Έγγραφο απαιτήσεων λογισμικού (SRS)

Multe-Pass

1. Εισαγωγή

1.1 Εισαγωγή: σκοπός του λογισμικού

Σκοπός του λογισμικού είναι η διαχείριση της διαλειτουργικότητας των σταθμών διοδίων σε αυτοκινητοδρόμους με διαφορετικά συστήματα αυτόματης διέλευσης. Τα διαφορετικά συστήματα που υπάρχουν τη δεδομένη χρονική στιγμή που αναπτύσσεται η πρώτη έκδοση του λογισμικού (21/11/2021) είναι των οδών: **aodos.gr, gefyra.gr, egnatia.eu, kentrikiodos.gr, moreas.com.gr, neaodos.gr, olympiaodos.gr.** Μέσω του λογισμικού καταγράφονται οι διελεύσεις από τους σταθμούς διοδίων των διαφόρων χειριστών, ετοιμάζονται τα έγγραφα διακανονισμού μεταξύ 2 χειριστών, γίνονται όλοι οι έλεγχοι για την ικανοποίηση αυτών των λειτουργιών και παίρνονται μέτρα για την ορθή λειτουργία του συστήματος καθώς και για την ύπαρξη ασφάλειας.

1.2 Διεπαφές (interfaces)

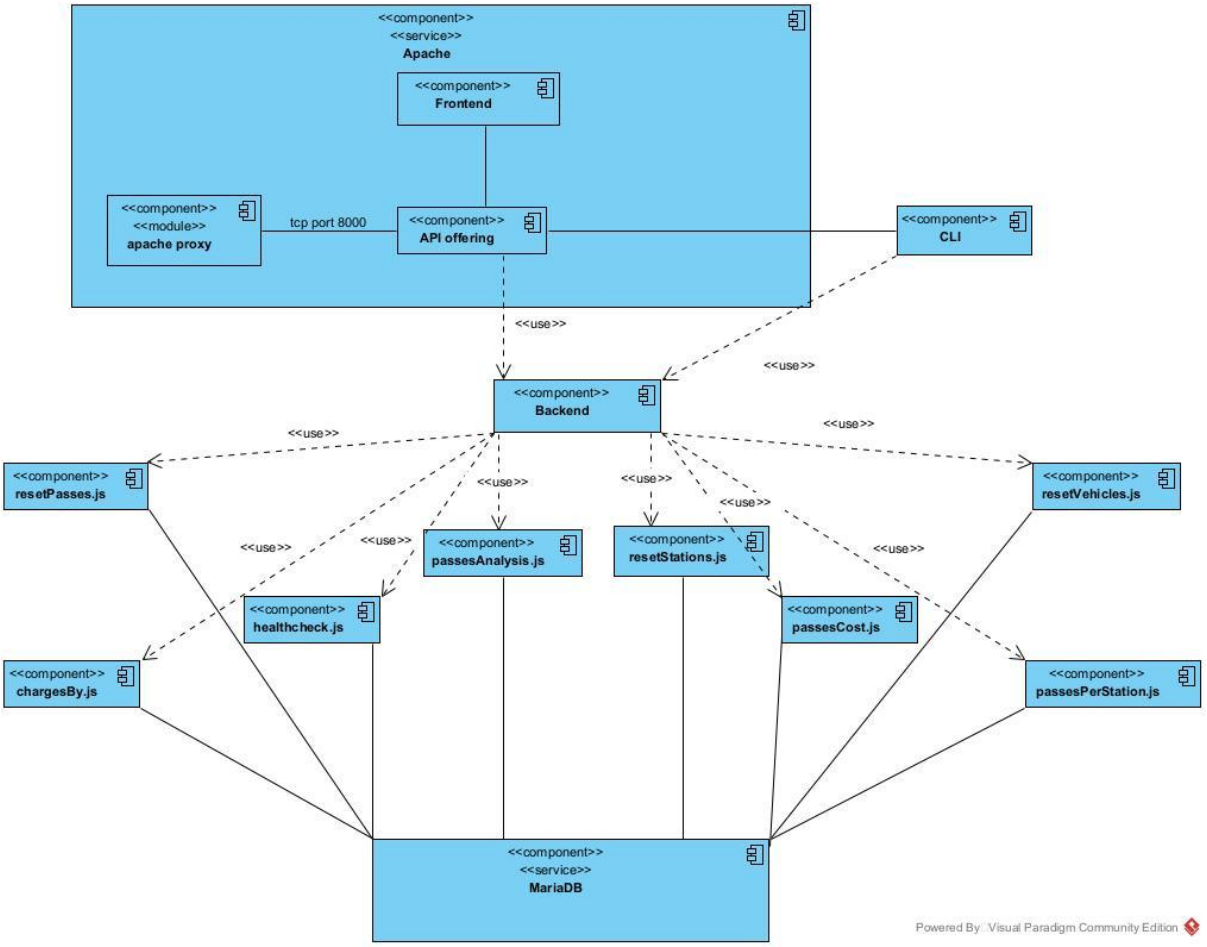
1.2.1 Διεπαφές με εξωτερικά συστήματα

MariaDB DBMS: Η διαχείριση των δεδομένων από την εφαρμογή μας γίνεται μέσω της τεχνολογίας MariaDB.

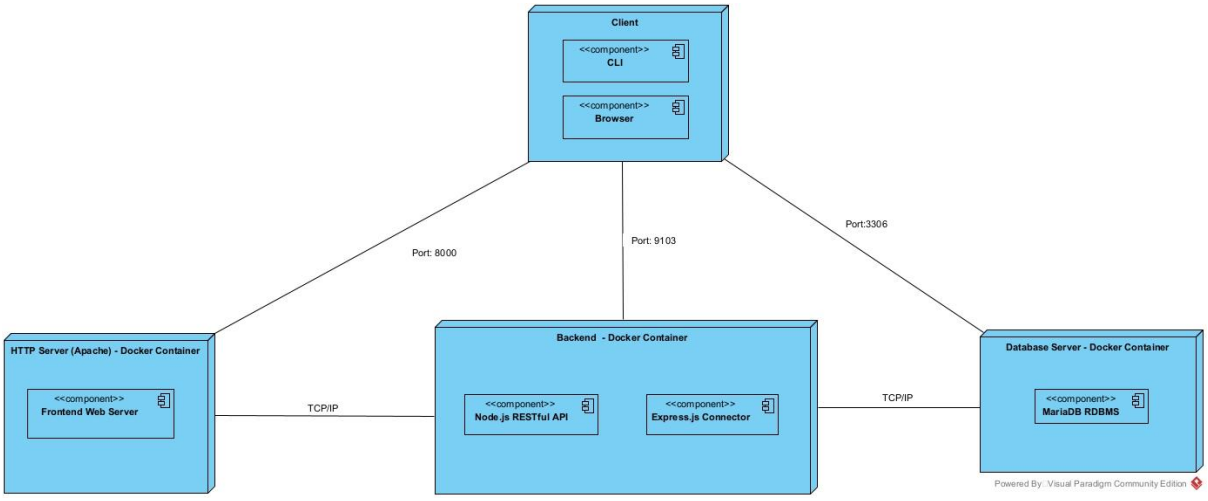
Apache HTTP WebServer: Η Web εφαρμογή Frontend γίνεται hosted από server Apache.

Η εφαρμογή έχει γίνει πλήρως “Dockerized” και μπορεί να τρέξει σαν Container Stack είτε locally από μηχάνημα χρήστη ή να γίνει hosted από εξωτερική υπηρεσία.

Το παρακάτω διάγραμμα τύπου UML Component δείχνει, με απλοποιημένη μορφή, την βασική ιδέα επικοινωνίας των συστημάτων μεταξύ τους καθώς και των Endpoints του RESTful API.



Το παρακάτω διάγραμμα τύπου UML Deployment δείχνει τις σχέσεις αλληλεπίδρασης μεταξύ του hardware(server, μηχανήματα client κ.λ.π.) και του λογισμικού Multe-pass.



1.2.2 Διεπαφές με το χρήστη

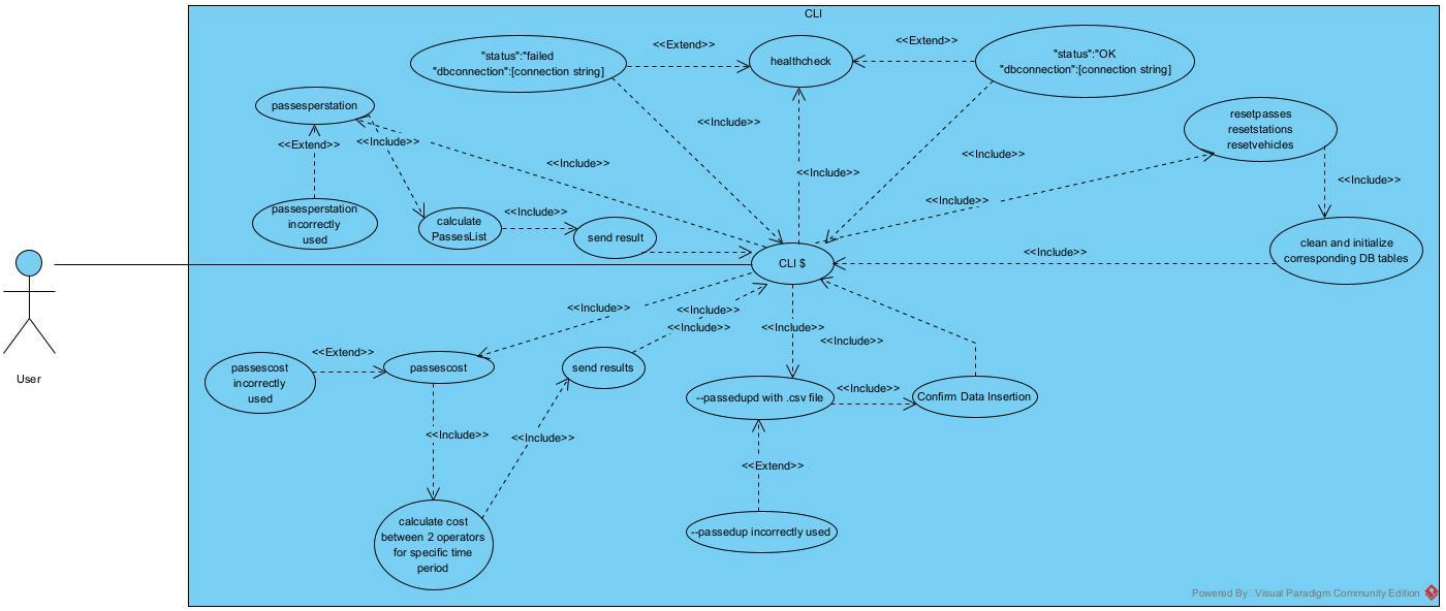
Η επαφή με τον χρήστη θα γίνεται μέσω ενός Command Line Interface(CLI) και μέσω μιας εφαρμογής Frontend. Η έξοδος των δεδομένων στην εφαρμογή Frontend, καθώς και στην διεπαφή CLI γίνεται μέσω ενός RESTful API υλοποιημένο σε JavaScript (node.JS).

RESTful API: Το API έχει περιορισμένη πρόσβαση στην βάση δεδομένων ώστε όταν δέχεται τα διάφορα HTTP requests (π.χ. get ή post) να μπορεί να εκτελέσει την ενέργεια που του ζητήθηκε. Η πρόσβαση στην βάση γίνεται μέσω συστήματος backend. Οι 2 μορφότυποι που υποστηρίζει το API είναι οι JSON και CSV. Προεπιλογή είναι ο μορφότυπος δεδομένων JSON.

Command Line Interface: Το CLI αποτελεί την βασική επαφή των χρηστών με την εφαρμογή. Μέσω αυτής γίνονται κλήσεις για την εκτέλεση των βασικών λειτουργιών της εφαρμογής όπως είναι η ανάκτηση δεδομένων. Επίσης είναι δυνατή η εκτέλεση κλήσεων για την οποιαδήποτε διαχειριστική ανάγκη της εφαρμογής. Είναι η μόνη διεπαφή που υποστηρίζει την εισαγωγή δεδομένων στην βάση, συγκεκριμένα την εισαγωγή δεδομένων στον πίνακα που είναι αποθηκευμένες οι διελεύσεις των οχημάτων. Υποστηρίζει, σαν το API, τους μορφότυπους JSON και CSV. Η υλοποίηση του έγινε σε γλώσσα Python.

Front-End Application: Η διεπαφή Frontend είναι μία δεύτερη επαφή των χρηστών με την εφαρμογή. Παρέχει στον χρήστη την δυνατότητα εκτέλεσης όλων των λειτουργιών που παρέχει η υπηρεσία, με την εξαίρεση την τοποθέτηση δεδομένων στην βάση, η οποία είναι δυνατή μόνο στο CLI. Εκτός από τις απαραίτητες λειτουργίες που εκτελεί, παρουσιάζει επίσης στατιστικά στοιχεία σε μορφή γραφημάτων πίτας ή γραφημάτων ράβδων. Η υλοποίηση της εφαρμογής έγινε με χρήση των τεχνολογιών HTTP/CSS/JS και PHP. Χρησιμοποιήθηκαν βιβλιοθήκες του Bootstrap 5.

Στο παρακάτω διάγραμμα Use Case της UML φαίνονται οι κύριες λειτουργίες/κινήσεις που μπορεί να ακολουθήσει ο χρήστης μέσω της διεπαφής του CLI. Το αντίστοιχο διάγραμμα για Front-end θα προστεθεί όταν υλοποιηθεί η συγκεκριμένη εφαρμογή.



Η κατάσταση CLI $ είναι όταν το CLI περιμένει είσοδο από τον χρήστη.

2. Αναφορές - πηγές πληροφοριών

Docker Software - https://www.docker.com/

3. Προδιαγραφές απαιτήσεων λογισμικού

3.1 Περιπτώσεις χρήσης

3.1.1 ΠΕΡΙΠΤΩΣΗ ΧΡΗΣΗΣ 1: ΔΙΕΛΕΥΣΕΙΣ (ΟΔΗΓΩΝ ΑΠΟ ΣΤΑΘΜΟΥΣ ΔΙΟΔΙΩΝ)

3.1.1.1 Χρήστες (ρόλοι) που εμπλέκονται

Οι χρήστες που εμπλέκονται στις διελεύσεις είναι οι οδηγοί και το σύστημά μας. Επί της ουσίας ο οδηγός είναι αυτός που χρησιμοποιεί τον e-pass πομποδέκτη σε έναν συγκεκριμένο σταθμό διοδίων και η πληροφορία της διέλευσης που πραγματοποιεί αποθηκεύεται στο πληροφοριακό σύστημα.

3.1.1.2 Προϋποθέσεις εκτέλεσης

Σε κάθε περίπτωση διέλευσης το σύστημά μας ενεργοποιείται.

Στην περίπτωση που η διέλευση αφορά χειριστή οδού ίδιο με αυτόν που αντιστοιχεί στην e-pass κάρτα, απλώς πραγματοποιείται έλεγχος επαρκούς υπολοίπου, επιτυχής διέλευση και αποθήκευση των πληροφοριών αυτής (ημερομηνία, ώρα, χειριστής σταθμού, χειριστής e-pass). Στην άλλη περίπτωση, όπου δηλαδή οι χειριστές είναι διαφορετικοί, πέρα των προαναφερθέντων εκτελείται επιπλέον και αποθήκευση του κόστους που χρωστάει ο ένας χειριστής στον άλλον.

3.1.1.3 Περιβάλλον εκτέλεσης

Το περιβάλλον που εκτελείται η χρήση είναι η διεπαφή CLI και η έξοδος της χρήσης αποθηκεύεται στο DBMS.

3.1.1.4 Δεδομένα εισόδου, εξόδου και συνθήκες εγκυρότητας

Δεδομένα Εισόδου:

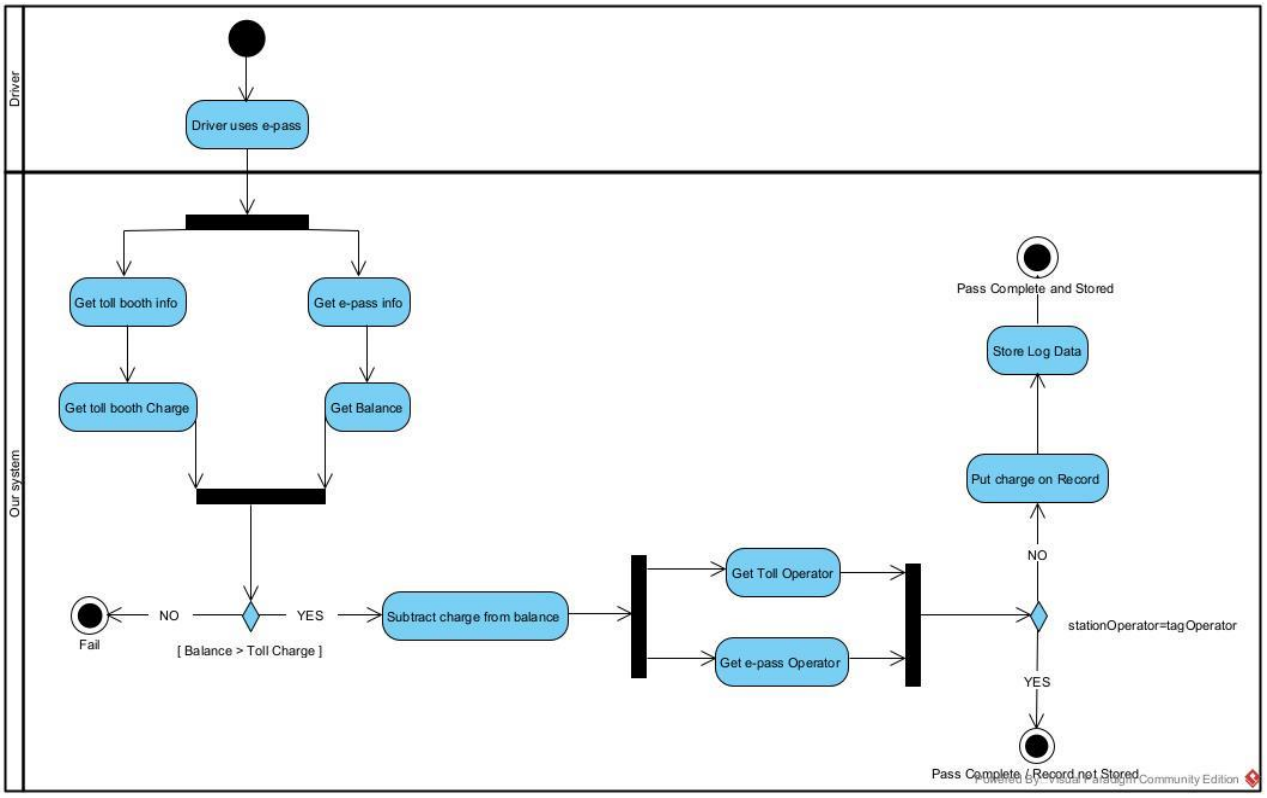
- Χρονική Στιγμή Διέλευσης
- TagID Πομποδέκτη
- ID Σταθμού Διοδίων
- Αντίτιμο Διέλευσης

Δεδομένα Εξόδου: - (αποθήκευση στη βάση δεδομένων του πληροφοριακού μας συστήματος των πληροφοριών εισόδου, σε περίπτωση εγκυρότητας αυτών και επαρκούς υπολοίπου διέλευσης)

Συνθήκες Εγκυρότητας:

- Τα παραπάνω δεδομένα να είναι μη κενά

3.1.1.5 Αλληλουχία ενεργειών - επιθυμητή συμπεριφορά



Βασική Ροή:

1. Ο οδηγός χρησιμοποιεί τον πομποδέκτη e-pass
2. Λαμβάνονται τα στοιχεία του σταθμού διοδίων και του πομποδέκτη, άμεσα μας ενδιαφέρουν το αντίτιμο διέλευσης και το ποσό υπολοίπου του πομποδέκτη
3. Συγκρίνονται τα ποσά αυτά (εναλλακτική ροή 1 αν το υπόλοιπο δεν επαρκεί)
4. Αφαιρείται το αντίτιμο διέλευσης από το υπόλοιπο πομποδέκτη
5. Λαμβάνονται πάλι στοιχεία του σταθμού και του πομποδέκτη, αυτή την φορά ο χειριστής του σταθμού διοδίων και ο πάροχος του πομποδέκτη
6. Συγκρίνονται τα στοιχεία αυτά (εναλλακτική ροή 2 εάν ο πάροχος είναι ο ίδιος με τον χειριστής)
7. Τοποθετείται η χρέωση στο μητρώο
8. Αποθηκεύεται η διέλευση στην βάση

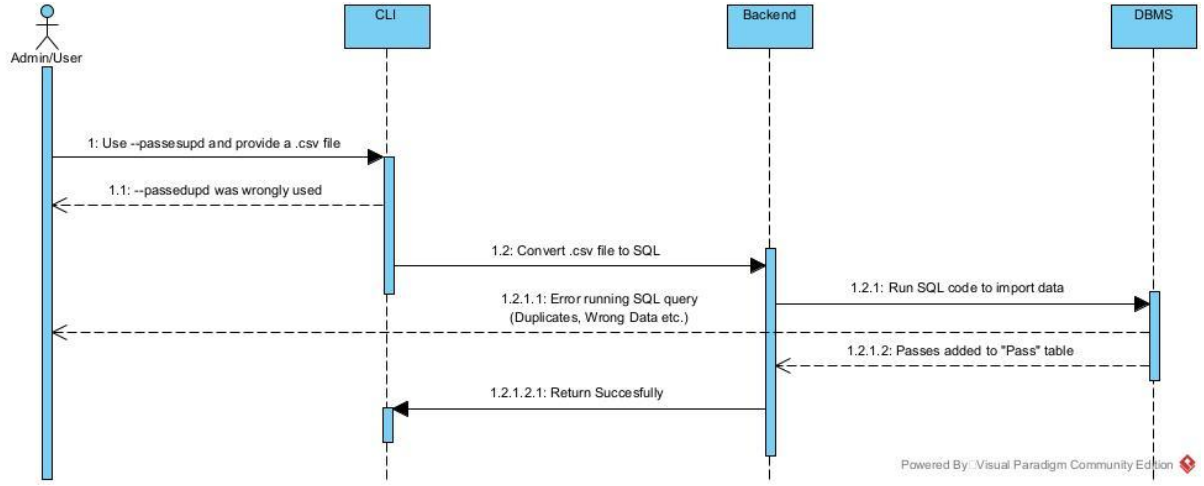
Εναλλακτική ροή 1:

- Η τιμή balance (το υπόλοιπο του πομποδέκτη του οδηγού) είναι μικρότερη από την τιμή Toll Charge (αντίτιμο διέλευσης) άρα απαγορεύεται η διέλευση.

Εναλλακτική ροή 2:

- Ο σταθμός διοδίων ανήκει στον ίδιο χειριστή με το TagID άρα δεν αποθηκεύεται το charge (κάποια οφειλή μεταξύ χειριστών οδών) - πραγματοποιείται η διέλευση.

3.1.1.7 Δεδομένα εξόδου



Η εγγραφή μιας λίστας διελεύσεων στον πίνακα “Pass” της βάσης δεδομένων μας.

3.1.1.8 Παρατηρήσεις

N/A

3.1.2 ΠΕΡΙΠΤΩΣΗ ΧΡΗΣΗΣ 2: ΔΗΜΙΟΥΡΓΙΑ ΙΣΟΖΥΓΙΟΥ ΜΕΤΑΞΥ 2 ΧΕΙΡΙΣΤΩΝ

3.1.2.1 Χρήστες (ρόλοι) που εμπλέκονται

Οι χρήστες που εμπλέκονται στις διελεύσεις είναι οι χειριστές οδών και το σύστημά μας.

3.1.2.2 Προϋποθέσεις εκτέλεσης

Η συγκεκριμένη λειτουργία του προγράμματός μας εκτελείται σε περίπτωση που κάποιος χειριστής οδού ζητήσει από το σύστημα να τον ενημερώσει σχετικά με το ποσό οφειλής ή χρέωσης που αφορά τον ίδιο και κάποιον άλλο (συγκεκριμένο) χειριστή.

3.1.2.3 Περιβάλλον εκτέλεσης

Το περιβάλλον που εκτελείται η χρήση είναι η διεπαφή CLI και η έξοδος της χρήσης αποστέλλεται στο χειριστή που ζητάει το ισοζύγιο και στον άλλο χειριστή που τον αφορά η επικείμενη συναλλαγή.

3.1.2.4 Δεδομένα εισόδου, εξόδου και συνθήκες εγκυρότητας

Δεδομένα Εισόδου:

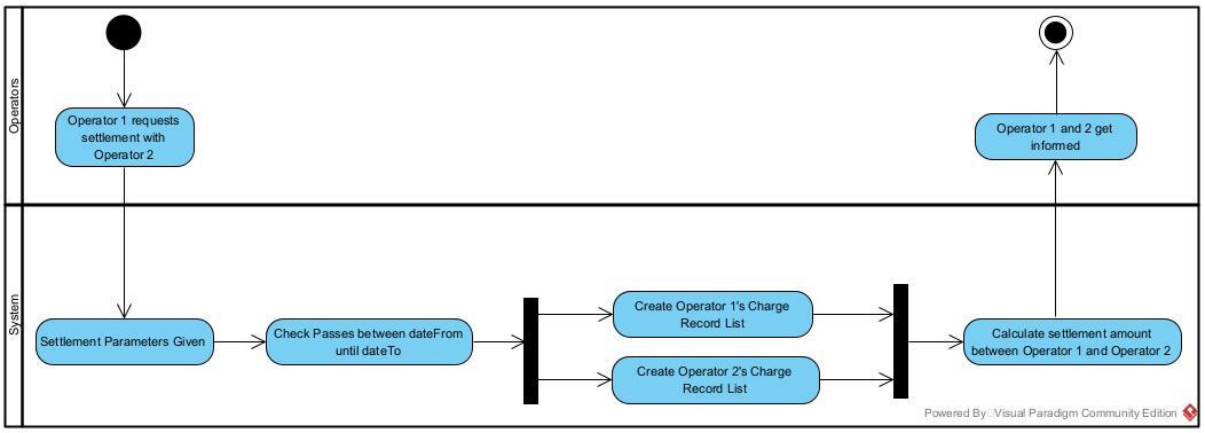
- operatorID1 χειριστή που ζητάει το ισοζύγιο
- operatorID2 χειριστή
- dateFrom & dateTo - οι ημερομηνίες μεταξύ των οποίων ζητείται ο υπολογισμός

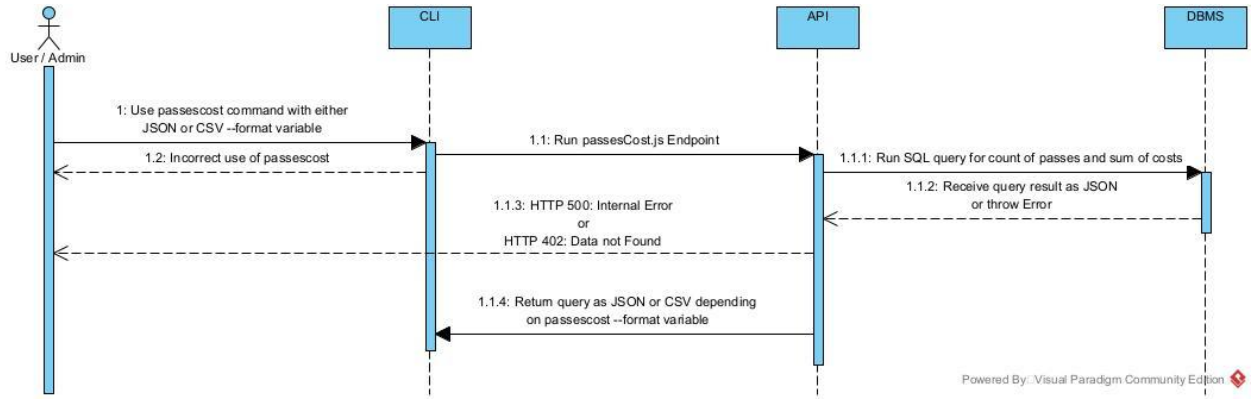
Δεδομένα Εξόδου: επιστρέφεται το ποσό που απαιτείται να δώσει ο ένας χειριστής και να λάβει ο άλλος ώστε μεταξύ των δύο ημερομηνιών να μην υπάρχουν οφειλές

Συνθήκες Εγκυρότητας:

- Τα παραπάνω δεδομένα να είναι μη κενά
- Οι χειριστές να είναι εγγεγραμμένοι στο σύστημά μας
- Οι ημερομηνίες να είναι έγκυρες

3.1.2.5 Αλληλουχία ενεργειών - επιθυμητή συμπεριφορά





Βασική Ροή:

1. Ο χειριστής 1 ζητάει από το σύστημα να ενημερωθεί για το ποσό του διακανονισμού με τον χειριστή 2
2. Το σύστημα λαμβάνει τα στοιχεία εισόδου
3. Πραγματοποιεί έλεγχο για όλες τις διελεύσεις που αφορούν τους συγκεκριμένους χειριστές οδών μεταξύ των ημερομηνιών που δόθηκαν
4. Δημιουργείται μία λίστα με όλες τις χρεώσεις του χειριστή 1 και αντίστοιχα άλλη μία για το χειριστή 2
5. Πραγματοποιείται υπολογισμός του ποσού οφειλής-χρέωσης του ενός προς τον άλλον
6. Ενημερώνονται οι δύο χειριστές για τους οποίους πραγματοποιούνται τα άνωθεν βήματα

Δεν έχουμε Εναλλακτικές Ροές

3.1.2.7 Δεδομένα εξόδου

Το sequence UML διάγραμμα παραπάνω, για το συγκεκριμένο Use Case, καλύπτει και τα δεδομένα εξόδου καθώς σαν έξοδο θεωρούμε το ποσό οφειλής/χρέωσης

Το ποσό οφειλής/χρέωσης μεταξύ των χειριστών. *Σε περίπτωση που γίνει η πληρωμή, αποθηκεύεται στον πίνακα “Transactions” οι πληροφορίες της συναλλαγής. Δηλαδή συμπληρώνονται τα δεδομένα (TransactionID, debitOperatorID, creditOperatorID, amount, dateFrom, dateTo) και αποθηκεύονται στην Βάση Δεδομένων.*

3.1.2.8 Παρατηρήσεις

N/A

3.2 Απαιτήσεις επιδόσεων

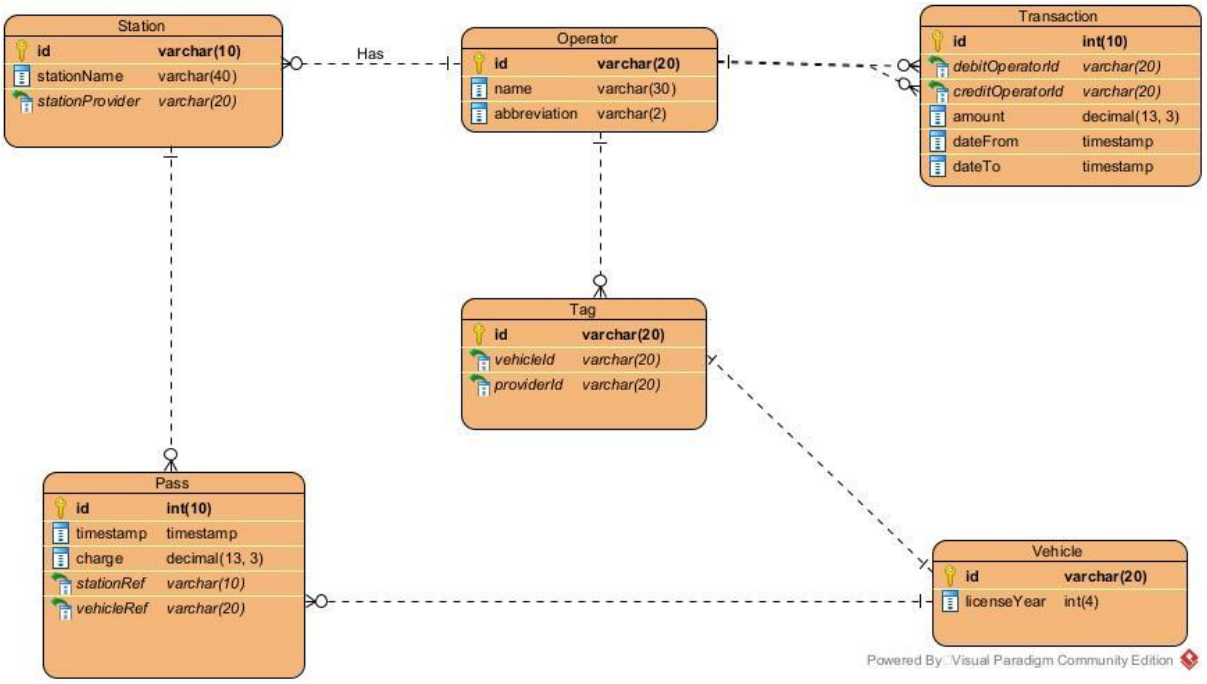
Βάσει των στοιχείων που μας δόθηκαν για την ανάλυση του προβλήματος (sampleData), πραγματοποιούνται κατά μέσο όρο προσεγγιστικά 25 διελεύσεις τη μέρα. Προφανώς και κάποιες μπορεί να πραγματοποιούνται συγχρόνως. Οι χρήστες που μας δίνονται είναι 100 και οι σταθμοί διοδίων 85. Η υλοποίησή μας θα λάβει υπόψη τα συγκεκριμένα αριθμητικά δεδομένα ως κάτω φράγματα.

Με αυτά τα δεδομένα, η αποθήκευση διελεύσεων καθώς και οι αιτήσεις από τους χειριστές για την δημιουργία των εγγράφων, ρεαλιστικά, θα είναι άμεσες, λόγω των μικρών αριθμών που δουλεύουμε. Σε ένα πραγματικό σενάριο οι απαιτήσεις του συστήματος θα ήταν πολύ υψηλότερες.

3.3 Απαιτήσεις οργάνωσης δεδομένων

3.3.1 Απαιτήσεις και περιορισμοί πρόσβασης σε δεδομένα

<table>
<tr><th>Χρήστης</th><th>Διελεύσεις</th><th>Tags</th><th>Οχήματα</th></tr>
<tr><td rowspan="5">Operator</td><td>Πρόσβαση μόνο σε διελεύσεις που αφορούν τους σταθμούς του ή εμπλέκουν tag id που του ανήκει.
Εγγραφή/Ανάγνωση</td><td>Πρόσβαση σε αυτά που ανήκουν στον operator.
(έρχονται από το δικό του σύστημα άλλωστε)
Εγγραφή/Ανάγνωση</td><td>Πρόσβαση σε οχήματα που έχουν tag id του operator
Ανάγνωση</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td>Συναλλαγές/Οφειλές</td><td>Σταθμοί</td><td></td></tr>
<tr><td>Ανάγνωση μόνο σε όποιες εμπλέκεται.</td><td>Πρόσβαση στους σταθμούς του operator
Εγγραφή/Ανάγνωση</td><td></td></tr>
<tr><td colspan="3"></td></tr>
<tr><td>Admin</td><td colspan="3">Full πρόσβαση σε όλα τα δεδομένα του συστήματος Εγγραφή/Ανάγνωση</td></tr>
</table>



Όλες οι οντότητες έχουν ως primary key ένα δικό τους μοναδικό id.

- Το χαρακτηριστικό station.id αποτελεί primary key της οντότητας Station
- Το χαρακτηριστικό operator.id αποτελεί primary key της οντότητας Operator
- Το χαρακτηριστικό pass.id αποτελεί primary key της οντότητας Pass
- Το χαρακτηριστικό tag.id αποτελεί primary key της οντότητας Tag
- Το χαρακτηριστικό vehicle.id αποτελεί primary key της οντότητας Vehicle

3.4 Λοιπές απαιτήσεις

3.4.1 Απαιτήσεις διαθεσιμότητας λογισμικού

Το λογισμικό θα πρέπει ει δυνατόν να είναι διαθέσιμο συνεχώς (τουλάχιστον 80%). Τις φορές που το σύστημα δεν θα είναι online, θα πρέπει να υπάρχει η δυνατότητα λειτουργίας offline στα διάφορα σημεία διοδίων και η αποθήκευση τοπικά πληροφοριών ώστε να ενσωματωθούν στη συνέχεια.

3.4.2 Απαιτήσεις ασφάλειας

- Στην βάση προστατεύουμε τα προσωπικά δεδομένα των χρηστών παραχορώντας τα κατάλληλα δικαιώματα ανάγνωσης και εγγραφής.
- Κατάλληλη υλοποίηση του API ώστε να μην επιτρέπονται SQL-injections.

4. Παράρτημα: ακρωνύμια και συντομογραφίες

CLI: Command Line Interface

API: Application Programming Interface

REST: REpresentational State Transfer

DBMS: Database Management System

JSON: JavaScript Object Notation

CSV: Comma Separated Values

HTTP: HyperText Transfer Protocol

CSS: Cascading Style Sheets

PHP: Preprocessor Hypertext

HTML: HyperText Markup Language

UML: Unified Modeling Language

SQL: Structured Query Language