Use Case 1: Είσοδος Χρήστη (Login)

#### Περιγραφη:

Ο χρήστης εισέρχεται στο σύστημα μέσω διαπιστευτηρίων (username, password). Εάν η διαπίστευση είναι επιτυχής, το σύστημα επιστρέφει ένα **token** για την εξουσιοδότηση του χρήστη στις επόμενες κλήσεις.

#### ****Λειτουργικές Απαιτήσεις****

1. Το σύστημα πρέπει να υποστηρίζει:
   1. Endpoint /login με τη μέθοδο POST.
   2. Παραμέτρους username και password, κωδικοποιημένες ως "application/x-www-form-urlencoded".
   3. Σε περίπτωση επιτυχίας, επιστρέφει ένα json object με το token αυτού: {πχ "token":"FOO"}
2. Το token πρέπει να αποθηκεύεται με ασφαλή τρόπο για επόμενες κλήσεις.
3. Σε περίπτωση αποτυχίας, το σύστημα πρέπει να επιστρέφει κατάλληλο μήνυμα λάθους και HTTP status code.

#### ****Μη Λειτουργικές Απαιτήσεις****

1. Το σύστημα πρέπει να χρησιμοποιεί ασφαλή επικοινωνία μέσω HTTPS.
2. Το token πρέπει να έχει χρονικό όριο λήξης (π.χ., 24 ώρες).
3. Το σύστημα πρέπει να υποστηρίζει **logging** για επιτυχημένες και αποτυχημένες προσπάθειες εισόδου.

#### Προϋποθέσεις εκτέλεσης:

Ο χρήστης πρέπει να διαθέτει έγκυρο **username** και **password** αποθηκευμένα στη βάση δεδομένων του συστήματος.

Το backend πρέπει να είναι συνδεδεμένο με τη βάση δεδομένων χρηστών.

Η κλήση πρέπει να γίνεται μέσω ασφαλούς πρωτοκόλλου HTTPS.

#### Δεδομένα εισόδου:

**Username**: Το μοναδικό αναγνωριστικό του χρήστη. **Password**: Ο κωδικός πρόσβασης του χρήστη.

#### Αλληλουχία ενεργειών - επιθυμητή συμπεριφορά:

1. Ο χρήστης εισάγει τα διαπιστευτήρια του (username, password).
2. Το frontend καλεί το endpoint /login με τη μέθοδο POST και κωδικοποιημένα δεδομένα.
3. Το backend επαληθεύει τα διαπιστευτήρια με βάση την αποθηκευμένη βάση δεδομένων χρηστών.
4. Εάν η επαλήθευση είναι επιτυχής:
5. Το σύστημα δημιουργεί ένα access token και το επιστρέφει στο χρήστη.
6. Το API επιστρέφει HTTP status code 200 και JSON object με το token: {"token": "FOO"}
7. Εναλλακτικά To API επιστρέφει HTTP status code:
8. 204 No content Σε περίπτωση που η κλήση είναι επιτυχής, αλλά απάντηση είναι κενή (δεν επέστρεψε δεδομένα)
9. 400 Bad request Σε περίπτωση που οι παράμετροι που δίνονται σε μία κλήση δεν είναι έγκυρες (π.χ. λείπει κάποιο υποχρεωτικό πεδίο)
10. 401 Not authorized Σε περίπτωση που η αίτηση γίνεται από μη διαπιστευμένο χρήστη (μόνο εφόσον υλοποιηθεί διαπίστευση χρηστών)
11. 500 Internal server error Σε περίπτωση οποιουδήποτε άλλου σφάλματος

#### Δεδομένα εξόδου:

Σε περίπτωση επιτυχίας: JSON object που περιέχει το access token του χρήστη. {"token": "FOO"}. HTTP Status Code: 200.

Εναλλακτικά το αντίστοιχο HTTP status code (204, 400, 401, 500)

Use Case 2: Έξοδος Χρήστη (Logout)

#### Περιγραφη:

Ο χρήστης εξέρχεται από το σύστημα. Το token που έχει εκδοθεί παύει να ισχύει, και το σύστημα διαγράφει ή λήγει το session του χρήστη.

#### ****Λειτουργικές Απαιτήσεις****

1. Το σύστημα πρέπει να υποστηρίζει:
2. Endpoint /logout με τη μέθοδο POST.
3. Το token πρέπει να παρέχεται μέσω του HTTP header X-OBSERVATORY-AUTH.
4. Το token πρέπει να ανακληθεί ή να λήξει μετά την επιτυχία του logout.
5. Σε περίπτωση επιτυχίας, το API πρέπει να επιστρέφει HTTP status code 200 (empty response body).

#### ****Μη Λειτουργικές Απαιτήσεις****

1. Το σύστημα πρέπει να διαχειρίζεται την ανάκληση του token (π.χ., αποθήκευση σε λίστα ανακλημένων tokens για τη διάρκεια ισχύος του).
2. Το σύστημα πρέπει να λειτουργεί μέσω HTTPS για ασφάλεια δεδομένων.
3. Πρέπει να υποστηρίζεται logging για τις ενέργειες logout.

#### Προϋποθέσεις εκτέλεσης:

Ο χρήστης πρέπει να έχει ήδη συνδεθεί στο σύστημα και να διαθέτει **έγκυρο access token**.

Το backend πρέπει να υποστηρίζει τον μηχανισμό ελέγχου και ανάκλησης του token.

Η κλήση πρέπει να γίνεται μέσω ασφαλούς πρωτοκόλλου HTTPS.

#### Δεδομένα εισόδου:

Access Token: Το token που παρέχεται στο HTTP header X-OBSERVATORY-AUTH.

#### Αλληλουχία ενεργειών - επιθυμητή συμπεριφορά:

1. Ο χρήστης καλεί το endpoint /logout με τη μέθοδο POST.
2. Το token παρέχεται στο HTTP header X-OBSERVATORY-AUTH.
3. Το backend επαληθεύει το token:
4. Εάν το token είναι έγκυρο, ακυρώνει το session ή καταχωρεί το token ως ανακλημένο.
5. Επιστρέφει HTTP status code 200.
6. Αλλιώς:

a) Το API επιστρέφει HTTP status code 204 No content, 400 Bad request, 401 Not authorized, 500 Internal server error αναλόγως

#### Δεδομένα εξόδου:

Σε περίπτωση επιτυχίας: Empty Response Body. HTTP Status Code: 200.

Εναλλακτικά το αντίστοιχο HTTP status code (204, 400, 401, 500)

Use Case 3: Έλεγχος Υγείας Συστήματος (Healthcheck)

#### ****Περιγραφή****

Επιβεβαιώνει τη συνδεσιμότητα του συστήματος (backend και βάση δεδομένων). Επιστρέφει πληροφορίες σχετικά με τη σύνδεση, τον αριθμό των σταθμών διοδίων, tags, και διελεύσεων.

#### ****Λειτουργικές Απαιτήσεις****

1. Υποστήριξη του endpoint /admin/healthcheck με τη μέθοδο GET.
2. Επιστροφή JSON object με την κατάσταση σύνδεσης και στατιστικά δεδομένα.
3. Επιστροφή HTTP status code 200 σε περίπτωση επιτυχίας και 401 σε περίπτωση αποτυχίας.

#### ****Μη Λειτουργικές Απαιτήσεις****

1. Ασφάλεια μέσω HTTPS.
2. Χρονικό όριο για την εκτέλεση του ελέγχου (timeout).
3. Logging για επιτυχημένες και αποτυχημένες αιτήσεις.

#### ****Αλληλουχία Ενεργειών / Επιθυμητή Συμπεριφορά****

1. Ο χρήστης καλεί το endpoint /admin/healthcheck μέσω GET.
2. Το backend ελέγχει:
3. Σύνδεση με τη βάση δεδομένων.
4. Αριθμό σταθμών, tags, και διελεύσεων.
5. Επιστρέφει:
   1. JSON object με την κατάσταση σύνδεσης.
   2. HTTP status code 200 ή 401 σε περίπτωση αποτυχία

**Δεδομένα Εξόδου**

Επιτυχία:

{"status": "OK",

"dbconnection": "connected",

"n\_stations": 10,

"n\_tags": 500,

"n\_passes": 10000}

Αποτυχία:

{"status": "failed",

"dbconnection": "not connected"}

### Use Case 4: Αρχικοποίηση Σταθμών Διοδίων (Reset Stations)

#### ****Περιγραφή****

Επαναφέρει τα δεδομένα των σταθμών διοδίων χρησιμοποιώντας το αρχείο tollstations2024.csv

#### ****Λειτουργικές Απαιτήσεις****

1. Υποστήριξη του endpoint /admin/resetstations με τη μέθοδο POST.
2. Ανάγνωση του αρχείου tollstations2024.csv για την αρχικοποίηση των σταθμών.
3. Επιστροφή HTTP status code 200 σε περίπτωση επιτυχίας ή 500 με μήνυμα σφάλματος.

#### ****Μη Λειτουργικές Απαιτήσεις****

1. Διασφάλιση της συνέπειας δεδομένων κατά την αρχικοποίηση.
2. Logging για την εκτέλεση του reset.

#### ****Αλληλουχία Ενεργειών / Επιθυμητή Συμπεριφορά****

1. Ο διαχειριστής καλεί το endpoint /admin/resetstations μέσω POST.
2. Το backend:
3. Επεξεργάζεται το αρχείο tollstations2024.csv.
4. Διαγράφει τα υπάρχοντα δεδομένα σταθμών και τα αντικαθιστά με τα νέα.
5. Επιστρέφει JSON object με την κατάσταση εκτέλεσης.

#### ****Προϋποθέσεις Εκτέλεσης****

1. Το αρχείο tollstations2024.csv να είναι διαθέσιμο και σωστά διαμορφωμένο.
2. Η βάση δεδομένων να είναι προσβάσιμη.

#### ****Δεδομένα Εισόδου****

* Αρχείο tollstations2024.csv.

**Δεδομένα Εξόδου**

Επιτυχία:

{"status": "OK"}

Αποτυχία:

{"status": "failed",

"info": "Invalid file format"}

### Use Case 5: Αρχικοποίηση Διελεύσεων (Reset Passes)

#### ****Περιγραφή****

Διαγράφει όλα τα γεγονότα διελεύσεων και επανεκκινεί τη βάση δεδομένων με ένα διαχειριστικό λογαριασμό.

#### ****Λειτουργικές Απαιτήσεις****

1. Υποστήριξη του endpoint /admin/resetpasses με τη μέθοδο POST.
2. Διαγραφή όλων των δεδομένων διελεύσεων.
3. Δημιουργία διαχειριστικού λογαριασμού με username admin και password freepasses4all.

#### ****Μη Λειτουργικές Απαιτήσεις****

1. Ασφάλεια κατά την εκτέλεση του reset.
2. Logging για την εκτέλεση της ενέργειας.

#### ****Προϋποθέσεις Εκτέλεσης****

1. Η βάση δεδομένων να είναι προσβάσιμη.

#### ****Αλληλουχία Ενεργειών / Επιθυμητή Συμπεριφορά****

1. Ο διαχειριστής καλεί το endpoint /admin/resetpasses μέσω POST.
2. Το backend:
3. Διαγράφει όλα τα δεδομένα διελεύσεων.
4. Επανεκκινεί την αρχικοποίηση του διαχειριστικού λογαριασμού.
5. Επιστρέφει JSON object με την κατάσταση εκτέλεσης.

#### ****Δεδομένα Εξόδου****

Επιτυχία:

{"status": "OK"}

Αποτυχία:

{"status": "failed",

"info": "Database error"}

### Use Case 6: Προσθήκη Διελεύσεων (Add Passes)

#### ****Περιγραφή****

Προσθέτει νέα δεδομένα διελεύσεων στη βάση δεδομένων από αρχείο CSV.

#### ****Λειτουργικές Απαιτήσεις****

1. Υποστήριξη του endpoint /admin/addpasses με τη μέθοδο POST.
2. Επεξεργασία αρχείου CSV με format passes-sample.csv.
3. Επιστροφή HTTP status code 200 σε περίπτωση επιτυχίας ή 500 με μήνυμα σφάλματος.

#### ****Μη Λειτουργικές Απαιτήσεις****

1. Το σύστημα πρέπει να ελέγχει τη μορφή του αρχείου πριν την επεξεργασία.
2. Logging για επιτυχημένες και αποτυχημένες εισαγωγές.

#### ****Αλληλουχία Ενεργειών / Επιθυμητή Συμπεριφορά****

1. Ο διαχειριστής καλεί το endpoint /admin/addpasses μέσω POST, ανεβάζοντας το αρχείο CSV.
2. Το backend:
   1. Ελέγχει τη μορφή του αρχείου.
   2. Ενημερώνει τα δεδομένα διελεύσεων.
3. Επιστρέφει JSON object με την κατάσταση εκτέλεσης.

#### ****Προϋποθέσεις Εκτέλεσης****

1. Το αρχείο CSV να είναι σωστά διαμορφωμένο.
2. Η βάση δεδομένων να είναι προσβάσιμη.

#### ****Δεδομένα Εισόδου****

* Αρχείο passes-sample.csv.

#### ****Δεδομένα Εξόδου****

Επιτυχία:

{ "status": "OK" }

Αποτυχία:   
{"status": "failed",

"info": "File format error"}

### Use Case 7: Διελεύσεις ανά Σταθμό (tollStationPasses)

#### ****Περιγραφή****

Το endpoint /tollStationPasses/:tollStationID/:date\_from/:date\_to υποστηρίζει τη μέθοδο GET και επιστρέφει αναλυτική λίστα με τις διελεύσεις για έναν συγκεκριμένο σταθμό διοδίων (identified by tollStationID) μέσα σε μια χρονική περίοδο που ορίζεται από τα path parameters date\_from και date\_to.

### ****Λειτουργικές Απαιτήσεις****

### Το σύστημα πρέπει να:

1. Υποστηρίζει τη μέθοδο GET στο endpoint /tollStationPasses/:tollStationID/:date\_from/:date\_to.
2. Λαμβάνει ως **path parameters**:
   * 1. tollStationID: Το μοναδικό ID του σταθμού.
     2. date\_from: Η αρχική ημερομηνία σε μορφή YYYYMMDD.
     3. date\_to: Η τελική ημερομηνία σε μορφή YYYYMMDD.
3. Επιστρέφει αντικείμενο JSON που περιέχει:
   * 1. Στοιχεία του σταθμού διοδίων.
     2. Ανάλυση όλων των διελεύσεων για την επιλεγμένη περίοδο.
4. Ελέγχει αν το date\_from είναι πριν ή ίσο με το date\_to.
5. Το JSON αποτέλεσμα πρέπει να περιέχει τα εξής πεδία:
6. Γενικά στοιχεία: stationID, stationOperator, requestTimestamp, periodFrom, periodTo, nPasses.
7. Αναλυτικές πληροφορίες διελεύσεων: passIndex, passID, timestamp, tagID, tagProvider, vehicleType, passCharge.
8. Σε περίπτωση που δεν υπάρχουν διελεύσεις για την επιλεγμένη περίοδο: Να επιστρέφεται κενή λίστα με status code 204 No Content.
9. Να επιστρέφεται 400 Bad Request αν οι παράμετροι δεν είναι έγκυροι (π.χ., κακή μορφοποίηση ημερομηνιών).

### ****Μη Λειτουργικές Απαιτήσεις****

1. **Ασφάλεια**: Το endpoint πρέπει να λειτουργεί αποκλειστικά μέσω HTTPS.
2. **Απόδοση**: Το σύστημα πρέπει να επεξεργάζεται μεγάλους όγκους δεδομένων σε λογικό χρόνο.
3. **Αποδοτικότητα**: Υποστήριξη caching για συχνές αιτήσεις (π.χ., Redis).

### ****Αλληλουχία Ενεργειών / Επιθυμητή Συμπεριφορά****

1. **Εισαγωγή Αιτήματος**: Ο χρήστης (ή το σύστημα) καλεί το endpoint /tollStationPasses/:tollStationID/:date\_from/:date\_to με έγκυρα path parameters.
2. **Έλεγχος Παράμετρων**: Το backend επαληθεύει:
3. Το format των ημερομηνιών date\_from και date\_to.
4. Την ύπαρξη του σταθμού με ID tollStationID.
5. **Ανάκτηση Δεδομένων**: Το backend ανακτά από τη βάση δεδομένων όλες τις διελεύσεις που αντιστοιχούν στον σταθμό tollStationID μέσα στο χρονικό διάστημα από date\_from έως date\_to.
6. **Επεξεργασία Αποτελεσμάτων**:
7. Εάν υπάρχουν δεδομένα: Επιστρέφονται σε JSON format.
8. Εάν δεν υπάρχουν δεδομένα: Επιστρέφεται HTTP status code 204 No Content.
9. **Απόκριση στο Χρήστη**:
10. Αν η κλήση ήταν επιτυχής, επιστρέφεται HTTP status code 200 με το JSON αντικείμενο.
11. Σε περίπτωση σφάλματος, επιστρέφεται το αντίστοιχο status code (400, 500).

### ****Προϋποθέσεις Εκτέλεσης****

1. Η βάση δεδομένων πρέπει να περιέχει δεδομένα για τον σταθμό tollStationID.
2. Ο χρήστης να έχει κατάλληλα δικαιώματα πρόσβασης, να έχει κάνει login.

### ****Δεδομένα Εισόδου****

**Path Parameters**:

1. tollStationID: Το μοναδικό ID του σταθμού διοδίων (π.χ., "NAO01").
2. date\_from: Η αρχική ημερομηνία (π.χ., "20240101").
3. date\_to: Η τελική ημερομηνία (π.χ., "20240131").

### ****Δεδομένα Εξόδου****

Επιτυχία (HTTP Status Code 200): JSON με τις ακόλουθες πληροφορίες:

{

"stationID": "NAO01",

"stationOperator": "NeaOdos",

"requestTimestamp": "2024-01-31 12:00",

"periodFrom": "2024-01-01",

"periodTo": "2024-01-31",

"nPasses": 2,

"passList": [

{

"passIndex": 1,

"passID": "P001",

"timestamp": "2024-01-02 10:15",

"tagID": "TAG123",

"tagProvider": "AttikiOdos",

"vehicleType": "visitor",

"passCharge": 2.50

},

{

"passIndex": 2,

"passID": "P002",

"timestamp": "2024-01-10 14:30",

"tagID": "TAG456",

"tagProvider": "KentrikiOdos",

"vehicleType": "home",

"passCharge": 1.80

}

]

}

Κενή Λίστα (HTTP Status Code 204): Κενή απόκριση (No Content).

Λάθος Παράμετροι (HTTP Status Code 400): JSON με μήνυμα σφάλματος: {"error": "Invalid date format or tollStationID"}

Σφάλμα Συστήματος (HTTP Status Code 500): JSON με μήνυμα σφάλματος: {"error": "Internal server error"}

### Use Case 8: Ανάλυση Διελεύσεων μεταξύ Operators (passAnalysis)

#### ****Περιγραφή****

Το endpoint /passAnalysis/:stationOpID/:tagOpID/:date\_from/:date\_to υποστηρίζει τη μέθοδο GET και επιστρέφει αναλυτική λίστα με τα γεγονότα διελεύσεων που πραγματοποιήθηκαν σε σταθμούς του λειτουργού stationOpID από tags που ανήκουν στον λειτουργό tagOpID μέσα σε ένα συγκεκριμένο χρονικό διάστημα (date\_from, date\_to).

### ****Λειτουργικές Απαιτήσεις****

1. Το σύστημα πρέπει να:
2. Υποστηρίζει τη μέθοδο GET στο endpoint /passAnalysis/:stationOpID/:tagOpID/:date\_from/:date\_to.
3. Λαμβάνει ως **path parameters**:
   * 1. stationOpID: Το ID του λειτουργού του σταθμού.
     2. tagOpID: Το ID του λειτουργού του tag.
     3. date\_from: Η αρχική ημερομηνία σε μορφή YYYYMMDD.
     4. date\_to: Η τελική ημερομηνία σε μορφή YYYYMMDD.
4. Επιστρέφει JSON αντικείμενο που περιέχει:
5. Γενικές πληροφορίες για το χρονικό διάστημα και τους operators.
6. Λεπτομέρειες για κάθε διέλευση.
7. Επιστροφή:
8. HTTP status code 200 με JSON αν υπάρχουν δεδομένα.
9. HTTP status code 204 No Content αν δεν υπάρχουν δεδομένα για το χρονικό διάστημα.
10. HTTP status code 400 Bad Request για μη έγκυρες παραμέτρους.
11. HTTP status code 500 Internal Server Error σε περιπτώσεις σφαλμάτων συστήματος.
12. Το σύστημα πρέπει να επαληθεύει ότι:
13. Το format των ημερομηνιών date\_from και date\_to είναι σωστό (YYYYMMDD).
14. Το date\_from είναι πριν ή ίσο με το date\_to.

### ****Μη Λειτουργικές Απαιτήσεις****

1. **Ασφάλεια**: Η επικοινωνία να γίνεται αποκλειστικά μέσω HTTPS.
2. **Απόδοση**: Το σύστημα πρέπει να επεξεργάζεται μεγάλους όγκους δεδομένων σε λογικό χρόνο.

### ****Αλληλουχία Ενεργειών / Επιθυμητή Συμπεριφορά****

1. **Εισαγωγή Αιτήματος**: Ο χρήστης καλεί το endpoint /passAnalysis/:stationOpID/:tagOpID/:date\_from/:date\_to με έγκυρα path parameters.
2. **Έλεγχος Παράμετρων**: Το backend επαληθεύει:
3. Την εγκυρότητα των ημερομηνιών.
4. Την ύπαρξη των stationOpID και tagOpID.
5. **Ανάκτηση Δεδομένων**: Το backend ανακτά από τη βάση δεδομένων όλα τα γεγονότα διελεύσεων για τους επιλεγμένους operators και την επιλεγμένη περίοδο.
6. **Επεξεργασία Αποτελεσμάτων**:

Εάν υπάρχουν δεδομένα: Επιστρέφονται σε JSON format.

Εάν δεν υπάρχουν δεδομένα: Επιστρέφεται HTTP status code 204 No Content.

1. **Απόκριση στο Χρήστη**: Αν η κλήση ήταν επιτυχής, επιστρέφεται HTTP status code 200 με το JSON αντικείμενο. Σε περίπτωση σφάλματος, επιστρέφεται το αντίστοιχο status code (400, 500).

### ****Προϋποθέσεις Εκτέλεσης****

1. Οι operators stationOpID και tagOpID να είναι έγκυροι και καταχωρημένοι στο σύστημα.
2. Η βάση δεδομένων να περιέχει δεδομένα για τους επιλεγμένους operators.

### ****Δεδομένα Εισόδου****

1. **Path Parameters**:
   1. stationOpID: Το ID του λειτουργού των σταθμών διοδίων (π.χ., "NeaOdos").
   2. tagOpID: Το ID του λειτουργού των tags (π.χ., "AttikiOdos").
   3. date\_from: Η αρχική ημερομηνία (π.χ., "20240101").
   4. date\_to: Η τελική ημερομηνία (π.χ., "20240131").

### ****Δεδομένα Εξόδου****

Επιτυχία (HTTP Status Code 200): JSON με τις ακόλουθες πληροφορίες:

{

"stationOpID": "NeaOdos",

"tagOpID": "AttikiOdos",

"requestTimestamp": "2024-01-31 12:00",

"periodFrom": "2024-01-01",

"periodTo": "2024-01-31",

"nPasses": 2,

"passList": [

{

"passIndex": 1,

"passID": "P001",

"stationID": "NAO01",

"timestamp": "2024-01-02 10:15",

"tagID": "TAG123",

"passCharge": 2.50

},

{

"passIndex": 2,

"passID": "P002",

"stationID": "NAO02",

"timestamp": "2024-01-10 14:30",

"tagID": "TAG456",

"passCharge": 1.80

}

]

}

Κενή Λίστα (HTTP Status Code 204): Κενή απόκριση (No Content).

Λάθος Παράμετροι (HTTP Status Code 400): JSON με μήνυμα σφάλματος: {"error": "Invalid date format or operator IDs"}

Σφάλμα Συστήματος (HTTP Status Code 500): JSON με μήνυμα σφάλματος: {"error": "Internal server error"}

### Use Case 9: Κόστος Διελεύσεων μεταξύ Operators (passesCost)

#### ****Περιγραφή****

Το endpoint /passesCost/:tollOpID/:tagOpID/:date\_from/:date\_to υποστηρίζει τη μέθοδο GET και επιστρέφει το συνολικό αριθμό των διελεύσεων και το κόστος τους για μια χρονική περίοδο. Αναλύει το ποσό που οφείλει ο operator των tags (tagOpID) στον operator των σταθμών (tollOpID).

### ****Λειτουργικές Απαιτήσεις****

### Το σύστημα πρέπει να:

### Υποστηρίζει τη μέθοδο GET στο endpoint /passesCost/:tollOpID/:tagOpID/:date\_from/:date\_to.

### Λαμβάνει ως ****path parameters****:

### tollOpID: Το ID του operator των σταθμών.

### tagOpID: Το ID του operator των tags.

### date\_from: Η αρχική ημερομηνία σε μορφή YYYYMMDD.

### date\_to: Η τελική ημερομηνία σε μορφή YYYYMMDD.

* 1. Επιστρέφει JSON αντικείμενο με:
     1. Γενικές πληροφορίες για τους operators και την επιλεγμένη περίοδο.
     2. Συνολικό αριθμό διελεύσεων και το συνολικό κόστος.

1. Επιστροφή:
   1. HTTP status code 200 με JSON αν υπάρχουν δεδομένα.
   2. HTTP status code 204 No Content αν δεν υπάρχουν δεδομένα.
   3. HTTP status code 400 Bad Request αν οι παράμετροι δεν είναι έγκυρες.
   4. HTTP status code 500 Internal Server Error για απρόβλεπτα σφάλματα.
2. Το σύστημα πρέπει να επαληθεύει ότι:
   1. Το format των ημερομηνιών date\_from και date\_to είναι σωστό (YYYYMMDD).
   2. Το date\_from είναι πριν ή ίσο με το date\_to.

### ****Μη Λειτουργικές Απαιτήσεις****

1. **Ασφάλεια**: Η επικοινωνία να γίνεται αποκλειστικά μέσω HTTPS.
2. **Απόδοση**: Το σύστημα πρέπει να επεξεργάζεται μεγάλους όγκους δεδομένων σε λογικό χρόνο.

### ****Αλληλουχία Ενεργειών / Επιθυμητή Συμπεριφορά****

1. **Εισαγωγή Αιτήματος**: Ο χρήστης καλεί το endpoint /passesCost/:tollOpID/:tagOpID/:date\_from/:date\_to με έγκυρα path parameters.
2. **Έλεγχος Παράμετρων**: Το backend επαληθεύει:
3. Την εγκυρότητα των ημερομηνιών.
4. Την ύπαρξη των operators tollOpID και tagOpID.
5. **Ανάκτηση Δεδομένων**: Το backend ανακτά από τη βάση δεδομένων το συνολικό αριθμό διελεύσεων και το συνολικό κόστος για τους επιλεγμένους operators και την επιλεγμένη περίοδο.
6. **Επεξεργασία Αποτελεσμάτων**: Εάν υπάρχουν δεδομένα: Επιστρέφονται σε JSON format. Εάν δεν υπάρχουν δεδομένα: Επιστρέφεται HTTP status code 204 No Content.
7. **Απόκριση στο Χρήστη**: Αν η κλήση ήταν επιτυχής, επιστρέφεται HTTP status code 200 με το JSON αντικείμενο. Σε περίπτωση σφάλματος, επιστρέφεται το αντίστοιχο status code (400, 500).

### ****Προϋποθέσεις Εκτέλεσης****

1. Οι operators tollOpID και tagOpID να είναι έγκυροι και καταχωρημένοι στο σύστημα.
2. Η βάση δεδομένων να περιέχει δεδομένα για τους επιλεγμένους operators.

### ****Δεδομένα Εισόδου****

**Path Parameters**:

* 1. tollOpID: Το ID του λειτουργού των σταθμών διοδίων (π.χ., "NeaOdos").
  2. tagOpID: Το ID του λειτουργού των tags (π.χ., "AttikiOdos").
  3. date\_from: Η αρχική ημερομηνία (π.χ., "20240101").
  4. date\_to: Η τελική ημερομηνία (π.χ., "20240131").

### ****Δεδομένα Εξόδου****

Επιτυχία (HTTP Status Code 200): JSON με τις ακόλουθες πληροφορίες:

{

"tollOpID": "NeaOdos",

"tagOpID": "AttikiOdos",

"requestTimestamp": "2024-01-31 12:00",

"periodFrom": "2024-01-01",

"periodTo": "2024-01-31",

"nPasses": 150,

"passesCost": 450.75

}

Κενή Λίστα (HTTP Status Code 204): Κενή απόκριση (No Content).

Λάθος Παράμετροι (HTTP Status Code 400): JSON με μήνυμα σφάλματος: {"error": "Invalid date format or operator IDs"}

Σφάλμα Συστήματος (HTTP Status Code 500): JSON με μήνυμα σφάλματος: {"error": "Internal server error"}

### Use Case 10: Διελεύσεις και Χρεώσεις Λοιπών Λειτουργών (chargesBy)

#### ****Περιγραφή****

Το endpoint /chargesBy/:tollOpID/:date\_from/:date\_to υποστηρίζει τη μέθοδο GET και επιστρέφει λίστα με δεδομένα που περιλαμβάνουν τον αριθμό διελεύσεων και το συνολικό κόστος που οφείλουν οι λοιποί operators (visiting operators) στον operator των σταθμών (tollOpID) για ένα συγκεκριμένο χρονικό διάστημα.

### ****Λειτουργικές Απαιτήσεις****

1. Το σύστημα πρέπει να:
   1. Υποστηρίζει τη μέθοδο GET στο endpoint /chargesBy/:tollOpID/:date\_from/:date\_to.
   2. Λαμβάνει ως **path parameters**:
      1. tollOpID: Το ID του operator των σταθμών.
      2. date\_from: Η αρχική ημερομηνία σε μορφή YYYYMMDD.
      3. date\_to: Η τελική ημερομηνία σε μορφή YYYYMMDD.
   3. Επιστρέφει JSON αντικείμενο με: Γενικές πληροφορίες για την περίοδο και τον tollOpID. Μία λίστα (vOpList) με όλους τους visiting operators, τον αριθμό διελεύσεων και το συνολικό κόστος.
2. Επιστροφή:
   1. HTTP status code 200 με JSON αν υπάρχουν δεδομένα.
   2. HTTP status code 204 No Content αν δεν υπάρχουν δεδομένα.
   3. HTTP status code 400 Bad Request για μη έγκυρες παραμέτρους.
   4. HTTP status code 500 Internal Server Error για απρόβλεπτα σφάλματα.
3. Το σύστημα πρέπει να επαληθεύει:
   1. Την εγκυρότητα των ημερομηνιών.
   2. Το date\_from να είναι πριν ή ίσο με το date\_to.

### ****Μη Λειτουργικές Απαιτήσεις****

1. **Ασφάλεια**: Η επικοινωνία να γίνεται αποκλειστικά μέσω HTTPS.
2. **Απόδοση**: Το σύστημα πρέπει να υποστηρίζει επεξεργασία μεγάλων όγκων δεδομένων σε λογικό χρόνο.

### ****Αλληλουχία Ενεργειών / Επιθυμητή Συμπεριφορά****

1. **Εισαγωγή Αιτήματος**: Ο χρήστης καλεί το endpoint /chargesBy/:tollOpID/:date\_from/:date\_to με έγκυρα path parameters.
2. **Έλεγχος Παράμετρων**: Το backend επαληθεύει:
3. Την εγκυρότητα των ημερομηνιών.
4. Την ύπαρξη του operator tollOpID.
5. **Ανάκτηση Δεδομένων**: Το backend ανακτά από τη βάση δεδομένων:
6. Όλα τα δεδομένα διελεύσεων από τους visiting operators.
7. Το συνολικό κόστος που οφείλουν στον operator tollOpID.
8. **Επεξεργασία Αποτελεσμάτων**: Εάν υπάρχουν δεδομένα: Επιστρέφονται σε JSON format.Εάν δεν υπάρχουν δεδομένα: Επιστρέφεται HTTP status code 204 No Content.
9. **Απόκριση στο Χρήστη**:
10. Αν η κλήση ήταν επιτυχής, επιστρέφεται HTTP status code 200 με το JSON αντικείμενο.
11. Σε περίπτωση σφάλματος, επιστρέφεται το αντίστοιχο status code (400, 500).

### ****Προϋποθέσεις Εκτέλεσης****

1. Ο tollOpID να είναι έγκυρος και καταχωρημένος στο σύστημα.
2. Η βάση δεδομένων να περιέχει δεδομένα για τους visiting operators.

### ****Δεδομένα Εισόδου****

**Path Parameters**:

* 1. tollOpID: Το ID του λειτουργού των σταθμών (π.χ., "NeaOdos").
  2. date\_from: Η αρχική ημερομηνία (π.χ., "20240101").
  3. date\_to: Η τελική ημερομηνία (π.χ., "20240131").

### ****Δεδομένα Εξόδου****

Επιτυχία (HTTP Status Code 200): JSON με τις ακόλουθες πληροφορίες:

{

"tollOpID": "NeaOdos",

"requestTimestamp": "2024-01-31 12:00",

"periodFrom": "2024-01-01",

"periodTo": "2024-01-31",

"vOpList": [

{

"visitingOpID": "AttikiOdos",

"nPasses": 150,

"passesCost": 450.75

},

{

"visitingOpID": "KentrikiOdos",

"nPasses": 80,

"passesCost": 240.00

}

]

}

Κενή Λίστα (HTTP Status Code 204): Κενή απόκριση (No Content).

Λάθος Παράμετροι (HTTP Status Code 400): JSON με μήνυμα σφάλματος: {"error": "Invalid date format or tollOpID"}

Σφάλμα Συστήματος (HTTP Status Code 500): JSON με μήνυμα σφάλματος: {"error": "Internal server error"}

### Use Case 11: Υπολογισμός Οφειλών μεταξύ λειτουργών (netCharges)

**Περιγραφή**

Το endpoint /netCharges/:OpID1/:OpID2/:date\_from/:date\_to υποστηρίζει την μέθοδο GET και επιστρέφει το συμψηφισμό που δημιουργείται για τις δυο εταιρείες για ένα συγκεκριμένο.

### ****Λειτουργικές Απαιτήσεις****

1. Το σύστημα πρέπει να:
2. Υποστηρίζει τη μέθοδο GET στο endpoint /netCharges/:OpID1/:OpID2/:date\_from/:date\_to.
3. Λαμβάνει ως **path parameters**:
   * 1. OpID1: Το ID του operator που ζητά τον συμψηφισμό.
     2. OpID2: Το ID του operator με τον οποιο θέλει να δει τον συμψηφισμό
     3. date\_from: Η αρχική ημερομηνία σε μορφή YYYYMMDD.
     4. date\_to: Η τελική ημερομηνία σε μορφή YYYYMMDD.
   1. Επιστρέφει JSON αντικείμενο με: συνολικό συμψηφισμό.
4. Επιστροφή:
5. HTTP status code 200 με JSON αν υπάρχουν δεδομένα.
6. HTTP status code 204 No Content αν δεν υπάρχουν δεδομένα.
7. HTTP status code 400 Bad Request για μη έγκυρες παραμέτρους.
8. HTTP status code 500 Internal Server Error για απρόβλεπτα σφάλματα.
9. Το σύστημα πρέπει να επαληθεύει:
10. Την εγκυρότητα των ημερομηνιών.
11. Το date\_from να είναι πριν ή ίσο με το date\_to.

**Προϋποθέσεις εκτέλεσης**

Επιτυχές Login

Όλα τα δεδομένα διελεύσεων έχουν φορτωθεί στη βάση και είναι διαθέσιμα.

### ****Δεδομένα Εισόδου****

**Path Parameters**:

OpID1: Το ID του λειτουργού των σταθμών διοδίων που κάνει το αίτημα(π.χ., "NeaOdos").

OpID2: Το ID του λειτουργού για τον οποίο ζητά τον συμψηφισμό (π.χ., "AttikiOdos").

date\_from: Η αρχική ημερομηνία (π.χ., "20240101").

date\_to: Η τελική ημερομηνία (π.χ., "20240131").

### ****Αλληλουχία Ενεργειών / Επιθυμητή Συμπεριφορά****

1. **Εισαγωγή Αιτήματος**: Ο χρήστης καλεί το endpoint /netCharges/:OpID1/:OpID2/:date\_from/:date\_to με έγκυρα path parameters.
2. **Έλεγχος Παράμετρων**: Το backend επαληθεύει:
   1. Την εγκυρότητα των ημερομηνιών.
   2. Την ύπαρξη των operators OpID.
3. **Ανάκτηση Δεδομένων**: Το backend ανακτά από τη βάση δεδομένων:
4. Τις ημερήσιες οφειλές της εταιρείας 1 προς την εταιρεία 2
5. Τις ημερησιες οφειλές της εταιρείας 2 προς την εταιρεία 1
6. **Επεξεργασία Αποτελεσμάτων**: Γίνεται η αφαίρεση αυτών (1\_το\_2 - 2\_το\_1) Εάν υπάρχουν δεδομένα: Επιστρέφονται σε JSON format. Εάν δεν υπάρχουν δεδομένα: Επιστρέφεται HTTP status code 204 No Content.
7. **Απόκριση στο Χρήστη**:
   1. Αν η κλήση ήταν επιτυχής, επιστρέφεται HTTP status code 200 με το JSON αντικείμενο.
   2. Σε περίπτωση σφάλματος, επιστρέφεται το αντίστοιχο status code (400, 500).

### ****Δεδομένα Εξόδου****

Επιτυχία (HTTP Status Code 200): JSON με τις ακόλουθες πληροφορίες:

{

"OpID1": "NeaOdos"

"OpID2": "AttikiOdos",

"requestTimestamp": "2024-01-31 12:00",

"periodFrom": "2024-01-01",

"periodTo": "2024-01-31",

"netCharge": "-100.50",

}

Κενή Λίστα (HTTP Status Code 204): Κενή απόκριση (No Content).

Λάθος Παράμετροι (HTTP Status Code 400): JSON με μήνυμα σφάλματος: {"error": "Invalid date format or OpID"}

Σφάλμα Συστήματος (HTTP Status Code 500): JSON με μήνυμα σφάλματος: {"error": "Internal server error"}

### Use Case 12: View Map with information (mapInfo)

**Περίληψη**

Αφορά τη λειτουργία ενός διαδραστικού χάρτη για τους απλούς χρήστες (κυρίως οδηγούς), μέσω του οποίου μπορούν να δουν τους σταθμούς διοδίων της χώρας και να λάβουν βασικές πληροφορίες για αυτούς. Οι χρήστες μπορούν να εφαρμόσουν φίλτρα και να επιλέξουν σταθμούς για περισσότερες πληροφορίες.

### ****Λειτουργικές Απαιτήσεις****

1. Το σύστημα πρέπει να υποστηρίζει:
   1. Δυνατότητα χρήσης του χάρτη χωρίς login (continue as guest).
   2. Εμφάνιση όλων των σταθμών διοδίων ως pins στον χάρτη.
   3. Παροχή επιλογής φιλτραρίσματος μέσω dropdown για εμφάνιση σταθμών συγκεκριμένου operator.
   4. Εμφάνιση popup με πληροφορίες όταν ο χρήστης επιλέγει ένα pin.
2. **API Endpoints που θα χρησιμοποιηθούν**:
3. **GET /tollStations**:
   * Επιστρέφει όλους τους σταθμούς διοδίων.
4. **GET /tollStations?operatorID={operatorID}**:
   * Επιστρέφει τους σταθμούς για συγκεκριμένο operator.
5. **GET /tollStations/:stationID**:
   * Επιστρέφει λεπτομέρειες για έναν συγκεκριμένο σταθμό.
6. **GET /mapHealthCheck**:
   * Επιβεβαιώνει τη διαθεσιμότητα του χάρτη.

3) Ο χάρτης πρέπει να:

* Φορτώνεται κατά την αρχική σύνδεση του χρήστη (on page load).
* Ενημερώνεται δυναμικά αν ο χρήστης εφαρμόσει φίλτρα.

### ****Μη Λειτουργικές Απαιτήσεις****

1. **Απόδοση**: Το σύστημα πρέπει να φορτώνει τον χάρτη και τα pins εντός 2 δευτερολέπτων για όλους τους σταθμούς.
2. **Ασφάλεια**: Όλες οι κλήσεις API πρέπει να εκτελούνται μέσω HTTPS.
3. **Ανθεκτικότητα**: Το σύστημα πρέπει να εμφανίζει κατάλληλα μηνύματα σφάλματος αν αποτύχει η φόρτωση του χάρτη ή των σταθμών.

### ****Προϋποθέσεις Εκτέλεσης****

1. Ο χρήστης έχει πρόσβαση στην πλατφόρμα (internet connection).
2. Όλα τα σημεία διέλευσης έχουν φορτωθεί στη βάση δεδομένων μέσω του /admin/resetstations.

### ****Δεδομένα Εισόδου****

Προαιρετικό φίλτρο μέσω dropdown: operatorID: Το ID του operator που θα φιλτραριστεί (π.χ., "AttikiOdos").

### ****Αλληλουχία Ενεργειών / Επιθυμητή Συμπεριφορά****

1. **Είσοδος χρήστη**: Ο χρήστης επιλέγει continue as guest και συνδέεται στην πλατφόρμα.

Το frontend κάνει health check της σύνδεσης με τον χάρτη και καλεί το endpoint: GET /mapHealthCheck. Επιτυχία: Επιστροφή (200 OK) { "status": "OK" }, Αν αποτύχει, εμφανίζεται μήνυμα: "Map service unavailable. Please try again later." (500 error)

1. **Φόρτωση όλων των Σταθμών**: Το frontend καλεί το endpoint: GET /tollStations
2. Το backend καλεί το endpoint GET /tollStations για να ανακτήσει τη λίστα όλων των σταθμών.

[

{

"stationID": "NAO01",

"location": { "lat": 38.045, "lng": 23.805 },

},

{

"stationID": "AO01",

"location": { "lat": 38.002, "lng": 23.745 },

}

]

1. **Απεικόνιση Σταθμών**:
2. Τα δεδομένα που επιστρέφονται από το /tollStations εμφανίζονται ως pins στον χάρτη
3. Ο χρήστης μπορεί να αλληλεπιδράσει με τον χάρτη (zoom-in, zoom-out, drag).
4. **Φιλτράρισμα Σταθμών**:
5. Ο χρήστης επιλέγει έναν operator μέσω του dropdown (π.χ., "AttikiOdos").
6. Το frontend καλεί το endpoint GET /tollStations?operatorID={operatorID} και ανανεώνει τον χάρτη με τα pins μόνο για τους σταθμούς του συγκεκριμένου operator.

{"stationID": "AO01",

"location": { "lat": 38.002, "lng": 23.745 } }

1. **Απεικόνιση των φιλτραρισμένων σταθμών**
2. **Επιλογή Σταθμού**:
3. Ο χρήστης επιλέγει ένα pin στον χάρτη.
4. Το frontend καλεί το endpoint GET /tollStations/:stationID για να ανακτήσει τις λεπτομέρειες του σταθμού.

{

"name": "Νέα Οδός Σταθμός 1",

"operator": "NeaOdos",

"location": "Κηφισίας 123, Μαρούσι",

"tollCostCar": 2.50

"tollCostBike":1.50

"tollCostTruck": 3.50

"tollCostBus": 2.00

}

1. Εμφανίζεται popup με τις παρακάτω πληροφορίες:
   * 1. Όνομα σταθμού, Operator, Τοποθεσία (διεύθυνση), Κόστοι διέλευσης (Αμάξι, Μηχανακι, Φορτηγο, Λεωφορειο).
2. **Αντιμετώπιση Σφαλμάτων**: Αν αποτύχει η κλήση για σταθμούς (GET /tollStations), εμφανίζεται μήνυμα σφάλματος: "Failed to load toll stations. Please try again."

### ****Δεδομένα Εξόδου****

* 1. Χάρτης με pins για τους σταθμούς και Popup πληροφορίες για κάθε σταθμό
  2. Αν αποτύχουν οι κλήσεις API, εμφανίζονται κατάλληλα μηνύματα σφάλματος.

### Use Case 13: Δημιουργία ή Τροποποίηση Χρήστη μέσω CLI (usermode)

#### ****Περιγραφή****

Ο διαχειριστής δημιουργεί έναν νέο χρήστη ή αλλάζει το password ενός υπάρχοντος χρήστη μέσω CLI. Αυτό επιτρέπει την προσθήκη νέων χρηστών ή την ενημέρωση κωδικών πρόσβασης.

#### ****Λειτουργικές Απαιτήσεις****

#### Η λειτουργία να εκτελείται μέσω της εντολής: se24xx admin --usermod --username [user] --passw [newpassw]

#### Το CLI να επαληθεύει ότι οι παράμετροι --username και --passw έχουν δοθεί.

#### Επιστροφή:

#### "success" αν η λειτουργία ολοκληρωθεί επιτυχώς.

#### "error" αν προκύψει σφάλμα (π.χ., πρόβλημα στη βάση δεδομένων).

#### ****Μη Λειτουργικές Απαιτήσεις****

1. Τα passwords να αποθηκεύονται με ασφάλεια (π.χ., κρυπτογραφημένα).
2. Να υπάρχει καταγραφή (logging) επιτυχημένων και αποτυχημένων προσπαθειών.
3. Ο χρόνος εκτέλεσης να είναι μικρός (< 1 δευτερόλεπτο).

#### ****Αλληλουχία Ενεργειών****

1. Ο διαχειριστής εισάγει την εντολή CLI με τα arguments --username και --passw.
2. Το CLI: Επαληθεύει ότι οι παράμετροι είναι παρόντες. Στέλνει το αίτημα στο backend.
3. Το backend ελέγχει αν ο χρήστης υπάρχει:
4. Αν υπάρχει: Ενημερώνει το password.
5. Αν δεν υπάρχει: Δημιουργεί νέο χρήστη.
6. Επιστρέφει success ή error.
7. Το CLI εμφανίζει το αποτέλεσμα.

#### ****Προϋποθέσεις Εκτέλεσης****

1. Η βάση δεδομένων να είναι λειτουργική.
2. Ο χρήστης να έχει επαρκή δικαιώματα για τη διαχείριση χρηστών.

#### ****Δεδομένα Εισόδου****

* --username: Το όνομα του χρήστη που θα δημιουργηθεί ή τροποποιηθεί.
* --passw: Ο νέος κωδικός πρόσβασης.

### Use Case 14: Εμφάνιση Λίστας Χρηστών μέσω CLI

#### ****Περιγραφή****

Ο διαχειριστής μπορεί να δει μια λίστα με όλα τα usernames των χρηστών που είναι καταχωρημένοι στο σύστημα μέσω CLI.

#### ****Λειτουργικές Απαιτήσεις****

#### Η λειτουργία να εκτελείται μέσω της εντολής: se24xx admin --users

#### Το CLI να επιστρέφει μια λίστα usernames.

#### Αν δεν υπάρχουν χρήστες, να επιστρέφει άδεια λίστα.

#### ****Μη Λειτουργικές Απαιτήσεις****

1. Τα δεδομένα να είναι ταξινομημένα αλφαβητικά για ευκολία.
2. Να διασφαλίζεται η ακεραιότητα των δεδομένων κατά την ανάκτηση από τη βάση.
3. Ο χρόνος εκτέλεσης να είναι μικρός (< 1 δευτερόλεπτο).

#### ****Αλληλουχία Ενεργειών****

1. Ο διαχειριστής εισάγει την εντολή CLI se24xx admin --users.
2. Το CLI: Στέλνει το αίτημα στο backend.
3. Το backend: Ανακτά τη λίστα usernames από τη βάση δεδομένων. Επιστρέφει τη λίστα στο CLI.
4. Το CLI εμφανίζει τη λίστα.

#### ****Προϋποθέσεις Εκτέλεσης****

1. Ο χρήστης να έχει δικαιώματα πρόσβασης για την προβολή της λίστας.

#### ****Δεδομένα Εξόδου****

Επιτυχία:

user1

user2

user3

Αν δεν υπάρχουν χρήστες: <empty list>

### Βελτιωμένο Use Case 12: View Map with information (mapInfo)

**Περίληψη**

Αφορά τη λειτουργία ενός διαδραστικού χάρτη για τους απλούς χρήστες (κυρίως οδηγούς), μέσω του οποίου μπορούν να δουν τους σταθμούς διοδίων της χώρας και να λάβουν βασικές πληροφορίες για αυτούς. Οι χρήστες μπορούν να εφαρμόσουν φίλτρα και να επιλέξουν σταθμούς για περισσότερες πληροφορίες.

### ****Λειτουργικές Απαιτήσεις****

1. Το σύστημα πρέπει να υποστηρίζει:
   1. Δυνατότητα χρήσης του χάρτη χωρίς login (continue as guest).
   2. Εμφάνιση όλων των σταθμών διοδίων ως pins στον χάρτη.
   3. Παροχή επιλογής φιλτραρίσματος μέσω dropdown για εμφάνιση σταθμών συγκεκριμένου operator.
   4. Εμφάνιση popup με πληροφορίες όταν ο χρήστης επιλέγει ένα pin.
2. **API Endpoints που θα χρησιμοποιηθούν**:
3. **GET /mapHealthCheck**: Επιβεβαιώνει τη διαθεσιμότητα του χάρτη.

**b) GET /tollStations**: Επιστρέφει όλους τους σταθμούς διοδίων.

3. Ο χάρτης πρέπει να:

* Φορτώνεται κατά την αρχική σύνδεση του χρήστη (on page load).
* Ενημερώνεται δυναμικά αν ο χρήστης εφαρμόσει φίλτρα.

### ****Μη Λειτουργικές Απαιτήσεις****

1. **Απόδοση**: Το σύστημα πρέπει να φορτώνει τον χάρτη και τα pins εντός 2 δευτερολέπτων για όλους τους σταθμούς.
2. **Ασφάλεια**: Όλες οι κλήσεις API πρέπει να εκτελούνται μέσω HTTPS.
3. **Ανθεκτικότητα**: Το σύστημα πρέπει να εμφανίζει κατάλληλα μηνύματα σφάλματος αν αποτύχει η φόρτωση του χάρτη ή των σταθμών.

### ****Προϋποθέσεις Εκτέλεσης****

1. Ο χρήστης έχει πρόσβαση στην πλατφόρμα (internet connection).
2. Όλα τα σημεία διέλευσης έχουν φορτωθεί στη βάση δεδομένων μέσω του /admin/resetstations.

### ****Δεδομένα Εισόδου****

Προαιρετικό φίλτρο μέσω dropdown: operatorID: Το ID του operator που θα φιλτραριστεί (π.χ., "AttikiOdos").

### ****Αλληλουχία Ενεργειών / Επιθυμητή Συμπεριφορά****

1. **Είσοδος χρήστη**: Ο χρήστης επιλέγει continue as guest και συνδέεται στην πλατφόρμα.

Το frontend κάνει health check της σύνδεσης με τον χάρτη και καλεί το endpoint: GET /mapHealthCheck. Επιτυχία: Επιστροφή (200 OK) { "status": "OK" }, Αν αποτύχει, εμφανίζεται μήνυμα: "Map service unavailable. Please try again later." (500 error)

1. **Φόρτωση όλων των Σταθμών**: Το frontend καλεί το endpoint: GET /tollStations για να ανακτήσει τη λίστα όλων των σταθμών.

[ {

"stationID": "NAO01",

"name": "Νέα Οδός Σταθμός 1",

"operator": "NeaOdos",

"location": { "lat": 38.045, "lng": 23.805 },

"tollCostCar": 2.50,

"tollCostBike": 1.50,

"tollCostTruck": 3.50,

"tollCostBus": 2.00

},

{

"stationID": "AO01",

"name": "Αττική Οδός Σταθμός 1",

"operator": "AttikiOdos",

"location": { "lat": 38.002, "lng": 23.745 },

"tollCostCar": 2.55,

"tollCostBike": 1.70,

"tollCostTruck": 3.00,

"tollCostBus": 2.00

} ]

1. **Απεικόνιση Σταθμών**:
   1. Οι τοποθεσίες που επιστρέφονται από το /tollStations εμφανίζονται ως pins στον χάρτη

allStations.forEach(station => {

addPinToMap({

id: station.stationID,

position: station.location // Χρησιμοποιείς μόνο τη θέση

});

});

* 1. Ο χρήστης μπορεί να αλληλεπιδράσει με τον χάρτη (zoom-in, zoom-out, drag).

1. **Φιλτράρισμα Σταθμών**:
   1. Ο χρήστης επιλέγει έναν operator μέσω του dropdown (π.χ., "AttikiOdos").
      1. const operators = [...new Set(allStations.map(station => station.operator))];

// Αφαιρείς τα διπλότυπα operators για το dropdown

* 1. Το frontend φιλτράρει τα δεδομένα που έχουν ήδη ληφθεί από την αρχική κλήση.
     1. const filteredStations = allStations.filter(station => station.operator === "AttikiOdos");

1. **Απεικόνιση των φιλτραρισμένων σταθμών**
2. **Επιλογή Σταθμού**:
   1. Ο χρήστης επιλέγει ένα pin στον χάρτη.
   2. το frontend αναζητά τα δεδομένα του σταθμού στη λίστα που έχει ήδη φορτωθεί.
   3. Εμφανίζεται popup με τις παρακάτω πληροφορίες:

const selectedStation = allStations.find(station => station.stationID === "NAO01");

showPopup({

name: selectedStation.name,

operator: selectedStation.operator,

tollCost: selectedStation.tollCost

});

1. **Αντιμετώπιση Σφαλμάτων**: Αν αποτύχει η κλήση για σταθμούς (GET /tollStations), εμφανίζεται μήνυμα σφάλματος: "Failed to load toll stations. Please try again."

### ****Δεδομένα Εξόδου****

1. Χάρτης με pins για τους σταθμούς και Popup πληροφορίες για κάθε σταθμό
2. Αν αποτύχουν οι κλήσεις API, εμφανίζονται κατάλληλα μηνύματα σφάλματος.