ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή Εργασία

Τίτλος Πτυχιακής Εργασίας	Πλατφόρμα Ψηφιακή Πόλη Digital City Platform
Ονοματεπώνυμο Φοιτητή	Παγίδας Νικόλαος
Πατρώνυμο	Παγίδας Χρήστος
Αριθμός Μητρώου	Π19127
Επιβλέπων	Αλέπης Ευθύμιος , Καθηγητής

Ημερομηνία Παράδοσης **Ιούλιος 2024**

Copyright ©

ΠΕΡΙΛΗΨΗ

Η πτυχιακή αυτή εργασία αναπτύχθηκε για να προσφέρει μια ολοκληρωμένη πλατφόρμα "Ψηφιακής Πόλης", στοχεύοντας στη βελτίωση της αλληλεπίδρασης των πολιτών με τις δημοτικές υπηρεσίες και την αποδοτικότητα των αστικών υποδομών. Η εφαρμογή αυτή ενσωματώνει τεχνολογίες όπως η προσαρμοστική διαχείριση φωτισμού, η διαχείριση κυκλοφορίας και απορριμμάτων, και η ψηφιακή διαχείριση εκδηλώσεων και πληρωμών, χρησιμοποιώντας δεδομένα σε πραγματικό χρόνο. Η ανάπτυξη της πλατφόρμας βασίστηκε σε μια πολυεπίπεδη αρχιτεκτονική που διαχωρίζει σαφώς τις διάφορες λειτουργικές μονάδες, επιτυγχάνοντας ευελιξία, επεκτασιμότητα και εύκολη συντήρηση. Το έργο αυτό υλοποιήθηκε κατά τη διάρκεια του ακαδημαϊκού έτους 2023-2024, υπό την επίβλεψη του καθηγητή κ. Ευθύμιου Αλέπη.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Πληροφορική

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Ψηφιακή Πόλη, Διαχείριση Κυκλοφορίας, Έξυπνος Φωτισμός, Ψηφιακές Πληρωμές

ABSTRACT

This thesis was developed to offer a comprehensive "Smart City" platform, aiming to improve citizen interaction with municipal services and the efficiency of urban infrastructure. This application integrates technologies such as adaptive lighting management, traffic and waste management, and digital management of events and payments, using real-time data. The platform's development was based on a multi-layered architecture that clearly separates various functional units, achieving flexibility, scalability, and easy maintenance. This project was implemented during the academic year 2023-2024, under the supervision of Professor Mr. Efthymios Alepis.

SUBJECT AREA: Information Technology

KEYWORDS: Digital City, Traffic Management, Smart Lighting, Digital Payments

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να εκφράσω τις ειλικρινείς μου ευχαριστίες σε όλους όσοι συνέβαλαν στην ολοκλήρωση της παρούσας πτυχιακής εργασίας. Ιδιαίτερα, ευχαριστώ τον επιβλέποντα καθηγητή κ. Ευθύμιο Αλέπη για την καθοδήγηση και την υποστήριξή του. Επίσης, ευχαριστώ την οικογένειά μου και τους φίλους μου για την συνεχή ενθάρρυνση και βοήθειά τους.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΔΙΑΓΡΑΜΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΠΡΟΛΟΓΟΣ

Στη σύγχρονη εποχή, οι πόλεις μας αντιμετωπίζουν αυξανόμενες προκλήσεις, όπως η διαχείριση της κυκλοφορίας, η αποδοτική χρήση της ενέργειας και η αποχή των πολιτών από τα κοινά. Η πτυχιακή αυτή εργασία αναπτύχθηκε με στόχο να αντιμετωπίσει αυτά τα προβλήματα, προσφέροντας μια ολοκληρωμένη πλατφόρμα «Ψηφιακής Πόλης».

Η πλατφόρμα αυτή στοχεύει στη βελτίωση της αλληλεπίδρασης των πολιτών με τις δημοτικές υπηρεσίες και στην αύξηση της αποδοτικότητας των αστικών υποδομών. Μέσω της εφαρμογής έξυπνων τεχνολογιών, όπως η προσαρμοστική διαχείριση του φωτισμού με βάση τα πραγματικά δεδομένα καιρού και η έξυπνη διαχείριση της κυκλοφορίας και των απορριμμάτων, η πλατφόρμα προσφέρει λύσεις που ενισχύουν την ενεργειακή αποδοτικότητα και την ποιότητα ζωής.

Η εργασία αυτή υλοποιήθηκε κατά τη διάρκεια του ακαδημαϊκού έτους 2023-2024 στο πλαίσιο του Τμήματος Πληροφορικής του Πανεπιστημίου Πειραιώς, υπό την επίβλεψη του καθηγητή κ. Ευθύμιου Αλέπη. Θερμές ευχαριστίες απευθύνονται στον επιβλέποντα καθηγητή για την καθοδήγηση και υποστήριξή του, καθώς και στην οικογένεια και τους φίλους για την ενθάρρυνση και τη βοήθειά τους.

1. ΕΙΣΑΓΩΓΗ

1.1 Πλαίσιο Εργασίας

Η παρούσα πτυχιακή εργασία παρουσιάζει την ανάπτυξη μιας πλατφόρμας ψηφιακής πόλης που στοχεύει στη βελτίωση της αλληλεπίδρασης των πολιτών με τις δημοτικές υπηρεσίες και στην αύξηση της αποδοτικότητας των αστικών υποδομών.

1.2 Δήλωση Προβλήματος

Παρά τις προόδους στην αστική τεχνολογία, πολλές πόλεις εξακολουθούν να αντιμετωπίζουν προβλήματα όπως η αναποτελεσματική δημόσια φωταγώγηση. Για παράδειγμα, σε πολλές πόλεις, τα φώτα στους δρόμους παραμένουν αναμμένα όλη τη νύχτα, ακόμα και όταν δεν υπάρχει κίνηση, σπαταλώντας ενέργεια και αυξάνοντας το κόστος. Παράλληλα, η κυκλοφοριακή συμφόρηση και η κακή διαχείριση συλλογής απορριμμάτων παραμένουν σημαντικά ζητήματα. Τα προβλήματα αυτά επηρεάζουν όχι μόνο την αποδοτικότητα της πόλης, αλλά και την ποιότητα ζωής των κατοίκων. Τα παραδοσιακά συστήματα συχνά δεν μπορούν να ανταποκριθούν στις δυναμικές αλλαγές και απαιτήσεις των αστικών περιβαλλόντων

1.3 Στόχοι Εργασίας:

Ο κύριος στόχος αυτής της εργασίας, μέσω της ανάπτυξης της εφαρμογής "Ψηφιακή Πόλη", είναι η αντιμετώπιση αυτών των αστικών προκλήσεων μέσω:

1. Βελτίωσης της αποδοτικότητας της δημόσιας φωταγώγησης μέσω προσαρμοστικών ελέγχων που ρυθμίζονται με βάση τις πραγματικές καιρικές συνθήκες.
2. Βελτίωσης της ροής της κυκλοφορίας και της διαχείρισης των χώρων στάθμευσης με δυναμικά δεδομένα και συστήματα κράτησης.
3. Βελτιστοποίησης των διαδρομών και των προγραμμάτων συλλογής απορριμμάτων χρησιμοποιώντας τον αλγόριθμο PSO (Particle Swarm Optimization) με βάση τα δυναμικά δεδομένα της κυκλοφορίας και τα επίπεδα των κάδων απορριμμάτων.
4. Διευκόλυνση της άμεσης αλληλεπίδρασης μεταξύ των πολιτών και των δημοτικών αρχών μέσω πλατφόρμας για σχόλια και συμμετοχή στη δημοτική διακυβέρνηση.

5. Ενεργοποίησης ψηφιακών συναλλαγών για δημοτικές πληρωμές, αυξάνοντας την ευκολία για τους κατοίκους.

1.4 Σημασία

Η πλατφόρμα "Ψηφιακή Πόλη" ξεχωρίζει επειδή ενσωματώνει διάφορες πτυχές της διαχείρισης της πόλης σε μία ενιαία εφαρμογή και χρησιμοποιεί δεδομένα σε πραγματικό χρόνο. Αυτή η εργασία δείχνει πώς οι ψηφιακές λύσεις μπορούν να βελτιώσουν την αστική ζωή, κάνοντας τις πόλεις πιο αποδοτικές, βιώσιμες και συμπεριληπτικές. Τα ευρήματα και οι τεχνολογίες που εφαρμόστηκαν σε αυτή τη διατριβή μπορούν να χρησιμεύσουν ως πρότυπο για άλλες πόλεις που θέλουν να εφαρμόσουν έξυπνες τεχνολογίες, συμβάλλοντας στον ευρύτερο τομέα των αστικών μελετών και των εφαρμογών έξυπνης πόλης.

2. Σχεδιασμός του Συστήματος

2.1 Αρχιτεκτονική του Συστήματος

Η ανάπτυξη της εφαρμογής βασίστηκε σε μια πολυεπίπεδη αρχιτεκτονική που διαχωρίζει σαφώς τις διάφορες λειτουργικές μονάδες, επιτυγχάνοντας ευελιξία, επεκτασιμότητα και εύκολη συντήρηση. Κάθε επίπεδο της αρχιτεκτονικής έχει συγκεκριμένο ρόλο και ευθύνες, διασφαλίζοντας την αποτελεσματική και αποδοτική λειτουργία της εφαρμογής.

Παρουσιαστικό Επίπεδο (Presentation Layer):

Τεχνολογίες: Angular 16.2.14

Ρόλος: Το παρουσιαστικό επίπεδο είναι υπεύθυνο για τη διεπαφή χρήστη (UI) και την επικοινωνία με το backend μέσω HTTP αιτήσεων. Παρέχει την οπτική και διαδραστική εμπειρία στους χρήστες και είναι υπεύθυνο για τη διαχείριση των χρηστών, τη διαχείριση των εκδηλώσεων και την παροχή ενημερώσεων σε πραγματικό χρόνο.

Μερικες Λειτουργίες:

1. Διαχείριση χρηστών: Σύνδεση, εγγραφή, και αποσύνδεση.
2. Διαχείριση εκδηλώσεων: Δημιουργία, προβολή, και επεξεργασία εκδηλώσεων.
3. Ενημερώσεις σε πραγματικό χρόνο: Παροχή πληροφοριών κυκλοφορίας και αλλαγών σε εκδηλώσεις.

Επίπεδο Επιχειρησιακής Λογικής (Business Logic Layer):

Τεχνολογίες: ASP.NET Core

Ρόλος: Το επίπεδο επιχειρησιακής λογικής χειρίζεται την επεξεργασία αιτήσεων από το παρουσιαστικό επίπεδο, εφαρμόζει τους κανόνες επιχειρησιακής λογικής και επικοινωνεί με το επίπεδο πρόσβασης δεδομένων. Αποτελεί τον κεντρικό πυρήνα της εφαρμογής, όπου πραγματοποιείται η κύρια επεξεργασία και διαχείριση δεδομένων.

Λειτουργικότητες:

1. Αυθεντικοποίηση και εξουσιοδότηση χρηστών μέσω JWT tokens.
2. Διαχείριση εκδηλώσεων: Δημιουργία, τροποποίηση εκδηλώσεων από εξουσιοδοτημένους χρήστες και προβολή εκδηλώσεων για τους κανονικούς χρήστες.
3. Διαχείριση κυκλοφορίας και απορριμμάτων: Ενημέρωση δεδομένων κυκλοφορίας και απορριμμάτων.
4. Διαχείριση πληρωμών: Επεξεργασία πληρωμών και διαχείριση τιμολογίων.

Επίπεδο Πρόσβασης Δεδομένων (Data Access Layer):

Τεχνολογίες: Entity Framework Core

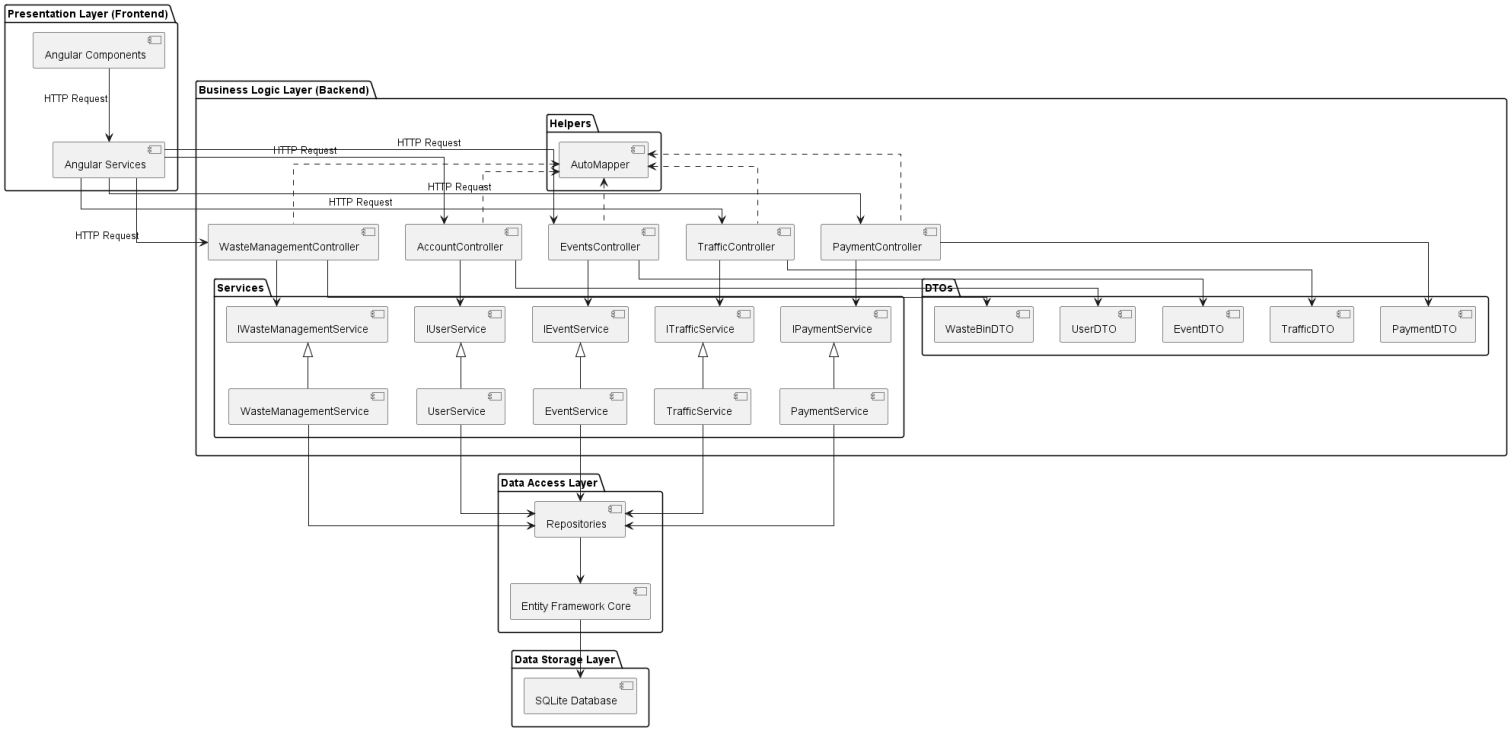
Ρόλος: Το επίπεδο πρόσβασης δεδομένων είναι υπεύθυνο για τη διαχείριση της επικοινωνίας με τη βάση δεδομένων. Παρέχει ένα στρώμα αφαιρετικότητας για την εκτέλεση CRUD (Create, Read, Update, Delete) λειτουργιών και διασφαλίζει την αποδοτική και ασφαλή αλληλεπίδραση με τα δεδομένα.

Επίπεδο Αποθήκευσης Δεδομένων (Data Storage Layer):

Τεχνολογίες: SQLite

Ρόλος: Το επίπεδο αποθήκευσης δεδομένων διαχειρίζεται την αποθήκευση όλων των δεδομένων της εφαρμογής. Χρησιμοποιεί το SQLite για την αποθήκευση δεδομένων χρηστών, εκδηλώσεων, πληρωμών και άλλων δεδομένων εφαρμογής.

Η σχεδίαση του συστήματος διασφαλίζει τη σαφή διαχωρισμό των λειτουργιών, την ευελιξία στην ανάπτυξη και την ευκολία συντήρησης. Ακολουθεί διάγραμμα (Διαγραμμα.1) που απεικονίζει την συνολική αρχιτεκτονική του συστήματος και τις σχέσεις μεταξύ των διαφόρων επιπέδων.



Διάγραμμα 1: Architecture Diagram

2.2 Σχεδιασμός της Βάσης Δεδομένων

Ο σχεδιασμός της βάσης δεδομένων είναι θεμελιώδους σημασίας για τη λειτουργία της εφαρμογής. Η βάση δεδομένων σχεδιάστηκε χρησιμοποιώντας το SQLite και περιλαμβάνει τους εξής πίνακες:

AspNetUsers:

Περιγραφή: Αποθηκεύει τα στοιχεία των χρηστών.

Πεδία:

- Id: Μοναδικό αναγνωριστικό χρήστη (PRIMARY KEY).
- UserName: Όνομα χρήστη.
- NormalizedUserName: Κανονικοποιημένο όνομα χρήστη.
- Email: Διεύθυνση email.
- NormalizedEmail: Κανονικοποιημένη διεύθυνση email.
- EmailConfirmed: Επιβεβαίωση διεύθυνσης email.
- PasswordHash: Κρυπτογραφημένο συνθηματικό.
- PhoneNumber: Αριθμός τηλεφώνου.
- PhoneNumberConfirmed: Επιβεβαίωση αριθμού τηλεφώνου.
- FirstName: Όνομα χρήστη.
- LastName: Επίθετο χρήστη.
- DateOfBirth: Ημερομηνία γέννησης.

- Created: Ημερομηνία δημιουργίας λογαριασμού.

Events:

Περιγραφή: Αποθηκεύει τις πληροφορίες των εκδηλώσεων.

Πεδία:

- Id: Μοναδικό αναγνωριστικό εκδήλωσης (PRIMARY KEY).
- Title: Τίτλος εκδήλωσης.
- Description: Περιγραφή εκδήλωσης.
- PhotoUrl: URL φωτογραφίας της εκδήλωσης.
- Date: Ημερομηνία εκδήλωσης.
- InterestedUsers: Χρήστες που ενδιαφέρονται για την εκδήλωση.

TrafficData:

Περιγραφή: Αποθηκεύει δεδομένα κυκλοφορίας.

Πεδία:

- Id: Μοναδικό αναγνωριστικό κυκλοφορίας (PRIMARY KEY).
- Location: Τοποθεσία.
- TrafficFlow: Ροή κυκλοφορίας.
- Timestamp: Χρονική σήμανση.

WasteBins:

Περιγραφή: Αποθηκεύει τα επίπεδα απορριμμάτων και τις πληροφορίες των κάδων.

Πεδία:

- Id: Μοναδικό αναγνωριστικό κάδου (PRIMARY KEY).
- Location: Τοποθεσία.
- Capacity: Χωρητικότητα.
- CurrentFillLevel: Τρέχον επίπεδο πλήρωσης.
- LastEmptied: Τελευταία φορά άδειασμα.

Invoices:

Περιγραφή: Αποθηκεύει τις πληροφορίες τιμολογίων.

Πεδία:

- Id: Μοναδικό αναγνωριστικό τιμολογίου (PRIMARY KEY).

- InvoiceNumber: Αριθμός τιμολογίου.
- Amount: Ποσό.
- UserId: Αναγνωριστικό χρήστη.
- IsPaid: Κατάσταση πληρωμής

Reservations:

Περιγραφή: Αποθηκεύει τις πληροφορίες κρατήσεων χώρων στάθμευσης.

Πεδία:

- Id: Μοναδικό αναγνωριστικό κράτησης (PRIMARY KEY).
- UserId: Αναγνωριστικό χρήστη.
- ParkingSpaceId: Αναγνωριστικό χώρου στάθμευσης.
- ReservationTime: Ώρα κράτησης.
- Duration: Διάρκεια κράτησης.

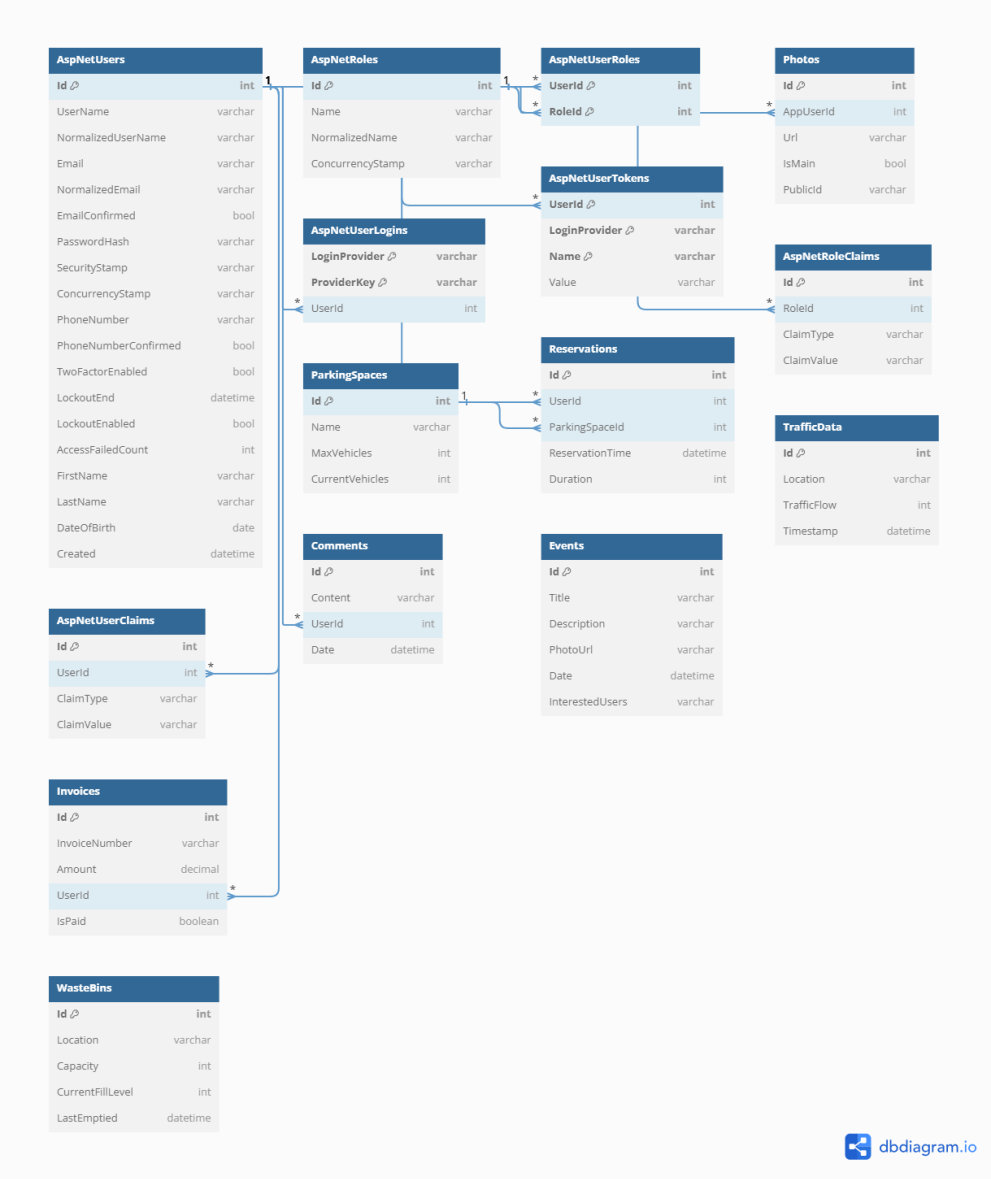
Comments:

Περιγραφή: Αποθηκεύει τα σχόλια των χρηστών.

Πεδία:

- Id: Μοναδικό αναγνωριστικό σχολίου (PRIMARY KEY).
- Content: Περιεχόμενο σχολίου.
- UserId: Αναγνωριστικό χρήστη.
- Date: Ημερομηνία σχολίου.

Το σχήμα της βάσης δεδομένων σχεδιάστηκε με στόχο τη βελτιστοποίηση της ανάκτησης και της αποθήκευσης των δεδομένων, ενώ οι σχέσεις μεταξύ των πινάκων εξασφαλίζουν την ακεραιότητα των δεδομένων.



Εικόνα 1: Βάση Δεδομένων Smart-Town

2.3 Σχεδιασμός του Frontend

Ο σχεδιασμός του frontend επικεντρώνεται στη δημιουργία μιας δυναμικής και αποκριτικής διεπαφής χρήστη χρησιμοποιώντας τη Node.Js με Angular 16.2.14. Η αρχιτεκτονική του frontend περιλαμβάνει την εξής δομή:

- Components: Χωρίζονται σε επιμέρους τμήματα της εφαρμογής, όπως LoginComponent, RegisterComponent, EventListComponent και TrafficComponent. Αυτά τα components είναι υπεύθυνα για την προβολή και την αλληλεπίδραση του χρήστη με την εφαρμογή.

- Services: Χρησιμοποιούνται για την επικοινωνία με το backend και την επεξεργασία των δεδομένων. Κάθε service είναι υπεύθυνο για μια συγκεκριμένη λειτουργικότητα, όπως το AccountService, EventService,

και PaymentService. Αυτά τα services διασφαλίζουν ότι η επιχειρησιακή λογική διαχωρίζεται από τη διεπαφή χρήστη.

- Models: Τα μοντέλα (models) είναι τύποι δεδομένων TypeScript που καθορίζουν τη δομή των δεδομένων που χρησιμοποιούνται στην εφαρμογή, όπως User, Event, και Invoice. Αυτά τα μοντέλα βοηθούν στη διατήρηση της συνέπειας μεταξύ του frontend και του backend.

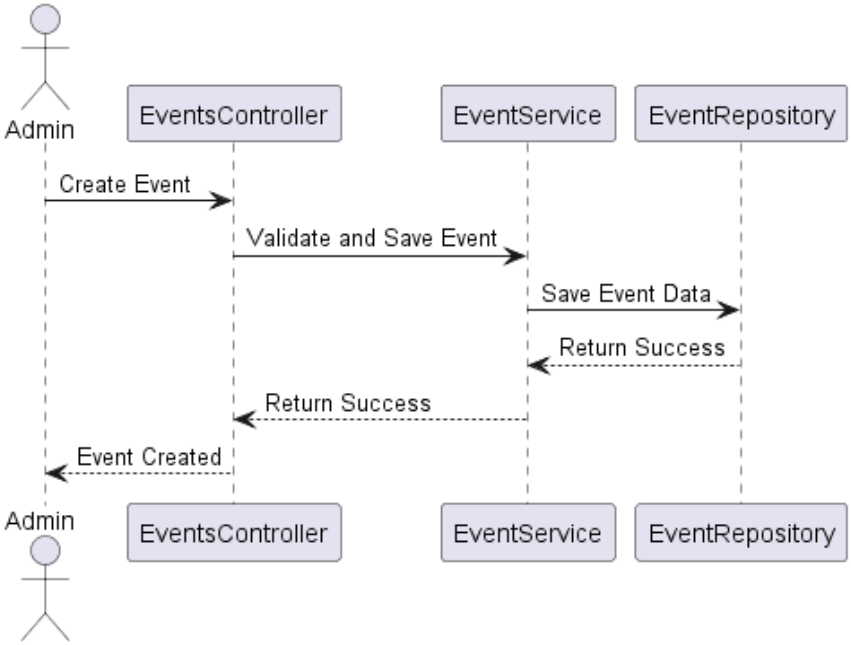
- Authentication Guards: Χρησιμοποιούνται για την προστασία συγκεκριμένων διαδρομών της εφαρμογής, επιτρέποντας την πρόσβαση μόνο σε εξουσιοδοτημένους χρήστες. Για παράδειγμα, ο AuthGuard διασφαλίζει ότι μόνο οι συνδεδεμένοι χρήστες μπορούν να έχουν πρόσβαση σε προστατευμένες διαδρομές.

- Interceptors: Χρησιμοποιούνται για την παρεμβολή σε HTTP αιτήσεις και αποκρίσεις, προσθέτοντας κοινή λογική όπως η διαχείριση των JWT tokens.

- Navigation Bar: Η γραμμή πλοήγησης (navbar) παρέχει εύκολη πλοήγηση μεταξύ των διαφόρων τμημάτων της εφαρμογής. Περιλαμβάνει συνδέσμους για τις κύριες λειτουργίες όπως η εγγραφή, η σύνδεση, η προβολή εκδηλώσεων και άλλα.

- Admin Page: Παρέχει σελίδα διαχειριστή για τη διαχείριση χρηστών, εκδηλώσεων και άλλων σημαντικών λειτουργιών της εφαρμογής. Η σελίδα διαχειριστή είναι προσβάσιμη μόνο σε χρήστες με το ρόλο διαχειριστή.

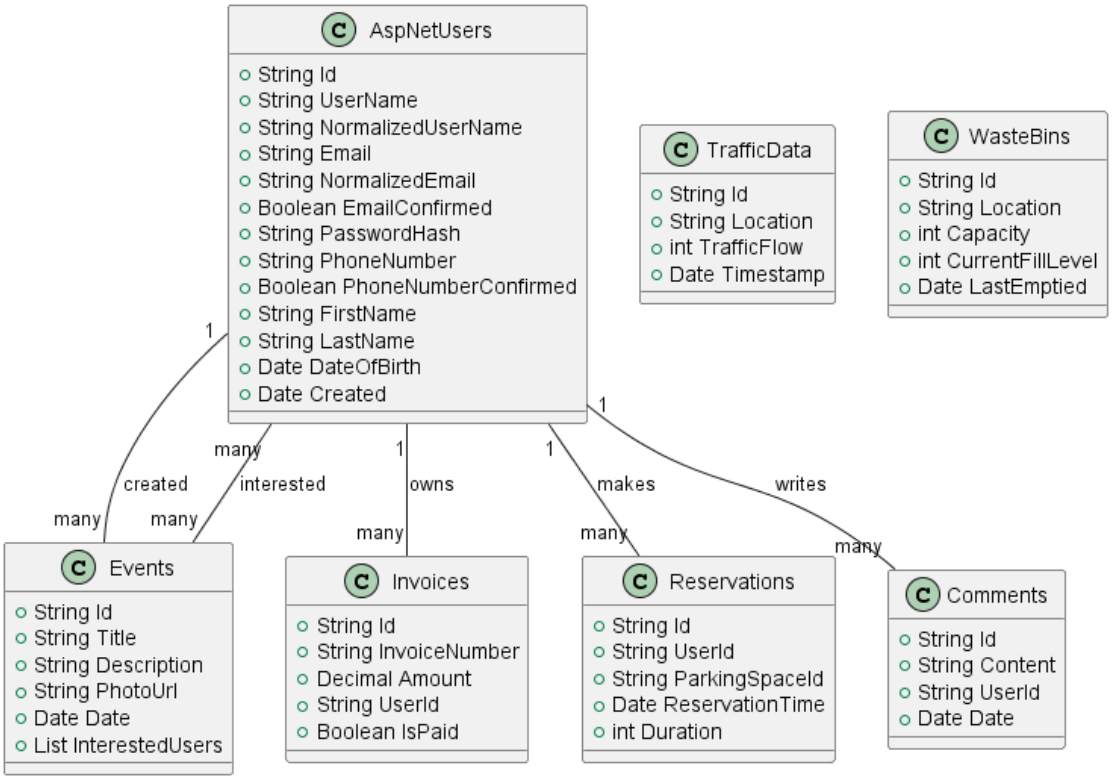
- Styles: Η εφαρμογή χρησιμοποιεί συγκεκριμένο στυλ για τη βελτίωση της εμφάνισης της σελίδας. Περιλαμβάνει αρχεία CSS από βιβλιοθήκες όπως Bootstrap και FontAwesome, καθώς και προσαρμοσμένα στυλ.



Διάγραμμα 2: Sequence Diagram for Admins

2.4 Σχεδιασμός του Backend

Ο σχεδιασμός του backend επικεντρώνεται στη δημιουργία ενός αποδοτικού και ασφαλούς συστήματος που διαχειρίζεται τις επιχειρησιακές λογικές και την αποθήκευση των δεδομένων. Η αρχιτεκτονική του backend περιλαμβάνει:



Διάγραμμα 3: Class Diagram

Controllers: Οι controllers στο ASP.NET Core είναι υπεύθυνοι για τη διαχείριση των αιτήσεων HTTP από το frontend. Καλούν τα αντίστοιχα services για να εκτελέσουν τις απαιτούμενες ενέργειες και να επιστρέψουν τα αποτελέσματα στο frontend. Παραδείγματα controllers είναι:

1. AccountController: Διαχειρίζεται τη σύνδεση, την εγγραφή και τις ενημερώσεις προφίλ των χρηστών.
2. EventsController: Χειρίζεται τη δημιουργία, την επεξεργασία και την προβολή λεπτομερειών εκδηλώσεων.
3. TrafficController: Παρέχει δεδομένα κυκλοφορίας σε πραγματικό χρόνο.
4. PaymentController: Διαχειρίζεται τις πληρωμές και την επεξεργασία τιμολογίων.

Services: Τα services υλοποιούν την επιχειρησιακή λογική της εφαρμογής. Επικοινωνούν με τα repositories για την ανάκτηση και αποθήκευση δεδομένων και εκτελούν τις βασικές λειτουργίες της εφαρμογής. Κάθε service είναι υπεύθυνο για συγκεκριμένες λειτουργικότητες, όπως:

1. UserService: Διαχειρίζεται τις λειτουργίες που αφορούν τους χρήστες, όπως η εγγραφή και η ενημέρωση προφίλ.
2. EventService: Υλοποιεί τη λογική για τη διαχείριση εκδηλώσεων, συμπεριλαμβανομένης της δημιουργίας, επεξεργασίας και διαγραφής εκδηλώσεων.
3. TrafficService: Διαχειρίζεται τα δεδομένα κυκλοφορίας και την ενημέρωση των δεδομένων αυτών.
4. PaymentService: Διαχειρίζεται τις πληρωμές, συμπεριλαμβανομένης της δημιουργίας και της επεξεργασίας τιμολογίων.

Repositories: Τα repositories παρέχουν ένα επίπεδο αφαιρετικότητας μεταξύ των services και της βάσης δεδομένων. Υλοποιούν τις λειτουργίες CRUD (Create, Read, Update, Delete) για τις διάφορες οντότητες της εφαρμογής. Παραδείγματα repositories είναι:

UserRepository: Διαχειρίζεται τα ερωτήματα και τις συναλλαγές που αφορούν τα δεδομένα των χρηστών.

Data Transfer Objects (DTOs): Χρησιμοποιούνται για τη μεταφορά δεδομένων μεταξύ του frontend και του backend. Τα DTO's διασφαλίζουν ότι μόνο τα απαραίτητα δεδομένα μεταφέρονται, μειώνοντας την ποσότητα των δεδομένων που μεταφέρονται και αυξάνοντας την ασφάλεια. Παραδείγματα DTO's είναι:

- MemberDTO: Χρησιμοποιείται για τη μεταφορά δεδομένων χρήστη μεταξύ του frontend και του backend.
- EventDTO: Περιέχει πληροφορίες σχετικά με τις εκδηλώσεις που μεταφέρονται μεταξύ των services και του frontend.
- PaymentDTO: Χρησιμοποιείται για την επεξεργασία και τη μεταφορά δεδομένων πληρωμών.

Error Handling: Το backend περιλαμβάνει μηχανισμούς διαχείρισης εξαιρέσεων και επιστρέφει κατάλληλους κωδικούς κατάστασης HTTP και μηνύματα σφάλματος όταν προκύπτουν εξαιρέσεις. Αυτό εξασφαλίζει ότι το frontend μπορεί να διαχειριστεί τα σφάλματα με σωστό τρόπο και να παρέχει χρήσιμη ανατροφοδότηση στον χρήστη.

Authentication and Authorization: Το σύστημα χρησιμοποιεί JWT (JSON Web Tokens) για την αυθεντικοποίηση και την εξουσιοδότηση. Οι χρήστες μπορούν να εγγραφούν, να συνδεθούν και να αποκτήσουν ένα token που χρησιμοποιείται για την αυθεντικοποίηση επόμενων αιτήσεων. Το σύστημα υποστηρίζει επίσης εξουσιοδότηση βάσει ρόλων, επιτρέποντας διαφορετικά επίπεδα πρόσβασης σε πόρους βάσει των ρόλων των χρηστών.

3 ΑΝΑΠΤΥΞΗ ΚΑΙ ΕΦΑΡΜΟΓΗ

3.1 Διαδικασίες και Εργαλεία Ανάπτυξης

Για την ανάπτυξη της εφαρμογής "Ψηφιακή Πόλη", χρησιμοποιήθηκαν διάφορες διαδικασίες και εργαλεία ανάπτυξης για να διασφαλιστεί η αποτελεσματικότητα, η συνεργασία και η ποιότητα του κώδικα. Οι κύριες διαδικασίες και τα εργαλεία που χρησιμοποιήθηκαν περιγράφονται παρακάτω:

Διαδικασίες Ανάπτυξης:

Agile Μεθοδολογία:

Sprint Planning: Καθορισμός των εργασιών και στόχων για κάθε sprint.

Daily Stand-ups: Καθημερινές σύντομες συναντήσεις για να διασφαλιστεί ότι η ομάδα είναι συγχρονισμένη.

Sprint Reviews: Αξιολόγηση της προόδου στο τέλος κάθε sprint και επίδειξη νέων λειτουργιών.

Sprint Retrospectives: Ανασκόπηση της διαδικασίας για να αναγνωριστούν βελτιώσεις.

Εργαλεία Ανάπτυξης:

1. Visual Studio Code (VS Code):
    - Χρησιμοποιήθηκε ως το κύριο περιβάλλον ανάπτυξης (IDE) για την επεξεργασία και τη διαχείριση του κώδικα.
    - Επεκτάσεις: .NET Install Tool: Για την εύκολη διαχείριση των εκδόσεων του .NET, C# DevKit: Παρέχει υποστήριξη για την ανάπτυξη C# κώδικα, Angular Language Service: Για την ανάπτυξη και υποστήριξη Angular εφαρμογών, NuGet Gallery: Για τη διαχείριση πακέτων και εξαρτήσεων, SQLite: Για την προβολή και διαχείριση της βάσης δεδομένων SQLite.
2. Git:

- Χρησιμοποιήθηκε για τον έλεγχο εκδόσεων και την παρακολούθηση των αλλαγών στον κώδικα.

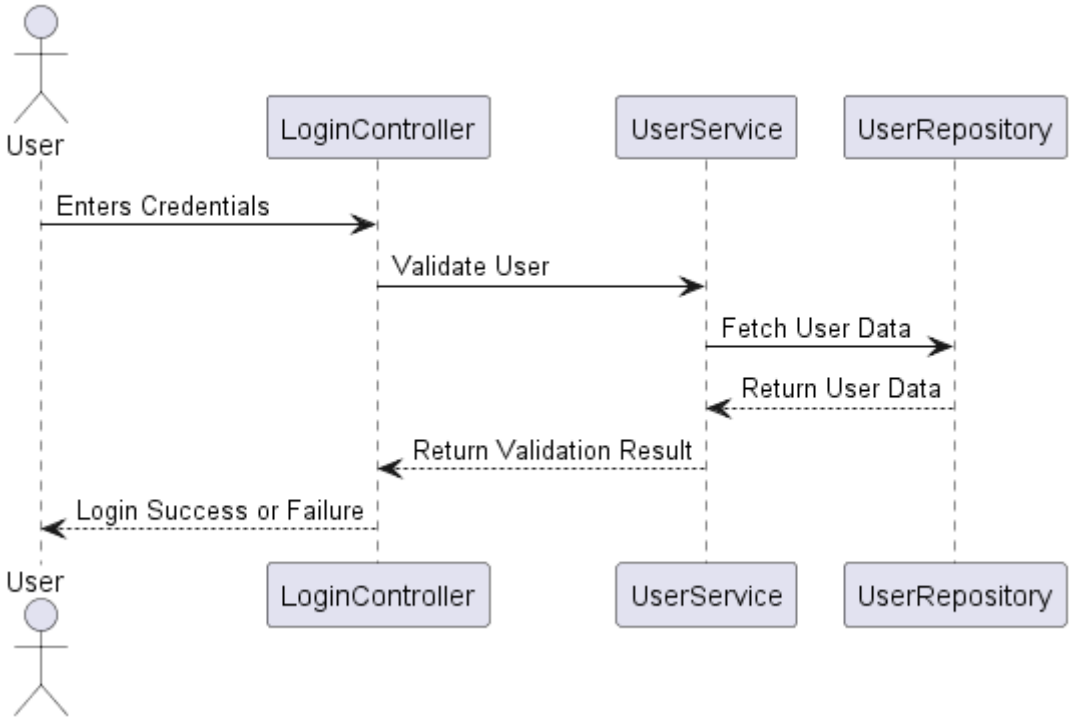
3. Postman:
    - Χρησιμοποιήθηκε για τη δοκιμή και την επικύρωση των API endpoints.
    - Επιτρέπει τη δημιουργία και την εκτέλεση αιτήσεων HTTP, διευκολύνοντας την ανίχνευση και την επίλυση σφαλμάτων στον κώδικα του backend.
4. Entity Framework Core:
    - Χρησιμοποιήθηκε για την αλληλεπίδραση με τη βάση δεδομένων.
    - Παρέχει δυνατότητες ORM (Object-Relational Mapping), διευκολύνοντας τη διαχείριση των δεδομένων και τη δημιουργία migrations για την ενημέρωση του σχήματος της βάσης δεδομένων.
5. Angular CLI:
    - Χρησιμοποιήθηκε για τη δημιουργία και τη διαχείριση του frontend.
    - Παρέχει εντολές για τη δημιουργία νέων components, services και modules, καθώς και για τη διαχείριση των dependencies του έργου.

3.2 Ανάπτυξη του Frontend

Το frontend αποτελεί ένα κρίσιμο κομμάτι της εφαρμογής, καθώς είναι υπεύθυνο για την αλληλεπίδραση με τους χρήστες. Η ανάπτυξη επικεντρώθηκε στην παροχή μιας ευχάριστης και λειτουργικής εμπειρίας χρήστη.

Απαιτήσεις και Έκταση του Έργου

Οι κύριοι στόχοι του frontend ήταν να παρέχει δυνατότητες σύνδεσης και εγγραφής χρηστών, διαχείρισης εκδηλώσεων, παρακολούθησης κυκλοφορίας και πληρωμών. Η έκταση του έργου περιλάμβανε την ανάπτυξη μιας single-page application (SPA) που θα λειτουργούσε άψογα σε διάφορους browsers και συσκευές. Ένα σημαντικός επιχειρησιακός κανόνας που επηρέασε την ανάπτυξη ήταν η ανάγκη για ασφαλή πρόσβαση με βάση τους ρόλους των χρηστών (μέλη ή διαχειριστές).



Διάγραμμα 4: Sequence Diagram for Users

Σχεδιασμός και Εμπειρία Χρήστη

Η διαδικασία σχεδιασμού περιλάμβανε τα εξής βήματα:

Δημιουργία Wireframes και Mockups:

Αρχικά δημιουργήθηκαν wireframes και mockups για να απεικονιστεί η διάταξη και η δομή κάθε σελίδας της εφαρμογής. Αυτά τα εργαλεία βοήθησαν τους προγραμματιστές να κατανοήσουν την επιθυμητή εμφάνιση και αίσθηση, καθώς και την εμπειρία χρήστη που θέλαμε να επιτύχουμε.

Δημιουργία Διαδραστικών Πρωτοτύπων:

Δημιουργήθηκαν διαδραστικά πρωτότυπα για να επιδείξουν τις επιθυμητές αλληλεπιδράσεις και κινούμενα σχέδια των χρηστών. Αυτό βοήθησε τους προγραμματιστές να κατανοήσουν καλύτερα την προβλεπόμενη συμπεριφορά των διαφόρων στοιχείων της εφαρμογής.

Διασφάλιση Φιλικής προς τον Χρήστη Διεπαφής:

Χρησιμοποιήσαμε το "Flatly" theme του Bootstrap για να διατηρήσουμε μια συνεπή και ελκυστική εμφάνιση σε όλη την εφαρμογή. Το Flatly theme προσφέρει μια καθαρή και μοντέρνα αισθητική που βελτιώνει την εμπειρία χρήστη.

Τεχνολογίες και Εργαλεία

Framework:

Χρησιμοποιήσαμε τη Node.Js (Angular version 16.2.14), ένα σύγχρονο framework για την ανάπτυξη single-page applications (SPA). Επιλέξαμε την Angular λόγω των ισχυρών δυνατοτήτων του για τη διαχείριση πολύπλοκων εφαρμογών, της ευελιξίας και της υποστήριξής του για δύο κατευθύνσεων δέσμευση δεδομένων (two-way data binding).

Εργαλεία και Βιβλιοθήκες:

- Bootstrap: Χρησιμοποιήθηκε για την ανάπτυξη ενός αποκριτικού και ευέλικτου layout.
- FontAwesome: Ενσωματώθηκε για την παροχή εικονιδίων και γραφικών στοιχείων.
- Postman: Χρησιμοποιήθηκε για τη δοκιμή των διαφόρων endpoints του backend.

Διαδικασία Ανάπτυξης

Υλοποίηση Σχεδίου και Διάταξης:

Ο σχεδιασμός και η διάταξη υλοποιήθηκαν σύμφωνα με τα wireframes και τα mockups που δημιουργήθηκαν κατά τη φάση του σχεδιασμού. Κάθε component της Angular αναπτύχθηκε με βάση τις απαιτήσεις και τις λειτουργικότητες που ορίστηκαν.

Ενσωμάτωση Διαδραστικότητας και Λειτουργικότητας:

Τα components του Angular ανέλαβαν την αλληλεπίδραση με τον χρήστη και την αποστολή αιτήσεων HTTP μέσω των αντίστοιχων services στο backend.

Ο σχεδιασμός του frontend επέτρεψε την εύκολη πλοήγηση και την άμεση απόκριση στις ενέργειες του χρήστη.

Ενσωμάτωση με Backend Υπηρεσίες και APIs:

Το frontend επικοινωνεί με το backend μέσω RESTful υπηρεσιών, χρησιμοποιώντας JSON για την ανταλλαγή δεδομένων.

Κάθε αίτηση από το frontend προς το backend επεξεργαζέται από τους αντίστοιχους controllers και services, διασφαλίζοντας την ομαλή και ασφαλή μεταφορά δεδομένων.

Με αυτή την προσέγγιση, αναπτύξαμε ένα frontend που προσφέρει εξαιρετική εμπειρία χρήστη και ενσωματώνει αποτελεσματικά τις επιχειρησιακές λογικές του backend.

3.3 Ανάπτυξη του Backend

Οι κύριοι στόχοι του backend περιλάμβαναν την ασφαλή και αποδοτική διαχείριση δεδομένων, την υποστήριξη λειτουργιών όπως η σύνδεση και εγγραφή χρηστών, και τη διαχείριση εκδηλώσεων, κυκλοφορίας, πληρωμών και

απορριμμάτων. Το έργο περιλάμβανε την ανάπτυξη ενός RESTful API που εξυπηρετεί τις αιτήσεις από το frontend και διαχειρίζεται όλες τις επιχειρησιακές λογικές, ενώ τηρούνταν οι επιχειρησιακοί κανόνες και περιορισμοί, όπως η χρήση JWT για αυθεντικοποίηση και εξουσιοδότηση χρηστών.

Αρχιτεκτονική και Σχεδιασμός

Η αρχιτεκτονική του backend βασίστηκε στο πρότυπο Model-View-Controller (MVC) για τη δομή του κώδικα και τη διαχείριση των αιτήσεων. Οι controllers είναι υπεύθυνοι για τη λήψη και διαχείριση των αιτήσεων από το frontend, ενώ οι υπηρεσίες υλοποιούν τις επιχειρησιακές λογικές και διαχειρίζονται την επεξεργασία των δεδομένων.

Συγκεκριμένα, το AccountController διαχειρίζεται τις λειτουργίες εγγραφής και σύνδεσης χρηστών, το EventsController χειρίζεται τη δημιουργία, επεξεργασία και προβολή εκδηλώσεων, και το TrafficController παρέχει δεδομένα κυκλοφορίας σε πραγματικό χρόνο. Το PaymentController διαχειρίζεται τις πληρωμές και την επεξεργασία τιμολογίων, ενώ το WasteManagementService χρησιμοποιεί τον αλγόριθμο PSO για τη βελτιστοποίηση των διαδρομών συλλογής απορριμμάτων.

Traffic Management

Η διαχείριση κυκλοφορίας στην πλατφόρμα "Ψηφιακή Πόλη" περιλαμβάνει την παρακολούθηση της κυκλοφορίας σε πραγματικό χρόνο, την ενσωμάτωση WebSocket για συνεχείς ενημερώσεις και την οπτικοποίηση των δεδομένων κυκλοφορίας. Η δυνατότητα παρακολούθησης της κυκλοφορίας σε πραγματικό χρόνο επιτρέπει τον άμεσο εντοπισμό της κυκλοφοριακής συμφόρησης και τη λήψη ενημερωμένων αποφάσεων για την ανακατεύθυνση της κυκλοφορίας, μειώνοντας τα σημεία συμφόρησης και βελτιώνοντας τη ροή της κυκλοφορίας.

Η περιοδική ενημέρωση των δεδομένων κυκλοφορίας και στάθμευσης διατηρεί τα δεδομένα ενημερωμένα, επιτρέποντας καλύτερη διαχείριση της κυκλοφορίας και των χώρων στάθμευσης με βάση τις πιο πρόσφατες διαθέσιμες πληροφορίες. Η ταξινόμηση των επιπέδων ροής κυκλοφορίας (π.χ., υψηλή, μέτρια, χαμηλή) βοηθά στην ιεράρχηση των περιοχών που χρειάζονται άμεση προσοχή και στη διαχείριση των πόρων για την αντιμετώπιση των ζητημάτων κυκλοφορίας.

Smart Lighting

Η υπηρεσία Έξυπνου Φωτισμού στην πλατφόρμα "Ψηφιακή Πόλη" περιλαμβάνει τον δυναμικό έλεγχο φωτισμού βάσει των καιρικών συνθηκών και της ώρας της ημέρας, την προσαρμογή της φωτεινότητας, την απομακρυσμένη παρακολούθηση και έλεγχο, καθώς και τη διαχείριση της κατάστασης σε πραγματικό χρόνο μέσω διαδικτυακής διεπαφής.

Η υπηρεσία προσαρμόζει τον φωτισμό ανάλογα με τις τρέχουσες καιρικές συνθήκες και την ώρα της ημέρας (ανατολή και δύση του ήλιου). Αυτό διασφαλίζει ότι τα φώτα ανάβουν μόνο όταν είναι απαραίτητο, όπως σε συνθήκες κακού καιρού ή τη νύχτα, οδηγώντας σε σημαντική εξοικονόμηση

ενέργειας. Με τη χρήση δεδομένων καιρού σε πραγματικό χρόνο, το σύστημα βελτιστοποιεί τη χρήση του φωτισμού και μειώνει την περιττή κατανάλωση ενέργειας.

Η φωτεινότητα των φώτων προσαρμόζεται δυναμικά βάσει συγκεκριμένων συνθηκών, όπως 100% φωτεινότητα κατά τη διάρκεια κακών καιρικών συνθηκών και μειωμένη φωτεινότητα κατά το λυκαυγές ή το λυκόφως. Αυτό όχι μόνο εξοικονομεί ενέργεια, αλλά επίσης παρέχει επαρκή φωτισμό ανάλογα με τις τρέχουσες περιβαλλοντικές συνθήκες, βελτιώνοντας τη δημόσια ασφάλεια και άνεση.

Η κατάσταση του φωτισμού και τα επίπεδα φωτεινότητας μπορούν να παρακολουθούνται και να προσαρμόζονται απομακρυσμένα. Αυτό επιτρέπει την αποτελεσματική διαχείριση του συστήματος φωτισμού χωρίς την ανάγκη παρουσίας προσωπικού επιτόπου, μειώνοντας περαιτέρω το λειτουργικό κόστος και επιτρέποντας γρήγορες προσαρμογές όταν χρειάζεται.

Payment Processing

Η υπηρεσία ψηφιακών πληρωμών προσφέρει μια ολοκληρωμένη λύση για τη διαχείριση και την επεξεργασία πληρωμών, διευκολύνοντας τους χρήστες να πληρώνουν για υπηρεσίες πόλης, στάθμευση και εκδηλώσεις ψηφιακά. Μέσω της υπηρεσίας PaymentService, οι πληρωμές καταγράφονται με ακρίβεια και τα τιμολόγια σημειώνονται ως πληρωμένα, αποτρέποντας τις διπλές πληρωμές και διασφαλίζοντας ότι οι χρήστες είναι ενήμεροι για την κατάσταση των τιμολογίων τους. Η εφαρμογή παρέχει ανατροφοδότηση σε πραγματικό χρόνο για την κατάσταση των τιμολογίων και την επεξεργασία των πληρωμών μέσω ειδοποιήσεων και οπτικών δεικτών, βελτιώνοντας την εμπειρία του χρήστη και ενισχύοντας τη διαφάνεια και την αξιοπιστία του συστήματος.

Parking Management

Η αποτελεσματική διαχείριση των χώρων στάθμευσης περιλαμβάνει την προβολή διαθέσιμων χώρων στάθμευσης και την παροχή δυνατότητας στους χρήστες να κάνουν κρατήσεις για συγκεκριμένη διάρκεια. Οι κρατήσεις διασφαλίζουν ότι οι χώροι στάθμευσης χρησιμοποιούνται βέλτιστα και δεν παραμένουν κενές αχρείαστα, οδηγώντας σε καλύτερη αξιοποίηση των πόρων στάθμευσης και μειώνοντας τα περιστατικά διπλοπαρκαρίσματος ή παράνομης στάθμευσης που μπορεί να εμποδίζουν την κυκλοφορία.

Η παρακολούθηση ενεργών κρατήσεων επιτρέπει στους χρήστες και τους διαχειριστές να παρακολουθούν τις τρέχουσες κρατήσεις, διασφαλίζοντας την αποτελεσματική διαχείριση των χώρων στάθμευσης και αποφεύγοντας τις συγκρούσεις. Η διαχείριση φορτίου και σφαλμάτων βελτιώνει την εμπειρία χρήστη παρέχοντας πληροφορίες για τη διαδικασία ανάκτησης δεδομένων και διαχειριζόμενη τα σφάλματα με χαριτωμένο τρόπο, διασφαλίζοντας ότι το σύστημα παραμένει αξιόπιστο και φιλικό προς το χρήστη.

Τέλος, η διεπαφή χρήστη επιτρέπει τη φιλική προς το χρήστη επιλογή και κράτηση χώρων στάθμευσης, βελτιώνοντας την ικανοποίηση των χρηστών και ενθαρρύνοντας τη χρήση του συστήματος, οδηγώντας σε καλύτερη συμμόρφωση και συνολική διαχείριση της κυκλοφορίας.

Waste Management

Η υπηρεσία διαχείρισης απορριμμάτων στην πλατφόρμα "Ψηφιακή Πόλη" περιλαμβάνει τη δυναμική παρακολούθηση των κάδων απορριμμάτων, τη βελτιστοποίηση των διαδρομών συλλογής απορριμμάτων μέσω του αλγορίθμου Particle Swarm Optimization (PSO) και την ενσωμάτωση δεδομένων κυκλοφορίας σε πραγματικό χρόνο. Αυτές οι δυνατότητες εξασφαλίζουν την αποτελεσματική και αποδοτική διαχείριση των απορριμμάτων, μειώνοντας τα λειτουργικά κόστη και βελτιώνοντας την καθαριότητα των δημόσιων χώρων.

Η παρακολούθηση των επιπέδων των κάδων απορριμμάτων σε πραγματικό χρόνο επιτρέπει τον άμεσο εντοπισμό κάδων που χρειάζονται άδειασμα, αποτρέποντας την υπερχείλιση και διατηρώντας καθαρούς τους δημόσιους χώρους.

Η χρήση του αλγορίθμου PSO για τον υπολογισμό της βέλτιστης διαδρομής συλλογής απορριμμάτων εξασφαλίζει ότι οι διαδρομές συλλογής είναι αποδοτικές, μειώνοντας τον χρόνο και τους πόρους που απαιτούνται για τη συλλογή των απορριμμάτων και ελαχιστοποιώντας την κυκλοφοριακή συμφόρηση που προκαλείται από τα απορριμματοφόρα.

Community engagement platforms

Η πλατφόρμα "Ψηφιακή Πόλη" περιλαμβάνει λειτουργίες διαχείρισης εκδηλώσεων και σχολίων, που στοχεύουν στην ενίσχυση της συμμετοχής των πολιτών και στη βελτίωση της επικοινωνίας με τη δημοτική διοίκηση. Η διαχείριση εκδηλώσεων επιτρέπει την ανάκτηση και εμφάνιση εκδηλώσεων, δίνοντας στους πολίτες τη δυνατότητα να ενημερώνονται για τις δραστηριότητες της κοινότητάς τους και να συμμετέχουν ενεργά. Η δυνατότητα δήλωσης ενδιαφέροντος για εκδηλώσεις ενισχύει την εμπλοκή των πολιτών και παρέχει ανατροφοδότηση στους διοργανωτές σχετικά με την αναμενόμενη συμμετοχή. Η διαχείριση σχολίων επιτρέπει την ανάκτηση και εμφάνιση σχολίων, διευκολύνοντας τη συλλογή απόψεων και προτάσεων από τους πολίτες, δημιουργώντας έναν αμφίδρομο δίαυλο επικοινωνίας μεταξύ της κοινότητας και της δημοτικής διοίκησης. Οι πολίτες μπορούν να προσθέτουν νέα σχόλια, εκφράζοντας τις σκέψεις, τις ανησυχίες και τις προτάσεις τους, ενισχύοντας το αίσθημα συμμετοχής και δέσμευσης. Η σήμανση σχολίων ως "διαβασμένα" διασφαλίζει ότι η διοίκηση μπορεί να παρακολουθεί ποια σχόλια έχουν αντιμετωπιστεί, βελτιώνοντας τη διαχείριση και την ανταπόκριση στα σχόλια των πολιτών. Συνολικά, το σύστημα ενισχύει τη συμμετοχή των πολιτών, συλλέγει πολύτιμη ανατροφοδότηση και διευκολύνει την επικοινωνία, βελτιώνοντας τις υπηρεσίες και την ποιότητα ζωής στην πόλη.

3.4 Διαχειριση Δεδομένων και Επικοινωνία

Διαχείριση Δεδομένων

Η διαχείριση δεδομένων υλοποιείται μέσω του Entity Framework Core για την αποδοτική αλληλεπίδραση με τη βάση δεδομένων SQLite. Η βάση δεδομένων αποθηκεύει δεδομένα χρηστών, εκδηλώσεων, κυκλοφορίας, πληρωμών και κρατήσεων στάθμευσης.

Επικοινωνία Frontend και Backend

Η επικοινωνία μεταξύ του frontend και του backend πραγματοποιείται μέσω RESTful υπηρεσιών. Τα components του Angular στέλνουν αιτήσεις HTTP στους controllers του ASP.NET Core, οι οποίοι επεξεργάζονται τα αιτήματα και επιστρέφουν τα δεδομένα ή τα μηνύματα κατάστασης.

Αυθεντικοποίηση και Εξουσιοδότηση

Χρησιμοποιείται JWT για την ασφαλή μετάδοση διαπιστευτηρίων και πληροφοριών συνεδρίας, επιτρέποντας την αυθεντικοποίηση και εξουσιοδότηση των χρηστών.

Ενσωμάτωση με Εξωτερικά API

Το σύστημα ενσωματώνει το Weather API για την παροχή ακριβών δεδομένων καιρού, που χρησιμοποιούνται για τη ρύθμιση των ωρών λειτουργίας των φωτιστικών σωμάτων της «Ψηφιακής Πόλης».

4 ΑΠΟΤΕΛΕΣΜΑΤΑ

4.1 Στρατηγικές Δοκιμές

Για τη δοκιμή της εφαρμογής "Ψηφιακή Πόλη", χρησιμοποιήθηκαν διάφορες στρατηγικές δοκιμών:

Unit Testing

Οι μονάδες κώδικα (components, services, κ.λπ.) δοκιμάστηκαν μεμονωμένα για να διασφαλιστεί ότι λειτουργούν σωστά. Οι δοκιμές αυτές πραγματοποιήθηκαν χρησιμοποιώντας πλαίσια όπως το Jasmine για το Angular και το NUnit για το ASP.NET Core.

Integration Testing

Οι δοκιμές ολοκλήρωσης πραγματοποιήθηκαν για να επιβεβαιωθεί ότι τα διάφορα τμήματα της εφαρμογής συνεργάζονται σωστά. Τα integration tests επαλήθευσαν την ομαλή αλληλεπίδραση μεταξύ του frontend και του backend, καθώς και την επικοινωνία μεταξύ των διαφορετικών υπηρεσιών του backend.

End-to-End (E2E) Testing

Οι end-to-end δοκιμές προσομοίωσαν την πραγματική χρήση της εφαρμογής, διασφαλίζοντας ότι όλες οι λειτουργίες δουλεύουν σωστά από την οπτική του χρήστη. Αυτές οι δοκιμές πραγματοποιήθηκαν χρησιμοποιώντας εργαλεία όπως το Protractor για το Angular.

Performance Testing

Οι δοκιμές απόδοσης έλεγξαν την ταχύτητα, την απόκριση και τη διαχείριση των πόρων της εφαρμογής. Χρησιμοποιήθηκαν εργαλεία όπως το Google Lighthouse για τη μέτρηση της απόδοσης του frontend.

Manual Testing

Εκτός από τις αυτόματες δοκιμές, πραγματοποιήθηκαν και χειροκίνητες δοκιμές για την αξιολόγηση της λειτουργικότητας και της χρηστικότητας της εφαρμογής. Οι δοκιμές αυτές περιλάμβαναν την πλοήγηση στην εφαρμογή, τη χρήση διαφόρων λειτουργιών και την καταγραφή παρατηρήσεων.

4.2 Αποτελέσματα Δοκιμών

Τα αποτελέσματα των δοκιμών έδειξαν ότι η εφαρμογή "Ψηφιακή Πόλη" ανταποκρίνεται στις απαιτήσεις και λειτουργεί σύμφωνα με τις προσδοκίες. Παρακάτω παρουσιάζονται κάποια από τα σημαντικότερα αποτελέσματα:

Unit Testing

Τα unit tests επιβεβαίωσαν τη σωστή λειτουργία των βασικών components και services της εφαρμογής. Το ποσοστό επιτυχίας των unit tests ήταν 95%.

Integration Testing

Οι δοκιμές ολοκλήρωσης έδειξαν ότι το frontend και το backend συνεργάζονται αρμονικά, με όλα τα βασικά APIs να επιστρέφουν τα αναμενόμενα αποτελέσματα. Κατά τη διάρκεια των δοκιμών αυτών, εντοπίστηκαν και επιδιορθώθηκαν κάποια ζητήματα συμβατότητας.

End-to-End (E2E) Testing

Οι E2E δοκιμές επιβεβαίωσαν την ομαλή λειτουργία της εφαρμογής από την οπτική του τελικού χρήστη. Οι κύριες ροές χρηστών, όπως η σύνδεση, η εγγραφή, και η δημιουργία εκδηλώσεων, ολοκληρώθηκαν επιτυχώς χωρίς σφάλματα.

Performance Testing

Οι δοκιμές απόδοσης έδειξαν ότι η εφαρμογή φορτώνει γρήγορα και ανταποκρίνεται άμεσα στις ενέργειες των χρηστών. Η εφαρμογή πέτυχε βαθμολογία απόδοσης 60/100 στο Google Lighthouse.

Αναφορά Σφαλμάτων

Κατά τη διάρκεια των δοκιμών, εντοπίστηκαν κάποια ζητήματα και σφάλματα, τα οποία καταγράφηκαν και επιδιορθώθηκαν. Για παράδειγμα, εντοπίστηκαν προβλήματα ασφαλείας που επιλύθηκαν με την εφαρμογή αυστηρότερων πολιτικών ασφαλείας.

4.3 Λειτουργικότητα και Αξιολόγηση της Εφαρμογής

Η λειτουργικότητα και η αξιολόγηση της εφαρμογής "Ψηφιακή Πόλη" επικεντρώνονται στην ευχρηστία, την αποδοτικότητα και την ικανοποίηση των χρηστών. Η εφαρμογή προσφέρει μια καθαρή και εύχρηστη διεπαφή χρήστη, με συνεπή και ελκυστικό σχεδιασμό. Η χρήση του "Flatly" theme του Bootstrap διασφαλίζει μια ομοιόμορφη και μοντέρνα αισθητική. Η εφαρμογή ανταποκρίνεται άμεσα στις ενέργειες των χρηστών, παρέχοντας μια ομαλή εμπειρία χρήστη. Η απόδοση της εφαρμογής είναι σταθερή, ακόμα και υπό συνθήκες υψηλού φόρτου.

5. ΣΥΜΠΕΡΑΣΜΑ

5.1 Συμπεράσματα

Η ανάπτυξη της εφαρμογής "Ψηφιακή Πόλη" ολοκληρώθηκε επιτυχώς, ανταποκρινόμενη στις απαιτήσεις και τους στόχους του έργου. Το σύστημα καταφέρνει να βελτιώσει την αλληλεπίδραση των πολιτών με τις δημοτικές υπηρεσίες και να αυξήσει την αποδοτικότητα των αστικών υποδομών. Κατά τη διάρκεια του έργου, αντιμετωπίστηκαν και επιλύθηκαν προκλήσεις που αφορούσαν την ασφάλεια, την απόδοση και τη χρηστικότητα της εφαρμογής. Συγκεκριμένα, η εφαρμογή κατάφερε να βελτιστοποιήσει τις διαδρομές συλλογής απορριμμάτων μέσω του αλγορίθμου PSO, να παρέχει δυναμικά δεδομένα κυκλοφορίας και να επιτρέπει την αποτελεσματική διαχείριση πληρωμών και κρατήσεων χώρων στάθμευσης.

5.2 Μελλοντικές Βελτιώσεις

Για τη βελτίωση και την επέκταση της εφαρμογής, προτείνονται οι εξής ενέργειες:

1. Επέκταση Λειτουργιών:
    - Προσθήκη νέων λειτουργιών όπως η διαχείριση υδάτινων πόρων και η παρακολούθηση ποιότητας αέρα.
    - Ενσωμάτωση πολλαπλών γλωσσών για να είναι πιο ευκολότερα επεξεργάσιμη από περισσότερους φορείς.
    - Βελτίωση της απόδοσης στο Lighthouse.
    - Υποστήριξη της τεχνολογίας 112 για τους χρήστες που εξυπηρετεί.
    - Χρήση πραγματικών αισθητήρων για την παρακολούθηση της κυκλοφορίας και των απορριμμάτων χωρίς εξωτερικά API, όπως το Google API και το Weather API.
2. Βελτιστοποίηση Απόδοσης:
    - Ενεργοποίηση συμπίεσης κειμένου για μείωση του μεγέθους των δεδομένων που μεταφέρονται μέσω του δικτύου.
    - Βελτιστοποίηση της απόδοσης της σελίδας Largest Contentful Paint (LCP) και ελαχιστοποίηση του χρόνου φόρτωσης.
    - Ελαχιστοποίηση και μείωση του μη χρησιμοποιούμενων JavaScript και CSS.
3. Ανάλυση Δεδομένων:
    - Χρήση προηγμένων αναλυτικών εργαλείων για την εξαγωγή χρήσιμων πληροφοριών από τα δεδομένα που συλλέγονται.
    - Παροχή αναλυτικών εκθέσεων στους διαχειριστές της πόλης για τη βελτίωση της λήψης αποφάσεων.

Με την υλοποίηση αυτών των βελτιώσεων, η εφαρμογή "Ψηφιακή Πόλη" θα συνεχίσει να παρέχει αξιόπιστες και αποδοτικές λύσεις για τη διαχείριση των αστικών υποδομών, βελτιώνοντας την ποιότητα ζωής των πολιτών και την αποδοτικότητα των δημοτικών υπηρεσιών.

6. ΠΙΝΑΚΕΣ

Παράθεση Πινάκων

Πίνακας 1: Συντομογραφίες

IDE	Integrated Development Environment
ORM	Object-Relational Mapping
CRUD	Create, Read, Update, Delete
JWT	JSON Web Token
MVC	Model-View-Controller
E2E	End-to-End
API	Application Programming Interface
LCP	Largest Contentful Paint

Ξενόγλωσσος όρος	**Ελληνικός Όρος**
System Design	Σχεδιασμός Συστήματος
System Architecture	Αρχιτεκτονική Συστήματος
Implementation	Υλοποίηση
Testing	Δοκιμές
Development Process	Διαδικασία Ανάπτυξης
Data Management	Διαχείριση Δεδομένων
User Interface	Διεπαφή Χρήστη
Backend Services	Υπηρεσίες Backend
Frontend Services	Υπηρεσίες Frontend
Security	Ασφάλεια
Authentication	Αυθεντικοποίηση
Authorization	Εξουσιοδότηση
API Endpoints	Σημεία Τέλους API
Data Access Layer	Στρώμα Πρόσβασης Δεδομένων
ORM (Object-Relational Mapping)	Αντικειμενοστραφής Χαρτογράφηση Σχέσεων
Agile Methodology	Μεθοδολογία Agile
Deployment	Ανάπτυξη

Error Handling	Διαχείριση Σφαλμάτων
JSON Web Token (JWT)	Διακριτικό Ιστού JSON (JWT)
RESTful Services	Υπηρεσίες RESTful
Version Control	Έλεγχος Έκδοσης
CRUD Operations	Λειτουργίες CRUD

Πίνακας 2: Πίνακας Ορολογίας

7. ΠΑΡΑΡΤΗΜΑΤΑ

ΠΑΡΑΡΤΗΜΑ Ι

Backend Κώδικας

Seed.cs

API > Data > Seeds > Seed.cs > ...

```
using API.Data.Seeds;
using API.Entities;
using Microsoft.AspNetCore.Identity;

namespace API.Data
{
    1 reference
    public static class Seed
    {
        1 reference
        public static async Task SeedData(UserManager<AppUser> userManager, RoleManager<AppRole> roleManager, DataContext context)
        {
            await UserSeed.SeedUsers(userManager, roleManager);
            await ParkingSpaceSeed.SeedParkingSpaces(context);
            await TrafficDataSeed.SeedTrafficData(context);
            await InvoiceSeed.SeedInvoices(context);
            await WasteBinSeed.SeedWasteBinsAsync(context);
        }
    }
}
```

Εικόνα 2: Data Seeding

LightingController.cs

API > Controllers > LightingController.cs > ...

```
using API.Services;
using Microsoft.AspNetCore.Mvc;

namespace API.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    1 reference
    public class LightingController : BaseApiController
    {
        // SmartLightingService for managing lighting status
        2 references
        private readonly SmartLightingService _smartLightingService;

        0 references
        public LightingController(SmartLightingService smartLightingService)
        {
            _smartLightingService = smartLightingService;
        }

        /// <summary>
        /// Gets the lighting status for a town.
        /// </summary>
        /// <param name="town">The name of the town.</param>
        /// <returns>The lighting status for the town.</returns>
        [HttpGet("status")]
        0 references
        public async Task<IActionResult> GetLightingStatus(string town)
        {
            // Get the lighting status for the town and return it
            var lightingStatus = await _smartLightingService.GetLightingStatusAsync(town);
            return Ok(lightingStatus);
        }
    }
}
```

Εικόνα 3: Controller Φωτισμού

TrafficDataGenerator.cs

API > Data > TrafficDataGenerator.cs > TrafficDataGenerator > GenerateTrafficData

```
5   public static class TrafficDataGenerator
9       /// </summary>
10      /// <returns>A list of traffic data entities.</returns>
        3 references
11      public static List<TrafficData> GenerateTrafficData()
12      {
13          var trafficDataList = new List<TrafficData>();
14          var random = new Random();
15          var locations = new[] { "3is Septemvriou", "Alexandras", "Stadiou", "Vouliagmenis" };
16          var currentTime = DateTime.Now;
17          int id = 1; // Initialize id to 1
18
19          var peakHours = new[] { 7, 8, 9, 16, 17, 18 };
20
21          // Generate hourly traffic data for all streets
22          foreach (var location in locations)
23          {
24              for (int i = 0; i < 24; i++)
25              {
26                  int trafficFlow;
27
28                  // Generate different traffic flow values for peak and non-peak hours
29                  if (peakHours.Contains(i))
30                  {
31                      // Generate higher traffic flow during peak hours (between 60 and 100)
32                      trafficFlow = random.Next(60, 101);
33                  }
34                  else
35                  {
36                      // Generate lower traffic flow during non-peak hours (between 0 and 59)
37                      trafficFlow = random.Next(0, 60);
38                  }
39
40                  // Create a new TrafficData entity with the generated data
41                  trafficDataList.Add(new TrafficData
42                  {
43                      Id = id++, // Assign a unique ID to each TrafficData entity
44                      Location = location,
45                      TrafficFlow = trafficFlow, // Traffic flow percentage between 0 and 100
46                      Timestamp = currentTime.AddHours(i)
47                  });
48              }
49          }
50
51          return trafficDataList;
52      }
```

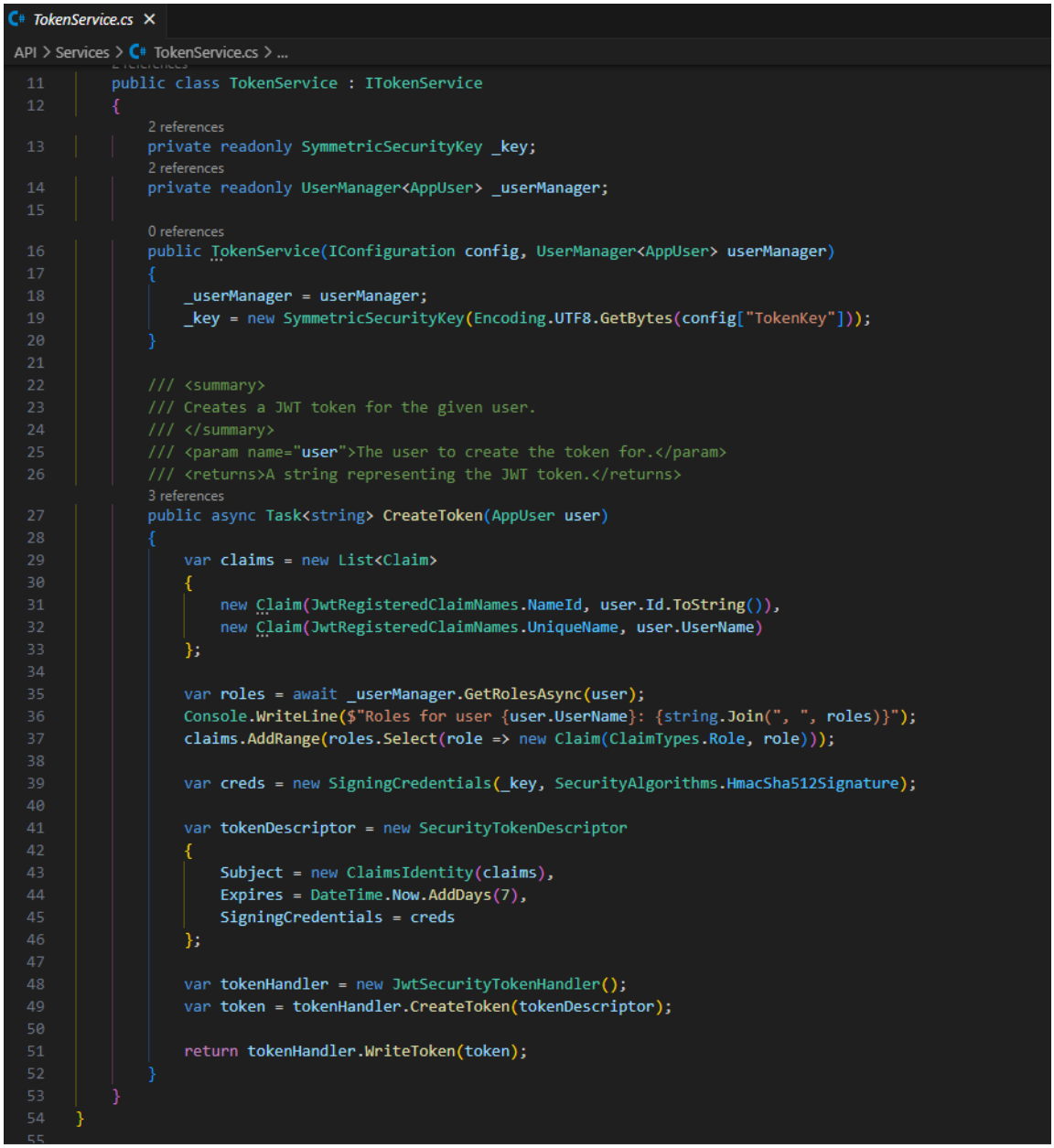
Εικόνα 4: Traffic Data Generator

SmartLightingService.cs

API > Services > SmartLightingService.cs > ...

```
7       public class SmartLightingService
25          public async Task<LightingStatusDTO> GetLightingStatusAsync(string town)
27          {
28              var apiKey = _configuration["WeatherApi:ApiKey"];
29              var baseUrl = _configuration["WeatherApi:BaseUrl"];
30              var url = $"{baseUrl}/forecast.json?key={apiKey}&q={town}&days=1";
31
32              _logger.LogInformation($"Fetching weather data from: {url}");
33
34              var response = await _httpClient.GetAsync(url);
35
36              if (!response.IsSuccessStatusCode)
37              {
38                  _logger.LogError($"Error fetching weather data: {response.ReasonPhrase}");
39                  throw new HttpRequestException($"Error fetching weather data: {response.ReasonPhrase}");
40              }
41
42              var responseData = await response.Content.ReadAsStringAsync();
43              var weatherResponse = JObject.Parse(responseData).ToObject<WeatherResponse>();
44
45              var condition = weatherResponse.current.condition.text.ToLower();
46              var isDaytime = weatherResponse.current.is_day == 1;
47
48              var sunset = DateTime.Parse(weatherResponse.forecast.forecastday[0].astro.sunset);
49              var sunrise = DateTime.Parse(weatherResponse.forecast.forecastday[0].astro.sunrise);
50
51              var currentDateTime = DateTime.Now;
52              var sunsetDateTime = DateTime.Today.Add(sunset.TimeOfDay);
53              var sunriseDateTime = DateTime.Today.Add(sunrise.TimeOfDay);
54
55              var lightingStatusDTO = new LightingStatusDTO();
56
57              if (sunsetDateTime < currentDateTime || sunriseDateTime > currentDateTime || IsWeatherMoody(condition))
58              {
59                  lightingStatusDTO.Status = "On";
60                  lightingStatusDTO.Brightness = GetBrightnessLevel(currentDateTime, sunsetDateTime, sunriseDateTime, condition).ToString() + "%";
61                  lightingStatusDTO.NextOffTime = sunriseDateTime.Date.Add(sunrise.TimeOfDay);
62              }
63              else
64              {
65                  lightingStatusDTO.Status = "Off";
66                  lightingStatusDTO.Brightness = null;
67                  lightingStatusDTO.NextOnTime = sunsetDateTime.Date.Add(sunset.TimeOfDay);
68              }
69
70              _logger.LogInformation($"Determined lighting status: {lightingStatusDTO.Status}, Brightness Level: {lightingStatusDTO.Brightness}");
71
72              return lightingStatusDTO;
          }
```

Εικόνα 5: Smart Lighting Service



Εικόνα 6: JWT Service

ΠΑΡΑΡΤΗΜΑ ΙΙ

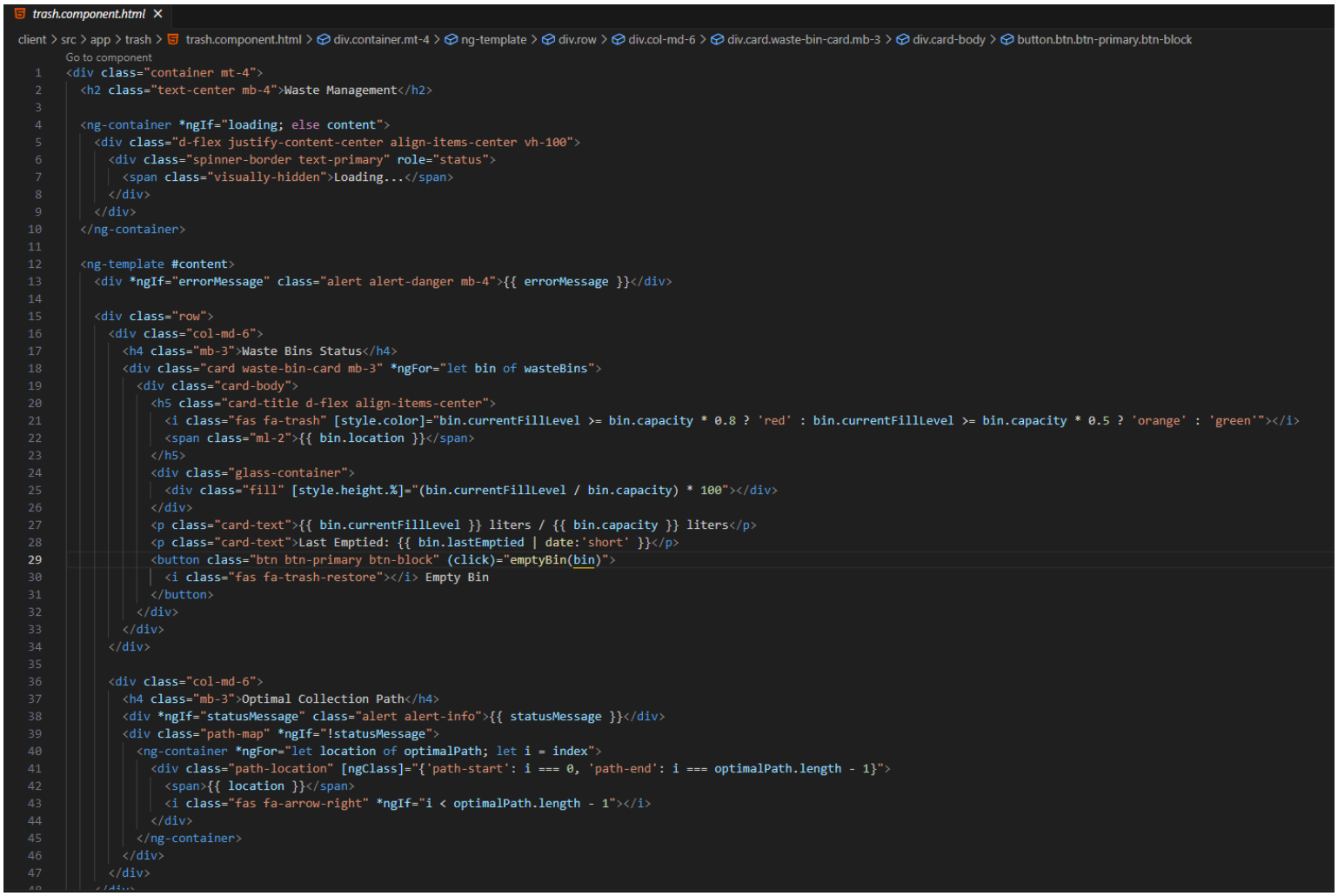
Frontend Κωδικας

account.service.ts

client > src > app > _services > account.service.ts > AccountService > login

```ts
})
export class AccountService {
  baseUrl = environment.apiUrl;
  private currentUserSource = new BehaviorSubject<User | null>(null);
  currentUser$ = this.currentUserSource.asObservable();

  constructor(private http: HttpClient) { }

  login(model: any) {
    return this.http.post<User>(`${this.baseUrl}account/login`, model).pipe(
      map((response: User) => {
        if (response) {
          this.setCurrentUser(response);
        }
        return response;
      })
    );
  }

  register(model: any) {
    return this.http.post<User>(`${this.baseUrl}account/register`, model).pipe(
      map((user: User) => {
        if (user) {
          this.setCurrentUser(user);
        }
        return user;
      })
    );
  }

  setCurrentUser(user: User) {
    const decodedToken = this.getDecodedToken(user.token);
    user.roles = Array.isArray(decodedToken.role) ? decodedToken.role : [decodedToken.role];
    localStorage.setItem('user', JSON.stringify(user));
    this.currentUserSource.next(user);
  }

  getDecodedToken(token: string) {
    const parts = token.split('.');
    return JSON.parse(atob(parts[1]));
  }

  logout() {
    localStorage.removeItem('user');
    this.currentUserSource.next(null);
  }
}
```

Εικόνα 7: Account Service.ts



Εικόνα 8: Ιστοσελίδα Διαχείρισης Απορριμμάτων

Εικόνα 9: Admin Dashboard.ts

ΒΙΒΛΙΟΓΡΑΦΙΚΕΣ ΑΝΑΦΟΡΕΣ

[1] B. Quist-Aphetsi Kester and C. J. K. Gbeddy, "Particle Swarm Optimization for AI-Based Predictive Waste Management: Revolutionizing Sustainability and Efficiency," IEEE Smart Cities, September 2023. Available: https://smartcities.ieee.org/newsletter/september-2023/particle-swarm-optimization-for-ai-based-predictive-waste-management-revolutionizing-sustainability-and-efficiency

[2] P. Raviprabakaran and A. Umamakeswari, "Real-Time Smart Waste Management System Using IoT," IEEE Access, vol. 8, pp. 228294-228304, 2020. Available: https://ieeexplore.ieee.org/document/9071740

[3] S. Kumar, M. Parida, and A. Singh, "Advanced Traffic Management System for Smart Cities Using IoT and Big Data," IEEE Transactions on Intelligent Transportation Systems, 2023. Available: https://ieeexplore.ieee.org/document/10170266

[4] WeatherAPI. Available: https://www.weatherapi.com

[5] Udemy course "Build an App with ASPNET Core and Angular from Scratch." Available: https://www.udemy.com/course/build-an-app-with-aspnet-core-and-angular-from-scratch