



Πανεπιστήμιο Κρήτης  
Τμήμα Επιστήμης Υπολογιστών

«Ανάλυση, σχεδίαση και  
υλοποίηση λειτουργιών δεδομένων  
(backend)  
σε πληροφοριακό σύστημα  
εταιρείας εμπορίας ελαιολάδου»

Μίνως Σταυρακάκης - *csd4120*

Ιούλιος 2024

Επιβλέπων: Τσατσάκης Νικόλαος

Επόπτης: Πλεξουσάκης Δημήτριος

Αξιολογητής: Τζιτζικας Ιωάννης

# Πίνακας Περιεχομένων

---

Πίνακας Περιεχομένων

Περίληψη

Επισκόπηση του Έργου

**Βασικοί Στόχοι**

**Κύρια Ευρήματα και Συνεισφορές**

## Κεφάλαιο 1: Εισαγωγή

1.1 Ιστορικό και Πλαίσιο

1.2 Περιγραφή του Προβλήματος

1.3 Στόχοι της Μελέτης

Αδυναμίες Υπάρχοντος Συστήματος

Κύριοι Στόχοι Του Έργου

1.4 Πεδίο του Έργου

Καλυπτόμενες Πτυχές του Έργου:

Μη Καλυπτόμενες Πτυχές του Έργου:

## Κεφάλαιο 2: Σχεδίαση και Αρχιτεκτονική Συστήματος

2.1 Ανάλυση Απαιτήσεων

2.1.1 Λειτουργικές Απαιτήσεις

2.1.2 Μη Λειτουργικές Απαιτήσεις

2.2 Αρχιτεκτονική Συστήματος

2.2.1 Επισκόπηση της Αρχιτεκτονικής του Συστήματος

Migration-service

API-service

2.2.2 Σχεδίαση του API:

API - Routes

Σχεδιαστικές Αρχές

Περιγραφή των Κύριων Routes

2.3 Σχεδίαση Βάσης Δεδομένων

2.3.1 Σχεδίαση Σχήματος

2.3.2 Διάγραμμα Οντοτήτων-Συσχετίσεων

2.4 Κατηγορίες Χρηστών του Συστήματος

2.4.1 Τύποι Χρηστών

2.4.2 Αυθεντικοποίηση Διαχειριστών

Μηχανισμός Εργασίας του Jsonwebtoken

Λειτουργία Logout

## Κεφάλαιο 3: Υλοποίηση

3.1 Περιβάλλον Ανάπτυξης

Visual Studio Code (VSCode)

Πλεονεκτήματα του VSCode για την Ανάπτυξη με Node.js

Yarn

pgAdmin 4

3.2 Τεχνολογίες και Εργαλεία που Χρησιμοποιήθηκαν:

3.2.1 Node.js

Χαρακτηριστικά της Node.js

Χρήση της Node.js στο Έργο

3.2.2 Express.js

Χαρακτηριστικά του Express.js

Ρόλος του Express.js στο Έργο

3.2.3 Sequelize ORM

Χαρακτηριστικά του Sequelize

Χρήση του Sequelize στο Έργο

Sequelize Migrations

Χαρακτηριστικά των Migrations	
Χρήση των Migrations στο Έργο	
Sequelize-auto	
Χαρακτηριστικά του sequelize-auto	
Χρήση του sequelize-auto στο Έργο	
<b>3.2.4 PostgreSQL</b>	
Χαρακτηριστικά της PostgreSQL	
Επιλογή της PostgreSQL για το Έργο	
<b>3.2.5 Άλλες Εξαρτήσεις Εφαρμογής</b>	
Εξαρτήσεις ανάπτυξης εφαρμογής (devDependencies)	
Εξαρτήσεις λειτουργίας εφαρμογής (dependencies)	
<b>3.3.1 Παραλαβές (Εισροές)   Oil Receptions (Inflows)</b>	
Στοιχεία που Εμπλέκονται	
Δημιουργία μιας Νέας Παραλαβής	
Διαχείριση των Παραλαβών	
Επικύρωση Παραλαβών	
Παραδείγματα:	
<b>3.3.2 Φορτώσεις (Εκροές)   Oil Loadings (Outflows)</b>	
Στοιχεία που Εμπλέκονται	
Δημιουργία μιας Νέας Φόρτωσης	
Διαχείριση των Φορτώσεων	
Επικύρωση Φορτώσεων	
Παραδείγματα Λειτουργιών	
<b>3.3.3 Μεταφορές   Oil Transfers</b>	
Στοιχεία που Εμπλέκονται	
Δημιουργία μιας Νέας Transfer	
Διαχείριση των Transfer	
<b>3.3.4 Χημικές Αναλύσεις   Oil Chemical Analyses</b>	
Στοιχεία που Εμπλέκονται	
Δημιουργία μιας Νέας Χημικής Ανάλυσης	
Διαχείριση των Χημικών Αναλύσεων	
Παραδείγματα Λειτουργιών	
<b>3.3.5 Δειγματοληψία Λαδιού   Oil Sampling</b>	
Στοιχεία που Εμπλέκονται	
Δημιουργία μιας Νέας SampleCard	
Διαχείριση των SampleCard	
Παραδείγματα Λειτουργιών	
<b>3.3.6 Διαχείριση Δεξαμενών</b>	
Στοιχεία που Εμπλέκονται	
Δημιουργία μιας Νέας Δεξαμενής	
Λειτουργίες Διαχείρισης των Δεξαμενών	
<b>3.4 Υλοποίηση Βοηθητικών Λειτουργιών</b>	
<b>3.4.1 Αρχείο Πελατών/Προμηθευτών</b>	
Στοιχεία που Εμπλέκονται	
Δημιουργία μιας Νέας Οντότητας Πελάτη/Προμηθευτή	
Λειτουργίες Διαχείρισης των Πελατών/Προμηθευτών	
Μηχανισμός Ψευδώνυμων (Alias)	
<b>3.4.2 Αρχείο Χρηστών</b>	
Στοιχεία που Εμπλέκονται	
Δημιουργία μιας Νέας Οντότητας Χρήστη	
Λειτουργίες Διαχείρισης των Χρηστών	
Μηχανισμός Υπογραφής (Signature Mechanism)	
<b>3.5 Υλοποίηση Λειτουργιών Συστήματος</b>	
<b>3.5.1 Event Sourcing: Περιγραφή</b>	
Σενάρια που οδηγούν σε λανθάνουσα κατάσταση:	
Θεωρία του Event Sourcing	

Πλεονεκτήματα του Event Sourcing

Εφαρμογή του Event Sourcing στο Υπάρχον Σύστημα

Παράδειγμα Εφαρμογής:

Συνοπτικά

### 3.5.2 **Event Sourcing:** Υλοποίηση

Ανάλυση Κατάστασης:

Υλοποίηση:

Δημιουργία Νέων Πινάκων

Υπηρεσίες (Services)

Μονάδα EventHandler

Χρήση του EventHandler

### 3.5.3 **Προγραμματισμένες Εργασίες Συστήματος**

TokenBlacklistJob

Προτάσεις για Άλλες Τακτικές Εργασίες

## **Κεφάλαιο 4: Συμπεράσματα και Μελλοντική Εργασία**

4.1 Περίληψη του Έργου

4.2 Συνεισφορές της Μελέτης

4.3 Περιορισμοί

4.4 Συστάσεις για Μελλοντική Εργασία

Αναφορές:

---

# Περίληψη

## Επισκόπηση του Έργου

Η παρούσα αναφορά περιγράφει τη λογική και υλοποίηση του backend μιας εφαρμογής για τη διαχείριση των λειτουργιών αποθήκευσης, σε μια εταιρεία εμπορίας ελαιολάδου. Η υλοποίηση του backend προσφέρει ένα API, που μπορεί να χρησιμοποιηθεί από ανάλογα σχεδιασμένο και υλοποιημένο frontend για την αλληλεπίδραση με τη βάση δεδομένων της εφαρμογής. Η εφαρμογή χρησιμοποιεί μια σχεσιακή βάση δεδομένων *PostgreSQL* για την καταγραφή και διαχείριση των δεδομένων που σχετίζονται με την αποθηκευτική λειτουργία της επιχείρησης.

Η αρχιτεκτονική του συστήματος είναι διαχωρισμένη σε δύο ξεχωριστές υπηρεσίες:

- **Migration-service:** Αυτή η υπηρεσία χρησιμοποιεί Sequelize ORM migrations για τον ορισμό και τη διαχείριση του σχήματος της βάσης δεδομένων (*PostgreSQL*). Μέσω αυτής της υπηρεσίας, μπορούμε να κρατάμε ιστορικό των αλλαγών στο σχήμα της βάσης αλλά και τη δυνατότητα να κάνουμε rollback σε πρότερη κατάσταση.
- **API-service:** Αυτή είναι η κύρια υπηρεσία του συστήματος και υποστηρίζει το API. Περιλαμβάνει τη λογική του backend και επιτρέπει στους χρήστες να αλληλεπιδρούν με τη βάση δεδομένων μέσω των API routes, υποστηρίζοντας πλήρως τις λειτουργίες CRUD (Create, Read, Update, Delete).

Οι ενέργειες τις οποίες υλοποιεί η εφαρμογή έχουν να κάνουν με τις ροές του λαδιού (εισροές - εκροές - μεταφορές), τις χημικές αναλύσεις που δημιουργούνται ή συνοδεύουν το εκάστοτε φορτίο ελαιόλαδου, τις αποθηκευτικές υποδομές (δεξαμενές - ντίνες), καθώς και τα δείγματα και τη δειγματοληψία που τελείται πάνω στα φορτία.

Οι κύριες λειτουργίες που υποστηρίζει το σύστημα περιλαμβάνουν:

- **Oil Receptions (Inflows):** Καταγραφή παραλαβών λαδιού από διάφορους προμηθευτές.
- **Oil Loadings (Outflows):** Καταγραφή φορτώσεων λαδιού σε βυτιοφόρα για παράδοση σε πελάτες.
- **Oil Transfers:** Μεταφορές λαδιού μεταξύ δεξαμενών εντός των εγκαταστάσεων αποθήκευσης.
- **Oil Chemical Analyses:** Καταγραφή των αποτελεσμάτων χημικών αναλύσεων που πραγματοποιούνται στο λάδι.
- **Oil Sampling:** Καταγραφή δειγμάτων λαδιού για αρχειοθέτηση ή αποστολή σε πιθανούς πελάτες.
- **Registry of Tanks:** Παρακολούθηση της κατάστασης των δεξαμενών αποθήκευσης.

Επιπλέον, το σύστημα περιλαμβάνει βοηθητικές λειτουργίες όπως:

- **Registry of Business Partners:** Μητρώο πελατών και προμηθευτών.
- **Registry of System Users:** Μητρώο χρηστών του συστήματος με διακριτούς ρόλους (admins και simple-users).

Οι κύριες τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής είναι οι εξής: Node.js, Express.js, Sequelize ORM και PostgreSQL, μαζί με διάφορες άλλες βιβλιοθήκες όπως bcrypt, express-validator και jsonwebtoken.

Το σύστημα χτίστηκε έχοντας σαν προτεραιότητες την αποδοτικότητα και τη φιλικότητα προς τον χρήστη, είναι προσανατολισμένο σε συγκεκριμένα σενάρια χρήσης, έχοντας υπόψιν συγκεκριμένους "τύπους" χρηστών και βασίζεται εν πολλοίς στην εμπειρία του συγγραφέα στον συγκεκριμένο κλάδο.

**Το πρόβλημα που προσπαθεί να λύσει είναι αρκετά εξειδικευμένο και ως εκ τούτου οι λύσεις που παρουσιάζονται είναι με τη σειρά τους εξειδικευμένες.**

Συνοπτικά, η εφαρμογή διαχείρισης αποθήκευσης που αναπτύχθηκε στο πλαίσιο αυτής της διπλωματικής εργασίας στοχεύει στην αυτοματοποίηση και βελτιστοποίηση των διαδικασιών διαχείρισης λαδιού στις εγκαταστάσεις αποθήκευσης, προσφέροντας μια ολοκληρωμένη και αξιόπιστη λύση για τους χρήστες της.

## Βασικοί Στόχοι

Κύριος στόχος του συστήματος είναι να ψηφιοποιήσει καταγραφές που μέχρι τώρα τελούνται χειρόγραφα, να παρέχει στους χρήστες μία εικόνα για την αποθήκη και το προϊόν με τρόπο αυτοματοποιημένο και πιο αποτελεσματικό, να ενισχύσει τον έλεγχο των χρηστών στα δεδομένα και να τους προστατεύσει από λάθη και παραβλέψεις.

Συγκεκριμένα το σύστημα παρέχει:

- Εύκολη πρόσβαση στην πληροφορία
- Διατήρηση της πληροφορίας εφ' όρου ζωής
- Εύκολη και φιλική αλλά και εν δυνάμει εξεζητημένη αναζήτηση πληροφορίας
- Ελέγχους, που σκοπό έχουν να καθοδηγήσουν τον χρήστη στη σωστή τήρηση των δεδομένων
- Επισκόπηση του περιβάλλοντος της αποθήκης από διάφορες οπτικές
- Αυτοματοποιημένες συσχετίσεις μεταξύ δεδομένων

## Κύρια Ευρήματα και Συνεισφορές

Η διαδικασία ανάπτυξης του συγκεκριμένου λογισμικού αποδείχτηκε για τον συγγραφέα τόσο εποικοδομητική όσο και επίπονη. Προβλήματα που αρχικά εμφανίζονταν τετριμμένα, όταν έφτανε η στιγμή υλοποίησης της λύσης τους, αποδεικνύονταν ιδιαίτερος περίπλοκα. Η σχεδίαση του λογισμικού αρκετές φορές άλλαξε για να μπορέσει να ξεπεράσει τους προαναφερθέντες "σκόπελους".

Γενικότερα, η ανάπτυξη του παρόντος λογισμικού δε στόχευε στην καινοτομία αλλά στην ενσωμάτωση εγνωσμένων, υφιστάμενων τεχνολογιών και στη σύζευξή τους, με απώτερο σκοπό τη δημιουργία ενός συστήματος που θα καλύπτει τις προδιαγραφές του.

Τα κύρια ευρήματα και οι συνεισφορές της εργασίας αυτής περιλαμβάνουν:

### 1. Ανάλυση των Αδυναμιών του Προϋπάρχοντος Χειρόγραφου Συστήματος:

- Αναλύθηκαν οι κύριες αδυναμίες της χειρόγραφης καταγραφής που χρησιμοποιούσε η εταιρεία, όπως η ασυνέπεια των δεδομένων, η δυσκολία στην ανάκτηση πληροφοριών και η απουσία συστήματος ελέγχου της ορθότητας των δεδομένων.
- Παρασχέθηκαν παραδείγματα και σενάρια που δείχνουν τις συνέπειες αυτών των αδυναμιών, όπως η δυσκολία στην παρακολούθηση των παραλαβών, των φορτώσεων και των μεταφορών.

## **2. Ανάπτυξη και Υλοποίηση των Βασικών και Βοηθητικών Λειτουργιών:**

- Αναπτύχθηκαν και υλοποιήθηκαν οι βασικές λειτουργίες του συστήματος, όπως οι παραλαβές (inflows), οι φορτώσεις (outflows), οι μεταφορές (transfers) και οι χημικές αναλύσεις (chemical analyses).
- Υλοποιήθηκαν οι βοηθητικές λειτουργίες, όπως η καταγραφή των πελατών/προμηθευτών (registry of business partners) και των χρηστών του συστήματος (registry of system users).

## **3. Ανάπτυξη Προηγμένων Τεχνικών Διαχείρισης Αποθήκευσης:**

- Η εργασία αυτή παρουσιάζει την ανάπτυξη ενός προηγμένου συστήματος διαχείρισης αποθήκευσης λαδιού, το οποίο βασίζεται σε σύγχρονες τεχνολογίες όπως το Node.js, το Express.js, το Sequelize ORM και το PostgreSQL.
- Η υλοποίηση των βασικών και βοηθητικών λειτουργιών του συστήματος συμβάλλει στη βελτίωση της διαχείρισης των αποθεμάτων, των παραλαβών, των φορτώσεων και των χημικών αναλύσεων του λαδιού.

## **4. Υιοθέτηση και Εφαρμογή του Event Sourcing:**

- Παρουσιάστηκε η θεωρία του event sourcing ως λύση για την αντιμετώπιση των αδυναμιών του χειρόγραφου συστήματος.
- Αναλύθηκε το πώς η υιοθέτηση του event sourcing μπορεί να βελτιώσει την αξιοπιστία και την ακρίβεια των δεδομένων, παρέχοντας πλήρες ιστορικό των αλλαγών και τη δυνατότητα αναπαραγωγής της κατάστασης του συστήματος σε οποιοδήποτε χρονικό σημείο.
- Υλοποιήθηκε ένας μηχανισμός event sourcing στο σύστημα, περιλαμβάνοντας τη δημιουργία των πινάκων EventLog και InflowTankSnapshot, καθώς και την ανάπτυξη των υπηρεσιών EventLogService και SnapshotService.

## **5. Βελτίωση της Αξιοπιστίας και της Ακρίβειας των Δεδομένων:**

- Με την εισαγωγή του event sourcing, η εργασία αυτή συμβάλλει στη βελτίωση της αξιοπιστίας και της ακρίβειας των δεδομένων, παρέχοντας τη δυνατότητα αναδρομικών ενημερώσεων και αναιρέσης αλλαγών χωρίς να επηρεάζεται η συνοχή της βάσης δεδομένων.
- Η καταγραφή κάθε αλλαγής ως γεγονός επιτρέπει την ακριβή αναπαραγωγή της κατάστασης του συστήματος σε οποιοδήποτε χρονικό σημείο.

## **6. Εισαγωγή Νέων Μηχανισμών Παρακολούθησης και Ελέγχου:**

- Η εισαγωγή των μηχανισμών καταγραφής γεγονότων και snapshots επιτρέπει την παρακολούθηση και τον έλεγχο της κατάστασης του συστήματος με μεγαλύτερη ακρίβεια και διαφάνεια.
- Η δυνατότητα αποθήκευσης και ανάκτησης των δεδομένων σε καθορισμένα χρονικά διαστήματα παρέχει τη δυνατότητα ιστορικής αναπαραγωγής και auditing.

## **7. Συμβολή στη Θεωρία και Πρακτική της Ανάπτυξης Λογισμικού:**

- Η εργασία αυτή συμβάλλει στη θεωρία και πρακτική της ανάπτυξης λογισμικού, παρουσιάζοντας μια ολοκληρωμένη προσέγγιση για την υλοποίηση ενός συστήματος διαχείρισης αποθήκευσης βασισμένου στο event sourcing.
- Παρέχει παραδείγματα και καλές πρακτικές για την υλοποίηση και τη διαχείριση συστημάτων αποθήκευσης και logistics.

Συνολικά, η διπλωματική αυτή εργασία παρέχει σημαντικά ευρήματα και συνεισφορές στη θεωρία και την πρακτική της ανάπτυξης λογισμικού, παρουσιάζοντας ένα προηγμένο σύστημα διαχείρισης αποθήκευσης λαδιού που αντικαθιστά τις παραδοσιακές χειρόγραφες μεθόδους και υποστηρίζει βελτιωμένη αξιοπιστία και ακρίβεια δεδομένων.



# Κεφάλαιο 1: Εισαγωγή

## 1.1 Ιστορικό και Πλαίσιο

Η παρούσα διπλωματική εργασία επικεντρώνεται στην ανάπτυξη ενός συστήματος διαχείρισης αποθήκευσης για μια εταιρεία που δραστηριοποιείται στο εμπόριο ελαιολάδου. Η εταιρεία, η οποία έχει καθιερωθεί ως ένας αξιόπιστος προμηθευτής και διανομέας υψηλής ποιότητας ελαιολάδου, διαχειρίζεται μεγάλες ποσότητες προϊόντος που αποθηκεύεται σε διάφορες δεξαμενές.

Η ανάγκη για ένα σύστημα διαχείρισης αποθήκευσης προέκυψε από την ανάγκη για αποτελεσματική παρακολούθηση και καταγραφή των λειτουργιών αποθήκευσης, όπως οι παραλαβές, οι φορτώσεις, οι μεταφορές και οι χημικές αναλύσεις. Το υπάρχον σύστημα καταγραφής είναι χειρόγραφο, γεγονός που οδηγεί σε δυσκολίες στην ανάκτηση πληροφοριών, ασυνέπειες στα δεδομένα και προβλήματα ασφάλειας. Ως εκ τούτου, η ανάπτυξη ενός ψηφιακού συστήματος διαχείρισης αποθήκευσης κρίθηκε απαραίτητη για την ενίσχυση της ακρίβειας, της αποτελεσματικότητας και ασφάλειας των λειτουργιών αποθήκευσης.

Οι λειτουργίες που τελούνται σε μία αποθήκη εταιρείας εμπορίας ελαιόλαδου και που αφορούν το σύστημα μας είναι πέντε τον αριθμό, ας τις δούμε μία προς μία:

- **Παραλαβή:** Κατά την παραλαβή το φορτίου λαδιού φτάνει στις εγκαταστάσεις, ζυγίζεται, μετράται / αναλύεται ως προς τα ποιοτικά του χαρακτηριστικά και τοποθετείται στον κατάλληλο ταμιευτήρα, έναν ή περισσότερους. Αυτός μπορεί να είναι ταμιευτήρας προσωρινής αποθήκευσης(ντίνα) ή δεξαμενή.
- **Μεταφορά:** Για να γίνει μεταφορά το λάδι βρίσκεται ήδη στις εγκαταστάσεις, μέσα σε κάποιο ταμιευτήρα και μέσα από το δίκτυο σωληνώσεων της αποθήκης θέλουμε να το μεταφέρουμε σε κάποιον άλλο ταμιευτήρα. Αυτό μπορεί να συμβεί είτε διότι θέλουμε να αδειάσουμε τον ταμιευτήρα προέλευσης, είτε για να συμπληρώσουμε τον ταμιευτήρα προορισμού.
- **Φόρτωση:** Στη διαδικασία της φόρτωσης, το λάδι που βρίσκεται σε έναν ή περισσότερους ταμιευτήρες, εντός των εγκαταστάσεων, φορτώνεται σε βυτία/οχήματα και φεύγει των εγκαταστάσεων κυρίως λόγω πώλησης. Σε αυτή τη λειτουργία μπορεί να συμπεριληφθεί και η υπολειτουργία της *Εξαποθήκευσης* στην οποία πάλι έχουμε εκροή ελαιόλαδου αλλά με τη διαφορά ότι πρόκειται για μικροποσότητες που προορίζονται για λιανική πώληση και δεν περιλαμβάνει φορτηγά-βυτία.
- **Χημική Ανάλυση:** Η χημ. ανάλυση είναι ο ποιοτικός έλεγχος του ελαιόλαδου και πάντα αφορά μία ή περισσότερες συγκεκριμένες παρτίδες ελαιόλαδου. Η ανάλυση μπορεί να γίνει σε οποιαδήποτε φάση του κύκλου ζωής του λαδιού μέσα στην αποθήκη ακόμα και σε λάδι το οποίο έχει εγκαταλείψει τις εγκαταστάσεις ή δεν έχει αφιχθεί ακόμα. Η χημική ανάλυση μπορεί να λάβει χώρα είτε στο χημικό εργαστήριο της εταιρείας, είτε στο χώρο της παραλαβής.
- **Δειγματοληψία:** Αντίστοιχα με την χημ. ανάλυση η δειγματοληψία επίσης συμβαίνει σε οποιαδήποτε φάση του κύκλου ζωής του λαδιού μέσα στην αποθήκη. Σκοπός της είναι να κρατάει ένα αρχείο των λαδιών που εμπορεύεται η εταιρεία για λόγους ποιοτικού ελέγχου αλλά και προώθησης του προϊόντος.

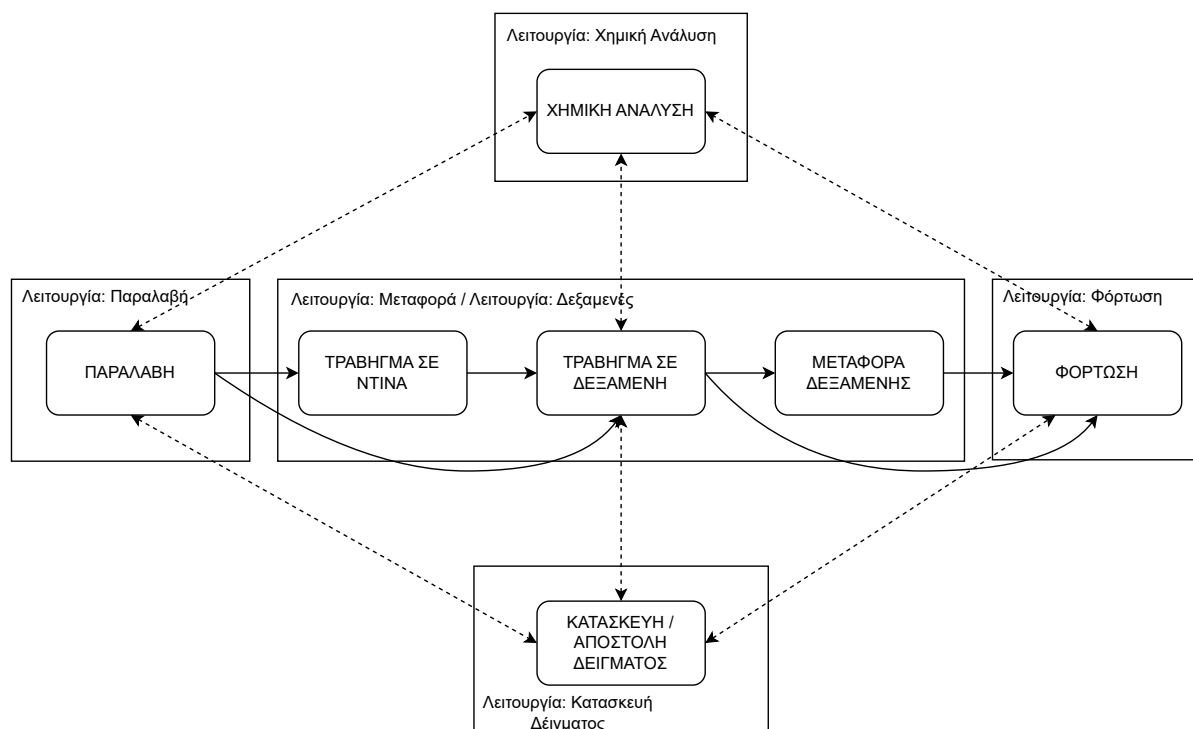


Figure 1.1: Η ροή του ελαιολάδου στην αποθήκη

Η εισροή του ελαιολάδου στην αποθήκη γίνεται πάντα κατά την παραλαβή (εισροή), οπότε το ελαιόλαδο μπορεί να αποθηκευτεί ή σε προσωρινούς ταμιευτήρες (ντίνες) ή απευθείας στις δεξαμενές. Κατά τη διάρκεια παραμονής του στην αποθήκη, ποσότητες ελαιολάδου μπορούν να μεταφερθούν σε διαφορετική δεξαμενή από αυτή που αρχικά αποθηκεύτηκε. Τέλος όλα τα λάδια κάποια στιγμή “φεύγουν” (εκροή) με τη διαδικασία της φόρτωσης.

Η χημική ανάλυση και η κατασκευή δείγματος είναι διεργασίες που μπορεί να πραγματοποιηθούν καθ’ όλα τα στάδια της ροής του ελαιολάδου, και να αφορούν σε λάδια τα οποία **δε** βρίσκονται στην αποθήκη της εταιρείας.

## 1.2 Περιγραφή του Προβλήματος

Επί του παρόντος η καταγραφή που γίνεται στις αναφερθείσες στην προηγούμενη ενότητα λειτουργίες, είναι στην ολότητά της χειρόγραφη. Ακολουθούν αναλυτικά τα βήματα που ακολουθούνται από τον αποθηκάριο σε κάθε μία από τις λειτουργίες:

- Παραλαβή:** Ο αποθηκάριος καταγράφει τα στοιχεία του προμηθευτή σε διπλότυπο έντυπο (εσωτερικής χρήσης της εταιρείας), ζυγίζει το όχημα / βυτίο που φέρει το φορτίο και καταγράφει το μεικτό. Στην συνέχεια συλλέγει δείγμα από το φορτίο και κάνει οξυμέτρηση ή και περαιτέρω χημικές αναλύσεις στο δείγμα. Από τα αποτελέσματα των αναλύσεων αποφαίνεται σε ποιο ταμιευτήρα θα “ξεφορτώσει” το φορτίο και ξεκινάει τη διαδικασία άντλησης του φορτίου στον ταμιευτήρα. Όταν ολοκληρωθεί η διαδικασία εκφόρτωσης το όχημα / βυτίο ζυγίζεται εκ νέου και καταγράφονται οι ποσότητες του καθαρού φορτίου και του απόβαρου καθώς και το σε ποιον ταμιευτήρα μπήκε το φορτίο. Τέλος δίνεται στον προμηθευτή το ένα απόκομμα από το διπλότυπο έντυπο, με το οποίο μεταβαίνει στο λογιστήριο της εταιρείας για τη σύνταξη του τιμολογίου. Ο αποθηκάριος πρέπει

επίσης να ενημερώσει το "τετράδιο των δεξαμενών" <sup>1</sup> με τα στοιχεία της παραλαβής που έκανε στην ενότητα του κατάλληλου ταμιευτήρα.

- **Μεταφορά:** Στην μεταφορά ο αποθηκάρχιος πρέπει να ενημερώσει "τετράδιο των δεξαμενών" <sup>1</sup> με την ποσότητα που μεταγγίστηκε από τη δεξαμενή Α στη δεξαμενή Β. Οι ενέργεια αυτή έχει ως αποτέλεσμα το "κρύψιμο" πληροφορίας καθώς δεν είναι δυνατό / πρακτικό όλες οι πληροφορίες που αφορούν τις παραλαβές που υπήρχαν στην ενότητα της δεξαμενής αφετηρίας, να μεταφερθούν στην ενότητα της δεξαμενής προορισμού.

IANΘΟΣ ΑΕ

ΕΝΤΥΠΟ ΕΛΕΓΧΟΥ ΠΑΡΑΛΑΒΗΣ  
ΕΛΛΙΟΛΑΔΟΥ

ΔΙΑΣΦΑΛΙΣΗ  
ΠΟΙΟΤΗΤΑΣ

Ημερομηνία Παραλαβής: .....

Ονοματεπώνυμο Προμηθευτή: .....

Δόση Προμηθευτή: .....

ΔΕΞΑΜΕΝΗ Α ΠΡΟΗΓΟΥΣΗΣ

Ποσότητα Παραλαβής: .....

ΑΠΟΤΕΛΕΣΜΑΤΑ ΕΛΕΓΧΟΥ

Αριθμός Αντικειμένου: .....

Οξύτητα (%) .....

Δείκτης Υπεροξειδίου (mEq O<sub>2</sub>/kg) .....

K<sub>232</sub> .....

K<sub>228</sub> .....

Χλωτοστερόλη (%) .....

Βροστασιστερόλη (%) .....

Καρμωστερόλη (%) .....

Στηγμαστερόλη (%) .....

β-επιστερόλη (%) .....

δ-7-στηγμαστερόλη (%) .....

Συνολικές στερόλες (mg/kg) .....

Ερυθροδύλη & στερόλη (%) .....

ECN 42 (HPLC) – ECN42 .....

Υγρασία (%) .....

ΔΕΞΑΜΕΝΗ Β ΠΡΟΟΡΙΣΜΟΥ

Εργαστήριο: .....

Figure 1.2a: Φόρμα Χημικών Αναλύσεων

IANΘΟΣ ΑΕ

ΗΜΕΡΟΜΗΝΙΑ: .....

ΔΕΛΤΙΟ ΠΑΡΑΛΑΒΗΣ  
(ΓΙΑ ΕΣΩΤΕΡΙΚΗ ΧΡΗΣΗ)

7899

Ο Κ. .... Α.Φ.Μ. ....

ΕΠΑΓΓΕΛΜΑ ..... ΔΩΣΗ .....

ΕΙΔΟΣ	ΠΟΣΟΤ.	ΟΞΥΤΗΤΑ	ΤΙΜΗ	ΑΞΙΑ
Μικτό				
Απόβαρο				
Καθαρό				
Μικτό				
Απόβαρο				
Καθαρό				
Μικτό				
Απόβαρο				

Ο ΠΑΡΑΛΑΒΩΝ

Figure 1.2b: Έγγραφο Παραλαβής

- **Φόρτωση:** Στη διαδικασία της φόρτωσης ο αποθηκάρχιος καταγράφει τα στοιχεία του οχήματος-βυτίου σε διπλότυπο έντυπο (εσωτερικής χρήσης της εταιρείας), τα στοιχεία προορισμού (πελάτη) του φορτίου καθώς και το απόβαρο του οχήματος. Το όχημα φορτώνεται από μία ή περισσότερες δεξαμενές, κατόπιν καταγράφεται το μεικτό και καθαρό βάρος, εξάγεται δείγμα από το φορτίο και τέλος το βυτίο σφραγίζεται. Ο οδηγός παραλαμβάνει ένα απόκομμα από το διπλότυπο και μεταβαίνει στο λογιστήριο για να τακτοποιήσει τα υπόλοιπα συνοδευτικά του φορτίου έγγραφα.
- **Χημική Ανάλυση:** Ανάλογα με το ποιο είναι και σε ποια φάση βρίσκεται το φορτίο το οποίο αφορά, η χημ. ανάλυση μπορεί να καταγραφεί: είτε στο έντυπο της παραλαβής, είτε στο "τετράδιο των δεξαμενών" <sup>1</sup>, είτε σε εξειδικευμένη φόρμα "Χημικών Αναλύσεων", είτε σε αρχείο του χημείου της εταιρείας.
- **Δειγματοληψία:** Τα δείγματα των φορτίων καταγράφονται σε ειδικές φόρμες "Δειγμάτων" οι οποίες φυλάσσονται σε αρχεία (ντοσιέ) δειγμάτων.

## 1.3 Στόχοι της Μελέτης

### Αδυναμίες Υπάρχοντος Συστήματος

Από την περιγραφή των εργασιών καταγραφής των λειτουργιών της αποθήκης όπως αυτή έγινε στην ενότητα [1.2](#), εύκολα γίνονται αντιληπτές "αδυναμίες" του υπάρχοντος συστήματος καταγραφής. Οι κύριες είναι:

- Η "κατάσταση" της αποθήκης βρίσκεται σε πολλά και ασύνδετα μεταξύ τους έγγραφα, τα οποία και ενημερώνονται το κάθε ένα ξεχωριστά.
- Πολλοί διαφορετικοί χρήστες μπορούν να έχουν διαφορετικούς τρόπους καταγραφής/"έκφρασης" της πληροφορίας με αποτέλεσμα να μην είναι πάντα ευανάγνωστη, ή να είναι λανθασμένη η ανάγνωση της πληροφορίας από διαφορετικούς χρήστες.
- Ανυπαρξία συστήματος ελέγχου της ορθότητας των δεδομένων κατά την καταγραφή τους.
- Δυσκολία στην ανάκτηση της πληροφορίας: Σενάρια που ο χρήστης πρέπει να ελέγξει διαφορετικά έγγραφα και διαφορετικές ενότητες στο ίδιο έγγραφο για να εξαγάγει συμπέρασμα.
- Θέματα που έχουν να κάνουν με την ασφάλεια και τη συνέπεια των δεδομένων.
- Δυσκολία στην πρόσβαση των δεδομένων: Απαιτείται φυσική παρουσία στον χώρο που βρίσκεται το εκάστοτε έγγραφο

Τα παραπάνω είναι κάποια από τα προβλήματα στα οποία το παρόν σύστημα καλείται να δώσει λύσεις.

### Κύριοι Στόχοι Του Έργου

Ο κύριος στόχος του παρόντος εγχειρήματος είναι η ανάπτυξη ενός ψηφιακού συστήματος διαχείρισης αποθήκευσης λαδιού που θα αντικαταστήσει το υπάρχον χειρόγραφο σύστημα καταγραφής, επιδιώκοντας να αντιμετωπίσει τις αδυναμίες και τα προβλήματα που εντοπίστηκαν. Οι συγκεκριμένοι στόχοι της μελέτης περιλαμβάνουν:

#### 1. Αυτοματοποίηση των Λειτουργιών Καταγραφής:

- Ανάπτυξη ενός ψηφιακού συστήματος που θα καταγράφει τις παραλαβές, τις φορτώσεις, τις μεταφορές και τις χημικές αναλύσεις του λαδιού.
- Μείωση του χρόνου και των σφαλμάτων που σχετίζονται με τη χειρόγραφη καταγραφή δεδομένων.

#### 2. Ενίσχυση της Ακρίβειας και της Συνέπειας των Δεδομένων:

- Διασφάλιση ότι όλα τα δεδομένα καταγράφονται με ακρίβεια και συνέπεια, αποτρέποντας την ύπαρξη ασυνεπειών που παρατηρούνται στο χειρόγραφο σύστημα.
- Εισαγωγή ελέγχων για την επαλήθευση και την επικύρωση των δεδομένων κατά την καταχώρηση τους.

#### 3. Βελτίωση της Ανάκτησης Πληροφοριών:

- Ανάπτυξη ενός συστήματος που θα επιτρέπει την εύκολη και γρήγορη ανάκτηση πληροφοριών, χωρίς την ανάγκη αναζήτησης σε πολλά και ασύνδετα έγγραφα.
- Παροχή δυνατοτήτων αναζήτησης και φιλτραρίσματος δεδομένων, διευκολύνοντας την εξαγωγή χρήσιμων συμπερασμάτων.

#### 4. Αύξηση της Ασφάλειας και της Συνέπειας των Δεδομένων:

- Διασφάλιση της ασφάλειας των δεδομένων μέσω μηχανισμών αυθεντικοποίησης και ελέγχου πρόσβασης.

- Ανάπτυξη μηχανισμών αντιγράφων ασφαλείας και αποκατάστασης δεδομένων για την προστασία από απώλεια δεδομένων.

#### 5. Δημιουργία Λειτουργιών Αναφορών και Αναλύσεων:

- Παροχή εργαλείων για τη δημιουργία αναφορών και την ανάλυση δεδομένων, παρέχοντας πολύτιμες πληροφορίες για τη διαχείριση των αποθεμάτων και τη λήψη αποφάσεων.
- Χρήση εργαλείων visual analytics για την παρουσίαση των δεδομένων με γραφήματα και διαγράμματα.

Οι στόχοι αυτοί αποσκοπούν στη δημιουργία ενός ολοκληρωμένου και αποδοτικού συστήματος διαχείρισης αποθήκευσης, που θα καλύπτει πλήρως τις ανάγκες της εταιρείας εμπορίας ελαιολάδου, παρέχοντας παράλληλα αυξημένη ακρίβεια, ασφάλεια και ευχρηστία.

## 1.4 Πεδίο του Έργου

Το παρόν σύστημα δε φιλοδοξεί να αλλάξει τους τρόπους με τους οποίους περαιώνονται οι λειτουργίες της αποθήκης. Ασχολείται αποκλειστικά με την καταγραφή αυτών και τη διατήρηση των δεδομένων που παράγονται από τις λειτουργίες.

Ακόμη, το σύστημα δεν απαιτεί, ούτε εισάγει επιπλέον hardware (συσκευές, σένσορες) για να λειτουργήσει πέρα από ένα συνηθισμένο ηλεκτρονικό υπολογιστή με πρόσβαση στον server που θα τρέχει το σύστημα.

Στόχος του έργου είναι η βελτίωση της διαχείρισης των διαδικασιών αποθήκευσης και η εξάλειψη των αδυναμιών του υπάρχοντος χειρόγραφου συστήματος. Το έργο περιλαμβάνει:

#### Καλυπτόμενες Πτυχές του Έργου:

##### 1. Ανάπτυξη του Συστήματος Διαχείρισης Αποθήκευσης:

- Σχεδιασμός και υλοποίηση της backend λογικής για την παρακολούθηση και καταγραφή των παραλαβών, φορτώσεων, μεταφορών και χημικών αναλύσεων.
- Χρήση των τεχνολογιών Node.js, Express.js, Sequelize ORM και PostgreSQL για την υλοποίηση του συστήματος.

##### 2. Υλοποίηση των Βασικών και Βοηθητικών Λειτουργιών:

- Ανάπτυξη των βασικών λειτουργιών για την καταγραφή των παραλαβών, των φορτώσεων και των μεταφορών.
- Υλοποίηση των βοηθητικών λειτουργιών για την καταγραφή των δεξαμενών, των πελατών/προμηθευτών και των χρηστών του συστήματος.

##### 3. Εφαρμογή του Event Sourcing:

- Εισαγωγή της θεωρίας του event sourcing και ανάπτυξη του μηχανισμού καταγραφής γεγονότων, για τη βελτίωση της ακρίβειας και της ακεραιότητας των δεδομένων.
- Δημιουργία των πινάκων EventLog και InflowTankSnapshot για την καταγραφή των γεγονότων και των καταστάσεων των δεξαμενών.

##### 4. Διαχείριση Χρηστών:

- Ανάπτυξη ενός συστήματος διαχείρισης χρηστών που θα επιτρέπει τη δημιουργία, ενημέρωση και διαγραφή χρηστών, καθώς και τον καθορισμό των δικαιωμάτων πρόσβασης.

## **Μη Καλυπτόμενες Πτυχές του Έργου:**

### **1. Ανάπτυξη Frontend Διεπαφής:**

- Το έργο αυτό επικεντρώνεται αποκλειστικά στην backend λογική του συστήματος και δεν περιλαμβάνει την ανάπτυξη frontend διεπαφής χρήστη.
- Οι λειτουργίες του συστήματος είναι προσβάσιμες μέσω API και προορίζονται για ενσωμάτωση με μελλοντικό frontend.

### **2. Λογιστική Διαχείριση:**

- Η υλοποίηση δεν περιλαμβάνει λειτουργίες λογιστικής διαχείρισης, όπως η έκδοση τιμολογίων και η διαχείριση πληρωμών.
- Το σύστημα παρέχει μόνο τις πληροφορίες που απαιτούνται για τη λογιστική διαχείριση μέσω της καταγραφής των παραλαβών και των φορτώσεων.

### **3. Εξωτερικές Διασυνδέσεις:**

- Το έργο δεν καλύπτει τη διασύνδεση με εξωτερικά συστήματα ή υπηρεσίες, όπως συστήματα ERP ή άλλες επιχειρησιακές εφαρμογές.
- Η ανάπτυξη διασυνδέσεων μπορεί να εξεταστεί σε μελλοντικές φάσεις του έργου.

Συνολικά, το έργο αυτό εστιάζει στην παροχή μιας ολοκληρωμένης λύσης για τη διαχείριση αποθήκευσης, καλύπτοντας τις βασικές ανάγκες της εταιρείας εμπορίας ελαιολάδου, ενώ αφήνει ανοιχτές δυνατότητες για μελλοντική επέκταση και ενσωμάτωση με άλλα συστήματα και λειτουργίες.

# Κεφάλαιο 2: Σχεδίαση και Αρχιτεκτονική Συστήματος

## 2.1 Ανάλυση Απαιτήσεων

### 2.1.1 Λειτουργικές Απαιτήσεις

Το σύστημα πρέπει να επιτρέπει στους χρήστες να καταγράφουν, να ανακτούν και να τροποποιούν τα δεδομένα που σχετίζονται με τις λειτουργίες της αποθήκης όπως αυτές παρουσιάστηκαν στην ενότητα [1.1](#), με τρόπο ασφαλή, ελεγχόμενο και διακριτό για την κάθε ξεχωριστή λειτουργία.

Ακολουθούν οι λειτουργικές απαιτήσεις της εφαρμογής διαχείρισης αποθήκευσης αναλυτικά για κάθε λειτουργία:

#### 1. Διαχείριση Χρηστών:

- **Εγγραφή και Αυθεντικοποίηση Admin Χρηστών:** Το σύστημα πρέπει να επιτρέπει την εγγραφή και την αυθεντικοποίηση των admin χρηστών. Οι admin χρήστες έχουν πλήρη έλεγχο του συστήματος, συμπεριλαμβανομένης της διαχείρισης άλλων χρηστών και της εκτέλεσης όλων των λειτουργιών.
- **Εγγραφή και Αυθεντικοποίηση Χρηστών:** Το σύστημα πρέπει να παρέχει μηχανισμό για την εγγραφή και την αυθεντικοποίηση κανονικών χρηστών (simple-users). Οι κανονικοί χρήστες μπορούν να εκτελούν συγκεκριμένες λειτουργίες με βάση τους ρόλους και τα δικαιώματά τους.
- **Role-Based Access Control:** Το σύστημα πρέπει να εφαρμόζει έλεγχο πρόσβασης βάσει ρόλων για να διασφαλίσει ότι οι χρήστες μπορούν να έχουν πρόσβαση μόνο σε λειτουργίες και δεδομένα που σχετίζονται με τους ρόλους τους.

#### 2. Διαχείριση Επιχειρηματικών Εταίρων:

- **Μητρώο Επιχειρηματικών Εταίρων:** Το σύστημα πρέπει να παρέχει λειτουργία για την εγγραφή και τη διαχείριση επιχειρηματικών εταίρων, συμπεριλαμβανομένων πελατών και προμηθευτών. Αυτό περιλαμβάνει την προσθήκη, ενημέρωση και διαγραφή πληροφοριών επιχειρηματικών εταίρων.
- **Κατηγοριοποίηση Επιχειρηματικών Εταίρων:** Το σύστημα πρέπει να επιτρέπει την κατηγοριοποίηση των επιχειρηματικών εταίρων ως 'customer' ή 'supplier' (ή και συνδυασμό των δύο) για τη διευκόλυνση της διαχείρισης διαφόρων τύπων συναλλαγών.

#### 3. Διαχείριση Εγκαταστάσεων Αποθήκευσης:

- **Μητρώο Δεξαμενών:** Το σύστημα πρέπει να παρέχει λειτουργία για την εγγραφή και τη διαχείριση δεξαμενών στις εγκαταστάσεις αποθήκευσης. Αυτό περιλαμβάνει την προσθήκη, ενημέρωση και διαγραφή πληροφοριών δεξαμενών.
- **Παρακολούθηση Δεξαμενών:** Το σύστημα πρέπει να παρακολουθεί και να καταγράφει τα φυσικά χαρακτηριστικά καθώς και την κατάσταση των δεξαμενών, συμπεριλαμβανομένων των τρεχουσών ποσοτήτων λαδιού που φιλοξενούνται καθώς και των ποιοτικών χαρακτηριστικών αυτών.

#### 4. Διαχείριση Παραλαβών Λαδιού:

- ο **Καταγραφή Παραλαβών Λαδιού:** Το σύστημα πρέπει να επιτρέπει την καταγραφή των παραλαβών λαδιού (inflows) από διάφορους προμηθευτές. Αυτό περιλαμβάνει την καταγραφή λεπτομερειών όπως η ημερομηνία παραλαβής, η ποσότητα λαδιού, οι πληροφορίες προμηθευτή και η συγκεκριμένη δεξαμενή όπου αποθηκεύεται το λάδι.
- ο **Επαλήθευση Παραλαβών Λαδιού:** Το σύστημα πρέπει να υποστηρίζει την επαλήθευση των παραληφθέντων λαδιών για να εξασφαλίζεται η ακρίβεια στην καταγραφή των εισροών.
- ο **Ιχνηλασιμότητα Παραλαβών Λαδιού:** Ο χρήστης θα πρέπει πέρα από τα παραπάνω, να έχει τη δυνατότητα να εξαγει λεπτομερή πληροφορία για όλα τα στάδια "ζωής" μίας παραλαβής, από την είσοδο της στις εγκαταστάσεις μέχρι και την αναχώρησή της, στην ολότητά αυτής <sup>2</sup>.

#### 5. Διαχείριση Φορτώσεων Λαδιού:

- ο **Καταγραφή Φορτώσεων Λαδιού:** Το σύστημα πρέπει να επιτρέπει την καταγραφή των φορτώσεων λαδιού (outflows) όταν το λάδι φορτώνεται σε βυτιοφόρα για παράδοση σε πελάτες. Αυτό περιλαμβάνει την καταγραφή λεπτομερειών όπως η ημερομηνία φόρτωσης, η ποσότητα λαδιού, οι πληροφορίες πελάτη και η συγκεκριμένη δεξαμενή από την οποία φορτώθηκε το λάδι.
- ο **Επαλήθευση Φορτώσεων Λαδιού:** Το σύστημα πρέπει να υποστηρίζει την επαλήθευση των φορτωμένων λαδιών για να εξασφαλίζεται η ακρίβεια στην καταγραφή των εκροών.

#### 6. Διαχείριση Μεταφορών Λαδιού:

- ο **Καταγραφή Μεταφορών Λαδιού:** Το σύστημα πρέπει να επιτρέπει την καταγραφή των μεταφορών λαδιού μεταξύ δεξαμενών στις εγκαταστάσεις αποθήκευσης. Αυτό περιλαμβάνει την καταγραφή λεπτομερειών όπως η ημερομηνία μεταφοράς, η ποσότητα λαδιού που μεταφέρθηκε και οι πηγές και προορισμοί δεξαμενών.

#### 7. Διαχείριση Χημικών Αναλύσεων Λαδιού:

- ο **Καταγραφή Χημικών Αναλύσεων:** Το σύστημα πρέπει να παρέχει λειτουργία για την καταγραφή των αποτελεσμάτων των χημικών αναλύσεων που πραγματοποιούνται στο λάδι. Αυτό περιλαμβάνει την καταγραφή λεπτομερειών όπως η ημερομηνία της ανάλυσης, τα αποτελέσματα της ανάλυσης και τυχόν σχετικές παρατηρήσεις.
- ο **Συσχέτιση Χημικών Αναλύσεων:** Το σύστημα πρέπει να παρέχει λειτουργία με την οποία ο χρήστης θα μπορεί να συσχετίσει τις χημ. αναλύσεις με ένα ή περισσότερα στιγμιότυπα των υπόλοιπων κύριων λειτουργιών του συστήματος, ονομαστικά: Παραλαβές, Φορτώσεις, Δεξαμενές, Δείγματα.

#### 8. Διαχείριση Δειγματοληψίας Λαδιού:

- ο **Καταγραφή Δειγμάτων Λαδιού:** Το σύστημα πρέπει να επιτρέπει την καταγραφή των δειγμάτων λαδιού που λαμβάνονται για αρχειοθέτηση ή για αποστολή σε πιθανούς πελάτες. Αυτό περιλαμβάνει την καταγραφή λεπτομερειών όπως η ημερομηνία δειγματοληψίας, η ποσότητα λαδιού που λήφθηκε και ο σκοπός της δειγματοληψίας.

#### 9. Αναφορές και Αναλύσεις:

- ο **Δημιουργία Αναφορών:** Το σύστημα πρέπει να παρέχει λειτουργία για τη δημιουργία διαφόρων αναφορών που σχετίζονται με τις λειτουργίες αποθήκευσης. Αυτό περιλαμβάνει αναφορές για παραλαβές λαδιού, φορτώσεις, μεταφορές, χημικές αναλύσεις και δείγματα.
- ο **Αναλύσεις Δεδομένων:** Το σύστημα πρέπει να περιλαμβάνει βασικές λειτουργίες ανάλυσης δεδομένων για την ανάλυση τάσεων και προτύπων στα δεδομένα λειτουργιών αποθήκευσης.

#### 10. Ασφάλεια και Ακεραιότητα Δεδομένων:



- **Κρυπτογράφηση Δεδομένων:** Το σύστημα πρέπει να εξασφαλίζει ότι όλα τα ευαίσθητα δεδομένα είναι κρυπτογραφημένα κατά την αποθήκευση και μετάδοση για να προστατεύονται από μη εξουσιοδοτημένη πρόσβαση.
- **Αρχεία Ελέγχου (Audit Logs):** Το σύστημα πρέπει να διατηρεί αρχεία ελέγχου όλων των δραστηριοτήτων των χρηστών για να εξασφαλίζεται η ανιχνευσιμότητα και η υπευθυνότητα.

Αυτές οι λειτουργικές απαιτήσεις αποτελούν τη βάση της εφαρμογής διαχείρισης αποθήκευσης, διασφαλίζοντας ότι ικανοποιεί τις επιχειρησιακές ανάγκες της εταιρείας εμπορίας ελαιολάδου, ενώ διατηρεί την ακεραιότητα και την ασφάλεια των δεδομένων.

## 2.1.2 Μη Λειτουργικές Απαιτήσεις

Οι μη λειτουργικές απαιτήσεις καθορίζουν τα κριτήρια που κρίνουν τη λειτουργία του συστήματος, όπως η απόδοση, η ασφάλεια και η χρηστικότητα. Αυτές οι απαιτήσεις είναι κρίσιμες για να διασφαλίσουν ότι το σύστημα δεν είναι μόνο λειτουργικά σωστό αλλά και αποτελεσματικό, ασφαλές και ευχάριστο στη χρήση.

### 1. Απόδοση (Performance):

- **Χρόνος Απόκρισης:** Το σύστημα πρέπει να εξασφαλίζει ταχεία απόκριση στις αιτήσεις των χρηστών, με τον μέσο χρόνο απόκρισης να είναι μικρότερος από 2 δευτερόλεπτα για τις περισσότερες λειτουργίες.
- **Κλιμακωσιμότητα (Scalability):** Το σύστημα πρέπει να είναι σε θέση να διαχειρίζεται αυξημένο φορτίο δεδομένων χωρίς σημαντική υποβάθμιση της απόδοσης.

### 2. Ασφάλεια (Security):

- **Κρυπτογράφηση Δεδομένων:** Το σύστημα πρέπει να εξασφαλίζει ότι όλα τα ευαίσθητα δεδομένα είναι κρυπτογραφημένα κατά την αποθήκευση και μετάδοση για να προστατεύονται από μη εξουσιοδοτημένη πρόσβαση.
- **Αρχεία Ελέγχου (Audit Logs):** Το σύστημα πρέπει να διατηρεί αρχεία ελέγχου όλων των δραστηριοτήτων των χρηστών για να εξασφαλίζεται η ανιχνευσιμότητα και η υπευθυνότητα.
- **Διαχείριση Πρόσβασης:** Το σύστημα πρέπει να υλοποιεί μηχανισμούς διαχείρισης πρόσβασης για να διασφαλίζεται ότι μόνο εξουσιοδοτημένοι χρήστες μπορούν να εκτελούν συγκεκριμένες λειτουργίες.

### 3. Χρηστικότητα (Usability):

- **Ευχρηστία Διεπαφής (API):** Το API πρέπει να είναι εύκολο στη χρήση και να παρέχει σαφή τεκμηρίωση για τους προγραμματιστές που θα αναπτύξουν το frontend.
- **Συμβατότητα:** Το σύστημα πρέπει να είναι συμβατό με διάφορα frontend frameworks και τεχνολογίες για να διευκολύνεται η ανάπτυξη των frontend εφαρμογών.

### 4. Αναφορές και Αναλύσεις (Reporting and Analytics):

- **Δημιουργία Αναφορών:** Το σύστημα πρέπει να παρέχει λειτουργία για τη δημιουργία διαφόρων αναφορών που σχετίζονται με τις λειτουργίες αποθήκευσης. Αυτό περιλαμβάνει αναφορές για παραλαβές λαδιού, φορτώσεις, μεταφορές, χημικές αναλύσεις και δείγματα.

- **Αναλύσεις Δεδομένων:** Το σύστημα πρέπει να περιλαμβάνει βασικές λειτουργίες ανάλυσης δεδομένων για την ανάλυση τάσεων και προτύπων στα δεδομένα λειτουργιών αποθήκευσης.

#### 5. Αξιοπιστία (Reliability):

- **Διαθεσιμότητα Συστήματος:** Το σύστημα πρέπει να έχει υψηλή διαθεσιμότητα, με στόχο την ελάχιστη διακοπή λειτουργίας και την ταχεία αποκατάσταση σε περίπτωση αποτυχίας.
- **Ανάκτηση από Σφάλματα:** Το σύστημα πρέπει να περιλαμβάνει μηχανισμούς ανάκτησης δεδομένων και επαναφοράς σε περίπτωση αποτυχίας ή απώλειας δεδομένων.

#### 6. Συντηρησιμότητα (Maintainability):

- **Ευκολία Συντήρησης:** Το σύστημα πρέπει να σχεδιαστεί με τέτοιο τρόπο ώστε να διευκολύνεται η συντήρηση και οι ενημερώσεις. Ο κώδικας πρέπει να είναι καλά τεκμηριωμένος και οργανωμένος.
- **Επεκτασιμότητα:** Το σύστημα πρέπει να είναι επεκτάσιμο για να επιτρέπει την προσθήκη νέων λειτουργιών και τη βελτίωση των υπαρχουσών χωρίς σημαντικές αλλαγές στην αρχιτεκτονική.

Αυτές οι μη λειτουργικές απαιτήσεις διασφαλίζουν ότι το σύστημα διαχείρισης αποθήκευσης είναι όχι μόνο λειτουργικό αλλά και αποδοτικό, ασφαλές και φιλικό προς τον χρήστη, ικανοποιώντας τις επιχειρησιακές ανάγκες της εταιρείας.

## 2.2 Αρχιτεκτονική Συστήματος

### 2.2.1 Επισκόπηση της Αρχιτεκτονικής του Συστήματος

Η αρχιτεκτονική της εφαρμογής διαχείρισης αποθήκευσης είναι δομημένη με βάση τη διαίρεση των λειτουργιών σε δύο ξεχωριστές υπηρεσίες: την **migration-service** και την **api-service**. Αυτή η προσέγγιση εξασφαλίζει διαχωρισμό των ευθυνών και διευκολύνει τη συντήρηση και την επεκτασιμότητα του συστήματος.

#### Migration-service

Η migration-service είναι υπεύθυνη για τον ορισμό και τη διαχείριση του σχήματος της βάσης δεδομένων PostgreSQL. Χρησιμοποιεί το Sequelize ORM για να εκτελεί migrations, τα οποία εφαρμόζουν αλλαγές στη δομή της βάσης δεδομένων. Η δομή της migration-service περιλαμβάνει τα εξής components:

- **config:** Περιέχει το αρχείο config.js που διαχειρίζεται τη σύνδεση με τη βάση δεδομένων.
- **migrations:** Περιέχει όλα τα migration scripts που εφαρμόζουν τις αλλαγές στη βάση δεδομένων, όπως η δημιουργία πινάκων και η τροποποίηση υπάρχοντων πινάκων.
- **models:** Περιέχει τα sequelize models που αντιστοιχούν στους πίνακες της βάσης δεδομένων.
- **seeders:** Περιέχει αρχεία seeder για την αρχικοποίηση της βάσης δεδομένων με δεδομένα επίδειξης.

#### API-service

Η api-service είναι η κύρια υπηρεσία του συστήματος που υποστηρίζει το API και περιλαμβάνει όλη τη λογική του backend. Η δομή της api-service περιλαμβάνει τα εξής components:

- **config:** Περιέχει τα αρχεία ρυθμίσεων DBConfig.mjs και ISMConfig.mjs που συνδέουν τον server με τη βάση δεδομένων και κανονίζουν τις ρυθμίσεις του συστήματος αντίστοιχα.
- **controllers:** Περιέχει τα modules που υλοποιούν τους ελεγκτές (controllers) για τις διάφορες λειτουργίες του API. Οι ελεγκτές χρησιμοποιούν τα modules των services για την επεξεργασία των αιτήσεων των χρηστών.
- **models:** Περιέχει τα Sequelize models που αντιστοιχούν στους πίνακες της βάσης δεδομένων, τα οποία χρησιμοποιούνται από την υπηρεσία.
- **services:** Περιέχει τα modules που υλοποιούν τις λειτουργίες CRUD χρησιμοποιώντας τα Sequelize models. Τα services παρέχουν τις λειτουργίες που χρειάζονται οι controllers για την αλληλεπίδραση με τη βάση δεδομένων.
- **routers:** Περιέχει τα API routes που κατευθύνουν τα requests στους κατάλληλους controllers. Τα routes χρησιμοποιούν τα modules των controllers ως middlewares.
- **jobs:** Περιέχει modules που εκτελούν προγραμματισμένες λειτουργίες του συστήματος, όπως η διαγραφή παλαιών tokens από τον πίνακα TokenBlacklist.
- **validators:** Περιέχει τα modules που ελέγχουν (εγκρίνουν / απορρίπτουν) και κάνουν sanitize τα δεδομένα που στέλνει ο χρήστης προς το server.

Η διασύνδεση μεταξύ των δύο υπηρεσιών γίνεται μέσω των Sequelize models, τα οποία δημιουργούνται από την migration-service και χρησιμοποιούνται από την api-service για την εκτέλεση των λειτουργιών CRUD. Η δομή αυτή εξασφαλίζει ότι οι αλλαγές στο σχήμα της βάσης δεδομένων είναι απομονωμένες και διαχειρίσιμες, ενώ η λογική του backend παραμένει καθαρή και ευέλικτη.

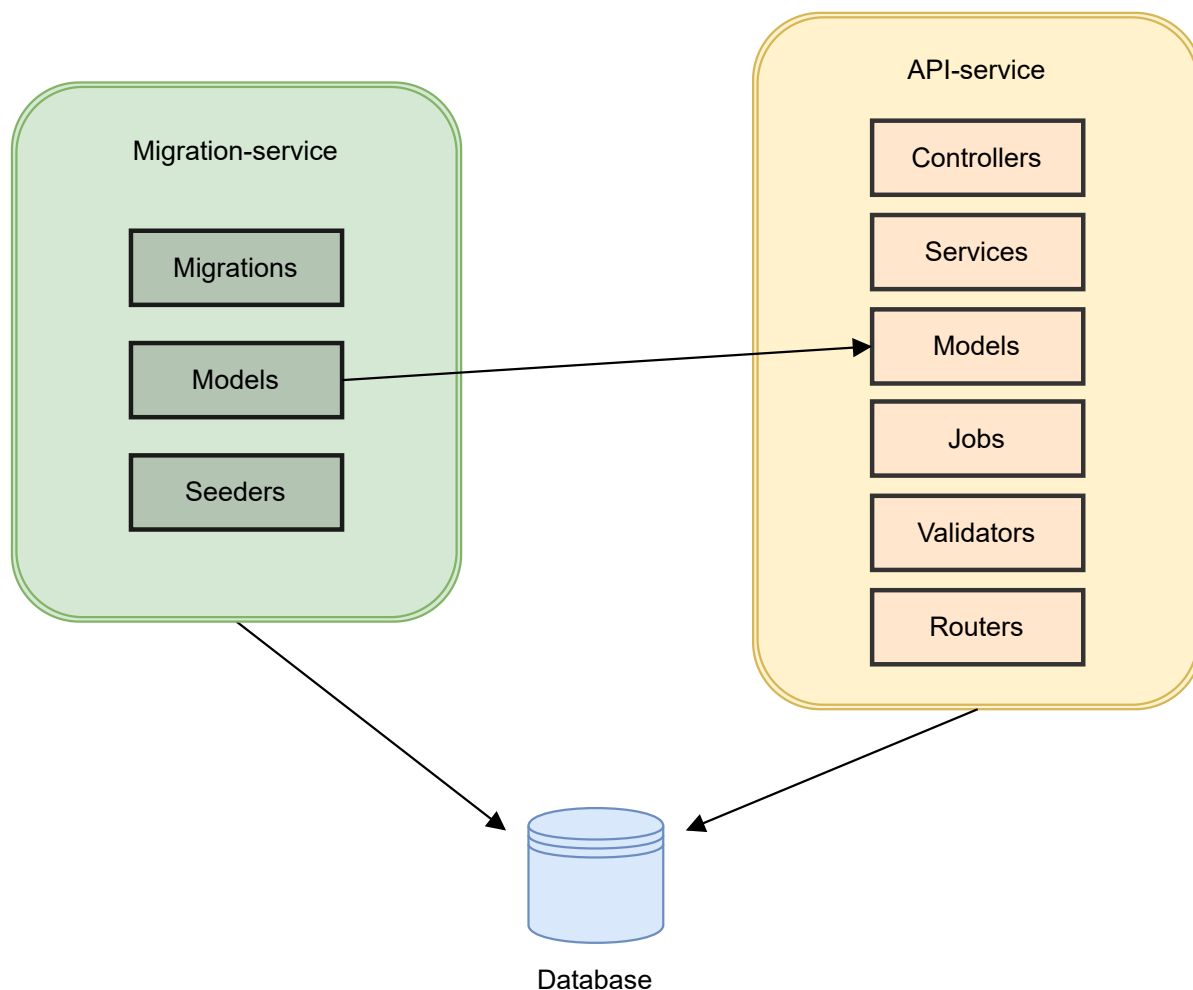


Figure 2.2.1: Η αρχιτεκτονική του συστήματος

Η αρχιτεκτονική αυτή διασφαλίζει ότι το σύστημα είναι διαχωρισμένο, επεκτάσιμο και εύκολο στη συντήρηση, ανταποκρινόμενο στις απαιτήσεις όπως αυτές τέθηκαν στην ενότητα [2.1](#).

### 2.2.2 Σχεδίαση του API:

Η σχεδίαση του API της εφαρμογής διαχείρισης αποθήκευσης επικεντρώνεται στην παροχή ευέλικτων και εύκολων στη χρήση endpoints για την αλληλεπίδραση με τη βάση δεδομένων. Το API παρέχει λειτουργίες CRUD (Create, Read, Update, Delete) για τους κύριους πόρους του συστήματος, όπως οι χρήστες, οι δεξαμενές, οι παραλαβές, οι φορτώσεις, οι μεταφορές, οι επιχειρηματικοί εταίροι, οι χημικές αναλύσεις και τα δείγματα.

Ο παρακάτω πίνακας παρουσιάζει τα κύρια routes του API και τις ενέργειες που υποστηρίζουν:

## API - Routes

Method	URL	Action
UserRoutes		
GET	/ism/users	<i>Shows all users</i>
GET	/ism/users/:id	<i>Show a specific user</i>
POST	/ism/users/new	<i>Creates a new user</i>
PUT	/ism/users/:id	<i>Updates an existing user</i>
DELETE	/ism/users/:id	<i>Removes an existing user</i>
POST	/ism/users/login	<i>Logs in an admin type user</i>
GET	/ism/users/logout	<i>Logs out an existing admin type user</i>
POST	/ism/users/sign	<i>Given the provided password returns the appropriate user data</i>
TankRoutes		
GET	/ism/tanks	<i>Shows all tanks</i>
GET	/ism/tanks/:id	<i>Show a specific tank</i>
POST	/ism/tanks/new	<i>Creates a new tank</i>
PUT	/ism/tanks/:id	<i>Updates an existing tank</i>
DELETE	/ism/tanks/:id	<i>Removes an existing tank</i>
PUT	/ism/tanks/:id/tag/analysis/:id	<i>Correlates an existing analysis to a specified tank</i>
GET	/ism/tanks/:id/get/fullness	<i>Calculates &amp; returns given fullness anew based on containing inflows</i>
GET	/ism/tanks/:id/get/acidity	<i>Calculates &amp; returns given acidity anew based on containing inflows</i>
PUT	/ism/tanks/:id/set/clean	<i>Resets tank contents to empty</i>

Method	URL	Action
TransferRoutes		
GET	/ism/transfers	<i>Shows all tranfers</i>
GET	/ism/transfers/:id	<i>Show a specific transfer</i>
POST	/ism/transfers/new	<i>Creates a new transfer</i>
PUT	/ism/transfers/:id	<i>Updates an existing transfer</i>
DELETE	/ism/transfers/:id	<i>Removes an existing transfer</i>
GET	/ism/transfers/latest/:no	<i>Shows n latest (date) transfers</i>
GET	/ism/transfers/show	<i>Shows all tranfers that satisfies the impending query</i>
InflowRoutes		
GET	/ism/inflows	<i>Shows all receptions</i>
GET	/ism/inflows/:id	<i>Show a specific reception</i>
POST	/ism/inflows/new	<i>Creates a new reception</i>
PUT	/ism/inflows/:id	<i>Updates an existing reception</i>
DELETE	/ism/inflows/:id	<i>Removes an existing reception</i>
PUT	/ism/inflows/:id/tag/analysis/:id	<i>Correlates an existing analysis to a specified reception</i>
PUT	/ism/inflows/:id/remove/analysis/:id	<i>Untags an existing analysis from a specified reception</i>
GET	/ism/inflows/latest/:no	<i>Shows n latest (date) reception</i>
GET	/ism/inflows/show	<i>Shows all receptions that satisfy the impending query</i>

Method	URL	Action
PUT	/ism/inflows/:id/validate	<i>Validates/Invalidates an existing reception, requires admin credentials</i>
OutflowRoutes		
GET	/ism/outflows	<i>Shows all outflows (Loadings &amp; Extractions)</i>
GET	/ism/outflows/:id	<i>Show a specific outflow (Loading or Extraction)</i>
POST	/ism/outflows/new	<i>Creates a new outflow (Loading or Extraction)</i>
PUT	/ism/outflows/:id	<i>Updates an existing outflow (Loading or Extraction)</i>
DELETE	/ism/outflows/:id	<i>Removes an existing outflow (Loading or Extraction)</i>
PUT	/ism/outflows/:id/tag/analysis/:id	<i>Correlates an existing analysis to a specified outflow (Loading or Extraction)</i>
PUT	/ism/outflows/:id/remove/analysis/:id	<i>Untags an existing analysis from a specified outflow (Loading or Extraction)</i>
GET	/ism/outflows/latest/:no	<i>Shows n latest (date) outflows (Loadings &amp; Extractions)</i>
GET	/ism/outflows/show	<i>Shows all outflows (Loadings &amp; Extractions) that satisfy the impending query</i>
PUT	/ism/outflows/:id/validate	<i>Validates/Invalidates an existing outflow, requires admin credentials</i>
BusinessPartnerRoutes		

Method	URL	Action
GET	/ism/partners	<i>Shows all Business partners</i>
GET	/ism/partners/:id	<i>Show a specific partner</i>
POST	/ism/partners/new	<i>Creates a new partner</i>
PUT	/ism/partners/:id	<i>Updates an existing partner</i>
DELETE	/ism/partners/:id	<i>Removes an existing partner</i>
GET	/ism/partners/show	<i>Shows all loadings that satisfy the impending query</i>
AnalysisRoutes		
GET	/ism/analysis	<i>Shows all analysis cards</i>
GET	/ism/analysis/:id	<i>Show a specific analysis card</i>
POST	/ism/analysis/new	<i>Creates a new analysis card</i>
PUT	/ism/analysis/:id	<i>Updates an existing analysis card</i>
DELETE	/ism/analysis/:id	<i>Removes an existing analysis card</i>
GET	/ism/analysis/latest/:no	<i>Shows n latest (date) analysis cards</i>
GET	/ism/analysis/show	<i>Shows all analysis cards that satisfy the impending query</i>
DELETE	/ism/analysis/unit/:ids	<i>Allows for bulk deletion! Removes analysis unit/s(e.g. 22,5,777) from an analysis card</i>
SampleRoutes		
GET	/ism/samples	<i>Shows all samples cards</i>
GET	/ism/samples/:id	<i>Show a specific samples card</i>



Method	URL	Action
POST	/ism/samples/new	<i>Creates a new samples card</i>
PUT	/ism/samples/:id	<i>Updates an existing samples card</i>
DELETE	/ism/samples/:id	<i>Removes an existing samples card</i>
GET	/ism/samples/latest/:no	<i>Shows n latest (date) samples cards</i>
PUT	/ism/samples/:id/tag/analysis/:id	<i>Correlates an existing analysis to a specified samples card</i>
GET	/ism/samples/show	<i>Shows all samples cards that satisfy the impending query</i>

## Σχεδιαστικές Αρχές

Το API σχεδιάστηκε με βάση τις εξής αρχές:

1. **RESTful Design:** Το API ακολουθεί τις αρχές του REST (Representational State Transfer) παρέχοντας ευανάγνωστες και προβλέψιμες διευθύνσεις URL για την αλληλεπίδραση με τους πόρους του συστήματος. Οι μέθοδοι HTTP (GET, POST, PUT, DELETE) χρησιμοποιούνται για να καθορίσουν την ενέργεια που θα εκτελεστεί σε κάθε πόρο.
2. **Σαφείς και Ευανάγνωστες URL:** Οι διευθύνσεις URL του API σχεδιάστηκαν να είναι σαφείς και ευανάγνωστες, διευκολύνοντας την κατανόηση και τη χρήση τους από τους προγραμματιστές. Κάθε πόρος έχει μια συγκεκριμένη διεύθυνση URL που αντικατοπτρίζει τη δομή του συστήματος.
3. **Ασφάλεια:** Το API περιλαμβάνει μηχανισμούς ασφαλείας, όπως η αυθεντικοποίηση χρηστών μέσω tokens και η χρήση HTTPS για την προστασία των δεδομένων κατά τη μεταφορά.
4. **Ανταπόκριση:** Το API παρέχει ταχείες αποκρίσεις στις αιτήσεις των χρηστών, διασφαλίζοντας την αποδοτική διαχείριση των λειτουργιών και των δεδομένων του συστήματος.
5. **Επεκτασιμότητα:** Το API σχεδιάστηκε έτσι ώστε να είναι επεκτάσιμο, επιτρέποντας την προσθήκη νέων λειτουργιών και τη βελτίωση των υπαρχουσών χωρίς σημαντικές αλλαγές στην αρχιτεκτονική.

## Περιγραφή των Κύριων Routes

- **UserRoutes:** Παρέχει λειτουργίες διαχείρισης χρηστών, όπως δημιουργία νέων χρηστών, ενημέρωση, διαγραφή και αυθεντικοποίηση.
- **TankRoutes:** Παρέχει λειτουργίες διαχείρισης δεξαμενών, όπως δημιουργία νέων δεξαμενών, ενημέρωση, διαγραφή, και ανάκτηση πληροφοριών σχετικά με την πληρότητα και την οξύτητα.
- **TransferRoutes:** Παρέχει λειτουργίες διαχείρισης μεταφορών λαδιού μεταξύ δεξαμενών.

- **InflowRoutes:** Παρέχει λειτουργίες διαχείρισης παραλαβών λαδιού, όπως δημιουργία νέων παραλαβών, ενημέρωση, διαγραφή και συσχέτιση με χημικές αναλύσεις.
- **OutflowRoutes:** Παρέχει λειτουργίες διαχείρισης φορτώσεων και εξαγωγών λαδιού, όπως δημιουργία νέων φορτώσεων, ενημέρωση, διαγραφή και συσχέτιση με χημικές αναλύσεις.
- **BusinessPartnerRoutes:** Παρέχει λειτουργίες διαχείρισης επιχειρηματικών εταίρων, όπως δημιουργία νέων εταίρων, ενημέρωση και διαγραφή.
- **AnalysisRoutes:** Παρέχει λειτουργίες διαχείρισης καρτών χημικών αναλύσεων, όπως δημιουργία νέων καρτών, ενημέρωση, διαγραφή και μαζική διαγραφή μονάδων αναλύσεων. <sup>3</sup>
- **SampleRoutes:** Παρέχει λειτουργίες διαχείρισης καρτών δειγμάτων, όπως δημιουργία νέων καρτών δειγμάτων, ενημέρωση και διαγραφή.

Η σχεδίαση του API εξασφαλίζει την εύκολη και αποδοτική αλληλεπίδραση των χρηστών με το σύστημα διαχείρισης αποθήκευσης, παρέχοντας όλες τις απαραίτητες λειτουργίες για τη διαχείριση των δεδομένων και των λειτουργιών της εταιρείας εμπορίας ελαιολάδου.

---

## 2.3 Σχεδίαση Βάσης Δεδομένων

### 2.3.1 Σχεδίαση Σχήματος

Η βάση δεδομένων της εφαρμογής διαχείρισης αποθήκευσης αποτελείται από διάφορους πίνακες που αντιπροσωπεύουν τους κύριους πόρους και τις λειτουργίες του συστήματος. Παρακάτω περιγράφονται οι κύριοι πίνακες της βάσης δεδομένων και οι δομές τους:

\*Η συντομογραφία **NN** αναφέρεται σε υποχρεωτικό πεδίο (Not Null)

#### 1. Table IOFlows

Ο πίνακας IOFlows περιγράφει τις εισροές (Παραλαβή) και τις εκροές (Φόρτωση, Εξαποθήκευση) του ελαιολάδου από και προς την αποθήκη. Ο λόγος για τον οποίο επιλέχθηκε ένας πίνακας να αναπαριστά όλες τις προαναφερθείσες λειτουργίες είναι ότι τα χαρακτηριστικά αυτών όπως και τα πεδία που τα περιγράφουν είναι τα ίδια. Η διαφοροποίηση γίνεται με το **enum** πεδίο **type**.

Αναλυτικά:

IOFlows			Σχόλια:
io_id		bigint	Αναγνωριστικό και πρωτεύον κλειδί. Αυξάνει αυτόματα.
type		enum	Το πεδίο περιγράφει τον τύπο εισροής / εκροής που έχει η εκάστοτε εγγραφή. Τύπος enum. Accepted values: [Reception, Loading, Extraction].
ioflow_date	timestamp	NN	Ημερομηνία καταγραφής. Αρχικοποιείται στην παρούσα.
business_partner_alias	varchar(255)	NN	Σε αυτό το πεδίο ο χρήστης εισάγει ένα alias για τον πελάτη/προμηθευτή. Αυτό θα χρησιμοποιηθεί απο τον admin στη διαδικασία της επικύρωσης, για την ταυτοποίηση του πελάτη/προμηθευτή.
quantity_gross	real	NN	Βάρος: μεικτό
quantity_tare	real	NN	Βάρος: απόβαρο
quantity_net	real	NN	Βάρος: καθαρό
quality_cat	varchar(100)	NN	Ποιοτική Κατηγορία στην οποία ανήκει η εισροή / εκροή. Μπορεί να πάρει τιμές: 'Extra Virgin', 'Virgin', 'Industrial', 'Dregs'(Mourga), 'BIO Extra', 'BIO Virgin'
comments		text	Πεδίο σχολίων
overseered_by	integer	NN	Το id του χρήστη που εκτελεί την παρούσα καταχώρηση.
business_partner_id	integer	NN	Το id του πελάτη/προμηθευτή τον οποίο αφορά η συγκεκριμένη καταχώρηση

validated	boolean	NN	Εκφράζει εάν η συγκεκριμένη εισροή/εκροή έχει επικυρωθεί από κάποιον <i>admin</i> . Αρχικοποιείται σε <b>false</b> .
username	varchar(100)	NN	Το <i>id</i> του <i>admin</i> που επικύρωσε την καταχώρηση.


#### Συσχετίσεις:

- overseered\_by > Users.u\_id (many-to-one)
- validated\_by > Users.u\_id (many-to-one)
- business\_partner\_id > BusinessPartners.bp\_id (many-to-one)

## 2. Table Shipments

Ο πίνακας Shipments περιέχει τις επιπλέον πληροφορίες που χρειάζονται για να καταγραφεί μία Φόρτωση στο σύστημα. Περιέχει κυρίως πληροφορίες που έχουν να κάνουν με το βυτιοφόρο όχημα που περαιώνει τη φόρτωση καθώς και πληροφορία που περιγράφει την κατάσταση των δεξαμενών κατά την στιγμή της φόρτωσης.

Αναλυτικά:

Shipments	Σχόλια:
io_id  bigint	Αναγνωριστικό και πρωτεύον κλειδί. Ξένο κλειδί που αναφέρεται στο IOFlows.io_id.
driver varchar(255) NN	Το όνομα του οδηγού του βυτιοφόρου οχήματος στο οποίο γίνεται η φόρτωση.
vehicle_plate varchar(10) NN	Η πινακίδα του φορτηγού οχήματος.
tanker_plate varchar(10)	Η πινακίδα του βυτίου.
load_seals varchar[]	Οι κωδικοί των σφραγίδων που χρησιμοποιήθηκαν κατά την επισφράγιση του φορτίου. Λίστα από strings
tank_snapshot json	Στο πεδίο αυτό αποθηκεύονται πληροφορίες που περιγράφουν την κατάσταση των εμπλεκόμενων δεξαμενών κατά την στιγμή της φόρτωσης. Τύπος δεδομένων json.


#### Συσχετίσεις:

- o\_id > IOFlows.io\_id (one-to-one)

## 3. Table Inflow\_Tank

Ο πίνακας Inflow\_Tank είναι πίνακας συσχέτισης των πινάκων IOFlows και Tanks. Αποτελεί έναν από τους πιο σημαντικούς πίνακες για το σύστημα καθώς περιέχει την πληροφορία του ποια εισροή βρίσκεται σε ποια δεξαμενή και σε ποια ποσότητα.

Αναλυτικά:

Inflow_Tank	Σχόλια:
i_id  bigint	Κομμάτι σύνθετου πρωτεύοντος κλειδιού. Ξένο κλειδί, αναφέρεται στο IOFlows.io_id.

<b>t_id</b> 🔑	<b>bigint</b>	Κομμάτι σύνθετου πρωτεύοντος κλειδιού. Ξένο κλειδί, αναφέρεται στο Tanks.t_id.
quantity	real <b>NN</b>	Η ποσότητα σε kg, της εισροής με i_id, στην δεξαμενή με t_id
received_at	timestamp <b>NN</b>	Η ημερομηνία της μετάγγισης στη δεξαμενή. Αρχικοποιείται στο χρόνο δημιουργίας της εγγραφής εάν δεν δοθεί ρητά.

#### Συσχετίσεις:

- i\_id > IOFlows.io\_id (many-to-one)
- t\_id > Tanks.t\_id (many-to-one)

#### 4. Table AnalysisUnit\_IOFlow

Ο πίνακας AnalysisUnit\_IOFlow είναι πίνακας συσχέτισης πινάκων IOFlows και AnalysisUnits. Δεν περιλαμβάνει περαιτέρω πληροφορία πέραν των ζευγών των ξένων κλειδιών.

Αναλυτικά:

AnalysisUnit_IOFlow		Σχόλια:
<b>io_id</b> 🔑	<b>bigint</b>	Κομμάτι σύνθετου πρωτεύοντος κλειδιού. Ξένο κλειδί, αναφέρεται στο IOFlows.io_id.
<b>au_id</b> 🔑	<b>bigint</b>	Κομμάτι σύνθετου πρωτεύοντος κλειδιού. Ξένο κλειδί, αναφέρεται στο AnalysisUnits.au_id.

#### Συσχετίσεις:

- io\_id > IOFlows.io\_id (many-to-one)
- au\_id > AnalysisUnits.au\_id (many-to-one)

#### 5. Table Tanks

Ο πίνακας Tanks αναφέρεται στα στιγμιότυπα των δεξαμενών που βρίσκονται στην αποθήκη. Οι πληροφορίες που περιέχει έχουν να κάνουν με τα κατασκευαστικά χαρακτηριστικά των δεξαμενών, τα ποιοτικά χαρακτηριστικά που τους προσδίδουν οι εισροές λαδιού που βρίσκονται μέσα, αλλά και την κατάσταση λειτουργικότητας στην οποία βρίσκονται τη δεδομένη στιγμή.

Αναλυτικά:

Tanks		Σχόλια:
<b>t_id</b> 🔑	<b>integer</b>	Αναγνωριστικό και πρωτεύον κλειδί. Αυξάνει αυτόματα.
name	varchar(100) <b>NN</b>	Το όνομα της δεξαμενής όπως αυτό δόθηκε από το χρήστη την στιγμή της δημιουργίας της εγγραφής. Έχει <b>μοναδική</b> τιμή για κάθε εγγραφή.
shape	enum <b>NN</b>	Το σχήμα της δεξαμενής. Τύπος enum. Accepted values: [CYLINDER, CUBE, RECT3D]
is_active	boolean <b>NN</b>	Δηλώνει εάν η δεξαμενή είναι ενεργή ή παροπλισμένη. Αρχικοποιείται σε <b>true</b> .
reception_tank	boolean <b>NN</b>	Εκφράζει εάν πρόκειται για προσωρινό ταμειυτήρα υποδοχής ή κυριώς δεξαμενή.

capacity	real	NN	Η μέγιστη ποσότητα λαδιού που μπορεί να φιλοξενήσει η δεξαμενή.
given_fullness	real	NN	Η τρέχουσα ποσότητα σε kg που βρίσκεται εντός της δεξαμενής. Αρχικοποιείται σε 0.
given_acidity	real		Η τρέχουσα οξύτητα της δεξαμενής.
given_quality_cat	varchar(100)		Η ποιοτική κατηγορία βάσει περιεχομένου στην οποία ανήκει η δεξαμενή. Μπορεί να πάρει τιμές: 'Extra Virgin', 'Virgin', 'Industrial', 'Dregs'(Mourga), 'BIO Extra', 'BIO Virgin'
clean	boolean		Δηλώνει εάν η δεξαμενή είναι καθαρή <b>και</b> άδεια. Αρχικοποιείται σε <b>true</b> .
comments	text		Πεδίο σχολίων

## 6. Table AnalysisUnit\_Tank

Ο πίνακας AnalysisUnit\_Tank είναι πίνακας συσχέτισης των πινάκων AnalysisUnits και Tanks. Εκτός των απαραίτητων ξένων κλειδιών περιλαμβάνει πληροφορία κατάστασης (ενεργή ή μη) για την συνδεδεμένη ανάλυση.

Αναλυτικά:

AnalysisUnit_Tank			Σχόλια:
t_id	integer		Κομμάτι σύνθετου πρωτεύοντος κλειδιού. Ξένο κλειδί, αναφέρεται στο Tanks.t_id.
au_id	bigint		Κομμάτι σύνθετου πρωτεύοντος κλειδιού. Ξένο κλειδί, αναφέρεται στο AnalysisUnits.au_id.
active	boolean		Αναφέρεται στην κατάσταση (επίκαιρη ή παρωχημένη) της συνδεδεμένης χημικής ανάλυσης. Αρχικοποιείται σε <b>true</b> .

Συσχετίσεις:

- t\_id > Tanks.t\_id (many-to-one)
- au\_id > AnalysisUnits.au\_id (many-to-one)

## 7. Table Transfers

Ο πίνακας Transfers καταγράφει τις μεταφορές (από δεξαμενή - προς δεξαμενή) που έλαβαν χώρα εντός της αποθήκης.

Αναλυτικά:

Transfers			Σχόλια:
tr_id	bigint		Αναγνωριστικό και πρωτεύον κλειδί. Αυξάνει αυτόματα.
transfer_date	timestamp	NN	Ημερομηνία τέλεσης της μεταφοράς. Αρχικοποιείται στην τρέχουσα εάν δεν δοθεί ρητά.
from_tank	integer	NN	Το id της δεξαμενής προέλευσης.
to_tank	integer	NN	Το id της δεξαμενής προορισμού.

quantity	real	NN	Η ποσότητα που μεταφέρθηκε.
overseered_by	integer	NN	Το id του χρήστη που κατέγραψε την μεταφορά.

#### Συσχετίσεις:

- overseered\_by > Users.u\_id (many-to-one)
- from\_tank > Tanks.t\_id (many-to-one)
- to\_tank > Tanks.t\_id (many-to-one)

### 8. Table AnalysisCards

Ο πίνακας AnalysisCards αναφέρεται στις κάρτες(συλλογές) που περιέχουν τις μονάδες χημικών αναλύσεων του συστήματος. Μία κάρτα δεν μπορεί να υπάρξει χωρίς εμπεριέχουσα μονάδα χημικής ανάλυσης.

Αναλυτικά:

AnalysisCards			Σχόλια:
a_id	bigint		Αναγνωριστικό και πρωτεύον κλειδί. Αυξάνει αυτόματα.
card_title	varchar		Τίτλος κάρτας
analysis_date	timestamp	NN	Ημερομηνία καταγραφής. Αρχικοποιείται στην παρούσα.
analysis_place	varchar(255)	NN	Το μέρος στο οποίο έλαβαν χώρα οι χημικές αναλύσεις που εμπεριέχονται στη κάρτα.
overseered_by	integer	NN	Το id του χρήστη που σύνταξε την κάρτα χημικών αναλύσεων.
comments	text		Πεδίο σχολίων.

#### Συσχετίσεις:

- overseered\_by > Users.u\_id (many-to-one)

### 9. Table AnalysisUnits

Ο πίνακας AnalysisUnits αναφέρεται στις χημικές αναλύσεις που εμπεριέχονται στις κάρτες χημικών αναλύσεων. Περιέχει πεδία που αναφέρονται στα πιο χρησιμοποιούμενα είδη αναλύσεων αλλά και πληροφορία για το δείγμα ελαιολάδου στο οποίο τελείται η χημική ανάλυση.

Αναλυτικά:

AnalysisUnits			Σχόλια:
au_id	bigint		Αναγνωριστικό και πρωτεύον κλειδί. Αυξάνει αυτόματα.
a_id	bigint	NN	Δηλώνει σε ποία κάρτα ανήκει η συγκεκριμένη χημική ανάλυση. Ξένο κλειδί. Αναφέρεται στο AnalysisCards.a_id.

sampling_date	timestamp	NN	Ημερομηνία που λήφθηκε το δείγμα προς χημική ανάλυση. Εάν δεν δηλωθεί ρητά, αρχικοποιείται στην παρούσα.
business_partner_alias	varchar(255)	NN	Προέλευση δείγματος. Προέλευση μπορεί να είναι η ίδια η εταιρεία.
sample_name	varchar(255)	NN	Ονομασία δείγματος χημικής ανάλυσης.
acidity	real		Τιμή ανάλυσης οξύτητας.
kappa_232	real		Τιμή ανάλυσης K232.
kappa_270	real		Τιμή ανάλυσης K270.
peroxide	real		Τιμή ανάλυσης υπεροξειδίων.
comments	text		Πεδίο σχολίων.

#### Συσχετίσεις:

- a\_id > AnalysisCards.a\_id (many-to-one)

### 10. Table ExtraAnalysis

Ο πίνακας ExtraAnalysis περιέχει πληροφορίες αναλύσεων που δεν ανήκουν στις τρεις πιο συνηθισμένες (οξύμετρηση, κάππα, υπεροξειδία) και ως εκ τούτου δεν έχουν δικά τους πεδία στον πίνακα AnalysisUnits. Συνδέεται με τον πίνακα AnalysisUnits

Αναλυτικά:

ExtraAnalysis			Σχόλια:
au_id	bigint		Κομμάτι σύνθετου πρωτεύοντος κλειδιού. Ξένο κλειδί, αναφέρεται στο IOFlows.io_id.
type	varchar(128)		Κομμάτι σύνθετου πρωτεύοντος κλειδιού. Αναφέρεται στο όνομα ή είδος της ανάλυσης.
value	varchar(100)	NN	Τιμή ανάλυσης για το συγκεκριμένο τύπο ανάλυσης.

#### Συσχετίσεις:

- au\_id > AnalysisUnits.au\_id (many-to-one)

### 11. Table SampleCards

Ο πίνακας SampleCards αναφέρεται στις κάρτες (συλλογές) που περιέχουν τους κωδικούς (μονάδες) δειγμάτων. Περιέχει πληροφορίες σχετικά με τον δειγματολήπτη, την ημερομηνία λήψης αλλά και τον παραλήπτη του δείγματος. Μία κάρτα δεν μπορεί να υπάρξει χωρίς εμπεριέχοντα κωδικό δείγματος.

Αναλυτικά:

SampleCards			Σχόλια:
s_id	bigint		Αναγνωριστικό και πρωτεύον κλειδί. Αυξάνει αυτόματα.



card_title	varchar	Τίτλος / περιγραφή κάρτας.
sampling_date	timestamp NN	Ημερομηνία λήψεως δείγματος. Αρχικοποιείται στην παρούσα.
sampler	integer NN	Το id του δειγματολήπτη / συντάκτη της κάρτας δειγμάτων.
recipient_alias	varchar(255) NN	Η επωνυμία του παραλήπτη των δειγμάτων.
comments	text	Πεδίο σχολίων.

#### Συσχετίσεις:

- sampler > Users.u\_id (many-to-one)

## 12. Table SampleCodes

Ο πίνακας SampleCodes αναφέρεται στα διάφορα δείγματα ελαιόλαδου που εξάχθηκαν και απαρτίζουν μία κάρτα δειγμάτων. Περιέχει πληροφορίες για την προέλευσή του δείγματος αλλά και τη σύνθεσή του εάν αυτό προήλθε από προσμίξεις διαφορετικών λαδιών.

Αναλυτικά:

SampleCodes		Σχόλια:
sc_id 🔗	bigint	Αναγνωριστικό και πρωτεύον κλειδί. Αυξάνει αυτόματα.
s_id	bigint NN	Ξένο κλειδί. Αναφέρεται στην κάρτα δειγμάτων που εμπεριέχει το παρόν κωδικό: SampleCard.s_id.
business_partner_alias	varchar(255)	Αναφέρεται στην προέλευση του δείγματος.
sample_name	varchar(10) NN	Κωδικός / όνομα δείγματος.
tank1_name	varchar(10) NN	Όνομα δεξαμενής προέλευσης δείγματος.
tank1_percentage	demical(5, 2) NN	Ποσοστό συμμετοχής της δεξαμενής.
tank2_name	varchar(10)	Όνομα δεξαμενής προέλευσης δείγματος.
tank2_percentage	demical(5, 2)	Ποσοστό συμμετοχής της δεξαμενής.
tank3_name	varchar(10)	Όνομα δεξαμενής προέλευσης δείγματος.
tank3_percentage	decimal(5, 2)	Ποσοστό συμμετοχής της δεξαμενής.
comments	text	Πεδίο σχολίων.

#### Συσχετίσεις:

- s\_id > SampleCards.s\_id (many-to-one)

### 13. Table AnalysisUnit\_SampleCode

Ο πίνακας AnalysisUnit\_SampleCode είναι πίνακας συσχέτισης πινάκων SampleCodes και AnalysisUnits. Δεν περιλαμβάνει περαιτέρω πληροφορία πέραν των ζευγών των ξένων κλειδιών.

Αναλυτικά:

AnalysisUnit_SampleCodes		Σχόλια:
sc_id 🔑	bigint	Κομμάτι σύνθετου πρωτεύοντος κλειδιού. Ξένο κλειδί, αναφέρεται στο SampleCodes.sc_id.
au_id 🔑	bigint	Κομμάτι σύνθετου πρωτεύοντος κλειδιού. Ξένο κλειδί, αναφέρεται στο AnalysisUnits.au_id.

Συσχετίσεις:

- sc\_id > SampleCodes.sc\_id (many-to-one)
- au\_id > AnalysisUnits.au\_id (many-to-one)

### 14. Table Users

Ο πίνακας Users κρατάει πληροφορία για του χρήστες του συστήματος. Περιέχει προσωπικές πληροφορίες του χρήστη, τα credentials του και πληροφορία για τον ρόλο του.

Αναλυτικά:

Users		Σχόλια:
u_id 🔑	integer	Αναγνωριστικό και πρωτεύον κλειδί. Αυξάνει αυτόματα.
fullname	varchar NN	Το ονοματεπώνυμο του χρήστη. Μοναδική τιμή για κάθε εγγραφή.
username	varchar(255) NN	Το username του χρήστη. Μοναδική τιμή για κάθε εγγραφή.
class	enum NN	Ο τύπος χρήστη. Πεδίο enum, μπορεί να πάει τιμές: [Admin, SimpleUser]
password	varchar NN	Ο κωδικός χρήστη. Μοναδική τιμή για κάθε εγγραφή.
is_active	boolean NN	Δηλώνει εάν πρόκειται για ενεργό ή μη-ενεργό χρήστη. Αρχικοποιείται σε <b>true</b> .

### 15. Table BusinessPartners

Ο πίνακας BusinessPartners περιέχει πληροφορίες για τους συνεργάτες(πελάτες / προμηθευτές) της επιχείρησης.

Αναλυτικά:

BusinessPartners		Σχόλια:
bp_id 🔑	integer	Αναγνωριστικό και πρωτεύον κλειδί. Αυξάνει αυτόματα.
type	enum NN	Το πεδίο περιγράφει τον τύπο συνεργάτη. Τύπος enum. Accepted values: [customer, supplier, customer-supplier].

name	varchar(255) NN	Η επωνυμία του πελάτη/προμηθευτή.
alias	varchar[]	Περιέχει όλα τα alias που είναι συνδεδεμένα με τον συγκεκριμένο πελάτη/προμηθευτή. Λίστα από strings.
address	varchar	Περιέχει τη διεύθυνση του πελάτη/προμηθευτή.
vat	varchar(32)	Περιέχει το ΑΦΜ του πελάτη/προμηθευτή.
phone	varchar[]	Περιέχει το/τα τηλέφωνο/-να του πελάτη/προμηθευτή. Λίστα από strings.
comments	text	Πεδίο σχολίων.

## 16. Table EditingLog

Ο πίνακας EditingLog υλοποιήθηκε με σκοπό να καταγράφει όλες τις αλλαγές(edits) που γίνονται από χρήστες πάνω στα records των άλλων πινάκων της βάσης, με σκοπό να υπάρχει ένα ιστορικό αλλαγών. Πρόκειται για πρώτη ιδέα του συγγραφέα κατά τη σχεδίαση του συστήματος και που μέχρι την στιγμή της συγγραφής του παρόντος report δεν έχει υλοποιηθεί.

Αναλυτικά:

EditingLog		Σχόλια:
e_id 🔗	bigint	Αναγνωριστικό και πρωτεύον κλειδί. Αυξάνει αυτόματα.
table_edited	enum NN	Δηλώνει το πίνακα στον οποίο έγινε το edit. Τύπος enum. Accepted values: Όλα τα ονόματα πινάκων του συστήματος.
key_1	bigint NN	Το πρωτεύον κλειδί της εγγραφής που έγινε edit.
key_2	bigint	Το πρωτεύον κλειδί της εγγραφής που έγινε edit εάν πρόκειται για εγγραφή με σύνθετο κλειδί.
attribute	varchar NN	Το όνομα του πεδίου που έγινε edit.
old_value	varchar	Η παλιά τιμή της εγγραφής στο πεδίο που έγινε edit.
edited_by	integer NN	Το id του χρήστη που έκανε το edit.

### Συσχετίσεις:

- edited\_by > Users.u\_id (many-to-one)

## 17. Table TokenBlacklist

Ο πίνακας TokenBlacklist περιέχει τα απενεργοποιημένα jsonwebtokens των admin-user. Η ύπαρξη του πίνακα έχει να κάνει με την ασφάλεια του συστήματος, ο λόγος είναι ένα παροπλισμένο token να μην μπορεί να ξαναχρησιμοποιηθεί πριν παρέλθει η ημερομηνία λήξης του.

Αναλυτικά:

TokenBlacklist		Σχόλια:
token 🔗	varchar	Πρωτεύον κλειδί. Συμβολοσειρά jsonwebtoken.
created_at	timestamp NN	Αρχικοποιείται στο παρόν.
updated_at	timestamp NN	Αρχικοποιείται στο παρόν.

## 18. Table EventLog

Στο πίνακα EventLog καταγράφονται οι ενέργειες που μεταβάλλουν το πίνακα Inflow\_Tank. Χρησιμοποιείται από τον μηχανισμό Event-handler <sup>4</sup> για να επαναφέρει τη βάση στη σωστή κατάσταση μετά από μεταβολή δεδομένων που αφορούν ενέργειες που τελέστηκαν στο παρελθόν.

Αναλυτικά:

EventLog		Σχόλια:
ev_id 🔗	bigint	Αναγνωριστικό και πρωτεύον κλειδί. Αυξάνει αυτόματα.
action_type	varchar(100) NN	Περιγράφει τον τύπο της ενέργειας που καταγράφεται.
action_id	bigint NN	Το id της ενέργειας που καταγράφεται.
action_date	timestamp NN	Η ημερομηνία που τελέστηκε η ενέργεια που καταγράφεται.
action_data	json	Όλα τα δεδομένα που χρειάζεται το σύστημα για να εκτελεσει την ενέργεια που καταγράφεται.
affected_inflows	json	Τα δεδομένα των εγγραφών στο πίνακα Inflow_Tank που επηρεάζονται από την ενέργεια που καταγράφεται.

## 19. Table InflowTankSnapshot

Ο πίνακας InflowTankSnapshot είναι ένα αρχείο στιγμιότυπων του πίνακα Inflow\_Tank όπως αυτά λαμβάνονται ανά ορισμένες χρονικές περιόδους. Χρησιμοποιείται από τον μηχανισμό Event-handler <sup>4</sup> για να επαναφέρει τη βάση στη σωστή κατάσταση μετά από μεταβολή δεδομένων που αφορούν ενέργειες που τελέστηκαν στο παρελθόν.

Αναλυτικά:

InflowTankSnapshots		Σχόλια:
snapshot_id 🔗	bigint	Αναγνωριστικό και πρωτεύον κλειδί. Αυξάνει αυτόματα.
snapshot	json NN	Στιγμιότυπο του πίνακα Inflow_Tank σε μορφή json.

### Γενικά:

- Σε όλους τους πίνακες της βάσης πέραν των πεδίων όπως αυτά παρουσιάστηκαν στην παρούσα ενότητα, υπάρχουν ακόμη δύο βοηθητικά πεδία. Αυτά είναι:

*			Σχόλια:
created_at	timestamp	NN	Αρχικοποιείται στο παρόν.
updated_at	timestamp	NN	Αρχικοποιείται στο παρόν.

- Η σχεδίαση του σχήματος της βάσης δεδομένων είναι δομημένη ώστε να υποστηρίζει όλες τις βασικές λειτουργίες της εφαρμογής διαχείρισης αποθήκευσης. Οι πίνακες και οι σχέσεις μεταξύ τους εξασφαλίζουν την αποθήκευση και διαχείριση των δεδομένων με αποδοτικό και ασφαλή τρόπο.
-

## 2.3.2 Διάγραμμα Οντοτήτων-Συσχετίσεων

Το διάγραμμα οντοτήτων-συσχετίσεων (ER διάγραμμα) απεικονίζει τη δομή της βάσης δεδομένων και τις σχέσεις μεταξύ των πινάκων. Ακολουθεί το ER διάγραμμα που δείχνει την συνολική δομή της βάσης δεδομένων του συστήματος διαχείρισης αποθήκευσης.



## 2.4 Κατηγορίες Χρηστών του Συστήματος

### 2.4.1 Τύποι Χρηστών

Το σύστημα υποστηρίζει δύο τύπους χρηστών. Τους διαχειριστές [ **Admins** ] και τους απλούς χρήστες [ **Users** ] και προσφέρει σε αυτούς έξι βασικές λειτουργίες και δύο βοηθητικές, οι οποίες εκτελούν ενέργειες σε αντίστοιχους πίνακες στη βάση δεδομένων.

#### Βασικές Λειτουργίες:

1. Παραλαβή
2. Φόρτωση
3. Μεταφορά
4. Χημική Ανάλυση
5. Δείγματα
6. Διαχείριση Δεξαμενών

#### Δευτερεύουσες Λειτουργίες:

7. Αρχείο Χρηστών
8. Αρχείο Πελατών / Προμηθευτών

[ **Users** ] Οι χρήστες (ή απλοί χρήστες) μπορούν να επιτελέσουν όλες τις βασικές λειτουργίες του συστήματος και τις σχετικές ενέργειες CRUD στους αντίστοιχους πίνακες της βάσης του συστήματος, μέσω του UI / API. Οι χρήστες είναι αυτοί που ευθύνονται για τη δημιουργία και τροποποίηση του κύριου όγκου δεδομένων στο σύστημα. Οι χρήστες δε χρειάζεται να εισάγουν κάποια credentials για την είσοδό τους στο σύστημα, είναι απαραίτητο όμως να εισάγουν τον κωδικό τους κάθε φορά που διατελούν ενέργεια η οποία μεταβάλλει την κατάσταση της βάσης δεδομένων του συστήματος. Σε αυτό το σενάριο χρησιμοποιούν τον *μηχανισμό υπογραφής των χρηστών* όπως αυτός παρουσιάζεται στην ενότητα [3.4.2](#).

[ **Admins** ] Οι Διαχειριστές μπορούν να επιτελέσουν όλες τις βασικές λειτουργίες, όπως οι απλοί χρήστες καθώς και τις δευτερεύουσες λειτουργίες, οι οποίες είναι διαθέσιμες μόνο σε αυτούς. Επιπλέον, είναι αρμόδιοι να επικυρώνουν ενέργειες των χρηστών <sup>5</sup> προσθέτοντας έτσι ένα ακόμα στάδιο ελέγχου στη διαδικασία της καταγραφής δεδομένων. Οι διαχειριστές για να αναγνωριστούν από το σύστημα ως τέτοιοι, θα πρέπει να κάνουν login στέλνοντας τα credentials τους στο κατάλληλο endpoint του API. Η διαδικασία της αυθεντικοποίησης των admin χρηστών περιγράφεται στην επόμενη ενότητα.

---

## 2.4.2 Αυθεντικοποίηση Διαχειριστών

Για την αυθεντικοποίηση ενός διαχειριστή (admin user), το σύστημα χρησιμοποιεί το πακέτο [jsonwebtoken](#). Ο διαχειριστής στέλνει ένα αίτημα login στο κατάλληλο endpoint του API <sup>6</sup> με τα διαπιστευτήριά του (username και password). Εάν η απάντηση είναι επιτυχής, ο διαχειριστής λαμβάνει ένα token με προκαθορισμένη διάρκεια ζωής.

### Μηχανισμός Εργασίας του Jsonwebtoken

Το **jsonwebtoken** επιτρέπει τη δημιουργία, την επαλήθευση και την αποκρυπτογράφηση των JSON Web Tokens (JWT). Ένα JWT είναι ένα ασφαλές και συμπαγές μέσο για τη μεταφορά πληροφοριών μεταξύ δύο μερών. Η διαδικασία λειτουργεί ως εξής:

1. **Δημιουργία Token:** Μετά την επιτυχή επαλήθευση των διαπιστευτηρίων του διαχειριστή, το σύστημα δημιουργεί ένα JWT χρησιμοποιώντας ένα μυστικό κλειδί. Το token περιλαμβάνει τις πληροφορίες του διαχειριστή (π.χ., user ID) και την ημερομηνία λήξης του.
2. **Αποστολή Token:** Το token αποστέλλεται στον διαχειριστή ως απάντηση στο αίτημα login.
3. **Χρήση Token:** Ο διαχειριστής χρησιμοποιεί το token για να πραγματοποιήσει αιτήματα στο σύστημα που απαιτούν αυθεντικοποίηση. Το token αποστέλλεται ως μέρος των headers των αιτημάτων.
4. **Επαλήθευση Token:** Κάθε φορά που το σύστημα λαμβάνει ένα αίτημα με το token, επαληθεύει την εγκυρότητά του χρησιμοποιώντας το μυστικό κλειδί. Εάν το token είναι έγκυρο και δεν έχει λήξει, το αίτημα εγκρίνεται.

### Λειτουργία Logout

Ο διαχειριστής μπορεί να αποσυνδεθεί ανά πάσα στιγμή. Κατά την αποσύνδεση, το token εισάγεται ως εγγραφή στον πίνακα **TokenBlacklist**, αποτρέποντας τη χρήση του από κακόβουλα τρίτα μέρη όσο η ημερομηνία λήξης του token παραμένει ενεργή. Η εισαγωγή αυτή διασφαλίζει ότι το token δεν μπορεί να χρησιμοποιηθεί για κακόβουλες ενέργειες ακόμα και πριν από την εκπνοή του.

---



# Κεφάλαιο 3: Υλοποίηση

## 3.1 Περιβάλλον Ανάπτυξης

Η ανάπτυξη της εφαρμογής διαχείρισης αποθήκευσης πραγματοποιήθηκε εξ ολοκλήρου στο Visual Studio Code (VSCode), κάνοντας χρήση της πληθώρας εργαλείων και τεχνολογιών που αυτό υποστηρίζει, συντελώντας έτσι στη βελτίωση της αποτελεσματικότητας και της παραγωγικότητας της διαδικασίας ανάπτυξης.

### Visual Studio Code (VSCode)

Το Visual Studio Code (VSCode)<sup>7</sup> είναι ένας ελαφρύς αλλά ισχυρός επεξεργαστής κώδικα που αναπτύχθηκε από τη Microsoft. Είναι ιδιαίτερα δημοφιλής στους προγραμματιστές λόγω των πλούσιων δυνατοτήτων του και της ευκολίας χρήσης του. Το VSCode υποστηρίζει μια μεγάλη ποικιλία γλωσσών προγραμματισμού και εργαλείων ανάπτυξης, καθιστώντας το ιδανικό για ανάπτυξη εφαρμογών με Node.js.

#### Πλεονεκτήματα του VSCode για την Ανάπτυξη με Node.js

- **Ενσωμάτωση με Node.js και JavaScript:** Το VSCode παρέχει εξαιρετική υποστήριξη για Node.js και JavaScript, με δυνατότητες όπως αυτόματη συμπλήρωση κώδικα, debugging, και διαχείριση πακέτων. Η ενσωματωμένη υποστήριξη για τη Node.js διευκολύνει την ανάπτυξη και την αποσφαλμάτωση εφαρμογών που βασίζονται στη Node.js.
- **Επεκτάσεις (Extensions):** Το VSCode διαθέτει μια μεγάλη συλλογή από επεκτάσεις που μπορούν να βελτιώσουν τη διαδικασία ανάπτυξης. Για παράδειγμα, επεκτάσεις όπως το ESLint για τον έλεγχο του κώδικα JavaScript, και το Prettier για τη μορφοποίηση του κώδικα, είναι απαραίτητες για τη διασφάλιση της ποιότητας του κώδικα.
- **Ενσωματωμένο Τερματικό (Integrated Terminal):** Ο ενσωματωμένος τερματικός του VSCode επιτρέπει στους προγραμματιστές να εκτελούν εντολές και scripts χωρίς να χρειάζεται να αλλάζουν παράθυρα, βελτιώνοντας την παραγωγικότητα.
- **Git Integration:** Το VSCode προσφέρει ενσωματωμένη υποστήριξη για Git, επιτρέποντας στους προγραμματιστές να διαχειρίζονται την έκδοση του κώδικα τους απευθείας μέσα από τον επεξεργαστή κώδικα.
- **Διαθέσιμο σε Πολλαπλές Πλατφόρμες:** Το VSCode είναι διαθέσιμο για Windows, macOS, και Linux, καθιστώντας το προσιτό για προγραμματιστές ανεξάρτητα από την πλατφόρμα που χρησιμοποιούν.

### Yarn

Το Yarn<sup>8</sup> είναι ένας διαχειριστής πακέτων που χρησιμοποιείται για τη διαχείριση των εξαρτήσεων του έργου. Παρέχει μια γρήγορη, αξιόπιστη και ασφαλή διαχείριση των πακέτων, κάτι που το καθιστά ιδανικό για την ανάπτυξη έργων σε Node.js. Ορισμένα από τα κύρια χαρακτηριστικά του Yarn περιλαμβάνουν:

- **Ταχύτητα:** Το Yarn βελτιστοποιεί τη διαδικασία εγκατάστασης των πακέτων, αποθηκεύοντας τα πακέτα τοπικά και αποφεύγοντας περιττές λήψεις, κάτι που μειώνει τον χρόνο εγκατάστασης.
- **Ασφάλεια:** Το Yarn ελέγχει την ακεραιότητα των πακέτων μέσω checksums, διασφαλίζοντας ότι τα εγκατεστημένα πακέτα δεν έχουν τροποποιηθεί.

- **Διαχείριση Εξαρτήσεων:** Το Yarn παρέχει εργαλεία για την εύκολη διαχείριση των εξαρτήσεων, συμπεριλαμβανομένων των συγκρούσεων έκδοσης και των υπό-εξαρτήσεων.

## pgAdmin 4

Το pgAdmin 4 <sup>9</sup> είναι ένα ισχυρό εργαλείο διαχείρισης και ανάπτυξης για τη βάση δεδομένων PostgreSQL. Χρησιμοποιήθηκε για την παρακολούθηση και τη διαχείριση της βάσης δεδομένων κατά τη διάρκεια της ανάπτυξης της εφαρμογής. Ορισμένα από τα κύρια χαρακτηριστικά του pgAdmin 4 περιλαμβάνουν:

- **Διαχείριση Βάσης Δεδομένων:** Παρέχει ένα γραφικό περιβάλλον για την εύκολη διαχείριση των βάσεων δεδομένων, επιτρέποντας στους χρήστες να δημιουργούν, να τροποποιούν και να διαγράφουν πίνακες, σχήματα και άλλες οντότητες της βάσης δεδομένων.
- **Εκτέλεση Ερωτημάτων SQL:** Προσφέρει έναν ισχυρό επεξεργαστή SQL που επιτρέπει την εκτέλεση ερωτημάτων και την ανάλυση των αποτελεσμάτων.
- **Παρακολούθηση Απόδοσης:** Παρέχει εργαλεία για την παρακολούθηση της απόδοσης της βάσης δεδομένων και την ανάλυση της χρήσης των πόρων.

Η χρήση του Visual Studio Code σε συνδυασμό με το Yarn και το pgAdmin 4 παρείχε ένα αποδοτικό και ευέλικτο περιβάλλον ανάπτυξης, επιτρέποντας την εύκολη διαχείριση των πακέτων και την αποτελεσματική παρακολούθηση της βάσης δεδομένων PostgreSQL.

---

## 3.2 Τεχνολογίες και Εργαλεία που Χρησιμοποιήθηκαν:

### 3.2.1 Node.js

Η Node.js <sup>10</sup> είναι μια πλατφόρμα JavaScript που βασίζεται στη μηχανή V8 του Google Chrome και χρησιμοποιείται για την ανάπτυξη δικτυακών εφαρμογών. Στο συγκεκριμένο έργο, η Node.js χρησιμοποιήθηκε ως το βασικό runtime περιβάλλον για την ανάπτυξη του backend της εφαρμογής διαχείρισης αποθήκευσης. Οι λόγοι για την επιλογή του Node.js και τα κύρια χαρακτηριστικά του περιλαμβάνουν:

#### Χαρακτηριστικά της Node.js

- **Ενιαίος Χρόνος Εκτέλεσης (Single-Threaded Event Loop):** Η Node.js χρησιμοποιεί ένα ενιαίο νήμα εκτέλεσης με μοντέλο event-driven, επιτρέποντας την ταυτόχρονη διαχείριση πολλαπλών αιτημάτων χωρίς την ανάγκη δημιουργίας πολλαπλών νημάτων. Αυτό βελτιώνει την απόδοση και την αποδοτικότητα της εφαρμογής.
- **Non-Blocking I/O:** Η non-blocking φύση του I/O στη Node.js επιτρέπει στις λειτουργίες εισόδου/εξόδου να εκτελούνται ασύγχρονα, βελτιώνοντας την απόδοση της εφαρμογής και επιτρέποντας την ταυτόχρονη διαχείριση πολλαπλών αιτημάτων.
- **Εκτενής Βιβλιοθήκη Πακέτων (NPM):** Η Node.js συνοδεύεται από το Node Package Manager (NPM), το οποίο παρέχει πρόσβαση σε χιλιάδες πακέτα και βιβλιοθήκες που μπορούν να χρησιμοποιηθούν για την ανάπτυξη εφαρμογών. Αυτή η εκτενής βιβλιοθήκη πακέτων διευκολύνει την προσθήκη νέων λειτουργιών και την επίλυση προβλημάτων.
- **Επέκταση με JavaScript:** Δεδομένου ότι η Node.js χρησιμοποιεί JavaScript, επιτρέπει στους προγραμματιστές να χρησιμοποιούν την ίδια γλώσσα για την ανάπτυξη τόσο του frontend όσο και του backend της εφαρμογής, καθιστώντας την ανάπτυξη πιο συνεπή και αποτελεσματική.
- **Κοινότητα και Υποστήριξη:** Η Node.js έχει μια μεγάλη και ενεργή κοινότητα προγραμματιστών που συνεισφέρουν σε πακέτα, εργαλεία και οδηγούς, καθιστώντας εύκολη την εύρεση λύσεων και την λήψη βοήθειας.

#### Χρήση της Node.js στο Έργο

Στο παρόν έργο, η Node.js χρησιμοποιήθηκε για την ανάπτυξη του backend λογισμικού, υποστηρίζοντας όλες τις βασικές λειτουργίες της εφαρμογής διαχείρισης αποθήκευσης. Η Node.js ήταν υπεύθυνη για την εκτέλεση του διακομιστή, την επεξεργασία των αιτημάτων των χρηστών, την επικοινωνία με τη βάση δεδομένων PostgreSQL και με την προσθήκη του Sequelize την εκτέλεση των λειτουργιών CRUD (Create, Read, Update, Delete).

Η επιλογή του Node.js βασίστηκε στα παραπάνω πλεονεκτήματα, τα οποία εξασφάλισαν την ανάπτυξη μιας αποδοτικής, επεκτάσιμης και ασφαλούς εφαρμογής.

---

## 3.2.2 Express.js

Το Express.js<sup>11</sup> είναι ένα ελαφρύ και ευέλικτο framework για τη Node.js που χρησιμοποιείται για την ανάπτυξη web εφαρμογών και APIs. Στο συγκεκριμένο έργο, το Express.js χρησιμοποιήθηκε για τη δημιουργία του API και τη διαχείριση των αιτημάτων των χρηστών. Τα κύρια χαρακτηριστικά και τα πλεονεκτήματα του Express.js περιλαμβάνουν:

### Χαρακτηριστικά του Express.js

- **Απλότητα και Ευελιξία:** Το Express.js παρέχει μια απλή και ευέλικτη διεπαφή για την ανάπτυξη web εφαρμογών, επιτρέποντας στους προγραμματιστές να δημιουργούν εύκολα routes και να διαχειρίζονται αιτήματα και απαντήσεις.
- **Middleware:** Το Express.js υποστηρίζει τη χρήση middleware, το οποίο επιτρέπει την εκτέλεση κώδικα σε διάφορα στάδια της επεξεργασίας των αιτημάτων. Αυτό διευκολύνει την προσθήκη λειτουργιών όπως ο έλεγχος ταυτότητας, η καταγραφή και η διαχείριση σφαλμάτων.
- **Διαχείριση Routes:** Το Express.js παρέχει ενσωματωμένα εργαλεία για τη διαχείριση των routes, επιτρέποντας την εύκολη διαμόρφωση των διευθύνσεων URL και την αντιστοίχιση τους σε συγκεκριμένες λειτουργίες.
- **Υποστήριξη για View Engines:** Το Express.js υποστηρίζει τη χρήση view engines όπως το Pug και το EJS για τη δημιουργία δυναμικών σελίδων HTML.
- **Συμβατότητα με άλλες Βιβλιοθήκες:** Το Express.js είναι συμβατό με πολλές άλλες βιβλιοθήκες και εργαλεία της Node.js, επιτρέποντας την εύκολη ενσωμάτωση πρόσθετων λειτουργιών και τη βελτίωση της ανάπτυξης.

### Ρόλος του Express.js στο Έργο

Στο παρόν έργο, το Express.js χρησιμοποιήθηκε για την ανάπτυξη του API που επιτρέπει την αλληλεπίδραση των χρηστών με τη βάση δεδομένων. Οι κύριες λειτουργίες που υλοποιήθηκαν με τη χρήση του Express.js περιλαμβάνουν:

- **Δημιουργία και Διαχείριση Routes:** Το Express.js χρησιμοποιήθηκε για τη δημιουργία των routes του API, επιτρέποντας την πρόσβαση στις λειτουργίες CRUD για τους διάφορους πόρους του συστήματος (όπως χρήστες, δεξαμενές, παραλαβές, φορτώσεις, μεταφορές, αναλύσεις και δείγματα).
- **Επεξεργασία Αιτημάτων και Απαντήσεων:** Το Express.js διαχειρίζεται την επεξεργασία των αιτημάτων των χρηστών και την αποστολή των κατάλληλων απαντήσεων, εξασφαλίζοντας την ομαλή και αποδοτική λειτουργία του API.
- **Χρήση Middleware:** Διάφορα middleware χρησιμοποιήθηκαν για τον έλεγχο ταυτότητας των χρηστών, την καταγραφή των αιτημάτων και την διαχείριση σφαλμάτων, βελτιώνοντας την ασφάλεια και την αξιοπιστία της εφαρμογής.

Η χρήση του Express.js στο έργο εξασφάλισε την ανάπτυξη ενός αποδοτικού και ευέλικτου API, επιτρέποντας την εύκολη διαχείριση των δεδομένων και την αλληλεπίδραση των χρηστών με το σύστημα.

---

### 3.2.3 Sequelize ORM

Το Sequelize <sup>12</sup> είναι μία Object-Relational Mapping (ORM) βιβλιοθήκη για Node.js που παρέχει ένα εύκολο στη χρήση API για την αλληλεπίδραση με σχεσιακές βάσεις δεδομένων όπως PostgreSQL, MySQL, SQLite, και MSSQL. Στο συγκεκριμένο έργο, το Sequelize χρησιμοποιήθηκε για τη διαχείριση της βάσης δεδομένων PostgreSQL, επιτρέποντας την απλοποίηση των λειτουργιών CRUD (Create, Read, Update, Delete) και την υλοποίηση του σχήματος της βάσης δεδομένων.

#### Χαρακτηριστικά του Sequelize

- **Εύκολη Χρήση και Καθαρός Κώδικας:** Το Sequelize παρέχει ένα κατανοητό API που διευκολύνει την αλληλεπίδραση με τη βάση δεδομένων μέσω JavaScript. Αυτό επιτρέπει στους προγραμματιστές να γράφουν καθαρό και κατανοητό κώδικα χωρίς να χρειάζεται να γράφουν απευθείας SQL εντολές.
- **Υποστήριξη Πολλαπλών Βάσεων Δεδομένων:** Το Sequelize υποστηρίζει διάφορους τύπους σχεσιακών βάσεων δεδομένων, όπως PostgreSQL, MySQL, SQLite, και MSSQL, επιτρέποντας τη χρήση του σε ένα ευρύ φάσμα έργων.
- **Μοντέλα και Συσχετίσεις:** Το Sequelize χρησιμοποιεί μοντέλα για να αναπαριστά τους πίνακες της βάσης δεδομένων και υποστηρίζει διάφορους τύπους συσχετίσεων μεταξύ των μοντέλων, όπως one-to-one, one-to-many, και many-to-many. Αυτό διευκολύνει τη δημιουργία και διαχείριση σύνθετων δομών δεδομένων.
- **Επαληθεύσεις και Περιορισμοί:** Το Sequelize επιτρέπει την προσθήκη επαληθεύσεων και περιορισμών στα μοντέλα, διασφαλίζοντας την ακεραιότητα των δεδομένων και την εφαρμογή επιχειρηματικών κανόνων.
- **Υποστήριξη για Migrations:** Το Sequelize παρέχει ενσωματωμένη υποστήριξη για migrations, επιτρέποντας την ευέλικτη διαχείριση των αλλαγών στο σχήμα της βάσης δεδομένων.

#### Χρήση του Sequelize στο Έργο

Στο παρόν έργο, το Sequelize χρησιμοποιήθηκε για τη δημιουργία και διαχείριση των μοντέλων που αντιστοιχούν στους πίνακες της βάσης δεδομένων PostgreSQL. Οι κύριες λειτουργίες που υλοποιήθηκαν με τη χρήση του Sequelize περιλαμβάνουν:

- **Δημιουργία Μοντέλων:** Το Sequelize χρησιμοποιήθηκε για τη δημιουργία μοντέλων που αναπαριστούν τους πίνακες της βάσης δεδομένων. Κάθε μοντέλο περιλαμβάνει τις στήλες του πίνακα και τους τύπους δεδομένων τους, καθώς και τους περιορισμούς και τις επαληθεύσεις.
- **CRUD Λειτουργίες:** Το Sequelize διευκόλυνε την υλοποίηση των λειτουργιών CRUD για τους διάφορους πόρους του συστήματος, επιτρέποντας την εύκολη δημιουργία, ανάγνωση, ενημέρωση και διαγραφή εγγραφών στη βάση δεδομένων μέσω του API.
- **Συσχετίσεις Μοντέλων:** Το Sequelize επιτρέπει τη δημιουργία συσχετίσεων μεταξύ των μοντέλων, διευκολύνοντας την αναπαράσταση σύνθετων δομών δεδομένων και τη διαχείριση των σχέσεων μεταξύ των πινάκων.
- **Sequelize Transactions:** Ο μηχανισμός transaction που παρέχει το Sequelize χρησιμοποιήθηκε εκτενώς στην ανάπτυξη του παρόντος λογισμικού, διασφαλίζοντας την ατομικότητα των ενεργειών των χρηστών, που τροποποιούσαν δεδομένα σε παραπάνω από ένα πίνακα, και επιτρέποντας στη βάση να διατηρεί την συνέπεια στα δεδομένα της ακόμα και σε περιπτώσεις σφάλματος.

# Sequelize Migrations

Τα migrations είναι ένας μηχανισμός του Sequelize που επιτρέπει την ευέλικτη διαχείριση των αλλαγών στο σχήμα της βάσης δεδομένων. Με τον μηχανισμό των migrations, οι προγραμματιστές μπορούν να δημιουργούν, να τροποποιούν και να διαγράφουν πίνακες και στήλες, καθώς και να διαχειρίζονται τις συσχετίσεις μεταξύ των πινάκων.

## Χαρακτηριστικά των Migrations

- **Ιστορικό Αλλαγών:** Τα migrations επιτρέπουν την καταγραφή του ιστορικού των αλλαγών στο σχήμα της βάσης δεδομένων, επιτρέποντας την ανίχνευση των αλλαγών και την επαναφορά σε προηγούμενες καταστάσεις εάν χρειαστεί.
- **Αυτοματοποιημένη Διαχείριση:** Τα migrations αυτοματοποιούν τη διαδικασία εφαρμογής των αλλαγών στη βάση δεδομένων, εξασφαλίζοντας ότι όλες οι αλλαγές εφαρμόζονται με συνέπεια και ακρίβεια.
- **Διαχειρίσιμες Αναβαθμίσεις:** Τα migrations επιτρέπουν την εφαρμογή αναβαθμίσεων στη βάση δεδομένων με ελεγχόμενο και διαχειρίσιμο τρόπο, μειώνοντας τον κίνδυνο σφαλμάτων και διακοπών λειτουργίας.

## Χρήση των Migrations στο Έργο

Στο παρόν έργο, ο μηχανισμός των migrations χρησιμοποιήθηκε για τη διαχείριση των αλλαγών στο σχήμα της βάσης δεδομένων PostgreSQL. Κάθε αλλαγή στο σχήμα της βάσης δεδομένων καταγράφηκε σε ένα migration script, το οποίο στη συνέχεια εκτελέστηκε για την εφαρμογή των αλλαγών.

- **Δημιουργία Νέων Πινάκων:** Τα migrations χρησιμοποιήθηκαν για τη δημιουργία νέων πινάκων στη βάση δεδομένων, εξασφαλίζοντας ότι η δομή της βάσης δεδομένων είναι ενημερωμένη και συνεπής.
- **Τροποποίηση Υπαρχόντων Πινάκων:** Τα migrations επέτρεψαν την προσθήκη νέων στηλών, την τροποποίηση των τύπων δεδομένων και την εφαρμογή νέων περιορισμών στους υπάρχοντες πίνακες.
- **Διαγραφή Πινάκων:** Migrations χρησιμοποιήθηκαν για τη διαγραφή πινάκων που δεν χρειάζονταν πλέον, διασφαλίζοντας την καθαριότητα και την αποδοτικότητα της βάσης δεδομένων.

## Sequelize-auto

Το sequelize-auto <sup>13</sup> είναι ένα πακέτο που επιτρέπει την αυτόματη δημιουργία των Sequelize models από το σχήμα της βάσης δεδομένων. Αυτό το εργαλείο είναι ιδιαίτερα χρήσιμο για έργα που χρησιμοποιούν υπάρχουσες βάσεις δεδομένων, καθώς διευκολύνει τη δημιουργία των models χωρίς την ανάγκη χειροκίνητης κωδικοποίησης.

## Χαρακτηριστικά του sequelize-auto

- **Αυτόματη Δημιουργία Models:** Το sequelize-auto σαρώνει το σχήμα της βάσης δεδομένων και δημιουργεί αυτόματα τα Sequelize models που αντιστοιχούν στους πίνακες της βάσης δεδομένων.
- **Υποστήριξη για Συσχετίσεις:** Το sequelize-auto αναγνωρίζει τις συσχετίσεις μεταξύ των πινάκων και τις συμπεριλαμβάνει στα δημιουργημένα models, διευκολύνοντας τη διαχείριση των σχέσεων μεταξύ των δεδομένων.
- **Ευκολία Χρήσης:** Το sequelize-auto παρέχει ένα απλό και κατανοητό CLI εργαλείο που επιτρέπει την εύκολη εκτέλεση της διαδικασίας δημιουργίας των models με μερικές μόνο εντολές.

## Χρήση του sequelize-auto στο Έργο

Στο παρόν έργο, το sequelize-auto χρησιμοποιήθηκε για την αυτόματη δημιουργία των Sequelize models από το σχήμα της βάσης δεδομένων PostgreSQL. Αυτό επέτρεψε την γρήγορη και αποδοτική δημιουργία των models, μειώνοντας τον χρόνο ανάπτυξης και ελαχιστοποιώντας τα σφάλματα.

- **Γρήγορη Εγκατάσταση:** Το sequelize-auto εγκαταστάθηκε και ρυθμίστηκε εύκολα, επιτρέποντας την άμεση έναρξη της διαδικασίας δημιουργίας των models.
- **Αυτόματη Δημιουργία:** Η διαδικασία δημιουργίας των models εκτελέστηκε αυτόματα, με το sequelize-auto να αναγνωρίζει τις στήλες, τους τύπους δεδομένων και τις συσχετίσεις μεταξύ των πινάκων.
- **Ενημέρωση Models:** Το sequelize-auto χρησιμοποιήθηκε επίσης για την ενημέρωση των models όταν έγιναν αλλαγές στο σχήμα της βάσης δεδομένων, διασφαλίζοντας ότι τα models ήταν πάντα ενημερωμένα και συνεπή με τη βάση δεδομένων.

Η χρήση του Sequelize ORM και των σχετικών εργαλείων όπως τα migrations και το sequelize-auto στο έργο εξασφάλισε την αποδοτική διαχείριση της βάσης δεδομένων, διευκολύνοντας την ανάπτυξη και τη συντήρηση της εφαρμογής διαχείρισης αποθήκευσης.

---

## 3.2.4 PostgreSQL

Το PostgreSQL <sup>14</sup> είναι ένα ισχυρό, ανοιχτού κώδικα, αντικείμενο-σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων (DBMS) <sup>15</sup>. Χρησιμοποιείται ευρέως για την αποθήκευση και τη διαχείριση δεδομένων σε διάφορες εφαρμογές, από μικρές προσωπικές ιστοσελίδες έως μεγάλες εταιρικές πλατφόρμες. Στο συγκεκριμένο έργο, η PostgreSQL επιλέχθηκε ως η βάση δεδομένων για την αποθήκευση και τη διαχείριση των δεδομένων της εφαρμογής διαχείρισης αποθήκευσης.

### Χαρακτηριστικά της PostgreSQL

- **Αξιοπιστία και Σταθερότητα:** Η PostgreSQL είναι γνωστή για την αξιοπιστία και τη σταθερότητά του, καθιστώντας το κατάλληλο για εφαρμογές που απαιτούν υψηλή διαθεσιμότητα και συνέπεια δεδομένων.
- **Συμμόρφωση με το SQL Πρότυπο:** Η PostgreSQL συμμορφώνεται με τα πρότυπα SQL, επιτρέποντας τη χρήση γνωστών εντολών SQL για την αλληλεπίδραση με τη βάση δεδομένων.
- **Επεκτασιμότητα:** Η PostgreSQL υποστηρίζει την κλιμάκωση τόσο κάθετα (προσθήκη πόρων σε έναν μεμονωμένο διακομιστή) όσο και οριζόντια (προσθήκη περισσότερων διακομιστών), καθιστώντας το ιδανικό για εφαρμογές που απαιτούν επεκτασιμότητα.
- **Υποστήριξη για Σύνθετους Τύπους Δεδομένων:** Η PostgreSQL υποστηρίζει διάφορους σύνθετους τύπους δεδομένων, όπως JSON, XML, και hstore, επιτρέποντας την αποθήκευση και διαχείριση σύνθετων δομών δεδομένων.
- **Προχωρημένες Λειτουργίες SQL:** Η PostgreSQL παρέχει προχωρημένες λειτουργίες SQL, όπως κοινές εκφράσεις πίνακα (CTEs), window functions, και λειτουργίες ανάλυσης, διευκολύνοντας τη δημιουργία πολύπλοκων ερωτημάτων και αναφορών.

- **Υποστήριξη για Συσχετίσεις:** Η PostgreSQL υποστηρίζει σχέσεις μεταξύ πινάκων, όπως foreign keys, joins, και cascades, διευκολύνοντας τη δημιουργία και διαχείριση πολύπλοκων σχέσεων δεδομένων.

### Επιλογή της PostgreSQL για το Έργο

Η επιλογή της PostgreSQL για την ανάπτυξη της εφαρμογής διαχείρισης αποθήκευσης βασίστηκε σε διάφορους λόγους που το καθιστούν κατάλληλο για τις απαιτήσεις του έργου:

- **Αξιοπιστία και Συνέπεια Δεδομένων:** Λόγω της αξιοπιστίας και της συνέπειας δεδομένων που παρέχει η PostgreSQL, διασφαλίζεται ότι τα δεδομένα αποθηκεύονται και ανακτώνται με ακρίβεια, κάτι που είναι κρίσιμο για τη διαχείριση αποθήκευσης ελαιολάδου.
- **Υποστήριξη για Σύνθετους Τύπους Δεδομένων:** Η δυνατότητα αποθήκευσης και διαχείρισης σύνθετων τύπων δεδομένων, όπως JSON και hstore, διευκολύνει την αναπαράσταση και αποθήκευση δεδομένων ανάλυσης και άλλων πληροφοριών που σχετίζονται με τη διαχείριση αποθήκευσης.
- **Προχωρημένες Λειτουργίες SQL:** Οι προχωρημένες λειτουργίες SQL που παρέχει η PostgreSQL επιτρέπουν τη δημιουργία πολύπλοκων ερωτημάτων και αναφορών, διευκολύνοντας την ανάλυση δεδομένων και την αναφορά.
- **Κλιμάκωση και Απόδοση:** Η δυνατότητα κλιμάκωσης τόσο κάθετα όσο και οριζόντια εξασφαλίζει ότι το σύστημα μπορεί να διαχειριστεί αυξανόμενο φορτίο δεδομένων και χρηστών, ενώ η καλή απόδοση της PostgreSQL διασφαλίζει γρήγορες απαντήσεις σε ερωτήματα και λειτουργίες.
- **Υποστήριξη Κοινότητας:** Η PostgreSQL έχει μια μεγάλη και ενεργή κοινότητα χρηστών και προγραμματιστών που παρέχουν συνεχή υποστήριξη και βελτιώσεις, εξασφαλίζοντας ότι το σύστημα παραμένει ενημερωμένο και ασφαλές.

Η χρήση της PostgreSQL ως βάσης δεδομένων για την εφαρμογή διαχείρισης αποθήκευσης αποδείχθηκε επιτυχημένη, εξασφαλίζοντας την αξιοπιστία, την επεκτασιμότητα και την αποδοτικότητα της διαχείρισης δεδομένων στο σύστημα.

---

## 3.2.5 Άλλες Εξαρτήσεις Εφαρμογής

Στο παρόν έργο χρησιμοποιήθηκαν διάφορες εξαρτήσεις (dependencies) για την ανάπτυξη της εφαρμογής διαχείρισης αποθήκευσης. Αυτές οι εξαρτήσεις διευκόλυναν τη διαδικασία ανάπτυξης, την ασφάλεια, την επικύρωση δεδομένων, τη διαχείριση περιβάλλοντος και τη βελτίωση της παραγωγικότητας. Παρακάτω παρουσιάζονται οι κύριες εξαρτήσεις που χρησιμοποιήθηκαν, με μια σύντομη περιγραφή και τη χρήση τους στο έργο.

### Εξαρτήσεις ανάπτυξης εφαρμογής (devDependencies)

#### 1. eslint

- **Περιγραφή:** Το ESLint <sup>16</sup> είναι ένα εργαλείο ανάλυσης στατικού κώδικα για τον εντοπισμό και τη διόρθωση προβλημάτων στον κώδικα JavaScript.



- **Χρήση:** Χρησιμοποιήθηκε για τη διασφάλιση της ποιότητας του κώδικα και τη συμμόρφωση με τις βέλτιστες πρακτικές προγραμματισμού. Παρείχε προτάσεις και διορθώσεις για κοινά λάθη στον κώδικα.

## 2. eslint-config-prettier

- **Περιγραφή:** Το eslint-config-prettier <sup>17</sup> απενεργοποιεί τους κανόνες του ESLint που μπορεί να έρχονται σε σύγκρουση με το Prettier.
- **Χρήση:** Χρησιμοποιήθηκε για να διασφαλιστεί ότι το ESLint και το Prettier συνεργάζονται χωρίς προβλήματα, εξαλείφοντας τις συγκρούσεις μεταξύ των δύο εργαλείων.

## 3. jshint

- **Περιγραφή:** Το JSHint <sup>18</sup> είναι ένα εργαλείο ανάλυσης στατικού κώδικα για JavaScript που βοηθά στον εντοπισμό προβλημάτων στον κώδικα.
- **Χρήση:** Χρησιμοποιήθηκε ως εναλλακτικό εργαλείο ελέγχου του κώδικα, παρέχοντας επιπλέον επίπεδο ασφάλειας και διασφάλισης ποιότητας.

## 4. nodemon

- **Περιγραφή:** Το Nodemon <sup>19</sup> είναι ένα εργαλείο που παρακολουθεί τις αλλαγές στα αρχεία του έργου και επανεκκινεί αυτόματα τον διακομιστή Node.js.
- **Χρήση:** Χρησιμοποιήθηκε κατά τη διάρκεια της ανάπτυξης για την αυτόματη επανεκκίνηση του διακομιστή κάθε φορά που γίνονταν αλλαγές στον κώδικα, βελτιώνοντας την παραγωγικότητα και επιταχύνοντας τον κύκλο ανάπτυξης.

## 5. prettier

- **Περιγραφή:** Το Prettier <sup>20</sup> είναι ένα εργαλείο μορφοποίησης κώδικα που επιβάλλει ένα συνεπές στυλ γραφής.
- **Χρήση:** Χρησιμοποιήθηκε για τη διασφάλιση της ομοιόμορφης μορφοποίησης του κώδικα, διευκολύνοντας τη συντήρηση και την αναγνωσιμότητα.

# Εξαρτήσεις λειτουργίας εφαρμογής (dependencies)

## 1. bcrypt

- **Περιγραφή:** Το bcrypt <sup>21</sup> είναι μια βιβλιοθήκη για την κρυπτογράφηση κωδικών πρόσβασης.
- **Χρήση:** Χρησιμοποιήθηκε για την ασφαλή κρυπτογράφηση των κωδικών πρόσβασης των χρηστών πριν την αποθήκευσή τους στη βάση δεδομένων.

## 2. cors

- **Περιγραφή:** Το CORS <sup>22</sup> (Cross-Origin Resource Sharing) είναι ένα middleware που επιτρέπει ή περιορίζει αιτήσεις από διαφορετικούς τομείς.
- **Χρήση:** Χρησιμοποιήθηκε για τη ρύθμιση των πολιτικών CORS, επιτρέποντας στο API να δέχεται αιτήσεις από εξουσιοδοτημένους τομείς.

### 3. dotenv

- **Περιγραφή:** Το dotenv <sup>23</sup> είναι μια βιβλιοθήκη που φορτώνει μεταβλητές περιβάλλοντος από ένα αρχείο .env στο process.env.
- **Χρήση:** Χρησιμοποιήθηκε για τη διαχείριση των μεταβλητών περιβάλλοντος, όπως οι ρυθμίσεις της βάσης δεδομένων και τα μυστικά κλειδιά, με ασφαλή και οργανωμένο τρόπο.

### 4. express-validator

- **Περιγραφή:** Το express-validator <sup>24</sup> είναι ένα σύνολο από middlewares για την επικύρωση και τον έλεγχο δεδομένων στο Express.
- **Χρήση:** Χρησιμοποιήθηκε για την επικύρωση των δεδομένων εισόδου από τους χρήστες, διασφαλίζοντας την ακεραιότητα και την ακρίβεια των δεδομένων που αποστέλλονται στο API.

### 5. jsonwebtoken

- **Περιγραφή:** Το jsonwebtoken <sup>25</sup> είναι μια βιβλιοθήκη για τη δημιουργία και την επαλήθευση JSON Web Tokens (JWT).
- **Χρήση:** Χρησιμοποιήθηκε για την υλοποίηση της αυθεντικοποίησης χρηστών μέσω tokens, επιτρέποντας την ασφαλή πρόσβαση στους προστατευμένους πόρους του API.

### 6. pg

- **Περιγραφή:** Το pg <sup>26</sup> είναι ένας πελάτης για την αλληλεπίδραση με βάσεις δεδομένων PostgreSQL από η Node.js.
- **Χρήση:** Χρησιμοποιήθηκε για τη σύνδεση και την εκτέλεση ερωτημάτων στη βάση δεδομένων PostgreSQL, επιτρέποντας την αποθήκευση και την ανάκτηση δεδομένων.

### 7. pg-hstore

- **Περιγραφή:** Το pg-hstore <sup>27</sup> είναι μια βιβλιοθήκη για την υποστήριξη του τύπου δεδομένων hstore του PostgreSQL στο Sequelize.
- **Χρήση:** Χρησιμοποιήθηκε για την υποστήριξη του τύπου δεδομένων hstore στο Sequelize, επιτρέποντας την αποθήκευση και τη διαχείριση hstore δεδομένων στη βάση δεδομένων PostgreSQL.

Η χρήση αυτών των εξαρτήσεων διευκόλυνε την ανάπτυξη της εφαρμογής διαχείρισης αποθήκευσης, επιτρέποντας την υλοποίηση βασικών λειτουργιών με αποδοτικό και οργανωμένο τρόπο, ενώ ταυτόχρονα διασφαλίστηκε η ασφάλεια, η απόδοση και η ποιότητα του κώδικα.

---

### 3.3.1 Παραλαβές (Εισροές) | Oil Receptions (Inflows)

Η υλοποίηση της λειτουργίας των παραλαβών λαδιού (oil receptions) αποτελεί μια από τις βασικές λειτουργίες του συστήματος διαχείρισης αποθήκευσης. Αυτή η λειτουργία περιλαμβάνει τη δημιουργία, την ενημέρωση και τη διαγραφή των εγγραφών που αφορούν τις παραλαβές λαδιού, καθώς και τη διαχείριση των αναλύσεων που σχετίζονται με αυτές. Στο σύστημα, κάθε λειτουργία παραλαβής χειρίζεται από διάφορα στοιχεία του συστήματος, όπως τα Sequelize models, τα services, οι controllers και τα API routes.

#### Στοιχεία που Εμπλέκονται

##### 1. Sequelize Models και Σχετικοί Πίνακες

Οι κύριοι πίνακες που εμπλέκονται στη δημιουργία μιας οντότητας Παραλαβής είναι:

- **IOFlow**: Περιέχει τις περισσότερες πληροφορίες που περιγράφουν μια Παραλαβή.
- **InflowTank**: Δείχνει σε ποια δεξαμενή (tank) αποθηκεύεται η Παραλαβή και σε ποια ποσότητα.
- **AnalysisUnit**: Δείχνει τις χημικές ιδιότητες του εισερχόμενου λαδιού μέσω της Παραλαβής και φιλοξενείται σε:
- **AnalysisCard**: Περιέχει το AnalysisUnit που αντιστοιχεί στη Παραλαβή.
- **AnalysisUnitIOFlow**: Δείχνει ποιο AnalysisUnit αντιστοιχεί στη νέα Παραλαβή.

Οι σχέσεις μεταξύ αυτών των πινάκων είναι οι εξής:

- IOFlow
  - : Σχετίζεται με τους πίνακες Users και BusinessPartners.
    - overseered\_by > Users.u\_id (many-to-one)
    - validated\_by > Users.u\_id (many-to-one)
    - business\_partner\_id > BusinessPartners.bp\_id (many-to-one)
- InflowTank
  - : Σχετίζεται με τον πίνακα Tanks.
    - t\_id > Tanks.t\_id (many-to-one)

##### 2. Υπηρεσίες (Services)

Η δημιουργία των εγγραφών στους αντίστοιχους πίνακες πραγματοποιείται από τις εξής υπηρεσίες:

- **IOFlowService**: Χειρίζεται τις λειτουργίες CRUD για τις εγγραφές στον πίνακα IOFlow.
- **InflowTankService**: Χειρίζεται τις λειτουργίες CRUD για τις εγγραφές στον πίνακα InflowTank.
- **AnalysisService**: Χειρίζεται τις λειτουργίες CRUD για τις εγγραφές στους πίνακες AnalysisCard, AnalysisUnit και AnalysisUnitIOFlow.

##### 3. Controllers και API Routes

Οι routes που αφορούν τις παραλαβές (Inflows) βρίσκονται στο **InflowsRouter** (χτισμένο με Express.js), το οποίο διαβιβάζει τα αιτήματα στη σωστή λειτουργία του InflowController. Το **InflowController** περιέχει τη λογική για την εξυπηρέτηση των αιτημάτων που σχετίζονται με τις παραλαβές, καλώντας τις αντίστοιχες λειτουργίες των υπηρεσιών IOFlowService και InflowTankService και στέλνοντας τις κατάλληλες απαντήσεις πίσω στους χρήστες.

## Δημιουργία μιας Νέας Παραλαβής

Η διαδικασία δημιουργίας μιας νέας παραλαβής περιλαμβάνει τα εξής βήματα:

1. **Αίτημα Δημιουργίας:** Ο χρήστης στέλνει ένα αίτημα δημιουργίας μέσω του API χρησιμοποιώντας τη διαδρομή (route) που παρέχεται από το InflowsRouter.
2. **Διαχείριση Αιτήματος:** Το InflowController λαμβάνει το αίτημα και καλεί τις αντίστοιχες λειτουργίες των υπηρεσιών IOFlowService και InflowTankService για τη δημιουργία των νέων εγγραφών στους πίνακες IOFlow και InflowTank.
3. **Δημιουργία ή Ανάθεση AnalysisUnit:** Αν υπάρχει ήδη ένα AnalysisUnit, μπορεί να ανατεθεί στη νέα παραλαβή μέσω της δημιουργίας μιας νέας εγγραφής στον πίνακα AnalysisUnitIOFlow. Διαφορετικά, δημιουργείται ένα νέο AnalysisUnit μαζί με μια νέα εγγραφή στον πίνακα AnalysisUnitIOFlow.
4. **Αποθήκευση στη Βάση Δεδομένων:** Οι νέες εγγραφές αποθηκεύονται στη βάση δεδομένων, συνδένοντας τα δεδομένα της παραλαβής με τα αντίστοιχα δεδομένα ανάλυσης και δεξαμενών.

## Διαχείριση των Παραλαβών

Μετά τη δημιουργία μιας νέας παραλαβής, ο χρήστης μπορεί να χρησιμοποιήσει τις διαδρομές του API για να εκτελέσει διάφορες λειτουργίες, όπως η ενημέρωση ή η διαγραφή της παραλαβής. Οι διαδρομές αυτές επιτρέπουν την αναζήτηση και την ανάκτηση των δεδομένων της παραλαβής είτε μέσω μοναδικών αναγνωριστικών είτε μέσω κριτηρίων αντιστοίχισης. Αλλαγή στα δεδομένα που βρίσκονται στο πίνακα **IOFlows** καθιστούν μία παραλαβή μη-επικυρωμένη ακόμη και εάν είχε ελεγχθεί και επικυρωθεί από κάποιο διαχειριστή.

## Επικύρωση Παραλαβών

Μία νέα Παραλαβή όταν δημιουργηθεί είναι μη-επικυρωμένη. Είναι καθήκον ενός Διαχειριστή (admin) του συστήματος, να τη αναγνώσει και να την επικυρώσει. Η διαδικασία αυτή εισήχθη στο σύστημα σαν μορφή ελέγχου εισαγωγής των δεδομένων, λαμβάνοντας υπόψη ότι τόσο η λειτουργία της Παραλαβής όσο και αυτή της Φόρτωσης είναι κομβικής σημασίας για την εταιρεία και τα δεδομένα που περιέχουν αντιστοιχούν σε οικονομικά μεγέθη. Μία μη επικυρωμένη Παραλαβή χρησιμοποιείται κανονικά από το σύστημα για υπολογισμούς που αφορούν ποσότητες και ποιότητες λαδιών εντός της αποθήκης.

## Παραδείγματα:

*Παράδειγμα Αιτήματος Δημιουργίας Νέας Παραλαβής*

```
1 POST {{baseUrl}}/inflows/new
2 Content-Type: application/json
3
4 {
5   "ioflowData": {
6     "type": "Reception",
7     "ioflow_date": "2023-11-12T15:47:56.789Z",
8     "business_partner_alias": "ΑΣ Αυγενικής",
9     "quantity_gross": 6500,
10    "quantity_tare": 5400,
11    "quantity_net": 1100,
12    "comments": "Deftero apo 2",
13    "overseered_by": 6,
14    "quality_cat": "Extra Virgin",
15    "analysisData": {
16      "analysisCardData": {
```

```

17     "analysis_date": "2023-11-12T13:55:16.789Z",
18     "analysis_place": "Reception-Spot",
19     "overseered_by": 6,
20     "analysisUnitData" : [
21         {
22             "sampling_date": "2023-11-12T15:48:11.591Z",
23             "business_partner_alias": "ΑΣ Αυγενικής",
24             "sample_name": "Παραλαβή ΑΣ Αυγενικής",
25             "acidity": 0.5
26         }
27     ]
28 },
29 "tankData" : [
30     {
31         "name": "D4",
32         "quantity": 1100
33     }
34 ]
35 }
36 }
37 }

```

Η υλοποίηση των παραλαβών λαδιού στο σύστημα διαχείρισης αποθήκευσης εξασφαλίζει την αποδοτική και ασφαλή διαχείριση των δεδομένων, επιτρέποντας στους χρήστες να καταχωρούν, να ενημερώνουν και να διαγράφουν παραλαβές, καθώς και να συνδέουν αυτές με τις αντίστοιχες χημικές αναλύσεις.

### 3.3.2 Φορτώσεις (Εκροές) | Oil Loadings (Outflows)

Η υλοποίηση της λειτουργίας των φορτώσεων λαδιού (oil loadings) αποτελεί μια από τις βασικές λειτουργίες του συστήματος διαχείρισης αποθήκευσης. Αυτή η λειτουργία περιλαμβάνει τη δημιουργία, την ενημέρωση και τη διαγραφή των εγγραφών που αφορούν τις φορτώσεις λαδιού, καθώς και τη διαχείριση των αναλύσεων που σχετίζονται με αυτές. Στο σύστημα, κάθε λειτουργία φόρτωσης χειρίζεται από διάφορα στοιχεία του συστήματος, όπως τα Sequelize models, τα services, οι controllers και τα API routes.

#### Στοιχεία που Εμπλέκονται

##### 1. Sequelize Models και Σχετικοί Πίνακες

Οι κύριοι πίνακες που εμπλέκονται στη δημιουργία μιας οντότητας Φόρτωση είναι:

- **IOFlow:** Περιέχει τις περισσότερες πληροφορίες που περιγράφουν μια Φόρτωση.
- **Shipments:** Περιέχει επιπλέον πληροφορίες που είναι συγκεκριμένες για τις φορτώσεις, όπως τα στοιχεία του οδηγού και του οχήματος, καθώς και ένα snapshot των δεξαμενών που εμπλέκονται στη φόρτωση.
- **InflowTank:** Οι υπάρχουσες εγγραφές ενημερώνονται για να αντανakλούν τις αλλαγές στις ποσότητες λαδιού στις δεξαμενές.

- **AnalysisUnit:** Δείχνει τις χημικές ιδιότητες του εξερχόμενου λαδιού μέσω της φόρτωσης και φιλοξενείται σε:
- **AnalysisCard:** Φιλοξενεί το AnalysisUnit που αντιστοιχεί στη φόρτωση.
- **AnalysisUnitIOfFlow:** Δείχνει ποιο AnalysisUnit αντιστοιχεί στο νέα Φόρτωση.

Οι σχέσεις μεταξύ αυτών των πινάκων είναι οι εξής:

- **IOFlow**  
: Σχετίζεται με τους πίνακες Users και BusinessPartners.
  - overseered\_by > Users.u\_id (many-to-one)
  - validated\_by > Users.u\_id (many-to-one)
  - business\_partner\_id > BusinessPartners.bp\_id (many-to-one)
- **InflowTank**  
: Σχετίζεται με τον πίνακα Tanks.
  - t\_id > Tanks.t\_id (many-to-one)

## 2. Υπηρεσίες (Services)

Η δημιουργία των εγγραφών στους αντίστοιχους πίνακες πραγματοποιείται από τις εξής υπηρεσίες:

- **IOFlowService:** Χειρίζεται τις λειτουργίες CRUD για τις εγγραφές στον πίνακα IOFlow.
- **InflowTankService:** Χειρίζεται τις λειτουργίες CRUD για τις εγγραφές στον πίνακα InflowTank.
- **AnalysisService:** Χειρίζεται τις λειτουργίες CRUD για τις εγγραφές στους πίνακες AnalysisCard, AnalysisUnit και AnalysisUnitIOfFlow.
- **TankService:** Χειρίζεται τις λειτουργίες διαχείρισης των δεξαμενών και των καταστάσεων τους.

## 3. Controllers και API Routes

Οι routes που αφορούν τις φορτώσεις (Outflows) βρίσκονται στο **OutflowsRouter**, το οποίο διαβιβάζει τα αιτήματα στη σωστή λειτουργία του OutflowController. Το **OutflowController** περιέχει τη λογική για την εξυπηρέτηση των αιτημάτων που σχετίζονται με τις φορτώσεις, καλώντας τις αντίστοιχες λειτουργίες των υπηρεσιών IOFlowService και InflowTankService και στέλνοντας τις κατάλληλες απαντήσεις πίσω στους χρήστες.

### Δημιουργία μιας Νέας Φόρτωσης

Η διαδικασία δημιουργίας μιας νέας φόρτωσης περιλαμβάνει τα εξής βήματα:

1. **Αίτημα Δημιουργίας:** Ο χρήστης στέλνει ένα αίτημα δημιουργίας μέσω του API χρησιμοποιώντας τη διαδρομή (route) που παρέχεται από το OutflowsRouter.
2. **Διαχείριση Αιτήματος:** Το OutflowController λαμβάνει το αίτημα και καλεί τις αντίστοιχες λειτουργίες των υπηρεσιών IOFlowService και InflowTankService για τη δημιουργία των νέων εγγραφών στους πίνακες IOFlow και Shipments.
3. **Δημιουργία ή Ανάθεση AnalysisUnit:** Αν υπάρχει ήδη ένα AnalysisUnit, μπορεί να ανατεθεί στη νέα φόρτωση μέσω της δημιουργίας μιας νέας εγγραφής στον πίνακα AnalysisUnitIOfFlow. Διαφορετικά, δημιουργείται ένα νέο AnalysisUnit μαζί με μια νέα εγγραφή στον πίνακα AnalysisUnitIOfFlow.
4. **Αποθήκευση στη Βάση Δεδομένων:** Οι νέες εγγραφές αποθηκεύονται στη βάση δεδομένων, συνδέοντας τα δεδομένα της φόρτωσης με τα αντίστοιχα δεδομένα ανάλυσης και δεξαμενών.

## Διαχείριση των Φορτώσεων

Μετά τη δημιουργία μιας νέας φόρτωσης, ο χρήστης μπορεί να χρησιμοποιήσει τις διαδρομές του API για να εκτελέσει διάφορες λειτουργίες, όπως η ενημέρωση ή η διαγραφή της φόρτωσης. Οι διαδρομές αυτές επιτρέπουν την αναζήτηση και την ανάκτηση των δεδομένων της φόρτωσης είτε μέσω μοναδικών αναγνωριστικών είτε μέσω κριτηρίων αντιστοίχισης. Αλλαγή στα δεδομένα που βρίσκονται στο πίνακα **IOFlows** καθιστούν μία φόρτωση μη-επικυρωμένη ακόμη και εάν είχε ελεγχθεί και επικυρωθεί από κάποιο διαχειριστή.

## Επικύρωση Φορτώσεων

Μία νέα Φόρτωση όταν δημιουργηθεί είναι μη-επικυρωμένη. Είναι καθήκον ενός Διαχειριστή (admin) του συστήματος, να τη αναγνώσει και να την επικυρώσει. Η διαδικασία αυτή εισήχθη στο σύστημα σαν μορφή ελέγχου εισαγωγής των δεδομένων, λαμβάνοντας υπόψη ότι τόσο η λειτουργία της Παραλαβής όσο και αυτή της Φόρτωσης είναι κομβικής σημασίας για την εταιρεία και τα δεδομένα που περιέχουν αντιστοιχούν σε οικονομικά μεγέθη. Μία μη επικυρωμένη Φόρτωση χρησιμοποιείται κανονικά από το σύστημα για υπολογισμούς που αφορούν ποσότητες και ποιότητες λαδιών εντός της αποθήκης.

## Παραδείγματα Λειτουργιών

### Παράδειγμα Αιτήματος Δημιουργίας Νέας Φόρτωσης

```
1  POST {{baseUrl}}/outflows/new
2  Content-Type: application/json
3
4  {
5    "ioflowData": {
6      "type": "Loading",
7      "ioflow_date": "2024-05-06T12:51:56.981Z",
8      "business_partner_alias": "Ierapetra AE",
9      "quantity_gross": 19000,
10     "quantity_tare": 14000,
11     "quantity_net": 5000,
12     "comments": null,
13     "overseered_by": 6,
14     "quality_cat": "Extra virgin",
15     "analysisData": {
16       "analysisCardData": {
17         "analysis_date": "2024-05-06T12:55:56.981Z",
18         "analysis_place": "Reception-Spot",
19         "overseered_by": 6,
20         "analysisUnitData" : [
21           {
22             "sampling_date": "2024-05-06T12:55:56.981Z",
23             "business_partner_alias": "Ierapetra AE",
24             "sample_name": "Φόρτωση Ierapetra AE 2024-05-06",
25             "acidity": 0.65
26           }
27         ]
28       }
29     },
30     "tankData" : [
31       {
32         "name": "D4",
33         "quantity": 1000
```

```

34     },
35     {
36         "name": "D1",
37         "quantity": 4000
38     }
39 ],
40 "shipmentData" : {
41     "driver": "Λαμπράκης Χάρης",
42     "vehicle_plate": "XZK 3380",
43     "tanker_plate": "P 51607",
44     "load_seals": ["IanthosSA 5610", "IanthosSA 5611", "IanthosSA 5612",
45                   "IanthosSA 5608"]
46 }
47 }

```

Η υλοποίηση των φορτώσεων λαδιού στο σύστημα διαχείρισης αποθήκευσης εξασφαλίζει την αποδοτική και ασφαλή διαχείριση των δεδομένων, επιτρέποντας στους χρήστες να καταχωρούν, να ενημερώνουν και να διαγράφουν φορτώσεις, καθώς και να συνδέουν αυτές με τις αντίστοιχες χημικές αναλύσεις και να διαχειρίζονται τις ποσότητες λαδιού στις δεξαμενές.

### 3.3.3 Μεταφορές | Oil Transfers

Η υλοποίηση της λειτουργίας των μεταφορών λαδιού (oil transfers) μεταξύ δεξαμενών αποτελεί μια από τις βασικές λειτουργίες του συστήματος διαχείρισης αποθήκευσης. Αυτή η λειτουργία περιλαμβάνει τη δημιουργία, την ενημέρωση και τη διαγραφή των εγγραφών που αφορούν τις μεταφορές λαδιού. Στο σύστημα, κάθε λειτουργία μεταφοράς χειρίζεται από διάφορα στοιχεία του συστήματος, όπως τα Sequelizeze models, τα services, οι controllers και τα API routes.

#### Στοιχεία που Εμπλέκονται

##### 1. Sequelizeze Models και Σχετικοί Πίνακες

Οι κύριοι πίνακες που εμπλέκονται στη δημιουργία μιας οντότητας Transfer είναι:

- **Transfers:** Περιέχει τις περισσότερες πληροφορίες που περιγράφουν μια Transfer.
- **InflowTank:** Οι υπάρχουσες εγγραφές ενημερώνονται για να αντανakλούν τις αλλαγές στις ποσότητες λαδιού στις δεξαμενές και ενδέχεται να εισαχθούν νέες εγγραφές για τις ποσότητες λαδιού στις δεξαμενές που λαμβάνουν τη μεταφορά.

Οι σχέσεις μεταξύ αυτών των πινάκων είναι οι εξής:

- Transfers
  - : Σχετίζεται με τους πίνακες Users και Tanks.
    - overseered\_by > Users.u\_id (many-to-one)
    - from\_tank > Tanks.t\_id (many-to-one)
    - to\_tank > Tanks.t\_id (many-to-one)
- InflowTank
  - : Σχετίζεται με τους πίνακες Tanks και IOFlows.



- `t_id > Tanks.t_id` (many-to-one)
- `i_id > IOFlows.io_id` (many-to-one)

## 2. Υπηρεσίες (Services)

Η δημιουργία των εγγραφών στους αντίστοιχους πίνακες πραγματοποιείται από τις εξής υπηρεσίες:

- **TransferService:** Χειρίζεται τις λειτουργίες CRUD για τις εγγραφές στον πίνακα Transfers.
- **InflowTankService:** Χειρίζεται τις λειτουργίες CRUD για τις εγγραφές στον πίνακα InflowTank.

## 3. Controllers και API Routes

Οι routes που αφορούν τις μεταφορές (Transfers) βρίσκονται στο **TransfersRouter**, το οποίο διαβιβάζει τα αιτήματα στη σωστή λειτουργία του TransferController. Το **TransferController** περιέχει τη λογική για την εξυπηρέτηση των αιτημάτων που σχετίζονται με τις μεταφορές, καλώντας τις αντίστοιχες λειτουργίες των υπηρεσιών TransferService και InflowTankService και στέλνοντας τις κατάλληλες απαντήσεις πίσω στους χρήστες.

### Δημιουργία μιας Νέας Transfer

Η διαδικασία δημιουργίας μιας νέας μεταφοράς περιλαμβάνει τα εξής βήματα:

1. **Αίτημα Δημιουργίας:** Ο χρήστης στέλνει ένα αίτημα δημιουργίας μέσω του API χρησιμοποιώντας τη διαδρομή (route) που παρέχεται από το TransfersRouter.
2. **Διαχείριση Αιτήματος:** Το TransferController λαμβάνει το αίτημα και καλεί τις αντίστοιχες λειτουργίες των υπηρεσιών TransferService και InflowTankService για τη δημιουργία των νέων εγγραφών στους πίνακες Transfers και InflowTank.
3. **Αποθήκευση στη Βάση Δεδομένων:** Οι νέες εγγραφές αποθηκεύονται στη βάση δεδομένων, συνδέοντας τα δεδομένα της μεταφοράς με τα αντίστοιχα δεδομένα των δεξαμενών.

### Διαχείριση των Transfer

Μετά τη δημιουργία μιας νέας μεταφοράς, ο χρήστης μπορεί να χρησιμοποιήσει τις διαδρομές του API για να εκτελέσει διάφορες λειτουργίες, όπως η ενημέρωση ή η διαγραφή της μεταφοράς. Οι διαδρομές αυτές επιτρέπουν την αναζήτηση και την ανάκτηση των δεδομένων της μεταφοράς είτε μέσω μοναδικών αναγνωριστικών είτε μέσω κριτηρίων αντιστοίχισης.

Η υλοποίηση των μεταφορών λαδιού στο σύστημα διαχείρισης αποθήκευσης εξασφαλίζει την αποδοτική και ασφαλή διαχείριση των δεδομένων, επιτρέποντας στους χρήστες να καταχωρούν, να ενημερώνουν και να διαγράφουν μεταφορές, καθώς και να διαχειρίζονται τις ποσότητες λαδιού στις δεξαμενές.

## 3.3.4 Χημικές Αναλύσεις | Oil Chemical Analyses

Η υλοποίηση της λειτουργίας των χημικών αναλύσεων του λαδιού αποτελεί μια από τις βασικές λειτουργίες του συστήματος διαχείρισης αποθήκευσης. Αυτή η λειτουργία περιλαμβάνει τη δημιουργία, την ενημέρωση και τη διαγραφή των εγγραφών που αφορούν τις χημικές αναλύσεις, καθώς και τη διαχείριση των δεδομένων που σχετίζονται με αυτές. Στο σύστημα, κάθε λειτουργία χημικής ανάλυσης χειρίζεται από διάφορα στοιχεία του συστήματος, όπως τα Sequelizeze models, τα services, οι controllers και τα API routes.

## Στοιχεία που Εμπλέκονται

### 1. Sequelize Models και Σχετικοί Πίνακες

Οι κύριοι πίνακες που εμπλέκονται στη δημιουργία μιας οντότητας Analysis είναι:

- **AnalysisCard**: Φιλοξενεί μια συλλογή από AnalysisUnits και πρέπει να έχει τουλάχιστον ένα AnalysisUnit.
- **AnalysisUnit**: Δείχνει τις χημικές ιδιότητες του δείγματος λαδιού.
- **ExtraAnalysis**: Παρέχει επιπλέον πληροφορίες που δεν εμφανίζονται στο AnalysisUnit με τη μορφή ζευγών κλειδιού-τιμής (key-value pairs).

Οι σχέσεις μεταξύ αυτών των πινάκων είναι οι εξής:

- AnalysisCards  
: Σχετίζεται με τον πίνακα Users.
  - overseered\_by > Users.u\_id (many-to-one)
- AnalysisUnit  
: Σχετίζεται με τους πίνακες IOFlows, SampleCodes και Tanks μέσω των πινάκων:
  - AnalysisUnitIOFlow
  - AnalysisUnitSampleCode
  - AnalysisUnitTanks

### 2. Υπηρεσίες (Services)

Η δημιουργία των εγγραφών στους αντίστοιχους πίνακες πραγματοποιείται από τις εξής υπηρεσίες:

- **AnalysisService**: Χειρίζεται τις λειτουργίες CRUD για τις εγγραφές στους πίνακες AnalysisCard, AnalysisUnit και ExtraAnalysis.

### 3. Controllers και API Routes

Οι routes που αφορούν τις χημικές αναλύσεις (Analyses) βρίσκονται στο **AnalysisRouter**, το οποίο διαβιβάζει τα αιτήματα στη σωστή λειτουργία του AnalysisController. Το **AnalysisController** περιέχει τη λογική για την εξυπηρέτηση των αιτημάτων που σχετίζονται με τις χημικές αναλύσεις, καλώντας τις αντίστοιχες λειτουργίες της υπηρεσίας AnalysisService και στέλνοντας τις κατάλληλες απαντήσεις πίσω στους χρήστες.

#### Δημιουργία μιας Νέας Χημικής Ανάλυσης

Η διαδικασία δημιουργίας μιας νέας χημικής ανάλυσης περιλαμβάνει τα εξής βήματα:

1. **Αίτημα Δημιουργίας**: Ο χρήστης στέλνει ένα αίτημα δημιουργίας μέσω του API χρησιμοποιώντας τη διαδρομή (route) που παρέχεται από το AnalysisRouter.
2. **Διαχείριση Αιτήματος**: Το AnalysisController λαμβάνει το αίτημα και καλεί τις αντίστοιχες λειτουργίες της υπηρεσίας AnalysisService για τη δημιουργία των νέων εγγραφών στους πίνακες AnalysisCard και AnalysisUnit.
3. **Δημιουργία ή Ανάθεση ExtraAnalysis**: Αν υπάρχει ανάγκη αποθήκευσης επιπλέον πληροφοριών πέρα από τις συνήθεις αναλύσεις (acidity, hyperoxide, kappa), μπορούν να δημιουργηθούν νέες εγγραφές στον πίνακα ExtraAnalysis και να ανατεθούν στο αντίστοιχο AnalysisUnit.
4. **Αποθήκευση στη Βάση Δεδομένων**: Οι νέες εγγραφές αποθηκεύονται στη βάση δεδομένων, συνδένοντας τα δεδομένα της χημικής ανάλυσης με τα αντίστοιχα δεδομένα των δειγμάτων λαδιού.

## Διαχείριση των Χημικών Αναλύσεων

Μετά τη δημιουργία μιας νέας χημικής ανάλυσης, ο χρήστης μπορεί να χρησιμοποιήσει τις διαδρομές του API για να εκτελέσει διάφορες λειτουργίες, όπως η ενημέρωση ή η διαγραφή της ανάλυσης. Οι διαδρομές αυτές επιτρέπουν την αναζήτηση και την ανάκτηση των δεδομένων της ανάλυσης είτε μέσω μοναδικών αναγνωριστικών είτε μέσω κριτηρίων αντιστοίχισης.

## Παραδείγματα Λειτουργιών

### Παράδειγμα Αιτήματος Δημιουργίας Νέας Χημικής Ανάλυσης

```
1 POST {{baseUrl}}/analysis/new
2 Content-Type: application/json
3
4 {
5   "analysisCardData": {
6     "analysis_date": "2023-11-17T10:00:56.789Z",
7     "analysis_place": "Χημείο Analysis - Φακουρέλης Νίκος - Χημικό
Εργαστήριο Τροφίμων",
8     "overseered_by": 7,
9     "comments": "Αναλύσεις από Φακουρέλη",
10    "analysisUnitData" : [
11      {
12        "sampling_date": "2023-11-15T08:34:56.789Z",
13        "business_partner_alias": "ΙΑΝΘΟΣ Α.Ε.",
14        "sample_name": "Δ1",
15        "acidity": 0.32,
16        "kappa_232": 1.43,
17        "kappa_270": 0.14,
18        "peroxide": 8
19      },
20      {
21        "sampling_date": "2023-11-15T08:34:56.789Z",
22        "business_partner_alias": "ΙΑΝΘΟΣ Α.Ε.",
23        "sample_name": "Δ2",
24        "acidity": 0.29,
25        "kappa_232": 1.50,
26        "kappa_270": 0.13,
27        "peroxide": 9
28      },
29      {
30        "sampling_date": "2023-11-15T08:40:56.789Z",
31        "business_partner_alias": "ΙΑΝΘΟΣ Α.Ε.",
32        "sample_name": "ΔΥΠ1",
33        "comments": "Βιομηχανικό",
34        "extraAnalysis": [
35          {
36            "type": "keria_1",
37            "value": "keriaValue1"
38          },
39          {
40            "type": "keria_2",
41            "value": "keriaValue2"
42          }
43        ]
44      }
45    ]
46  }
```

```
45     ]
46   }
47 }
```

Η υλοποίηση των χημικών αναλύσεων στο σύστημα διαχείρισης αποθήκευσης εξασφαλίζει την αποδοτική και ασφαλή διαχείριση των δεδομένων, επιτρέποντας στους χρήστες να καταχωρούν, να ενημερώνουν και να διαγράφουν αναλύσεις, καθώς και να διαχειρίζονται τις χημικές ιδιότητες του λαδιού που εισέρχεται και εξέρχεται από την αποθήκη.

### 3.3.5 Δειγματοληψία Λαδιού | Oil Sampling

Η υλοποίηση της λειτουργίας της δειγματοληψίας λαδιού (oil sampling) αποτελεί μια από τις βασικές λειτουργίες του συστήματος διαχείρισης αποθήκευσης. Αυτή η λειτουργία περιλαμβάνει τη δημιουργία, την ενημέρωση και τη διαγραφή των εγγραφών που αφορούν τα δείγματα λαδιού, καθώς και τη διαχείριση των δεδομένων που σχετίζονται με αυτά. Στο σύστημα, κάθε λειτουργία δειγματοληψίας χειρίζεται από διάφορα στοιχεία του συστήματος, όπως τα Sequelizeze models, τα services, οι controllers και τα API routes.

#### Στοιχεία που Εμπλέκονται

##### 1. Sequelizeze Models και Σχετικοί Πίνακες

Οι κύριοι πίνακες που εμπλέκονται στη δημιουργία μιας οντότητας SampleCard είναι:

- **SampleCard:** Φιλοξενεί μια συλλογή από SampleCodes και πρέπει να έχει τουλάχιστον ένα SampleCode.
- **SampleCode:** Δείχνει τις χημικές ιδιότητες του δείγματος λαδιού και επιτρέπει την έκφραση της σύνθεσης του δείγματος από έως και τρεις διαφορετικές πηγές.

Οι σχέσεις μεταξύ αυτών των πινάκων είναι οι εξής:

- **SampleCards**
  - : Σχετίζεται με τον πίνακα Users.
    - sampler > Users.u\_id (many-to-one)
- **SampleCode:** Μπορεί να σχετίζεται με τον πίνακα AnalysisUnit μέσω του πίνακα AnalysisUnitSampleCode.

##### 2. Υπηρεσίες (Services)

Η δημιουργία των εγγραφών στους αντίστοιχους πίνακες πραγματοποιείται από τις εξής υπηρεσίες:

- **SampleService:** Χειρίζεται τις λειτουργίες CRUD για τις εγγραφές στους πίνακες SampleCard και SampleCode.

##### 3. Controllers και API Routes

Οι routes που αφορούν τα δείγματα (Samples) βρίσκονται στο **SampleRouter**, το οποίο διαβιβάζει τα αιτήματα στη σωστή λειτουργία του SampleController. Το **SampleController** περιέχει τη λογική για την εξυπηρέτηση των αιτημάτων που σχετίζονται με τα δείγματα, καλώντας τις αντίστοιχες λειτουργίες της υπηρεσίας SampleService και στέλνοντας τις κατάλληλες απαντήσεις πίσω στους χρήστες.

## Δημιουργία μιας Νέας SampleCard

Η διαδικασία δημιουργίας μιας νέας δειγματοληψίας περιλαμβάνει τα εξής βήματα:

1. **Αίτημα Δημιουργίας:** Ο χρήστης στέλνει ένα αίτημα δημιουργίας μέσω του API χρησιμοποιώντας τη διαδρομή (route) που παρέχεται από το SampleRouter.
2. **Διαχείριση Αιτήματος:** Το SampleController λαμβάνει το αίτημα και καλεί τις αντίστοιχες λειτουργίες της υπηρεσίας SampleService για τη δημιουργία των νέων εγγγραφών στους πίνακες SampleCard και SampleCode.
3. **Έκφραση Σύνθεσης Δείγματος:** Ο χρήστης μπορεί να εκφράσει τη σύνθεση του δείγματος από έως και τρεις διαφορετικές πηγές, χρησιμοποιώντας ζεύγη κλειδιού-τιμής (source: , percentage of participation in the mixture: ).
4. **Αποθήκευση στη Βάση Δεδομένων:** Οι νέες εγγραφές αποθηκεύονται στη βάση δεδομένων, συνδέοντας τα δεδομένα της δειγματοληψίας με τα αντίστοιχα δεδομένα των δειγμάτων λαδιού.

## Διαχείριση των SampleCard

Μετά τη δημιουργία μιας νέας δειγματοληψίας, ο χρήστης μπορεί να χρησιμοποιήσει τις διαδρομές του API για να εκτελέσει διάφορες λειτουργίες, όπως η ενημέρωση ή η διαγραφή της δειγματοληψίας. Οι διαδρομές αυτές επιτρέπουν την αναζήτηση και την ανάκτηση των δεδομένων της δειγματοληψίας είτε μέσω μοναδικών αναγνωριστικών είτε μέσω κριτηρίων αντιστοίχισης.

## Παραδείγματα Λειτουργιών

*Παράδειγμα Αιτήματος Δημιουργίας Νέας Φόρμας Δειγμάτων*

```
1 POST {{baseUrl}}/samples/new
2 Content-Type: application/json
3
4 {
5   "sampleCardData": {
6     "card_title": "Γλυ Castel",
7     "sampling_date": "2023-07-16T12:34:56.789Z",
8     "sampler": 6,
9     "recipient_alias": "Castel",
10    "sampleCodeData" : [
11      {
12        "business_partner_alias": "ΙΑΝΘΟΣ Α.Ε.",
13        "sample_name": "Δ10",
14        "tank1_name": "Δ10",
15        "tank1_percentage": 100.00,
16        "comments": "Λάδι από Αυγενική"
17      },
18      {
19        "business_partner_alias": "ΙΑΝΘΟΣ Α.Ε.",
20        "sample_name": "M12",
21        "tank1_name": "Δ1",
22        "tank1_percentage": 50.00,
23        "tank2_name": "Δ2",
24        "tank2_percentage": 50.00
25      },
26      {
27        "business_partner_alias": "ΙΑΝΘΟΣ Α.Ε.",
28        "sample_name": "M526",
29        "tank1_name": "Δ5",
```

```

30         "tank1_percentage": 50.00,
31         "tank2_name": "Δ2",
32         "tank2_percentage": 25.00,
33         "tank3_name": "Δ6",
34         "tank3_percentage": 25.00
35     }
36 ]
37 }
38 }

```

Η υλοποίηση της δειγματοληψίας λαδιού στο σύστημα διαχείρισης αποθήκευσης εξασφαλίζει την αποδοτική και ασφαλή διαχείριση των δεδομένων, επιτρέποντας στους χρήστες να καταχωρούν, να ενημερώνουν και να διαγράφουν δείγματα, καθώς και να διαχειρίζονται τις χημικές ιδιότητες του λαδιού που αποθηκεύεται και αποστέλλεται από την αποθήκη.

### 3.3.6 Διαχείριση Δεξαμενών

Η υλοποίηση της λειτουργίας καταγραφής και παρακολούθησης των δεξαμενών αποτελεί μια από τις βασικές λειτουργίες του συστήματος διαχείρισης αποθήκευσης. Αυτή η λειτουργία περιλαμβάνει τη δημιουργία, την ενημέρωση και τη διαγραφή των εγγραφών που αφορούν τις δεξαμενές, καθώς και τη διαχείριση των δεδομένων που σχετίζονται με αυτές. Στο σύστημα, κάθε λειτουργία δεξαμενής χειρίζεται από διάφορα στοιχεία του συστήματος, όπως τα Sequelize models, τα services, οι controllers και τα API routes.

#### Στοιχεία που Εμπλέκονται

##### 1. Sequelize Models και Σχετικοί Πίνακες

Ο κύριος πίνακας που εμπλέκεται στη δημιουργία μιας οντότητας Tank είναι:

- **Tanks:** Περιέχει όλες τις πληροφορίες που περιγράφουν μια δεξαμενή, όπως γεωμετρικά και ποιοτικά χαρακτηριστικά, καθώς και το ρόλο της δεξαμενής (reception\_tank ή όχι).

Οι σχέσεις μεταξύ του πίνακα Tanks και άλλων πινάκων περιλαμβάνουν:

- **AnalysisUnit:** Σχετίζεται μέσω του πίνακα AnalysisUnitTank.
- **IOFlows:** Σχετίζεται μέσω του πίνακα InflowTank.

##### 2. Υπηρεσίες (Services)

Η δημιουργία των εγγραφών στον πίνακα Tanks και η διαχείριση των λειτουργιών που σχετίζονται με τις δεξαμενές πραγματοποιείται από την εξής υπηρεσία:

- **TankService:** Χειρίζεται τις λειτουργίες CRUD για τις εγγραφές στον πίνακα Tanks και τις λειτουργίες που σχετίζονται με τη διαχείριση των δεξαμενών.

##### 3. Controllers και API Routes

Οι routes που αφορούν τις δεξαμενές (Tanks) βρίσκονται στο **TankRouter**, το οποίο διαβιβάζει τα αιτήματα στη σωστή λειτουργία του TankController. Το **TankController** περιέχει τη λογική για την εξυπηρέτηση των αιτημάτων που σχετίζονται με τις δεξαμενές, καλώντας τις αντίστοιχες λειτουργίες της υπηρεσίας TankService και στέλνοντας τις κατάλληλες απαντήσεις πίσω στους χρήστες.

## Δημιουργία μιας Νέας Δεξαμενής

Η διαδικασία δημιουργίας μιας νέας δεξαμενής περιλαμβάνει τα εξής βήματα:

1. **Αίτημα Δημιουργίας:** Ο χρήστης στέλνει ένα αίτημα δημιουργίας μέσω του API χρησιμοποιώντας τη διαδρομή (route) που παρέχεται από το TankRouter.
2. **Διαχείριση Αιτήματος:** Το TankController λαμβάνει το αίτημα και καλεί τις αντίστοιχες λειτουργίες της υπηρεσίας TankService για τη δημιουργία της νέας εγγραφής στον πίνακα Tanks.
3. **Καθορισμός Χαρακτηριστικών:** Κατά τη δημιουργία, ο χρήστης πρέπει να καθορίσει τα γεωμετρικά και ποιοτικά χαρακτηριστικά της δεξαμενής, καθώς και το ρόλο της (reception\_tank ή όχι).
4. **Αποθήκευση στη Βάση Δεδομένων:** Η νέα εγγραφή αποθηκεύεται στη βάση δεδομένων, συνδένοντας τα δεδομένα της δεξαμενής με τα αντίστοιχα δεδομένα εισροών και αναλύσεων.

## Λειτουργίες Διαχείρισης των Δεξαμενών

Μετά τη δημιουργία μιας νέας δεξαμενής, ο χρήστης μπορεί να χρησιμοποιήσει τις διαδρομές του API για να εκτελέσει διάφορες λειτουργίες, όπως η ενημέρωση ή η διαγραφή της δεξαμενής. Είναι σημαντικό να αναφερθεί ότι σε μία νεοδημιουργηθείσα δεξαμενή, από τη στιγμή που θα κατατεθούν λάδια, αυτή δεν μπορεί να διαγραφεί από τον χρήστη ακόμη και αν αδειάσει. Οι δεξαμενές που αποσύρονται αντί να διαγράφονται, αλλάζουν την τιμή της ιδιότητας `is_active` σε *false*.

Οι λειτουργίες που μπορούν να εκτελεστούν περιλαμβάνουν:

- **Υπολογισμός Ποσότητας:** Ο χρήστης μπορεί να ζητήσει από το σύστημα να υπολογίσει την τρέχουσα ποσότητα λαδιού στη δεξαμενή, λαμβάνοντας υπόψη τις εισροές που έχουν αποθηκευτεί.
- **Υπολογισμός Οξύτητας:** Ο χρήστης μπορεί να ζητήσει από το σύστημα να υπολογίσει την τρέχουσα οξύτητα του λαδιού στη δεξαμενή, λαμβάνοντας υπόψη τις οξύτητες των εισροών που έχουν αποθηκευτεί.
- **Καθαρισμός και Άδειασμα:** Ο χρήστης μπορεί να ορίσει τη δεξαμενή ως καθαρή και άδεια, επαναφέροντας την ποσότητα και την οξύτητα.
- **Χειροκίνητη Ρύθμιση Ποσότητας:** Ο χρήστης μπορεί να ρυθμίσει χειροκίνητα την τρέχουσα ποσότητα στη δεξαμενή, προσαρμόζοντας, διορθώνοντας και αντισταθμίζοντας τις απώλειες λαδιού μέσω μεταφορών.

Η υλοποίηση του συστήματος παρακολούθησης των δεξαμενών εξασφαλίζει την αποδοτική και ασφαλή διαχείριση των δεδομένων, επιτρέποντας στους χρήστες να καταχωρούν, να ενημερώνουν και να διαγράφουν δεξαμενές, καθώς και να διαχειρίζονται τις ποσότητες και τις ποιοτικές παραμέτρους του λαδιού που αποθηκεύεται στις δεξαμενές.

## 3.4 Υλοποίηση Βοηθητικών Λειτουργιών

### 3.4.1 Αρχείο Πελατών/Προμηθευτών

Η υλοποίηση της λειτουργίας καταγραφής και παρακολούθησης των Πελατών/Προμηθευτών (Business Partners) αποτελεί μια από τις βοηθητικές λειτουργίες του συστήματος διαχείρισης αποθήκευσης. Αυτή η λειτουργία περιλαμβάνει τη δημιουργία, την ενημέρωση και τη διαγραφή των εγγραφών που αφορούν τους Πελάτες/Προμηθευτές, καθώς και τη διαχείριση των δεδομένων που σχετίζονται με αυτούς. Στο σύστημα, κάθε λειτουργία καταγραφής Πελατών/Προμηθευτών χειρίζεται από διάφορα στοιχεία του συστήματος, όπως τα Sequelize models, τα services, οι controllers και τα API routes. Η πρόσβαση, δημιουργία και διαχείριση δεδομένων των Πελατών/Προμηθευτών είναι λειτουργίες δεσμευμένες καθαρά για διαχειριστές του συστήματος.

#### Στοιχεία που Εμπλέκονται

##### 1. Sequelize Models και Σχετικοί Πίνακες

Ο κύριος πίνακας που εμπλέκεται στη δημιουργία μιας οντότητας Πελάτη/Προμηθευτή είναι:

- **BusinessPartners:** Περιέχει όλες τις πληροφορίες που περιγράφουν έναν Πελάτη/Προμηθευτή, όπως το όνομα, τις διευθύνσεις και τους τύπους των Πελατών/Προμηθευτών (customer, supplier, customer-supplier).

Οι σχέσεις μεταξύ του πίνακα BusinessPartners και άλλων πινάκων περιλαμβάνουν:

- IOFlows  
: Σχετίζεται μέσω της στήλης business\_partner\_id.
  - IOFlows.business\_partner\_id > BusinessPartners.bp\_id (many-to-one)

##### 2. Υπηρεσίες (Services)

Η δημιουργία των εγγραφών στον πίνακα BusinessPartners και η διαχείριση των λειτουργιών που σχετίζονται με τους Πελάτες/Προμηθευτές πραγματοποιείται από την εξής υπηρεσία:

- **BusinessPartnerService:** Χειρίζεται τις λειτουργίες CRUD για τις εγγραφές στον πίνακα BusinessPartners και τις λειτουργίες που σχετίζονται με τη διαχείριση των Πελατών/Προμηθευτών.

##### 3. Controllers και API Routes

Οι routes που αφορούν τους Πελάτες/Προμηθευτές βρίσκονται στο BusinessPartnerRouter, το οποίο διαβιβάζει τα αιτήματα στη σωστή λειτουργία του BusinessPartnerController. Το BusinessPartnerController περιέχει τη λογική για την εξυπηρέτηση των αιτημάτων που σχετίζονται με τους Πελάτες/Προμηθευτές, καλώντας τις αντίστοιχες λειτουργίες της υπηρεσίας BusinessPartnerService και στέλνοντας τις κατάλληλες απαντήσεις πίσω στους χρήστες.

#### Δημιουργία μιας Νέας Οντότητας Πελάτη/Προμηθευτή

Η διαδικασία δημιουργίας μιας νέας οντότητας Πελάτη/Προμηθευτή περιλαμβάνει τα εξής βήματα:

1. **Αίτημα Δημιουργίας:** Ο χρήστης στέλνει ένα αίτημα δημιουργίας μέσω του API χρησιμοποιώντας τη διαδρομή (route) που παρέχεται από το BusinessPartnerRouter.
2. **Διαχείριση Αιτήματος:** Το BusinessPartnerController λαμβάνει το αίτημα και καλεί τις αντίστοιχες λειτουργίες της υπηρεσίας BusinessPartnerService για τη δημιουργία της νέας εγγραφής στον πίνακα BusinessPartners.



3. **Καθορισμός Τύπου:** Κατά τη δημιουργία, ο χρήστης πρέπει να καθορίσει τον τύπο του Πελάτη/Προμηθευτή (customer, supplier, customer-supplier).

### Λειτουργίες Διαχείρισης των Πελατών/Προμηθευτών

Μετά τη δημιουργία μιας νέας οντότητας Πελάτη/Προμηθευτή, ο χρήστης μπορεί να χρησιμοποιήσει τις διαδρομές του API για να εκτελέσει διάφορες λειτουργίες, όπως η ενημέρωση ή η διαγραφή της οντότητας. Οι λειτουργίες που μπορούν να εκτελεστούν περιλαμβάνουν:

- **Ενημέρωση Εγγραφών:** Η ενημέρωση των πληροφοριών ενός Πελάτη/Προμηθευτή, όπως το όνομα, η διεύθυνση και οι τύποι των Πελατών/Προμηθευτών.
- **Διαγραφή Εγγραφών:** Η διαγραφή ενός Πελάτη/Προμηθευτή από το σύστημα. Σημειώνεται ότι αυτές οι λειτουργίες είναι περιορισμένες μόνο για χρήστες τύπου Admin.
- **Αναζήτηση και Ανάκτηση Δεδομένων:** Η αναζήτηση και η ανάκτηση των δεδομένων ενός Πελάτη/Προμηθευτή είτε μέσω μοναδικών αναγνωριστικών είτε μέσω κριτηρίων αντιστοίχισης.

### Μηχανισμός Ψευδώνυμων (Alias)

Ο πίνακας BusinessPartners έχει ένα πεδίο `name` που αντιστοιχεί στο επίσημο όνομα του ατόμου ή της εταιρείας συνεργάτη καθώς και ένα πεδίο `alias`. Το `alias` είναι ένα array από strings (αναγνωριστικά / ψευδώνυμα) που ο απλός χρήστης του συστήματος αναθέτει στο πεδίο `business_partnet_alias` κατά τη δημιουργία μιας παραλαβής (Reception) ή φόρτωσης (Loading). Αυτό συμβαίνει επειδή ένας απλός χρήστης του συστήματος δεν γνωρίζει πάντα το επίσημο όνομα του Πελάτη/Προμηθευτή με τον οποίο ασχολείται. Έτσι, αντί να αναζητήσει το επίσημο όνομα και να χάσει χρόνο, αναθέτει ένα ψευδώνυμο στον Πελάτη/Προμηθευτή και γίνεται ευθύνη του admin χρήστη κατά τη διαδικασία επικύρωσης της εγγραφής παραλαβής (Reception) ή φόρτωσης (Loading) να ταιριάζει το ψευδώνυμο με τον σωστό Πελάτη/Προμηθευτή. Στο τέλος της διαδικασίας επικύρωσης, αν δεν υπάρχει ήδη, το ψευδώνυμο προστίθεται στο πεδίο `alias` του αντίστοιχου Πελάτη/Προμηθευτή.

Η υλοποίηση του συστήματος καταγραφής και παρακολούθησης των Πελατών/Προμηθευτών εξασφαλίζει την αποδοτική και ασφαλή διαχείριση των δεδομένων, επιτρέποντας στους χρήστες να καταχωρούν, να ενημερώνουν και να διαγράφουν Πελάτες/Προμηθευτές, καθώς και να διαχειρίζονται τις συναλλαγές και τις σχέσεις με αυτούς.

## 3.4.2 Αρχείο Χρηστών

Η υλοποίηση της λειτουργίας διαχείρισης χρηστών (user management) αποτελεί μια από τις βοηθητικές λειτουργίες του συστήματος διαχείρισης αποθήκευσης. Αυτή η λειτουργία περιλαμβάνει τη δημιουργία, την ενημέρωση και τη διαγραφή των εγγραφών που αφορούν τους χρήστες του συστήματος, καθώς και τη διαχείριση των δεδομένων που σχετίζονται με αυτούς. Στο σύστημα, κάθε λειτουργία διαχείρισης χρηστών χειρίζεται από διάφορα στοιχεία του συστήματος, όπως τα Sequelize models, τα services, οι controllers και τα API routes. Η πρόσβαση, δημιουργία και διαχείριση δεδομένων των χρηστών είναι λειτουργίες δεσμευμένες καθαρά για διαχειριστές του συστήματος.

### Στοιχεία που Εμπλέκονται

#### 1. Sequelize Models και Σχετικοί Πίνακες

Ο κύριος πίνακας που εμπλέκεται στη δημιουργία μιας οντότητας χρήστη είναι:

- **Users:** Περιέχει όλες τις πληροφορίες που περιγράφουν έναν χρήστη, όπως το όνομα χρήστη, το πλήρες όνομα, τον κωδικό πρόσβασης και την κλάση χρήστη (Admin, SimpleUser).

Οι σχέσεις μεταξύ του πίνακα Users και άλλων πινάκων περιλαμβάνουν:

- IOFlows  
: Σχετίζεται μέσω των στηλών overseered\_by και validated\_by.
  - IOFlows.overseered\_by > Users.u\_id (many-to-one)
  - IOFlows.validated\_by > Users.u\_id (many-to-one)
- Transfers  
: Σχετίζεται μέσω της στήλης overseered\_by.
  - Transfers.overseered\_by > Users.u\_id (many-to-one)
- AnalysisCards  
: Σχετίζεται μέσω της στήλης overseered\_by.
  - AnalysisCards.overseered\_by > Users.u\_id (many-to-one)
- SampleCards  
: Σχετίζεται μέσω της στήλης sampler.
  - SampleCards.sampler > Users.u\_id (many-to-one)

## 2. Υπηρεσίες (Services)

Η δημιουργία των εγγραφών στον πίνακα Users και η διαχείριση των λειτουργιών που σχετίζονται με τους χρήστες πραγματοποιείται από την εξής υπηρεσία:

- **UserService:** Χειρίζεται τις λειτουργίες CRUD για τις εγγραφές στον πίνακα Users και τις λειτουργίες που σχετίζονται με τη διαχείριση των χρηστών.

## 3. Controllers και API Routes

Οι routes που αφορούν τους χρήστες βρίσκονται στο **UserRouter**, το οποίο διαβιβάζει τα αιτήματα στη σωστή λειτουργία του UserController. Το **UserController** περιέχει τη λογική για την εξυπηρέτηση των αιτημάτων που σχετίζονται με τους χρήστες, καλώντας τις αντίστοιχες λειτουργίες της υπηρεσίας UserService και στέλνοντας τις κατάλληλες απαντήσεις πίσω στους χρήστες.

### Δημιουργία μιας Νέας Οντότητας Χρήστη

Η διαδικασία δημιουργίας μιας νέας οντότητας χρήστη περιλαμβάνει τα εξής βήματα:

1. **Αίτημα Δημιουργίας:** Ο χρήστης στέλνει ένα αίτημα δημιουργίας μέσω του API χρησιμοποιώντας τη διαδρομή (route) που παρέχεται από το UserRouter.
2. **Διαχείριση Αιτήματος:** Το UserController λαμβάνει το αίτημα και καλεί τις αντίστοιχες λειτουργίες της υπηρεσίας UserService για τη δημιουργία της νέας εγγραφής στον πίνακα Users.
3. **Καθορισμός Κλάσης Χρήστη:** Κατά τη δημιουργία, ο χρήστης πρέπει να καθορίσει την κλάση του χρήστη (Admin, SimpleUser), η οποία καθορίζει τις διαθέσιμες λειτουργίες και τα δικαιώματα πρόσβασης στα δεδομένα του συστήματος.
4. **Αποθήκευση στη Βάση Δεδομένων:** Η νέα εγγραφή αποθηκεύεται στη βάση δεδομένων, συνδένοντας τα δεδομένα του χρήστη με τα αντίστοιχα δεδομένα εισροών, μεταφορών, αναλύσεων και δειγμάτων.

## Λειτουργίες Διαχείρισης των Χρηστών

Μετά τη δημιουργία μιας νέας οντότητας χρήστη, ο χρήστης μπορεί να χρησιμοποιήσει τις διαδρομές του API για να εκτελέσει διάφορες λειτουργίες, όπως η ενημέρωση ή η διαγραφή της οντότητας. Οι λειτουργίες που μπορούν να εκτελεστούν περιλαμβάνουν:

- **Ενημέρωση Εγγραφών:** Η ενημέρωση των πληροφοριών ενός χρήστη, όπως το όνομα χρήστη, το πλήρες όνομα και τον κωδικό πρόσβασης.
- **Διαγραφή Εγγραφών:** Η διαγραφή ενός χρήστη από το σύστημα. Σημειώνεται ότι αυτές οι λειτουργίες είναι περιορισμένες μόνο για χρήστες τύπου Admin.
- **Αναζήτηση και Ανάκτηση Δεδομένων:** Η αναζήτηση και η ανάκτηση των δεδομένων ενός χρήστη είτε μέσω μοναδικών αναγνωριστικών είτε μέσω κριτηρίων αντιστοίχισης.

## Μηχανισμός Υπογραφής (Signature Mechanism)

Όταν ένας χρήστης, ανεξαρτήτως κλάσης, εκτελεί μια λειτουργία που αλλάζει την κατάσταση της βάσης δεδομένων (δημιουργία, ενημέρωση, διαγραφή), του ζητείται να εισάγει τον κωδικό πρόσβασής του για να ολοκληρώσει τη λειτουργία <sup>28</sup>. Αυτό προσθέτει ένα επιπλέον επίπεδο ασφαλείας και χρησιμοποιείται για την καταγραφή του χρήστη που επέβλεψε τη λειτουργία στη βάση δεδομένων. Για παράδειγμα, κατά τη δημιουργία μιας υποδοχής (Reception) ή μιας κάρτας ανάλυσης (Analysis Card), το πεδίο `overseered_by` συμπληρώνεται με το αναγνωριστικό του χρήστη που εκτέλεσε τη λειτουργία. Σε περίπτωση ενημέρωσης ή διαγραφής δεδομένων, αυτό μπορεί να χρησιμοποιηθεί για ένα μηχανισμό καταγραφής επεξεργασιών (editing log) που μπορεί να προστεθεί στο μέλλον.

Η υλοποίηση του συστήματος διαχείρισης χρηστών εξασφαλίζει την αποδοτική και ασφαλή διαχείριση των δεδομένων, επιτρέποντας στους διαχειριστές να καταχωρούν, να ενημερώνουν και να διαγράφουν χρήστες, καθώς και να διαχειρίζονται τα δικαιώματα πρόσβασης και τις λειτουργίες που είναι διαθέσιμες σε κάθε χρήστη.

---

## 3.5 Υλοποίηση Λειτουργιών Συστήματος

### 3.5.1 Event Sourcing: Περιγραφή

Το υπάρχον σύστημα, όπως έχει παρουσιαστεί μέχρι τώρα, παρουσιάζει ορισμένες αδυναμίες όσον αφορά την υποστήριξη αναδρομικών ενημερώσεων. Συγκεκριμένα, η διαγραφή και η ενημέρωση δεδομένων που έχουν καταχωρηθεί στο παρελθόν μπορούν να αφήσουν τη βάση δεδομένων σε μια λανθασμένη κατάσταση, να οδηγήσουν σε ασυνέπειες και απώλεια της ακεραιότητας των δεδομένων. Αυτή η αδυναμία προκύπτει από το γεγονός ότι οι εγγραφές στη βάση δεδομένων δεν καταγράφουν την ιστορικότητα των αλλαγών, αλλά μόνο την τρέχουσα κατάσταση των δεδομένων. Ακολουθούν μερικά σενάρια που αντικατοπτρίζουν τα προαναφερθέντα.

**Σενάρια που οδηγούν σε λανθάνουσα κατάσταση:**

#### **Παράδειγμα 1**

- Ο χρήστης εκτελεί μια λειτουργία μεταφοράς, μεταφέροντας 5 τόνους από την δεξαμενή Α στην δεξαμενή Β. Μετά από αυτό, ο χρήστης εκτελεί κάποιες παραλαβές που αποθηκεύονται στην δεξαμενή Β. Ακολουθούν και άλλες ενέργειες που περιλαμβάνουν τις δεξαμενές Α και Β. Σε κάποιο σημείο, ο χρήστης συνειδητοποιεί ότι η πρώτη μεταφορά που αναφέρθηκε καταχωρήθηκε λανθασμένα ως μεταφορά από την δεξαμενή Α στην δεξαμενή Β, ενώ στην πραγματικότητα ήταν μεταφορά από την δεξαμενή Α στην δεξαμενή C. Αυτό σημαίνει ότι η βάση δεδομένων βρίσκεται σε μια λανθασμένη κατάσταση πραγματικότητας. Η αλλαγή αυτής της εγγραφής στη βάση δεδομένων χωρίς να υπάρχει ιστορικότητα των αλλαγών θα μπορούσε να οδηγήσει σε ασυνέπειες και απώλεια της ακεραιότητας των δεδομένων.

#### **Παράδειγμα 2**

- Ο χρήστης παραλαμβάνει λάδι στην αποθήκη και το τοποθετεί στη δεξαμενή Α χωρίς να το καταγράψει στο σύστημα. Έπειτα έρχεται ένα φορτηγό βυτιοφόρο και ο χρήστης πραγματοποιεί μία φόρτωση από την δεξαμενή Α την οποία και καταγράφει στο σύστημα. Μετά το πέρας της φορτώσεως θυμάται ότι η τελευταία παραλαβή δεν περάστηκε στο σύστημα και την καταγράφει εκ των υστέρων, έστω με ημερομηνία προγενέστερη της φορτώσεως. Η βάση δεδομένων βρίσκεται σε μια λανθασμένη κατάσταση πραγματικότητας.

*Ο τρέχων σχεδιασμός του συστήματος δεν μπορεί να αντιμετωπίσει τα παραπάνω προβλήματα χωρίς να επιφέρει επιπλέον επιπλοκές και ασυνέπειες.*

### **Θεωρία του Event Sourcing**

Το event sourcing είναι μια προσέγγιση στον σχεδιασμό λογισμικού που περιλαμβάνει την αποθήκευση της ιστορίας των αλλαγών των δεδομένων ως μια ακολουθία γεγονότων (events). Αντί να αποθηκεύεται μόνο η τρέχουσα κατάσταση των δεδομένων, κάθε αλλαγή που γίνεται στα δεδομένα αποθηκεύεται ως ένα ξεχωριστό γεγονός.

Σύμφωνα με τον Martin Fowler <sup>29</sup>, το event sourcing είναι μια τεχνική που προτείνει την καταγραφή κάθε αλλαγής που γίνεται σε μια εφαρμογή ως ένα γεγονός (event). Αυτά τα γεγονότα καταγράφονται σε ένα αποθετήριο (event store) και μπορούν να αναπαραχθούν για να αναδημιουργήσουν την κατάσταση του συστήματος σε οποιοδήποτε χρονικό σημείο.

## Πλεονεκτήματα του Event Sourcing

- 1. Ιστορικότητα και Αναδρομικές Ενημερώσεις:** Το event sourcing επιτρέπει την παρακολούθηση κάθε αλλαγής που γίνεται στα δεδομένα, παρέχοντας έτσι πλήρες ιστορικό των ενεργειών. Αυτό καθιστά δυνατή την αναδρομική ενημέρωση των δεδομένων χωρίς να επηρεάζεται η συνοχή της βάσης δεδομένων.
- 2. Αναδημιουργία Καταστάσεων:** Δεδομένου ότι κάθε αλλαγή καταγράφεται ως γεγονός, η αναδημιουργία της κατάστασης του συστήματος σε οποιοδήποτε χρονικό σημείο είναι δυνατή. Αυτό μπορεί να είναι εξαιρετικά χρήσιμο για σκοπούς debugging και auditing.
- 3. Αναίρεση και Επανάληψη:** Η δυνατότητα αναίρεσης των αλλαγών γίνεται πιο εφικτή, καθώς κάθε γεγονός μπορεί να αντιστραφεί για να ανακληθεί μια συγκεκριμένη αλλαγή.

## Εφαρμογή του Event Sourcing στο Υπάρχον Σύστημα

Η υιοθέτηση του event sourcing στο υπάρχον σύστημα διαχείρισης αποθήκευσης λαδιού θα μπορούσε να επιλύσει τις προαναφερθείσες αδυναμίες. Κάθε αλλαγή που γίνεται στα δεδομένα (όπως οι μεταφορές, οι παραλαβές, και οι φορτώσεις) θα καταγράφεται ως ένα γεγονός. Αυτά τα γεγονότα θα αποθηκεύονται σε ένα αποθετήριο γεγονότων (event store) και θα μπορούν να αναπαραχθούν για να αναδημιουργήσουν την κατάσταση του συστήματος σε οποιοδήποτε χρονικό σημείο.

### Παράδειγμα Εφαρμογής:

Σε περίπτωση που ο χρήστης εκτελεί μια μεταφορά από την δεξαμενή Α στην δεξαμενή Β και αργότερα διαπιστώνει ότι έγινε λάθος, μπορεί να εκτελέσει ένα νέο γεγονός που να αντιστρέφει την αρχική μεταφορά και να καταγράφει τη σωστή μεταφορά από την δεξαμενή Α στην δεξαμενή C. Με αυτόν τον τρόπο, η βάση δεδομένων θα διατηρεί πλήρες ιστορικό των αλλαγών και θα είναι σε θέση να αναδημιουργήσει την ακριβή κατάσταση του συστήματος οποιαδήποτε στιγμή.

### Συνοπτικά

Η θεωρία του event sourcing αποτελεί ένα σημαντικό κομμάτι του σχεδιασμού λογισμικού που στοχεύει στη βελτίωση της διαχείρισης των δεδομένων και της ιστορικότητας των ενεργειών μέσα σε ένα σύστημα. Η υιοθέτηση μηχανισμού event sourcing θα βελτιώσει την αξιοπιστία και την ακρίβεια του συστήματος, παρέχοντας παράλληλα μεγαλύτερη ευελιξία στη διαχείριση των δεδομένων και στη διόρθωση λαθών.

---

## 3.5.2 Event Sourcing: Υλοποίηση

Η υλοποίηση του μηχανισμού event sourcing στο σύστημα αποθήκευσης λαδιού αποτελεί κρίσιμο βήμα για την αντιμετώπιση των αδυναμιών που εντοπίστηκαν στο υπάρχον σύστημα. Ο στόχος είναι η βελτίωση της διαχείρισης των δεδομένων και της ιστορικότητας των ενεργειών, καθώς και η εξασφάλιση της ακρίβειας και της ακεραιότητας των δεδομένων.

## Ανάλυση Κατάστασης:

Αναλύοντας την κατάσταση στο σύστημα, έγινε κατανοητό ότι ο πίνακας **Inflow\_Tank** είναι ο κύριος πίνακας που επηρεάζεται από τις λειτουργίες δημιουργίας, ενημέρωσης και διαγραφής. Οι χρήστες εκτελούν τις παρακάτω ενέργειες που μπορούν να προκαλέσουν αλλαγές στον πίνακα **Inflow\_Tank** και να οδηγήσουν σε ασυνέπειες:

- Δημιουργία / Ενημέρωση / Διαγραφή - Παραλαβής (inflow).
- Δημιουργία / Ενημέρωση / Διαγραφή - Φόρτωσης (outflow).
- Δημιουργία / Ενημέρωση / Διαγραφή - Μεταφοράς (transfer).

## Υλοποίηση:

### Δημιουργία Νέων Πινάκων

Για την υποστήριξη του μηχανισμού event sourcing, δημιουργήθηκαν δύο νέοι πίνακες στη βάση δεδομένων:

- **EventLog**: Χρησιμοποιείται για την καταγραφή των γεγονότων που τροποποιούν τον πίνακα **Inflow\_Tank**.
- **InflowTankSnapshot**: Χρησιμοποιείται για την αποθήκευση snapshots/καταστάσεων του πίνακα **Inflow\_Tank** σε καθορισμένα διαστήματα (κάθε N γεγονότα).

### Υπηρεσίες (Services)

Δημιουργήθηκαν οι κατάλληλες υπηρεσίες για τους νέους πίνακες:

- **EventLogService**: Χειρίζεται τις λειτουργίες CRUD για τον πίνακα **EventLog**.
- **SnapshotService**: Χειρίζεται τις λειτουργίες CRUD για τον πίνακα **InflowTankSnapshot**.

### Μονάδα EventHandler

Μια μονάδα ελέγχου (controller) με την ονομασία `EventHandler` δημιουργήθηκε και περιλαμβάνει τις παρακάτω λειτουργίες:

### Λειτουργίες

- `createEvent(actionType, actionId, actionDate, actionData, t /* transaction */) :` Δημιουργεί ένα νέο γεγονός που αντιστοιχεί σε μια επικείμενη εισαγωγή εγγραφής στον πίνακα **EventLog**. Οι τύποι των γεγονότων (actionTypes) είναι:
  - `'inflow'`
  - `'outflow'`
  - `'transfer'`
  - `'tankQuantityAdjustment'` // sets the tanks *given\_quantity* by adjusting all corresponding **Inflow\_Tank** table records
  - `'checkpoint'` // takes a snapshot of **Inflow\_Tank** table and stores it in **InflowTankSnapshot** table
- `applyEvent(ev, t /* transaction */) :` Χρησιμοποιεί τις λειτουργίες του **InflowTankService** για να εκτελέσει τις κατάλληλες λειτουργίες στον πίνακα **Inflow\_Tank** με βάση το γεγονός που παρέχεται ως παράμετρος, και αναλαμβάνει την αποθήκευση του εφαρμοσμένου γεγονότος στον πίνακα **EventLog**.

- `replayEvents(orderedEventList, t /* transaction */) :` Δέχεται μια λίστα γεγονότων (ταξινομημένα κατά ημερομηνία) και καλεί τη λειτουργία `applyEvent()` για κάθε γεγονός, ώστε να εφαρμόσει τις λειτουργίες των γεγονότων.
- `handleEvent(ev, t /* transaction */) :` Περιέχει τη βασική λογική πίσω από τον μηχανισμό χειρισμού γεγονότων. Είναι υπεύθυνη για την ρύθμιση του πίνακα `Inflow_Tank` στην επιθυμητή κατάσταση (φορτώνοντας snapshots από τον πίνακα `InflowTankSnapshot`), την εφαρμογή/επανεφαρμογή των γεγονότων (καλώντας τις `applyEvent()`, `replayEvents()`) και την εισαγωγή γεγονότων τύπου `'checkpoint'` όταν χρειάζεται.

### Βοηθητικές Λειτουργίες

- `isRealityBreaking(actionType, actionData, /* transaction */) :` Χρησιμοποιείται στις λειτουργίες ενημέρωσης για να καθορίσει αν η επικείμενη ενημέρωση θα οδηγήσει τη βάση δεδομένων σε λανθασμένη κατάσταση ή όχι. Παραδείγματα πεδίων που οδηγούν σε λανθασμένη κατάσταση κατά την ενημέρωση είναι:
  - **Inflow:** ημερομηνία, δεξαμενές, ποσότητα
  - **Outflow:** ημερομηνία, δεξαμενές, ποσότητα
  - **Transfer:** ημερομηνία, δεξαμενές (από-προς), ποσότητα
  - **TankQuantityAdjustment:** ημερομηνία, δεξαμενή, ποσότητα

### Χρήση του EventHandler

#### Παράδειγμα χρήσης: update Transfer

```

1 // file: TransferRoutes.mjs
2 import express from 'express'
3 import * as TransferController from '../controllers/TransferController.mjs'
4
5 const transfersRouter = express.Router()
6
7 // Updates an existing transfer
8 transfersRouter.put('/ism/transfers/:id(\\d+)',
  TransferController.editTransfer)
9
10 //-----
11
12 // file: TransferController.mjs
13 import sequelize from '../config/DBConfig.mjs'
14 import TransferService from '../services/TransferService.mjs'
15 import InflowTankService from '../services/InflowTankService.mjs'
16 import EventHandler from './EventHandler.mjs'
17
18 // Edit specified transfer
19 export async function editTransfer(req, res /* , next */) {
20   const trId = parseInt(req.params.id, 10)
21   const data = req.body
22
23   try {
24     const updatedTransfer = await TransferService.updateTransfer({
25       transferData: { tr_id: trId, ...data.transferData },
26     })
27
28     // Check if tanks, quantity, date are modified

```

```

29     if (EventHandler.isRealityBreaking('transfer', data)) {
30         //
31
32         let updatedEvent = await EventHandler.getAndDeleteEvent(
33             'transfer',
34             updatedReception.io_id,
35             t
36         )
37
38         updatedEvent = EventHandler.modifyEvent(updatedEvent, data)
39
40         await EventHandler.handleEvent(updatedEvent, t)
41     }
42
43     if (updatedTransfer) {
44         res.status(200).json(updatedTransfer)
45     } else {
46         res.status(404).json({ error_msg: `Transfer with ID ${trId} not
found.` })
47     }
48     } catch (error) {
49         console.error(`Error in editTransfer: ${error.message}`)
50         res.status(500).json({ error_msg: error.message })
51     }
52 }

```

Με την υλοποίηση του event sourcing, το σύστημα αποθήκευσης λαδιού διασφαλίζει την ακεραιότητα των δεδομένων και την δυνατότητα αναπαραγωγής της κατάστασης του συστήματος σε οποιοδήποτε χρονικό σημείο. Η καταγραφή κάθε αλλαγής ως γεγονός και η αποθήκευση snapshots του πίνακα Inflow\_Tank παρέχει τη δυνατότητα αναδρομικής ενημέρωσης και αποκατάστασης σφαλμάτων χωρίς να επηρεάζεται η συνοχή της βάσης δεδομένων.

### 3.5.3 Προγραμματισμένες Εργασίες Συστήματος

Το σύστημα περιλαμβάνει μια σειρά από προγραμματισμένες εργασίες (scheduled jobs) για τη διασφάλιση της ομαλής λειτουργίας και της συντήρησης του συστήματος. Η μόνη προγραμματισμένη εργασία που έχει υλοποιηθεί μέχρι τώρα είναι η **TokenBlacklistJob**.

#### TokenBlacklistJob

Αυτή η εργασία είναι προγραμματισμένη να εκτελείται κάθε 16 ώρες από προεπιλογή και έχει ως σκοπό τη συντήρηση του πίνακα **TokenBlacklist**. Συγκεκριμένα, η εργασία αυτή διαγράφει τα ληγμένα tokens που έχουν τοποθετηθεί στον πίνακα, με στόχο τη μείωση του μεγέθους του πίνακα και τη βελτίωση της απόδοσης του συστήματος. Η εργασία χρησιμοποιεί το **TokenBlacklistService** module για να πραγματοποιήσει τις απαραίτητες λειτουργίες CRUD στον πίνακα TokenBlacklist.



## Προτάσεις για Άλλες Τακτικές Εργασίες

Στα πλαίσια επιπλέον ανάπτυξης του συστήματος, εξετάζονται οι εξής προγραμματισμένες εργασίες για μελλοντική υλοποίηση:

### 1. Database Backup Job:

- Προγραμματισμένη εργασία για τη δημιουργία αντίγραφων ασφαλείας της βάσης δεδομένων σε τακτά χρονικά διαστήματα (π.χ., καθημερινά ή εβδομαδιαία).
- Στόχος είναι η διασφάλιση της ακεραιότητας των δεδομένων και η δυνατότητα αποκατάστασης σε περίπτωση απώλειας δεδομένων.

### 2. Data Archiving Job:

- Εργασία για την αρχειοθέτηση παλαιών δεδομένων που δεν χρειάζονται άμεση πρόσβαση αλλά πρέπει να διατηρηθούν για ιστορικούς λόγους.
- Η αρχειοθέτηση θα βοηθήσει στη διαχείριση του μεγέθους της βάσης δεδομένων και στη βελτίωση της απόδοσης του συστήματος.

### 3. Event Log Cleanup Job:

- Προγραμματισμένη εργασία για τον καθαρισμό παλαιών εγγραφών από τον πίνακα EventLog.
- Στόχος είναι η διατήρηση της βάσης δεδομένων σε αποδοτική κατάσταση και η αποφυγή υπερβολικής συσσώρευσης δεδομένων.

### 4. Performance Monitoring Job:

- Εργασία για την παρακολούθηση της απόδοσης του συστήματος και την καταγραφή βασικών δεικτών απόδοσης (KPIs).
- Θα βοηθήσει στον εντοπισμό πιθανών προβλημάτων και στη λήψη προληπτικών μέτρων για τη βελτίωση της απόδοσης.

Η υλοποίηση αυτών των προγραμματισμένων εργασιών θα συμβάλει στη βελτίωση της λειτουργικότητας και της αποδοτικότητας του συστήματος, παρέχοντας παράλληλα αυξημένη ασφάλεια και αξιοπιστία.

# Κεφάλαιο 4: Συμπεράσματα και Μελλοντική Εργασία

## 4.1 Περίληψη του Έργου

Η παρούσα διπλωματική εργασία είχε ως στόχο την ανάπτυξη ενός συστήματος διαχείρισης αποθήκευσης για μια εταιρεία που δραστηριοποιείται στο εμπόριο ελαιολάδου. Αναλύθηκαν οι υπάρχουσες αδυναμίες του χειρόγραφου συστήματος καταγραφής, όπως η ασυνέπεια των δεδομένων και η δυσκολία στην ανάκτηση πληροφοριών. Στη συνέχεια, προτάθηκε και υλοποιήθηκε ένα ψηφιακό σύστημα βασισμένο σε σύγχρονες τεχνολογίες όπως το Node.js, το Express.js, το Sequelize ORM και το PostgreSQL.

Το σύστημα περιλαμβάνει τις βασικές λειτουργίες καταγραφής παραλαβών, φορτώσεων, μεταφορών και χημικών αναλύσεων, καθώς και τις βοηθητικές λειτουργίες καταγραφής δεξαμενών, πελατών/προμηθευτών και χρηστών του συστήματος. Επιπλέον, εισήχθη η θεωρία του event sourcing για την καταγραφή των αλλαγών στη βάση δεδομένων, βελτιώνοντας την αξιοπιστία και την ακρίβεια των δεδομένων.

## 4.2 Συνεισφορές της Μελέτης

Η εργασία αυτή παρέχει σημαντικές συνεισφορές στη θεωρία και την πρακτική της ανάπτυξης λογισμικού διαχείρισης αποθήκευσης. Οι κύριες συνεισφορές περιλαμβάνουν:

### 1. Ανάπτυξη Προηγμένου Συστήματος Διαχείρισης Αποθήκευσης:

- Παρουσιάστηκε ένα ολοκληρωμένο σύστημα διαχείρισης αποθήκευσης που καλύπτει τις ανάγκες μιας εταιρείας εμπορίας ελαιολάδου.
- Χρησιμοποιήθηκαν σύγχρονες τεχνολογίες και τεχνικές για την υλοποίηση του συστήματος.

### 2. Εισαγωγή του Event Sourcing:

- Εισήχθη η θεωρία του event sourcing για τη βελτίωση της ακρίβειας και της ακεραιότητας των δεδομένων.
- Αναπτύχθηκαν οι πίνακες EventLog και InflowTankSnapshot και οι αντίστοιχες υπηρεσίες για την καταγραφή των γεγονότων και των καταστάσεων των δεξαμενών.

### 3. Βελτίωση της Αξιοπιστίας των Δεδομένων:

- Η καταγραφή κάθε αλλαγής ως γεγονός επιτρέπει την ακριβή αναπαραγωγή της κατάστασης του συστήματος σε οποιοδήποτε χρονικό σημείο.
- Παρέχεται η δυνατότητα αναδρομικών ενημερώσεων και αναίρεσης αλλαγών χωρίς να επηρεάζεται η συνοχή της βάσης δεδομένων.

### 4. Διαχείριση Χρηστών και Ασφάλεια:

- Αναπτύχθηκε ένα σύστημα διαχείρισης χρηστών με δυνατότητα καθορισμού δικαιωμάτων πρόσβασης και μηχανισμό υπογραφής για την ασφαλή εκτέλεση των λειτουργιών.

## 4.3 Περιορισμοί

Παρά τις σημαντικές συνεισφορές της, η εργασία αυτή παρουσιάζει ορισμένους περιορισμούς. Οι κύριοι εξ'αυτών:

- **Απουσία Frontend Διεπαφής:** Το έργο επικεντρώθηκε αποκλειστικά στην ανάπτυξη της backend λογικής και δεν περιλαμβάνει την ανάπτυξη ενός frontend για την αλληλεπίδραση των χρηστών με το σύστημα.
- **Περιορισμένη Διασύνδεση με Εξωτερικά Συστήματα:** Η υλοποίηση δεν περιλαμβάνει διασύνδεση με εξωτερικά συστήματα ή υπηρεσίες, όπως συστήματα ERP ή άλλες επιχειρησιακές εφαρμογές.
- **Έλλειψη Αυτοματοποίησης σε Ορισμένες Λειτουργίες:** Ορισμένες λειτουργίες, όπως η επαλήθευση και η επικύρωση των δεδομένων, απαιτούν χειροκίνητη παρέμβαση από τους χρήστες του συστήματος.
- **Περιορισμοί στην Κλιμάκωση:** Το σύστημα, όπως υλοποιήθηκε, ενδέχεται να αντιμετωπίσει προκλήσεις κλιμάκωσης σε περιπτώσεις πολύ μεγάλου όγκου δεδομένων και ταυτόχρονων χρηστών.

## 4.4 Συστάσεις για Μελλοντική Εργασία

Με βάση τα ευρήματα και τους περιορισμούς της παρούσας εργασίας, προτείνονται οι εξής κατευθύνσεις για μελλοντική έρευνα και βελτιώσεις:

- **Ανάπτυξη Frontend Διεπαφής:**
  - Ανάπτυξη μιας frontend διεπαφής χρήστη που θα επιτρέπει στους χρήστες να αλληλεπιδρούν με το σύστημα με ευκολία και αποτελεσματικότητα.
  - Ενσωμάτωση μοντέρνων τεχνολογιών frontend, όπως React ή Angular, για την ανάπτυξη μιας δυναμικής και διαδραστικής εμπειρίας χρήστη.
- **Διασύνδεση με Εξωτερικά Συστήματα:**
  - Ανάπτυξη διασυνδέσεων με εξωτερικά συστήματα ERP και άλλες επιχειρησιακές εφαρμογές για την αυτοματοποίηση και τον συγχρονισμό των δεδομένων.
  - Χρήση APIs και web services για την ασφαλή και αποδοτική επικοινωνία μεταξύ των συστημάτων.
- **Αυτοματοποίηση Λειτουργιών Επαλήθευσης:**
  - Ανάπτυξη αυτοματοποιημένων μηχανισμών για την επαλήθευση και επικύρωση των δεδομένων κατά την καταχώρηση και την επεξεργασία τους.
  - Ενσωμάτωση εργαλείων machine learning για την ανίχνευση ανωμαλιών και σφαλμάτων στα δεδομένα.
- **Βελτιστοποίηση και Κλιμάκωση:**
  - Βελτιστοποίηση των λειτουργιών του συστήματος για τη διαχείριση μεγάλου όγκου δεδομένων και ταυτόχρονων χρηστών.
  - Χρήση τεχνικών κατανομής φορτίου (load balancing) και κατανεμημένων συστημάτων βάσεων δεδομένων για την επίτευξη υψηλής απόδοσης και συνέπειας.
- **Ανάπτυξη Λειτουργιών Αναφορών και Αναλύσεων:**

- Ενσωμάτωση λειτουργιών δημιουργίας αναφορών και ανάλυσης δεδομένων για την παροχή πολύτιμων πληροφοριών και insights στους χρήστες του συστήματος.
- Χρήση εργαλείων data visualization για την παρουσίαση των δεδομένων με γραφήματα και διαγράμματα.

Οι προτάσεις αυτές αποσκοπούν στη βελτίωση και την επέκταση του συστήματος διαχείρισης αποθήκευσης, παρέχοντας στους χρήστες αυξημένη λειτουργικότητα, ακρίβεια και αποτελεσματικότητα στη διαχείριση των αποθεμάτων και των επιχειρησιακών διαδικασιών.

# Αναφορές:

---

1. Το "τετράδιο των δεξαμενών" είναι το έντυπο στο οποίο αναγράφονται οι κύριες παραλαβές και οι ποσότητες που υπάρχουν σε κάθε δεξαμενή. [↵](#) [↵](#)
2. Παράδειγμα: Η παραλαβή με αναγνωριστικό 'id', παραλήφθηκε τότε από τον Χ και τοποθετήθηκε στην Α δεξαμενή. Κατόπιν μία ποσότητα x μεταφέρθηκε στην Β δεξαμενή και μία ποσότητα y φορτώθηκε από την Β δεξαμενή στο Φ βυτίο και έφυγε από την αποθήκη. Η τωρινή κατάσταση της παραλαβής είναι: Μία x ποσότητα στην Α δεξαμενή, μία y ποσότητα στην Β δεξαμενή, μία z ποσότητα έφυγε με το φορτίο Φ. [↵](#)
3. βλέπε ενότητα [3.3.4 Oil Chemical Analyses](#). [↵](#)
4. βλέπε ενότητα [3.5.2 Event Sourcing](#): Υλοποίηση. [↵](#) [↵](#)
5. βλέπε Λειτουργίες: [Παραλαβή](#) και [Φόρτωση](#). [↵](#)
6. POST `/ism/users/login` [↵](#)
7. **Visual Studio Code**: Microsoft. Visual Studio Code. [Available online](#) [↵](#)
8. **Yarn**: Yarn - Fast, reliable, and secure dependency management. [Available online](#) [↵](#)
9. **pgAdmin 4**: pgAdmin - PostgreSQL Tools. [Available online](#) [↵](#)
10. **Node.js**: Node.js. [Available online](#) [↵](#)
11. **Express.js**: Express - Node.js web application framework. [Available online](#) [↵](#)
12. **Sequelize ORM**: Sequelize - Easy-to-use multi SQL dialect ORM for Node.js. [Available online](#) [↵](#)
13. **sequelize-auto**: sequelize-auto - Automatically generate models for SequelizeJS via the command line. [Available online](#) [↵](#)
14. **PostgreSQL**: PostgreSQL - The world's most advanced open source database. [Available online](#) [↵](#)
15. **PostgreSQL Documentation**: PostgreSQL 13 Documentation. [Available online](#) [↵](#)
16. **ESLint**: ESLint - Find and fix problems in your JavaScript code. [Available online](#) [↵](#)
17. **eslint-config-prettier**: eslint-config-prettier - Turns off all rules that are unnecessary or might conflict with Prettier. [Available online](#) [↵](#)
18. **JSHint**: JSHint - A Static Code Analysis Tool for JavaScript. [Available online](#) [↵](#)
19. **Nodemon**: Nodemon - Automatically restart your node application. [Available online](#) [↵](#)
20. **Prettier**: Prettier - An opinionated code formatter. [Available online](#) [↵](#)
21. **bcrypt**: bcrypt - A library to help you hash passwords. [Available online](#) [↵](#)
22. **cors**: cors - Node.js CORS middleware. [Available online](#) [↵](#)
23. **dotenv**: dotenv - Loads environment variables from .env file. [Available online](#) [↵](#)
24. **express-validator**: express-validator - Express middleware for validation. [Available online](#) [↵](#)
25. **jsonwebtoken**: jsonwebtoken - An implementation of JSON Web Tokens. [Available online](#) [↵](#)
26. **pg**: pg - Non-blocking PostgreSQL client for Node.js. [Available online](#) [↵](#)
27. **pg-hstore**: pg-hstore - A node.js hstore serializer for use with node-postgres. [Available online](#) [↵](#)
28. Το σύστημα σχεδιάστηκε με γνώμονα το συγκεκριμένο concept, η υλοποίηση ωστόσο είναι ευθύνη του frontend. [↵](#)
29. [Martin Fowler: Event Sourcing](#) [↵](#)