**
   Η χρήση **SQL Server Express** εξασφαλίζει σταθερότητα και ταχύτητα σε περιβάλλον ανάπτυξης ή μικρομεσαίων επιχειρήσεων, ενώ παράλληλα η δομή της βάσης επιτρέπει την εύκολη μεταφορά σε **Azure SQL Database**, καλύπτοντας έτσι διαφορετικά σενάρια εγκατάστασης.
4. **Απλοποιημένες διαδικασίες τιμολόγησης και διαχείρισης**
   Η διαδικασία έκδοσης παραστατικών στο SmartOps έχει σχεδιαστεί με έμφαση στη ροή και τη χρηστικότητα, επιτρέποντας στον χρήστη να επιλέγει σειρά παραστατικού, πελάτη και τρόπο πληρωμής με λίγα μόνο βήματα. Η λογική αυτή μειώνει τα λειτουργικά σφάλματα και αυξάνει την παραγωγικότητα.
5. **Εστίαση στην ευελιξία και την εκπαιδευτική αξία**
   Σε αντίθεση με τα κλειστά εμπορικά λογισμικά, το SmartOps είναι πλήρως ανοιχτό και επεξηγηματικό ως προς τη δομή και τον κώδικά του, αποτελώντας έτσι ένα ισχυρό εκπαιδευτικό εργαλείο για φοιτητές και επαγγελματίες πληροφορικής που επιθυμούν να κατανοήσουν τη λειτουργία ενός ERP σε πρακτικό επίπεδο.
6. **Προσαρμοστικότητα στις ανάγκες της ελληνικής αγοράς**
   Παρότι αποτελεί ακαδημαϊκή εφαρμογή, το SmartOps λαμβάνει υπόψη του τις ελληνικές εμπορικές πρακτικές και απαιτήσεις (όπως το ΑΦΜ, η κατηγορία πελάτη και η κατηγορία ΦΠΑ), γεγονός που του προσδίδει ρεαλισμό και λειτουργικότητα σε πραγματικές επιχειρησιακές συνθήκες.

3.Τεχνολογίες Ανάπτυξης

3.1 SQL Server Express

Η **SQL Server Express** αποτελεί την ελαφριά και δωρεάν έκδοση του Microsoft SQL Server. Είναι ιδανική για εκπαιδευτικούς, αναπτυξιακούς και μικρής κλίμακας παραγωγικούς σκοπούς, προσφέροντας τα βασικά χαρακτηριστικά ενός ισχυρού σχεσιακού συστήματος διαχείρισης βάσεων δεδομένων (RDBMS).

Χαρακτηριστικά και ρόλος στην εφαρμογή:

- Χρησιμοποιείται για την αποθήκευση και διαχείριση των δεδομένων της εφαρμογής (πελάτες, προϊόντα, παραστατικά κ.λπ.).
- Υποστηρίζει συναλλαγές (transactions), constraints, stored procedures και indexes.
- Επιτρέπει ασφαλή και αποδοτική αναζήτηση, τροποποίηση και διαγραφή δεδομένων.
- Συνδέεται απευθείας με την εφαρμογή μέσω του Entity Framework Core και της σύνδεσης DbContext.

3.2 SQL Server Management Studio (SSMS)

Το **SSMS** είναι το επίσημο εργαλείο της Microsoft για τη διαχείριση και εποπτεία βάσεων δεδομένων SQL Server. Παρέχει γραφική διεπαφή για την εκτέλεση SQL εντολών, τη διαχείριση πινάκων, views, indexes, καθώς και την παρακολούθηση της απόδοσης του διακομιστή.

Χρήση στο πλαίσιο της διπλωματικής:

- Δημιουργία και έλεγχος της βάσης δεδομένων.
- Επιθεώρηση των δεδομένων που καταχωρούνται από το σύστημα.
- Εκτέλεση χειροκίνητων ερωτημάτων για έλεγχο και επιβεβαίωση ορθότητας της λειτουργίας της εφαρμογής.
- Υποβοήθηση του εντοπισμού λαθών και προβλημάτων σύνδεσης.

3.3 ASP.NET Core MVC (.NET 8)

Το **ASP.NET Core MVC** αποτελεί το κύριο framework ανάπτυξης web εφαρμογών από τη Microsoft. Βασίζεται στο πρότυπο Model-View-Controller (MVC), το οποίο προάγει τον διαχωρισμό της επιχειρησιακής λογικής από τη διεπαφή χρήστη, διευκολύνοντας τη συντήρηση και την εξέλιξη του λογισμικού.

Κύρια χαρακτηριστικά και ρόλος:

- Υλοποιεί την αρχιτεκτονική της εφαρμογής διαχωρίζοντας:
  - Models (λογική δεδομένων),
  - Views (παρουσίαση προς τον χρήστη),
  - Controllers (χειρισμός αιτημάτων και επιχειρησιακή λογική).
- Η έκδοση .NET 8 προσφέρει βελτιώσεις σε απόδοση, υποστήριξη για cloud εφαρμογές και εξελιγμένο dependency injection.
- Επιτρέπει τη δημιουργία RESTful endpoints, την υποστήριξη Razor views και τη χρήση ισχυρού routing.

Οφέλη:

- Υψηλή απόδοση και ασφάλεια.
- Διαλειτουργικότητα με άλλα εργαλεία και APIs.
- Ενσωματωμένη υποστήριξη για middleware, authentication και validation.

3.4 Entity Framework Core

Το **Entity Framework Core (EF Core)** είναι ένα σύγχρονο Object-Relational Mapping (ORM) εργαλείο της Microsoft. Επιτρέπει τη διαχείριση της βάσης δεδομένων μέσω αντικειμενοστραφών κλάσεων της C#, χωρίς την ανάγκη γραφής απευθείας SQL εντολών.

Χρήση στο έργο SmartOps:

- Υλοποιήθηκε με τη μέθοδο Code First, όπου πρώτα ορίζονται οι κλάσεις (οντότητες) και στη συνέχεια το EF δημιουργεί αυτόματα τη βάση δεδομένων.
- Χρησιμοποιούνται migrations για την παρακολούθηση αλλαγών στο σχήμα της βάσης δεδομένων.
- Επιτρέπει τη χρήση LINQ (Language Integrated Query) για σύνταξη ερωτημάτων με C# συντακτικό.
- Ενισχύει την ασφάλεια αποφεύγοντας SQL injection και απλοποιεί τη διαχείριση δεδομένων.

3.5 Visual Studio

Το **Visual Studio 2022** αποτελεί το κύριο ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) που χρησιμοποιήθηκε για τη δημιουργία και τη διαχείριση του έργου.

Χαρακτηριστικά:

- Υποστηρίζει πλήρως το .NET 8 και τα εργαλεία του EF Core.
- Παρέχει IntelliSense, debugging, live server preview και ενσωμάτωση με Git.
- Διευκολύνει την οργάνωση των controllers, models, views και services.
- Προσφέρει εργαλεία για migration management, παρακολούθηση exceptions, και testing.

4.Δομή της Βάσης Δεδομένων

4.1 Server

Χρησιμοποιείται ο **Microsoft SQL Server Express** σε τοπικό περιβάλλον (με δυνατότητα μελλοντικής μεταφοράς σε Azure SQL Database), και η σύνδεση με την εφαρμογή γίνεται μέσω του Entity Framework Core, το οποίο λειτουργεί ως ORM (Object-Relational Mapping).

4.2 Οι βασικές οντότητες/πίνακες

Παρακάτω παρατίθενται οι βασικοί πίνακες της βάσης SmartOpsDb καθώς και οι σχέσεις μεταξύ τους:

1. Customers

- Περιέχει πληροφορίες πελατών
- Πεδία: Id, CustomerCode, Name, TaxIdentificationNumber, Country, Address City, Postal Code, VatStatus, CustomerCategory, UserId.

2. Items

- Κατάλογος προϊόντων της επιχείρησης
- Πεδία: Id, ItemCode, Description, Unit, VAT, RetailPrice, WholeSalePrice, UserId.

3. Services

- Παροχές υπηρεσιών προς τους πελάτες
- Πεδία: Id, ServiceCode, Description, Unit, VAT, RetailPrice, WholeSalePrice, UserId.

4. Invoices

- Παραστατικά πώλησης
- **Πεδία**: Id, Series, Number, IssueDate, Year, CustomerId., PaymentMethod, InvoiceType, TotalNet, TotalVat, TotalGross, PaidAmount, IsPaid, UserId.

5. **InvoicesLines**

- Γραμμές παραστατικού
- **Πεδία**: InvoiceLineId, InvoiceId, ProductId, ServiceId, Quantity, UnitPrice.

6. InvoiceItems

- Συσχετίζει τα προϊόντα/υπηρεσίες με τα παραστατικά
- **Πεδία**: Id, InvoiceId, ItemtId, ServiceId, Quantity, UnitPrice, VatRate.

7. Suppliers

- Προμηθευτές προϊόντων
- **Πεδία**: Id, SupplierCode, Name, TaxIdentificationNumber, Country, Address, City, PostalCode, SupplierCategory, VatStatus, UserId.

8. Users

- Χρήστες του συστήματος (π.χ. υπεύθυνος επιχείρισης, υπάλληλοι)
- **Πεδία**: Id, Username, PasswordHash.

4.3 Σχέσεις Οντοτήτων

- Ένας πελάτης μπορεί να έχει πολλά παραστατικά (Customers → Invoices).
- Ένα παραστατικό μπορεί να περιλαμβάνει πολλαπλά προϊόντα ή υπηρεσίες (Invoices → InvoiceItems → Products/Services).

- Υπάρχει αναφορά του τρόπου πληρωμής και της σειράς παραστατικού σε κάθε Invoice, με δυνατότητα επέκτασης σε μελλοντικούς πίνακες όπως PaymentMethods, InvoiceSeriesTypes.

4.4 Τεχνικά χαρακτηριστικά

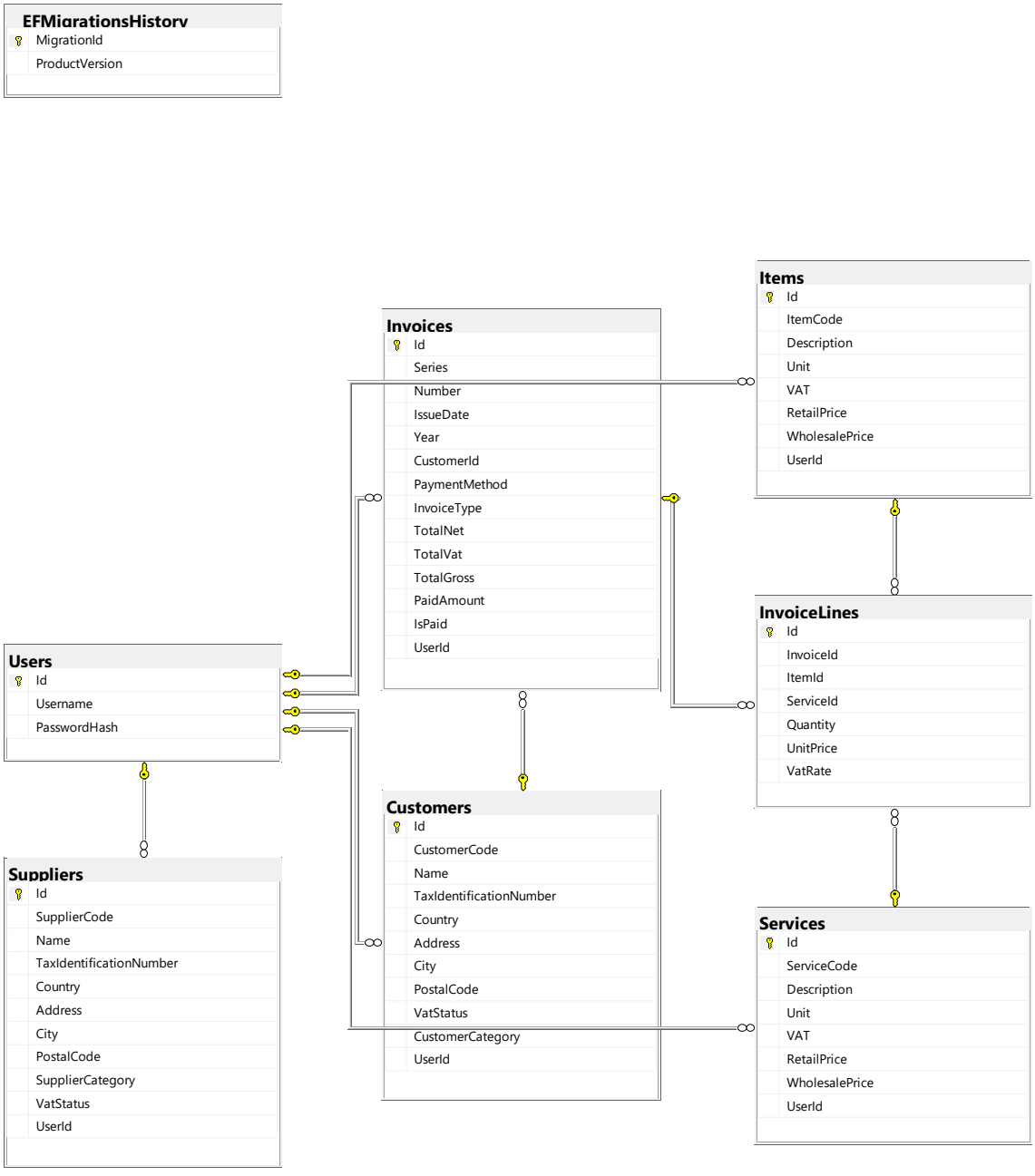
- **Database Engine**: SQL Server Express LocalDB
- **Database Name**: SmartOpsDb
- **ORM**: Entity Framework Core (.NET 8)
- **Μοντέλο**: Code First με χρήση Migrations
- **Primary Keys**: Auto-incremented integers σε κάθε πίνακα
- **Relationships**: Foreign keys & Navigation properties
- **Constraints**: Not Null, Foreign Key constraints, Check constraints (σε μελλοντικό refactoring)

4.5 Προοπτικές επέκτασης

- Υποστήριξη πολλών νομισμάτων (π.χ. πεδίο Currency στον πίνακα Invoices)
- Module για Payments, Expenses, Warehouses
- Σύνδεση με APIs (π.χ. e-invoicing ή Taxisnet, Skroutz)

4.6 Αναλυτικός Σχεδιασμός Συστήματος

4.6.1 Διάγραμμα ER



Εικόνα 4.1 Διάγραμμα Οντοτήτων–Σχέσεων (ERD) της βάσης δεδομένων SmartOpsDb

Το διάγραμμα παρουσιάζει τη λογική δομή της βάσης δεδομένων του SmartOps ERP Prototype. Κάθε πλαίσιο αντιπροσωπεύει έναν πίνακα (οντότητα), ενώ οι γραμμές δείχνουν τις σχέσεις μεταξύ των οντοτήτων μέσω ξένων κλειδιών (foreign keys).

Περιγραφή Γενικής Δομής

Η βάση δεδομένων του SmartOps είναι σχεδιασμένη με κανονικοποιημένη δομή, ώστε να εξασφαλίζεται η συνέπεια, η επεκτασιμότητα και η αποφυγή πλεονασμού δεδομένων. Αποτελείται από τους εξής κύριους πίνακες:

Πίνακας	Περιγραφή
Users	Περιέχει τους χρήστες του συστήματος και τα credentials τους (Username, PasswordHash). Όλες οι οντότητες συνδέονται με το UserId για λόγους ιχνηλασιμότητας.
Customers	Αποθηκεύει τα στοιχεία πελατών, όπως κωδικό, επωνυμία, ΑΦΜ, στοιχεία επικοινωνίας και κατηγορία πελάτη. Συνδέεται 1:N με τα Invoices.
Suppliers	Περιέχει πληροφορίες για τους προμηθευτές της επιχείρησης.
Items	Περιλαμβάνει τα προϊόντα με τιμές λιανικής/χονδρικής, Φ.Π.Α. και μονάδα μέτρησης.
Services	Παρόμοια με τα Items, αλλά αφορά υπηρεσίες που παρέχονται.
Invoices	Ο πίνακας των παραστατικών. Συνδέεται με Customers (πολλές εγγραφές ανά πελάτη) και περιέχει αναλυτικά στοιχεία όπως σειρά, αριθμό, ημερομηνία, σύνολα και τρόπο πληρωμής.
InvoicesLines	Ο συνδετικός πίνακας που υλοποιεί τη σχέση N:N μεταξύ Invoices και Items (ή Services), κρατώντας τις γραμμές κάθε τιμολογίου με ποσότητα, τιμή, συντελεστή ΦΠΑ και αξίες.

Πίνακας 4.6.1.1 Οι Πίνακες στην Βάση Δεδομενων και οι σχέσεις μεταξύ τους

Κύριες Σχέσεις (Relationships)

- **Customers → Invoices (1:N)**
  Κάθε πελάτης μπορεί να έχει πολλά παραστατικά, ενώ κάθε παραστατικό ανήκει σε έναν μόνο πελάτη (CustomerId).
- **Invoices → InvoicesItems (1:N)**
  Κάθε παραστατικό αποτελείται από μία ή περισσότερες γραμμές προϊόντος ή υπηρεσίας (InvoiceId).
- **Items → InvoicesLines (1:N)**
  Ένα προϊόν μπορεί να συμμετέχει σε πολλά παραστατικά.
- **Users → [Customers, Suppliers, Items, Services, Invoices] (1:N)**
  Δηλώνει ποιος χρήστης καταχώρησε ή τροποποίησε την εγγραφή, εξασφαλίζοντας audit trail.

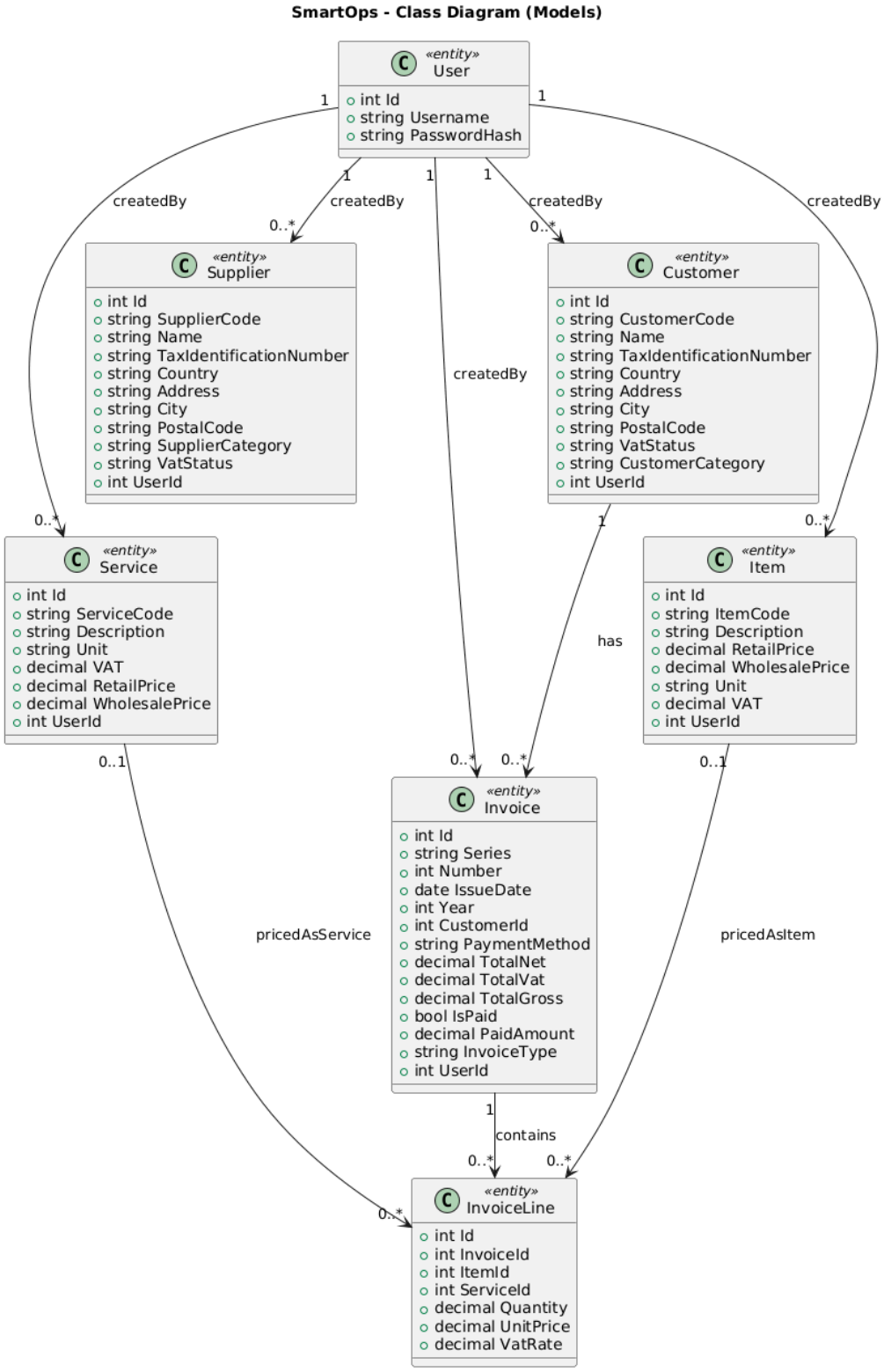
Σχόλια Σχεδιασμού

1. **Απλότητα και σαφήνεια:**
   Η βάση είναι καθαρή και ευανάγνωστη, χωρίς περιττές συσχετίσεις. Επικεντρώνεται στα απολύτως απαραίτητα για εμπορική και τιμολογιακή λειτουργία.
2. **Κανονικοποίηση:**
   Κάθε πίνακας περιλαμβάνει μόνο σχετικά πεδία και συνδέεται μέσω ξένων κλειδιών, εξασφαλίζοντας ακεραιότητα αναφορών.
3. **Επεκτασιμότητα:**
   Το σχήμα μπορεί να επεκταθεί εύκολα — π.χ. με προσθήκη Payments, Expenses, InvoiceSeries ή VatRates χωρίς τροποποίηση των βασικών πινάκων.

4. **Ιχνηλασιμότητα:**
   Το πεδίο UserId σε κάθε βασικό πίνακα επιτρέπει πλήρη παρακολούθηση ενεργειών χρηστών.
5. **Ασφάλεια:**
   Ο πίνακας Users περιέχει μόνο hash του κωδικού (PasswordHash), αποτρέποντας την αποθήκευση ευαίσθητων δεδομένων σε απλό κείμενο.

4.6.2 Class Diagram

Το διάγραμμα αποτυπώνει τη δομή των βασικών κλάσεων της εφαρμογής (Models), τα γνωρίσματα (properties) και τις συσχετίσεις τους. Αντιστοιχεί στο αντικειμενοστραφές επίπεδο (C#), ανεξάρτητα από το φυσικό σχήμα της βάσης.



Εικόνα 4.6.2.1 – Διάγραμμα Κλάσεων (UML Class Diagram) του SmartOps ERP

Σκοπός

Να παρουσιαστεί η αντικειμενοστραφής μοντελοποίηση του SmartOps, όπως υλοποιείται σε C# models (ASP.NET Core MVC + EF Core). Το διάγραμμα κλάσεων βοηθά στη συντήρηση, την επεκτασιμότητα και την κατανόηση της επιχειρησιακής λογικής.

Επισκόπηση Κλάσεων

Κλάση	Ρόλος	Βασικά Properties
User	Χρήστης/λογαριασμός που καταχωρεί δεδομένα και εκτελεί ενέργειες.	Id, Username, PasswordHash
Customer	Πελάτης της επιχείρησης.	Id, CustomerCode, Name, TaxIdentificationNumber, Country, Address, City, PostalCode, VatStatus, CustomerCategory, UserId
Supplier	Προμηθευτής της επιχείρησης	Id, SupplierCode, Name, TaxIdentificationNumber, Country, Address, City, PostalCode, VatStatus, SupplierCategory, UserId
Item	Εμπορεύσιμο προϊόν αποθήκης.	Id, ItemCode, Description, RetailPrice, WholesalePrice, Unit, VAT, UserId
Service	Παρεχόμενη υπηρεσία.	Id, ServiceCode, Description, Unit, VAT, RetailPrice, WholesalePrice, UserId
Invoice	Κεφαλίδα παραστατικού.	Id, Series, Number, IssueDate, Year, CustomerId, PaymentMethod, TotalNet, TotalVat, TotalGross, IsPaid, PaidAmount, UserId
InvoiceLine	Γραμμή παραστατικού (συσχέτιση με είδος).	Id, InvoiceId, ItemId, ServiceId, Quantity, UnitPrice, VatRate.

Πίνακας 4.6.2.1 Επισκόπηση της κάθε κλάσης

Σχέσεις

- **User → {Customer, Supplier, Item, Service, Invoice} (1:N)**
  Ιχνηλασιμότητα: ποιος δημιούργησε/ενημέρωσε κάθε εγγραφή.
- **Customer → Invoice (1:N)**
  Ένας πελάτης έχει πολλά παραστατικά.
- **Invoice → InvoiceLine (1:N, composition)**
  Οι γραμμές «ανήκουν» στο παραστατικό. Διαγραφή του Invoice συνεπάγεται διαγραφή των InvoiceItems.
- **Item → InvoiceLine (1:N)**
  Ένα προϊόν μπορεί να συμμετέχει σε πολλές γραμμές τιμολόγησης.
- **Service → InvoiceLine (1:N)**
  Μία υπηρεσία μπορεί να συμμετέχει σε πολλές γραμμές τιμολόγησης.

Σχεδιαστικές Αρχές

1. **Καθαρός διαχωρισμός ευθυνών (SRP):**
   Κεφαλίδα (Invoice) vs γραμμές (InvoiceLine). Οντότητες πελατών/προμηθευτών/προϊόντων/υπηρεσιών ανεξάρτητες.
2. **Επεκτασιμότητα:**
   Ευκολία προσθήκης modules (Payments, Expenses, VatRates) χωρίς αλλαγή στις βασικές κλάσεις.
3. **Ιχνηλασιμότητα:**
   UserId σε βασικές οντότητες για auditing.
4. **Σαφής τιμολόγηση:**
   Υπολογιστικές ιδιότητες σε InvoiceLine και αθροίσεις στην Invoice (ενισχύει αναγνωσιμότητα και testing).
5. **Συμμόρφωση με MVC/EF Core:**
   Τα Models είναι «λεπτά» (POCOs). Η επιχειρησιακή λογική υλοποιείται σε Services/Controllers.

Χαρτογράφηση σε EF Core (Mapping Notes)

- **Keys & FKs:**
  - InvoiceLine.InvoiceId → Invoice.Id
  - Invoice.CustomerId → Customer.Id
  - Invoice.SupplierId → Supplier.Id
  - InvoiceLine.ItemId → Item.Id
  - InvoiceLine.ServiceId → Service.Id
- **Cascade Rules:**
  - Invoice → InvoiceLine : Cascade Delete (composition)
  - Προσοχή στις κύκλους cascade· έλεγξε το User → * αν χρειάζεται Restrict.
- **Indexes (ενδεικτικά):**
  - Invoice (Series, Year, Number) – μοναδικός συνδυασμός για αρίθμηση.
  - Invoice (CustomerId, IssueDate) – συχνά φίλτρα.
  - InvoiceLine (InvoiceId) – ταχύτητα στα joins.
  - Item (ItemCode) – unique.
- **Precision:**
  - Χρησιμοποίησε HasColumnType("decimal(18,2)") για ποσά/ΦΠΑ.

Κανόνες & Validation (στο επίπεδο Model/Service)

- **Customer / Supplier**: υποχρεωτικό Name, TaxIdentificationNumber (μορφή/μήκος), CustomerCode/SupplierCode μοναδικά.

- **Item / Service**: Code μοναδικός, UnitPrice ≥ 0, VAT στους επιτρεπτούς συντελεστές.
- **Invoice**: Series υποχρεωτικό, Number θετικός, IssueDate έγκυρη, τουλάχιστον μία γραμμή.
- **InvoiceLine**: Quantity > 0, UnitPrice ≥ 0, VatRate επιτρεπτός, υπολογισμοί γραμμής συνεπείς.

Παρατηρήσεις Σχεδίασης & Επεκτάσεις

1. **Υπηρεσίες σε γραμμές**:
   Επιλογές:
   - (α) InvoiceLine με ItemId ή ServiceId (nullable) + constraint να είναι συμπληρωμένο ένα από τα δύο.
   - (β) Ενιαία οντότητα Product με Type = Item/Service.
2. **Πληρωμές / Οικονομικά**:
   Προσθήκη Payment με σχέση Invoice (1:N) Payment, ενημέρωση IsPaid/PaidAmount.
3. **Τιμοκατάλογοι & Εκπτώσεις**:
   PriceList, DiscountPolicy με εφαρμογή στο InvoiceService.
4. **Audit fields**:
   CreatedAt, UpdatedAt, CreatedBy, UpdatedBy σε βασικές κλάσεις.
5. **Soft-delete**:
   Προσθήκη IsActive/IsDeleted και global query filters.

Δοκιμές (Model-level / Mapping)

ID	Έλεγχος	Αναμενόμενο
TC-CLASS-01	Uniqueness ItemCode	Απόπειρα διπλότυπου → σφάλμα
TC-CLASS-02	Cascade Invoice → InvoiceItem	Διαγραφή Invoice διαγράφει γραμμές
TC-CLASS-03	Precision ποσών	Αποθήκευση 2 δεκαδικών χωρίς rounding errors
TC-CLASS-04	Index Series+Year+Number	Γρήγορη αναζήτηση & προστασία αρίθμησης
TC-CLASS-05	Validation γραμμών	Quantity>0, UnitPrice≥0 → σφάλμα αν όχι

Πίνακας 4.6.2.2 Πίνακας Διάφορων Δοκιμών

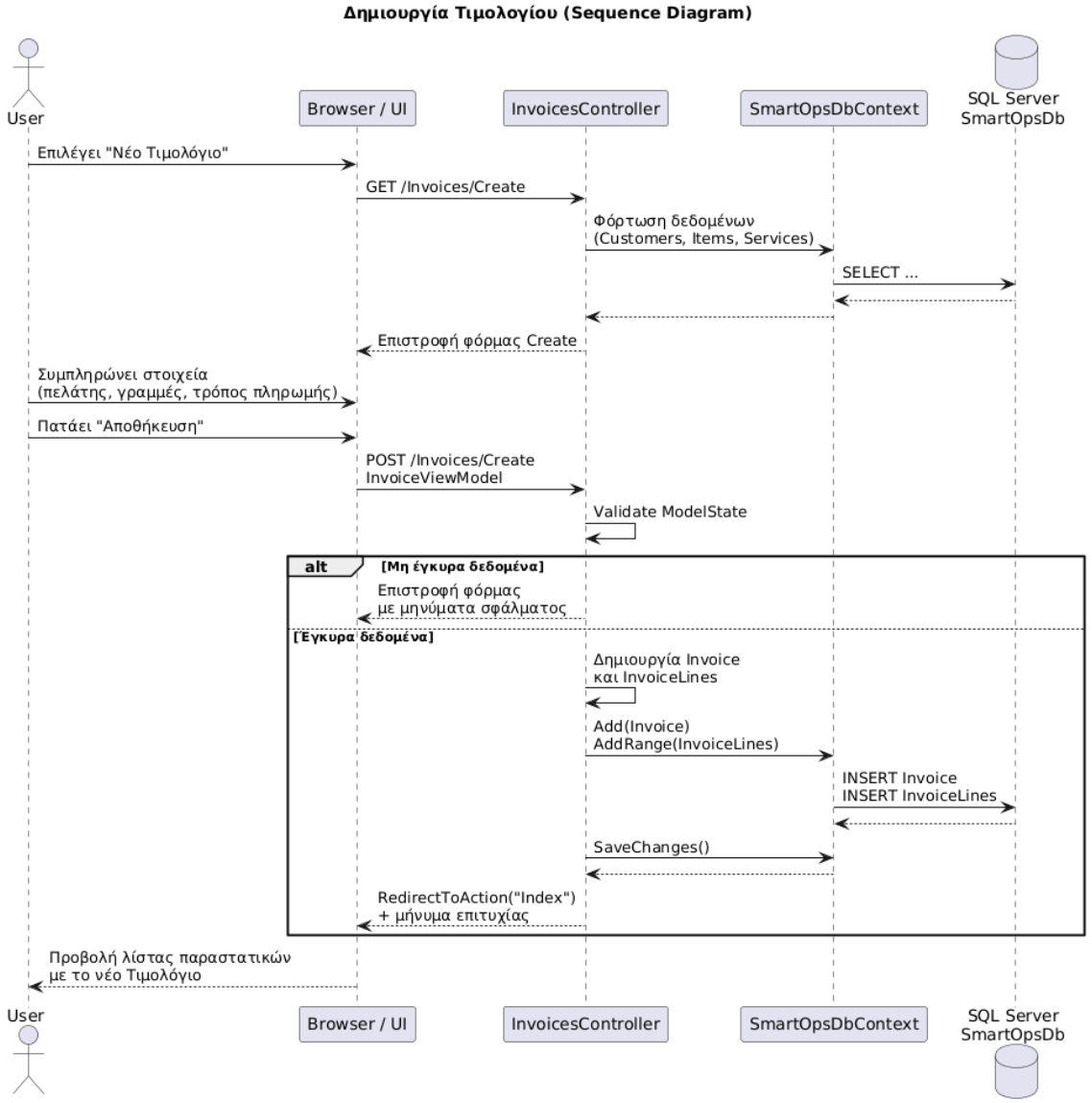
Συμπεράσματα

Το Διάγραμμα Κλάσεων του SmartOps αποτυπώνει μία λιτή και επεκτάσιμη δομή. Η διάκριση κεφαλίδας/γραμμών, η ιχνηλασιμότητα χρηστών και οι σαφείς συσχετίσεις εξασφαλίζουν

σταθερότητα, καλή απόδοση και ευκολία συντήρησης.
Με ελάχιστες επεκτάσεις (Payments, PriceLists, CRM κτλ) το μοντέλο καλύπτει ευρύτερα εμπορικά σενάρια χωρίς ανασχεδιασμό.

4.6.3 Sequence Diagram

Το παρακάτω διάγραμμα απεικονίζει αναλυτικά τη ροή της διαδικασίας δημιουργίας παραστατικού πώλησης στο SmartOps ERP Prototype. Απεικονίζονται οι αλληλεπιδράσεις μεταξύ του χρήστη, του γραφικού περιβάλλοντος, των controllers, των υπηρεσιών (services) και της βάσης δεδομένων SQL Server.



Εικόνα 4.6.3.1 – Sequence Diagram "Δημιουργία Τιμολογίου"

Σκοπός

SmartOps ERP – Πρωτότυπο Μοντέλο
Υλοποιημένο σε ASP.NET Core MVC (.NET 8
Technology) και Entity Framework

Σκοπός της συγκεκριμένης ροής είναι να παρουσιαστούν τα βήματα που εκτελούνται από τη στιγμή που ο χρήστης ανοίγει τη φόρμα “Νέο Παραστατικό”, μέχρι τη στιγμή που το σύστημα αποθηκεύει το τιμολόγιο και τις γραμμές του (InvoiceLines) στη βάση δεδομένων.

Η διαδικασία περιλαμβάνει έλεγχο εγκυρότητας των δεδομένων, δημιουργία του αντικειμένου Invoice μαζί με τις γραμμές του, αποθήκευση μέσω Entity Framework και επιστροφή ενημέρωσης στον χρήστη.

Περιγραφή Ροής

1. Άνοιγμα φόρμας (GET /Invoices/Create)

Ο χρήστης επιλέγει “Νέο Τιμολόγιο” από το UI. Ο InvoicesController καλεί τη μέθοδο Create() (HTTP GET), η οποία:

- φορτώνει τα απαιτούμενα δεδομένα (Customers, Items, Services),
- δημιουργεί το αντίστοιχο ViewModel,
- επιστρέφει τη φόρμα Create.cshtml.

Ο χρήστης βλέπει μια κενή φόρμα συμπλήρωσης τιμολογίου.

2. Συμπλήρωση δεδομένων και υποβολή (POST /Invoices/Create)

Ο χρήστης συμπληρώνει:

- στοιχεία κεφαλίδας (σειρά, αριθμός, ημερομηνία, πελάτης),
- τρόπο πληρωμής,
- γραμμές τιμολόγησης (InvoiceLines με Item ή Service, Quantity, UnitPrice, VAT).

Με το πάτημα του κουμπιού “Αποθήκευση”, το UI αποστέλλει όλα τα δεδομένα στον Controller μέσω POST.

3. Έλεγχος εγκυρότητας (Validation)

Ο Controller πραγματοποιεί ελέγχους ModelState και επιβεβαιώνει ότι:

- έχουν συμπληρωθεί υποχρεωτικά πεδία,
- υπάρχει τουλάχιστον μία γραμμή,
- οι ποσότητες και οι τιμές είναι έγκυρες (θετικοί αριθμοί),
- οι συντελεστές ΦΠΑ είναι αποδεκτοί,
- οι γραμμές έχουν είτε ItemId είτε ServiceId.

Αν υπάρχουν λάθη, το σύστημα επιστρέφει τη φόρμα στον χρήστη με αναλυτικά μηνύματα σφάλματος.

4. Επιχειρησιακή λογική & Αποθήκευση

Αν τα δεδομένα είναι έγκυρα, ο Controller:

1. **Δημιουργεί νέο αντικείμενο Invoice.**
2. **Δημιουργεί τη λίστα από InvoiceLine που δόθηκαν από το UI.**

3. **Υπολογίζει τα συνολικά ποσά:**
    - Καθαρή αξία (TotalNet)
    - Συνολικό ΦΠΑ (TotalVat)
    - Συνολική αξία (TotalGross)
4. **Καλεί το DbContext για:**
    - Add(Invoice)
    - AddRange(InvoiceLines)
5. **Εκτελεί SaveChanges()**
   Το Entity Framework κάνει:
    - INSERT στο Invoices
    - INSERT στο InvoiceLines

Σε περίπτωση σφάλματος της βάσης, εκτελείται rollback της συναλλαγής.

5. Απόκριση προς τον χρήστη

Μετά την επιτυχή καταχώριση:

- ο Controller κάνει RedirectToAction("Index"),
- αποστέλλεται μήνυμα επιτυχίας μέσω TempData,
- ο χρήστης βλέπει τη λίστα παραστατικών ενημερωμένη με το νέο τιμολόγιο.

Εναλλακτικά Σενάρια

α) Μη έγκυρα δεδομένα

- Επιστροφή στη φόρμα Create
- Προβολή μηνυμάτων ModelState
- Ο χρήστης βλέπει τα λάθη και τα διορθώνει

β) Σφάλμα βάσης ή αποτυχία αποθήκευσης

- Το EF αποτυγχάνει στο SaveChanges (π.χ. SQL σφάλμα)
- Γίνεται rollback
- Ο χρήστης βλέπει γενικό μήνυμα:
  “Η αποθήκευση δεν ολοκληρώθηκε – δοκιμάστε ξανά.”

γ) Μη εξουσιοδοτημένος χρήστης

- Το σύστημα εντοπίζει ότι UserId δεν υπάρχει στο session
- Γίνεται redirect στη σελίδα Login

Ασφάλεια και Ακεραιότητα

- Κάθε παραστατικό συνδέεται με το UserId του τρέχοντος χρήστη.

- Οι εγγραφές γίνονται μέσω EF Core με χρήση συναλλαγών.
- Η πρόσβαση στο module επιτρέπεται μόνο σε authenticated users.
- Όλα τα ποσά χρησιμοποιούν decimal για αποφυγή σφαλμάτων rounding.

Δοκιμές (Test Cases)

ID	Περιγραφή Δοκιμής	Είσοδος	Αναμενόμενο Αποτέλεσμα
TC-INV-01	Επιτυχής δημιουργία τιμολογίου	2 γραμμές με έγκυρα στοιχεία	Εμφάνιση οθόνης Details + μήνυμα επιτυχίας
TC-INV-02	Χωρίς γραμμές	Κενό InvoiceLines	Μήνυμα “τουλάχιστον μία γραμμή είναι υποχρεωτική”
TC-INV-03	Λανθασμένες τιμές	Quantity < 0 ή Price < 0	Μήνυμα σφάλματος validation
TC-INV-04	Σφάλμα βάσης	Π.χ. DB exception	Rollback + μήνυμα λάθους
TC-INV-05	Χωρίς login	Χρήστης χωρίς session	Redirect στη σελίδα Login
TC-INV-06	Μεγάλο πλήθος γραμμών	100+ γραμμές είδους	Επιτυχής δημιουργία χωρίς καθυστέρηση

΄Πίνακας 4.6.3.1 Περιπτώσεις Test Cases με το αναμενόμενο αποτέλεσμα

Συμπεράσματα

Το Sequence Diagram αναδεικνύει τη φυσική ροή της διαδικασίας δημιουργίας παραστατικού και τον τρόπο με τον οποίο συνεργάζονται UI – Controller – DbContext – Βάση δεδομένων.

Η διαστρωμάτωση της εφαρμογής, ο καθαρός διαχωρισμός ευθυνών και η χρήση EF Core εξασφαλίζουν:

- ευκολία συντήρησης,
- επεκτασιμότητα,
- ακρίβεια υπολογισμών,
- ακεραιότητα δεδομένων,
- ασφαλή και αξιόπιστη λειτουργία ακόμα και σε περιβάλλον πολλών χρηστών.

5.Λειτουργικότητα Πεδίων

5.1 Εισαγωγή στη Λειτουργικότητα

Η ενότητα αυτή παρουσιάζει αναλυτικά τη λειτουργικότητα του συστήματος SmartOps ERP Prototype, όπως υλοποιήθηκε στο πλαίσιο της παρούσας διπλωματικής εργασίας.

Το SmartOps έχει αναπτυχθεί με στόχο να προσφέρει ένα απλό, κατανοητό και επεκτάσιμο περιβάλλον διαχείρισης επιχειρησιακών δεδομένων, καλύπτοντας βασικές ανάγκες όπως η διαχείριση πελατών, προμηθευτών, προϊόντων, υπηρεσιών, τιμολογίων και στατιστικών αναφορών.

Η αρχιτεκτονική της εφαρμογής ακολουθεί το μοντέλο **MVC (Model–View–Controller)** του ASP.NET Core, επιτρέποντας τη σαφή διάκριση μεταξύ:

- **Model:** όπου ορίζονται οι δομές δεδομένων και οι κανόνες επικύρωσης,
- **View:** όπου παρουσιάζονται τα δεδομένα στον χρήστη μέσω δυναμικών σελίδων Razor,
- **Controller:** όπου υλοποιείται η επιχειρησιακή λογική και η αλληλεπίδραση με τη βάση δεδομένων.

Κάθε ενότητα (module) της εφαρμογής είναι πλήρως αυτόνομη, με δικό της controller, views και μοντέλα, αλλά ταυτόχρονα συνδέεται με τα υπόλοιπα μέρη του συστήματος μέσω κοινών σχέσεων στη βάση δεδομένων (π.χ. ένας πελάτης συνδέεται με τα τιμολόγιά του).

Η λειτουργικότητα της εφαρμογής χωρίζεται στα παρακάτω βασικά modules:

1. **Customers:** Διαχείριση πελατών (προσωπικά στοιχεία, κατηγοριοποίηση, χώρα, ΑΦΜ).
2. **Suppliers:** Καταχώριση και παρακολούθηση προμηθευτών.
3. **Items:** Διαχείριση προϊόντων με τιμές και ΦΠΑ.
4. **Services:** Διαχείριση υπηρεσιών με τιμές και ΦΠΑ.
5. **Invoices:** Δημιουργία, προβολή και αναζήτηση τιμολογίων πώλησης.
6. **Dashboard:** Εμφάνιση στατιστικών και αναφορών (π.χ. Top-5 πελάτες, συνολικές πωλήσεις, Top-5 προϊόντα).
7. **Users / Account:** Διαχείριση χρηστών, σύνδεση, εγγραφή και αλλαγή κωδικού.

Για κάθε module παρουσιάζονται στη συνέχεια:

- οι κύριες λειτουργίες,
- τα βασικά πεδία που περιλαμβάνονται στις φόρμες,
- ενδεικτικά αποσπάσματα κώδικα (Controller / ViewModel),
- εικόνες (screenshots) που αποτυπώνουν τη ροή της χρήσης,
- και η ανάλυση της ροής χρήστη (user flow) για κάθε βασική ενέργεια.

Με τον τρόπο αυτό, επιτυγχάνεται μια πλήρης αποτύπωση του τρόπου με τον οποίο ο χρήστης αλληλεπιδρά με το σύστημα και πώς οι λειτουργίες της εφαρμογής μεταφράζονται σε πραγματικά επιχειρησιακά αποτελέσματα.

5.2. Λειτουργικότητα πελατών (Customers Functionality)

Το module Customers αποτελεί μία από τις βασικότερες λειτουργικές ενότητες του SmartOps ERP Prototype και έχει ως στόχο τη διαχείριση των πελατών της επιχείρησης. Μέσα από αυτήν την ενότητα, ο χρήστης μπορεί να δημιουργεί, να επεξεργάζεται, να αναζητά ή να διαγράφει πελάτες, καθώς και να συνδέει κάθε πελάτη με τα αντίστοιχα τιμολόγιά του.

Σκοπός και ρόλος του module

Η διαχείριση πελατών είναι κρίσιμη για την οργάνωση των εμπορικών δραστηριοτήτων μιας επιχείρησης. Το SmartOps παρέχει έναν απλό και ευέλικτο μηχανισμό καταχώρισης στοιχείων, εξασφαλίζοντας ταυτόχρονα την εγκυρότητα των δεδομένων μέσω ελέγχων (validation).

Η οντότητα Customer χρησιμοποιείται ως σημείο αναφοράς για τη δημιουργία τιμολογίων και αποτελεί βασικό κόμβο στη λογική της εφαρμογής.

Περιγραφή λειτουργικότητας

Η λειτουργικότητα του module χωρίζεται σε τέσσερις βασικές ενέργειες:

1. **Προβολή λίστας πελατών (Index)**
   Ο χρήστης μπορεί να δει όλους τους καταχωρημένους πελάτες σε πίνακα, να κάνει αναζήτηση βάσει ονόματος ή ΑΦΜ ή και άλλων στοιχείων του και να μεταβεί γρήγορα σε επεξεργασία ή διαγραφή εγγραφής.
2. **Δημιουργία νέου πελάτη (Create)**
   Η φόρμα δημιουργίας περιλαμβάνει πεδία όπως:
   - *Customer Code*
   - *Name*
   - *Tax Identification Number (ΑΦΜ)*
   - *Country*
   - *Address*
   - *City*
   - *Postal Code*
   - *VAT Status*
   - *Customer Category* (χονδρικής ή λιανικής)

Κατά την αποθήκευση, πραγματοποιείται έλεγχος εγκυρότητας για το ΑΦΜ, τα υποχρεωτικά πεδία και τη μοναδικότητα του κωδικού πελάτη.

3. **Επεξεργασία υπαρχόντων πελατών (Edit)**
   Μέσα από τη φόρμα “Edit”, ο χρήστης μπορεί να ενημερώσει στοιχεία χωρίς να χαθεί η αναφορά σε υπάρχοντα τιμολόγια.
4. **Διαγραφή πελάτη (Delete)**
   Επιτρέπεται η διαγραφή μόνο αν δεν υπάρχουν τιμολόγια συνδεδεμένα με τον πελάτη, προκειμένου να διασφαλιστεί η ακεραιότητα των δεδομένων.

Η λειτουργικότητα του module υλοποιείται μέσω του **CustomersController**, ο οποίος αλληλεπιδρά με το SmartOpsDbContext για την αποθήκευση και ανάκτηση δεδομένων από τη βάση.

Παράδειγμα κώδικα δημιουργίας νέου πελάτη:

```
    // CREATE (POST)
    [HttpPost]
    [ValidateAntiForgeryToken]
    0 references
    public async Task<IActionResult> Create(
        [Bind("Name,TaxIdentificationNumber,Country,Address,City,PostalCode,VatStatus,CustomerCategory,CustomerCode")]
Customer customer)
    {

        // Έλεγχος αν ο χρήστης είναι συνδεδεμένος
        var uid = HttpContext.Session.GetInt32("UserId") ?? 0;
        if (uid == 0)
        {
            TempData["ErrorMessage"] = "Η συνεδρία έληξε. Παρακαλώ συνδεθείτε ξανά.";
            return RedirectToAction("Login", "Account");
        }


        ModelState.Remove(nameof(Customer.User));

        // Σύνδεση του πελάτη με τον χρήστη που τον δημιούργησε
        customer.UserId = uid;

        // Έλεγχος για manual ή αυτόματο CustomerCode
        var codeInput = customer.CustomerCode?.Trim();

        // Αν ο χρήστης αφήσει κενό, "*" ή "-" → γίνεται αυτόματη αρίθμηση
        var autoCode = string.IsNullOrEmpty(codeInput) || codeInput == "*" || codeInput == "-";

        if (autoCode)
        {
            // Παραγωγή του επόμενου διαθέσιμου κωδικού
            customer.CustomerCode = await GenerateNextCustomerCodeAsync(uid);
        }
        // Re-validation με τον πραγματικό CustomerCode
        ModelState.Clear();
        TryValidateModel(customer);

        if (!ModelState.IsValid)
        {
            // Φόρτωση dropdowns (VAT Status & Category) για να εμφανιστούν σωστά στο View
            SetDropDowns(customer.VatStatus, customer.CustomerCategory);

            // Αν ο χρήστης είχε ζητήσει αυτόματη αρίθμηση, ξαναδείξε "*"
            if (autoCode)
                customer.CustomerCode = "*";

            return View(customer);
        }
```

```
        // Αποθήκευση στη βάση δεδομένων
        try
        {
            await _customerService.AddAsync(customer);

            TempData["SuccessMessage"] = "Ο πελάτης δημιουργήθηκε με επιτυχία.";
            return RedirectToAction(nameof(Index));
        }
        // Unique violation (διπλό CustomerCode)
        catch (DbUpdateException ex) when (ex.InnerException is SqlException sql &&
                                           (sql.Number == 2601 || sql.Number == 2627))
        {
            if (autoCode)
            {
                // Σφάλμα κατά την αυτόματη δημιουργία κωδικού
                ModelState.AddModelError(string.Empty, "Παρουσιάστηκε σφάλμα στην αυτόματη δημιουργία κωδικού.");
            }
            else
            {
                // Ο κωδικός υπάρχει ήδη
                ModelState.AddModelError(nameof(Customer.CustomerCode), "Ο κωδικός χρησιμοποιείται ήδη.");
            }
        }
        // Γενικά DB errors
        catch (DbUpdateException ex)
        {
            ModelState.AddModelError(string.Empty, "Σφάλμα βάσης δεδομένων: " +
                                                   (ex.InnerException?.Message ?? ex.Message));
        }
        // Επιστροφή στη φόρμα αν κάτι απέτυχε
        SetDropDowns(customer.VatStatus, customer.CustomerCategory);

        if (autoCode)
            customer.CustomerCode = "*";   // UX: εμφανίζεται όπως το έδωσε ο χρήστης

        return View(customer);
    }
```

Στον παραπάνω κώδικα φαίνεται η τυπική λογική αποθήκευσης:

- Έλεγχος εγκυρότητας Κωδικού, Επωνυμίας και ΑΦΜ (ModelState.IsValid)
- Εισαγωγή νέας εγγραφής στο context
- Αποθήκευση αλλαγών στη βάση (SaveChangesAsync)
- Προβολή μηνύματος επιτυχίας στον χρήστη μέσω TempData

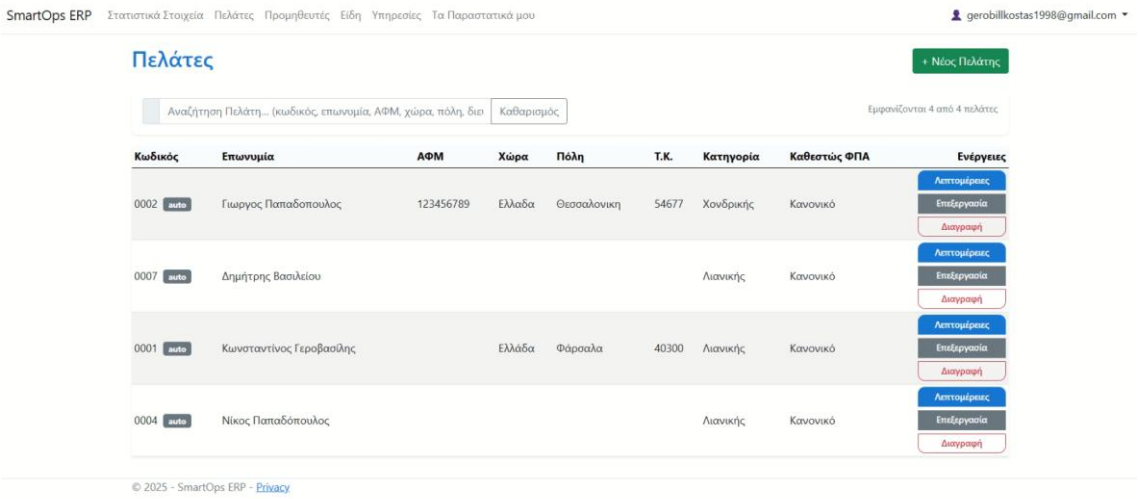
Ανάλυση ροής χρήστη

1. Ο χρήστης επιλέγει «Νέος Πελάτης».
2. Συμπληρώνει τα πεδία στη φόρμα.
3. Το σύστημα ελέγχει τα δεδομένα και εμφανίζει τυχόν σφάλματα.
4. Με επιτυχές validation, η εγγραφή αποθηκεύεται στη βάση.
5. Εμφανίζεται μήνυμα επιτυχίας και ο χρήστης επιστρέφει στη λίστα πελατών.

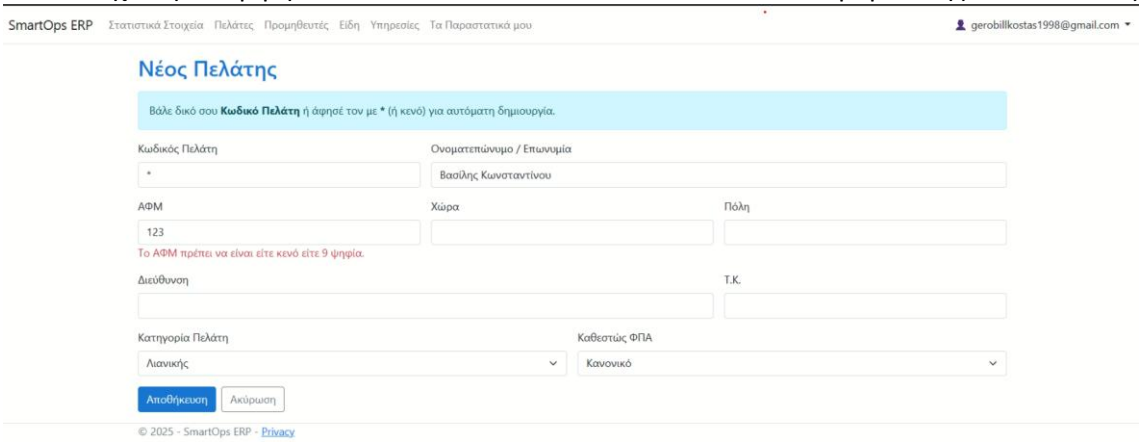
Η παραπάνω διαδικασία εξασφαλίζει απλότητα, αξιοπιστία και σαφή ενημέρωση του χρήστη για κάθε ενέργεια που εκτελεί.

Παρακάτω ακολουθούν κάποιες εικόνες με τις λειτουργίες της φόρμας του πελάτη, με ελέγχους εγκυρότητας, με τυχόν σφάλματα και με μηνύματα επιτυχούς αποθήκευσης.

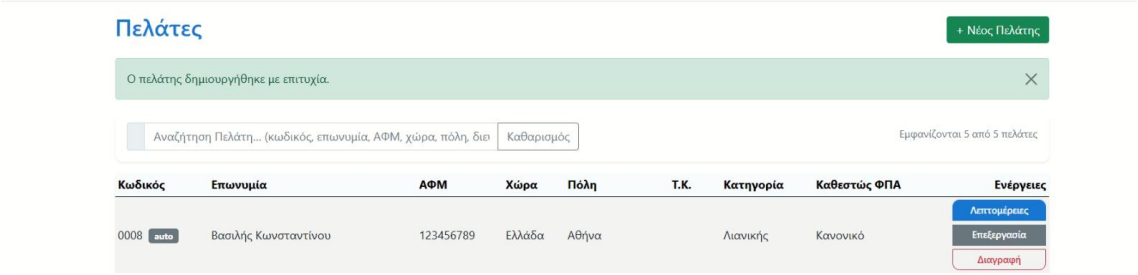
Εικόνες – Ενδεικτικές Οθόνες



Εικόνα 5.2.1: Οθόνη λίστας πελατών (Index) με δυνατότητα αναζήτησης και φίλτρων.



Εικόνα 5.2.2: Φόρμα δημιουργίας νέου πελάτη με έλεγχο εγκυρότητας (validation messages).



Εικόνα 5.2.3: Μήνυμα επιτυχούς αποθήκευσης μέσω TempData.

5.3 Λειτουργικότητα Προμηθευτών (Suppliers Functionality)

Το module **Suppliers** έχει ως σκοπό τη διαχείριση των προμηθευτών της επιχείρησης, παρέχοντας όλες τις βασικές λειτουργίες για την εισαγωγή, επεξεργασία, προβολή και διαγραφή εγγραφών. Οι προμηθευτές, όπως και οι πελάτες, αποτελούν θεμελιώδη επιχειρησιακή οντότητα στο SmartOps ERP Prototype και μπορούν να συνδεθούν με διάφορες μελλοντικές επεκτάσεις, όπως αγορές, αποθήκες ή αναφορές προμηθευτών.

Σκοπός και ρόλος του module

Το **module Suppliers** σχεδιάστηκε για να προσφέρει έναν ομοιόμορφο και απλό τρόπο καταχώρισης στοιχείων προμηθευτών, με στόχο την ταχεία πρόσβαση και την αποφυγή λαθών κατά την εισαγωγή δεδομένων.

Η δομή του είναι σχεδόν πανομοιότυπη με αυτήν του module Customers, κάτι που εξασφαλίζει ομοιομορφία στη χρήση και μειωμένη εκμάθηση για τον χρήστη.

Περιγραφή λειτουργικότητας

Η ενότητα Suppliers περιλαμβάνει τα εξής βασικά μέρη:

1. **Προβολή λίστας προμηθευτών (Index)**
   Εμφανίζεται ένας πίνακας με όλους τους καταχωρημένους προμηθευτές. Παρέχονται λειτουργίες αναζήτησης βάσει ονόματος ή χώρας, καθώς και κουμπιά για προσθήκη, επεξεργασία ή διαγραφή εγγραφών.

2. **Δημιουργία νέου προμηθευτή (Create)**
   Ο χρήστης μπορεί να εισάγει νέα εγγραφή συμπληρώνοντας πεδία όπως:
   - *Supplier Code*
   - *Name*
   - *Tax Identification Number (ΑΦΜ)*
   - *Country*
   - *Address*
   - *City*
   - *Postal Code*
   - *Supplier Category*
   - *Vat Status*

Εφαρμόζονται έλεγχοι εγκυρότητας για το ΑΦΜ και τα υποχρεωτικά πεδία.

3. **Επεξεργασία (Edit)**
   Δυνατότητα ενημέρωσης στοιχείων ενός προμηθευτή χωρίς διαγραφή της υπάρχουσας εγγραφής.
4. **Διαγραφή (Delete)**
   Ο χρήστης μπορεί να αφαιρέσει έναν προμηθευτή μόνο αν δεν υπάρχουν ενεργές εγγραφές που σχετίζονται με αυτόν.

Τεχνική υλοποίηση

Η λειτουργικότητα του module “Suppliers” βασίζεται στον **SuppliersController**, ο οποίος ακολουθεί το ίδιο μοτίβο CRUD όπως οι πελάτες, χρησιμοποιώντας το SmartOpsDbContext.

Παράδειγμα μεθόδου διαγραφής προμηθευτή:

```
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
0 references
public async Task<IActionResult> DeleteConfirmed(int id)
{
    if (CurrentUserId == 0) return RedirectToAction("Login", "Account");
    var supplier = await _supplierService.GetByIdForUserAsync(id, CurrentUserId);
    if (supplier == null) return NotFound();

    await _supplierService.DeleteAsync(supplier);
    TempData["SuccessMessage"] = "Ο προμηθευτής διαγράφηκε.";
    return RedirectToAction(nameof(Index));
}
```

Στο παραπάνω παράδειγμα:

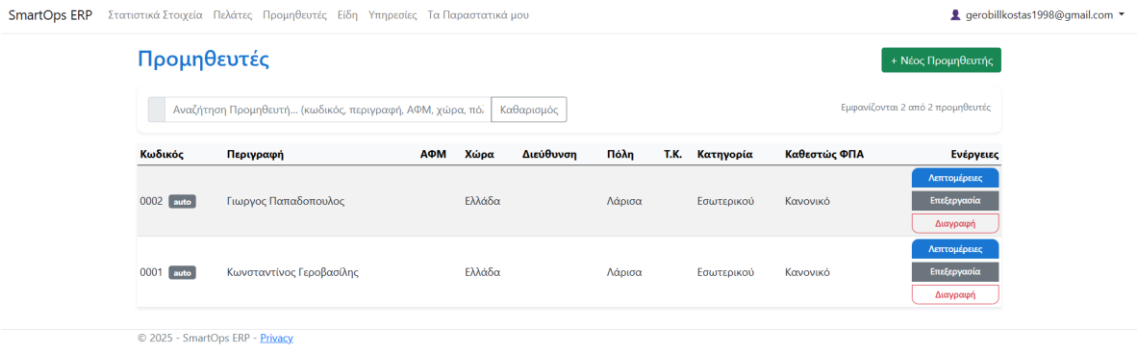
- Ανακτάται ο προμηθευτής βάσει id.
- Αν βρεθεί, διαγράφεται από τη βάση και εμφανίζεται μήνυμα επιτυχίας.
- Η λειτουργία ολοκληρώνεται με ανακατεύθυνση στην αρχική λίστα.

Ανάλυση ροής χρήστη

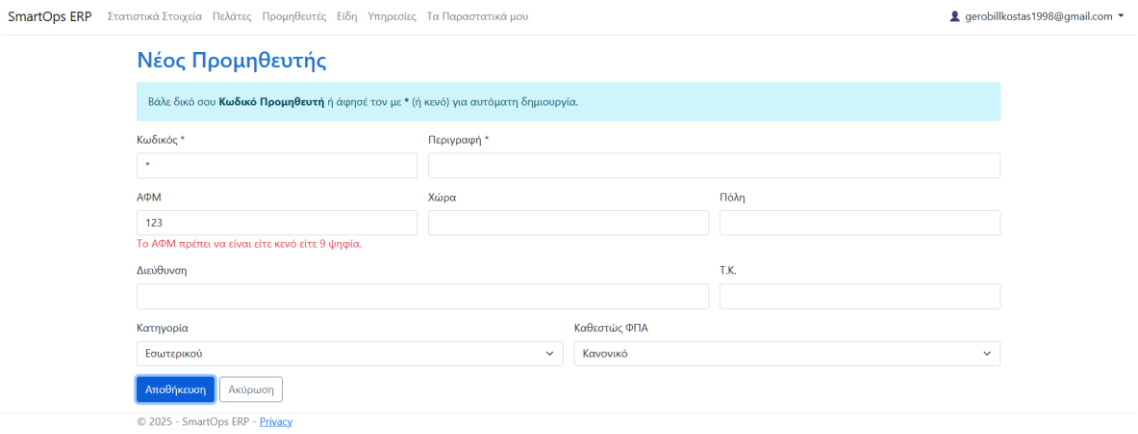
1. Ο χρήστης επιλέγει “Προμηθευτές” από το βασικό μενού.
2. Εμφανίζεται η λίστα όλων των εγγραφών (Index).
3. Μέσω του κουμπιού “Νέος Προμηθευτής” ανοίγει η φόρμα Create.
4. Μετά την καταχώριση, εμφανίζεται μήνυμα επιτυχίας.
5. Εφόσον χρειαστεί, ο χρήστης μπορεί να επεξεργαστεί ή να διαγράψει μια εγγραφή.

Η ομοιότητα της ροής με το module “Customers” επιτρέπει στον χρήστη να προσαρμοστεί εύκολα στη διεπαφή, διατηρώντας την εμπειρία χρήσης συνεπή σε όλο το σύστημα.

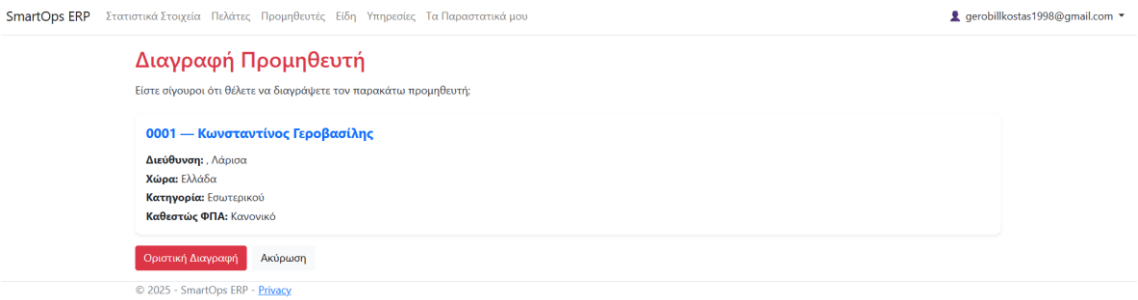
Εικόνες – Ενδεικτικές Οθόνες



Εικόνα 5.3.1: Οθόνη λίστας προμηθευτών (Index) με δυνατότητα αναζήτησης και κουμπιά ενεργειών.



Εικόνα 5.3.2: Φόρμα δημιουργίας νέου προμηθευτή με πεδία ΑΦΜ, χώρα και πόλη και έλεγχο εγκυρότητας ΑΦΜ.



Εικόνα 5.3.3: Εμφάνιση προειδοποιητικού μηνύματος οριστικής διαγραφής προμηθευτή.

5.4 Λειτουργικότητα Ειδών

Το module **Items** έχει ως στόχο τη διαχείριση των προϊόντων που εμπορεύεται η επιχείρηση. Αποτελεί βασικό κομμάτι του SmartOps ERP Prototype, καθώς τα είδη χρησιμοποιούνται απευθείας κατά τη δημιουργία τιμολογίων και στον υπολογισμό των συνολικών αξιών.

Σκοπός και ρόλος του module

Μέσα από το **module Items,** ο χρήστης μπορεί να καταχωρεί, να ενημερώνει και να παρακολουθεί όλα τα διαθέσιμα προϊόντα που προσφέρει η επιχείρηση.
Η σωστή οργάνωση των ειδών εξασφαλίζει ομαλή τιμολόγηση, ακριβή υπολογισμό Φ.Π.Α. και σαφή κατηγοριοποίηση των προϊόντων, κάτι που αποτελεί θεμέλιο για οποιοδήποτε ERP σύστημα.

Περιγραφή λειτουργικότητας

Το module προσφέρει τις ακόλουθες βασικές λειτουργίες:

1. **Προβολή λίστας ειδών (Index)**
   Εμφανίζεται ένας πίνακας με όλα τα είδη, ταξινομημένος ανά κατηγορία ή όνομα. Παρέχεται δυνατότητα αναζήτησης με βάση το όνομα ή τον κωδικό προϊόντος.
2. **Δημιουργία νέου είδους (Create)**
   Η φόρμα δημιουργίας περιλαμβάνει τα εξής βασικά πεδία:
   - *Item Code*
   - *Description*
   - *Unit*
   - *VAT*
   - *Retail Price*
   - *WholeSale Price*

Εφαρμόζεται validation για να αποτρέπεται η εισαγωγή αρνητικών τιμών ή μη έγκυρου ποσοστού Φ.Π.Α.

3. **Επεξεργασία (Edit)**
   Ο χρήστης μπορεί να τροποποιήσει στοιχεία όπως την τιμή ή την κατηγορία ενός προϊόντος. Οι αλλαγές εφαρμόζονται αυτόματα και στα νέα τιμολόγια που θα δημιουργηθούν.
4. **Διαγραφή (Delete)**
   Αν δεν υπάρχουν τιμολόγια που περιέχουν το συγκεκριμένο προϊόν, η εγγραφή μπορεί να διαγραφεί οριστικά.

Τεχνική υλοποίηση

Η υλοποίηση βασίζεται στον **ItemsController**, ο οποίος επικοινωνεί με το SmartOpsDbContext για την ανάγνωση και ενημέρωση της βάσης.
Ο controller υποστηρίζει δυναμική αναζήτηση και pagination για βελτιωμένη εμπειρία χρήσης.

Παράδειγμα μεθόδου Index με δυνατότητα αναζήτησης:

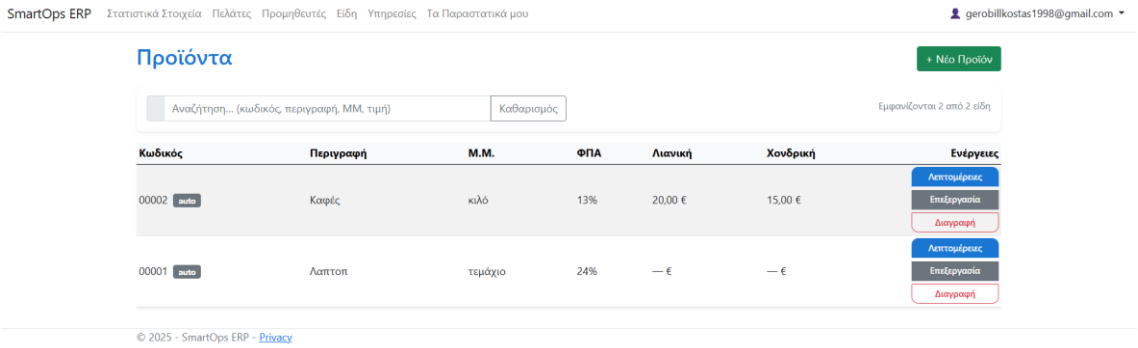
```
// INDEX
[HttpGet]
3 references
public async Task<IActionResult> Index()
{
    if (CurrentUserId == 0) return RedirectToAction("Login", "Account");
    var items = await _itemService.GetAllByUserAsync(CurrentUserId);
    return View(items);
}
```

Με τον τρόπο αυτό, ο χρήστης μπορεί να αναζητήσει προϊόντα γρήγορα, χωρίς καθυστέρηση, ακόμα και σε μεγάλες βάσεις δεδομένων.

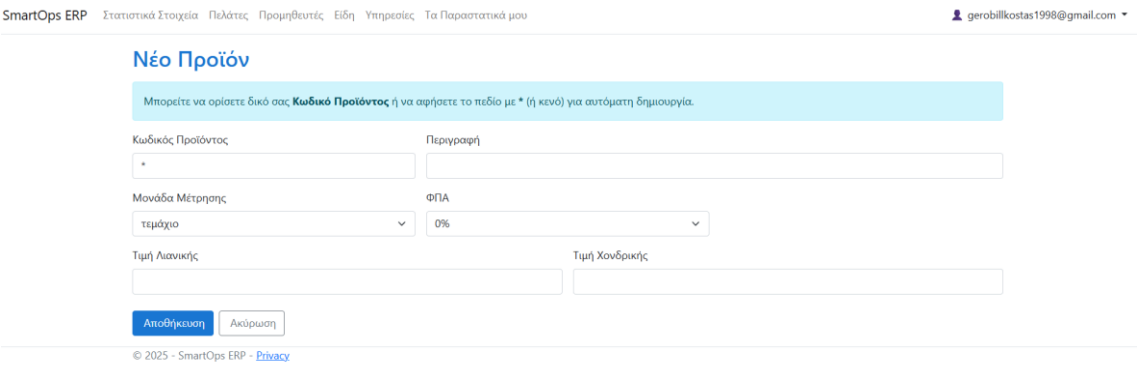
Ανάλυση ροής χρήστη

1. Ο χρήστης επιλέγει “Προϊόντα” από το κεντρικό μενού.
2. Εμφανίζεται η λίστα με τα καταχωρημένα προϊόντα.
3. Πατώντας “Νέο Προϊόν”, μεταβαίνει στη φόρμα καταχώρισης.
4. Συμπληρώνει τα απαιτούμενα πεδία και αποθηκεύει.
5. Εμφανίζεται μήνυμα επιτυχίας και το νέο προϊόν εμφανίζεται στη λίστα.

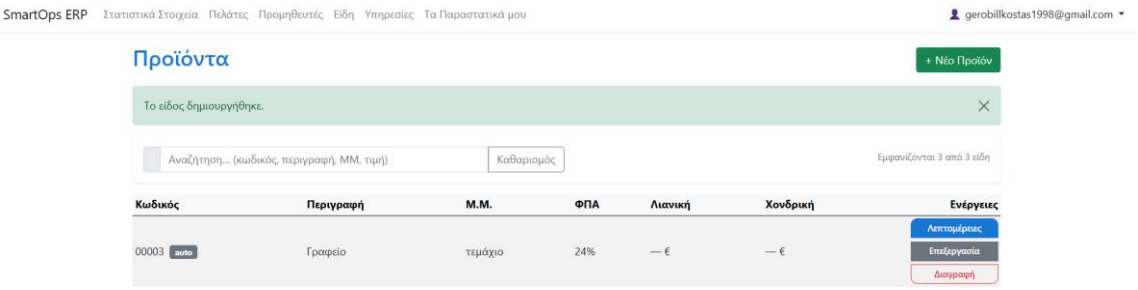
Εικόνες – Ενδεικτικές Οθόνες



Εικόνα 5.4.1: Οθόνη λίστας προϊόντων (Index) με πεδίο αναζήτησης.



Εικόνα 5.4.2: Φόρμα δημιουργίας νέου είδους με dropdown επιλογές για ΦΠΑ και μονάδα μέτρησης.



Εικόνα 5.4.3: Μήνυμα επιτυχούς αποθήκευσης προϊόντος.

5.5 Λειτουργικότητα Υπηρεσιών

Το module **Services** είναι υπεύθυνο για τη διαχείριση των παρεχόμενων υπηρεσιών της επιχείρησης. Σε αντίθεση με τα προϊόντα (Items), οι υπηρεσίες δεν διαθέτουν φυσικό απόθεμα, ωστόσο τιμολογούνται με βάση τον χρόνο ή το συμφωνημένο ποσό. Το module έχει σχεδιαστεί ώστε να παρέχει ευελιξία και απλότητα στην καταχώριση υπηρεσιών, διατηρώντας ταυτόχρονα τη συνέπεια στη δομή του συστήματος.

Σκοπός και ρόλος του module

Ο κύριος σκοπός του module “Services” είναι να επιτρέπει στον χρήστη να καταγράφει, ενημερώνει και διαχειρίζεται όλες τις υπηρεσίες που προσφέρει η επιχείρηση — όπως τεχνική υποστήριξη, συντήρηση, συμβουλευτικές υπηρεσίες ή λογιστικές εργασίες.
Οι υπηρεσίες χρησιμοποιούνται άμεσα κατά τη δημιουργία τιμολογίων, προσφέροντας στον χρήστη τη δυνατότητα να τιμολογεί όχι μόνο προϊόντα αλλά και υπηρεσίες.

Περιγραφή λειτουργικότητας

1. **Προβολή λίστας υπηρεσιών (Index)**
   Ο χρήστης μπορεί να δει όλες τις καταχωρημένες υπηρεσίες σε μορφή πίνακα.
   Παρέχονται φίλτρα αναζήτησης με βάση το όνομα της υπηρεσίας ή την κατηγορία της.
2. **Δημιουργία νέας υπηρεσίας (Create)**
   Η φόρμα δημιουργίας περιλαμβάνει πεδία όπως:
   - *Service Code*
   - *Description*
   - *Unit*
   - VAT
   - *Retail Price*
   - *WholeSale Price*

Εφαρμόζονται κανόνες εγκυρότητας ώστε να αποφεύγεται η εισαγωγή αρνητικών τιμών ή ποσοστών ΦΠΑ εκτός των επιτρεπτών ορίων (0–24%).

3. **Επεξεργασία (Edit)**
   Ο χρήστης μπορεί να ενημερώνει τα στοιχεία μιας υπηρεσίας, όπως την περιγραφή ή την τιμή της, χωρίς να επηρεάζονται οι ήδη εκδοθείσες εγγραφές.
4. **Διαγραφή (Delete)**
   Η διαγραφή μιας υπηρεσίας επιτρέπεται μόνο εφόσον δεν έχει συσχετιστεί με υπάρχοντα τιμολόγια, ώστε να διασφαλίζεται η ακεραιότητα της βάσης δεδομένων.

Τεχνική υλοποίηση

Η υλοποίηση βασίζεται στον **ServicesController**, ο οποίος ακολουθεί την ίδια λογική CRUD όπως τα υπόλοιπα modules και επικοινωνεί με τον πίνακα Services στη βάση μέσω του SmartOpsDbContext.

Παράδειγμα κώδικα δημιουργίας υπηρεσίας:

```
// CREATE (POST)
[HttpPost]
[ValidateAntiForgeryToken]
0 references
public async Task<IActionResult> Create(Service service)
{
    // Έλεγχος αν έχει λήξει η συνεδρία
    var uid = HttpContext.Session.GetInt32("UserId") ?? 0;
    if (uid == 0)
        return RedirectToAction("Login", "Account");

    // Δεν θέλουμε validation στο navigation property
    ModelState.Remove(nameof(Service.User));

    // Ανάθεση χρήστη στην υπηρεσία
    service.UserId = uid;

    // Έλεγχος πεδίου "Κωδικός"
    // Τι έγραψε ο χρήστης στο input
    var codeInput = service.ServiceCode?.Trim();

    // Αν είναι κενό, '*' ή '-' τότε ζητά αυτόματο κωδικό
    var autoCode = string.IsNullOrEmpty(codeInput)
                   || codeInput == "*"
                   || codeInput == "-";

    // Αυτόματη αρίθμηση με * στον κωδικό: 0001, 0002, 0003...
    if (autoCode)
    {
        service.ServiceCode = await GenerateNextServiceCodeAsync(uid);
    }

    // Ξανακάνουμε validate με τον τελικό κωδικό
    ModelState.Clear();
    TryValidateModel(service);

    // Αν υπάρχουν validation errors τότε επέστρεψε στη φόρμα
    if (!ModelState.IsValid)
    {
        // Ξαναγέμισμα dropdowns
        SetUnits(service.Unit);
        SetVATs(service.VAT);

        // Αν είχε ζητήσει auto, να ξαναφαίνεται '*' στην φόρμα για να μην μπερδευτεί
        if (autoCode) service.ServiceCode = "*";

        return View(service);
    }
```

```
        // Αποθήκευση της υπηρεσίας
        try
        {
            await _serviceService.AddAsync(service);
            TempData["SuccessMessage"] = "Η υπηρεσία δημιουργήθηκε με επιτυχία.";
            return RedirectToAction(nameof(Index));
        }
        // Unique constraint στο ServiceCode
        catch (DbUpdateException ex) when (ex.InnerException is SqlException sql &&
                                           (sql.Number == 2601 || sql.Number == 2627))
        {
            // έλεγχος του αυτόματου κωδικού αν υπάρξει κάποιο σφάλμα ή απλα αν χρησιμοποιείται ήδη
            if (autoCode)
                ModelState.AddModelError(string.Empty, "Παρουσιάστηκε σφάλμα στην αυτόματη δημιουργία κωδικού.");
            else
                ModelState.AddModelError(nameof(Service.ServiceCode), "Ο κωδικός χρησιμοποιείται ήδη.");
        }
        catch (DbUpdateException ex)
        {
            // Γενικό database error
            ModelState.AddModelError(string.Empty,
                "DB error: " + (ex.InnerException?.Message ?? ex.Message));
        }

        SetUnits(service.Unit);
        SetVATs(service.VAT);

        if (autoCode) service.ServiceCode = "*";

        return View(service);
    }
```

Η μέθοδος αυτή:

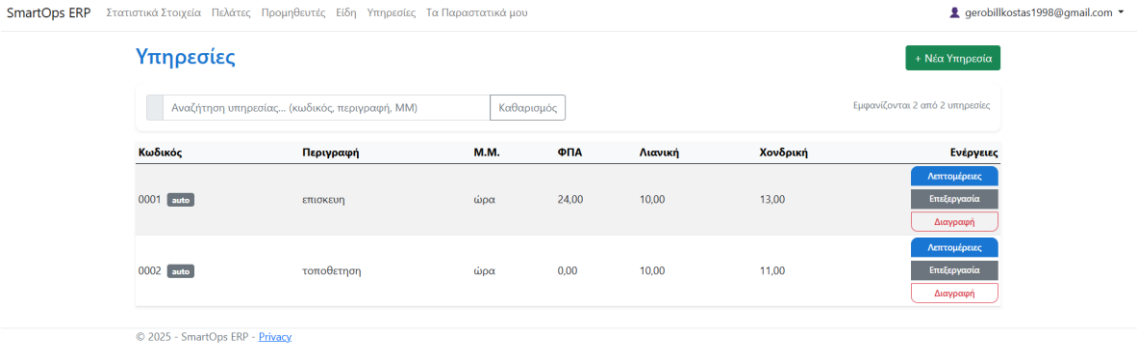
- Ελέγχει την εγκυρότητα των δεδομένων (ModelState.IsValid).
- Αποθηκεύει τη νέα υπηρεσία στη βάση δεδομένων.
- Εμφανίζει μήνυμα επιτυχίας στον χρήστη.
- Επιστρέφει στην αρχική λίστα υπηρεσιών.

Ανάλυση ροής χρήστη

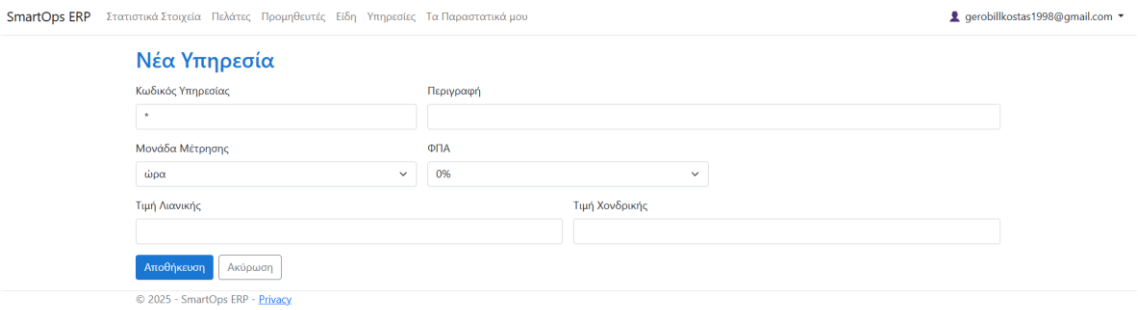
1. Ο χρήστης μεταβαίνει στο μενού “Υπηρεσίες”.
2. Προβάλλεται η λίστα των διαθέσιμων υπηρεσιών.
3. Επιλέγει “Νέα Υπηρεσία” και συμπληρώνει τη φόρμα.
4. Το σύστημα ελέγχει την εγκυρότητα των τιμών και αποθηκεύει την εγγραφή.
5. Εμφανίζεται μήνυμα επιτυχίας και η υπηρεσία προστίθεται στη λίστα.

Η ροή παραμένει απλή και συνεπής με τα υπόλοιπα modules του SmartOps, διευκολύνοντας τη χρήση από μη εξειδικευμένους χρήστες.

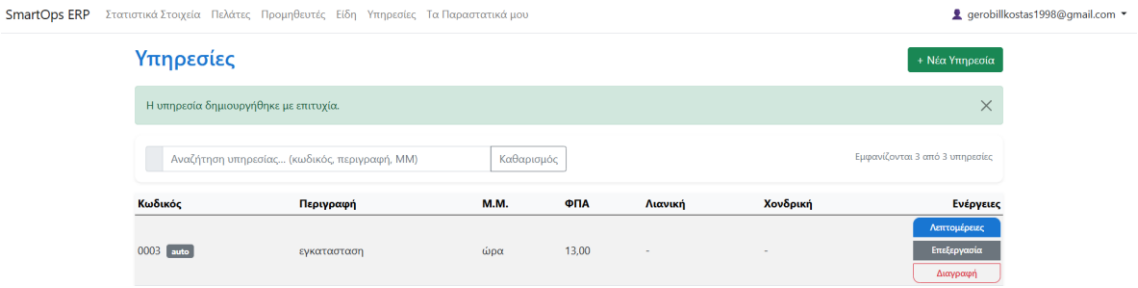
Εικόνες – Ενδεικτικές Οθόνες



Εικόνα 5.5.1: Οθόνη λίστας υπηρεσιών (Index) με δυνατότητα αναζήτησης και φίλτρων.



Εικόνα 5.5.2: Φόρμα δημιουργίας νέας υπηρεσίας με πεδία περιγραφής, και ορισμού τιμής λιανικής και τιμής χονδρικής.



Εικόνα 5.5.3: Μήνυμα επιτυχούς αποθήκευσης μέσω TempData.

5.6 Λειτουργικότητα Παραστατικών

Το module **Invoices** αποτελεί την καρδιά του συστήματος SmartOps ERP Prototype, καθώς διαχειρίζεται την έκδοση, αποθήκευση και προβολή των παραστατικών πώλησης και αγοράς. Μέσω αυτής της ενότητας, επιτυγχάνεται η ενοποίηση όλων των υπολοίπων modules, καθώς κάθε τιμολόγιο συνδέεται άμεσα με πελάτες προμηθευτές, είδη ή υπηρεσίες, αλλά και με τον χρήστη που το δημιουργεί.

Σκοπός και ρόλος του module

Ο βασικός σκοπός του module “Invoices” είναι να παρέχει ένα πλήρες και ευέλικτο περιβάλλον τιμολόγησης, με δυνατότητα δημιουργίας, επεξεργασίας, προβολής και αναζήτησης τιμολογίων. Το module υποστηρίζει τόσο παραστατικά πώλησης όσο και υπηρεσιών, επιτρέποντας την άμεση καταγραφή συναλλαγών και την αυτόματη ενημέρωση των σχετικών αναφορών στο Dashboard.

Περιγραφή λειτουργικότητας

Η ενότητα Invoices περιλαμβάνει τις ακόλουθες λειτουργίες:

1. **Προβολή λίστας τιμολογίων (Index)**
   Εμφανίζει όλα τα εκδοθέντα τιμολόγια, ταξινομημένα κατά ημερομηνία ή πελάτη. Παρέχεται δυνατότητα αναζήτησης και φιλτραρίσματος βάσει πελάτη, ημερομηνίας ή αριθμού παραστατικού.
2. **Δημιουργία νέου τιμολογίου (Create)**
   Η φόρμα δημιουργίας τιμολογίου είναι το πιο σύνθετο σημείο του συστήματος και περιλαμβάνει:
   - Επιλογή σειράς παραστατικού
   - Επιλογή πελάτη από λίστα
   - Επιλογή είδους ή υπηρεσίας
   - Καθορισμό ποσότητας, τιμής.

- Αυτόματο υπολογισμό καθαρής αξίας, ΦΠΑ και συνολικού ποσού
- Επιλογή τρόπου πληρωμής (μετρητά, κάρτα, τραπεζική μεταφορά)

3. **Προβολή τιμολογίου (Details)**
   Προσφέρει αναλυτική παρουσίαση του τιμολογίου με όλα τα στοιχεία του πελάτη, τις γραμμές ειδών και το τελικό ποσό.
4. **Διαγραφή τιμολογίου (Delete)**
   Δυνατότητα διαγραφής μόνο για παραστατικά που δεν έχουν ακόμη αποσταλεί στο MyData.

Δομή δεδομένων και σχέσεων

Το module Invoices αποτελείται από δύο βασικά μοντέλα:

- **Invoice** – περιέχει τα γενικά στοιχεία του τιμολογίου (ID, Ημερομηνία, Πελάτης, Ποσότητα, Τιμή, Τρόπος Πληρωμής κτλ).
- **InvoiceLine** – περιλαμβάνει τις γραμμές του τιμολογίου (InvoiceId, ItemId, ServiceId, Quantity, UnitPrice, VatRate).

Η σχέση μεταξύ των δύο μοντέλων είναι **ένα-προς-πολλά (1–N)**, καθώς ένα τιμολόγιο μπορεί να περιέχει πολλές γραμμές ειδών ή υπηρεσιών.

Τεχνική υλοποίηση

Η λειτουργικότητα του module υλοποιείται μέσω του **InvoicesController**, ενώ η επιχειρησιακή λογική (όπως υπολογισμοί και validation) συγκεντρώνεται στην κλάση InvoiceService.

Παράδειγμα δημιουργίας τιμολογίου:

```
// CREATE (POST)
[HttpPost]
[ValidateAntiForgeryToken]
0 references
public async Task<IActionResult> Create(InvoiceCreateVm vm)
{
    // Αν η συνεδρία έχει λήξει τότε επέστρεψε στο login
    if (CurrentUserId == 0)
        return RedirectToAction("Login", "Account");

    // Έλεγχος ότι ο τύπος παραστατικού είναι ένας από τους τρεις επιτρεπτούς
    if (vm.InvoiceType != "Items" &&
        vm.InvoiceType != "Services" &&
        vm.InvoiceType != "Purchases")
    {
        return RedirectToAction(nameof(SelectType));
    }

    // Φιλτράρουμε γραμμές που δεν έχουν ουσιαστικά δεδομένα
    vm.Lines = vm.Lines?
        .Where(l => l.CatalogId > 0 && l.Quantity > 0)
        .ToList() ?? new();

    //Έλεγχος πελάτη / προμηθευτή
    // Για Purchases, στο CustomerId έχουμε προμηθευτή
    var customerExists = await _db.Customers
        .AnyAsync(c => c.Id == vm.CustomerId && c.UserId == CurrentUserId);

    if (!customerExists)
    {
        ModelState.AddModelError(nameof(vm.CustomerId),
            vm.InvoiceType == "Purchases"
            ? "Μη έγκυρος προμηθευτής."
            : "Μη έγκυρος πελάτης.");
    }
```

```
        // Αν υπάρχουν λάθη στο ModelState ή δεν υπάρχουν γραμμές τότε γύρνα στο View
        if (!ModelState.IsValid || vm.Lines.Count == 0)
        {
            if (vm.Lines.Count == 0)
                ModelState.AddModelError("", "Προσθέστε τουλάχιστον μία γραμμή.");

            // Ξαναγέμισμα dropdowns
            await FillDropdownsAsync(vm.InvoiceType);

            // Αναδημιουργία καταλόγου ειδών/υπηρεσιών (απαραίτητο για να δουλεύουν τα dropdowns)
            var uid = CurrentUserId;

            if (vm.InvoiceType == "Items" || vm.InvoiceType == "Purchases")
            {
                vm.CatalogItems = await _db.Items
                    .Where(i => i.UserId == uid)
                    .OrderBy(i => i.Description)
                    .Select(i => new InvoiceCreateVm.CatalogItemVm
                    {
                        Type = "Item",
                        Id = i.Id,
                        Code = i.ItemCode,
                        Description = i.Description,
                        RetailPrice = i.RetailPrice,
                        WholesalePrice = i.WholesalePrice,
                        VatRate = i.VAT
                    }).ToListAsync();
            }
            else
            {
                vm.CatalogItems = await _db.Services
                    .Where(s => s.UserId == uid)
                    .OrderBy(s => s.Description)
                    .Select(s => new InvoiceCreateVm.CatalogItemVm
                    {
            {
                vm.CatalogItems = await _db.Services
                    .Where(s => s.UserId == uid)
                    .OrderBy(s => s.Description)
                    .Select(s => new InvoiceCreateVm.CatalogItemVm
                    {
                        Type = "Service",
                        Id = s.Id,
                        Code = s.ServiceCode,
                        Description = s.Description,
                        RetailPrice = s.RetailPrice,
                        WholesalePrice = s.WholesalePrice,
                        VatRate = s.VAT
                    }).ToListAsync();
            }

            return View(vm);
        }

        // Μετατροπή τιμής από μικτή σε καθαρή
        var priceIncludesVat = vm.Series == "ΑΛΠ" || vm.Series == "ΑΠΥ";

        if (priceIncludesVat)
        {
            foreach (var l in vm.Lines)
            {
                // Αν το VAT έχει δοθεί ως 24 αντί για 0.24 τότε διόρθωσε το
                var rate = l.VatRate > 1 ? l.VatRate / 100m : l.VatRate;

                if (rate > 0)
                {
                    var gross = l.UnitPrice;
                    var net = gross / (1 + rate);
                    l.UnitPrice = Math.Round(net, 2);
                }
            }
        }
```

```
        // Δημιουργία παραστατικού
        var inv = new Invoice
        {
            UserId = CurrentUserId,
            Series = vm.Series,
            Number = await GetNextNumberAsync(vm.Series, vm.IssueDate.Year), // αυτόματη αρίθμηση
            IssueDate = vm.IssueDate,
            Year = vm.IssueDate.Year,
            CustomerId = vm.CustomerId,     // Στα Purchases εδώ μπαίνει προμηθευτής
            PaymentMethod = vm.PaymentMethod,
            InvoiceType = vm.InvoiceType
        };

        // Γραμμές παραστατικού
        inv.Lines = vm.Lines.Select(x =>
        {
            var line = new InvoiceLine
            {
                Quantity = x.Quantity,
                UnitPrice = x.UnitPrice,
                // Αποθηκεύουμε το ίδιο ποσοστό που διάλεξε ο χρήστης (π.χ. 24)
                VatRate = x.VatRate
            };

            // Items & Purchases τότε ItemId
            // Services τότε ServiceId
            if (vm.InvoiceType == "Items" || vm.InvoiceType == "Purchases")
            {
                line.ItemId = x.CatalogId;
                line.ServiceId = null;
            }
            else
            {
                line.ServiceId = x.CatalogId;
                line.ItemId = null;
            }
            // Services τότε ServiceId
            if (vm.InvoiceType == "Items" || vm.InvoiceType == "Purchases")
            {
                line.ItemId = x.CatalogId;
                line.ServiceId = null;
            }
            else
            {
                line.ServiceId = x.CatalogId;
                line.ItemId = null;
            }

            return line;
        }).ToList();

        // Υπολογισμός συνόλων
        inv.RecalculateTotals();

        // Αποθήκευση
        _db.Invoices.Add(inv);

        try
        {
            await _db.SaveChangesAsync();
        }
        catch (DbUpdateException ex)
        {
            // Μήνυμα λάθους αποθήκευσης παραστατικού
            var msg = ex.InnerException?.Message ?? ex.Message;
            ModelState.AddModelError("", "Αποτυχία αποθήκευσης παραστατικού. " + msg);

            await FillDropdownsAsync(vm.InvoiceType);

            // Ξαναγέμισμα catalog (απαραίτητο αν σκάσει SQL error)
            var uid = CurrentUserId;

            if (vm.InvoiceType == "Items" || vm.InvoiceType == "Purchases")
            {
```

```
        // Ξαναγεμίζουμε catalog (απαραίτητο αν όχι θα έχουμε SQL error)
        var uid = CurrentUserId;

        if (vm.InvoiceType == "Items" || vm.InvoiceType == "Purchases")
        {
            vm.CatalogItems = await _db.Items
                .Where(i => i.UserId == uid)
                .OrderBy(i => i.Description)
                .Select(i => new InvoiceCreateVm.CatalogItemVm
                {
                    Type = "Item",
                    Id = i.Id,
                    Code = i.ItemCode,
                    Description = i.Description,
                    RetailPrice = i.RetailPrice,
                    WholesalePrice = i.WholesalePrice,
                    VatRate = i.VAT
                }).ToListAsync();
        }
        else
        {
            vm.CatalogItems = await _db.Services
                .Where(s => s.UserId == uid)
                .OrderBy(s => s.Description)
                .Select(s => new InvoiceCreateVm.CatalogItemVm
                {
                    Type = "Service",
                    Id = s.Id,
                    Code = s.ServiceCode,
                    Description = s.Description,
                    RetailPrice = s.RetailPrice,
                    WholesalePrice = s.WholesalePrice,
                    VatRate = s.VAT
                }).ToListAsync();
        }

        return View(vm);
    }
```

```
        // Επιτυχία
        TempData["Ok"] = $"Καταχωρήθηκε το παραστατικό {inv.Series}-{inv.Number}/{inv.Year}.";
        return RedirectToAction(nameof(Details), new { id = inv.Id });
    }
```

Το παραπάνω απόσπασμα δείχνει τη βασική ροή:

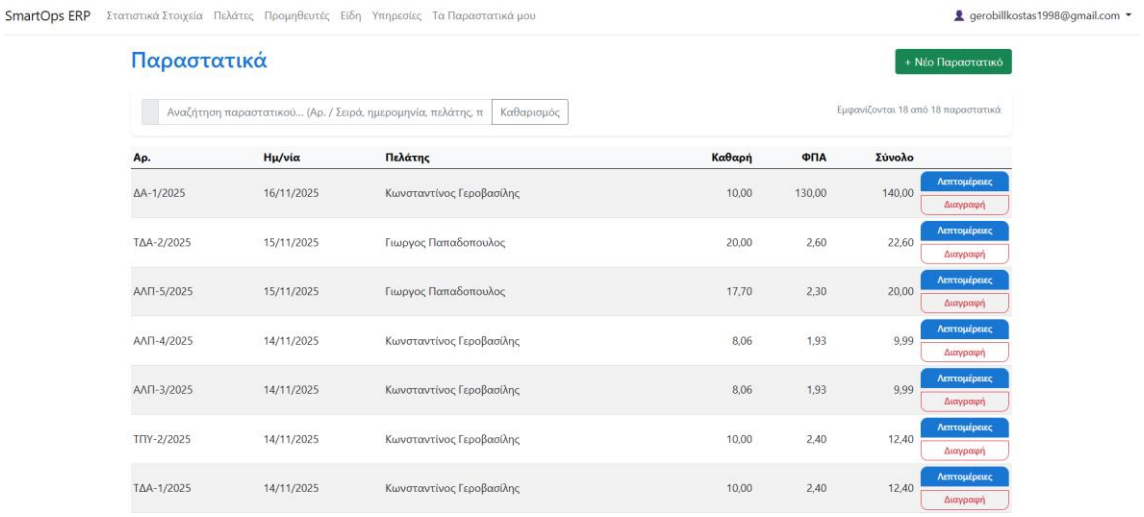
1. Έλεγχος εγκυρότητας δεδομένων.
2. Δημιουργία εγγραφής Invoice.
3. Προσθήκη γραμμών InvoiceLine.
4. Υπολογισμός συνολικού ποσού με ΦΠΑ.
5. Αποθήκευση και επιστροφή στο Index με μήνυμα επιτυχίας.

Ανάλυση ροής χρήστη

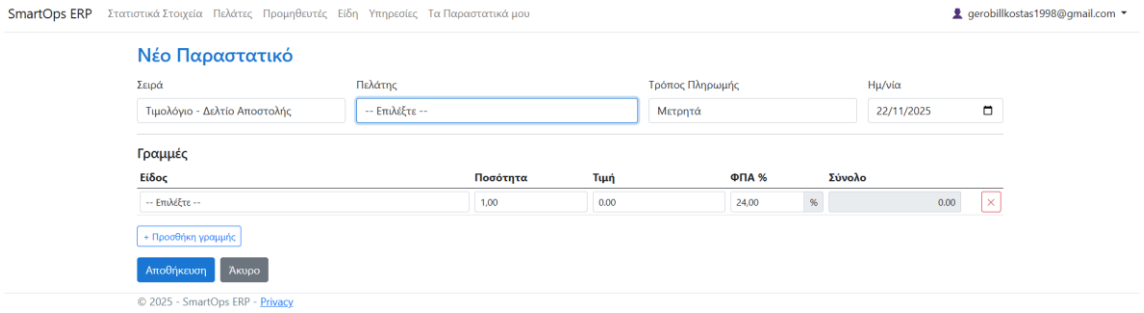
1. Ο χρήστης επιλέγει “Δημιουργία Τιμολογίου”.
2. Ο χρήστης επιλέγει Τύπο Παραστατικού
3. Εισάγει τα στοιχεία του πελάτη και τα προϊόντα ή τις υπηρεσίες.
4. Το σύστημα υπολογίζει αυτόματα το ΦΠΑ και τα σύνολα.
5. Ο χρήστης επιβεβαιώνει και αποθηκεύει το τιμολόγιο.
6. Εμφανίζεται μήνυμα επιτυχίας και το παραστατικό προστίθεται στη λίστα.

Η διεπαφή έχει σχεδιαστεί ώστε η διαδικασία να ολοκληρώνεται με τον ελάχιστο δυνατό αριθμό ενεργειών, προσφέροντας ταχύτητα και εργονομία.

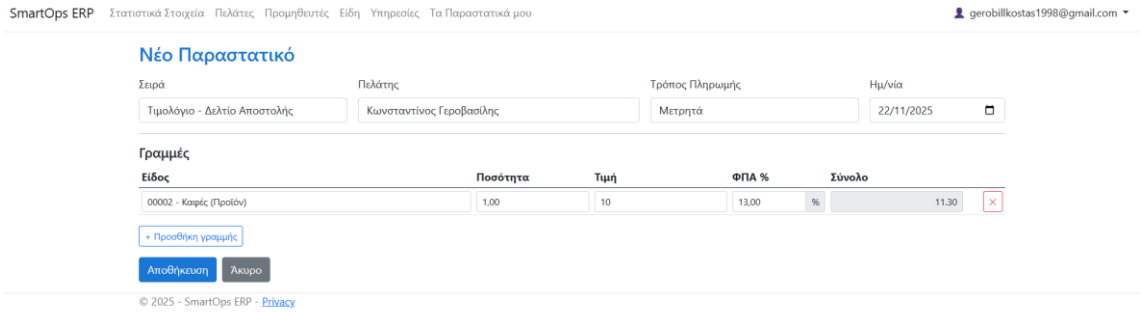
Εικόνες – Ενδεικτικές Οθόνες



Εικόνα 5.6.1: Λίστα τιμολογίων (Index) με πεδία αναζήτησης πελάτη και ημερομηνίας.

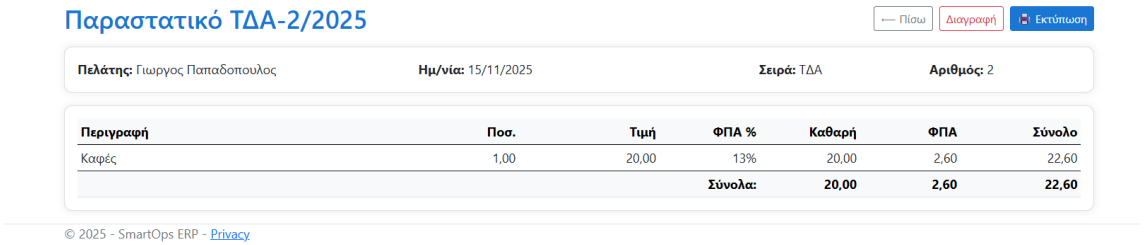


Εικόνα 5.6.2: Φόρμα δημιουργίας τιμολογίου με επιλογή πελάτη και είδους.



Εικόνα 5.6.3: Υπολογισμός συνόλων βάση του ΦΠΑ και της τιμής.

Στην εικόνα 5.6.3 έχουμε επιλέξει το προϊόν καφές με ποσότητα 1, τιμή 10€, ΦΠΑ 13% και στο σύνολο υπολογίζεται αυτόματα πόσο βγαίνει η συνολική αξία της γραμμής.



Εικόνα 5.6.4: Αναλυτική προβολή τιμολογίου (Details View).

Στην εικόνα 5.6.4 φαίνεται αναλυτικά οι λεπτομέρειες του παραστατικού όπως τιμή, καθαρή αξία, ΦΠΑ και το σύνολο. Επίσης υπάρχει η δυνατότητα εκτύπωσης ή διαγραφής του παραστατικού.

5.7 Λειτουργικότητα Dashboard

Το module **Dashboard** αποτελεί τη βασική οθόνη εποπτείας και ανάλυσης δεδομένων του SmartOps ERP Prototype. Παρέχει στον χρήστη μια συνοπτική εικόνα της επιχειρησιακής δραστηριότητας, εμφανίζοντας συγκεντρωτικά στατιστικά τόσο για τους πελάτες όσο και για τις πωλήσεις.

Σκοπός και ρόλος του module

Ο σκοπός του Dashboard είναι να προσφέρει άμεση πληροφόρηση και οπτική κατανόηση των δεδομένων, ώστε ο χρήστης να μπορεί να αξιολογεί τη δραστηριότητα της επιχείρησης χωρίς να χρειάζεται να μεταβαίνει σε διαφορετικές ενότητες.
Μέσα από γραφήματα και πίνακες, το Dashboard απεικονίζει κρίσιμες πληροφορίες όπως:

- τους κορυφαίους πελάτες σε πωλήσεις (Top-5 Customers),
- τις συνολικές πωλήσεις ανά μήνα,
- τα κορυφαία προϊόντα σε πωλήσεις (Top-5 προϊόντα)

Περιγραφή λειτουργικότητας

Η βασική οθόνη του Dashboard περιλαμβάνει τα εξής:

Γραφήματα και στατιστικές αναφορές

- **Top 5 πελάτες** με βάση τη συνολική αξία των τιμολογίων.
- **Μηνιαίες πωλήσεις** σε γραφική μορφή (bar chart).
- **Top 5 προϊόντα** με βάση τη συνολική ποσότητα από τα τιμολογία.

Η οθόνη έχει σχεδιαστεί ώστε να είναι responsive, επιτρέποντας ομαλή λειτουργία τόσο σε υπολογιστές όσο και για κινητές συσκευές μελλοντικά.

Τεχνική υλοποίηση

Η λογική του module υλοποιείται μέσω του **DashboardController**, ο οποίος ανακτά δεδομένα από τη βάση και τα προβάλλει σε δυναμικούς πίνακες και γραφήματα.

Παράδειγμα κώδικα (Top 5 πελάτες):

```
    [HttpGet]
    0 references
    public async Task<IActionResult> Index()
    {
        if (CurrentUserId == 0)
            return RedirectToAction("Login", "Account");

        var year = DateTime.Today.Year;

        // Top 5 πελάτες σε πωλήσεις
        var topCustomers = await _dash.GetTopCustomersAsync(CurrentUserId, 5);

        // Πωλήσεις ανά μήνα
        var salesByMonth = await _dash.GetSalesByMonthAsync(CurrentUserId, year);

        // Top 5 είδη σε ποσότητα
        var topItems = await _dash.GetTopItemsAsync(CurrentUserId, year, 5);

        // Δημιουργία του ViewModel που θα περάσει στο Dashboard view
        var vm = new DashboardVm
        {
            Year = year, // Το έτος που ζήτησε ο χρήστης για το dashboard
            TopCustomers = topCustomers, // Λίστα με τους καλύτερους πελάτες (βάσει πωλήσεων)
            SalesByMonth = salesByMonth, // Συγκεντρωτικές πωλήσεις ανά μήνα
            TopItems = topItems  // Προϊόντα με τις περισσότερες πωλήσεις
        };

        return View(vm);
    }
}
```

Στο παραπάνω απόσπασμα:

- Υπολογίζεται το συνολικό ποσό που έχει τιμολογηθεί σε κάθε πελάτη.
- Εμφανίζονται οι πέντε πελάτες με τη μεγαλύτερη αξία πωλήσεων.
- Υπολογίζεται η συνολική ποσότητα που έχει τιμολογηθεί για κάθε προϊόν.

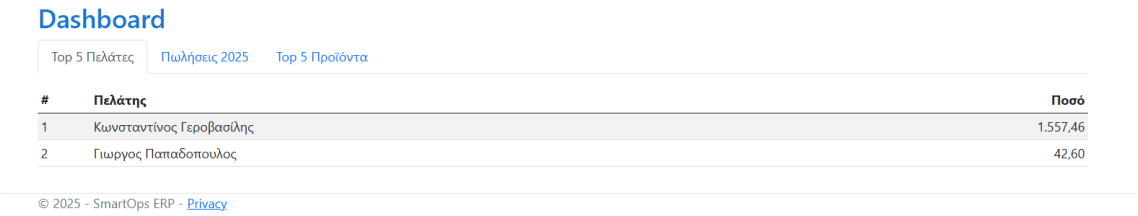
Η προβολή γίνεται μέσω δυναμικού πίνακα ή γραφήματος.

Ανάλυση ροής χρήστη

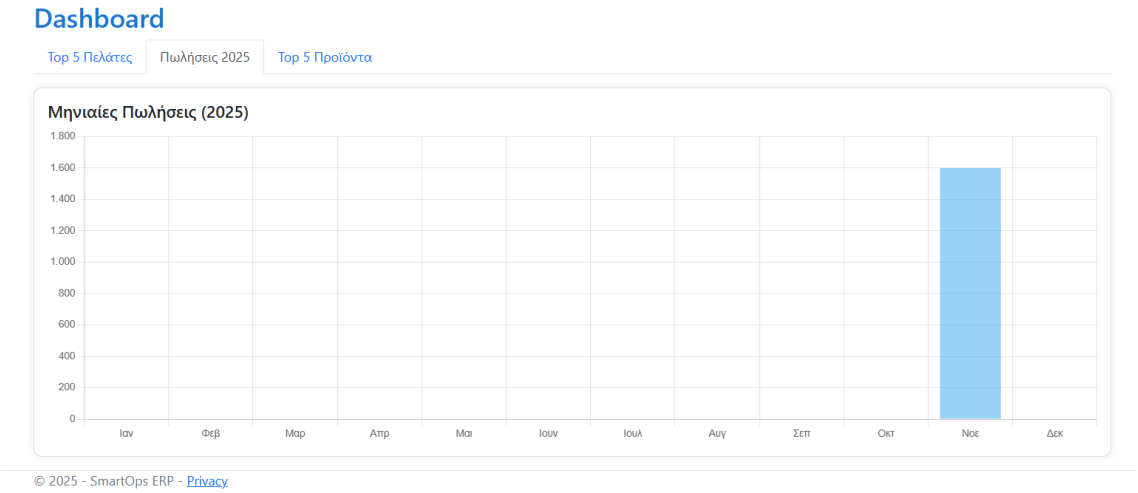
1. Με τη σύνδεση στο σύστημα, το Dashboard φαίνεται πάνω αριστερά και όχι απευθείας στην κεντρική οθόνη. ώστε να αποκρύβεται από τρίτους που μπορεί να κοιτάξουν στο πρόγραμμα αυτά τα στατιστικά.
2. Πατώντας πάνω στο Dashboard παρουσιάζονται τα γραφήματα (π.χ. Top-5 πελάτες, πωλήσεις ανά μήνα, Top-5 προϊόντα).
3. Ακόμη δίνεται η δυνατότητα στον χρήστη να επιλέξει να δει το Dashboard σε όποια ενότητα του προγράμματος και αν βρίσκεται (π.χ. Πελάτες, προϊόντα, τιμολόγια κτλ)

Η σχεδίαση του συγκεκριμένου πεδίου είναι προσανατολισμένη στην αμεσότητα και απλότητα, ώστε ακόμη και μη έμπειροι χρήστες να κατανοούν τα βασικά οικονομικά στοιχεία με μια ματιά.

Εικόνες – Ενδεικτικές Οθόνες



Εικόνα 5.7.1: Γράφημα "Top 5 Πελάτες" σε μορφή πίνακα.



Εικόνα 5.7.2: Μηνιαίες πωλήσεις σε bar chart.

Εικόνα 5.7.3: Γράφημα "Top 5 Προϊόντα" σε μορφή πίνακα.

5.8 Λειτουργικότητα Χρηστών και Λογαριασμών

Το module **Users / Account** είναι υπεύθυνο για τη διαχείριση των χρηστών και της ασφάλειας πρόσβασης στο SmartOps ERP Prototype. Περιλαμβάνει όλες τις απαραίτητες λειτουργίες για σύνδεση, εγγραφή, αλλαγή κωδικού πρόσβασης και βασικό έλεγχο εξουσιοδότησης.

Η σωστή διαχείριση των χρηστών διασφαλίζει την ακεραιότητα των δεδομένων και την ορθή λειτουργία του συστήματος.

Σκοπός και ρόλος του module

Ο κύριος στόχος του module Users / Account είναι η προστασία του συστήματος από μη εξουσιοδοτημένη πρόσβαση και η παροχή ενός εύχρηστου περιβάλλοντος για τη διαχείριση των λογαριασμών.

Το σύστημα υλοποιεί βασικές αρχές **authentication** (έλεγχος ταυτότητας) και **authorization** (δικαιώματα πρόσβασης), ενώ παράλληλα φροντίζει για την ασφάλεια των αποθηκευμένων κωδικών μέσω hashing.

Περιγραφή λειτουργικότητας

1. **Εγγραφή νέου χρήστη (Register)**
   - Ο χρήστης συμπληρώνει τα στοιχεία του (Email, Κωδικός, Επιβεβαίωση Κωδικού).
   - Ελέγχεται η εγκυρότητα του email και η αντιστοιχία των δύο πεδίων κωδικού.
   - Ο κωδικός αποθηκεύεται με ασφάλεια μέσω αλγορίθμου hashing.
2. **Σύνδεση (Login)**
   - Ο χρήστης εισάγει τα στοιχεία του (Email – Password).

- Αν τα στοιχεία είναι σωστά, αποκτά πρόσβαση στην εφαρμογή.
- Σε περίπτωση λάθους, εμφανίζεται μήνυμα “Λάθος email ή κωδικός”.

3. **Αλλαγή κωδικού (Change Password)**
   - Ο χρήστης μπορεί να αλλάξει τον κωδικό του μέσα από απλή φόρμα.
   - Το σύστημα ελέγχει τον νέο κωδικό και τον αποθηκεύει μετά από hashing.
4. **Αποσύνδεση (Logout)**
   - Πατώντας αποσύνδεση η συνεδρία τερματίζεται και ο χρήστης επιστρέφει στην οθόνη Login.

Τεχνική υλοποίηση

Η υλοποίηση βασίζεται στον AccountController και την κλάση UserService, οι οποίες διαχειρίζονται τις διαδικασίες αυθεντικοποίησης και ασφάλειας.

Παράδειγμα μεθόδου σύνδεσης και authentication:

```
// POST: Account/Login
[HttpPost]
[ValidateAntiForgeryToken]
0 references
public async Task<IActionResult> Login(LoginViewModel model)
{
    // Αν το μοντέλο έχει validation errors τότε επιστροφή στη φόρμα
    if (!ModelState.IsValid)
        return View(model);

    // Αναζήτηση χρήστη με βάση το email
    var user = await _userService.GetByEmailAsync(model.Email);

    // Έλεγχος αν υπάρχει χρήστης και αν ο κωδικός είναι σωστός
    if (user == null || !PasswordHelper.VerifyPassword(user.PasswordHash, model.Password))
    {
        ModelState.AddModelError(string.Empty, "Λάθος email ή κωδικός.");
        return View(model);
    }

    // SESSION
    // Πάντα πρώτα καθαρίζουμε για να μην μείνουν παλιές τιμές
    HttpContext.Session.Clear();

    // Αποθηκεύουμε το Id του χρήστη (για έλεγχο σε κάθε request)
    HttpContext.Session.SetInt32("UserId", user.Id);

    // Προτιμάμε το email που έδωσε ο χρήστης στη φόρμα
    HttpContext.Session.SetString("UserEmail", model.Email);

    // Αν υπάρχει Username το βάζουμε, αλλιώς fallback στο email
    HttpContext.Session.SetString("UserName",
        string.IsNullOrWhiteSpace(user.Username) ? model.Email : user.Username);

    // Αν χρησιμοποιούμε ημερομηνία εργασίας τότε την αποθηκεύουμε σε session
    HttpContext.Session.SetString("SelectedDate", model.Date.ToString("yyyy-MM-dd"));

    // Μήνυμα επιτυχίας
    TempData["SuccessMessage"] = "Επιτυχής σύνδεση.";

    // Μετά τη σύνδεση πήγαινε στην αρχική σελίδα του προγράμματος
    return RedirectToAction("Index", "Home");
}
```

Η μέθοδος ελέγχει:

- Αν ο χρήστης υπάρχει στη βάση.
- Αν ο κωδικός είναι σωστός μέσω hashing.
- Αν όλα είναι έγκυρα, δημιουργεί session και οδηγεί στο Dashboard.

Η μέθοδος authentication χρησιμοποιεί τη βοηθητική κλάση PasswordHelper για έλεγχο του hashed κωδικού, εξασφαλίζοντας ότι τα διαπιστευτήρια δεν αποθηκεύονται ποτέ σε απλή μορφή.

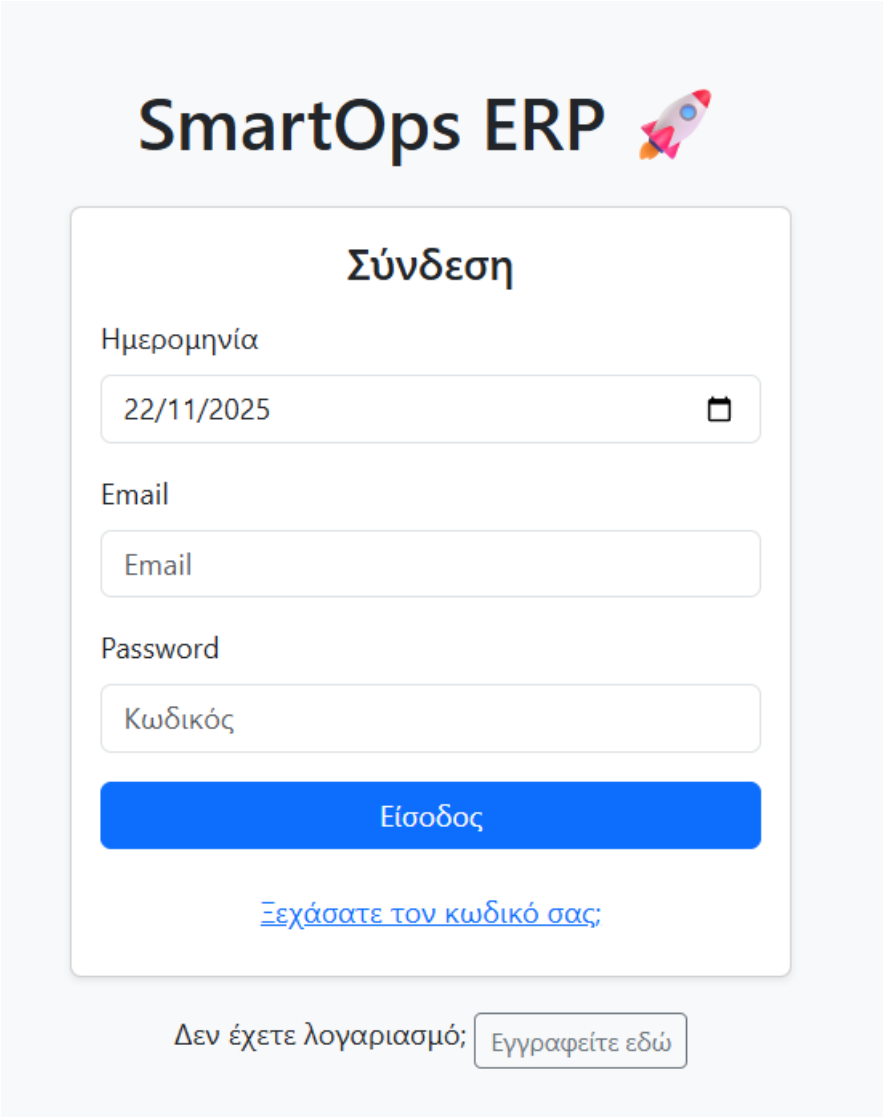
Ανάλυση ροής χρήστη

1. Ο χρήστης ανοίγει την εφαρμογή και μεταβαίνει στη φόρμα σύνδεσης.
2. Εισάγει email και κωδικό πρόσβασης.

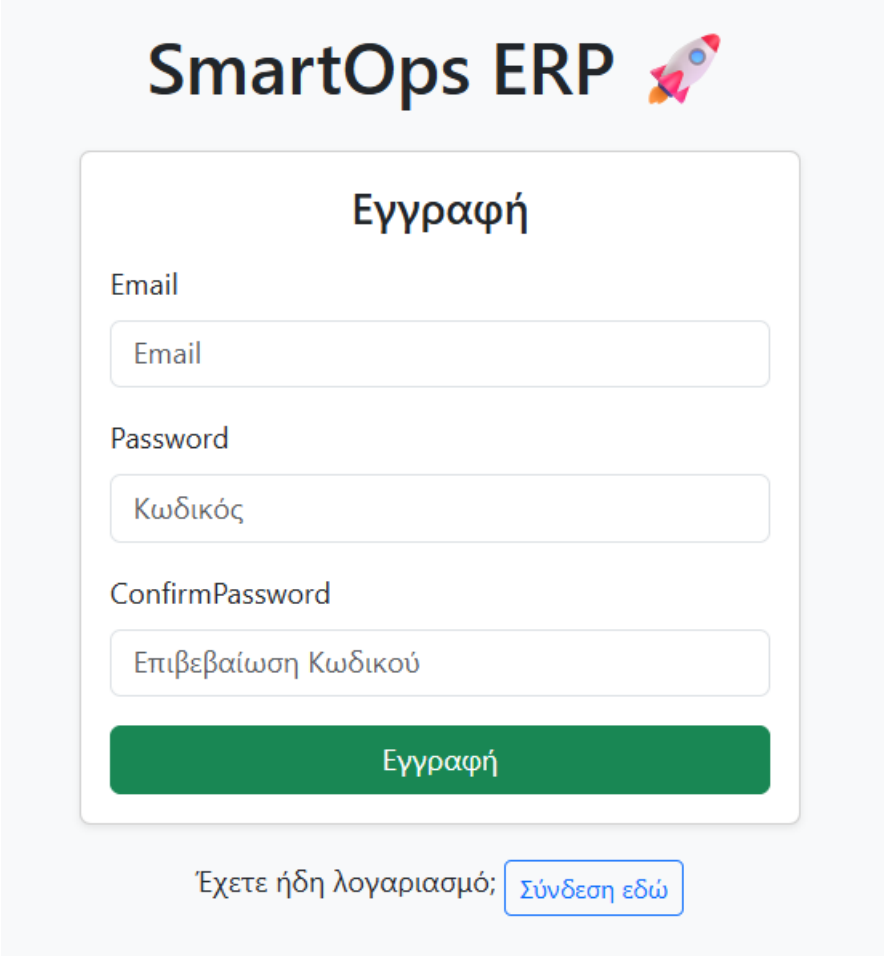
3. Το σύστημα ελέγχει τα στοιχεία και πραγματοποιεί authentication.
4. Αν είναι έγκυρα, γίνεται ανακατεύθυνση στο Dashboard.
5. Ο χρήστης μπορεί να αλλάξει τον κωδικό του ανά πάσα στιγμή μέσω της φόρμας Change Password.
6. Ο χρήστης μπορεί να κάνει εγγραφή με νέα στοιχεία και μετά να ανακατευθυνθεί στην φόρμα Login για να συνδεθεί στο πρόγραμμα.

Η διαδικασία έχει σχεδιαστεί ώστε να είναι απλή, γρήγορη και ασφαλής, προσφέροντας θετική εμπειρία χρήσης και ταυτόχρονα προστασία δεδομένων.

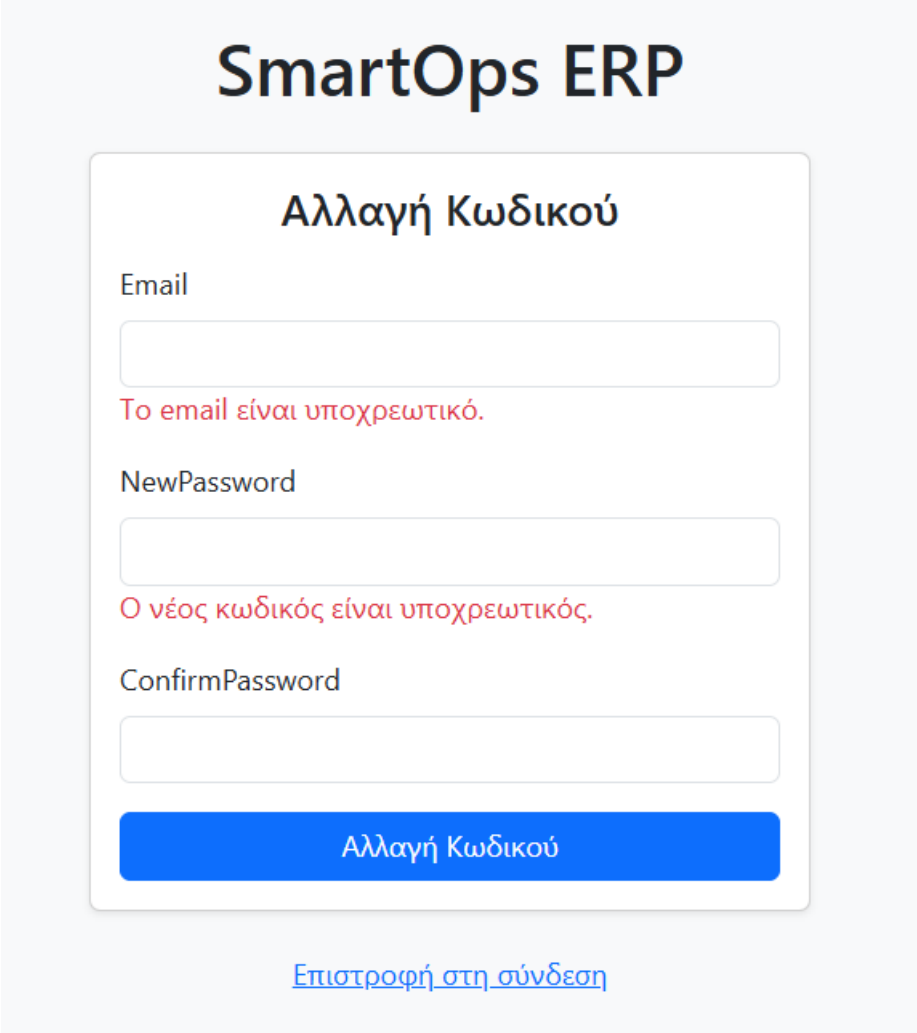
Εικόνες – Ενδεικτικές Οθόνες



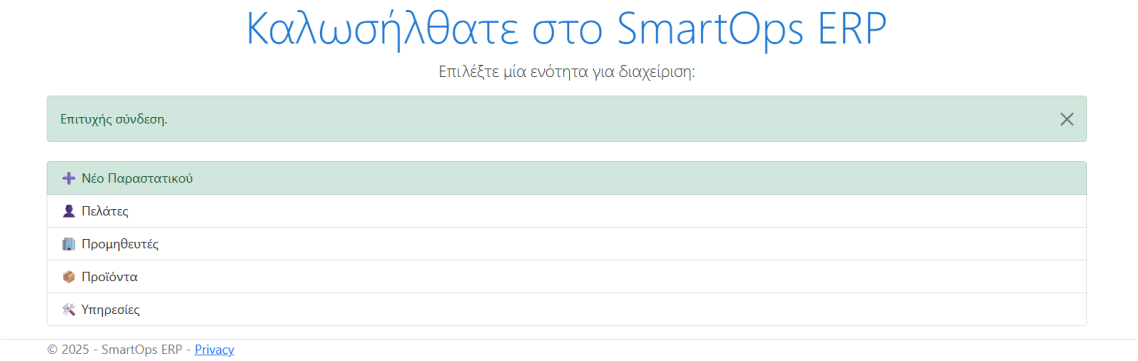
Εικόνα 5.8.1: Φόρμα σύνδεσης (Login) με έλεγχο εγκυρότητας πεδίων.



Εικόνα 5.8.2: Φόρμα εγγραφής νέου χρήστη (Register).



Εικόνα 5.8.3: Φόρμα αλλαγής κωδικού πρόσβασης (Change Password).



Εικόνα 5.8.4: Μήνυμα επιτυχούς σύνδεσης μέσω TempData.

5.9 Validation Rules & Business Logic

Η σωστή επικύρωση των δεδομένων (validation) και η ορθή εφαρμογή των επιχειρησιακών κανόνων (business logic) αποτελούν κρίσιμα στοιχεία για τη σταθερή, αξιόπιστη και ασφαλή λειτουργία του SmartOps ERP Prototype.

Στην ενότητα αυτή παρουσιάζονται οι βασικές αρχές και οι κανόνες που υλοποιήθηκαν στο σύστημα, καθώς και ο τρόπος με τον οποίο εξασφαλίζουν την ακεραιότητα των δεδομένων και τη συνεπή ροή εργασιών.

1. Γενική Φιλοσοφία Validation

Το SmartOps εφαρμόζει validation τόσο:

- **σε επίπεδο Model** (Data Annotations),
- **σε επίπεδο Controller** (ModelState checks),
- **όσο και σε Business Services**, όπου απαιτείται πιο σύνθετος έλεγχος (π.χ. αρίθμηση τιμολογίων, υπολογισμοί αξιών).

Με αυτόν τον τρόπο εξασφαλίζεται ότι:

- δεν αποθηκεύονται λανθασμένα ή ελλιπή δεδομένα,
- ο χρήστης λαμβάνει άμεση ανατροφοδότηση,
- η βάση παραμένει συνεπής και καθαρή,
- αποφεύγονται σφάλματα κατά τη δημιουργία τιμολογίων ή πελατών.

2. Validation σε Customers & Suppliers

Οι βασικοί έλεγχοι που εφαρμόζονται κατά την καταχώριση πελατών και προμηθευτών περιλαμβάνουν:

Υποχρεωτικά πεδία

- Customer/SuplierCode
- Name

Έλεγχος ΑΦΜ

Το ΑΦΜ ελέγχεται για:

- μήκος 9 ψηφίων,
- έγκυρο αριθμητικό περιεχόμενο,
- αποφυγή λευκών χαρακτήρων ή μη-αριθμητικών τιμών.

Έλεγχος μοναδικότητας (unique)

Customer Code / Supplier Code ελέγχονται ώστε να μην επαναλαμβάνονται.

Αυτό αποτρέπει διπλές εγγραφές που θα μπορούσαν να δημιουργήσουν ασυμφωνίες στα τιμολόγια.

3. Validation σε Items & Services

Οι βασικοί κανόνες περιλαμβάνουν:

Τιμές και ΦΠΑ

- UnitPrice ≥ 0

- VatRate ∈ {0%, 6%, 13%, 24%}
- Quantity ≥ 0 (σε περιπτώσεις αποθέματος)

Κωδικοί

- ItemCode / ServiceCode πρέπει να είναι μοναδικά
- Δεν επιτρέπεται κενός ή μη έγκυρος κωδικός

Υποχρεωτικά πεδία

- ItemCode / ServiceCode
- Name
- UnitPrice
- Vat

Το σύστημα εμφανίζει άμεσα μηνύματα σφάλματος (validation messages) κάτω από κάθε πεδίο, οδηγώντας τον χρήστη στην εύρεση και διόρθωση λαθών.

4. Validation & Business Logic στα Invoices

Το module των Παραστατικών είναι το πιο σύνθετο, καθώς απαιτεί πολλαπλά επίπεδα ελέγχων.

4.1 Έλεγχος ότι υπάρχει τουλάχιστον μία γραμμή (InvoiceLine)

Απαγορεύεται η δημιουργία τιμολογίου χωρίς προϊόντα ή υπηρεσίες.
Ο Controller επιστρέφει μήνυμα:

«Προσθέστε τουλάχιστον μία γραμμή.»

4.2 Έλεγχος ποσότητας και τιμής

Κάθε γραμμή του τιμολογίου πρέπει να πληροί:

- Quantity > 0
- UnitPrice ≥ 0
- Valid VatRate

4.3 Επιχειρησιακός κανόνας: Αρίθμηση παραστατικών

Κατά την αποθήκευση, το InvoiceService:

1. αναζητά το τελευταίο παραστατικό της ίδιας σειράς και του ίδιου έτους,
2. παράγει τον επόμενο αριθμό (Number + 1),
3. διασφαλίζει μοναδικότητα μέσω συναλλαγής (transaction).

Αν δύο χρήστες δημιουργήσουν παραστατικό ταυτόχρονα, το σύστημα αποτρέπει τη σύγκρουση αρίθμησης.

4.4 Υπολογισμοί (Business Logic)

Για κάθε γραμμή υπολογίζονται αυτόματα:

- LineNet
- LineVat

- LineGross

Στη συνέχεια το τιμολόγιο υπολογίζει:

- TotalNet
- TotalVat
- TotalGross

Οι υπολογισμοί γίνονται αποκλειστικά στη πλευρά του Service, ώστε να διατηρείται σταθερότητα και ακρίβεια.

4.5 Έλεγχος Πελάτη

Πριν την αποθήκευση:

- Ελέγχεται ότι ο CustomerId είναι υπαρκτός,
- Ελέγχεται ότι ο πελάτης δεν είναι ανενεργός (εφόσον εφαρμοστεί στο μέλλον).

5. Validation κατά το Authentication

Στο module Users/Account εφαρμόζονται:

- έλεγχος μήκους κωδικού,
- σύγκριση “Password” και “Confirm Password”,
- έλεγχος ύπαρξης email στη βάση,
- ασφαλές hashing/verification χωρίς plain text κωδικούς.

Τα σφάλματα εμφανίζονται με έντονο κόκκινο, ώστε ο χρήστης να γνωρίζει ακριβώς τι πρέπει να διορθώσει.

6. Ανατροφοδότηση προς τον χρήστη (Feedback)

Το SmartOps χρησιμοποιεί:

- ModelState errors υπό τα πεδία,
- validation summary στην κορυφή της φόρμας,
- TempData["Success"] / TempData["Error"] για καθαρά μηνύματα.

Η λογική είναι:

καθαρή καθοδήγηση → λιγότερα λάθη → πιο ομαλή χρήση.

7. Πλεονεκτήματα της προσέγγισης αυτής

Η εφαρμογή των validation rules και της business logic σε πολλαπλά επίπεδα προσφέρει:

- **Ακεραιότητα δεδομένων** (no invalid records)
- **Αξιοπιστία** (σωστή αρίθμηση, σωστοί υπολογισμοί)
- **Καλύτερη εμπειρία χρήστη** (ξεκάθαρα μηνύματα σφάλματος)
- **Ευκολία συντήρησης** (καθαρός κώδικας, κοινή λογική σε όλα τα modules)

- **Επέκταση στο μέλλον** (MyData, Warehouses, Payments)

5.10 Ενοποίηση λειτουργιών

Η ενότητα αυτή παρουσιάζει τον τρόπο με τον οποίο τα επιμέρους modules του SmartOps ERP Prototype συνεργάζονται μεταξύ τους, σχηματίζοντας ένα ενιαίο, πλήρως λειτουργικό πληροφοριακό σύστημα.

Η αρχιτεκτονική της εφαρμογής βασίζεται στην ολοκληρωμένη ροή δεδομένων από τη μία ενότητα στην άλλη, χωρίς την ανάγκη επανακαταχώρισης ή χειροκίνητης ενημέρωσης.

Σκοπός της ενοποίησης

Ο βασικός στόχος είναι να επιτευχθεί αδιάκοπη επικοινωνία μεταξύ των modules, έτσι ώστε:

- Τα δεδομένα να εισάγονται μία φορά και να είναι διαθέσιμα παντού.
- Οι ενέργειες του χρήστη σε ένα σημείο (π.χ. δημιουργία πελάτη) να αντικατοπτρίζονται άμεσα στα υπόλοιπα modules (π.χ. Invoices, Dashboard).
- Η συνολική εμπειρία να παραμένει ομοιόμορφη, γρήγορη και ασφαλής.

Ροή πληροφορίας μεταξύ modules

1. **Customers → Invoices**
   Οι πελάτες που δημιουργούνται στο module Customers είναι άμεσα διαθέσιμοι κατά τη δημιουργία τιμολογίου. Ο χρήστης μπορεί να επιλέξει έναν πελάτη από λίστα και να προχωρήσει σε έκδοση παραστατικού χωρίς επιπλέον καταχωρίσεις.
2. **Items / Services → Invoices**
   Κατά τη σύνταξη ενός τιμολογίου, τα προϊόντα και οι υπηρεσίες που έχουν ήδη καταχωρηθεί στα αντίστοιχα modules εμφανίζονται σε dropdown menus. Οι τιμές λιανικής ή χονδρικής, οι συντελεστές ΦΠΑ και οι περιγραφές φορτώνονται αυτόματα.
3. **Invoices → Dashboard**
   Κάθε φορά που δημιουργείται ή ενημερώνεται ένα τιμολόγιο, το Dashboard ανανεώνεται δυναμικά, υπολογίζοντας εκ νέου τα σύνολα πωλήσεων, τους κορυφαίους πελάτες και τους μηνιαίους δείκτες απόδοσης.
4. **Users → Όλα τα modules**
   Οι χρήστες συνδέονται μέσω του module Account και αποκτούν πρόσβαση σε όλες τις λειτουργίες σύμφωνα με τα δικαιώματά τους. Κάθε ενέργεια (όπως η δημιουργία τιμολογίου ή η εισαγωγή πελάτη) μπορεί να συσχετιστεί με τον λογαριασμό του κάθε ενεργού χρήστη ξεχωριστά.

Ενδεικτική ροή εργασίας (Workflow)

1. Ο χρήστης εισάγει έναν νέο πελάτη στο module Customers
   Στη συνέχεια, μεταβαίνει στο module Items και προσθέτει προϊόντα.
2. Έπειτα, δημιουργεί νέο τιμολόγιο στο module Invoices, επιλέγοντας τον πελάτη και τα προϊόντα που εισήχθησαν προηγουμένως.
3. Το σύστημα υπολογίζει αυτόματα τα ποσά και ενημερώνει το Dashboard.
4. Ο χρήστης μπορεί να δει άμεσα τα αποτελέσματα των ενεργειών του μέσα από τα γραφήματα και τους πίνακες του Dashboard.

Με αυτόν τον τρόπο, διασφαλίζεται πλήρης συνέργεια μεταξύ όλων των τμημάτων του συστήματος και επιτυγχάνεται αυτοματοποιημένη ροή πληροφορίας χωρίς περιττές επαναλήψεις.

Τεχνική περιγραφή της ενοποίησης

Η ενοποίηση επιτυγχάνεται μέσω κοινών αναφορών (foreign keys) και σχέσεων **1–N** στη βάση δεδομένων:

- Ένας Customer μπορεί να έχει πολλά Invoices.
- Κάθε Invoice περιέχει πολλές εγγραφές InvoiceLines.
- Κάθε InvoiceLine συνδέεται με ένα Item ή Service.
- Οι ενέργειες των χρηστών καταγράφονται μέσω του UserId σε κάθε παραστατικό.

Η επικοινωνία μεταξύ των modules διευκολύνεται από την κοινή χρήση του SmartOpsDbContext, ο οποίος λειτουργεί ως ενιαίος μηχανισμός πρόσβασης στη βάση δεδομένων.

Η ενοποίηση των λειτουργιών είναι το στοιχείο που καθιστά το SmartOps ERP ενιαίο πληροφοριακό σύστημα και όχι απλώς μια συλλογή επιμέρους εφαρμογών. Μέσα από τη ροή δεδομένων και την αλληλεξάρτηση των modules, το SmartOps επιτυγχάνει αξιοπιστία, συνέπεια και παραγωγικότητα σε κάθε επίπεδο λειτουργίας.

5.11 Privacy Policy Page

Η εφαρμογή SmartOps περιλαμβάνει ένα κουμπί **Privacy Policy**, μέσω του οποίου παρουσιάζονται συνοπτικά οι αρχές συλλογής, χρήσης και προστασίας δεδομένων που εισάγει ο χρήστης στο σύστημα. Η σελίδα παρέχει αναλυτικές πληροφορίες σχετικά με τα αποθηκευμένα στοιχεία (π.χ. πελάτες, προμηθευτές, παραστατικά, χρήστες), τους μηχανισμούς ασφάλειας που χρησιμοποιούνται (session, κρυπτογράφηση κωδικών, έλεγχος πρόσβασης), καθώς και τα δικαιώματα του χρήστη ως προς τη διαχείριση των δεδομένων του.

Η πολιτική αυτή λειτουργεί ενημερωτικά και αντανακλά τη φιλοσοφία του SmartOps ERP Prototype για ασφαλή, περιορισμένη και διαφανή επεξεργασία δεδομένων. Δεν πραγματοποιείται διαμοιρασμός πληροφοριών σε τρίτους ούτε χρήση cookies πέραν των απαραίτητων για τη σύνδεση του χρήστη.

Η σελίδα Privacy Policy αποτελεί μέρος της βασικής πλοήγησης του συστήματος και χρησιμοποιείται αποκλειστικά για ενημερωτικούς σκοπούς για τον χρήστη.

6.Αντιμετώπιση Προβλημάτων

6.1 Συνδέσεις με την βάση

Πρόβλημα Σύνδεσης με τη Βάση Δεδομένων

Περιγραφή: Κατά την αρχική σύνδεση με τον SQL Server μέσω του DbContext, εμφανίστηκε το σφάλμα:

```
A connection was successfully established with the server, but then an error occurred during the login process.

(provider: SSL Provider, error: 0 - Η αλυσίδα πιστοποίησης δεν ήταν έγκυρη)
```

Αιτία: Η σύνδεση προσπαθούσε να χρησιμοποιήσει SSL πιστοποιητικό, το οποίο δεν ήταν έγκυρο ή δεν είχε ρυθμιστεί σωστά στο τοπικό περιβάλλον.

Λύση: Ενημερώθηκε το connection string ώστε να περιλαμβάνει TrustServerCertificate=True;, επιτρέποντας την παράκαμψη της επικύρωσης του πιστοποιητικού:

```
"DefaultConnection":"Server=localhost\\SQLEXPRESS;Database=SmartOpsDb_Final;Trusted_Connection=True;TrustServerCertificate=True;"
```

6.2 Σφάλματα Namespace και Αναγνωρισιμότητας Τύπων

Περιγραφή: Κατά την αναφορά σε models ή services, εμφανιζόταν σφάλμα τύπου:

```
The type or namespace name 'Data' does not exist in the namespace 'SmartOpsProject'
```

Αιτία: Έλλειψη using δήλωσης ή λάθος τοποθέτηση των αρχείων σε φακέλους χωρίς σωστό namespace.

Λύση: Ελέγχθηκαν τα namespaces των αρχείων και προστέθηκε η κατάλληλη δήλωση:

```
using SmartOpsProject.Data;
```

Επίσης, διορθώθηκε η δομή των φακέλων ώστε να τηρεί τη λογική της εφαρμογής MVC.

6.3 Μη εύρεση View

Περιγραφή: Κατά την εκτέλεση του Create action για παραστατικά, εμφανίστηκε σφάλμα:

```
The view 'Create' was not found. The following locations were searched:
...
```

Αιτία: Το αρχείο Create.cshtml είτε δεν υπήρχε στον σωστό φάκελο (/Views/Invoices/) είτε είχε λανθασμένο όνομα.

Λύση: Δημιουργήθηκε σωστά το view στη διαδρομή:

```
/Views/Invoices/Create.cshtml
```

και διασφαλίστηκε ότι το action Create() επιστρέφει return View(); χωρίς συγκεκριμένο όνομα view.

6.4 Μη απόδοση τιμών σε SelectList

Περιγραφή: Οι επιλογές InvoiceSeries και PaymentMethod δεν εμφανίζονταν σωστά στο DropDownList.

Αιτία: Δεν είχε ανατεθεί τιμή στο ViewBag.SeriesOptions από τον controller.

Λύση: Στον controller InvoicesController, προστέθηκε κώδικας ώστε να δοθούν δυναμικά οι επιλογές από τη βάση:

```
ViewBag.SeriesOptions = new SelectList(_context.InvoiceSeries, "Id", "Name");
```

6.5 Σφάλματα σε Migrations και Reset Βάσης

Περιγραφή: Κατά την αναδιάρθρωση των entities και του domain model, προέκυψαν προβλήματα κατά την εκτέλεση των migrations, ειδικά όταν είχε προηγηθεί χειροκίνητη δημιουργία της βάσης ή αλλαγές στα ονόματα πινάκων/οντοτήτων.

Αιτία: Τα metadata των προηγούμενων migrations δεν αντιστοιχούσαν στη νέα κατάσταση της βάσης, με αποτέλεσμα η εντολή Update-Database να αποτυγχάνει ή να μην αντανακλά τις αλλαγές.

Λύση: Εφαρμόστηκε πλήρες reset με διαγραφή όλων των προηγούμενων migrations και δημιουργία νέας βάσης με τελικό όνομα SmartOpsDb:

```
PM> Remove-Migration
PM> Drop-Database
PM> Add-Migration InitialCreate
PM> Update-Database
```

Αυτό επέτρεψε την εκκίνηση από καθαρό σημείο με σωστή αρχιτεκτονική.

6.6 Μετονομασία Οντοτήτων και Αντιστοίχιση με UI

Περιγραφή: Σε αρκετές περιπτώσεις προέκυψε ανάγκη μετονομασίας πινάκων ή ιδιοτήτων (π.χ. Clients → Customers, Products → Items) ώστε να είναι πιο κατανοητά και γενικά.

Αιτία: Η αρχική ονοματολογία δεν αντανακλούσε επαρκώς τη λογική των δεδομένων της εφαρμογής, ή υπήρχε αλληλοεπικάλυψη λειτουργιών (π.χ. προϊόντα και υπηρεσίες στον ίδιο πίνακα).

Λύση: Πραγματοποιήθηκε μετονομασία στα models, στα views, στους controllers και στο database schema, και διασφαλίστηκε η συνέπεια σε όλο το project. Επίσης, αποφασίστηκε η διατήρηση ενός ενιαίου πίνακα για Items και Services, με χρήση ιδιότητας τύπου (Type) για διαχωρισμό.

6.7 Πλοήγηση και UI Τοποθέτηση Κουμπιών

Περιγραφή:
Κατά τη διαγραφή ενός προϊόντος (Item) από τον κατάλογο, εμφανίστηκε σφάλμα τύπου:

```
The DELETE statement conflicted with the REFERENCE constraint
'FK_InvoiceLines_Items_ItemId'. The conflict occurred in database "SmartOpsDb",
table "dbo.InvoiceLines", column 'ItemId'.
```

Αιτία:
Το συγκεκριμένο είδος χρησιμοποιείται ήδη σε γραμμές παραστατικών (InvoiceLines).
Υπάρχει ενεργό Foreign Key από τον πίνακα InvoiceLines προς τον πίνακα Items χωρίς ON DELETE CASCADE.
Άρα η βάση δεν επιτρέπει τη διαγραφή του προϊόντος όσο υπάρχουν συνδεδεμένα παραστατικά.

Λύση:
Υλοποιήθηκε έλεγχος στο ItemsController με try / catch πάνω στο DbUpdateException.
Όταν ο SQL Server επιστρέφει σφάλμα SqlException με κωδικό 547 (παράβαση Foreign Key), η εφαρμογή δεν “σκάει”, αλλά εμφανίζει φιλικό μήνυμα στον χρήστη ότι το προϊόν δεν μπορεί να διαγραφεί γιατί συμμετέχει σε παραστατικά.

6.8 Δεν βρέθηκε το URL /Customers – Error 404

- **Αιτία:** Ο controller είχε όνομα CustomerController αντί για CustomersController, ενώ το URL προσπαθούσε να προσπελάσει /Customers.
- **Λύση:** Μετονομασία του controller σε CustomersController ή χρήση asp-controller="Customer" στο view.

6.9 SqlException: Invalid object name 'Suppliers'

- **Αιτία:** Δεν είχε εκτελεστεί Update-Database μετά τη δημιουργία του μοντέλου Supplier.
- **Λύση:** Τρέξιμο των εντολών:

```
<Add-Migration AddSuppliersInvoiceItemsUsers
Update-Database
```

6.10 Συντακτικό σφάλμα σε foreach στο Index.cshtml των Προϊόντων

- **Αιτία:** Το </tr> έκλεινε έξω από την foreach, προκαλώντας λάθος HTML.
- **Λύση:** Τοποθέτηση του </tr> μέσα στο μπλοκ του foreach.

7.Σενάρια χρήσης/Screenshots

Η παρούσα ενότητα παρουσιάζει ένα **ενδεικτικό σενάριο χρήσης** του συστήματος SmartOps ERP Prototype, το οποίο περιγράφει τη ροή των ενεργειών που πραγματοποιεί ένας χρήστης κατά την καθημερινή λειτουργία του συστήματος.

Το σενάριο αποτυπώνει τη φυσική αλληλουχία βημάτων, από τη σύνδεση στο σύστημα, έως τη δημιουργία και προβολή ενός τιμολογίου συνοδευόμενη από πραγματικά screenshots της εφαρμογής.

7.1 Περιγραφή Σεναρίου Χρήσης

Το σενάριο βασίζεται σε έναν τυπικό χρήστη μιας εμπορικής επιχείρησης, ο οποίος χρησιμοποιεί το SmartOps για να εκτελέσει μια πλήρη ροή τιμολόγησης:

Σκοπός σεναρίου:

Να καταγραφεί η πλήρης διαδικασία δημιουργίας και έκδοσης ενός τιμολογίου, ξεκινώντας από την εισαγωγή νέου πελάτη και προϊόντος, έως την προβολή του παραστατικού.

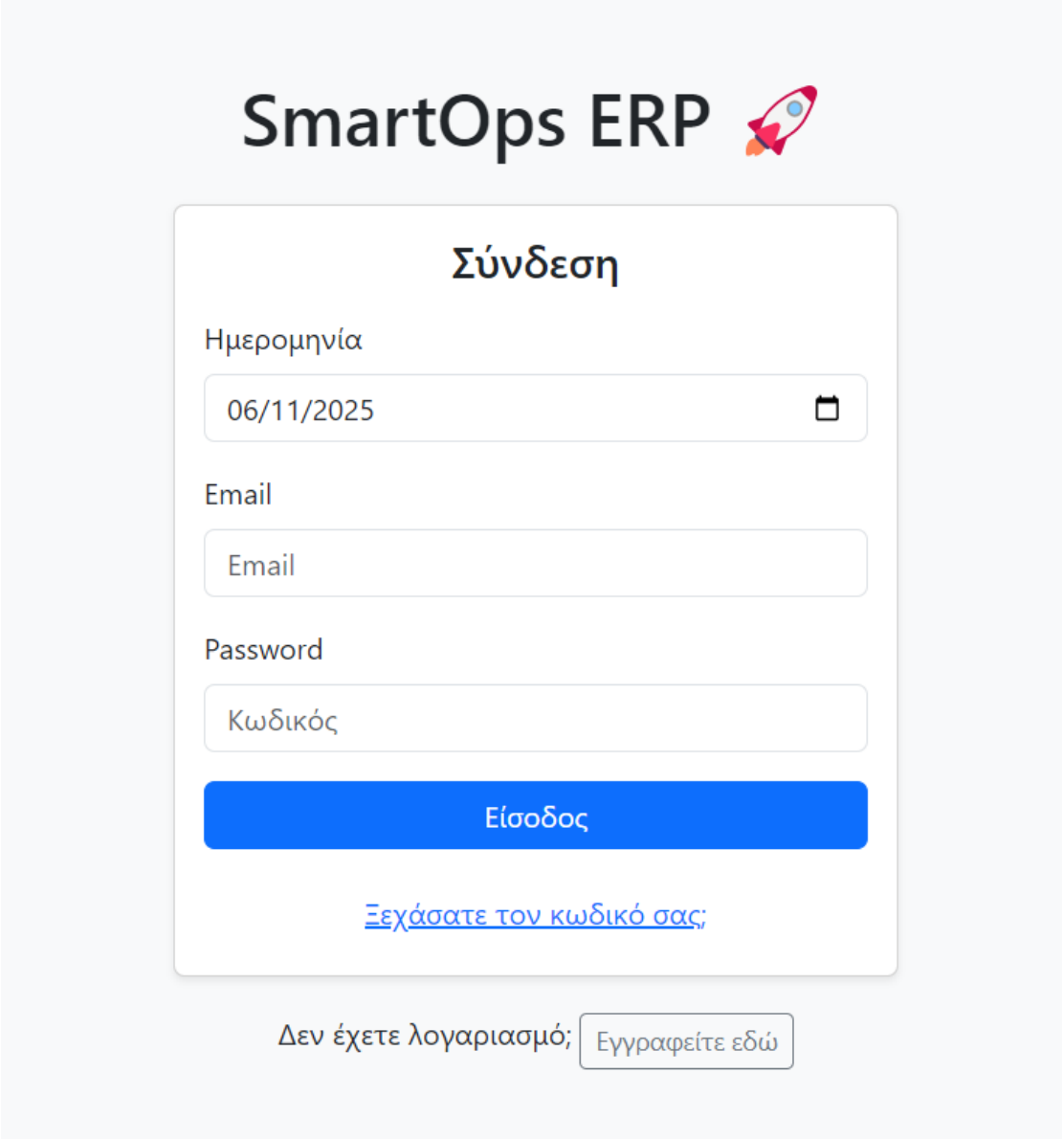
Εμπλεκόμενα modules:

- Users / Account
- Customers
- Items
- Invoices
- Dashboard

7.2 Ροή Ενεργειών Χρήστη

Βήμα 1: Είσοδος στο σύστημα (Login)

Ο χρήστης ανοίγει την εφαρμογή SmartOps και μεταβαίνει στη φόρμα σύνδεσης.
Συμπληρώνει τα στοιχεία του (Email και Κωδικός πρόσβασης) και πατά το κουμπί **"Είσοδος"**.



Εικόνα 7.2.1 Φόρμα σύνδεσης (Login screen) με έλεγχο εγκυρότητας πεδίων.

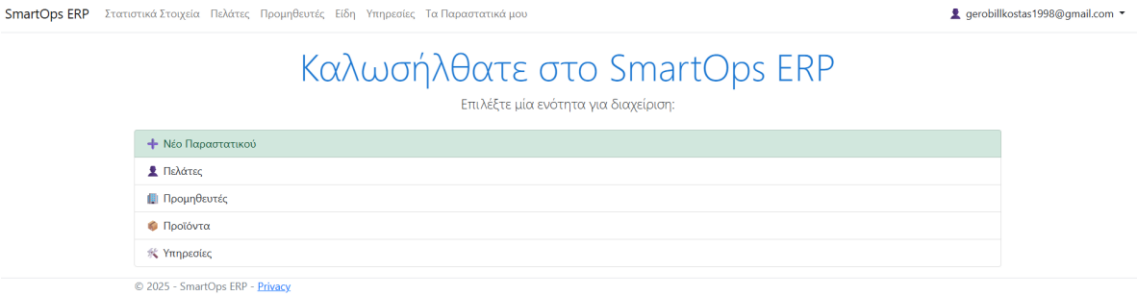
Με επιτυχή σύνδεση, ο χρήστης μεταφέρεται αυτόματα στην κεντρική σελίδα του προγράμματος.

Βήμα 2: Προβολή Κεντρικής Σελίδας του SmartOps

Στην αρχική οθόνη εμφανίζεται ένα μήνυμα το οποίο γράφει “Καλωσήλθατε στο SmartOps ERP” και ακριβώς από κάτω εμφανίζονται οι ενότητες πελάτες, προμηθευτές, προϊόντα, υπηρεσίες καθώς και η επιλογή νέο παραστατικό με πράσινο χρώμα για να δείχνει ποιο έντονη από τις άλλες ενότητες.

Ακόμη πάνω δεξιά εμφανίζεται το username του χρήστη που συνδέθηκε στην εφαρμογή καθώς και επιλογή να αποσυνδεθεί αν επιθυμεί.

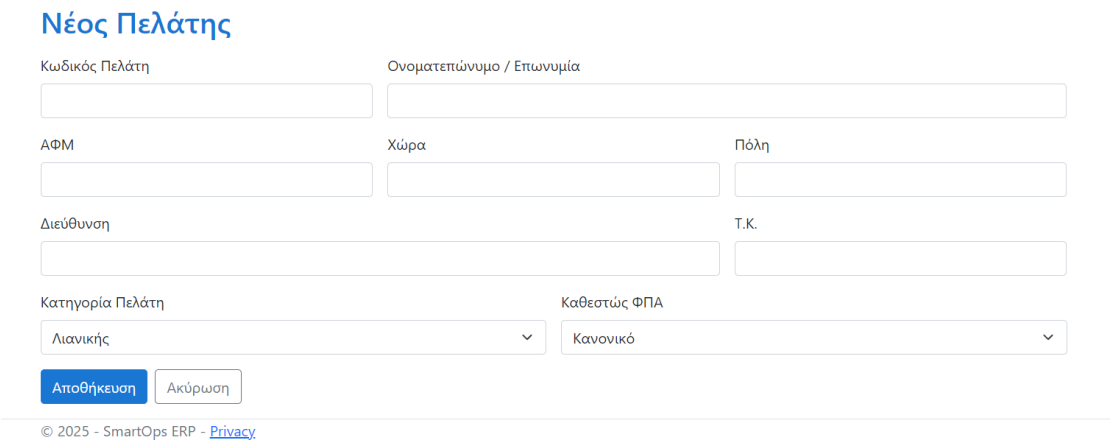
Επίσης πάνω αριστερά και προς το κέντρο φαίνονται κάποια head button που είτε εμφανίζεται ξανά εκεί όπως και στο κέντρο της οθόνης είτε είναι καινούργια όπως τα στατιστικά στοιχεία της επιχείρησης καθώς και τα παραστατικά της.



Εικόνα 7.2.2 Κεντρική Σελίδα του προγράμματος με εμφανή τα σημαντικά πεδία.

Βήμα 3: Δημιουργία νέου πελάτη

Από το κεντρικό μενού ο χρήστης επιλέγει **Πελάτες** και στην συνέχεια Νέος Πελάτης. Συμπληρώνει τη φόρμα με τα στοιχεία του πελάτη (Όνομα, ΑΦΜ, Χώρα, Πόλη, Διεύθυνση, ΤΚ. Κατηγορία και Καθεστώς ΦΠΑ) και πατά Αποθήκευση για να δημιουργηθεί ο νέος πελάτης.

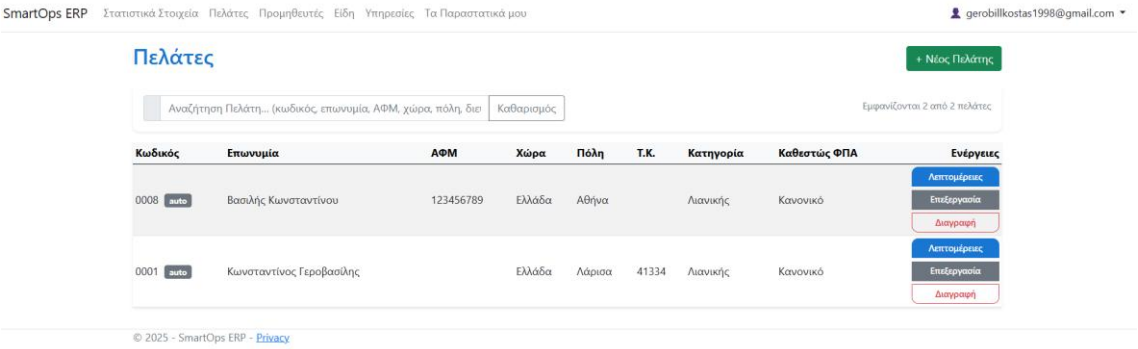


Εικόνα 7.2.3 Φόρμα δημιουργίας νέου πελάτη (Create Customer).

Ο πελάτης προστίθεται αυτόματα στη βάση δεδομένων και είναι διαθέσιμος για τιμολόγηση.

Βήμα 4: Προβολή στοιχείων πελάτη (Details Views)

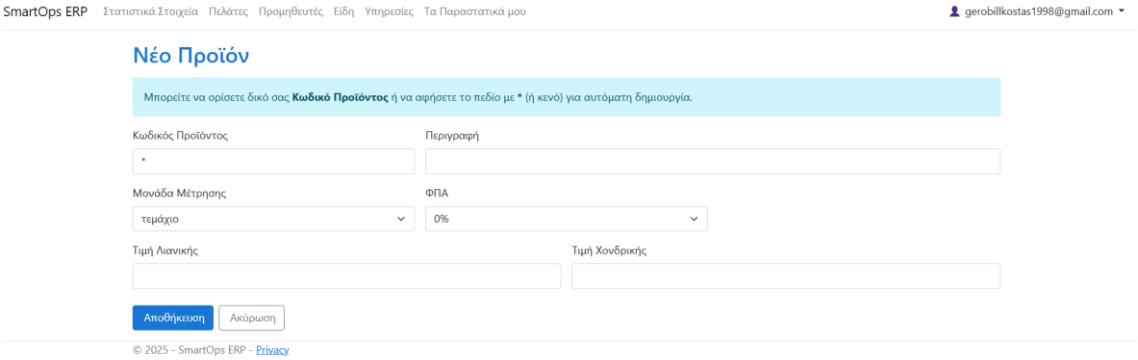
Μετά την αποθήκευση, ο πελάτης εμφανίζεται στη λίστα των πελατών.
Ο χρήστης μπορεί να επιλέξει το πεδίο Πελάτης για να δει τα αναλυτικά στοιχεία του κάθε πελάτη(κωδικός, επωνυμία, ΑΦΜ, στοιχεία διεύθυνσης, κατηγορία και Καθεστώς ΦΠΑ).



Εικόνα 7.2.4 Λίστα προβολής πελατών και των στοιχείων τους αναλυτικά

Βήμα 5: Καταχώριση νέου προϊόντος

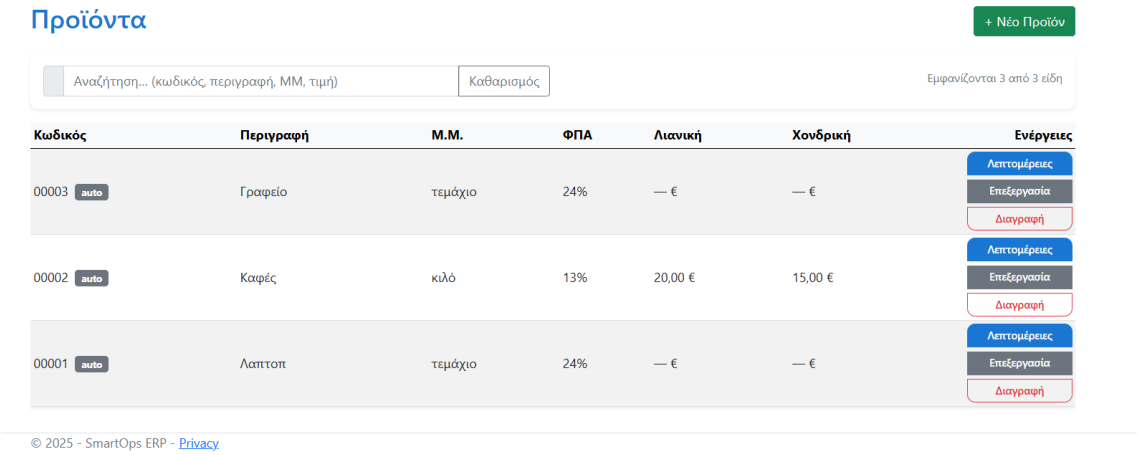
Ο χρήστης μεταβαίνει στο Προϊόντα → Νέο Προϊόν και εισάγει τα στοιχεία του προϊόντος (Κωδικός, Περιγραφή, Μ.Μ., ΦΠΑ, Τιμή Λιανικής και Χονδρικής).



Εικόνα 7.2.5 Φόρμα καταχώρισης νέου προϊόντος με αυτοματοποίηση κωδικού και default μονάδα μέτρησης και ΦΠΑ.

Το προϊόν αποθηκεύεται και είναι πλέον διαθέσιμο για χρήση στα παραστατικά.

Βήμα 6: Προβολή αναλυτικών στοιχείων προϊόντων

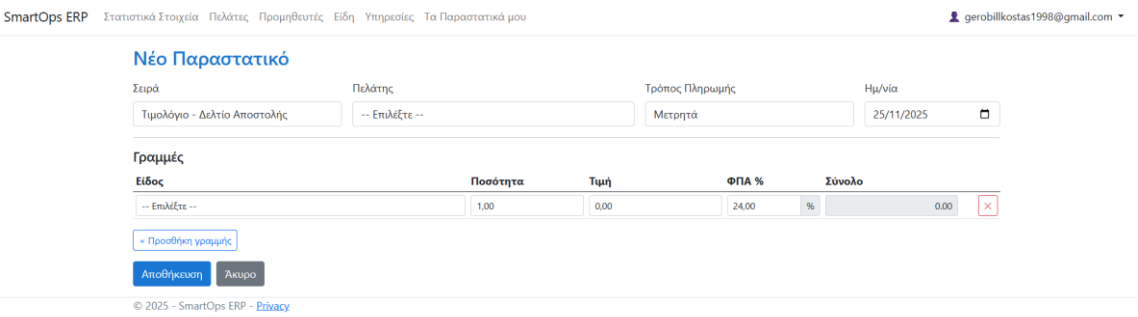


Εικόνα 7.2.6 Λίστα προβολής προϊόντων με τα απαιτούμε στοιχεία τους

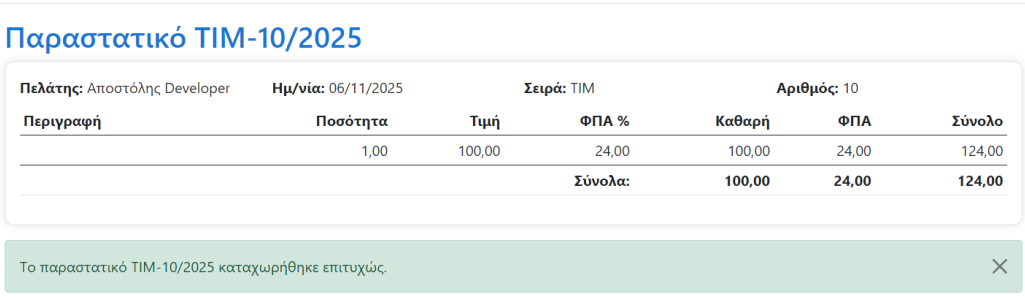
Βήμα 7: Δημιουργία νέου τιμολογίου

Από το μενού ο χρήστης επιλέγει **Νέο Παραστατικό**.
Στη επόμενη φόρμα:

- Επιλέγει σε τι παραστατικό θα ήθελα να τιμολογήσει (π.χ.Παραστατικό πωλήσεων για προϊόντα)
- Επιλέγει σειρά παραστατικού (π.χ. Τιμολόγιο – Δελτίο Αποστολής).
- Διαλέγει πελάτη από τη λίστα που εισήγαγε προηγουμένως.
- Προσθέτει ένα ή περισσότερα προϊόντα.
- Βάζει την ποσότητα και την τιμή που επιθυμεί.
- Το σύστημα υπολογίζει αυτόματα το σύνολο ΦΠΑ και το τελικό ποσό.
- Τέλος πατά Αποθήκευση.



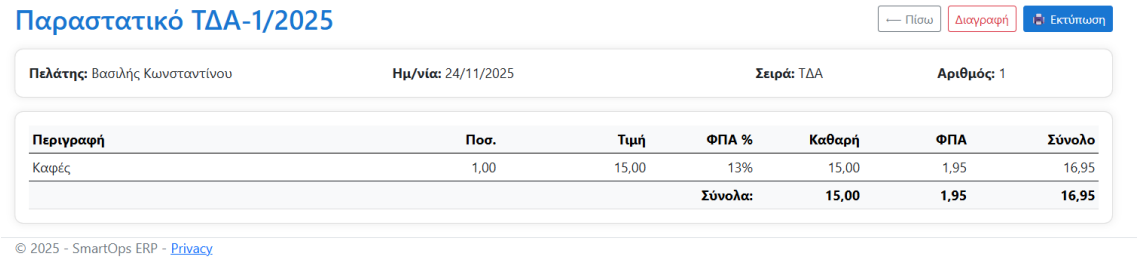
Εικόνα 7.2.7 Φόρμα δημιουργίας τιμολογίου με επιλογή πελάτη και προϊόντων.



Εικόνα 7.2.8 Υπολογισμός συνολικού ποσού και μηνύματος επιτυχίας.

Βήμα 8: Προβολή αναλυτικού τιμολογίου

Μετά την αποθήκευση, το τιμολόγιο εμφανίζεται στη λίστα των εκδοθέντων παραστατικών.
Ο χρήστης μπορεί να επιλέξει Προβολή για να δει τα αναλυτικά στοιχεία του τιμολογίου (πελάτης, γραμμές, ποσά, ημερομηνία).

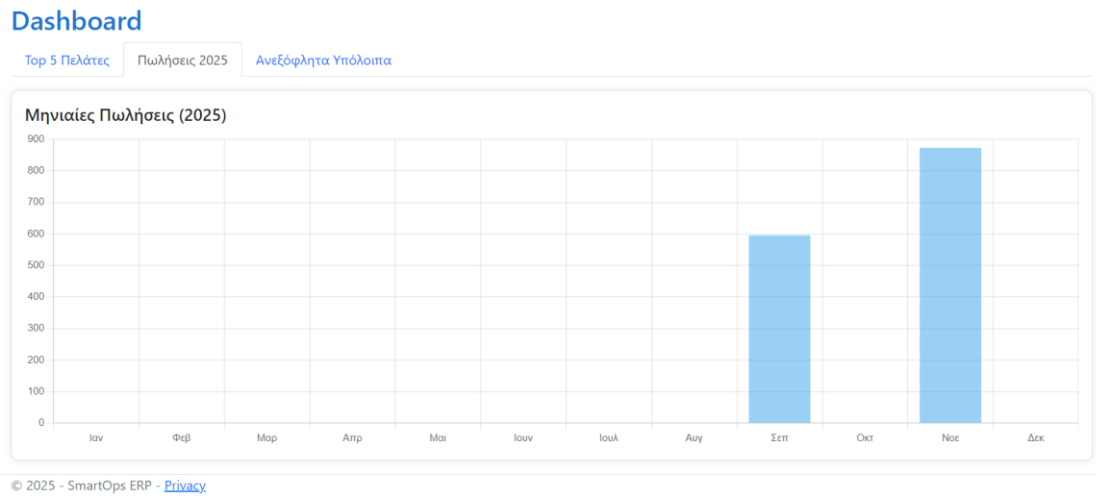


Εικόνα 7.2.9 Προβολή τιμολογίου (Details View) με αναλυτική εμφάνιση πεδίων.

Βήμα 8: Ενημέρωση Dashboard

Με την έκδοση του τιμολογίου, το Dashboard ενημερώνεται αυτόματα, εμφανίζοντας:

- Αναπροσαρμογή του γραφήματος των Top 5 Πελατών.
- Ενημέρωση των bar charts των μηνιαίων πωλήσεων του 2025.
- Αναπροσαρμογή του γραφήματος των Top 5 Προϊόντων σε πωλήσεις.



Εικόνα 7.2.10 Dashboard μετά τη δημιουργία νέου τιμολογίου - ενημερωμένοι δείκτες μηνιαίων πωλήσεων.

7.3 Παρατηρήσεις σεναρίου

Η παραπάνω διαδικασία αποδεικνύει την **πλήρη διασύνδεση** των επιμέρους ενοτήτων του SmartOps ERP Prototype.

Ο χρήστης μπορεί να πραγματοποιήσει ολόκληρο τον κύκλο εργασιών από την εισαγωγή δεδομένων έως την παρακολούθηση των αποτελεσμάτων χωρίς να εγκαταλείψει το περιβάλλον της εφαρμογής.

Η ομαλή ροή των ενεργειών, η άμεση ενημέρωση των δεδομένων και τα μηνύματα καθοδήγησης του συστήματος ενισχύουν τη φιλικότητα και αξιοπιστία της εφαρμογής.

8.Συμπεράσματα και Μελλοντική εξέλιξη

8.1 Τι μάθαμε από την ανάπτυξη

Η ανάπτυξη του SmartOps ERP Prototype αποτέλεσε μια ολοκληρωμένη διαδικασία σχεδιασμού, υλοποίησης και αξιολόγησης ενός σύγχρονου πληροφοριακού συστήματος. Η υλοποίηση του έργου ανέδειξε τόσο τη σημασία της ορθής αρχιτεκτονικής όσο και την αξία της απλότητας στη δημιουργία αποτελεσματικών επιχειρησιακών εφαρμογών.

Το SmartOps επιβεβαιώνει ότι ένα ERP δεν χρειάζεται απαραίτητα να είναι πολύπλοκο ή υπερφορτωμένο με λειτουργίες, μπορεί να είναι ευέλικτο, καθαρό και στοχευμένο, καλύπτοντας αποτελεσματικά τις βασικές ανάγκες των επιχειρήσεων. Η εργασία αυτή ανέδειξε επίσης την αξία της modular προσέγγισης, επιτρέποντας κάθε ενότητα να λειτουργεί αυτόνομα, αλλά ταυτόχρονα να συνδέεται αρμονικά με τις υπόλοιπες (Customers, Items, Services, Invoices, Dashboard).

Τι μάθαμε από την ανάπτυξη

Η υλοποίηση του SmartOps προσέφερε σημαντικά μαθήματα τόσο σε τεχνικό όσο και σε μεθοδολογικό επίπεδο:

1. Την αξία ενός καθαρού και καλοσχεδιασμένου μοντέλου δεδομένων

Ο σωστός σχεδιασμός των οντοτήτων και των σχέσεών τους (Customers, Items, Invoice–InvoiceLines κ.λπ.) αποδείχθηκε καθοριστικός για τη σταθερότητα και την επεκτασιμότητα του συστήματος.

2. Την σημασία της αρχιτεκτονικής MVC

Ο διαχωρισμός λογικής, δεδομένων και παρουσίασης πρόσφερε ευκολία συντήρησης, καθαρό κώδικα και καλύτερη οργάνωση σε όλα τα modules.

3. Την πρακτική χρησιμότητα του Entity Framework Core

Με τη χρήση migrations, το EF Core διευκόλυνε τη διαχείριση της βάσης δεδομένων και απλοποίησε τη ροή ανάπτυξης.

4. Την αξία των μικρών, συνεχών βημάτων ανάπτυξης

Η διαδικασία ανάπτυξης και δοκιμών ανέδειξε πόσο σημαντικές είναι οι μικρές αλλαγές, ο συνεχής έλεγχος και ο επανασχεδιασμός όπου χρειάζεται.

5. Τη σημασία της εμπειρίας χρήστη

Η καθαρή και ομοιόμορφη διεπαφή έκανε το SmartOps εύκολο στην εκμάθηση και πιο αποτελεσματικό στην καθημερινή χρήση.

Προκλήσεις που αντιμετωπίστηκαν

Κατά την ανάπτυξη εμφανίστηκαν αρκετές προκλήσεις, οι οποίες συνέβαλαν στη βελτίωση του τελικού αποτελέσματος:

- Ζητήματα με migrations και reset της βάσης δεδομένων.
- Προβλήματα namespace και οργάνωσης αρχείων.
- Θέματα με Razor views και σωστή απόδοση δεδομένων.
- Ανάγκη για consistent naming σε controllers, models και views.
- Επανυλοποίηση του Invoices module ώστε η ροή δημιουργίας τιμολογίου να γίνει πιο καθαρή και σταθερή.

Η επίλυση αυτών των προβλημάτων ενίσχυσε την ποιότητα του έργου και οδήγησε σε βαθύτερη κατανόηση των εργαλείων και των τεχνολογιών.

8.2 Πώς μπορεί να επεκταθεί (Cloud hosting, MyData, Reports, κ.λπ.)

Η παρούσα έκδοση του SmartOps ERP Prototype αποτελεί ένα ολοκληρωμένο, αλλά ταυτόχρονα ανοιχτό σύστημα, το οποίο μπορεί να επεκταθεί σημαντικά μελλοντικά. Μερικές από τις πιο σημαντικές δυνατότητες επέκτασης είναι οι εξής:

1. Μετάβαση σε Cloud Hosting

Η εφαρμογή μπορεί να μεταφερθεί σε Azure App Service και η βάση σε Azure SQL Database, προσφέροντας:

- 24/7 διαθεσιμότητα
- καλύτερη απόδοση
- αυτοματοποιημένα backups
- πρόσβαση από οποιοδήποτε μέρος

2. Ενσωμάτωση MyData

Προσθήκη module που θα:

- στέλνει αυτοματοποιημένα παραστατικά στην ΑΑΔΕ
- δίνει MARK στα παραστατικά
- υποστηρίζει την απαραίτητη λογική χαρακτηρισμών

3. Module Εισπράξεων & Πληρωμών

Νέα modules όπως:

- Payments (πληρωμές & εισπράξεις)
- Expenses Tracking
- Cash flow overview

4. Role-Based Access & Permissions

Διαχωρισμός δικαιωμάτων ανά χρήστη:

- Admin
- Sales
- Accounting
- Services

5. Reporting & Export

Δημιουργία αναφορών:

- PDF reports
- Excel exports

6. AI Αυτοματισμοί

Μπορεί να εξελιχθεί ένα AI Bot ώστε να:

- απαντά ερωτήματα (NLP → database queries)
- στέλνει ειδοποιήσεις (υπόλοιπα πελατών, χαμηλό απόθεμα)
- κάνει προβλέψεις βασισμένες σε ιστορικά δεδομένα

8.3 Μη Λειτουργικές απαιτήσεις του SmartOps ERP Prototype

Οι μη λειτουργικές απαιτήσεις περιγράφουν τα ποιοτικά χαρακτηριστικά που πρέπει να διαθέτει το SmartOps προκειμένου να λειτουργεί αξιόπιστα, αποδοτικά και με τρόπο φιλικό προς τον χρήστη. Αποτελούν συμπληρωματικό αλλά καθοριστικό μέρος της συνολικής αρχιτεκτονικής του συστήματος.

1. Απόδοση (Performance)

Το SmartOps σχεδιάστηκε έτσι ώστε να ανταποκρίνεται γρήγορα σε όλα τα βασικά αιτήματα του χρήστη.
Η απόδοση επιτυγχάνεται μέσω:

- χρήσης SQL Server Express με ευέλικτες δομές δεδομένων,
- LINQ queries που εκτελούνται αποδοτικά μέσω EF Core,
- ελαφριάς Razor διεπαφής χωρίς περιττά scripts,
- caching βασικών λιστών (π.χ. πελάτες, προϊόντα).

Οι βασικές λειτουργίες (φόρτωση λίστας, δημιουργία τιμολογίου, αναζήτηση) ολοκληρώνονται σε ελάχιστο χρόνο, διασφαλίζοντας ομαλή ροή εργασίας.

2. Ασφάλεια (Security)

Η ασφάλεια αποτελεί θεμελιώδες χαρακτηριστικό σε κάθε ERP σύστημα.
Το SmartOps ενσωματώνει:

- ασφαλή αποθήκευση κωδικών μέσω hashing,
- μηχανισμό εισόδου με validation και session authentication,
- προστασία από SQL Injection μέσω EF Core,
- αποσύνδεση χρήστη μετά από ένα συγκεκριμένο χρονικό όριο

Η υλοποίηση αυτή εξασφαλίζει βασικό επίπεδο προστασίας δεδομένων και αποτρέπει την πρόσβαση από τρίτους.

3. Ευχρηστία (Usability)

Η ευχρηστία αποτέλεσε κεντρική σχεδιαστική αρχή.
Η διεπαφή είναι:

- απλή,
- καθαρή,
- με ενιαία δομή σε όλο το σύστημα,
- με drop-down menus, σαφή labels και άμεση ανατροφοδότηση.

Ο χρήστης μπορεί να εκτελέσει όλες τις βασικές λειτουργίες χωρίς εκτεταμένη εκπαίδευση.

4. Επεκτασιμότητα (Scalability)

Παρότι αποτελεί ERP Prototype, το SmartOps έχει σχεδιαστεί με τρόπο που επιτρέπει μελλοντική ανάπτυξη χωρίς ανασχεδιασμό:

- modular controllers για κάθε ενότητα (Customers, Items, Invoices),
- καθαρή αρχιτεκτονική MVC,
- επεκτάσιμες οντότητες στη βάση,
- χρήση υπηρεσιών (services) για επιχειρησιακή λογική.

Το σύστημα μπορεί εύκολα να επεκταθεί με νέα modules (π.χ. Payments, Warehouses, MyData).

5. Αξιοπιστία & Ακεραιότητα (Reliability & Data Integrity)

Η αξιοπιστία εξασφαλίζεται μέσω:

- χρήσης foreign keys και περιορισμών (constraints),
- transactions κατά τη δημιουργία τιμολογίων ώστε να αποφεύγονται μερικές εγγραφές,
- validation των δεδομένων πριν από κάθε αποθήκευση.

Έτσι, το σύστημα αποτρέπει λανθασμένες καταχωρήσεις και διατηρεί συνεπή τα δεδομένα.

6. Συντηρησιμότητα (Maintainability)

Το SmartOps ακολουθεί αρχές καθαρού κώδικα (Clean Architecture σε βασικό επίπεδο):

- ευδιάκριτος διαχωρισμός Models–Controllers–Views,
- χρήση ViewModels όπου χρειάζεται,
- ομοιόμορφη ονοματολογία και κοινή δομή κώδικα,
- αρχεία οργανωμένα σε φακέλους ανά module.

Οι παραπάνω πρακτικές επιτρέπουν εύκολη συντήρηση και εξέλιξη του έργου από άλλους προγραμματιστές.

7. Διαθεσιμότητα (Availability)

Στην παρούσα έκδοση λειτουργεί ως τοπική web εφαρμογή (Local Hosting).
Με μελλοντική μετάβαση σε cloud (Azure), το SmartOps μπορεί να επιτύχει:

- υψηλή διαθεσιμότητα,
- συνεχή πρόσβαση,
- πρόσβαση από οποιοδήποτε μέρος

8. Φιλικότητα στην εγκατάσταση (Installability)

Το σύστημα μπορεί να εγκατασταθεί και να τρέξει:

- σε οποιονδήποτε σύγχρονο υπολογιστή Windows,
- με SQL Server Express,
- χρησιμοποιώντας Visual Studio και .NET 8.

Απαιτεί ελάχιστη παραμετροποίηση, ενώ η βάση δημιουργείται αυτόματα μέσω migrations.

8.4 SWOT Analysis – SmartOps ERP Prototype

Strengths (Δυνατά Σημεία)

- Απλή και καθαρή διεπαφή, εύκολη στην εκμάθηση.
- Καθαρή και επεκτάσιμη αρχιτεκτονική (MVC + EF Core).
- Γρήγορη απόδοση και σταθερή ροή χρήστη.
- Πλήρης διασύνδεση των modules (Customers → Invoices → Dashboard).
- Υποστήριξη τόσο προϊόντων όσο και υπηρεσιών.
- Εκπαιδευτική αξία και πλήρη διαφάνεια κώδικα.
- Δυνατότητα ενσωμάτωσης AI Bot μελλοντικά.

Weaknesses (Αδυναμίες)

- Δεν υποστηρίζει ακόμη MyData διασύνδεση.
- Δεν υπάρχει module για πληρωμές ή αποθήκη.
- Έλλειψη προηγμένων ρυθμίσεων ασφαλείας (κυρίως στην ανάκτηση κωδικού πρόσβασης).
- Βασίζεται σε τοπική εγκατάσταση — όχι πλήρως cloud-based στην τρέχουσα μορφή.

Opportunities (Ευκαιρίες)

- Μετάβαση σε cloud hosting (Azure).
- Ενσωμάτωση AI για προβλέψεις, ειδοποιήσεις, έξυπνες αναζητήσεις.
- Υποστήριξη MyData για αποστολή παραστατικών.
- Ανάπτυξη modules για Expenses και Warehouses.
- Μελλοντική εμπορική αξιοποίηση ως mini business app.

Threats (Κίνδυνοι / Απειλές)

- Ο ανταγωνισμός από μεγάλα εμπορικά ERP.
- Απαιτήσεις για νομική συμμόρφωση (MyData, GDPR).
- Τεχνολογικές αλλαγές που απαιτούν συχνές αναβαθμίσεις.
- Εξάρτηση από SQL Server και .NET οικοσύστημα.

8.5 Συνολική Αξιολόγηση

Η ανάπτυξη του SmartOps ERP Prototype απέδειξε ότι ένα μικρό, καθαρό και σωστά σχεδιασμένο ERP μπορεί να είναι πλήρως λειτουργικό, επεκτάσιμο και πραγματικά χρήσιμο. Η εργασία πέτυχε:

- να υλοποιήσει ένα ολοκληρωμένο πληροφοριακό σύστημα,
- να παρουσιάσει καθαρή αρχιτεκτονική MVC,

- να τεκμηριώσει τη διαδικασία μέσω αναλυτικής θεωρίας και πρακτικών σεναρίων,
- και να ανοίξει τον δρόμο για μελλοντική ανάπτυξη σε πιο επαγγελματικό επίπεδο.

Το SmartOps μπορεί εύκολα να αποτελέσει τη βάση για μια πραγματική εμπορική εφαρμογή ή για περαιτέρω ερευνητική εργασία πάνω σε ERP συστήματα, αυτόματες διαδικασίες ή AI λειτουργικότητα.

9. Πηγές – Βιβλιογραφία

1. **Andriotis, G. (2017)** *Ανάλυση και Σχεδίαση Πληροφοριακών Συστημάτων. Αθήνα: Κλειδάριθμος.*
2. **Bootstrap (2024)** *Bootstrap 5 Documentation*. Διαθέσιμο στο: https://getbootstrap.com
3. **Epsilon Net (2025)** *Epsilon Net – Λογισμικό και Επιχειρηματικές Λύσεις.* Διαθέσιμό στο: https://www.epsilonnet.gr
4. **Entersoft (2025)** *Entersoft Business Software Solutions*. Διαθέσιμο στο: https://www.entersoft.eu
5. **Manolopoulos, I. (2015)** *Βάσεις Δεδομένων: Σχεδίαση και Υλοποίηση*. Θεσσαλονίκη: Τζιόλα.
6. **Martin, R.C. (2009)** *Clean Code: A Handbook of Agile Software Craftsmanship*. Upper Saddle River, NJ: Prentice Hall.
7. **Microsoft (2024)** *ASP.NET Core MVC Documentation*. Διαθέσιμο στο: https://learn.microsoft.com/aspnet/core
8. **Microsoft (2024)** *Entity Framework Core Documentation*. Διαθέσιμο στο: https://learn.microsoft.com/ef/core
9. **Microsoft (2024)** *SQL Server Documentation*. Διαθέσιμο στο: https://learn.microsoft.com/sql
10. **Sommerville, I. (2016)** *Software Engineering*. 10th edn. Harlow: Pearson Education.
11. **SoftOne (2025)** *SoftOne Cloud ERP & Business Solutions*. Διαθέσιμο στο: https://www.softone.gr
12. **Vasileiou, G. (2020)** *Πληροφοριακά Συστήματα Διοίκησης*. Αθήνα: Παπαζήσης.