ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή Εργασία

Τίτλος Πτυχιακής Εργασίας	(Ελληνικά) Εφαρμογή Παραγγελιών Οχημάτων με Υπηρεσιοστρεφή Αρχιτεκτονική (SOA) πελάτη-εξυπηρετητή σε .NET (Αγγλικά) Vehicle Ordering Application based on Service-Oriented Architecture client-server using .NET
Ονοματεπώνυμο Φοιτητή	ΣΤΑΜΑΤΙΟΣ ΛΙΑΤΣΟΣ
Πατρώνυμο	ΔΗΜΗΤΡΙΟΣ
Αριθμός Μητρώου	Π/06198
Επιβλέπων	ΧΡΙΣΤΟΣ ΔΟΥΛΗΓΕΡΗΣ, ΚΑΘΗΓΗΤΗΣ

September 2024/Σεπτέμβριος 2024

Copyright ©

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα διπλωματική εργασία εκπονήθηκε στα πλαίσια της ολοκλήρωσης των σπουδών μου στο Προπτυχιακό Πρόγραμμα Σπουδών του Τμήματος Πληροφορικής του Πανεπιστημίου Πειραιώς στα Πληροφοριακά Συστήματα.

Θα ήθελα να ευχαριστήσω τον καθηγητή μου κ. Χρήστο Δουληγέρη για το θέμα της παρούσας διπλωματικής εργασίας, την οποία μου προσέφεραν την δυνατότητα να εκπονήσω.

Θα ήθελα επίσης να ευχαριστήσω τον Δρ. Απόστολο Καράλη, του οποίου η βοήθεια ήταν σημαντική καθ’ όλη την διάρκεια της ανάπτυξης και εκπόνησης της συγκεκριμένης εργασίας.

Τέλος, ευχαριστίες θα ήθελα να αποδώσω στους γονείς μου και στη γυναίκα μου, που με βοήθησαν με τον δικό τους τρόπο ο καθένας τους καθ’ όλη την διάρκεια της εκπόνησης της εργασίας αυτής.

Περίληψη

Ο σκοπός της συγκεκριμένης διπλωματικής εργασίας είναι η ανάπτυξη ενός πληροφοριακού συστήματος, το οποίο ως στόχο έχει την διασύνδεση προμηθευτών οχημάτων (importers) με τις διάφορες αντιπροσωπείες αυτοκινήτων με σκοπό την παραγγελία νέων οχημάτων.

Κατά την υλοποίηση χρησιμοποιήθηκε το service oriented architecture (SOA) για την υλοποίηση των διάφορων υπηρεσιών (services), που το καθένα με την σειρά του έχει δημιουργηθεί αναλόγως με τον τομέα (domain) πάνω στο οποίο επιδρά. Ως εκ τούτου, με βάση το πεδίο στο οποίο επικεντρώθηκε η πτυχιακή (παραγγελίες οχημάτων) δημιουργήθηκαν τρεις διαφορετικές υπηρεσίες, μία για την διαχείριση των πελάτων/χρηστών, μία για τη διαχείριση των οχημάτων/μοντέλων και μία για τη διαχείριση των παραγγελιών.

Η επικοινωνία μεταξύ των υπηρεσιών πραγματοποιείται ασύγχρονα, έτσι ώστε να υπάρχει μια χαλαρή σύνδεση (loosely coupled) μεταξύ των υπηρεσιών. Η διασύνδεση του γραφικού περιβάλλοντος (Graphical User Interface) με τις υπηρεσίες γίνεται σύγχρονα με την χρήση GRPC (Google Remote Procedure Calls).

Για τη δημιουργία του User Interface (UI) χρησιμοποιήθηκε η βιβλιοθήκη Blazor.NET. Το γραφικό περιβάλλον περιλαμβάνει το διαχειριστικό μέρος καθώς και το περιβάλλον διασύνδεσης του χρήστη με το σύστημα.

Λέξεις Κλειδιά: Ordering, Microservice, Docker, SOA, GRPC

Abstract

The purpose of this specific thesis is the development of an information system, which aims to connect vehicle importers with various car dealerships for the purpose of ordering new vehicles.

During the implementation, service-oriented architecture (SOA) was used to implement various services, each created according to the specific domain it influences. Therefore, based on the focus of the thesis (vehicle orders), three different services were created for managing customers/users, vehicles/models, and orders.

The communication between services is asynchronous to ensure a loosely coupled connection between them. The graphical user interface (GUI) is connected to the services synchronously using GRPC (Google Remote Procedure Calls).

For the creation of the User Interface (UI), the Blazor.NET library was used. The graphical environment includes the management part as well as the user interface for interacting with the system.

Key Words: Ordering, Microservice, Docker, SOA, GRPC

Περιεχόμενα

Εικόνες

Εισαγωγή

Στον χώρο της αυτοκινητοβιομηχανίας υπάρχουν διάφορα συστήματα τα οποία καλύπτουν ως σκοπό έχουν να καλύπτουν τις καθημερινές ανάγκες των χρηστών της βιομηχανίας αυτής. Οι πιο συνήθεις εργασίες που καλούνται να καλύψουν αυτά τα συστήματα είναι η διαχείριση πελατών και παραγγελιών.

Τα δύο πιο γνωστά ολοκληρωμένα συστήματα τα οποία καλύπτουν τις ανάγκες αυτές, είναι αυτά το «CDK DMS» (της CDK Global) και το «PYLON Auto & Moto Cube» (της EPSILONNET). Τα συστήματα αυτά παρέχουν στον χρήστη της δυνατότητα δημιουργίας και διαχείρισης του πελατολογίου της εκάστοτε επιχείρησης καθώς και την δημιουργία και την καταχώρηση/ολοκλήρωση των παραγγελιών καθώς και την διαχείριση των βιβλίων τις εταιρείας σε ηλεκτρονική μορφή (σύνδεση με myData της ΑΑΔΕ).

Ο σκοπός της συγκεκριμένης διπλωματικής εργασίας είναι η ανάπτυξη ενός πληροφοριακού συστήματος, το οποίο ως στόχο έχει τη διασύνδεση προμηθευτών οχημάτων (importers) με τις διάφορες αντιπροσωπείες αυτοκινήτων για την παραγγελία νέων οχημάτων. Η παρούσα μελέτη επικεντρώνεται στη δημιουργία ενός ολοκληρωμένου και αποδοτικού δικτύου επικοινωνίας και συνεργασίας μεταξύ των εμπλεκόμενων μερών, επιτρέποντας την αυτοματοποίηση και βελτιστοποίηση της διαδικασίας παραγγελιοδοσίας και διανομής νέων οχημάτων.

Το πληροφοριακό σύστημα που θα αναπτυχθεί θα παρέχει μια πλατφόρμα, όπου οι αντιπροσωπείες αυτοκινήτων θα μπορούν να δημιουργούν νέους χρήστες και πελάτες, και να διαχειρίζονται τις παραγγελίες τους. Οι νέοι πελάτες θα δημιουργούνται βάσει του προφίλ των χρηστών, διευκολύνοντας την παρακολούθηση και διαχείριση των παραγγελιών.

Στον κόσμο της σύγχρονης ανάπτυξης λογισμικού, η υιοθέτηση της Αρχιτεκτονικής Προσανατολισμένης προς τις Υπηρεσίες (Service-Oriented Architecture - SOA) έχει αναδειχθεί ως η προσέγγιση που ενισχύει την αρθρωτότητα (modularity), την επεκτασιμότητα (scalability) και τη βελτιωμένη ευελιξία του συστήματος (flexilibity). Αυτή η εργασία εξετάζει την εφαρμογή των αρχών της SOA στη δημιουργία ενός πλήρους πληροφοριακού συστήματος. Κεντρικό σημείο αυτής της προσπάθειας είναι η δημιουργία τριών διακριτών μικρο-υπηρεσιών - των μικρο-υπηρεσιών Οχήματος, Πελάτη και Παραγγελιών - σχεδιασμένων να αλληλοεπιδρούν απροβλημάτιστα μέσα σε ένα δυναμικό οικοσύστημα.

Το περιβάλλον επαφής του χρήστη (frontend) του συστήματος δημιουργήθηκε χρησιμοποιώντας το Blazor, ένα πλαίσιο που διευκολύνει την ανάπτυξη διαδραστικών και δυναμικών εφαρμογών ιστού. Αυτή η διεπαφή χρήστη καλύπτει τις διαχειριστικές

ανάγκες για την παραμετροποίηση του συστήματος και ταυτόχρονα παρέχει ένα φιλικό προς τον χρήστη περιβάλλον για την αλληλεπίδραση με το σύστημα με σκοπό την καταχώρηση των απαραίτητων συναλλαγών.

Μόλις οι παραγγελίες οριστικοποιηθούν από τους χρήστες των αντιπροσωπειών, θα μεταφέρονται αυτόματα στο σύστημα ERP του εισαγωγέα, το οποίο είναι το Microsoft Business Central. Στο πλαίσιο αυτής της εργασίας, θα δημιουργηθεί ένα extension app για το Microsoft Business Central, που θα παρέχει τη διασύνδεση και θα επιτρέπει την αυτοματοποίηση της διαδικασίας εισαγωγής και διαχείρισης των παραγγελιών στο ERP σύστημα. . Επιπλέον, στο ERP θα ενημερώνονται όχι μόνο οι παραγγελίες, αλλά και οι πληροφορίες για τους πελάτες και τα οχήματα/μοντέλα που επιλέχθηκαν για παραγγελία. Αυτή η συνεργασία με το Microsoft Business Central ERP ενισχύει τη συνολική αποτελεσματικότητα και συνοχή της διαχείρισης παραγγελιών, προσθέτοντας ένα επίπεδο εξελιγμένης λειτουργικότητας στο πληροφοριακό σύστημα.

Η επικοινωνία μεταξύ των συστημάτων θα γίνεται ασύγχρονα με τη χρήση Kafka messages, διασφαλίζοντας την αποτελεσματική και αξιόπιστη μεταφορά των δεδομένων. Αυτή η αρχιτεκτονική όχι μόνο προάγει την αυτονομία των μεμονωμένων υπηρεσιών, αλλά θεμελιώνει επίσης μια βάση για την επεκτασιμότητα και την προσαρμοστικότητα ανταποκρινόμενη στις εξελισσόμενες επιχειρηματικές ανάγκες.

Το σύστημα αυτό θα ενσωματώνει λειτουργίες όπως:

- **Διαχείριση Χρηστών και Πελατών**: Οι αντιπροσωπείες θα μπορούν να δημιουργούν νέους χρήστες για το σύστημα παραγγελιοληψίας, καθώς και να δημιουργούν και να διαχειρίζονται προφίλ πελατών βασισμένα στα προφίλ των χρηστών.
- **Υποβολή και Διαχείριση Παραγγελιών**: Οι χρήστες των αντιπροσωπειών θα μπορούν να υποβάλλουν νέες παραγγελίες , να τις επεξεργάζονται και να τις οριστικοποιούν. Μόλις μια παραγγελία οριστικοποιηθεί, μεταφέρεται αυτόματα στο Microsoft Business Central μέσω του extension app.
- **Αυτόματη Μεταφορά Δεδομένων στο Microsoft Business Centra**l: Η αυτόματη διασύνδεση με το Microsoft Business Central θα επιτρέπει την άμεση ενημέρωση των παραγγελιών, εξαλείφοντας την ανάγκη για χειροκίνητη εισαγωγή δεδομένων και μειώνοντας τα σφάλματα.
- **Ασύγχρονη Επικοινωνία μέσω Kafka Messages**: Η χρήση του Apache Kafka για την ασύγχρονη επικοινωνία μεταξύ των συστημάτων διασφαλίζει την αξιόπιστη μεταφορά δεδομένων, επιτρέποντας την επεξεργασία των μηνυμάτων σε πραγματικό χρόνο και την κλιμάκωση της εφαρμογής ανάλογα με τις ανάγκες.

- **Παρακολούθηση Παραγγελιών μέσω Email**: Η πορεία των παραγγελιών θα παρακολουθείται μέσω ειδοποιήσεων email, με έμφαση στην τιμολόγηση των παραγγελιών. Οι αντιπροσωπείες θα λαμβάνουν ενημερώσεις σχετικά με την τιμολόγηση των παραγγελιών, διασφαλίζοντας τη διαφάνεια και την έγκαιρη ενημέρωση των οικονομικών στοιχείων.
- **Ενημέρωση Αρχικού Συστήματος Παραγγελιών**: Μόλις ολοκληρωθεί η τιμολόγηση στο ERP, το αρχικό σύστημα παραγγελιών ενημερώνεται εκ νέου ώστε να έχει την οριστικοποιημένη εικόνα της παραγγελίας και το αντίγραφο του τιμολογίου.

Οφέλη από την υλοποίηση αυτού του πληροφοριακού συστήματος περιλαμβάνουν την αύξηση της αποδοτικότητας, τη μείωση του κόστους διαχείρισης παραγγελιών, τη βελτίωση της ακρίβειας και της διαφάνειας στις συναλλαγές, καθώς και την ενίσχυση της συνεργασίας και της επικοινωνίας μεταξύ των προμηθευτών και των αντιπροσωπειών. Επιπλέον, η αυτοματοποίηση πολλών διαδικασιών αναμένεται να μειώσει τα σφάλματα και να βελτιώσει τη συνολική εμπειρία των χρηστών του συστήματος.

Στην παρούσα διπλωματική εργασία, θα αναλύσουμε τόσο την συνολική αρχιτεκτονική του συστήματος, τα επιμέρους συστατικά αλλά και την μεταξύ τους σύνδεση. Αρχικά, θα παρουσιάσουμε τα βασικά στοιχεία της Αρχιτεκτονική Προσανατολισμένη προς τις Υπηρεσίες (SOA) και των μικρο-υπηρεσιών, αναλύοντας τα πλεονεκτήματα και τα μειονεκτήματα αυτής της προσέγγισης. Στη συνέχεια, θα περιγράψουμε την αρχιτεκτονική του συστήματος που αναπτύχθηκε, εστιάζοντας στις επιμέρους υπηρεσίες, τη λειτουργικότητά τους, και τις τεχνολογίες που χρησιμοποιούνται. Τέλος, θα παρουσιάσουμε τις λειτουργίες του συστήματος, περιγράφοντας τη ροή των λειτουργιών με τη βοήθεια screenshots για καλύτερη κατανόηση. Αυτή η δομή θα μας επιτρέψει να καταγράψουμε και να επεξηγήσουμε λεπτομερώς την αρχιτεκτονική και την υλοποίηση του συστήματος, καθώς και τα πρακτικά οφέλη και τις προκλήσεις που αντιμετωπίστηκαν κατά τη διάρκεια της ανάπτυξής του.

Κεφάλαιο 1 - Υπηρεσιοστρεφής Αρχιτεκτονική

Η Αρχιτεκτονική Προσανατολισμένη προς τις Υπηρεσίες (Service-Oriented Architecture - SOA) αναπτύχθηκε ως μια προσέγγιση για τη σχεδίαση και την υλοποίηση λογισμικών συστημάτων που είναι εύκαμπτα, επεκτάσιμα και ανθεκτικά. Βασίζεται στην ιδέα της ανάπτυξης λειτουργικών μονάδων ως ανεξάρτητες υπηρεσίες που παρέχουν συγκεκριμένες λειτουργίες μέσω καλά ορισμένων διεπαφών.

Σύμφωνα με τον Thomas Erl, έναν από τους κορυφαίους ειδικούς στον τομέα της SOA, η αρχιτεκτονική αυτή βασίζεται σε τέσσερις κεντρικές αρχές:

- **Υπηρεσίες**: Οι υπηρεσίες αντιπροσωπεύουν τη βασική μονάδα λειτουργικότητας στην SOA. Κάθε υπηρεσία είναι αυτόνομη, ανεξάρτητη και προσφέρει συγκεκριμένη λειτουργικότητα.
- **Διασύνδεση**: Οι υπηρεσίες επικοινωνούν μεταξύ τους μέσω καλά καθορισμένων διεπαφών, που ορίζουν τις λειτουργίες που προσφέρονται και πώς πρέπει να χρησιμοποιούνται.
- **Ανακάλυψη**: Η SOA προωθεί την εύρεση και την ανακάλυψη υπηρεσιών μέσω καταλόγων και υπηρεσιών, προκειμένου να επαναχρησιμοποιούνται και να ενσωματώνονται εύκολα σε διάφορα συστήματα.
- **Διαχείριση**: Η διαχείριση των υπηρεσιών περιλαμβάνει την παρακολούθηση της απόδοσης, τη διαθεσιμότητας και την ασφάλεια τους, καθώς και τη διαχείριση των αλληλεπιδράσεών τους με άλλες υπηρεσίες.

Αυτές οι αρχές διαμορφώνουν τη βάση για τον σχεδιασμό και την υλοποίηση της SOA, επιτρέποντας τη δημιουργία ευέλικτων και επεκτάσιμων συστημάτων που μπορούν να προσαρμοστούν στις μεταβαλλόμενες ανάγκες των επιχειρήσεων.

Ένα άλλο σημαντικό στοιχείο της SOA είναι η επαναχρησιμοποίηση των υπηρεσιών, η οποία επιτρέπει την αποδοτική ανάπτυξη και συντήρηση των συστημάτων, αλλά και την ευελιξία στην αλλαγή και την επέκτασή τους με την πάροδο του χρόνου. Αυτές οι αρχές και πρακτικές αποτελούν τη θεμελιώδη βάση για την ανάπτυξη ενός ευέλικτου, ανθεκτικού και αποδοτικού πληροφοριακού συστήματος βασισμένου σε SOA.

Για να γίνει πιο κατανοητή η χρήση αλλά και τα πλεονεκτήματα αυτής της αρχιτεκτονικής είναι χρήσιμο να την συγκρίνουμε με το μονολιθικό τρόπο ανάπτυξης εφαρμογών όπου οι επιχειρησιακές εφαρμογές χτίζονται ως ένα μοναδικό σύνολο αποτελούμενο από τρία βασικά μέρη: μια διεπαφή χρήστη που τρέχει στη μηχανή του χρήστη (αποτελούμενη από σελίδες HTML και JavaScript που εκτελούνται στο πρόγραμμα περιήγησης στον υπολογιστή του χρήστη), μια βάση δεδομένων

(αποτελούμενη από πολλούς πίνακες που εισάγονται σε ένα κοινό, και συνήθως σχεσιακό, σύστημα διαχείρισης βάσεων δεδομένων), και μια εφαρμογή στον διακομιστή. Η εφαρμογή στον διακομιστή διαχειρίζεται αιτήματα HTTP, εκτελεί την λογική, ανακτά και ενημερώνει δεδομένα από τη βάση δεδομένων, και εν τέλει επιλέγει και τροφοδοτεί προβολές HTML που θα σταλούν στο πρόγραμμα περιήγησης. Αυτή η εφαρμογή στον διακομιστή είναι ένας μονόλιθος - ένα μόνο λογικό εκτελέσιμο. Οποιεσδήποτε αλλαγές στο σύστημα περιλαμβάνουν την κατασκευή και ανάπτυξη μιας νέας έκδοσης της εφαρμογής στον διακομιστή. Όλη η λογική για την επεξεργασία ενός αιτήματος εκτελείται σε ένα μόνο διεργασιακό περιβάλλον, χρησιμοποιώντας τα βασικά χαρακτηριστικά της γλώσσας σας για να διαιρεθεί η εφαρμογή σε κλάσεις, συναρτήσεις και ονοματοχώρους. Οι μονολιθικές εφαρμογές μπορούν να είναι επιτυχείς, αλλά η διαχείριση τους γίνεται όλο και δυσκολότερη - ειδικά καθώς όλο και περισσότερες εφαρμογές αναπτύσσονται στο cloud. Οι κύκλοι αλλαγών είναι δεμένοι μαζί - μια αλλαγή που γίνεται σε μικρό μέρος της εφαρμογής απαιτεί τον ολοκληρωτικό ανασχεδιασμό και ανάπτυξη του μονολιθικού. Με την πάροδο του χρόνου είναι συχνά δύσκολο να διατηρηθεί μια καλή δομημένη δομή, καθιστώντας πιο δύσκολο το να διατηρηθούν οι αλλαγές που θα έπρεπε να επηρεάζουν μόνο μία προγραμματιστική ενότητα. Η κλιμάκωση απαιτεί την κλιμάκωση ολόκληρης της εφαρμογής αντί για μέρη της που απαιτούν μεγαλύτερο πόρο.

Αυτές οι δυσκολίες οδήγησαν στο αρχιτεκτονικό στυλ των μικρο-υπηρεσιών: την κατασκευή εφαρμογών ως συνόλων υπηρεσιών. Εκτός από το γεγονός ότι οι υπηρεσίες είναι ανεξάρτητες στην αναπτυξιακή διαδικασία και στην κλιμάκωση, κάθε υπηρεσία παρέχει επίσης ένα ακέραιο όριο μονάδας, επιτρέποντας ακόμη και για τη χρήση διαφορετικών γλωσσών προγραμματισμού σε διαφορετικές υπηρεσίες. Μπορούν επίσης να τις διαχειριστούν διαφορετικές ομάδες. Ακόμα και η λογική της διαίρεσης και ανάθεσης ομάδων διαφέρει αναμέσα στις παραδοσιακές εφαρμογές και στις μικρο-υπηρεσίες. Συχνά η διοίκηση επικεντρώνεται στο επίπεδο τεχνολογίας, οδηγώντας σε διάσπαση των ομάδων σύμφωνα με την τεχνολογία: ομάδες για το UI, τη λογική στον διακομιστή και τη βάση δεδομένων. Όταν οι ομάδες διαχωρίζονται με βάση αυτές τις γραμμές, ακόμη και απλές αλλαγές μπορούν να οδηγήσουν σε πρόγραμμα που απαιτεί συνεργασία μεταξύ των ομάδων που μεταφράζεται σε χρόνο και έγκριση προϋπολογισμού. Η προσέγγιση των μικρο-υπηρεσιών στη διαίρεση είναι διαφορετική, διαιρώντας τις υπηρεσίες γύρω από τις επιχειρηματικές δυνατότητες. Τέτοιες υπηρεσίες παρέχουν μια ευρεία υλοποίηση λογισμικού για αυτήν την επιχειρηματική περιοχή, συμπεριλαμβανομένου του UI, της αποθήκευσης δεδομένων και οποιασδήποτε εξωτερικής συνεργασίας. Ως εκ τούτου, οι ομάδες είναι αυτόνομες και περιλαμβάνουν το πλήρες φάσμα των δεξιοτήτων που απαιτούνται για την ανάπτυξη και την υλοποίηση των αλλαγών.

Ένα ακόμα πρόβλημα της κεντρικής διακυβέρνησης είναι η τάση να επιλέγουμε μονόπλευρα τεχνολογικά πλαίσια. Η εμπειρία δείχνει ότι αυτή η προσέγγιση είναι περιοριστική. Είναι προτιμότερο να χρησιμοποιούμε το κατάλληλο εργαλείο για τη δουλειά και αν και οι μονολιθικές εφαρμογές μπορούν να επωφεληθούν από διαφορετικές γλώσσες κατά κάποιο τρόπο, αυτό δεν είναι τόσο συνηθισμένο. Διαιρώντας τα συστατικά του μονόλιθου σε υπηρεσίες, έχουμε τη δυνατότητα να επιλέξουμε κάθε φορά ποια τεχνολογία θα χρησιμοποιήσουμε κατά την κατασκευή τους. Οι ομάδες που αναπτύσσουν μικρο-υπηρεσίες προτιμούν επίσης διαφορετική προσέγγιση στα πρότυπα.

Όταν πρόκειται για τη διαχείριση των βάσεων δεδομένων σε μια μονολιθική εφαρμογή ένας κεντρικός συστατικός ρόλος παίζεται από την κεντρική βάση δεδομένων. Συνήθως, όλα τα στοιχεία της εφαρμογής έχουν πρόσβαση σε αυτήν την κεντρική βάση δεδομένων, ενώ οποιαδήποτε αλλαγή ή ενημέρωση απαιτεί αλλαγές στην ίδια τη βάση δεδομένων. Αυτό μπορεί να οδηγήσει σε πολύπλοκες διαδικασίες ανάπτυξης και αναβάθμισης, καθώς και στον κίνδυνο της επικίνδυνης συγχύσεως δεδομένων. Αντίθετα, σε μια εφαρμογή που διαχωρίζεται σε μικρο-υπηρεσίες, κάθε υπηρεσία διατηρεί τη δική της αυτόνομη βάση δεδομένων. Αυτό σημαίνει ότι κάθε υπηρεσία είναι υπεύθυνη για τη διαχείριση των δεδομένων της, και έχει την ελευθερία να επιλέξει τον τύπο και τη δομή της βάσης δεδομένων που καλύπτει καλύτερα τις ανάγκες της. Αυτό έχει ως αποτέλεσμα την αποφυγή της επιρροής μιας κεντρικής βάσης δεδομένων και τη μείωση του κινδύνου πιθανών αλληλεπιδράσεων μεταξύ των υπηρεσιών. Η αυτονομία των βάσεων δεδομένων κάθε μικρο-υπηρεσίας σημαίνει επίσης ότι οι ομάδες που αναπτύσσουν αυτές τις υπηρεσίες έχουν τον πλήρη έλεγχο των δεδομένων τους και μπορούν να τα προσαρμόζουν και να τα επεξεργάζονται κατάλληλα χωρίς την ανάγκη να συντονίζονται με άλλες ομάδες ή να περιμένουν την έγκριση από κεντρικές αρχές. Αυτή η ευελιξία επιτρέπει στις ομάδες να αναπτύσσουν και να ενημερώνουν τις υπηρεσίες τους με αυξημένη ταχύτητα και αποτελεσματικότητα, ενώ ελαχιστοποιεί τον κίνδυνο διακοπών στο σύνολο της εφαρμογής.

Κεφάλαιο 2 - Αρχιτεκτονική Συστήματος

Η εφαρμογή παραγγελίας οχημάτων που αναπτύχθηκε στα πλαίσια αυτής της εργασίας έχει ως στόχο να επιδείξει την διασύνδεση μεταξύ διαφόρων υπηρεσιών χρησιμοποιώντας την Αρχιτεκτονική Προσανατολισμένη προς τις Υπηρεσίες (Service – Oriented Architecture) με ασφαλή διασύνδεση μεταξύ των συστημάτων. Για τον λόγο αυτό κατασκευάστηκαν μικρο-υπηρεσίες οι οποίες οι οποίες αλληλεπιδρούν είτε σύγχρονα είτε ασύγχρονα ανάλογα με το πόσο γρήγορος θέλουμε να είναι ο χρόνος απόκρισης κάθε υπηρεσίας.

Μικρο-υπηρεσία Χρηστών/Πελάτων (service 1): Η πρώτη μικρο-υπηρεσία έχει να κάνει με την διαχείριση των χρηστών του συστήματος καθώς και των πελατών. Υπάρχει μία σχέση εξάρτησης μεταξύ χρήστη και πελάτη. Ο διαχειριστής του συστήματος μπορεί να δημιουργήσει έναν χρήστη και στην συνέχεια από αυτόν τον χρήστη να δημιουργήσει έναν πελάτη. Ο χρήστης, κατά την στιγμή της καταχώρησης του, δημιουργείται και συνδέεται με μια νέα εγγραφή σε ένα παράλληλο σύστημα keycloak που χρησιμοποιείται για την αυθεντικοποίηση και την εξουσιοδότηση των χρηστών. Μέσω του Keycloak, οι χρήστες που καταχωρούνται στο σύστημα δημιουργούνται και συνδέονται αυτόματα με μια εγγραφή στο σύστημα ταυτοποίησης και εξουσιοδότησης των χρηστών, το Keycloak. Αυτό εξασφαλίζει όχι μόνο την εύκολη διαχείριση των χρηστών, αλλά και την αποτελεσματική προστασία των πληροφοριών και των προνομίων πρόσβασης στο σύστημα. Επιπλέον, το Keycloak προσφέρει επιπλέον λειτουργίες, όπως η δυνατότητα ενιαίας σύνδεσης (SSO), που βελτιώνουν την εμπειρία του χρήστη και μειώνουν τον κίνδυνο ασφαλείας. Με το Keycloak, η διαχείριση των χρηστών γίνεται αποδοτική και ασφαλής, ενώ η εξουσιοδότηση των προνομίων πρόσβασης γίνεται πιο ευέλικτη και αποτελεσματική.

Μικρο-υπηρεσία Οχημάτων (service 2): Η δεύτερη υπηρεσία αναλαμβάνει να διαχειριστεί τις οντότητες οι οποίες είναι σχετικές με τις πληροφορίες των οχημάτων που θέλουμε να αποθηκεύσουμε στην βάση μας. Σε αυτό το σημείο ο διαχειριστής του συστήματος μπορεί να δημιουργήσει τα μοντέλα και τις μάρκες των αυτοκινήτων που θα χρησιμοποιηθούν στο δεύτερο στάδιο αυτό της δημιουργίας και επεξεργασίας των παραγγελιών από τους χρήστες. Επιπλέον, ο διαχειριστής μπορεί να ορίσει τα χαρακτηριστικά και πακέτα (options) που σχετίζονται με κάθε μοντέλο αυτοκινήτου. Αυτά τα δεδομένα είναι ουσιώδη για την επόμενη διαδικασία, καθώς οι πελάτες θα χρησιμοποιούν αυτές τις πληροφορίες κατά την παραγγελία οχημάτων. Ένα από τα πλεονεκτήματα της υπηρεσίας διαχείρισης των οχημάτων είναι ότι αυτή η βάση δεδομένων είναι ανεξάρτητη από αυτήν των πελατών και των χρηστών. Αυτό σημαίνει ότι η διαχείριση των πληροφοριών σχετικά με τα μοντέλα και τις μάρκες των οχημάτων δεν επηρεάζεται από τις αλλαγές ή τις ενέργειες των πελατών ή των χρηστών του

συστήματος. Αυτό το πλεονέκτημα εξασφαλίζει τη σταθερότητα και τη συνέπεια των δεδομένων που σχετίζονται με τα οχήματα, ανεξάρτητα από τις δράσεις των χρηστών ή των πελατών. Έτσι, οι αλλαγές στις πληροφορίες των πελατών ή των χρηστών, όπως η προσθήκη νέων πελατών ή η αλλαγή στα δικαιώματα πρόσβασης των χρηστών, δεν επηρεάζουν την ακεραιότητα ή την αξιοπιστία των δεδομένων που αφορούν τα οχήματα. Αυτή η ανεξαρτησία επιτρέπει στην εφαρμογή να διατηρεί μια οργανωμένη δομή δεδομένων και να διασφαλίζει ότι οι πληροφορίες για τα οχήματα παραμένουν ακέραιες και αναλλοίωτες, ανεξαρτήτως των αλλαγών στο υπόλοιπο σύστημα.

Μικρο-υπηρεσία Παραγγελιών (service 3): Η τρίτη μικρο-υπηρεσία σχετίζεται με την διαχείριση των παραγγελιών. Ο χρήστης μπορεί να δημιουργήσει μια νέα παραγγελία, η οποία με την σειρά της αποθηκεύεται σε ένα σύστημα ουράς μηνυμάτων, το οποίο χρησιμοποιεί το Apache Kafka με σκοπό την διασύνδεση με άλλα συστήματα. Η χρήση του Kafka επιτρέπει την αξιόπιστη διαμετακόμιση των μηνυμάτων μεταξύ των διαφορετικών συστημάτων και υπηρεσιών. ο Apache Kafka ακολουθεί το μοντέλο δυνατότητας διάχυσης, όπου οι παραγωγοί μηνυμάτων (publishers – η μικρουπηρεσία παραγγελιών) - στέλνουν μηνύματα σε θέματα (topics), ενώ οι καταναλωτές μηνυμάτων (subscribers – το ERP) εγγράφονται σε αυτά τα θέματα για να λαμβάνουν τα μηνύματα που τους ενδιαφέρουν.

ERP: Συγκεκριμένα, το μήνυμα της παραγγελίας παραλαμβάνεται και επεξεργάζεται από το Business Central ERP, ένα ολοκληρωμένο σύστημα εργασιών που χρησιμοποιείται για τη διαχείριση της επιχείρησης. Κατά την επεξεργασία του μηνύματος δημιουργείται τόσο ο πελάτης όσο και η παραγγελία του που σε αυτό το στάδιο διαχειρίζεται ως προσφορά. Το σύστημα του ERP παραλαμβάνει αυτόματα το μήνυμα της προσφοράς μέσω μιας περιοδικής διεργασίας που εκτελείται στο παρασκήνιο ανά τακτά χρονικά διαστήματα. Ο χρήστης του ERP μπορεί να επιθεωρήσει τα δεδομένα της προσφοράς και εν συνεχεία να την μετατρέψει σε παραγγελία και τελικά να τιμολογήσει την παραγγελία αυτή. Κατά την τιμολόγηση ένα μήνυμα αποστέλλεται από το ERP αυτή τη φορά στην ουρά μηνυμάτων και παραλαμβάνεται από την υπηρεσία διαχείρισης παραγγελιών. Η αρχική παραγγελία ενημερώνεται και δημιουργούνται οι αντίστοιχες εγγραφές μέσα στην βάση δεδομένων.

Ο συνδετικός κρίκος όλων των παραπάνω υπηρεσιών, είναι το γραφικό περιβάλλον χρήστη το οποίο παρέχει την δυνατότητα τόσο στους διαχειριστές όσο και στους χρήστες να εκτελέσουν τις διάφορες ενέργειες τις οποίες παρέχει το σύστημα.

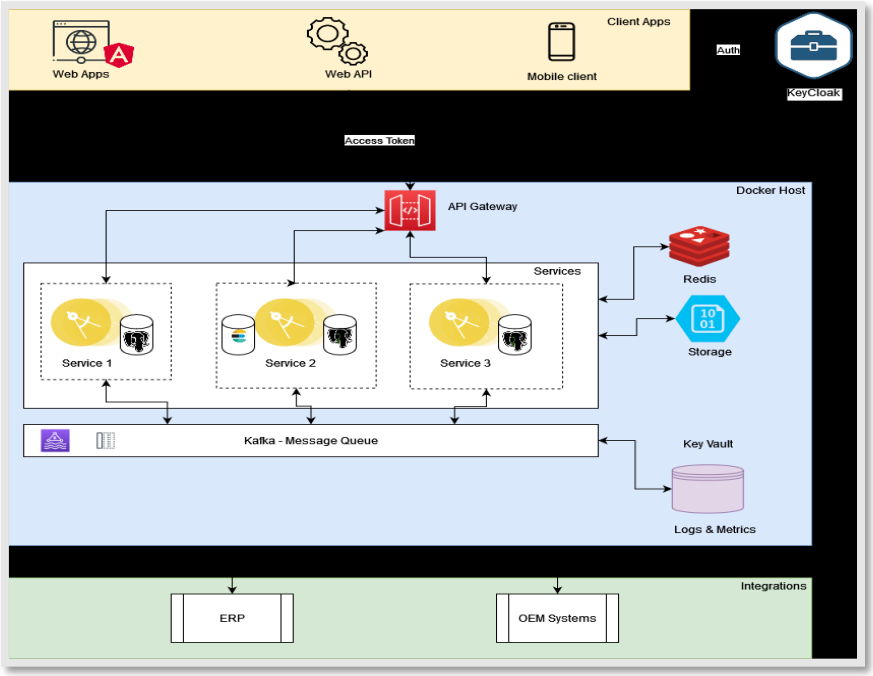
Το γραφικό περιβάλλον χρήστη, που υλοποιείται χρησιμοποιώντας την τεχνολογία Blazor, Χρησιμοποιώντας το Blazor, οι προγραμματιστές μπορούν να δημιουργήσουν δυναμικά γραφικά περιβάλλοντα χρήστη με εκπληκτική απόκριση και αποδοτικότητα.

Η ενσωμάτωση του Blazor στο γραφικό περιβάλλον του συστήματος επιτρέπει στους χρήστες να αλληλοεπιδρούν με τις υπηρεσίες με άνεση και ευκολία, ενώ παράλληλα προσφέρει στους διαχειριστές ευέλικτα εργαλεία για τη διαχείριση και την παρακολούθηση των λειτουργιών του συστήματος.

Έτσι μέσω του γραφικού περιβάλλοντος ο χρήστης μπορεί να δημιουργεί χρήστες και πελάτης (πρώτη μικρο-υπηρεσία), μπορούν να δημιουργήσουν και να διαχειριστούν τα μοντέλα και τις μάρκες των οχημάτων προς πώληση (δεύτερη μικρο-υπηρεσία), μπορούν να δημιουργήσουν και να επεξεργαστούν παραγγελίες (τρίτης μικρο-υπηρεσία).

Τέλος, είναι ουσιώδες τα συστήματα αυτά να εξασφαλίζουν ένα υψηλό επίπεδο ασφάλειας, προκειμένου να διασφαλίζεται ότι κανένας εξωτερικός ή κακόβουλος χρήστης δεν μπορεί να αποκτήσει πρόσβαση στα δεδομένα του συστήματος. Για τον λόγο αυτό, όλες οι επικοινωνίες είναι κρυπτογραφημένες χρησιμοποιώντας το πρωτόκολλο TLS (Transport Layer Security). Πέρα όμως από την ασφαλή επικοινωνία κάθε χρήστης αναλόγως με την βαθμίδα που έχει μέσα στο σύστημα θα πρέπει να μπορεί να εκτελεί συγκεκριμένες λειτουργίες. Αυτό επιτυγχάνεται με την χρήση μιας υπάρχουσας υπηρεσίας διαχείρισης ταυτότητας και πρόσβασης (Keycloak). Μέσω του Keycloak, οι χρήστες λαμβάνουν εξουσιοδότηση για προσδιορισμένες ενέργειες ανάλογα με τον ρόλο τους στο σύστημα, εξασφαλίζοντας τη σωστή διαχείριση των προσβάσεων και την προστασία των δεδομένων. Κάθε χρήστης θα λάβει ένα διαπιστευτήριο (access token) το με το οποίο θα αποφασίζεται από το σύστημα αν ο συγκεκριμένος χρήστης έχει πρόσβαση στις συγκεκριμένες μεθόδους του συστήματος.

Στο παρακάτω διάγραμμα απεικονίζονται όλα τα παραπάνω συστήματα καθώς και οι διασυνδέσεις μεταξύ τους:



Εικόνα 1 – Αρχιτεκτονική

Κεφάλαιο 2.1 - Τεχνολογίες

Στην ενότητα αυτή θα αναπτύξουμε την λειτουργικότητα και τη χρησιμότητα στην υλοποίηση των επιμέρους υπηρεσιών που χρησιμοποιήθηκαν. Κατά την ανάπτυξη της εφαρμογής, επιλέχθηκε να γίνει χρήση εξωτερικών μικρο-υπηρεσιών καθώς είναι μια στρατηγική που προσφέρει σημαντικά πλεονεκτήματα όσον αφορά την ευελιξία, την επεκτασιμότητα και την αποδοτικότητα της ανάπτυξης. Επιτρέπει την αξιοποίηση εξειδικευμένων λύσεων για συγκεκριμένες λειτουργίες της εφαρμογής, αποφεύγοντας την ανάγκη να ανάπτυξης αυτών των δυνατότητων από την αρχή. Αυτό μειώνει το χρόνο και το κόστος ανάπτυξης, ενώ παράλληλα αυξάνει την ποιότητα και την αξιοπιστία των υπηρεσιών που χρησιμοποιούνται. Οι εξωτερικές μικρο-υπηρεσίες προσφέρουν επίσης τη δυνατότητα κλιμάκωσης και βελτιωμένης απόδοσης, καθώς είναι σχεδιασμένες για να υποστηρίζουν μεγάλα φορτία και αυξημένη ζήτηση.

Η ταχύτητα ενισχύεται καθώς η ανάπτυξη εστιάζεται στις βασικές και πρωτότυπες λειτουργίες της εφαρμογής. Αυτό επιταχύνει τον κύκλο ανάπτυξης και τοποθετεί την εφαρμογή στην αγορά πιο γρήγορα.

Η μείωση του κόστους είναι σημαντική, καθώς η χρήση εξωτερικών υπηρεσιών μειώνει το κόστος ανάπτυξης και συντήρησης. Δεν απαιτούνται πόροι για τη διαχείριση των υποδομών και τη συνεχή βελτίωση των υπηρεσιών αυτών.

Η εξειδίκευση των εξωτερικών υπηρεσιών εξασφαλίζει ότι οι υπηρεσίες είναι βέλτιστες και ακολουθούν τις καλύτερες πρακτικές της βιομηχανίας, διασφαλίζοντας υψηλή ποιότητα και αξιοπιστία.

Η κλιμακωσιμότητα και η απόδοση βελτιώνονται καθώς οι εξωτερικές υπηρεσίες είναι σχεδιασμένες για υψηλή απόδοση και κλιμάκωση, επιτρέποντας στις εφαρμογές να χειρίζονται μεγάλους όγκους δεδομένων και υψηλές αιχμές ζήτησης χωρίς προβλήματα.

Η ασφάλεια ενισχύεται, καθώς πολλές εξωτερικές υπηρεσίες προσφέρουν ισχυρές δυνατότητες ασφαλείας, όπως κρυπτογράφηση δεδομένων, πιστοποίηση και εξουσιοδότηση, βοηθώντας στην προστασία των δεδομένων της εφαρμογής και των χρηστών της, ενώ διασφαλίζεται η συμμόρφωση με κανονισμούς.

Στην παρούσα εφαρμογή οι υπηρεσίες που χρησιμοποιήθηκαν είναι:

- **Kafka**: Το Apache Kafka είναι μια πλατφόρμα κατανεμημένου συστήματος ροής συμβάντων ανοικτού κώδικα, που χρησιμοποιείται για τη δημιουργία πραγματικού χρόνου αγωγών δεδομένων και εφαρμογών ροής. Αρχικά αναπτύχθηκε από το LinkedIn και αργότερα δόθηκε στη δημοσιότητα ως μέρος του Apache Software Foundation. Το Kafka παρέχει μια ανθεκτική σε βλάβες και επεκτάσιμη λύση για τη διαχείριση μεγάλων όγκων δεδομένων σε πραγματικό χρόνο.
- **Kong**: Το Kong είναι ένα API gateway ανοικτού κώδικα και ένα επίπεδο διαχείρισης των υπηρεσιών. Δρα ως κεντρική πλατφόρμα για τη διαχείριση, την ασφάλεια και την επέκταση των API (Διεπαφών Προγραμματισμού Εφαρμογών) σε διάφορες υπηρεσίες ή εφαρμογές. Το Kong σχεδιάστηκε για να απλοποιήσει την πολυπλοκότητα που σχετίζεται με τη διαχείριση των API, παρέχοντας χαρακτηριστικά όπως αυθεντικοποίηση, εξουσιοδότηση, περιορισμό ρυθμού, καταγραφή και άλλα.
- **Redis**: Είναι μια online προσωρινή-κρυφή μνήμη την οποία μπορούν να μοιραστούν οι διάφορες υπηρεσίες μεταξύ τους, χωρίς να κάνει η κάθε μια διαχείριση της μνήμης αυτής ξεχωριστά. Αυτό δίνει το πλεονέκτημα ότι εάν μια υπηρεσία χρειαστεί να γίνει επανεκκίνηση θα έχει στην διάθεσή της δεδομένα αυτής της μνήμης.
- **Keycloak**: Είναι η υπηρεσία που παρέχει στα προγράμματά μας την δυνατότητα για αυθεντικοποίηση και εξουσιοδότηση. Σαν υπηρεσία είναι επεκτάσιμη και διαθέτει ακόμα και υπηρεσίες αυθεντικοποίησης μέσω facebook/google etc.

- **ElasticSearch**: Το Elasticsearch είναι ένα ανοικτού κώδικα, κατανεμημένο σύστημα αναζήτησης που βασίζεται στο Apache Lucene. Σχεδιάστηκε για να παρέχει μια εμπειρία αναζήτησης κλιμακούμενη και υψηλής απόδοσης για διάφορους τύπους δεδομένων, από κείμενα έως δομημένα δεδομένα και πληροφορίες γεωχωρικής τοποθεσίας. Το Elasticsearch αποτελεί μέρος της Elastic Stack (προηγουμένως ELK Stack), που περιλαμβάνει εργαλεία όπως το Kibana, τα Beats και το Logstash.
- **KafkaConnect**: Το Kafka Connect είναι ένα σύνολο διεπαφών εντός του Apache Kafka που απλοποιεί την ενσωμάτωση του Kafka με εξωτερικά συστήματα. Παρέχει έναν κλιμακούμενο και αξιόπιστο τρόπο σύνδεσης του Kafka με διάφορες πηγές και απορροφητές δεδομένων, επιτρέποντας την άρρηκτη μεταφορά δεδομένων μεταξύ θεμάτων Kafka και εξωτερικών συστημάτων.
- **Kibana**: Το Kibana είναι ένα εργαλείο ανοικτού κώδικα για την απεικόνιση και την εξερεύνηση δεδομένων, σχεδιασμένο να λειτουργεί με την πλατφόρμα Elasticsearch. Αποτελεί μέρος της Elastic Stack (προηγουμένως ELK Stack), η οποία περιλαμβάνει επίσης το Elasticsearch και το Logstash. Το Kibana επιτρέπει στους χρήστες να αλληλοεπιδρούν με και να απεικονίζουν δεδομένα που αποθηκεύονται στο Elasticsearch, καθιστώντας πιο εύκολο τον κατανοητό και αναλυτικό χειρισμό μεγάλων όγκων δεδομένων.
- **PostgreSQL**: Είναι μια ισχυρή, ανοικτού κώδικα σχεσιακή βάση δεδομένων που χρησιμοποιείται για την αποθήκευση και διαχείριση δεδομένων. Υποστηρίζει προηγμένες λειτουργίες όπως πλήρεις συναλλαγές ACID, replication και ισχυρές δυνατότητες ευρετήριού, καθιστώντας την ιδανική για χρήση σε εφαρμογές που απαιτούν υψηλή αξιοπιστία και απόδοση.
- **Kafdrop**: Είναι ένα εργαλείο διαχείρισης και παρακολούθησης για το Kafka. Παρέχει γραφικό περιβάλλον για την οπτικοποίηση και τη διαχείριση των topics, των partitions και των consumers, κάνοντας πιο εύκολη την παρακολούθηση και την ανάλυση των δεδομένων που ρέουν μέσω του Kafka.
- **Zookeeper**: Χρησιμοποιείται για τον συντονισμό και τη διαχείριση κατανεμημένων συστημάτων. Είναι κρίσιμο για τη λειτουργία του Kafka, παρέχοντας υπηρεσίες όπως configuration management, synchronization και naming registry, που είναι απαραίτητες για την αξιόπιστη λειτουργία κατανεμημένων εφαρμογών.

Στις επόμενες ενότητες παρουσιάζεται η χρήση και οι σχέσεις των παραπάνω υπηρεσιών.

Κεφάλαιο 2.1.1 - Διασύνδεση/επικοινωνία υπηρεσιών

Η ανάγκη για σύγχρονη και ασύγχρονη επικοινωνία μεταξύ των μικρο-υπηρεσιών προκύπτει από τις διάφορες απαιτήσεις και σενάρια χρήσης των εφαρμογών όπως αυτά περιγράφονται στην παρουσίαση του συστήματος. Οι δυνατότητες αυτές επιτρέπουν τη δημιουργία ευέλικτων και αποδοτικών συστημάτων, επιτρέποντας στις υπηρεσίες να επικοινωνούν μεταξύ τους με τρόπους που εξυπηρετούν τις ανάγκες των εφαρμογών και των επιχειρηματικών σεναρίων. Κατάλληλη χρήση και συνδυασμός αυτών των μεθόδων επικοινωνίας βοηθά στη δημιουργία ανθεκτικών και εύκαμπτων συστημάτων.

Στο κεφάλαιο αυτό θα περιγράψουμε τις τεχνολογίες που χρησιμοποιούνται για την κάλυψη των αναγκών και για τους δύο τύπους.

Σύγχρονη Επικοινωνία: Σε περιπτώσεις όπου οι μικρο-υπηρεσίες πρέπει να αλληλοεπιδράσουν μεταξύ τους και να ανταλλάσσουν δεδομένα αμέσως ή να περιμένουν την άμεση απάντηση από την άλλη υπηρεσία. Για παράδειγμα, κατά τη δημιουργία παραγγελίας μέσω του γραφικού περιβάλλοντος (blazor) καλείται η υπηρεσία των χρηστών/πελατών για την ανάκτηση των στοιχείων του πελάτη και περιμένει μια άμεση απάντηση για να συνεχίσει την εκτέλεσή της και να δημιουργηθεί η παραγγελία. Για την διεπαφή του χρήστη (frontend) χρησιμοποιούμε σύγχρονα μηνύματα επικοινωνίας με το πρωτόκολλο gRPC.

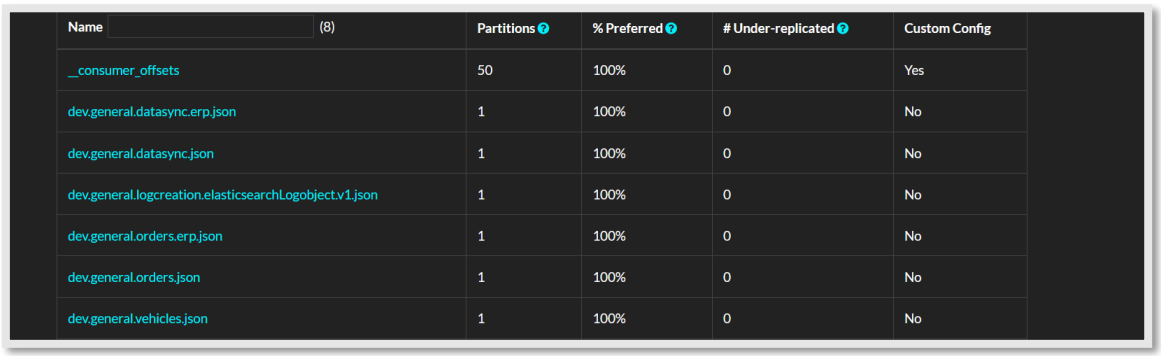
Για την σύγχρονη επικοινωνία έχουμε χρησιμοποιήσει ένα reverse proxy (kong) ο οποίος αναλαμβάνει να κάνει ανακατεύθυνση των μηνυμάτων προς την αντίστοιχη υπηρεσία (service).

Η ανακατεύθυνση αυτή γίνεται με βάση το ενοποιημένο αναγνωριστικό πόρου του σημείου πρόσβασης του οποίου καλείται. Κάθε service κατά την εκκίνησή του δημιουργεί μια εγγραφή, στον reverse proxy χρησιμοποιώντας μια Διεπαφή Προγραμματισμού Εφαρμογών (API), για κάθε controller που έχουμε δημιουργήσει.

Στην συνέχεια, όταν κάποιο αίτημα (request) σταλθεί στον reverse proxy αυτός είναι υπεύθυνος ώστε να αποστείλει το μήνυμα (dispatch) στην αντίστοιχη υπηρεσία που το εξυπηρετεί.

Ασύγχρονη Επικοινωνία: Σε περιπτώσεις όπου η επικοινωνία μεταξύ των μικρο-υπηρεσιών μπορεί να γίνει ανεξάρτητα από την άμεση απάντηση. Οι υπηρεσίες μπορούν να στείλουν μηνύματα ή αιτήσεις χωρίς να περιμένουν άμεση απάντηση. Αυτό μπορεί να είναι χρήσιμο για διαδικασίες που απαιτούν χρόνο, όπως η επεξεργασία μεγάλου όγκου δεδομένων ή η εκτέλεση διαδικασιών που δεν είναι αμέσως απαιτούμενες για την αμεσότητα της απάντησης στο αρχικό αίτημα. Για παράδειγμα, κατά την οριστικοποίηση μιας παραγγελίας αποστέλλεται ένα μήνυμα σε μια ουρά

χωρίς να περιμένει την απάντηση για την επεξεργασία της από το ERP. Οι διάφορες υπηρεσίες επικοινωνούν μεταξύ τους, ασύγχρονα, με την χρήση ουράς μηνυμάτων (Kafka)



Name (8)	Partitions	% Preferred	# Under-replicated	Custom Config
__consumer_offsets	50	100%	0	Yes
dev.general.datasync.erp.json	1	100%	0	No
dev.general.datasync.json	1	100%	0	No
dev.general.logcreation.elasticsearchLogobject.v1.json	1	100%	0	No
dev.general.orders.erp.json	1	100%	0	No
dev.general.orders.json	1	100%	0	No
dev.general.vehicles.json	1	100%	0	No

Εικόνα 2 - Θέματα Ουράς Μηνυμάτων

Για την επικοινωνία με την ουρά μηνυμάτων (Kafka) έχουμε δύο διαφορετικά μέρη που επικοινωνούν μεταξύ τους, τον Εκδότη μηνυμάτων (Publisher) και τον Καταναλωτή αυτών (Consumer). Κάθε εκδότης δημιουργεί ένα μήνυμα μέσα στην ουρά σε ένα συγκεκριμένο θέμα (Topic). Στην συνέχεια ένας ή περισσότεροι καταναλωτές εγγράφονται στο συγκεκριμένο θέμα και κάθε φορά που υπάρχει ένα νέο μήνυμα στην ουρά το λαμβάνουν και το επεξεργάζονται.

Κεφάλαιο 2.1.2 - Πρωτόκολλα Επικοινωνίας

Μια «Διεπαφή Προγραμματισμού Εφαρμογών (API)» αποτελείται από ένα σύνολο κανόνων και πρωτοκόλλων τα οποία επιτρέπουν σε ένα σύνολο εφαρμογών να αλληλοεπιδρούν και να ανταλλάσσουν δεδομένα.

Υπάρχουν πολλές διαφορετικές αρχιτεκτονικές για την δημιουργία μιας τέτοια διεπαφής και κάθε μία από αυτές έχει τα δικά της πλεονεκτήματα και μειονεκτήματα. Δύο από αυτές τις αρχιτεκτονικές είναι το REST και το gRPC. Το REST ορίζει ένα σύνολο από κανόνες για την δημιουργία εφαρμογών και βασίζεται στην απλότητα, την μη διατήρηση κατάστασης (stateless) καθώς και στην επικοινωνία με βάση τους πόρους (resources). Από την άλλη, το gRPC, είναι ένα σύστημα υψηλής απόδοσης για απομακρυσμένη κλήση διαδικασιών (RPC) το οποίο χρησιμοποιεί τα Protocol Buffers και το HTTP/2 για την αποτελεσματική επικοινωνία σε κατανεμημένα συστήματα.

Το REST είναι ο πιο διαδεδομένος τρόπος για την δημιουργία «Διεπαφών Προγραμματισμού Εφαρμογών» αλλά, με την σειρά του, το gRPC έχει σημειώσει σημαντική άνοδο λόγο της υποστήριξής του για ροές δεδομένων διπλή κατεύθυνσης (bidirectional streaming), της ισχυρούς αποτύπωσης των τύπων δεδομένων (strong data typing) και της δημιουργίας κώδικα ανεξαρτήτου γλώσσας προγραμματισμού.

Το REST (Representation State Transfer), είναι ο πιο διαδεδομένος τρόπος για την δημιουργία «Διεπαφών Προγραμματισμού Εφαρμογών». Σε μία τέτοια εφαρμογή, οι πόροι προς διαχείριση , αναγνωρίζονται από το «Ενιαίο Αναγνωριστικό Πόρων» (Uniform Resource Identifier – URI) και οι διάφορες λειτουργίες προς τους πόρους αυτούς γίνονται χρησιμοποιώντας ένα σύνολο μεθόδων του HTTP πρωτοκόλλου. Οι πόροι αναπαρίστανται σε μορφή JSON, η οποία μεταφέρεται μεταξύ του πελάτη και του διακομιστή στα σώματα των αιτημάτων και των απαντήσεων του HTTP.

Το gRPC είναι ένα ανοιχτού κώδικα, υψηλής απόδοσης πλαίσιο το οποίο παρέχει την δυνατότητα σε κατανεμημένα συστήματα να επικοινωνούν μεταξύ τους με τον βέλτιστο τρόπο. Παρέχει μια υλοποίηση του RPC πρωτοκόλλου, το οποίο δίνει την δυνατότητα σε εφαρμογές να καλούν απομακρυσμένες μεθόδους σαν να ήταν μέρος της υλοποίησής τους.

Το πρωτόκολλο αυτό, δημιουργήθηκε από την Google το 2015 και περιλαμβάνει πολλές βελτιστοποιήσεις προς τον τρόπο με τον οποίο γίνονται οι απομακρυσμένες κλήσεις. Για παράδειγμα, κάνει χρήση των Protocol Buffers (Protobuf) το οποίο παρέχει ισχυρή αποτύπωση στους τύπους δεδομένων, σειριοποίηση δεδομένων και παραγωγή κώδικα σε ποικίλες γλώσσες προγραμματισμού.

Το REST (Representational State Transfer) περιγράφει μια αρχιτεκτονική πελάτη-εξυπηρετήτη, κατά την οποία τα δεδομένα μεταφέρονται στην εφαρμογή πελάτη δομημένα μέσω μηνυμάτων JSON ή ακόμα και xml.

Ένα REST σύστημα, διαθέτει τις παρακάτω ιδιότητες:

- **Προγραμματιστική διεπαφή (API)** : Η διεπαφή αυτή διαθέτει ορισμένους πόρους (resources) προς του καταναλωτές (clients).
- **Ανεξαρτησία πελάτη/εξυπηρετητή**: Ο πελάτης και ο εξυπηρετητής δρουν αυτόνομα. Η εφαρμογή πελάτη γνωρίζει μόνο συγκεκριμένα αναγνωριστικά πόρων (URIs) τα οποία διατίθονται απο την εφαρμογή του εξυπηρετητή.
- **Stateless**: Η εφαρμογή του εξυπηρετητή δεν αποθηκεύει την κατάσταση ενός αιτήματος από την εφαρμογή πελάτη. Κάθε αίτημα προς την εφαρμογή του εξυπηρετητή επιδρά σε μόνο ένα πόρο και κάθε επιμέρους αίτημα είναι αυτοτελές.
- **Πολυεπίπεδο (Layered)** : Η αρχιτεκτονική αυτή είναι πολυεπίπεδη, το οποίο της δίνει την δυνατότητα τα διαφορετικά μέρη της να υλοποιούνται από διαφορετικές υλοποιήσεις εξυπηρετητών.
- **Κώδικας κατά απαίτηση (Code on Demand)** : Δίνει την δυνατότητα στην εφαρμογή του εξυπηρετητή να στείλει κομμάτια κώδικα στην εφαρμογή πελάτη, ώστε η πρώτη να καθορίσει τον τρόπο και την ροή εκτέλεσης.

Το gRPC χρησιμοποιεί τη μορφοποίηση μηνυμάτων Protobuf αντί των JSON που συνηθίζονται στα REST APIs και RPC APIs. Η χρήση του Protobuf βοηθά στην αντιμετώπιση θεμάτων ταχύτητας και όγκου δεδομένων, προσφέροντας μεγαλύτερη αποδοτικότητα στην αποστολή μηνυμάτων. Είναι ανεξάρτητο πλατφόρμας και γλώσσας όπως το JSON, αλλά επικοινωνεί μεταξύ τους μέσω δυαδικής σειριοποίησης.

Το gRPC χρησιμοποιεί το πρωτόκολλο HTTP/2 αντί του HTTP/1.1 που χρησιμοποιείται συνήθως στα συστήματα REST. Το REST API χρησιμοποιεί συχνότερα το πρωτόκολλο HTTP/1 αντί του HTTP/2 για διάφορους λόγους, συμπεριλαμβανομένης της απλότητας, της παλαιότητας (Το HTTP/1 είναι το παλαιότερο πρωτόκολλο και έχει χρησιμοποιηθεί ευρέως για μεγάλο χρονικό διάστημα πριν από την εμφάνιση του HTTP/2), της προσαρμοστικότητας και της συμβατότητας με τις υπάρχουσες υποδομές. Η χρήση του HTTP-2 δεν αποκλείεται από το REST API αλλά δεν είναι σύνηθες καθότι είναι, αντίθετα το gRPC ως νεότερο ενσωμάτωσε εξ’ αρχής το νέο πρωτόκολλο παλαιότερο (όπως αναφέρεται στη Wikipedia: REST: 2000, HTTP /2: 2015, gRPC: 2015). Το HTTP/2 είναι πιο γρήγορο, αποδοτικό και μειώνει την καθυστέρηση του δικτύου μέσω του multiplexing. Το gRPC προσφέρει τρεις τύπους ροών: εξυπηρετητή, πελάτη και αμφίδρομες ροές μηνυμάτων.

Σε αντίθεση με τα REST APIs που συνήθως χρησιμοποιούν εργαλεία τρίτων, το gRPC παρέχει ενσωματωμένη γεννήτρια κώδικα (Protoc compiler). Αυτό επιτρέπει τη δημιουργία κώδικα σε ποικίλες γλώσσες και χρησιμοποιείται σε πολύγλωσσα περιβάλλοντα, όπου οι υπηρεσίες τρέχουν σε ξεχωριστές πλατφόρμες και προγραμματίζονται σε διαφορετικές γλώσσες προγραμματισμού.

Τα κύρια χαρακτηριστικά του είναι τα εξής:

- Ανεξάρτητο από πλατφόρμες και γλώσσες προγραμματισμού.
- Χρησιμοποιεί δυαδική απεικόνιση για την αναπαράσταση των μηνυμάτων.
- Αυξάνει την ταχύτητα μετάδοσης των μηνυμάτων λόγω της δομής των μηνυμάτων.
- Κάνει χρήση του HTTP/2 πρωτοκόλλου.
- Παρέχει μηχανισμό για διαχείριση των διαφορετικών εκδόσεων των μηνυμάτων.

Για εφαρμογές, μεγάλης κλίμακας, οι οποίες καλούνται να εξυπηρετούν πολλά αιτήματα σε σύντομο χρονικό διάστημα και οι οποίες είναι δημιουργημένες, βασιζόμενες πάνω στην αρχιτεκτονική υπηρεσιοστρεφούς εφαρμογής η χρήση gRPC δείχνει να είναι μια καλύτερη λύση σε σχέση με την χρήση του REST πρωτοκόλλου.

	Goal	REST (HTTP/JSON)	gRPC
COMPATIBILITY	Single source of truth	✗	✔
	Multi-platform + languages built in	✗	✔
	Handle non-breaking changes.	✗	✔
PERFORMANCE	Network: connection handling	Manual, 1 per call ✗	Built-in, Multi per conn ✔
	Speed: Transmission of data	Human-readable Text ✗	Binary ✔
	CPU: Improved resource usage	✗	✔
MAINTENANCE	Tracing	Manual ✗	Easy to plug in ✔
	Logging	Manual ✗	Easy to plug in ✔
	Monitoring	Manual ✗	Easy to plug in ✔

Εικόνα 3 - Σύγκριση gRPC – REST

Σε γενικές γραμμές οι κύριες διαφορές οι οποίες καθιστούν το gRPC μια καλύτερη επιλογή για την ανάπτυξη της τρέχουσας εφαρμογής (όπως περιγράφονται και στην βιβλιογραφία, βλ. gRPCvsREsT), είναι ότι το gRPC παρέχει την δυνατότητα με την ίδια δομή μηνύματος να δημιουργηθούν σε διάφορες γλώσσες προγραμματισμού εφαρμογές, τόσο πελάτη όσο και εξυπηρετητή.

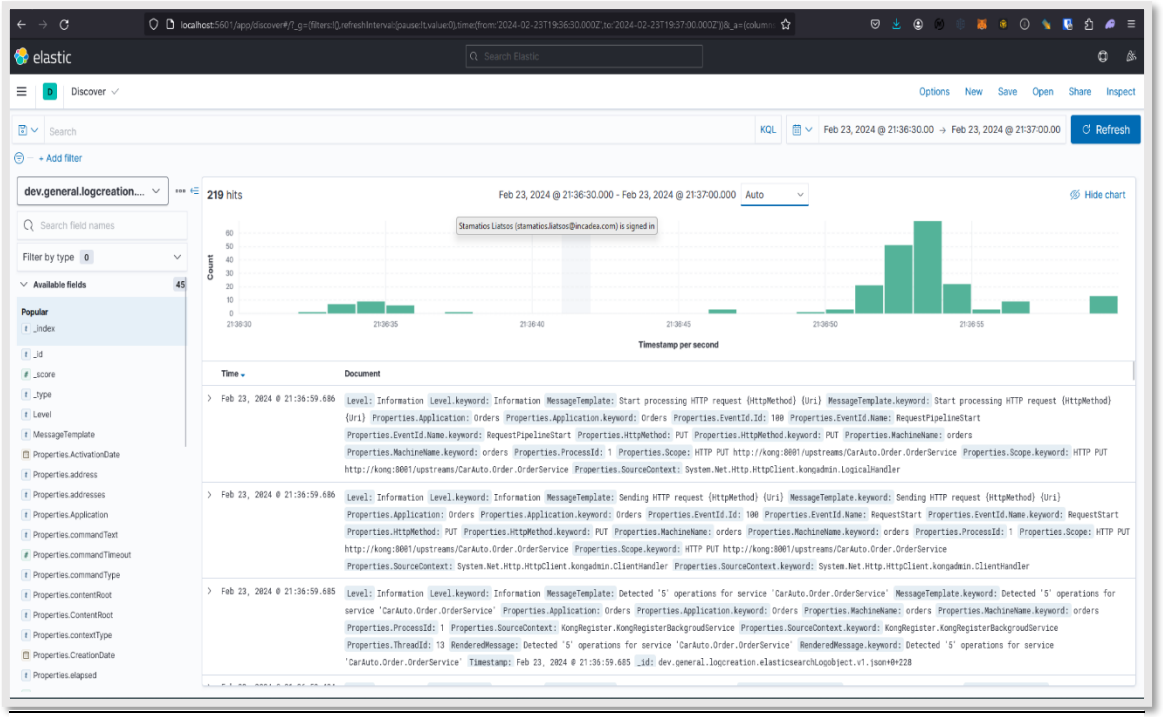
Παρέχει την δυνατότητα παράλληλης διαχείρισης των μηνυμάτων, λόγω της χρήσης του HTTP/2 καθώς και την ταχύτατη μεταφορά των μηνυμάτων λόγω της δυαδικής αναπαράστασης των μηνυμάτων. Σαν αντίκτυπο, η δυαδική αναπαράσταση, προσφέρει και την καλύτερη διαχείριση των πόρων του συστήματος.

Κεφάλαιο 2.1.2 - Καταγραφή Συμβάντων

Η συνήθης τακτική χρήσης του ELK (Elasticsearch, Logstash, Kibana) σε συνδυασμό με το Kibana για την καταγραφή συμβάντων και την αξιοποίησή τους από τους χρήστες για τον εντοπισμό και την αντιμετώπιση προβλημάτων.

Συλλογή και μεταφορά δεδομένων: Αρχικά, τα δεδομένα καταγράφονται από διάφορες πηγές στο σύστημα, όπως αρχεία καταγραφής (logs), μηνύματα συμβάντων (events), μετρήσεις από συστήματα μέτρησης απόδοσης (performance metrics) και άλλα. Τα δεδομένα μεταφέρονται από το το Kafka Connect στο Elasticsearch για αποθήκευση και ευέλικτη αναζήτηση. Με τον συνδυασμό του Kafka και του Kafka Connect, μπορούμε να συλλέγουμε μηνύματα από διάφορες πηγές και να τα μεταφέρουμε στο Elasticsearch για αποθήκευση και ανάλυση.

Ανάλυση και Αντιμετώπισης: Στην συνέχεια χρησιμοποιώντας το Kibana μπορούμε να ανατρέξουμε στα καταγεγραμμένα συμβάντα και να έχουμε μια πιο εύχρηστη αποτύπωση αυτών. Το Kibana είναι ένα ισχυρό εργαλείο οπτικοποίησης και ανάλυσης δεδομένων που συνήθως χρησιμοποιείται σε συνδυασμό με το Elasticsearch για τη διαχείριση μεγάλων όγκων δεδομένων και την παρουσίαση επιχειρησιακών πληροφοριών. Ανάλογα με το πλαίσιο χρήσης, το Kibana προσφέρει διάφορες λειτουργίες και δυνατότητες. Οι χρήστες μπορούν να δημιουργήσουν πίνακες ελέγχου, γραφήματα, και αναφορές για να εξετάσουν την απόδοση, την κατάσταση και τις τάσεις του συστήματος. Χρησιμοποιώντας τις δυνατότητες αναζήτησης και φιλτραρίσματος, οι χρήστες μπορούν να εντοπίσουν πιθανά προβλήματα ή ανωμαλίες στα δεδομένα, όπως συμβάντα σφαλμάτων, αυξημένη κατανάλωση πόρων ή αποτυχίες συστήματος. Με βάση τις πληροφορίες που παρέχονται, οι χρήστες μπορούν να λάβουν μέτρα για την αντιμετώπιση των εντοπισμένων προβλημάτων, όπως η επισκευή σφαλμάτων, η βελτιστοποίηση απόδοσης ή η επαναρύθμιση των συστημάτων.



Εικόνα 4 - Kibana

Κεφάλαιο 2.1.3 - Διαχείριση Δεδομένων

Τα δεδομένα τα οποία ανταλλάσσονται μεταξύ των συστημάτων, αποθηκεύονται μέσα σε μία βάση δεδομένων (postgres). Κάθε υπηρεσία διαθέτει την δική της βάση και σχήμα δεδομένων, έτσι ώστε να διασφαλίσουμε την ακεραιότητα τους σε περίπτωση που κάποια από τις βάσεις καταστεί μη λειτουργική για κάποιο χρονικό διάστημα.

Στο κεφάλαιο αυτό παραθέτουμε το σχήμα της βάσης δεδομένων που χρησιμοποιεί η κάθε υπηρεσία για να αποθηκεύσει τα δεδομένα της. Η απεικόνιση βασίζεται στο μοντέλο οντοτήτων συσχετίσεων, παρουσιάζοντας τις οντότητες, τις σχέσεις τους και τα πεδία που περιέχουν. Αναλυτικότερα, παρουσιάζουμε τις δομές των πινάκων και τις μεταξύ τους συνδέσεις, όπως αυτές έχουν διαμορφωθεί για να εξυπηρετήσουν τις ανάγκες των υπηρεσιών διαχείρισης χρηστών και πελατών, οχημάτων και παραγγελιών.

Κατά το σχεδιασμό των βάσεων λήφθηκαν υπόψιν οι βασικές αρχές σχεδίασης των βάσεων δεδομένων που περιλαμβάνουν:

Κανονικοποίηση, η οποία οργανώνει τα δεδομένα για την ελαχιστοποίηση της πλεονασμού και την αποφυγή ανακολουθιών, διασφαλίζοντας ότι κάθε δεδομένο

αποθηκεύεται μόνο μία φορά.
Ακεραιότητα δεδομένων, που εξασφαλίζει την ακρίβεια και συνέπεια των δεδομένων μέσω περιορισμών όπως πρωτεύοντα και εξωτερικά κλειδιά, μοναδικότητα και έλεγχοι.
Επεκτασιμότητα, που επιτρέπει στη βάση δεδομένων να χειριστεί την αύξηση του όγκου των δεδομένων και των χρηστών χωρίς να επηρεάζεται η απόδοση, μέσω της κατανομής δεδομένων και της χρήσης κατανεμημένων συστημάτων αποθήκευσης.
Απόδοση, που βελτιστοποιεί τα ερωτήματα και τις δομές δεδομένων για γρήγορη και αποδοτική ανάκτηση δεδομένων, χρησιμοποιώντας ευρετήρια και βελτιστοποίηση συναλλαγών.
Ασφάλεια, που προστατεύει τα δεδομένα από μη εξουσιοδοτημένη πρόσβαση και κακόβουλες επιθέσεις μέσω ελέγχων πρόσβασης, κρυπτογράφησης δεδομένων και αναθεώρησης πολιτικών ασφάλειας.

Αρχιτεκτονική μικροϋπηρεσιών, που σχεδιάζει τη βάση δεδομένων με τρόπο που να εξυπηρετεί τις ανεξάρτητες μικροϋπηρεσίες, επιτρέποντας σε κάθε υπηρεσία να έχει τη δική της βάση δεδομένων ή σχήμα.

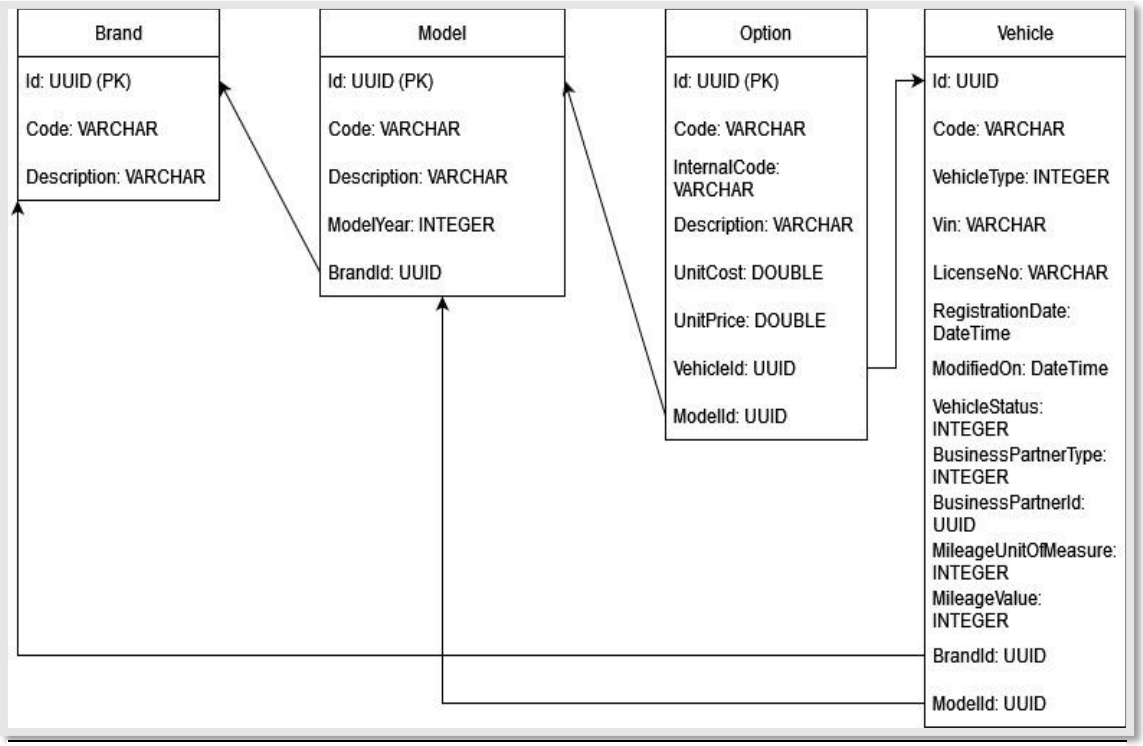
Τέλος, η **αξιοπιστία** διασφαλίζει ότι τα δεδομένα είναι πάντα διαθέσιμα και ανακτήσιμα, ακόμη και σε περιπτώσεις αποτυχίας συστήματος, μέσω αντιγράφων ασφαλείας, αποκατάστασης καταστροφών και μηχανισμών αναπαραγωγής. Αυτές οι αρχές συνδυάζονται για να δημιουργήσουν μια βάση δεδομένων που είναι σταθερή, ευέλικτη και ικανή να ανταποκριθεί στις απαιτήσεις του συστήματος και των χρηστών του. Υπάρχει μια σύνδεση μεταξύ των δεδομένων των διαφορετικών βάσεων με στόχο να μπορούν να ανακτηθούν τα συσχετιζόμενα δεδομένα και να εμφανιστούν στην γραφική διεπαφή των χρηστών.

Οι διαφορετικές μικρο-υπηρεσίες του συστήματος διαθέτουν ανεξάρτητες βάσεις δεδομένων, όπου οι σχέσεις μεταξύ των δεδομένων είναι θεωρητικές και δεν υπάρχουν τεχνικοί περιορισμοί όπως τα Foreign Keys. Παρά την απουσία αυτών των τεχνικών περιορισμών, οι συσχετίσεις μεταξύ των δεδομένων διατηρούνται και χρησιμοποιούνται από το γραφικό περιβάλλον της εφαρμογής. Οι σχέσεις των δεδομένων μεταξύ των βάσεων δεδομένων των μικρο-υπηρεσιών είναι loosely coupled, δηλαδή οι υπηρεσίες είναι χαλαρά συνδεδεμένες μεταξύ τους. Αυτό σημαίνει ότι κάθε υπηρεσία μπορεί να λειτουργεί και να αναπτύσσεται ανεξάρτητα από τις άλλες, επιτρέποντας ευκολότερη συντήρηση και κλιμάκωση. Το γραφικό περιβάλλον, το οποίο χρησιμοποιεί την τεχνολογία Blazor, αναλαμβάνει να αντλεί δεδομένα από τις διαφορετικές βάσεις δεδομένων και να τα συνδυάζει για να προσφέρει μια ενιαία και ολοκληρωμένη εμπειρία στους χρήστες.

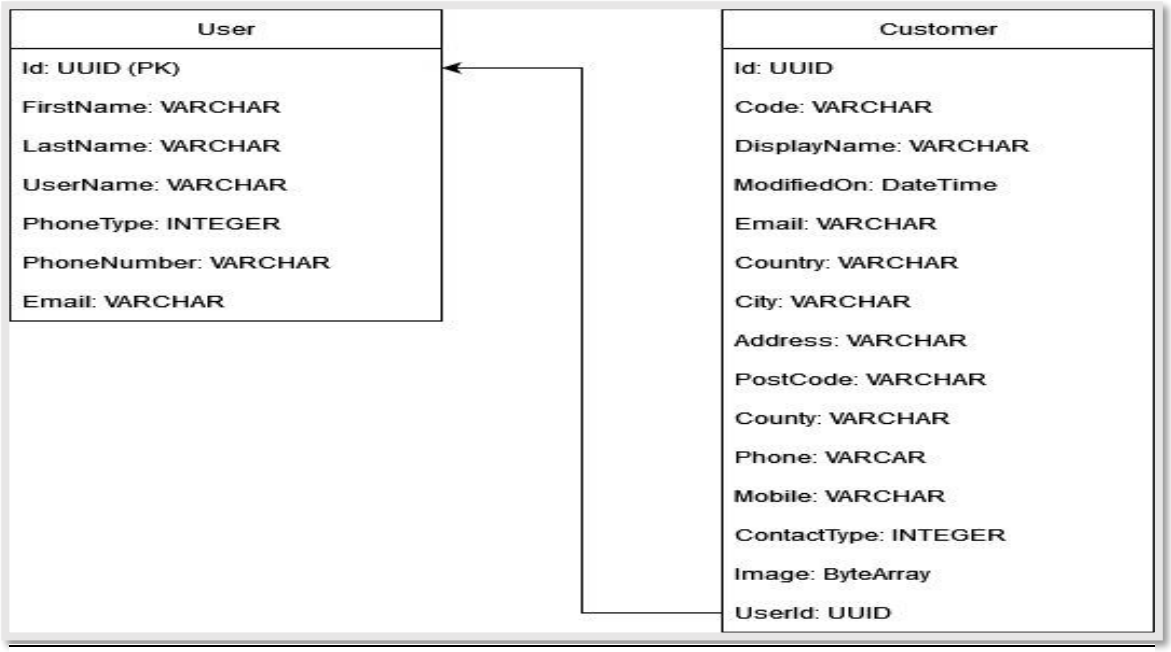
Για παράδειγμα, η υπηρεσία διαχείρισης χρηστών και πελατών μπορεί να έχει δεδομένα για χρήστες και πελάτες, ενώ η υπηρεσία διαχείρισης οχημάτων έχει δεδομένα για τα μοντέλα και τις μάρκες των αυτοκινήτων. Οι συσχετίσεις μεταξύ αυτών των δεδομένων, όπως ποιος πελάτης έχει παραγγείλει ποιο όχημα, διατηρούνται στο γραφικό περιβάλλον χωρίς να υπάρχουν άμεσοι τεχνικοί περιορισμοί στις βάσεις δεδομένων. Αυτή η χαλαρή σύνδεση επιτρέπει στο σύστημα να παραμένει ευέλικτο και επεκτάσιμο, ενώ παράλληλα εξασφαλίζει την συνοχή και την ολοκληρωμένη προβολή των δεδομένων.

Ένα σημαντικό πλεονέκτημα της χρήσης μικρο-υπηρεσιών είναι η δυνατότητα αποφυγής απώλειας δεδομένων σε περίπτωση αποτυχίας κάποιας υπηρεσίας. Επειδή οι υπηρεσίες είναι χαλαρά συνδεδεμένες, η αποτυχία μιας υπηρεσίας δεν επηρεάζει άμεσα τις άλλες υπηρεσίες. Κάθε υπηρεσία αποθηκεύει τα δεδομένα της ανεξάρτητα, και έτσι, σε περίπτωση αποτυχίας, τα δεδομένα παραμένουν ασφαλή και ακέραια στις βάσεις δεδομένων των άλλων υπηρεσιών. Επιπλέον, οι διεργασίες επικοινωνίας μεταξύ

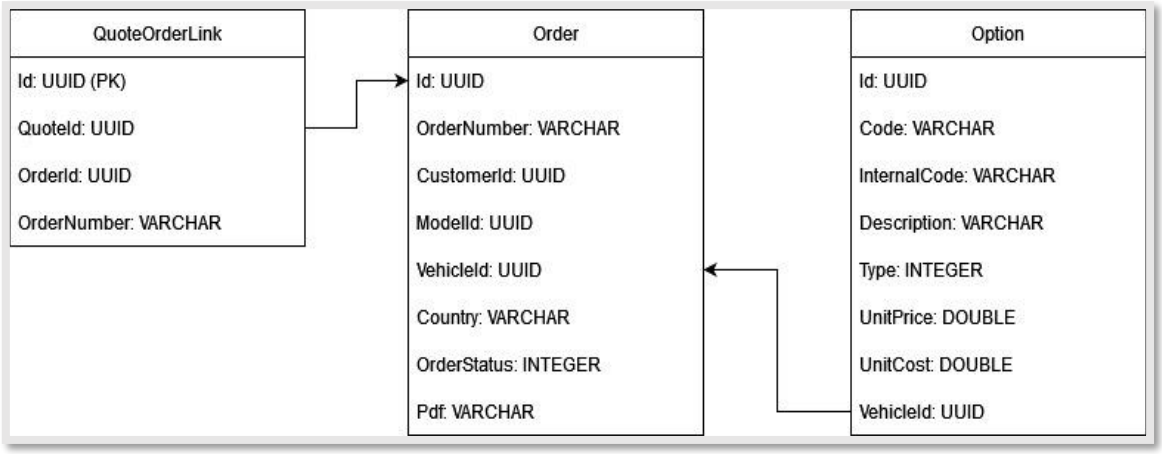
των υπηρεσιών μπορούν να επανεκκινήσουν ή να επαναληφθούν χωρίς να χάνονται δεδομένα, εξασφαλίζοντας έτσι την αξιοπιστία και την ανθεκτικότητα του συστήματος.



Εικόνα 5 -Διάγραμμα Υπηρεσίας Οχημάτων



Εικόνα 5 - Διάγραμμα Υπηρεσίας Χρηστών/Πελάτων



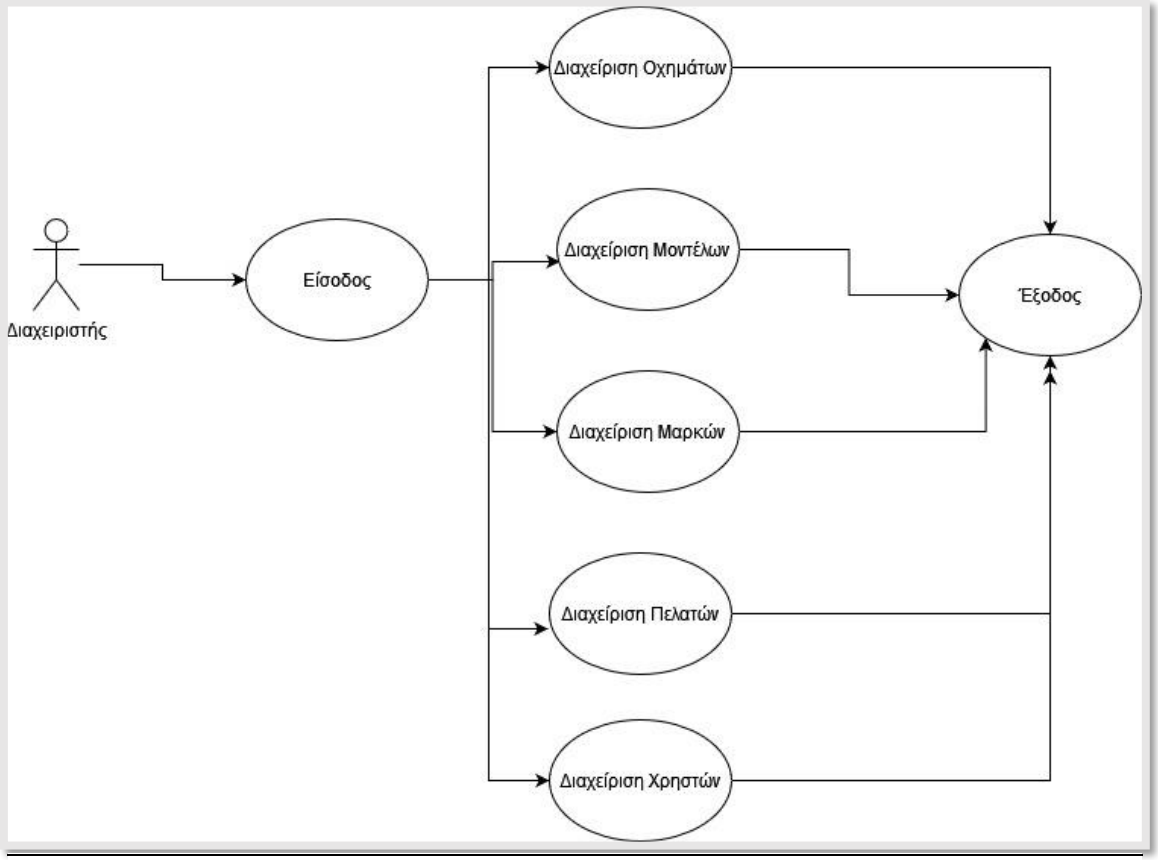
Εικόνα 6 - Διάγραμμα Υπηρεσίας Παραγγελιών

Κεφάλαιο 3 - Παρουσίαση Συστήματος

Στο κεφάλαιο αυτό θα αναφερθούμε στην περιγραφή των λειτουργιών του συστήματος που καλείται να εκτελέσει ώστε να είναι χρήσιμο και ορθά κατασκευασμένο. Όπως ήδη έχουμε αναφέρει το σύστημα χωρίζεται σε τρία επιμέρους λειτουργικά κομμάτια, στην σελίδα που χειρίζεται ο διαχειριστής του συστήματος, την σελίδα με την οποία αλληλοεπιδρούν οι χρήστες και το ERP στο οποίο γίνεται η ολοκλήρωση και καταχώρηση των παραγγελιών.

Κεφάλαιο 3.1 - Υποσύστημα Διαχειριστή

Στο υποσύστημα του Διαχειριστή, ο χρήστης θα έχει την δυνατότητα να δημιουργεί, να ανανεώνει και να διαγράφει νέους χρήστες, πελάτες, μοντέλα και μάρκες οχημάτων καθώς και τα ίδια τα οχήματα.

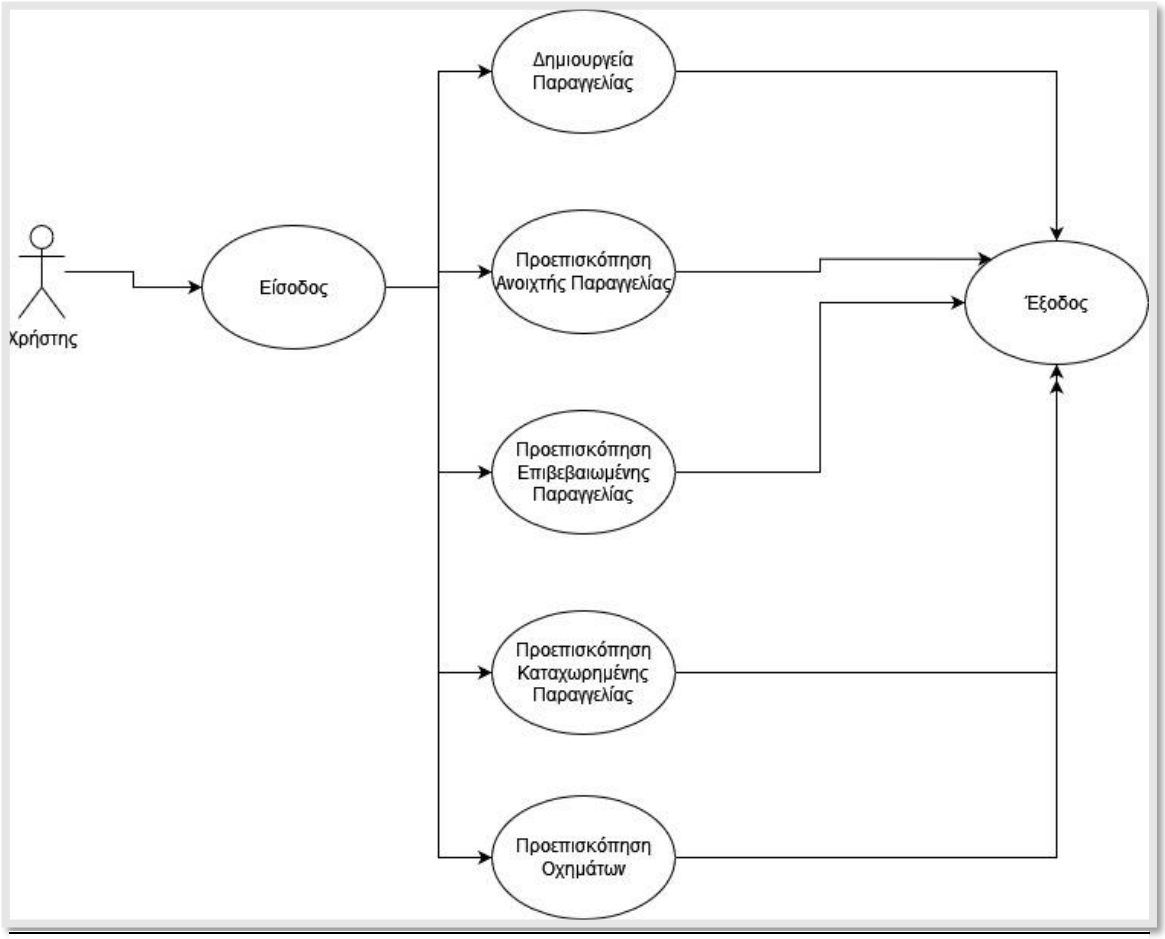


Εικόνα 7 - Διάγραμμα Περιπτώσεων Χρήσης Διαχειριστή

Στο παραπάνω διάγραμμα φαίνονται οι δυνατότητες που προσφέρει η σελίδα στον διαχειριστή του συστήματος. Ο διαχειριστής μπορεί να δημιουργήσει, να τροποποιήσει ή να διαγράψει όποια από τις οντότητες διαχειρίζεται το σύστημά. Η δημιουργία νέων

χρηστών δίνει την δυνατότητα σε νέους χρήστες να εισέλθουν στο σύστημα αλλά μόνο ένας χρήστης με δικαιώματα διαχειριστή μπορεί να δημιουργηθεί και αυτός δημιουργείται αυτόματα κατά την αρχικοποίηση του συστήματος. Πολλές από τις οντότητες είναι συνδεδεμένες μεταξύ τους, οπότε η καταχώρηση των εγγραφών θα πρέπει να γίνεται με την αντίστοιχη προτεραιότητα. Για παράδειγμα ο διαχειριστής δεν μπορεί να προβεί στην δημιουργία ενός μοντέλου πριν δημιουργήσει την αντίστοιχη μάρκα.

Κεφάλαιο 3.2 - Υποσύστημα Χρήστη



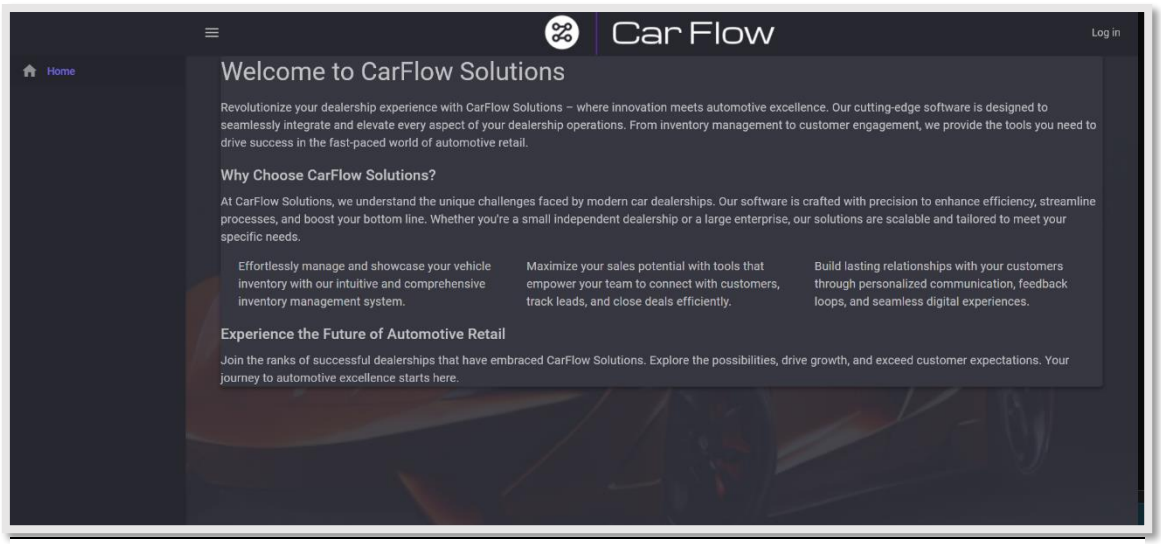
Εικόνα 8 - Διάγραμμα Περιπτώσεων Χρήσης Χρήστη

Στο υποσύστημα του χρήστη, ο εκάστοτε χρήστης που έχει συνδεθεί στο σύστημα θα μπορεί να δημιουργήσει νέες παραγγελίες αυτοκινήτων καθώς και να κάνει προεπισκόπηση αυτών. Στο παραπάνω διάγραμμα χρήσης απεικονίζονται οι διάφορες λειτουργίες στις οποίες μπορεί να προβεί ο χρήστης της εφαρμογής. Έτσι σαν αρχική διαδικασία ο χρήστης έχει την επιλογή να ξεκινήσει μια παραγγελία ενός οχήματος.

Στην συνέχεια μπορεί να ανατρέχει στις ανοιχτές παραγγελίες, στις παραγγελίες οι οποίες έχουν επιβεβαιωθεί, καθώς και στις παραγγελίες οι οποίες έχουν φτάσει σε επίπεδο τιμολόγησης και είναι πλέον κατοχυρωμένες. Κάθε κατοχυρωμένη παραγγελία δημιουργεί και ένα όχημα, οπότε ο χρήστης μπορεί να δει και τα οχήματα που έχει στην διάθεση του.

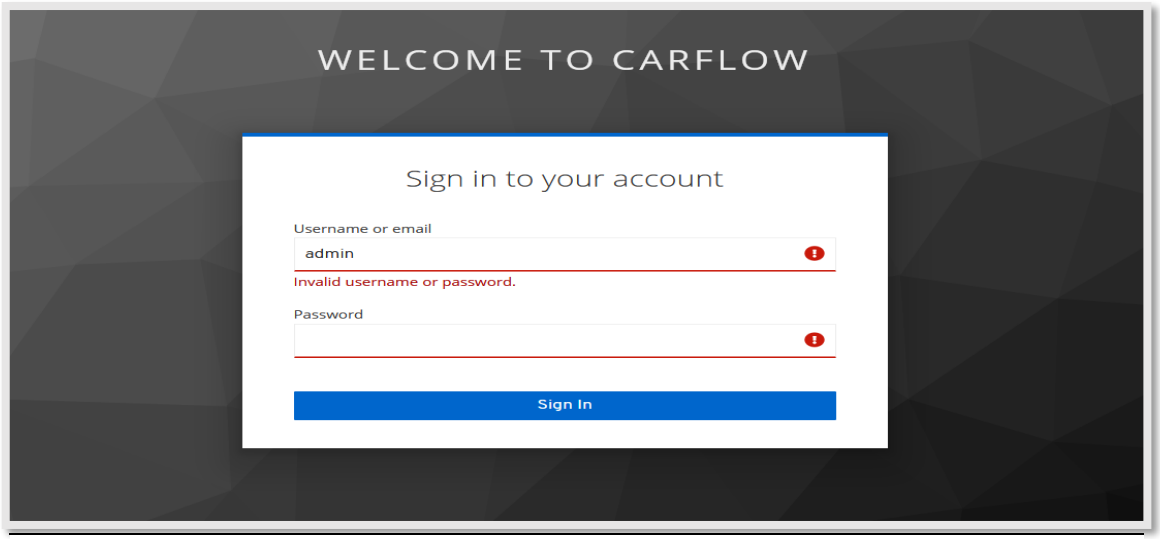
Κεφάλαιο 3.3 - Εφαρμογή Διαχειριστή

Αρχικά με την εκκίνηση της εφαρμογής, ο μόνος χρήστης που δημιουργείται αυτόματα είναι ο χρήστης διαχειριστή. Αφού ο χρήστης εισέλθει στην εφαρμογή έχει την επιλογή να πραγματοποιήσει σύνδεση, έτσι ώστε να γίνει αυθεντικοποίηση με βάση τα στοιχεία τα όποια θα εισάγει.



Εικόνα 9 - Αρχική Οθόνη

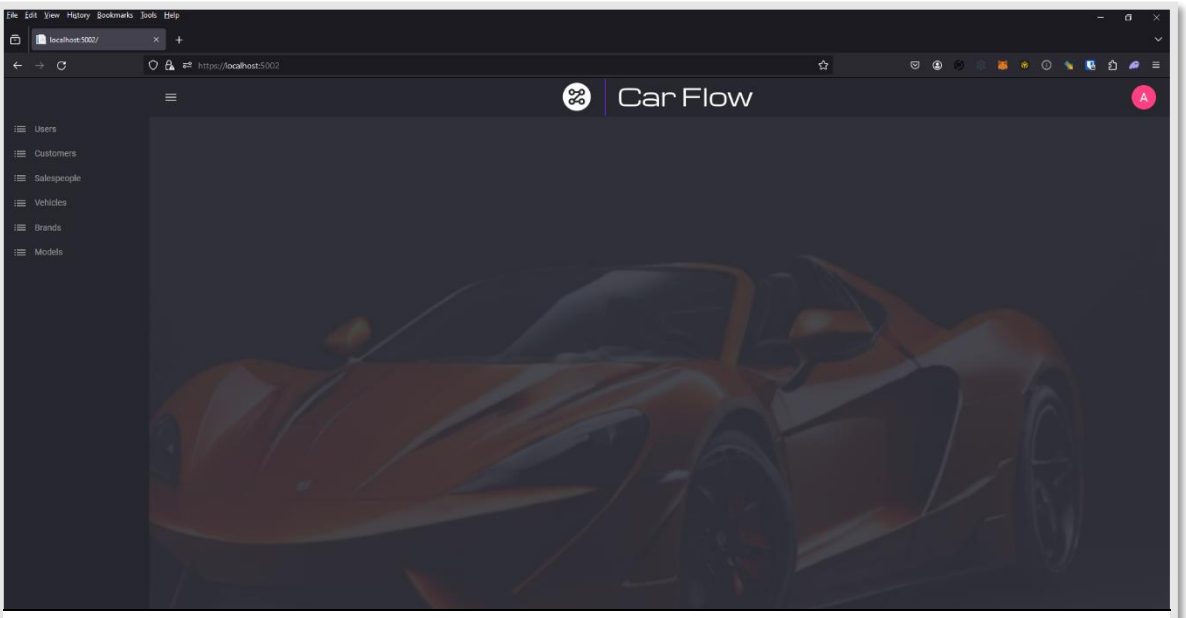
Κατά την επιλογή για σύνδεση στο σύστημα, ο χρήστης ανακατευθύνεται στην οθόνη του Keycloak για την αυθεντικοποίηση του. Σε αυτό το σημείο ο χρήστης πρέπει να εισάγει τα στοιχεία του (όνομα χρήστη και κωδικό πρόσβασης). Σε περίπτωση λάθους θα εμφανιστεί το αντίστοιχο μήνυμα στην οθόνη του, έτσι ώστε ο χρήστης να εισάγει ξανά τα στοιχεία του.



Εικόνα 10 - Αποτυχία σύνδεσης

Αφού πραγματοποιηθεί σύνδεση, ο χρήστης θα μεταφερθεί στην επόμενη σελίδα η οποία θα του προσφέρει ένα μενού για την διαχείριση των δεδομένων του συστήματος.

Στο μενού αυτό, ο διαχειριστής της εφαρμογής έχει την δυνατότητα να δημιουργήσει τις εγγραφές που χρειάζεται το σύστημα, έτσι ώστε να είναι λειτουργικό. Οι βασικές εγγραφές είναι οι Χρήστες (Users), οι οποίοι είναι οι χρήστες του συστήματος, αυτοί

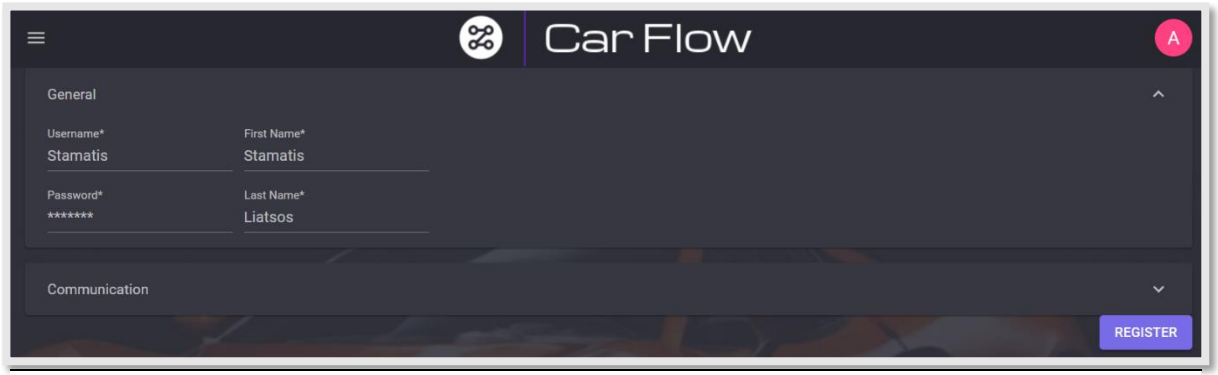


Εικόνα 11 - Μενού διαχειριστή

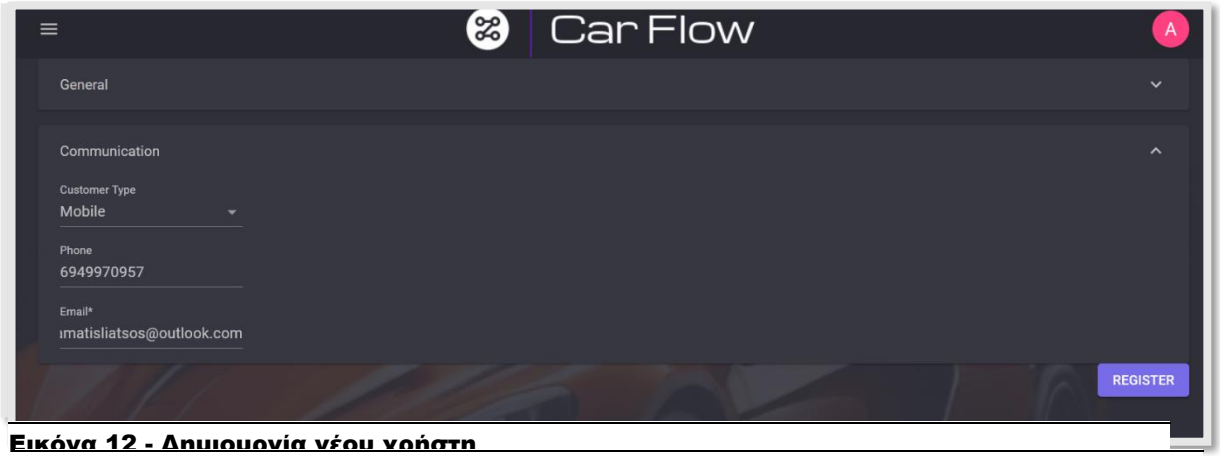
δηλαδή που θα έχουν την δυνατότητα για την δημιουργία κάποιας παραγγελίας μέσα στο σύστημα.

Για να δημιουργήσει ο διαχειριστής έναν νέο χρήστη, πρέπει να κατευθυνθεί στην επιλογή «Users» του μενού και στην συνέχεια να επιλέξει το κουμπί δημιουργίας (+) νέου χρήστη.

Στην συνέχεια θα πραγματοποιηθεί ανακατεύθυνση στην οποία θα πρέπει να εισάγει τα απαραίτητα στοιχεία για την δημιουργία της εγγραφής στο σύστημα. Τα στοιχεία αυτά είναι το όνομα χρήστη καθώς και ο κωδικός πρόσβασης και κάποια βασικά στοιχεία επικοινωνίας.



Εικόνα 13 - Στοιχεία σύνδεσης



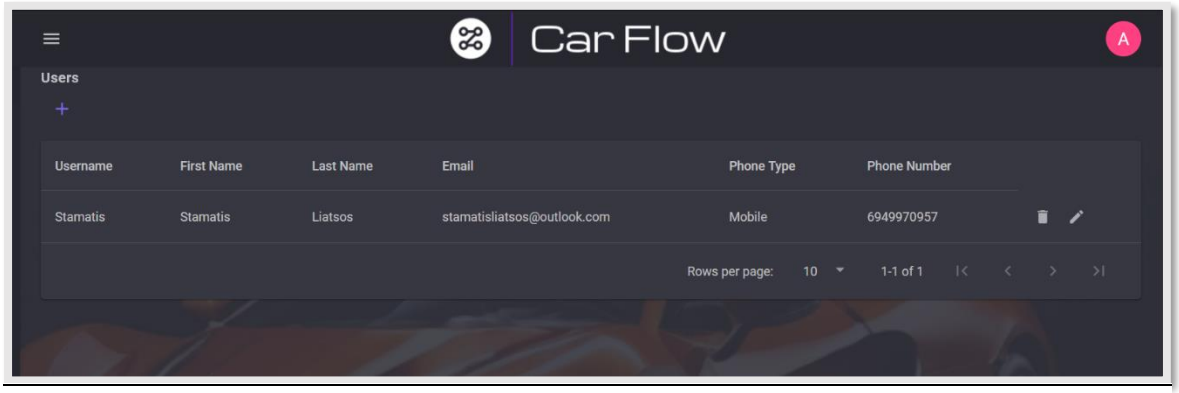
Εικόνα 12 - Δημιουργία νέου χρήστη

Εικόνα 14 - Στοιχεία επικοινωνίας

Εν συνεχεία, αφού ο διαχειριστής έχει συμπληρώσει όλα τα απαραίτητα πεδία, το κουμπί εγγραφής (Register) θα ενεργοποιηθεί αυτόματα, ώστε να μπορέσει να ολοκληρωθεί η διαδικασία της εγγραφής. Αφού ο διαχειριστής πατήσει στο κουμπί, η εγγραφή του χρήστη θα δημιουργηθεί μέσα στην βάση καθώς και μέσα στο keycloak, ώστε να μπορεί ο χρήστης να πραγματοποιήσει την σύνδεση στο σύστημα μέσω της αυθεντικοποίησης. Ο κωδικός πρόσβασης δεν αποθηκεύεται ως μέρος της βάσης, γιατί

αυτό θα ήταν κάτι το οποίο θα εγκυμονούσε κινδύνους, αλλά μεταφέρεται στο keycloak κατά την εγγραφή του χρήστη.

Εικόνα 15 - Δημιουργία χρήστη στο Keycloak



Εικόνα 16 - Ολοκληρωμένη Εγγραφή Χρήστη

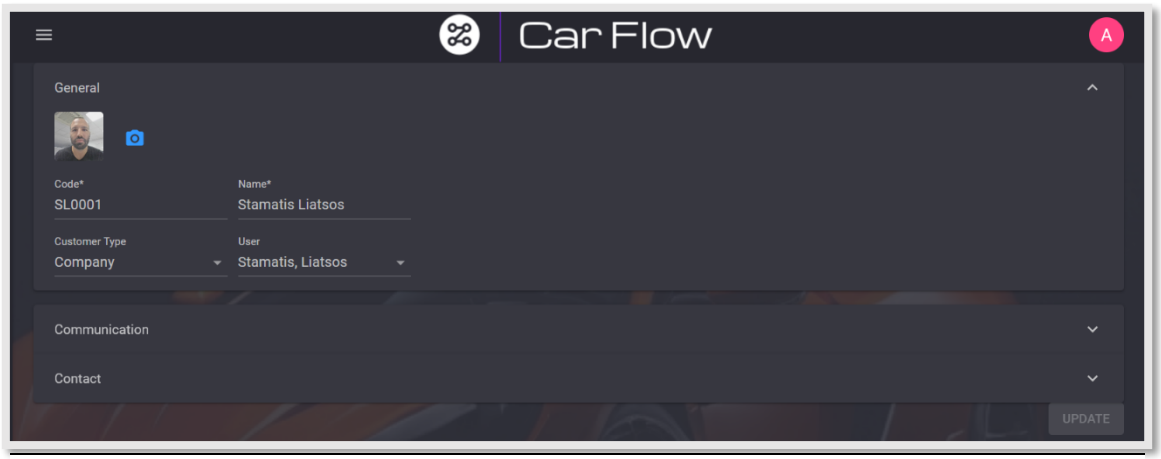
Στην συνέχεια, ο διαχειριστής μπορεί να δημιουργήσει έναν πελάτη, ο οποίος πρέπει να συνδεθεί με κάποιον χρήστη, ο οποίος θα έχει δημιουργηθεί με βάση τα προηγούμενα βήματα. Όπως και κατά την δημιουργία της εγγραφής του χρήστη στο σύστημα, έτσι και εδώ ο διαχειριστής έχει την δυνατότητα να εισάγει τα δεδομένα του πελάτη. Μερικά από αυτά τα δεδομένα είναι:

- Φωτογραφία του πελάτη: επιλογή αρχείου από το δίσκο
- Κωδικός πελάτη: Ένας μοναδικός κωδικός για την αναγνώριση του πελάτη μέσα στο σύστημα.
- Όνομα: όνομα εταιρείας ή ονοματεπώνυμο ιδιώτη
- Τύπος πελάτη: Εταιρεία ή πελάτης λιανικής
- Χρήστης: Ο χρήστης ο οποίος θα συνδεθεί με τον πελάτη.

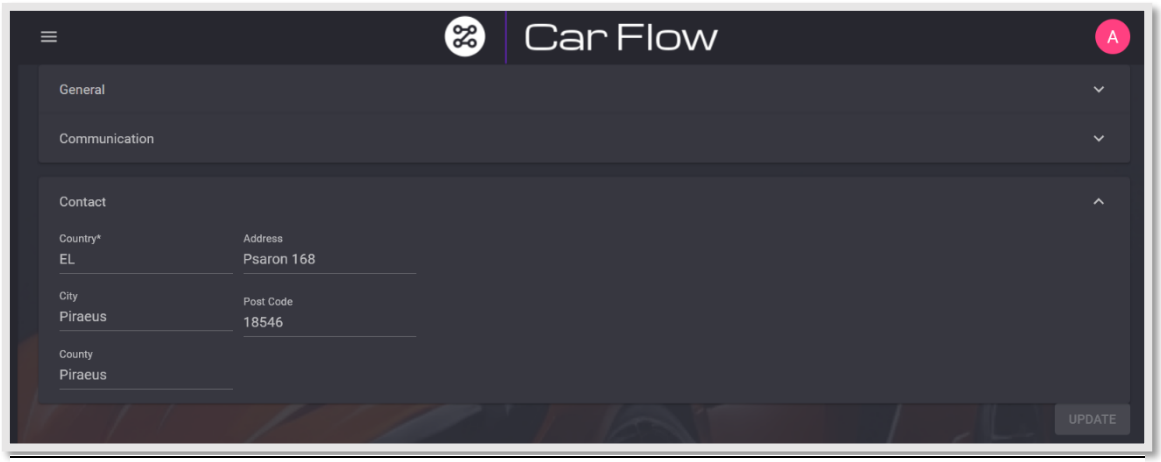
Κάποια περαιτέρω στοιχεία που θα χρειαστεί να συμπληρώσει ο διαχειριστής είναι τα στοιχεία επικοινωνίας του πελάτη καθώς και η διεύθυνση κατοικίας του.

- Διεύθυνση ηλεκτρονικού ταχυδρομείου: το email του πελάτη είναι απαραίτητο για την αποστολή μηνυμάτων κατά την επεξεργασία της παραγγελίας του

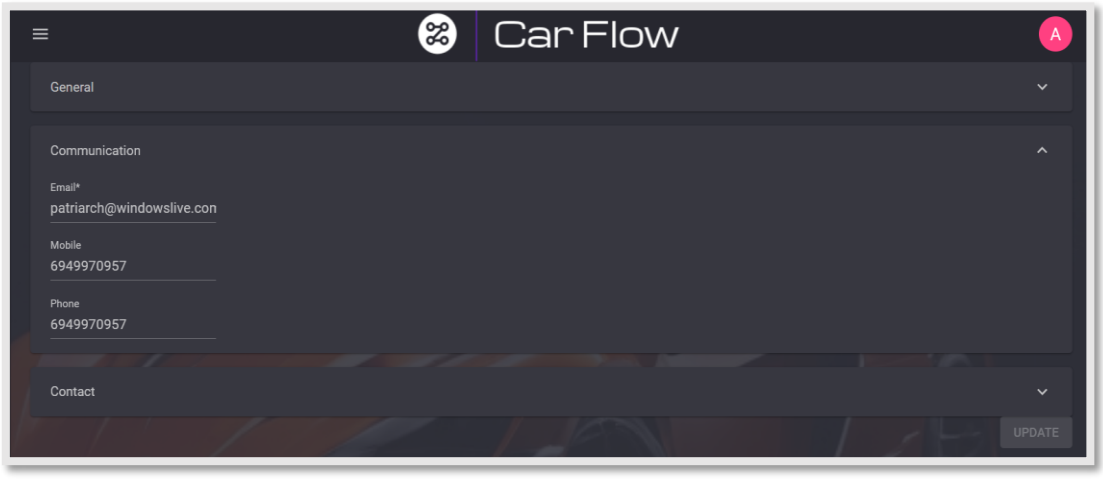
- Κινητό τηλέφωνο
- Σταθερό τηλέφωνο
- Χώρα
- Διεύθυνση
- Ταχυδρομικός κώδικας
- Πόλη



Εικόνα 17 - Βασικά στοιχεία πελάτη



Εικόνα 18 - Στοιχεία διεύθυνσης πελάτη

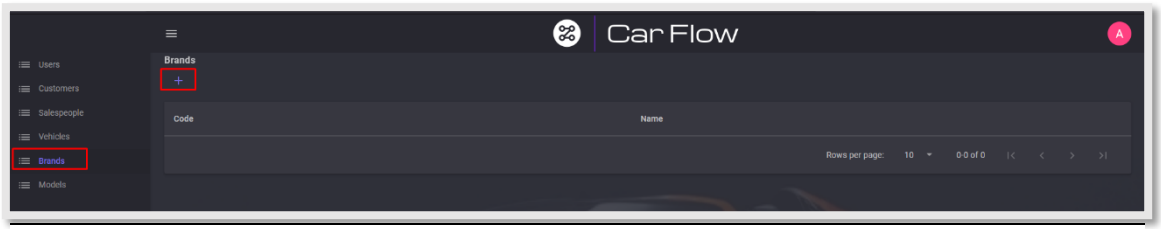


Εικόνα 19 - Στοιχεία επικοινωνίας πελάτη

Σε τελευταίο στάδιο, για να ολοκληρωθεί η διαδικασία της καταχώρησης των βασικών δομών του συστήματος, ο διαχειριστής καλείται να δημιουργήσει τις

εγγραφές των μοντέλων καθώς και μαρκών των αυτοκινήτων που θα υποστηρίζει το σύστημα.

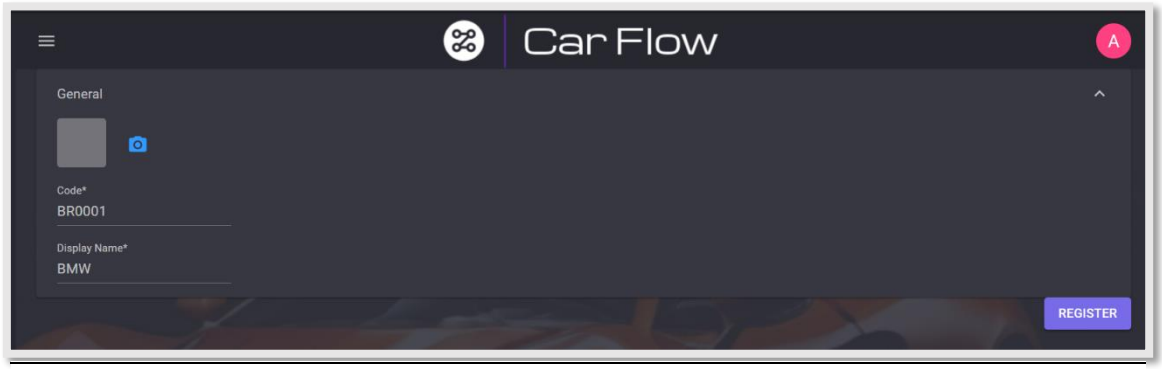
Για την δημιουργία των μαρκών, θα πρέπει να κατευθυνθεί στο μενού



Εικόνα 20 - Δημιουργία Μάρκας

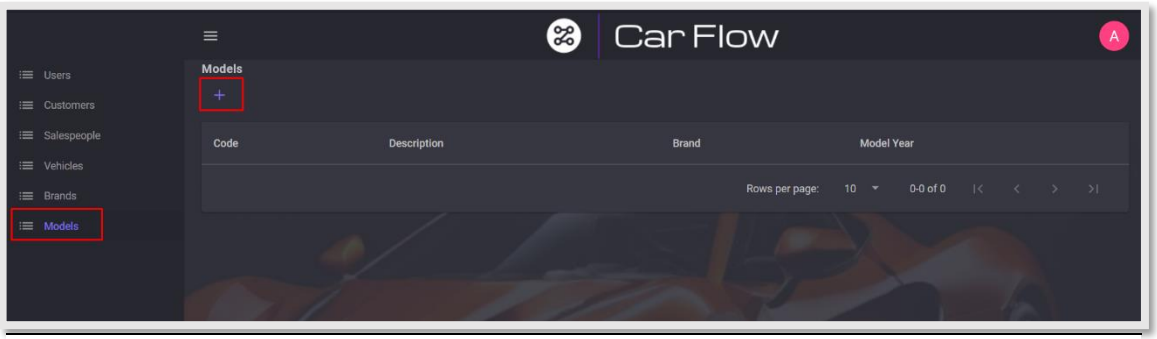
«Brands» και όπως και προηγουμένως να πατήσει στο κουμπί για την δημιουργία (+) νέας μάρκας.

Στην συνέχεια, στην σελίδα στην οποία θα κατευθυνθεί, θα πρέπει να κάνει εισαγωγή των στοιχείων για την μάρκα την οποία θέλει να δημιουργήσει. Τα στοιχεία αυτά είναι το logo (φωτογραφία), ο κωδικός/μοναδικό αναγνωριστικό καθώς και το όνομα της μάρκας στο σύστημα.



Εικόνα 21 - Στοιχεία Μάρκας

Ως τελευταίο βήμα για την ολοκλήρωση της εγκατάστασης του συστήματος, πρέπει να δημιουργηθούν και τα μοντέλα των οχημάτων. Για την δημιουργία των μοντέλων, όπως και στα προηγούμενα βήματα, θα πρέπει ο χρήστης μέσα απο το μενού επιλογών να κατευθυνθεί στο μενού «Models» και να πατήσει στο κουμπί για την δημιουργία (+) νέου μοντέλου.

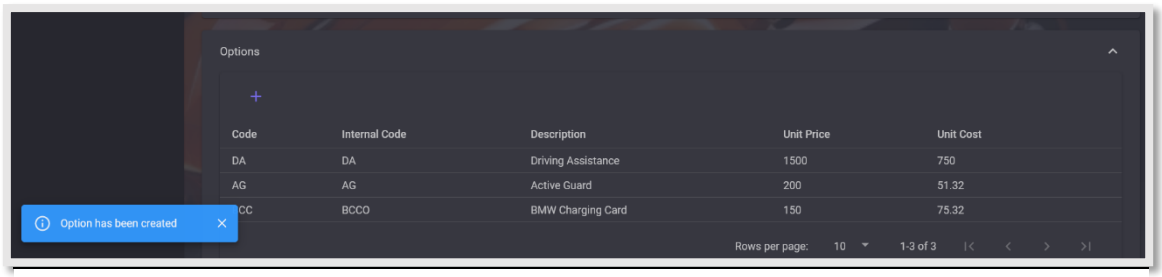


Εικόνα 22 - Δημιουργία Μοντέλου

Στην συνέχεια, με την ίδια ακριβώς λογική που ακολουθήθηκε και στις προηγούμενες σελίδες, ο χρήστης καλείται να εισάγει κάποια βασικά στοιχεία και χαρακτηριστικά του μοντέλου όπως ο κωδικός του μοντέλου, ο οποίος θα χρησιμοποιηθεί σαν μοναδικό αναγνωριστικό της εγγραφής στο σύστημα, η περιγραφή, το έτος κατασκευής του μοντέλου και η μάρκα του μοντέλου.

Εικόνα 23 - Στοιχεία Μοντέλου

Για το μοντέλο, χρειαζόμαστε επίσης και τον εξοπλισμό τον οποίο χαρακτηρίζει το μοντέλο. Ο εξοπλισμός απαρτίζεται από τον βασικό εξοπλισμό και τον προαιρετικό. Στο σύστημά μας, το είδος του εξοπλισμού (βασικός ή προαιρετικός), καθορίζεται από τις τιμές που έχουν τα πεδία «Internal Code» και «Code». Στις περιπτώσεις που τα πεδία αυτά έχουν την ίδια τιμή στο σύστημα ο εξοπλισμός θεωρείται βασικός ενώ σε κάθε άλλη περίπτωση θεωρείται προαιρετικός.

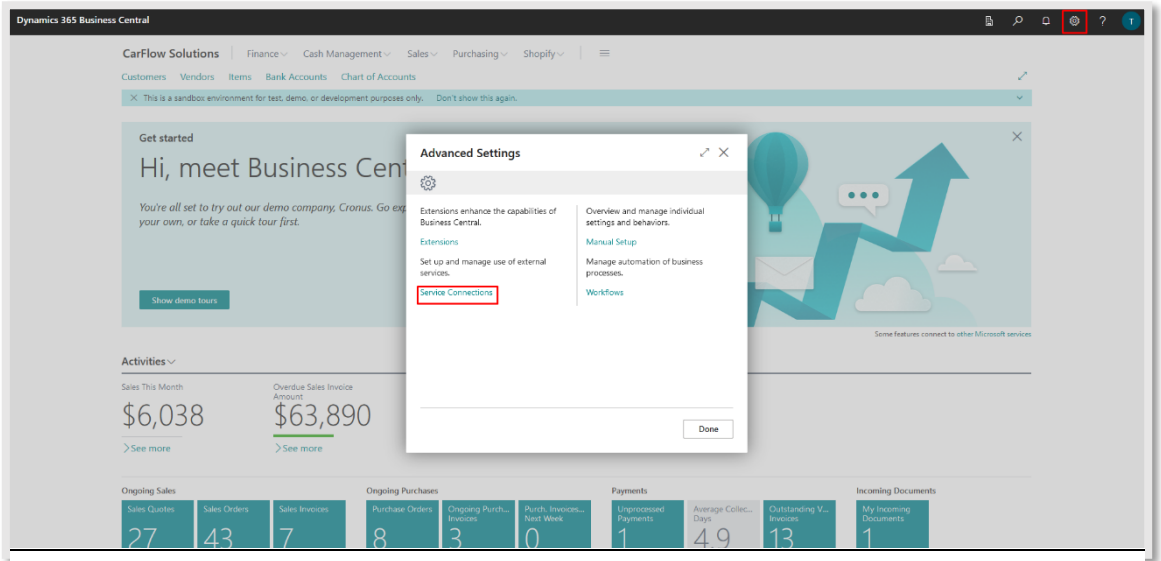


Εικόνα 24 - Δημιουργία εξοπλισμού μοντέλου

Κεφάλαιο 3.4 - Εφαρμογή Διαχειριστή - ERP

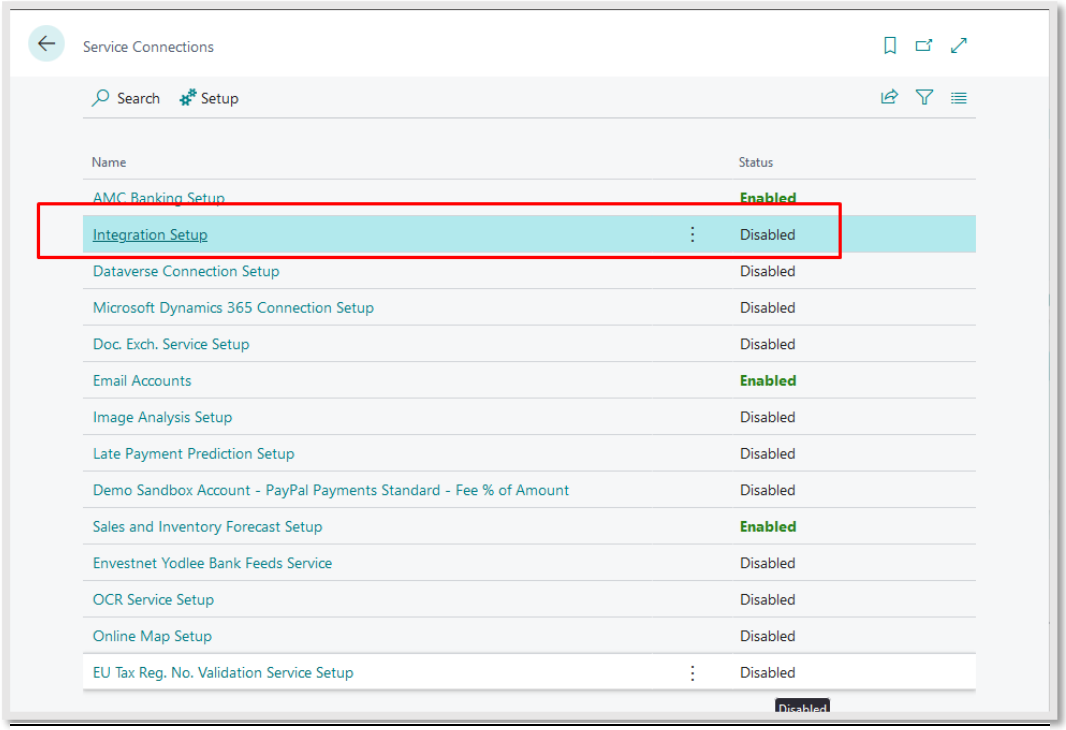
Αφού έχει ολοκληρωθεί όλη η διαδικασία με την δημιουργία εγγραφών, ώστε να είναι λειτουργικό το σύστημα, θα πρέπει στην συνέχεια να γίνει παραμετροποίηση του ERP συστήματος με το οποίο θα συγχρονίζονται τα δεδομένα.

Αρχικά θα πρέπει ο διαχειριστής του συστήματος να μεταβεί στο μενού «Settings» και από εκεί να επιλέξει το «Advanced Settings» και στην συνέχεια να μεταβεί στο μενού «Service Connections».



Εικόνα 25 - Service Connections

Από το μενού που θα εμφανιστεί θα πρέπει να επιλέξει το «Integration Setup» το οποίο θα εμφανίσει την σελίδα για την εγκατάσταση των παραμέτρων για την διασύνδεση με το σύστημα μας.



Εικόνα 26 - Integration Setup

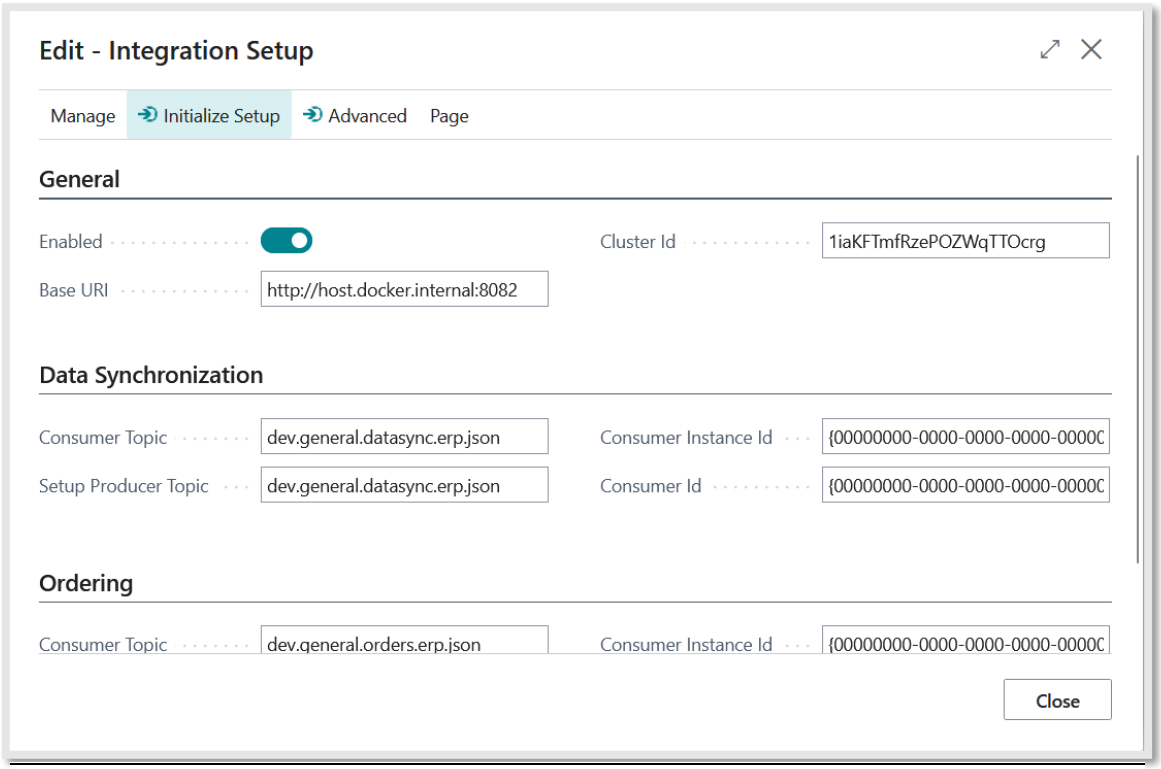
Στην οθόνη που θα εμφανιστεί θα πρέπει να εισάγουμε κάποιες τιμές στα κελιά ώστε να ενεργοποιήσουμε την επικοινωνία μέσω του Kafka με το σύστημα μας. Τα πεδία αυτά με τις αντίστοιχες τιμές τους είναι

- **General Tab**
  - Base Uri: Το Uri του Kafka Rest Proxy –

    προεπιλεγμένη τιμή: http://host.docker.internal:8082
- **Data Synchronization Tab**
  - Consumer Topic: Το θέμα από το οποίο θα λαμβάνουμε μηνύματα σχετικά με τα δεδομένα από το σύστημά μας σε περίπτωση δημιουργίας ή ανανέωσης αυτών – προεπιλεγμένη τιμή: dev.general.datasync.erp.json
  - Setup Topic Producer: Το θέμα στο οποίο θα καταγράφουμε τις αλλαγές του ERP ώστε να μεταφέρονται στο σύστημά μας – προεπιλεγμένη τιμή: dev.general.datasync.erp.json

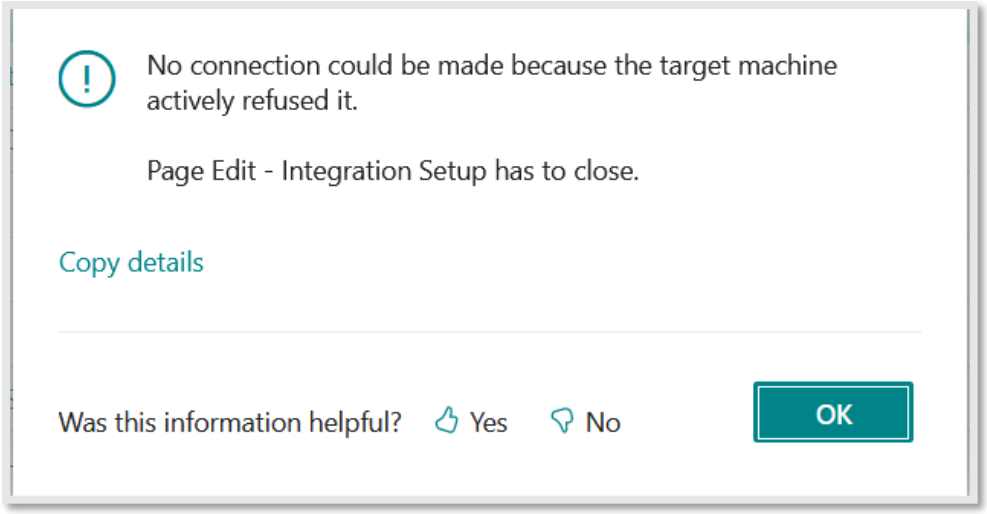
- **Ordering Tab**
    - Consumer Topic: Το θέμα από το οποίο θα λαμβάνουμε μηνύματα σχετικά με νέες παραγγελίες – προεπιλεγμένη τιμή: dev.general.orders.erp.json
    - Producer Topic: Στο θέμα αυτό θα καταγράφονται οι ανανεώσεις των παραγγελιών ανάλογα με την αλλαγή της κατάστασής τους μέσα στο ERP – προεπιλεγμένη τιμή: dev.general.orders.json

Αφού έχουν συμπληρωθεί οι προαναφερθείσες τιμές, θα πρέπει να ενεργοποιηθεί το Enabled ώστε να ενεργοποιήσουμε την διασύνδεση μεταξύ των συστημάτων. Κατά την επιτυχή διασύνδεση των συστημάτων η ένδειξη Enabled θα παραμείνει επιλεγμένο και το πεδίο Cluster Id θα πάρει αυτόματα τιμή με αυτή που προέκυψε από την επιτυχή σύνδεση με το Kafka Cluster.



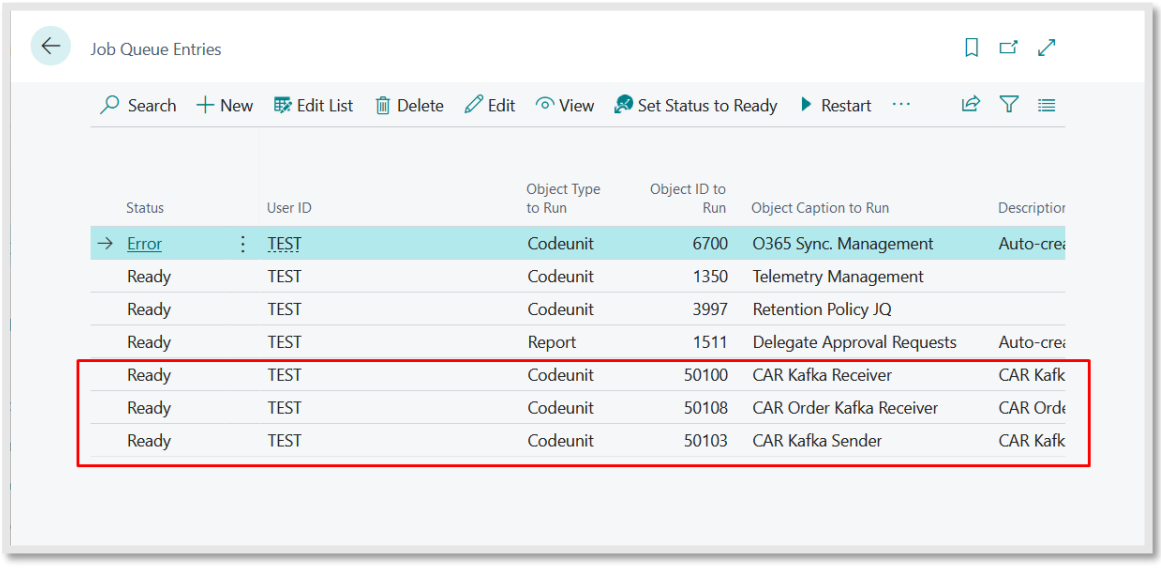
Εικόνα 27 - Επιτυχής διασύνδεση

Σε περίπτωση λάθους με την διασύνδεση θα εμφανιστεί και το αντίστοιχο μήνυμα λάθους και η ένδειξη Enabled θα απενεργοποιηθεί αυτόματα.



Εικόνα 28 - Μήνυμα Λάθους

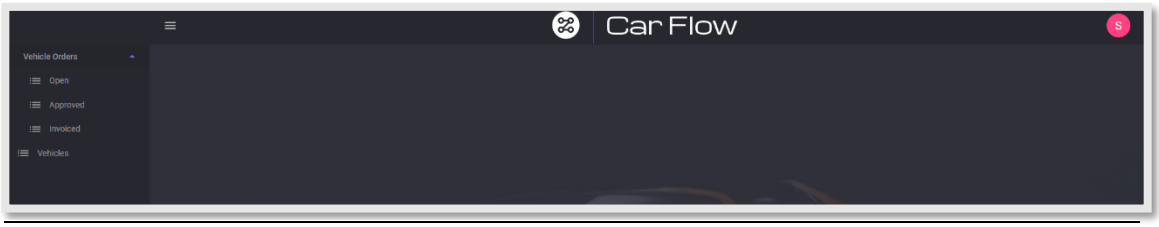
Τελευταίο στάδιο, είναι η παραμετροποίηση και ενεργοποίηση των εργασιών που τρέχουν στο παρασκήνιο, οι οποίες ανταλλάσσουν δεδομένα με το σύστημά μας, ώστε και τα δύο συστήματα να είναι συγχρονισμένα. Αφού τις ενεργοποιήσουμε, η κάθε διεργασία θα τρέχει στο παρασκήνιο ανά 5 λεπτά και είτε θα μεταβιβάζει είτε θα παραλαμβάνει δεδομένα.



Εικόνα 29 - Λίστα Διεργασιών Παρασκηνίου

Κεφάλαιο 3.5 - Εφαρμογή Χρήστη 1/2

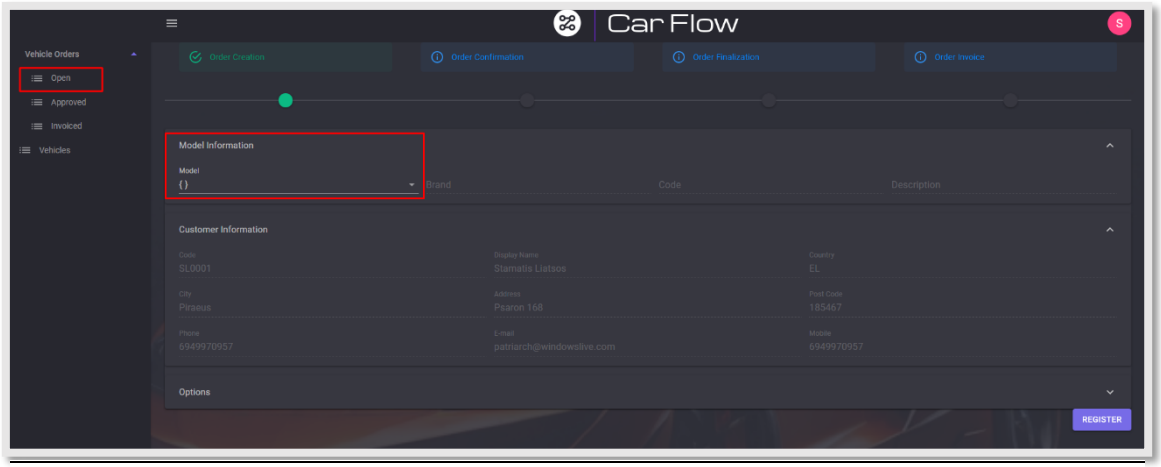
Εισερχόμενος στην εφαρμογή ο χρήστης, μέσω αυθεντικοποίησης με το Keycloak, μπορεί να δει το μενού των παραγγελιών, ανάλογα με την κατάσταση στην οποία βρίσκεται κάθε παραγγελία καθώς και την λίστα με τα οχήματα που έχουν δημιουργηθεί μέσα από τις διάφορες παραγγελίες.



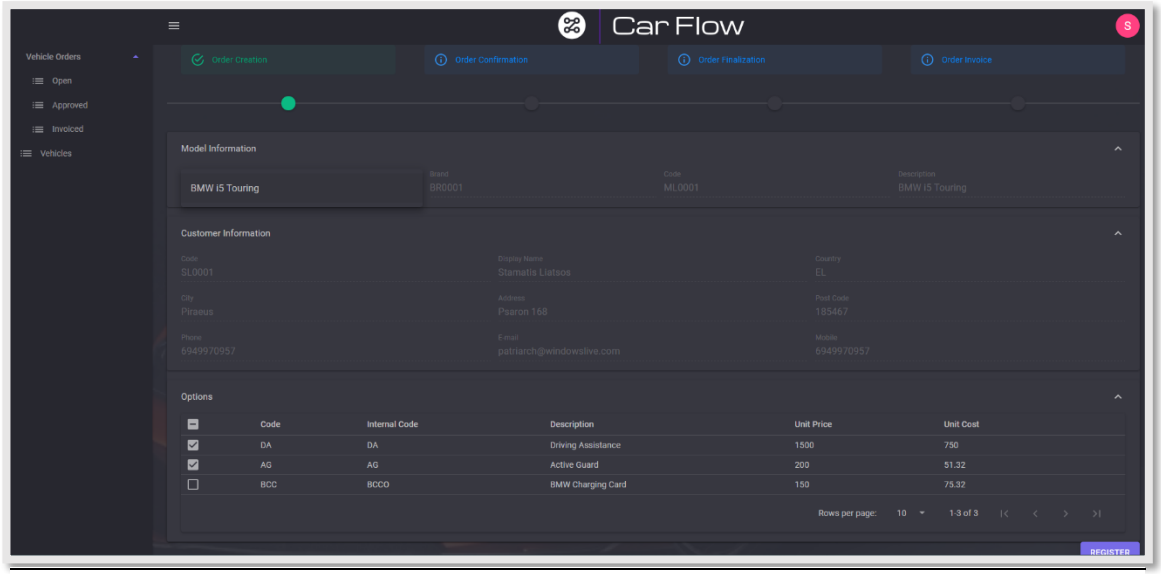
Εικόνα 30 - Μενού χρήστη

Πηγαίνοντας στο μενού των ανοιχτών παραγγελιών, μπορούμε να δημιουργήσουμε μια νέα προσφορά (αρχικό στάδιο παραγγελίας). Η προσφορά αυτή θα προωθηθεί στο σύστημα του ERP προς αξιολόγηση και από αυτήν θα δημιουργηθεί μια παραγγελία.

Κατά την δημιουργία της προσφοράς, θα πρέπει ο χρήστης να επιλέξει το μοντέλο για το οποίο ενδιαφέρεται. Πολλά από τα πεδία της φόρμας, έχουν προ-συμπληρωθεί αυτόματα με βάση τα στοιχεία του χρήστη μέσα από το σύστημα.

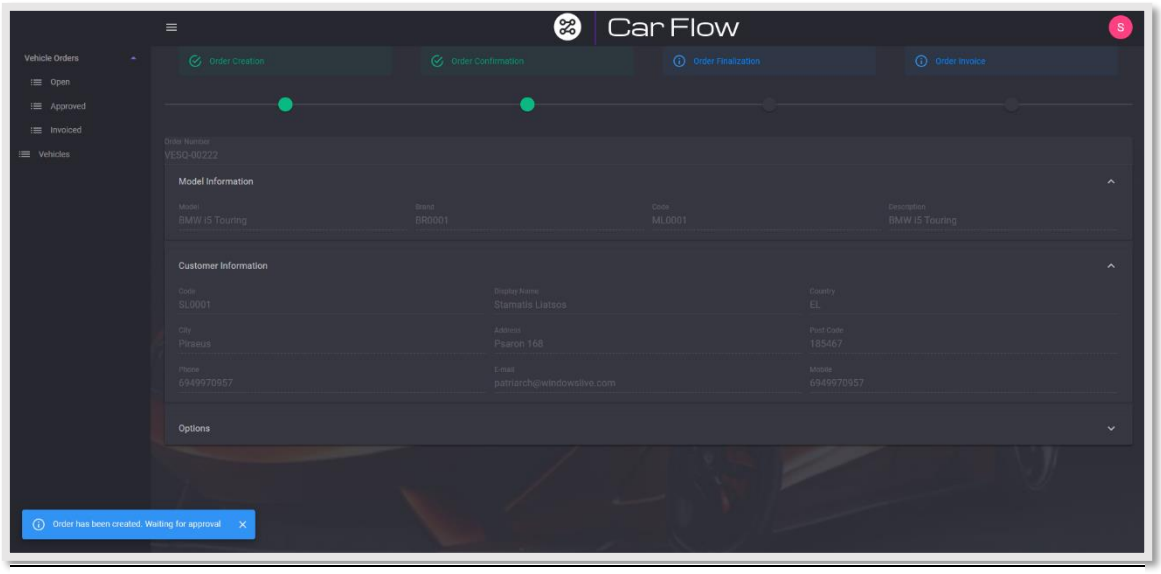


Εικόνα 31 -Δημιουργία Προσφοράς (1/2)

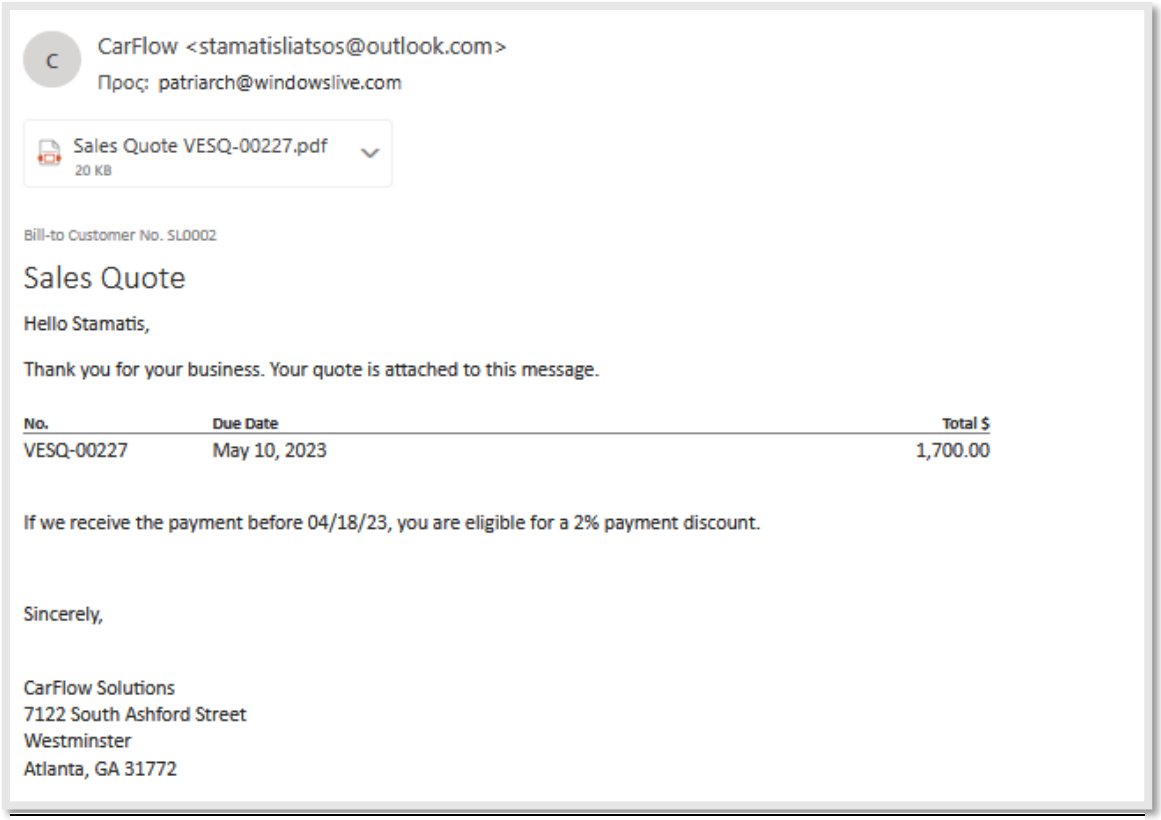


Εικόνα 32 - Δημιουργία Προσφοράς (2/2)

Κατά την επιλογή του μοντέλου, μεταφέρθηκαν στην προσφορά αυτόματα τα εξαρτήματα του μοντέλου και ο χρήστης έχει την επιλογή να επιλέξει ή όχι κάποια από αυτά. Στην συνέχεια, καταχωρώντας την προσφορά, αυτή πλέον δεν μπορεί να τροποποιηθεί και η κατάστασή της έχει αλλάξει σε «Επιβεβαίωση Παραγγελίας».



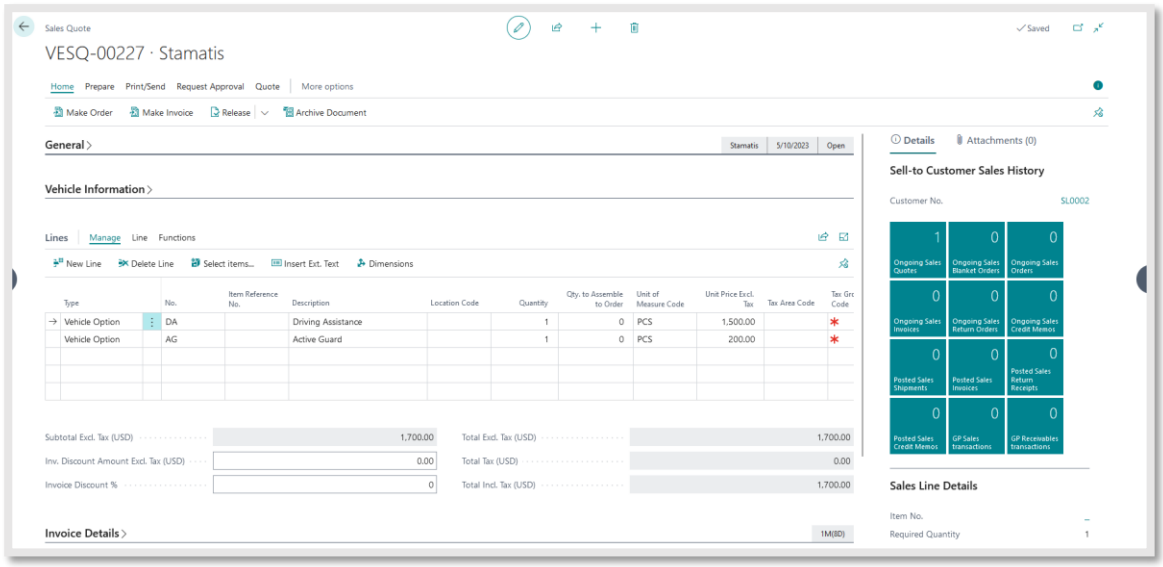
Εικόνα 33 - Επιβεβαίωση Παραγγελίας



Εικόνα 34 - Ενημερωτικό email προσφοράς

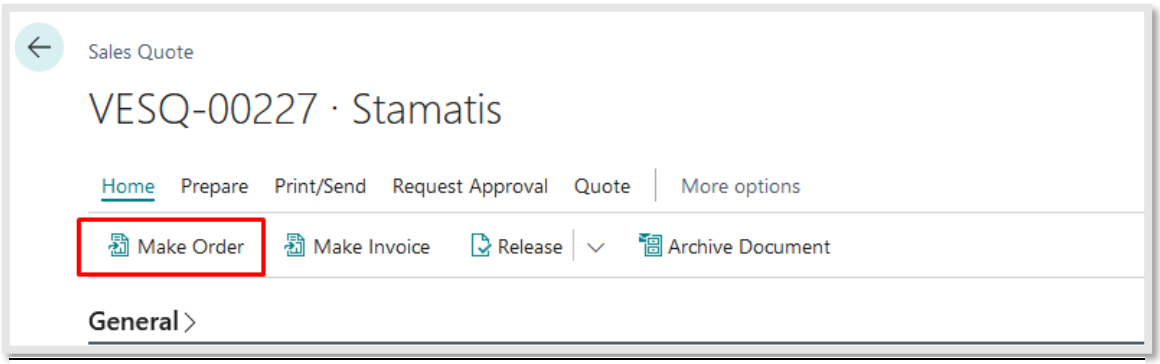
Κεφάλαιο 3.6 - Εφαρμογή Χρήστη – ERP

Η προσφορά έχει πλέον μεταφερθεί αυτόματα στο ERP σύστημα (μέσω των εργασιών παρασκηνίου) και ταυτόχρονα έχει σταλεί και ενημερωτικό email στον πελάτη ώστε να γνωρίζει το κόστος και την κατάσταση της προσφοράς.



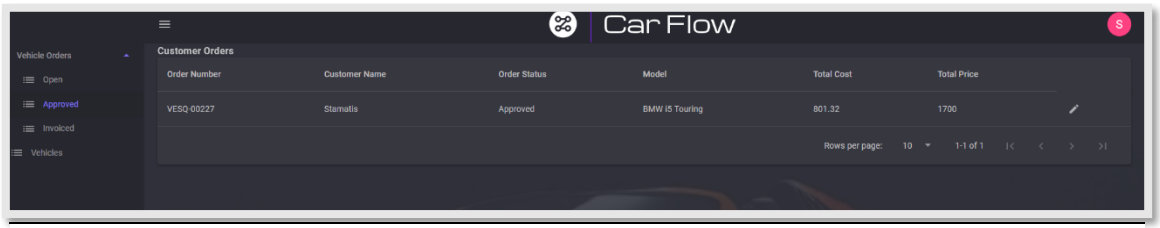
Εικόνα 35 - Προσφορά στο ERP

Σε αυτό το στάδιο κι εφόσον συμφωνήσει ο πελάτης ο χρήστης του ERP καλείται να μετατρέψει την προσφορά αυτή σε παραγγελία. Για να το επιτύχει, θα πρέπει να εκτελέσει το «Make Order» από την γραμμή ενεργειών.

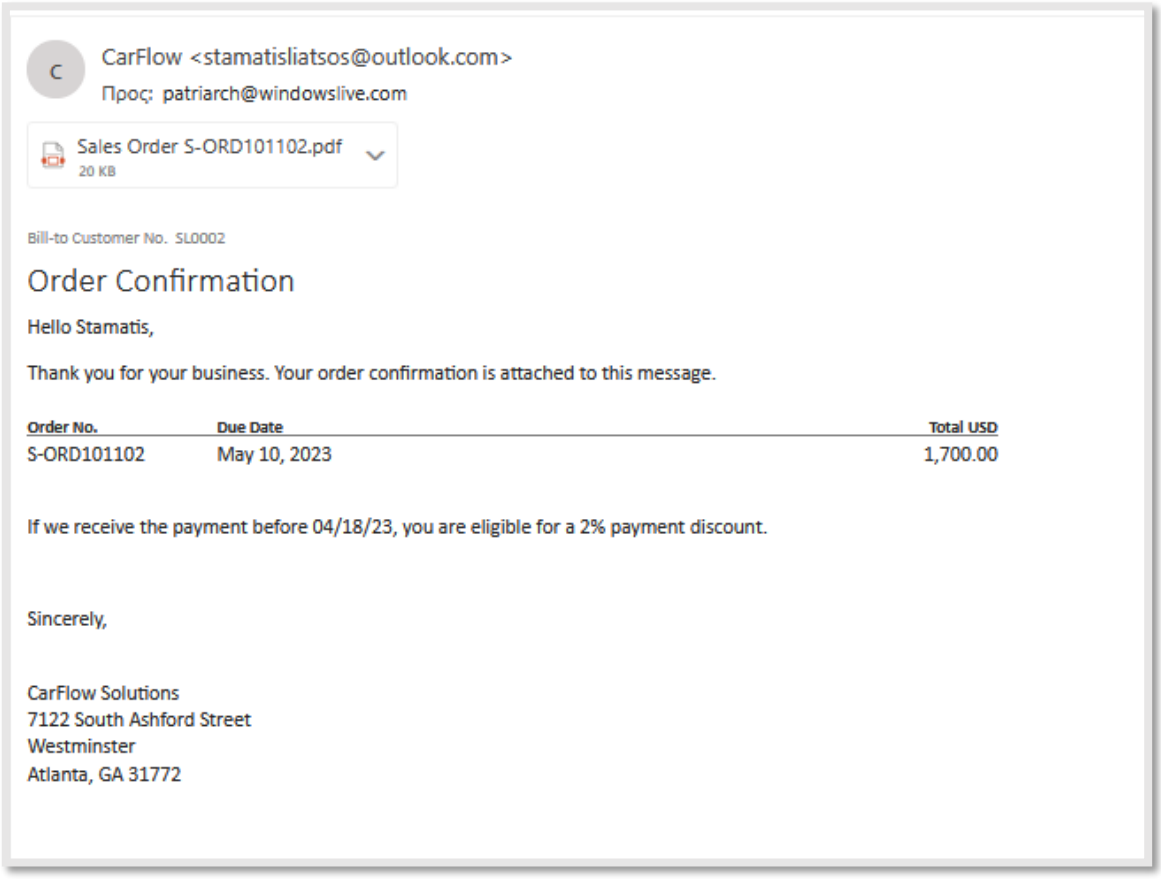


Εικόνα 36 - Make Order Action Ribbon

Μετά από την εκτέλεση της ενέργειας αυτής, ο χρήστης μπορεί να βρει την παραγγελία του πλέον κάτω από τις εγκεκριμένες παραγγελίες και αντίστοιχος θα έχει λάβει ένα μήνυμα ηλεκτρονικού ταχυδρομείου με την παραγγελία αυτή την φορά.

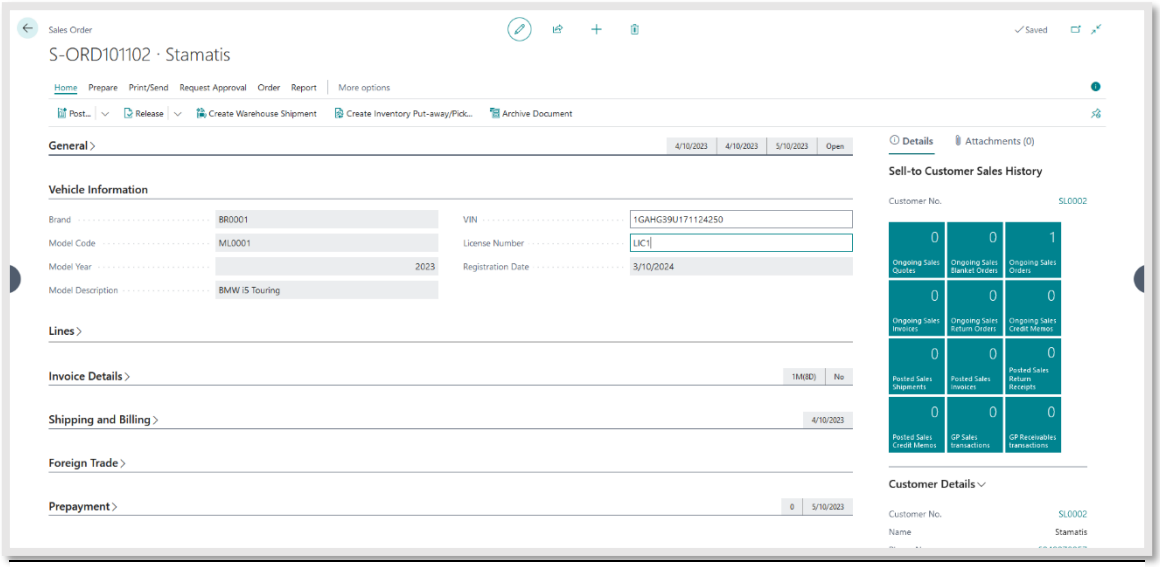


Εικόνα 37 - Εγκεκριμένη παραγγελία



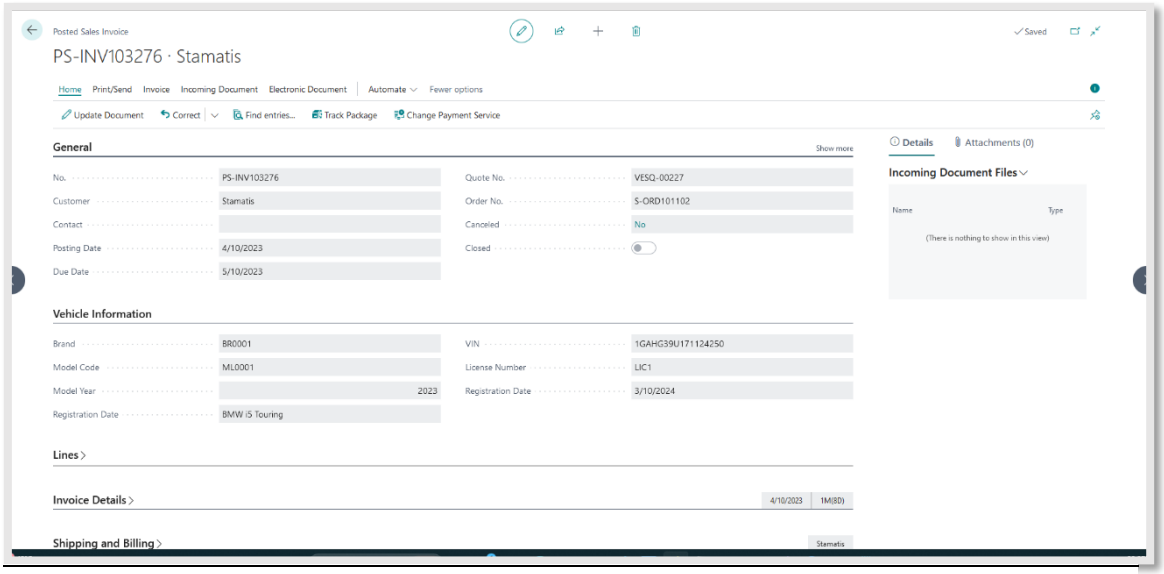
Εικόνα 38 – Ενημερωτικό email Παραγγελίας

Στην συνέχεια, ο χρήστης του ERP καλείται, ώστε να προχωρήσει στην τιμολόγηση, να συμπληρώσει τα απαραίτητα πεδία του οχήματος: Αριθμός Πλαισίου (VIN) και τον Αριθμό Πινακίδας (License Plate Number).



Εικόνα 39 - Παραγγελία ERP

Έπειτα, από την γραμμή ενεργειών θα πρέπει να προχωρήσει στην εκτέλεση της ενέργειας της τιμολόγησης. Κατά την τιμολόγηση ο χρήστης θα μπορεί πλέον να βρει την παραγγελία του κάτω από το μενού των τιμολογημένων παραγγελιών και θα έχει την δυνατότητα να δει το τελικό παραστατικό του τιμολογίου.



Εικόνα 40 - Τιμολογημένη Παραγγελία (ERP)

Κατά την τιμολόγηση στο ERP, ένα μήνυμα αποστέλλεται και επεξεργάζεται αυτόματα από την υπηρεσία παραγγελιών ώστε να ενημερωθεί η κατάσταση της αρχικής εγγραφής στο παράλληλο σύστημα.

Κεφάλαιο 3.7 - Εφαρμογή Χρήστη - 2/2

Το εκτυπώσιμη μορφή του τιμολόγιου είναι πλέον διαθέσιμη και στα δύο συστήματα και η κατάσταση της παραγγελίας είναι συγχρονισμένη.

Εικόνα 41 - Παραστατικό τιμολογημένης παραγγελίας

Μετά το πέρας της τιμολόγησης δημιουργείται στο σύστημα και το αντίστοιχο όχημα, όπως αυτό προκύπτει από την διεκπεραίωση της όλης ροής. Το όχημα αυτό, είναι διαθέσιμο μέσα στην λίστα των οχημάτων πελάτων.

Εικόνα 42 - Όχημα Πελάτη

Συμπεράσματα

Αυτή η διπλωματική εργασία διερευνά τη χρήση του gRPC (Remote Procedure Call) ως εναλλακτική έναντι της REST με σκοπό την επικοινωνία των μικρουπηρεσιών και της διασύνδεσης τους με το Microsoft Business Central ERP. Η μελέτη αποσκοπεί στο να αξιολογήσει τα πλεονεκτήματα και τις προκλήσεις που συνδέονται με τη χρήση του gRPC σε σύγκριση με τα παραδοσιακά πρωτόκολλα επικοινωνίας βασισμένα σε JSON (βλ. REST).

Η έρευνα ξεκινά με μια σφαιρική ανάλυση της αρχιτεκτονικής των μικρουπηρεσιών και της αυξανόμενης χρήσης τους στον τομέα του Enterprise Resource Planning (ERP). Οι μικρουπηρεσίες παρέχουν αρθρωτοτήτα (modularity) και επεκτασιμότητα, αλλά εξαρτώνται σημαντικά από την αποτελεσματική επικοινωνία μεταξύ τους.

Ο κύριος τομέας εστίασης της διπλωματικής εργασίας είναι η υλοποίηση του gRPC ως πρωτοκόλλου επικοινωνίας μεταξύ των μικρουπηρεσιών. Το gRPC προσφέρει σημαντικά οφέλη ως προς την απόδοση σε σύγκριση με το παραδοσιακό REST χρησιμοποιώντας το Protocol Buffers, ένα δυαδικό σύστημα σειριοποίησης, που οδηγεί στη μειωμένη καθυστέρηση και την αυξημένη αποτελεσματικότητα στη μεταφορά δεδομένων.

Το εμπειρικό κομμάτι της μελέτης περιλαμβάνει την ανάπτυξη και την εφαρμογή ενός πρωτότυπου συστήματος που ενσωματώνει το gRPC στην αρχιτεκτονική των μικρουπηρεσιών και τη σύνδεση τους με το Microsoft Business Central ERP. Η διπλωματική αξιολογεί βασικούς δείκτες απόδοσης, όπως η καθυστέρηση, η επιτάχυνση και η χρήση πόρων, προκειμένου να αξιολογήσει την επίδραση της υιοθέτησης του gRPC στη συνολική αποτελεσματικότητα του συστήματος.

Επιπλέον, η έρευνα εξετάζει τις προκλήσεις και τους σκοπούς που συνδέονται με την εφαρμογή του gRPC, συμπεριλαμβανομένης της συμβατότητάς του με τα υπάρχοντα συστήματα, τις επιπτώσεις ασφαλείας και την ευκολία διασύνδεσης του με το Business Central ERP.

Συνοψίζοντας, η διπλωματική εργασία παρουσιάζει μια σφαιρική ανάλυση της ενσωμάτωσης του gRPC σε μια αρχιτεκτονική μικρουπηρεσιών, ειδικά σχεδιασμένη για την επικοινωνία με το Business Central ERP. Τα ευρήματα συνεισφέρουν σημαντικές πληροφορίες για τα πλεονεκτήματα και τις προκλήσεις της υιοθέτησης του gRPC ως πρωτοκόλλου επικοινωνίας στον τομέα των συστημάτων ERP, παρέχοντας έτσι βάση για περαιτέρω έρευνα και πρακτικές εφαρμογές στον τομέα.

Παράρτημα Α

GitHub - sliatsos/carauto_public

Βιβλιογραφία

- https://en.m.wikipedia.org/wiki/REST
- https://en.m.wikipedia.org/wiki/GRPC
- https://en.m.wikipedia.org/wiki/HTTP/2
- https://www.researchgate.net/publication/307906792_Impact_of_Implementing_HTTP2_in_Web_Services
- "gRPC vs Rest" (https://grpc.io/)
- "gRPC vs. REST: Key Similarities and Differences" (https://blog.dreamfactory.com/grpc-vs-rest-how-does-grpc-compare-with-traditional-rest-apis/)
- gRPC vs REST: Understanding gRPC, OpenAPI and REST and when to use them in API design | Google Cloud Blog
- "Microservice Architecture" (https://microservices.io/)
- "Microservices" (https://en.wikipedia.org/wiki/Microservices)
- "ELK" (https://www.elastic.co/guide/index.html)
- "SOA Principles of Service Design", Thomas Erl – 2008
- Lewis, J., & Fowler, M. (2014). Microservices: a definition of this new architectural term. Retrieved from https://martinfowler.com/articles/microservices.html