מסמך עיצוב

מערכת חיפוש חנייה

מגישים: מיכל רבינוביץ' ותומר טולדו

סדנה בתכנות מונחה עצמים – 20586 – 2025ב

האוניברסיטה הפתוחה

תוכן עניינים

מבנה המערכת

הנחות עבודה

מוסכמות כתיבה במערכת

בסיס הנתונים

טבלאות מסד הנתונים

דיאגרמת בסיס הנתונים

שרת האפליקציה

API

מחלקות העברת מידע (DTO)

דיאגרמות מחלקות העברת מידע משותפת עבור הAPI

נקודות קצה

שירותים – Services

דיאגרמת ממשקים ומחלקות שירות

מחלקות עזר שונות

אפליקציית ParkSpotTLV למשתמש

הרשאות

ארכיטקטורה

מחלקות מידע משותפות

שירותים Services

דיאגרמת ממשקים ומחלקות שירות

מחלקות עזר – Helpers ודיאגרמות

מחלקות Components ודיאגרמות

מחלקות Model

דיאגרמות מחלקות המודל

דפוסי עיצוב Design Patterns

1 2 3 4

שינויים אפשריים עתידיים

התראות לפני גמר חנייה

לחיצה על רחוב פירוט כללי על חנייה

דיאגרמות רצף

רישום משתמש והנפקת זוג טוקנים – צד שרת POST /auth/register —

יצירת רכב (כולל בדיקות שם ואופציונלית היתרים) – צד שרת POST /vehicles —

שליפת מקטעים בבוקס והערכה – צד שרת POST /map/segments —

פתיחת סשן חניה – צד שרת POST /parking/start —

סגירת סשן וחישוב חיובים/תקציב – צד שרת POST /parking/stop —

תהליך Log-in לאפליקציה – צד לקוח

תהליך הרשמה לאפליקציה – צד לקוח

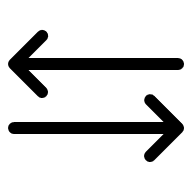
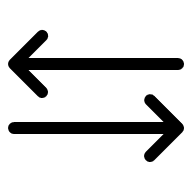
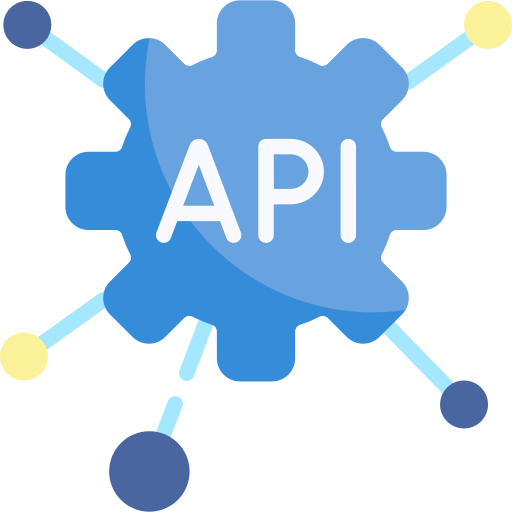
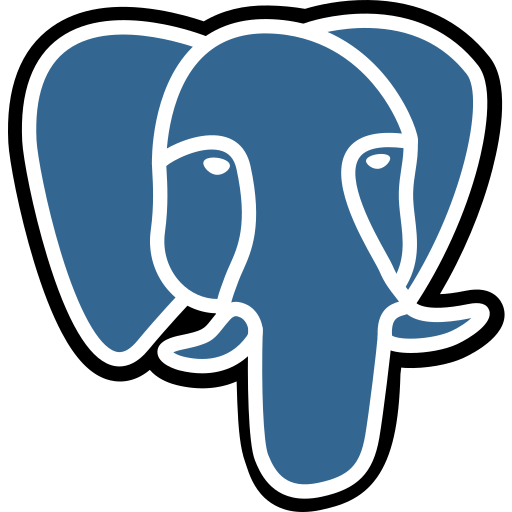
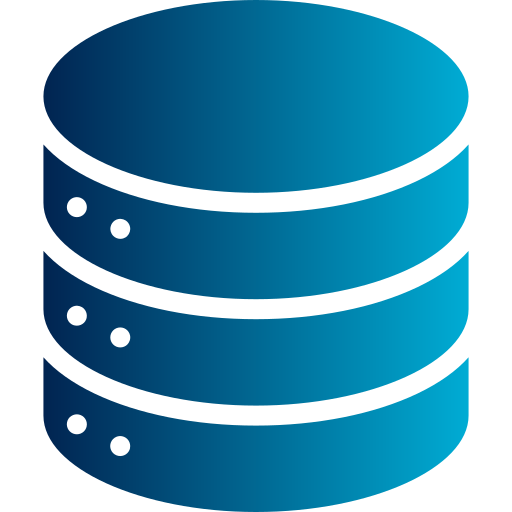
תהליך התחלת חנייה – צד לקוח

תהליך עצירת חנייה – צד לקוח

מבנה המערכת

**המערכת מורכבת מ3 גורמים עיקריים**:

* בסיס נתונים – מנהל מידע על משתמשים, רכבים ותווי חנייה בבעלותם, איזורי חנייה ורחובות.
* שרת אפליקציה – שרת מרכזי שמתחבר לבסיס הנתונים ומשרת את האפליקציה.
* אפליקצית אנדרואית ParkSpotTLV – אפליקציה המיועדת למחפשי חנייה.



הנחות עבודה

**תחום הפיילוט והיקף**

* המערכת רצה בשלב פיילוט ומוגבלת גיאוגרפית לעיר תל-אביב-יפו.
* תמיכה ב Android בלבד (.NET MAUI).
* אין פאנל ניהול בשלב הפיילוט.
* אין חיוב/סליקה ואין אינטגרציה לתשלום חניה (Pango/CityPay) בשלב זה, למעט כפתור לפתיחת אפליקציה ייעודית.

**זמינות ותקינות**

* זמינות השירות תלויה בזמינות שרת ה-API ובסיס הנתונים (PostgreSQL + PostGIS).
* תקינות תצוגת המפה תלויה בזמינות שירותי מפה חיצוניים (Google Maps) ובחיבור אינטרנט תקין במכשיר.

**הרשאות ודורשי מכשיר (Android)**

* נדרשות הרשאות מיקום ואינטרנט.
* ללא הרשאות מיקום, חוויית השימוש מוגבלת: לא יוצגו שכבות קרובות, לא תוצג אינדיקציית מיקום, ולא יהיה ניתן להפעיל את החנייה.

**זיהוי משתמש והרשמה**

* הרשמה והתחברות באמצעות שם משתמש וסיסמה.
* משתמש יכול לרשום מספר כלי רכב לחשבון אחד.

מוסכמות כתיבת המערכת

**מוסכמות שמות (C#/.NET)**

* מחלקות / ממשקים / מתודות / Enums ייכתבו בPascalCase:  
  ParkingSession StartParkingSession, ResidentZoneCode
* ממשקים יתחילו באות I: IVehicleRepository, IRulesEvaluator.
* שדות פרטיים ייכתבו עם קו תחתי מוביל: \_dbContext, \_clock
* פרמטרים ומשתנים מקומיים ייכתבו בcamelCase: vehicleId, minParking
* שמות קבצים/מחלקות זהים: קובץParkingSession.cs יכיל את המחלקה ParkingSession.
* סיומות קבצים / מחלקות יהיו בהתאם לתפקיד.   
  לדוגמה: services יסיימו עם הסיומת Service  
  בקשות / תשובות / יחידות העברת מידע יסתיימו עם Request / Response / Dto.
* דפים ורכיבי View יסתיימו עם סיומת מתאימה כגון MapPage / MapView.
* שמות הטבלאות ועמודות במאגר הנתונים ישמרו ב-snake\_case ובלשון רבים, לדוגמה: parking\_sessions.
* מפתחות ושדות זמניים יירשמו בקוד בתצורה מובנת, אך נגישים באמצאות snake\_case.   
  לדוגמה: עבור כל רכב יש ID השמור בשם VehicleId, אך נפנה אליו במאגר הנתונים כvehicle\_id.

**מוסכמות קוד**

* כל רכיב מטפל בתחומו הצר בלבד.
* Clean Code – קריא, קצר, עקבי ; פונקציות קטנות ושמות בעלי משמעות.
* Design Patterns – שימוש ממוקד לפי צורך (Singleton, Builder, Strategy).
* טיפול בזמן: השרת כולו ב-UTC. האפליקציה מופעלת בזמני Asia/Jerusalem. אין שימוש ב-DateTime.Now בצד השרת – רק בשעון מופשט (IClock).
* חריגות ולוגים: שימוש ב-ProblemDetails עבור ה-API ולוגים בSerilog.

בסיס הנתונים

בסיס הנתונים של המערכת משמש מקור אמת יחיד (Single Source of Truth) לכל המידע התפעולי, ובכללי משתמשים, כלי רכב, אזורי חניה, מקטעי רחוב, כללי חניה, סשני חניה, תקציבי חניה יומיים, וטוקני דחיפה.

**בחירת טכנולוגיה**

* סוג מסד נתונים: PostgreSQL 16 - עםPostGIS 3  ליכולות גיאו־מרחביות.
* גישה וישימות בקוד: Entity Framework Core בשיטת Code First (Controlled Migrations).

**ישויות וטבלאות מרכזיות**

משתמשים – Users

טבלה המכילה את משתמשי המערכת והפרטים האישיים שלהם ובפרט טוקני אימות.

|  |  |  |
| --- | --- | --- |
| שם עמודה | טיפוס | שימוש |
| Id | Guid | מזהה משתמש |
| Username | string | מזהה משתמש |
| PasswordHash | string | ססמא מוצפנת |
| Vehicles | ICollection<Vehicle> | רשימת רכבים ברשות המשתמש |
| RefreshTokens | ICollection<RefreshToken> | רשימת תוקני דחיפה לצורך אימות |

רכבים – Vehicles

טבלה המכילה את כלל כלי הרכב, משייכת אותם למשתמשים ומסווגת לפי סוגים והיתרים.

|  |  |  |
| --- | --- | --- |
| שם עמודה | טיפוס | שימוש |
| Id | Guid | מזהה רכב |
| Owner | User | בעלים |
| OwnerId | Guid | מזהה בעלים |
| Name | string | שם הרכב |
| Type | VehicleType | סוג הרכב |
| Xmin | uint | קטע קריטי |
| Permits | ICollection<Permit> | תווים ברשות הרכב |

תווים – Permits

טבלה המכילה את כלל התווים המשויכים למשתמש / רכב ספציפי.

|  |  |  |
| --- | --- | --- |
| שם עמודה | טיפוס | שימוש |
| Id | Guid | מזהה תו |
| Vehicle | Vehicle | רכב בעלים |
| VehicleId | Guid | מזהה רכב |
| Type | PermitType | סוג תו חנייה |
| ZoneCode | Int | איזור משוייך |
| Zone | Zone | איזור משוייך |
| Xmin | uint | קטע קריטי |
| LastUpdatedUtc | DateTimeOffset | זמן עדכון אחרון |

איזורים – Zones

טבלה המכילה את כלל איזורי החנייה בתל-אביב המוגדרים על ידי MultiPolygon.

|  |  |  |
| --- | --- | --- |
| שם עמודה | טיפוס | שימוש |
| Id | Guid | מזהה איזור בטבלה |
| Code | Int? | מספר איזור בתל-אביב |
| Taarif | Tariff | תעריף איזורי |
| Name | string | שם האיזור |
| Geom | MultiPolygon | גבולות האיזור |
| Segments | ICollection<StreetSegments> | רשימת מקטעים באיזור |
| LastUpdatedUtc | DateTimeOffset | ניהול מידע |

טוקני דחיפה – RefreshTokens

טבלה המנהלת את טוקני הדחיפה לכלל המשתמשים.

|  |  |  |
| --- | --- | --- |
| שם עמודה | טיפוס | שימוש |
| Id | Guid | מזהה טוקן |
| UserId | Guid | מזהה משתמש |
| TokenHash | string | טוקן מוצפן |
| CreatedAtUtc | DateTimeOffset | זמן יצירה |
| ExpiresAtUtc | DateTimeOffset | זמן פקיעת תוקף |
| RevokedAtUtc | DateTimeOffset | זמן פקיעה |
| ReplacedByTokenHash | String | טוקן שהחליף את טוקן זה |
| User | User | משתמש משוייך |

קטעי רחוב – StreetSegments

טבלה המכילה את כלל מקטעי הרחוב בתל-אביב המוגדרים על ידי LineString. כל מקטע משוייך לאיזור ומוגדר לו סוג חנייה.

|  |  |  |
| --- | --- | --- |
| שם עמודה | טיפוס | שימוש |
| Id | Guid | מזהה משתמש |
| Geom | LineString | מזהה משתמש |
| OSMid | string | ססמא מוצפנת |
| NameEnglish | string | רשימת רכבים ברשות המשתמש |
| NameHebrew | ICollection<RefreshToken> | רשימת תוקני דחיפה לצורך אימות |
| ZoneId | Guid | מזהה איזורי |
| Zone | Zone | איזור משוייך |
| ParkingType | ParkingType | סוג חנייה |
| Side | SegmentSide | צד חנייה בכביש |

כללי חנייה – TariffWindow

טבלה המכילה את הכללי חנייה הבסיסיים היום-יומיים בתל-אביב.

|  |  |  |
| --- | --- | --- |
| שם עמודה | טיפוס | שימוש |
| Id | Int | מזהה יום |
| Tariff | Tariff | מזהה תעריף |
| DayOfWeek | DayOfWeek | יום בשבוע |
| StartLocal | TimeOnly | שעת התחלת חוק |
| EndLocal | TimeOnly | שעת סיום חוק |

פעולת חנייה – Parking Sessions

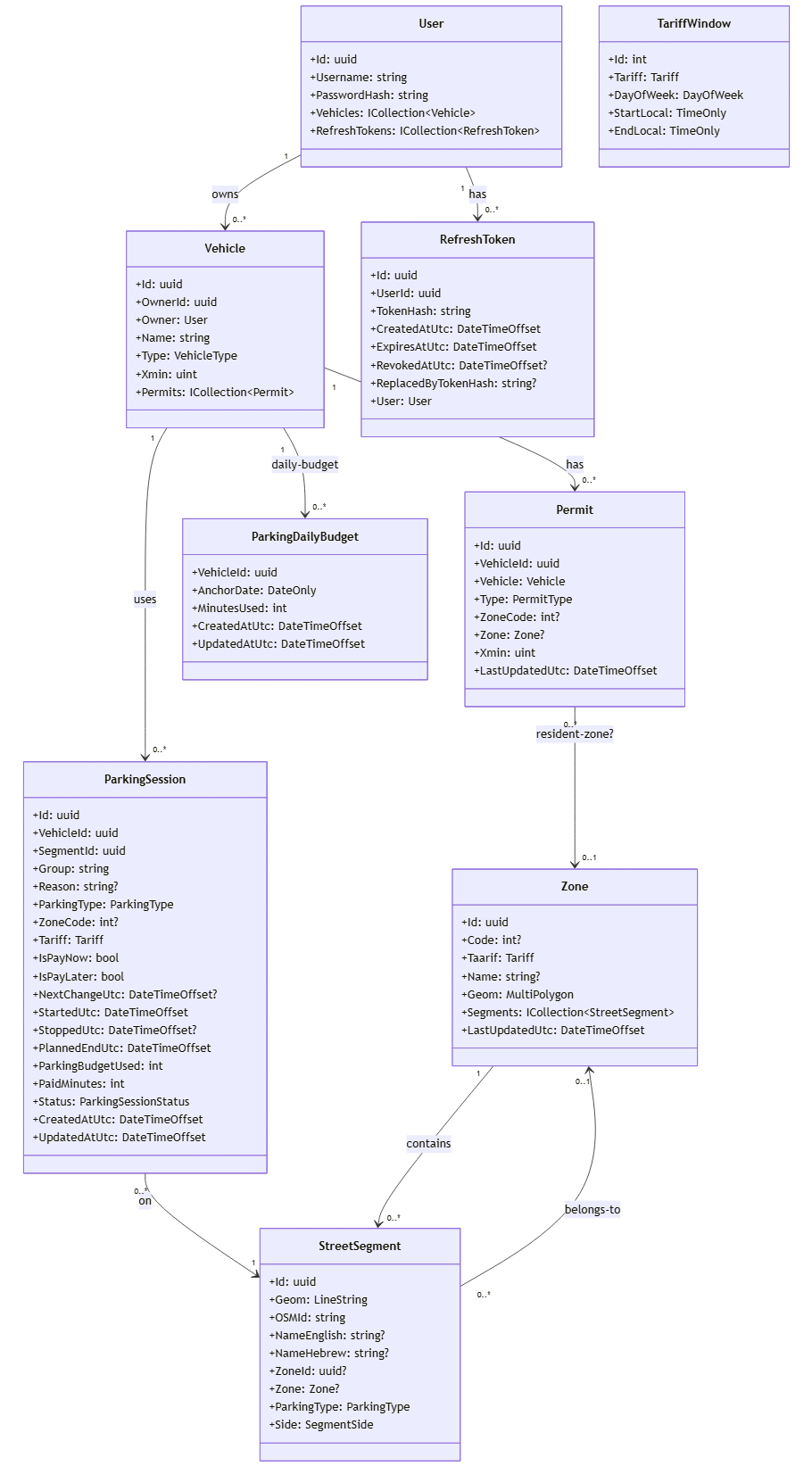
טבלה המנהלת את סטטוס, תחילת/סיום/תפוגה של כלל החניות הפעילות.

|  |  |  |
| --- | --- | --- |
| שם עמודה | טיפוס | שימוש |
| Id | Guid | מזהה סשן |
| VehicleId | Guid | מזהה רכב |
| SegmentId | Guid | מזהה רחוב |
| Group | string | קבוצת חנייה |
| Reason | string | סיבת שיוך לקבוצת החנייה |
| ParkingType | ParkingType | סוג חנייה (חינם/בתשלום) |
| ZoneCode | Int | שיוך איזורי |
| Tariff | Tariff | תעריף |
| IsPayNow | Bool | תשלום מנקודת התחלה? |
| IsPayLater | Bool | תשלום מנקודה מאוחרת יותר? |
| NextChangeUtc | DateTimeOffset | זמן שינוי מצב |
| StartedUtc | DateTimeOffset | זמן התחלה |
| StoppedUtc | DateTimeOffset | זמן עצירה |
| PlannedEndUtc | DateTimeOffset | זמן עצירה מתוכנן |
| ParkingBudgetUsed | Int | כמות חנייה חופשית שנצרכה |
| PaidMinutes | Int | כמות חנייה בתשלום שנצרכה |
| Status | ParkingSessionStatus | סטטוס נוכחי |
| CreatedAtUtc | DateTimeOffset | זמן יצירה |
| UpdatedAtUtc | DateTimeOffset | זמן עדכון אחרון |

חנייה חופשית יומית – Parking Daily Budget

טבלה המנהלת את תקציב החניה היומית החינמית לכלל המשתמשים.

|  |  |  |
| --- | --- | --- |
| שם עמודה | טיפוס | שימוש |
| Id | Guid | מזהה משתמש |
| AnchorDate | string | מזהה משתמש |
| MinutesUsed | Int | כמות יומית שנצרכה |
| CreatedAtUtc | DateTimeOffset | מתי נוצר (לאיזה יום משוייך) |
| UpdatedAtUtc | DateTimeOffset | זמן עדכון אחרון |

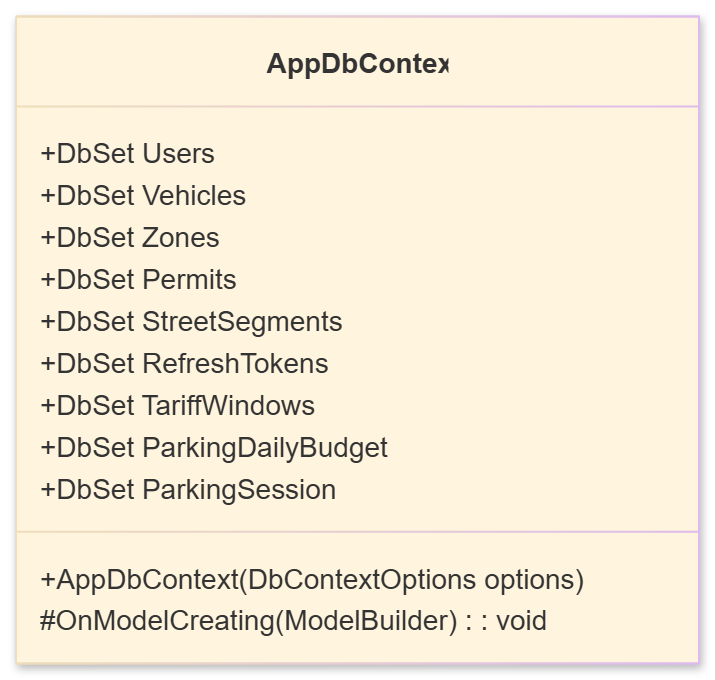
**דיאגרמת בסיס הנתונים**

שרת האפליקציה

שרת האפליקציה מרכז את כלל הלוגיקה של ParkSpotTLV:

ניהול משתמשים והרשאות (JWT + Refresh), ניהול כלי רכב והיתרים, שליפה והערכה של מקטעי רחוב ואזורי חניה (PostGIS), פתיחה/סגירה של סשני חניה וניהול תקציב חניה יומי. השרת סטטלס, מתחבר למסד PostgreSQL + PostGIS באמצעות EF Core, ונשען על אינדקסי GiST לעמודות גאומטריות.

מימוש החיבור (Data Access Layer) ומודל הנתונים מתואר במחלקה AppDbContext. במחלקה זו מוגדרים כל המודלים והמחלקות שינוהלו בבסיס הנתונים, מחלקה זו נדרשת על מנת לממש את החיבור בעזרת EntityFramework, לצורך זה מחלקה זו יורשת ממחלקה DbContext ומאפיינת את המודלים בעזרת המחלקה הגנרית DbSet.

****אין מחלקות מידע משותפות בין השרת לקליינט. החוזה בין הצדדים מוגדר דרך DTOs של הAPI ומתואר בעזרת OpenAPI. הדבר מצמצם זיקה הדדית בין פרויקטים ומאפשר לשמור יציבות API תוך שינויי מימוש פנימיים.

**Minimal API**

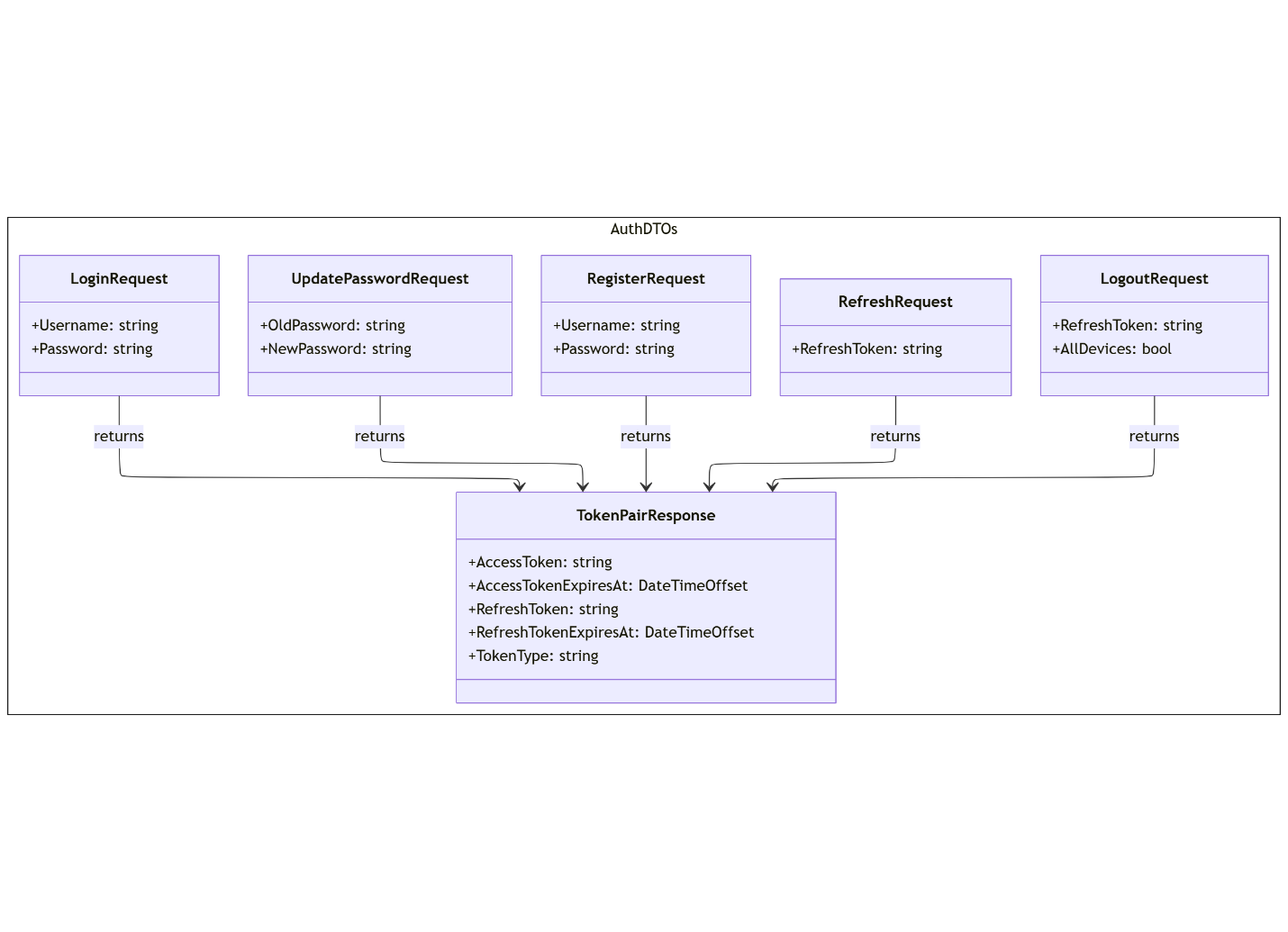
ה-API ממומש בעזרת Minimal API, ואין שימוש בControllers. כל תחום (Auth / Vehicles / Map / Sessions / Permits) מקבל נקודת קצה משלו.

כל נקודת קצה עובר בתחילת התהליך את ה-Verifications הדרושים לו, ולאחר מכן מקבל במידת הצורך DTO, מוזרקים לו שירותים/DbContext דרך DI, מריץ לוגיקה ומחזיר תוצאה בפורמט JSON.

לכל Endpoint יש מחלקת Problems אשר מנהלת את השגיאות בנקודת הקצה.

**מחלקות העברת מידע (DTO)**

Auth DTOs

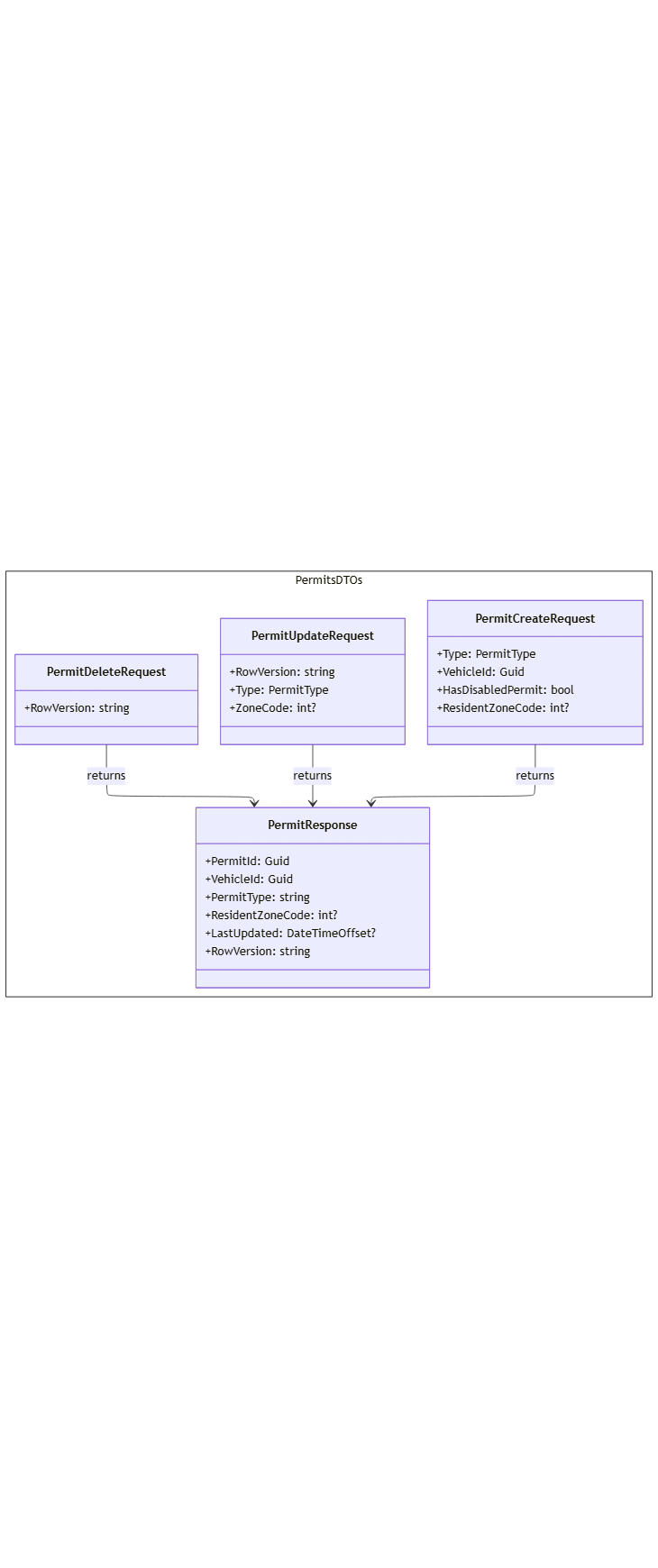
מחלקת העברת מידע על הרשמה, כניסה ויציאה והעברת טוקני הרשאות.

Vehicle DTOs

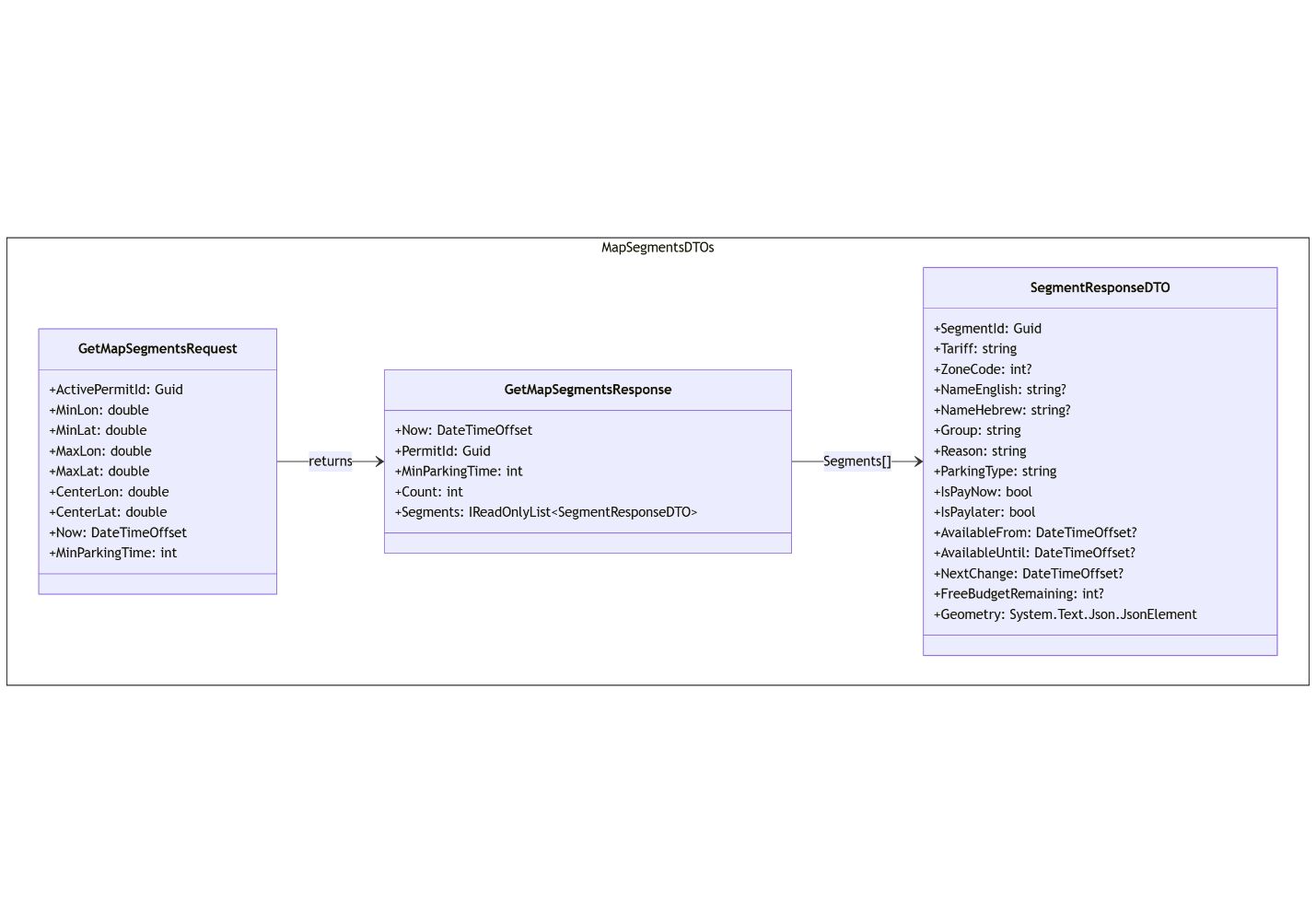
מחלקת העברת מידע על בקשות הוספת / מחיקת / עדכון רכבים ברשות המשתמש.



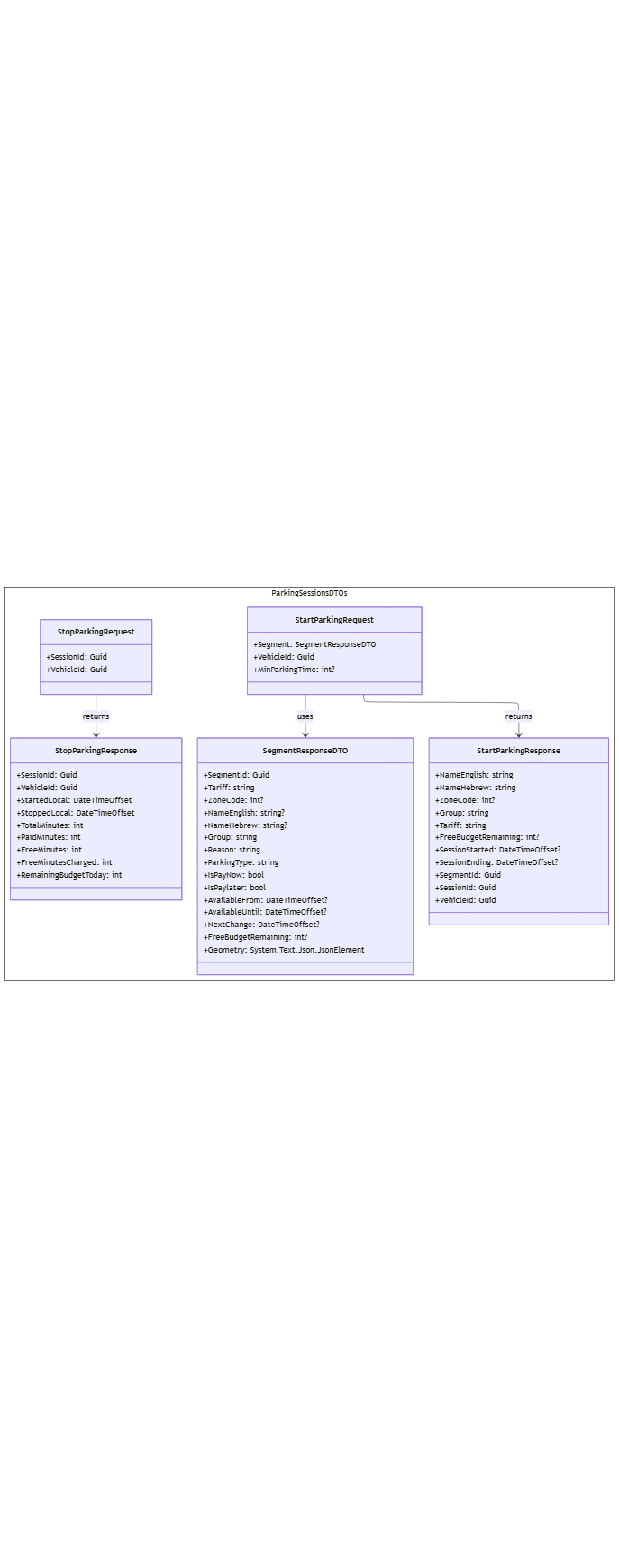
Permit DTOs

מחלקת העברת מידע על הוספת / מחיקת / עדכון תווים ברשות המשתמש.

Map Segment DTOs

מחלקת העברת מידע על רחובות בקרבת המשתמש.

Parking Session DTOs

מחלקת העברת מידע על התחלת / סיום חנייה.

**נקודות קצה (Endpoints)**

לAPI ישנו בסך הכל שישה נקודות קצה. כל נקודת קצה מכסה תחום אחר בהתקשרות עם מסד הנתונים (Auth / Vehicles / Map / Sessions / Permits). את נקודות הקצה ניתן לראות ולהבין ביתר בירור בעזרת OpenAPI לאחר הפעלתו בממשק <http://localhost:8080/scalar/>

בריאות המערכת

|  |  |  |
| --- | --- | --- |
| הערות | כתובת | פקודה |
| מחזיר סטטוס, גרסה, זמן ריצה וזמן מקומי (OK 200) | /health | GET |
| בדיקת PostGIS + DB ; מחזיר ירוק / אדום (200 / 503) | /ready | GET |
| מחזיר את גירסת האפליקציה (OK 200) | /version | GET |

/map – מקטעי רחוב ומפה

|  |  |  |
| --- | --- | --- |
| הערות | כתובת | פקודה |
| פלט: כלל המקטעים באיזור הנצפה | /map/segments | POST |

/auth - אימות וזהויות

|  |  |  |
| --- | --- | --- |
| הערות | כתובת | פקודה |
| רישום משתמש חדש | /auth/register | POST |
| התחברות | /auth/login | POST |
| חידוש זוג טוקנים | /auth/refresh | POST |
| התנתקות | /auth/logout | POST |
| פרטי המשתמש המחובר (בעיקר לצורך דיבוג) | /auth/me | GET |
| שינוי סיסמה | /auth/change-password | POST |

/vehicles – רכבים

|  |  |  |
| --- | --- | --- |
| הערות | כתובת | פקודה |
| פלט: רשימת רכבים של המשתמש | /vehicles | GET |
| פלט: רכב ספציפי לפי מזהה | /vehicles/{id} | GET |
| יצירת רכב | /vehicles | POST |
| עדכון חלקי לרכב | /vehicles/{id} | PATCH |
| מחיקת רכב | /vehicles/{id} | DELETE |

/permits – היתרים

|  |  |  |
| --- | --- | --- |
| הערות | כתובת | פקודה |
| יצירת היתר חדש לרכב | /permits | POST |
| פלט: היתר לפי מזהה | /permits/{id} | GET |
| עדכון היתר | /permits/{id} | PATCH |
| מחיקת היתר | /permits/{id} | DELETE |

/parking – חניה (סשנים ותקציב)

|  |  |  |
| --- | --- | --- |
| הערות | כתובת | פקודה |
| פלט: רשימת סשנים פעילים | /parking/sessions | GET |
| פלט: אובייקט סטטוס | /parking/status/{id} | GET |
| פלט: משך זמן נותר לחנייה יומית בחינם | /parking/budget-remaining/{id} | GET |
| פלט: סטטוס 201 CREATED | /parking/start | POST |
| פלט: סטטוס 200 OK | /parking/stop | POST |

**שירותים – Services**

האפליקציה בנויה לפי Composition Root + Feature Modules. כל פיצ'ר מגדיר / מספק מתודות הרחבה IServiceCollection / IendpointRouteBuilder, והקובץ הראשי Program.cs מרכיב את המערכת בקריאות קצרות. עבור כל שירות מוגדר ממשק (Interface) ומימוש קונקרטי, בסגנון Strategy/Facade, כך שניתן להחליף לוגיקה בקלות ולבודד בדיקות.

* הוספת Dependencies מתבצעת בתוך הרחבת פיצ'ר ולא ב-Program.cs.
* ברירת המחדל לשירותי דומיין היא Scoped כך שלכל בקשה יווצר מופע נפרד.

קווי מדיניות ל-Lifetimes

* DbContext וDependencies בתוכו Scoped.
* כלי עזר Stateless (כגון Clock, Hashing, JWT) Singleton.
* Hosted Services שיוצרים Scopes פנימיים בעת גישה ל-DB Singleton Workers.

שכבת Infrastructure

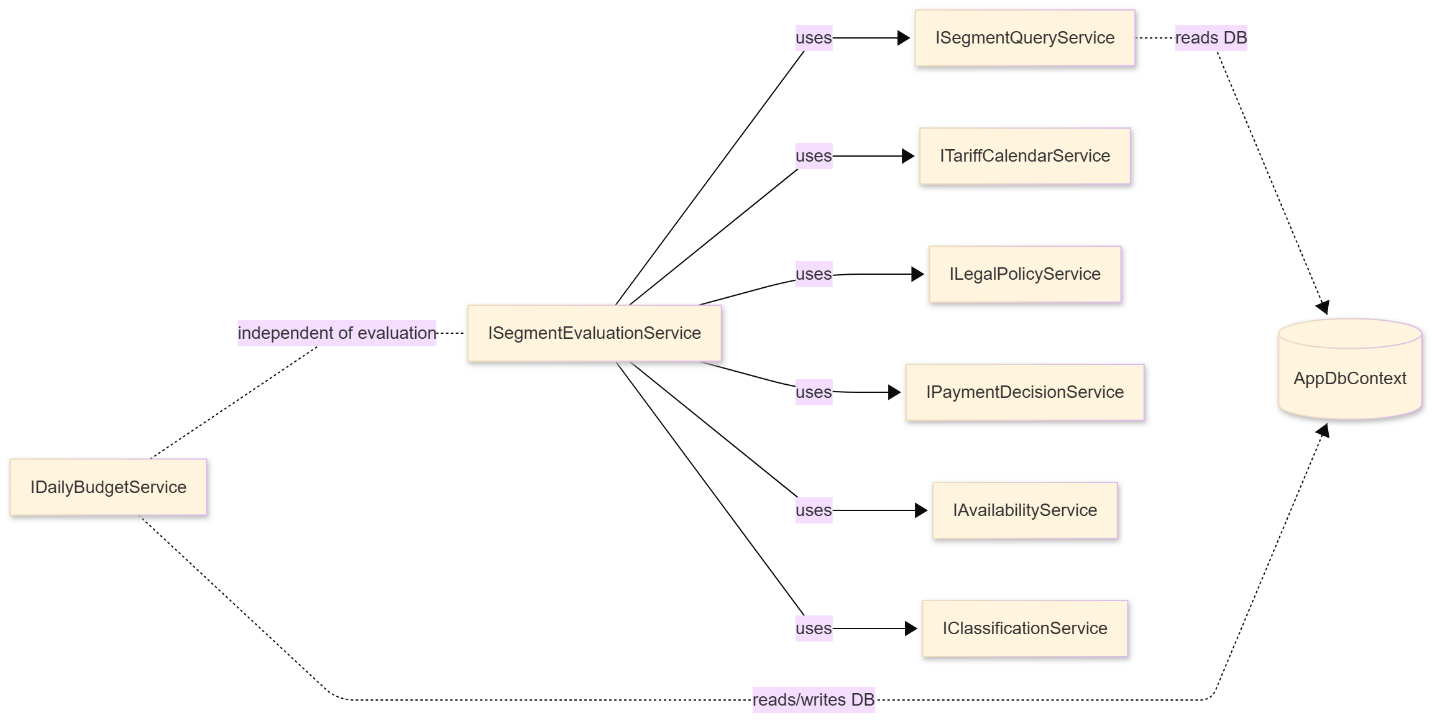
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Service | מטרה | ממשק | מימוש | Lifetime | הערות |
| Database | גישה ל-EF Core | - | AppDbContext | Scoped | נרשם ב-AddDbContext |
| Data Seeding | Database Seeding | - | SeedRunner | Hosted | יוצר Scope פנימי בכל הרצה |
| Runtime Health | Uptime / Version | - | RuntimeHealth | Singleton | סטטי וללא תלות ב-DB |
| Time Provider | נגישות לזמן מערכת | - | TimeProvider.System | Singleton | מוזרק לשירותים הזוקוקים לניהול זמן |
| API Documentation | Swagger/Scalar Endpoints Explorer | - | OpenAPI | - | הגדרות גלובליות ל-API |

שכבת Auth

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Service | מטרה | ממשק | מימוש | Lifetime | הערות |
| Authentication options | תצורה מרוכזת לאימות / חתימה | - | AuthOptions | - | - |
| Password hashing | האשים בטוחים | IPasswordHasher | Argon2PasswordHasher | Singleton | ללא תלות ב-DB |
| Access tokens | יצירה ואימות טוקנים | IJwtService | JwtService | Singleton | - |
| Refresh-token store | ניהול Refresh Tokens | - | EfRefreshTokenStore | Scoped | ניגש ל-DB |
| Refresh-token service | לוגיקה עבור Refresh Tokens | IRefreshTokenService | RefreshTokenService | Scoped | תלוי באחסון / DbContext |
| Authentication / Authorization pipeline | מדיניות + Bearer | - | JwtBearer / Authorization | - | - |

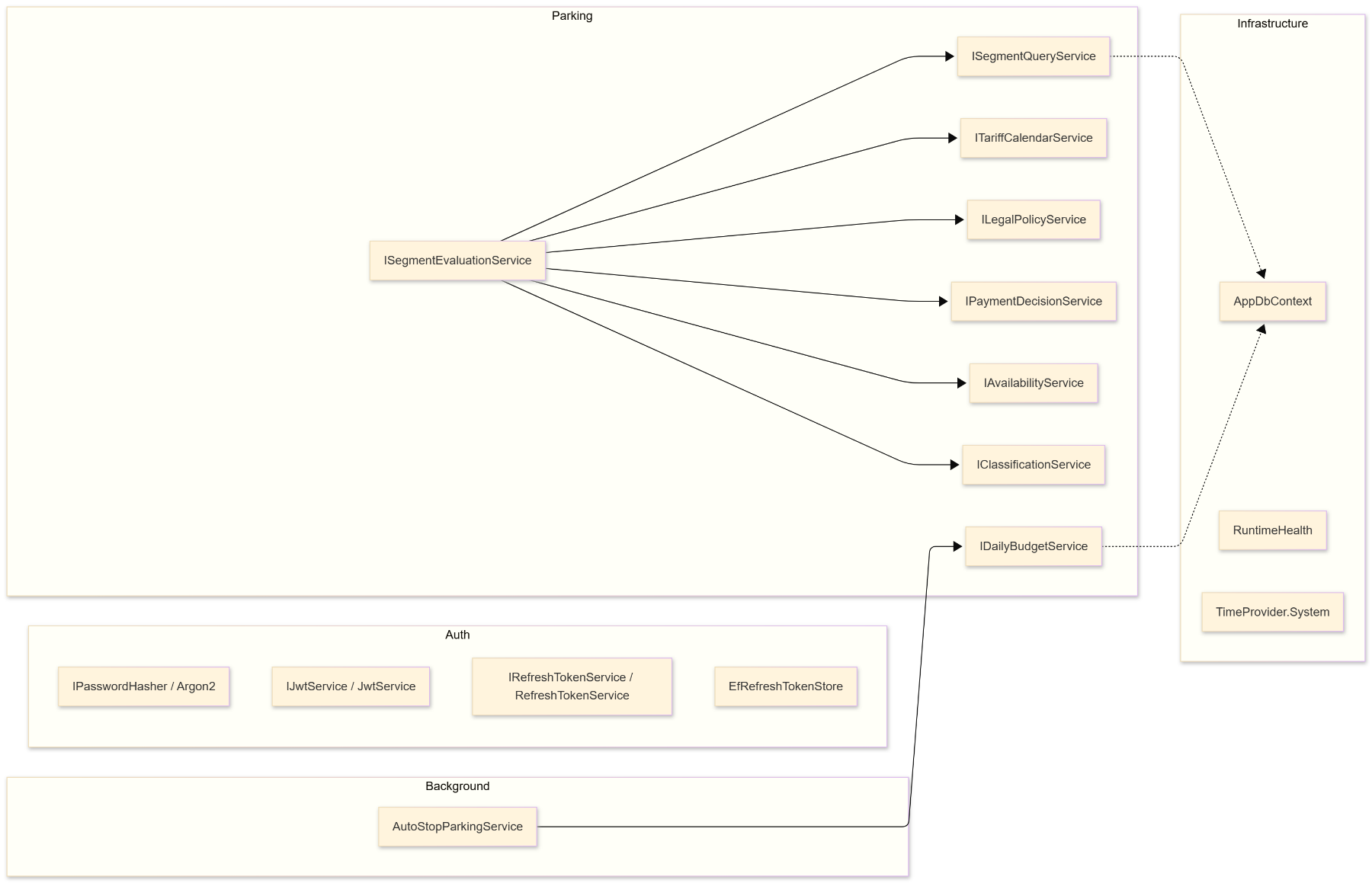
שכבת Parking (Business Logic)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Service | מטרה | ממשק | מימוש | Lifetime | הערות |
| Segment queries | שליפת קטעי רחוב / איזורים | ISegmentQueryService | SegmentQueryService | Scoped | תלוי ב-DbContext |
| Tariff calendar | חלונות חוקי חנייה יומיים | ITariffCalendarService | TariffCalendarService | Scoped | Timed Events |
| Legal policy | זכאות ומגבלות חנייה | ILegalPolicyService | LegalPolicyService | Scoped | - |
| Payment decision | קבעיה אם תשלום נדרש | IPaymentDecisionService | PaymentDecisionService | Scoped | מבוסס פרופיל / תעריף |
| Daily budget | איתחול, צריכה ויתרה של תקציב יומי | IDailyBudgetService | DailyBudgetService | Scoped | מנהל מגבלת דקות יומית |
| Availability | חישוב זמינות | IAvailabilityService | AvailabilityService | Scoped | - |
| Classification | סיווג רחוב בהתאם לפרופיל משתמש | IClassificationService | ClassificationService | Scoped | - |
| Segment evaluation | הערכה מקצה לקצה | ISegmentEvaluationService | SegmentEvaluationService | Scoped | - |
| Auto-stop worker | עצירת סשנים בזמן מוגדר | - | AutoStopParkingService | Hosted | יוצר Scope פנימי לכל פעולה |



שכבת API / Pipeline

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Service | מטרה | ממשק | מימוש | Lifetime | הערות |
| Problem details (Error handling) | מיפוי חריגות | - | ProblemDetails Middleware | - | חלק מהPipeline |
| Request logging | Logs & Traces | - | RequestLogging / Tracing | - | חלק מהPipeline |
| API documentation | מסמכי OpenAPI | - | Scalar/Swagger | - | - |
| System clock | נגישות לזמן | IClock | SystemClock | Singleton | כלי עזר |



**מחלקות עזר שונות**

OpenApiExtensions – מחלקת הרחבות לריכוז תיעוד ה-API והצגת Scalar/Swagger. מרכז את כל ההגדרות במקום אחד.

AppPipelineExtensions – מחלקת הרחבות המגדירה את צנרת היישום: אימות והרשאות, מיפוי שגיאות לתשובות סטנדרטיות, תיעוד בקשות ו-Tracing. שומרת על Program.cs נקיא וקריא.

InfrastructureExtensions – מחלקת הרחבות עבור רכיבי התשתית: רישום הDatabase (EFCore עם PostGIS), אתחול ה-Seeder ברקע, חיבור לLogger, ועזרי מערכת כמו Clock ו-Runtime Health. מספקת נקודת ריכוז אחת לכל התשתיות.

AuthExtensions – מחלקת הרחבות לרישום והגדרת שכבת האימות והרשאות: גיבוב סיסמאות, שירות חתימה, אימות ורענון Tokens. מעניקה קריאה אחת שמרכיבה את כל ררכיבי האבטחה.

ParkingExtensions – מחלקת הרחבות לרישום שירותי הדומיין של החניה. כולל שירותי Query, לוח תעריפים וחוקים, החלטת תשלום, זמינות סיווג והערכת מקטעים. מוסיפה גם שירות רקע לעצירה אוטומטית, כך שכל לוגיקת החנייה מרוכזת בקריאה יחידה.

אפליקציית ParkSpotTLV צד המשתמש

האפליקציה מיודעת למשתמש העיקרי של המערכת – הלקוח, מחפש החנייה. פלטפורמט הייעוד הראשונית היא למכשירי אנדרואיד בלבד.

האפליקציה מציעה רישום מהיר ופשוט, ומאפשר כניסה מהירה לאפליקציה בעת כניסה חוזרת לאחר הרשמה.

בשימוש ראשון באפליקציה יעלו המסכים הבאים:

* עמוד Splash
* עמוד טעינה - בו נעשה ניסיון התחברות מהירה
* עמוד התחברות למערכת
* אפשרות למעבר לעמוד רישום למערכת
* לאחר התחברות/רישום - מסך ראשי - מסך המפה

האפליקציה תאפשר התחברות מהירה עבור משתמשים שכבר מחוברים למערכת ונכנסים שוב לאפליקציה. ההתחברות המהירה תלויה בזמן האחרון בו המשתמש היה מחובר. בהתחברות מהירה מוצלחת - המשתמש יעבור באופן מידי למסך הראשי - מסך המפה.

מסך המפה הוא המסך הראשי של האפליקציה - הוא מציג מפה דינמית בה מופיעים סימונים צבעוניים על הרחובות המסמלים את כללי החניה בזמן אמת. במידה והמשתמש אפשר שימוש במיקום - המפה תתמרכז על מיקום המשתמש הנוכחי. המשתמש יכול להפעיל אפשרות מעקב אחר תנועתו, ובכך המפה תתמרכז סביב המשתמש גם בזמן תזוזה.

במסך זה מתאפשר לערוך את הגדרות המפה (בחירת רכב, זמן ושעה על פיהם יוצגו צבעי הרחובות, וגם אפשרות סינון לצבעי הרחובות), להפעיל סשן חניה, לעבור לתשלום באמצעות אפליקציית תשלום החניה – Pango - וסיום חניה.

מסכים נוספים באפליקציה אליהם ניתן לנווט באופן קל ופשוט דרך כפתורי תפריט -

* מסך פרטי משתמש - מציג פרטי חשבון ורשימת רכבים
* מסך עריכת רכב - מאפשר לערוך פרטי רכב רשום
* מסך הוספת רכב - מאפשר להוסיף רכב לרשימת הרכבים של המשתמש
* מסך העדפות - מאפשר לעדכן העדפות בנוגע לתצגות החניה
* יציאה מהמערכת - log out.

**הרשאות**

לפעילות אופטימלית על המשתמש לאפשר שימוש האפליקציה בשירותי מיקום.

**ארכיטקטורה**

צד האפליקציה מורכב מכמה גורמים עיקריים:

עמודים – Pages

* קבצי xaml - מבנה בסיסי של העמודים המוצגים למשתמש
* קבצי xaml.cs - מתארים את התנהגות העמודים המוצגים למשתמש - טעינת מידע, לחיצה על כפתורים ועוד'.

מחלקות שירותים - Services

* מחלקות מיועדות לתקשורת מול בסיס הנתונים והשרת
* מחלקות המיועדות לחישובי עזר ומתודות עזר לניהול ה-UI

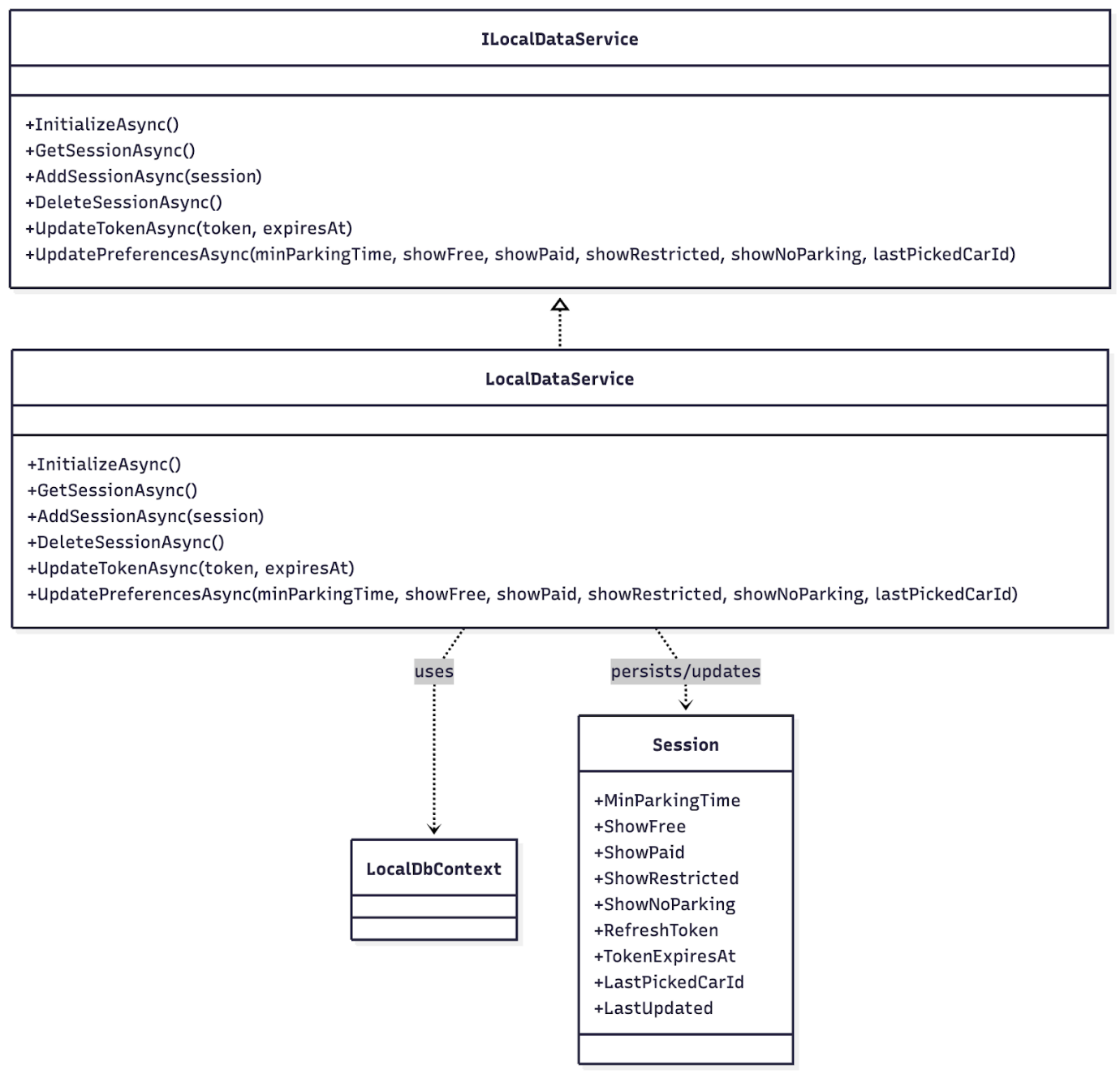
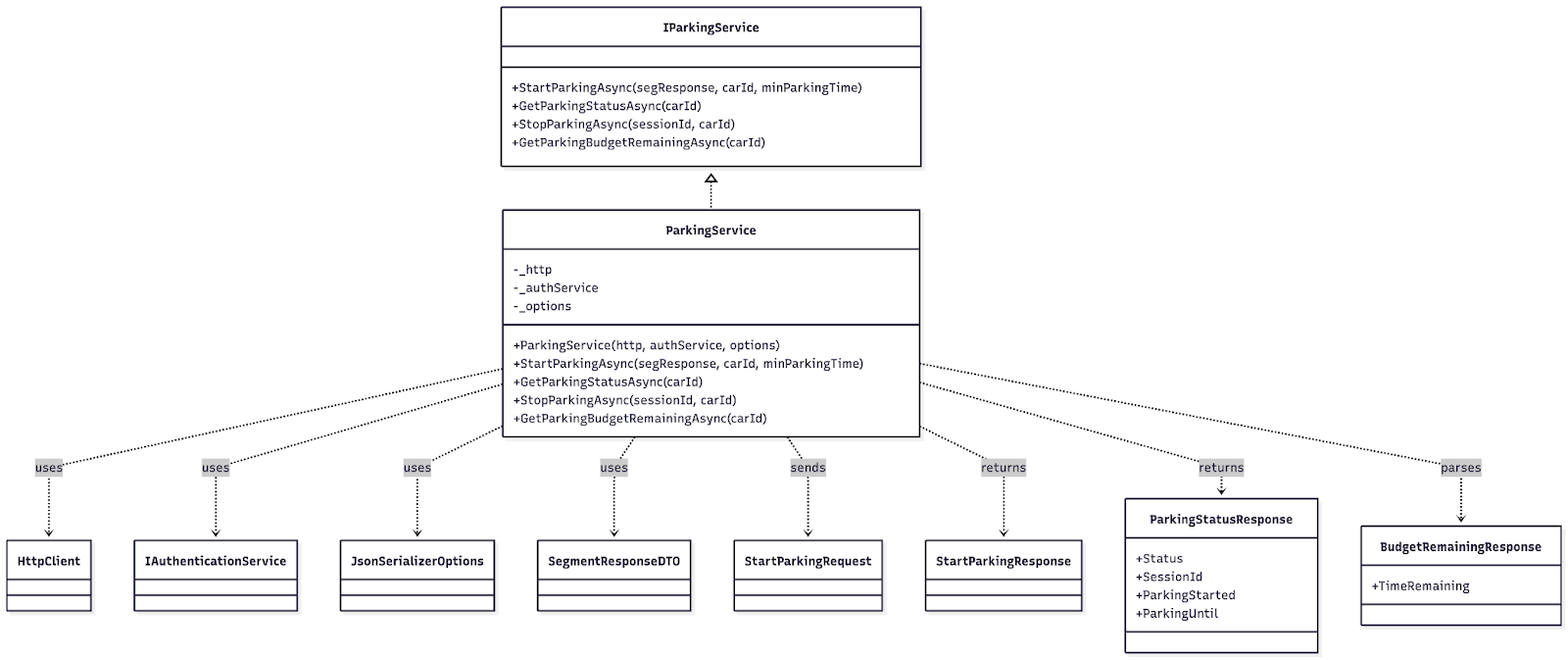
בסיס נתונים מקומי

* בסיס נתונים קל ומקומי השומר על נתוני אותנטיקציה של המשתמש הנוכחי, על העדפות המשתמש הנוכחי ועל נתונים נוספים המאפשרים ניהול תקין ומהיר של האפליקציה.

**מחלקות מידע משותפות**

**מחלקות השירות Services**

**דיאגרמת ממשקים ומחלקות שירות**

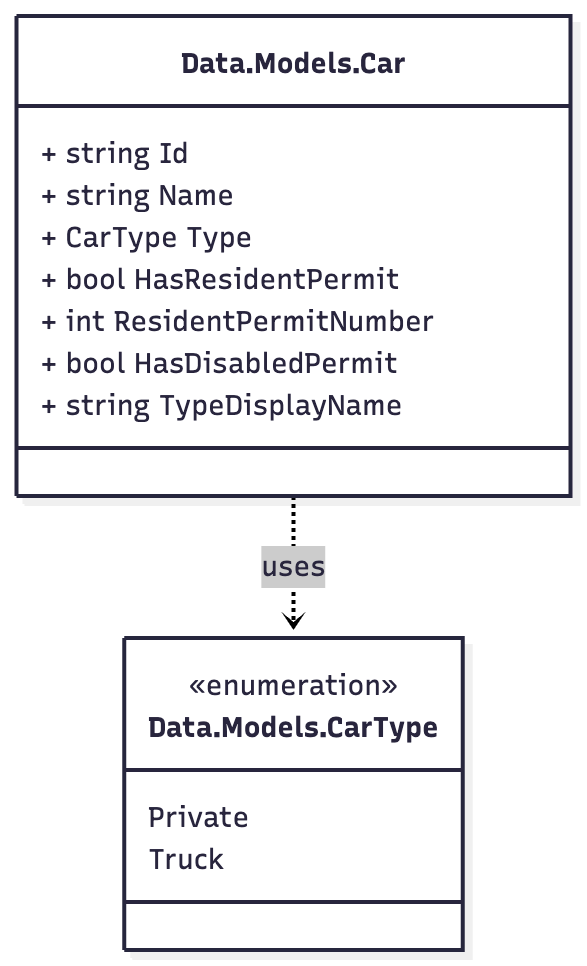
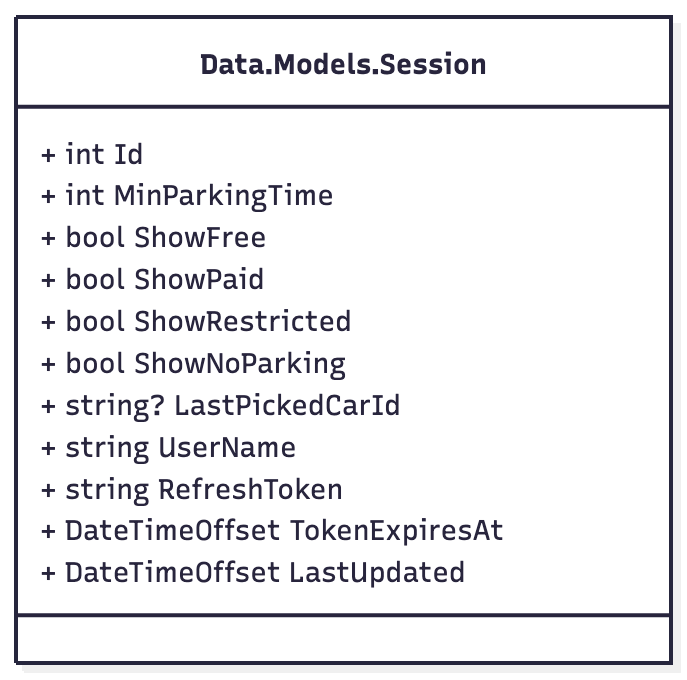
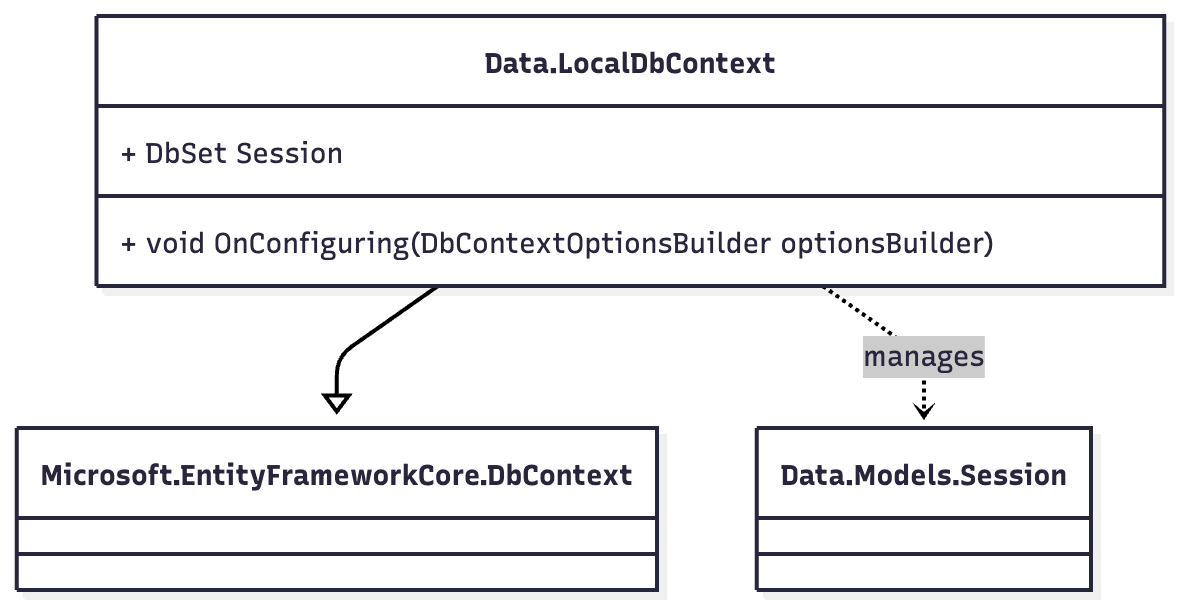


**מחלקות עזר – Helpers ודיאגרמות**

**מחלקות Components ודיאגרמות**

**מחלקות Model**

**דיאגרמות מחלקות המודל**



דפוסי עיצוב – Design patterns

**Inversion of Control / Dependency Injection (IoC/DI)**

דפוס עיצוב שאחראי על ניתוק תלותים ישירים בין מחלקות באמצעות רישום ממשקים (Interfaces) והזרקת מימושים דרך ה-IoC Container של ASP.NET/MAUI. בעזרת דפוס עיצוב זה ניתן לבדוק, להחליף ולרכיב את המערכת בצורה מודולרית.

בפרויקט, שרת האפליקציה ממומש בעזרת דפוס עיצוב, ובא לידי ביטוי בכמה מקומות:

* קובץ Program.cs משמש כ-Composition Root: כל רישומי ה-DI, ה- Middleware, והקונפיגורציה נאספים בו. בשורות קצרות וקריאות אנו מרכיבים את המערכת ומפרידים אחריות לוגית לקבצע Extension.

AddInfrastructure() – רישום AppDbContext, קומפיגורציות EF וConstraints כלליים.

AddAuthFeature() – הגדרת JwtBearer, רישום JwtService ומדיניות הרשאות.

AddParking() – רישום שירותי דומיין והתנהלות עם חניות. בנוסף אחראי על רישום AutoStopParkingService כ-Hosted Service.

בנוסף, נעשה שימוש בדפוס זה גם בקובץ MauiProgram.cs באפליקצית המשתמש. שם נרשמים רכיבים כמו CarService, AuthService, LocalDataService במערכת ההזרקה. לאחר מכן, כאשר אחד העמודים זקוק לשירות מסוים, הוא מקבל אותו אוטומטית באמצעות ההרקה, מבלי ליצור אותו בעצמו.

**Singleton**

דפוס עיצוב שמבטיח מופע יחיד של מחלקה לאורך חיי האפליקציה. ממומש באמצעות AddSingleton() או באמצעות Hosted Service יחיד. אנו משתמשים בsingleton לשירותים stateless, רגישי ביצועים או כאלה שעליהם להישאר יציבים לאורך זמן.

בפוריקט, שרת האפליקציה מממש את דפוס זה בכמה מקומות:

* JwtService – נרשם כAddSingleton. שירות זה מספיק חתימה / אימות של טוקנים.
* Argon2PasswordHasher – נרשם כAddSingleton. שירות זה מחזיק פונקציות גיבוב דטרמיניסטיות וללא state פנימי, ולכן אינן דורשות מופעים מרובים. שמירת מופע יחיד היא טבעית עבור שירות זה.
* AutoStopParkingService – נרשם כHostedService יחיד שמבצע כל מחזור (כל 30 שניות):

- פתיחת Scope פנימי לכל ריצה בעזרת IServiceScopeFactory כדי לקבל AppDbContext וscoped dependencies כמו IDailyBudgetService לצורך חישוב וסגירת סשנים פעילים.

- איתור סשנים פעילים שהגיע זמנם להיעצר לפי כללים שהגדרנו מראש, ועצירתם.

**Builder**

דפוס עיצוב שמרכיב אובייקט מורכב בשלבים דרך API זורם. בפרויקט אנו משתמשים בשלושה Builders עיקריים: WebApplicationBuilder של ModelBuilder, ASP.NET Core של EF Core (ביחד עם IEntityTypeConfiguration), ו-MauiApp.CreateBuilder() של .NET MAUI.

EF Core Model Builder

* AppDbContext.OnModelCreating מפעיל ApplyConfigurationsFromAssembly כדי לבנות את המודל מכל המחלקות שמיישמות את IEntityTypeConfiguration<T>.
* AppDbContextFactory בונה DbContextOptions.
* כל קובץ קונפיגורציה (…/Infrastructures/Config) תורמת רכיב אחד לבמנה הDatabase.

WebApplicationBuilder

* var builder = WebApplication.CreateBuilder(args) – שלב איסוף קונפיגורציות, רישומי DI כגון Controllers, Hosted Services, DbContext, Auth.
* var app = builder.Build() – יצירת היישום מתוך ה-Builder.
* app.Use(..) / app.Map(…) – שלבי הרכבה סופיים (Middlewares, Endpoints)
* app.Run() – הפעלת האפליקציה.

לבסוף מוחזר האוביקט המוכן (WebApplication).

**Observer**

דפוס זה נועד לאפשר למספר רכיבים במערכת להגיב לשינוי שמתרחש באוביקט מסויים, מבלי שיהיה ביניהם קשר ישיר או תלות הדוקה. במילים אחרות – כשאוביקט אחד משתנה, כל האוביקטים ש"צופים" בו מקבלים עדכון אוטומטי.

דפוס זה בא לידי ביטוי באפליקצית המשתמש בשתי מקומות:

דוגמה 1:

המשתמש גורר/מגדיל את המפה Map.VisibileRegion משתנה השירות מאזין ל-map.PropertyChanged מופעל VisibleBoundsChanged העמוד קולט את האירוע ומבצע FetchAndRenderSegments כדי להביא ולצייר סגמנטים חדשים.

דוגמה 2:

כשה-GPS מדווח על מיקום חדש השירות מקבל אירוע Geolocation.LocationChanged

מופעל OnLocationChanged אשר ממקד את המפה למיקום העדכני.

שינויים אפשריים עתידיים

נציג מספר שינויים עתידיים אפשריים

**התראות לפני גמר חנייה**

בכדי לממש את מנגנון ההתראות פרק זמן מוגדר לפני גמר סשן חנייה נוסיף את הרכיבים הבאים:

שרת האפליקציה

1. נקודת קצה חדשה ב-API:

POST /notifications/register-device – רישום עדכון Device Token לפי משתמש/מכשיר.

POST /notifications/preferences – שמירת העדפות (דקות התראה לפני סיום, quiet hours וכו')

DELETE /notifications/register-device – הסרת מכשיר (בעת Logout או מחיקת התקנה)

1. טבלאות חדשות ב-DB/EF:

DeviceRegistration (UserId, DeviceId, FcmToken, Platform, UpdatedAtUtc, IsActive)

NotificationPreference (UserId, LeadMinutesDefault,QuietHoursStart/End, PushEnabled)

ScheduledNotification (Id, ParkingSessionId, FireAtUtc, Type=BeforeEnd, Status = Pending / Sent / Cancelled, PayloadJson)

1. Services חדשים:

INotificationService + FirebaseNotificationService – בניית הודעות, שליחה ל-FCM, טיפול בשגיאות וריענון Tokens.

INotificationScheduler – ירוץ כHostedService ברקע ויתזמן יצירה/עדכון/ביטול התראות בעת יצירה, עדכון או עצירה מוקדמת של parking sessions.

NotificationDispatcher – ירוץ כHostedService ברקע, כל X זמן (לפי מה שנגדיר), מושך ScheduledNotification עם FireAtUtc now ו-Status=Pending, ושולח את ההתראה.

אפליקצית המשתמש

1. הטמעת FCM Client ו-FirebaseMessagingService:

רישום / ריענון FcmToken ושליחתו ל-notifications/register-device.

קליטת הודעות (SessionId, Remaining, PlannedEndUtc, Action = OpenSession/Summary)

1. מסך העדפות התראות (תוספת ל-Preferences הקיימים):

Lead minutes – הצבת ברירת מחדל למשתמש, ולתת למשתמש אפשרות להגדיר Quiet Hours /

מתג הפעלה/כיבוי להתראות Push.

סנכרון לאחר מכן לשרת.

1. הוספת פרמטר ה-Lead minutes לאחסון המקומי.

**לחיצה על רחוב תראה את פירוט כללי החניה ברחוב זה**

בכדי לממש את הפונקציונאליות של פירוט כללי וסטטוס חנייה ברחוב ספציפי נוסיף את המימושים הבאים:

שרת האפליקציה

* 1. הוספת הרחבה לנקודות קצה קיימות (קבוצת map/segments):

GET /map/segments/{segmentId}/details – מחזיר פירוט למקטע ספציפי.

GET /map/segments/batch-details?ids – אופציונלי, תלוי מימוש שנבחר, האם להחזיר לרחוב ספציפי כל פעם, או להחזיר כבר את כל הקבוצה שנטענה למשתמש על המסך.

* 1. שימוש חוזר בServices שכבר קיימים על מנת לשלוף ולחשב את המידע הנכון לנקודת זמן הרצויה.

אפליקצית המשתמש

1. הוספת tap handler לאוביקט מקטע במפה:

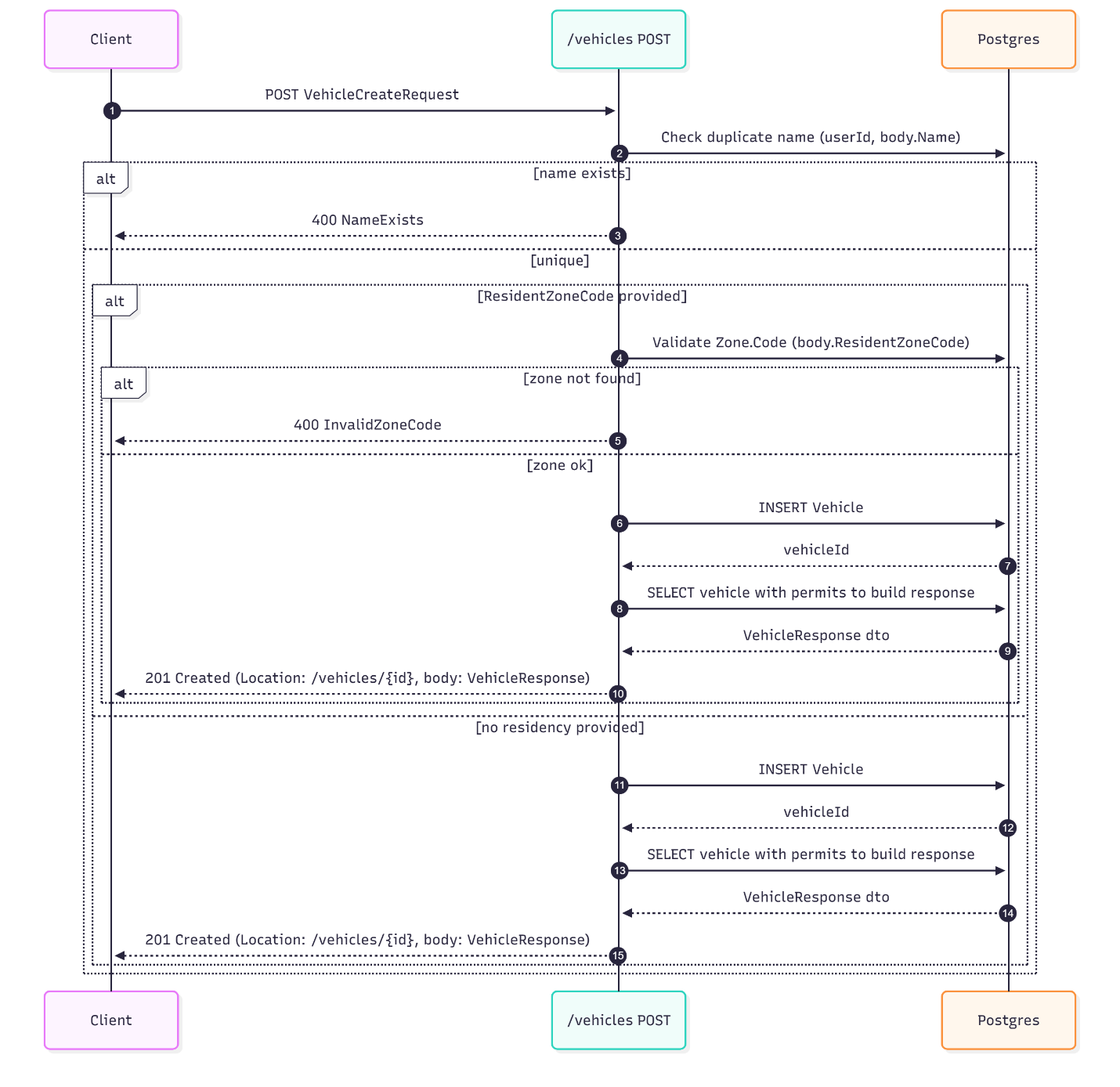
בביצוע Tap – שולח בקשת Get /map/segments/{id}/details ומציג Popup.

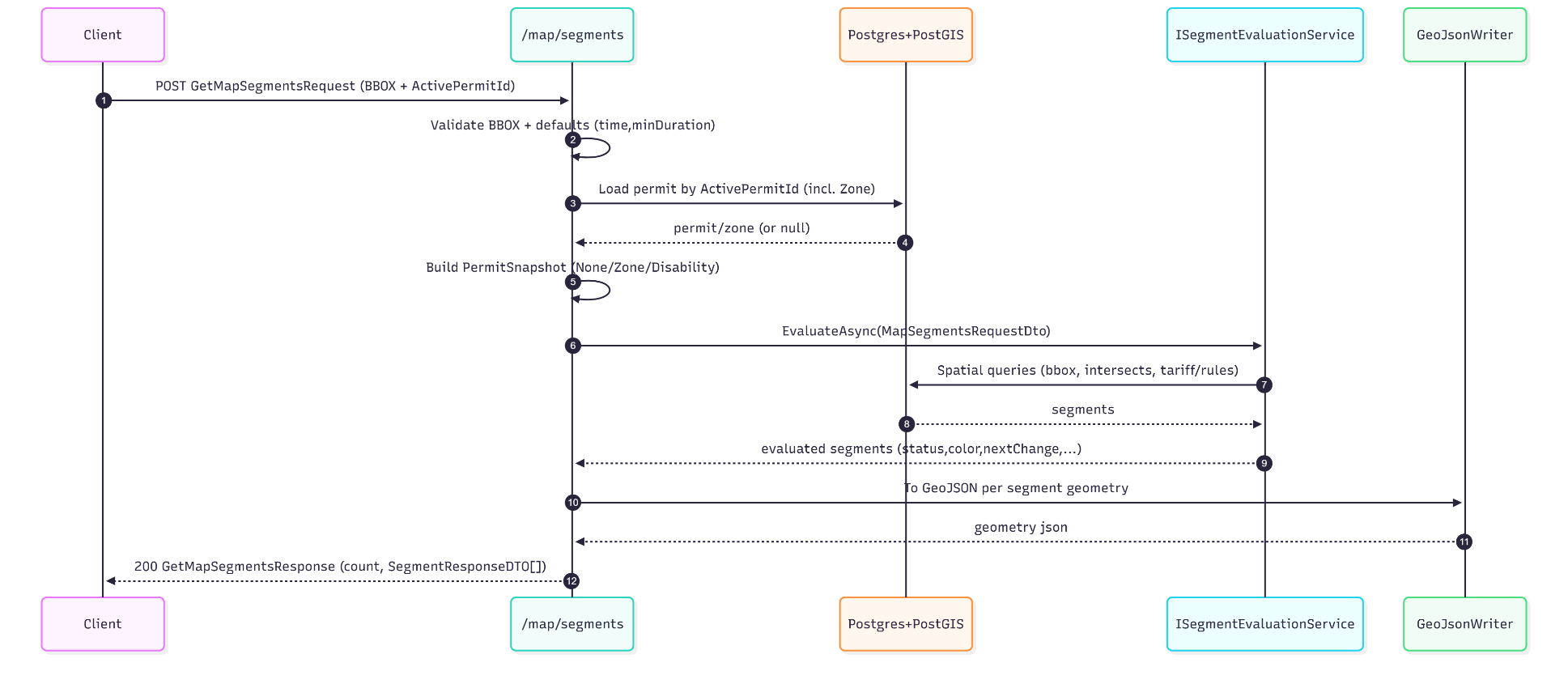
הצגת סטטוס, כותרת רחוב/איזור, טווחים להיום, זכאות לפי פרטי המשתמש, והאפשרות להתחיל חנייה. (פעולה מהירה)

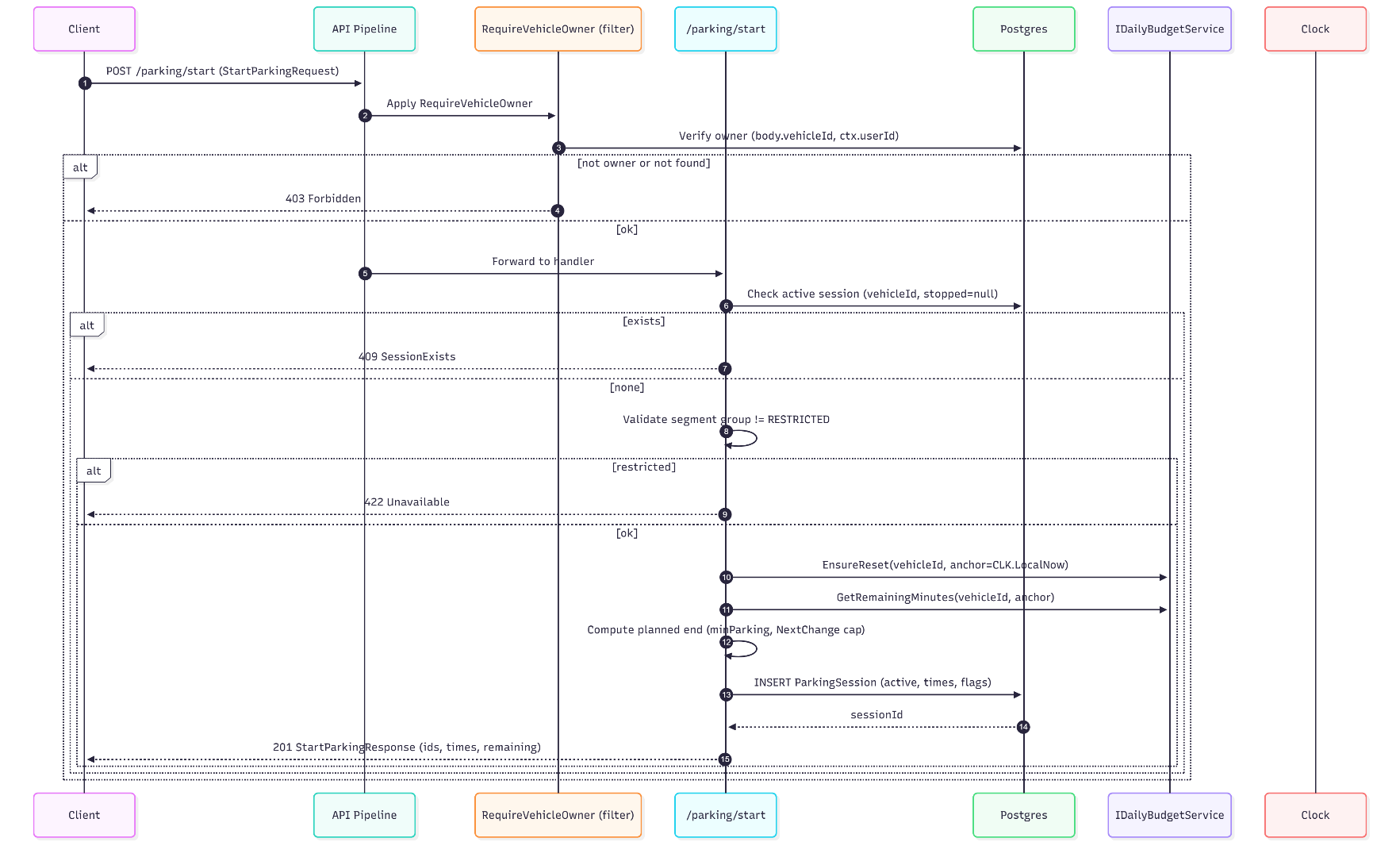
1. רענון דינמי של nextChange לעדכון צבע/טקסט בתוך ה-Bottom Sheet ללא סגירה.

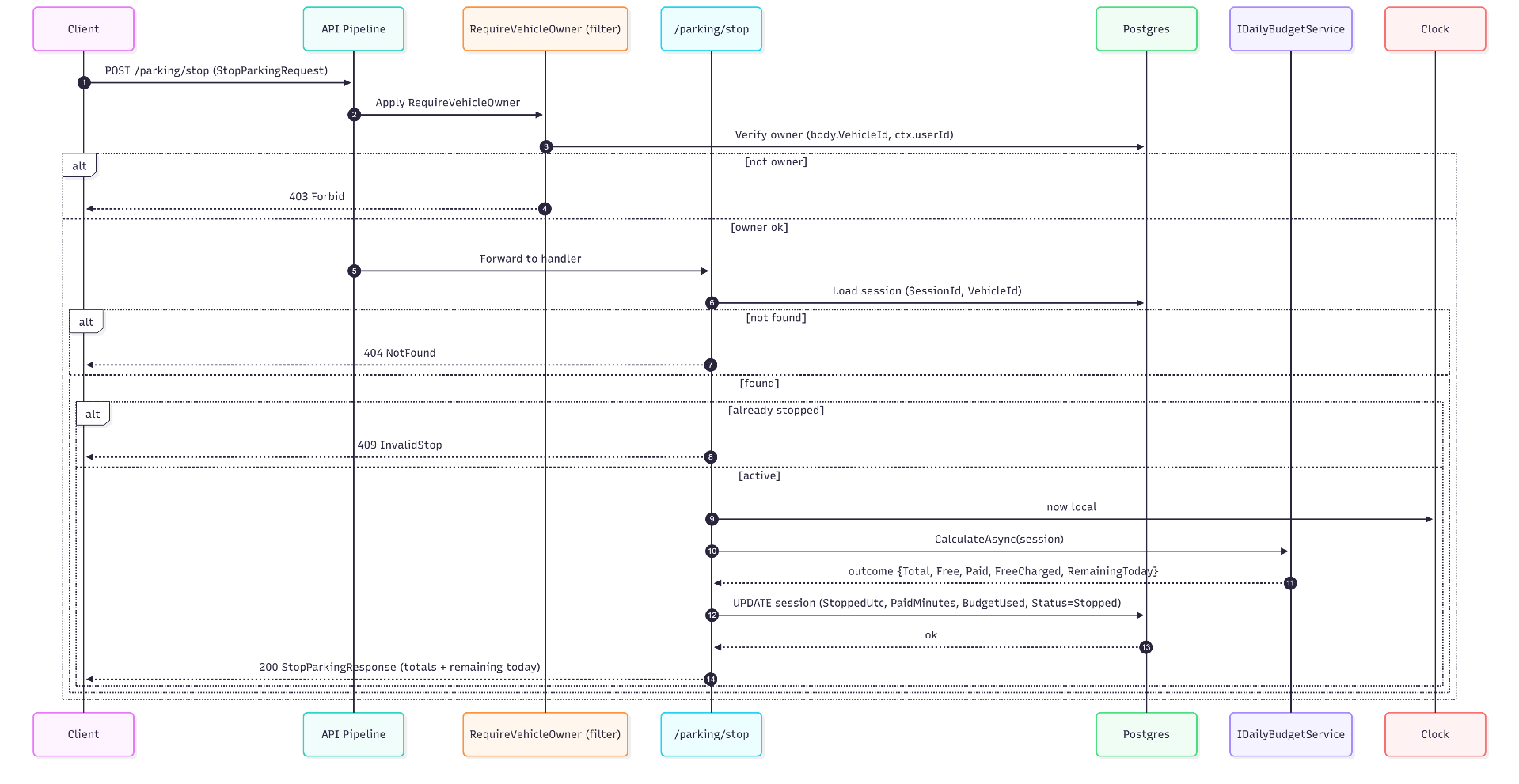
דיאגרמות רצף

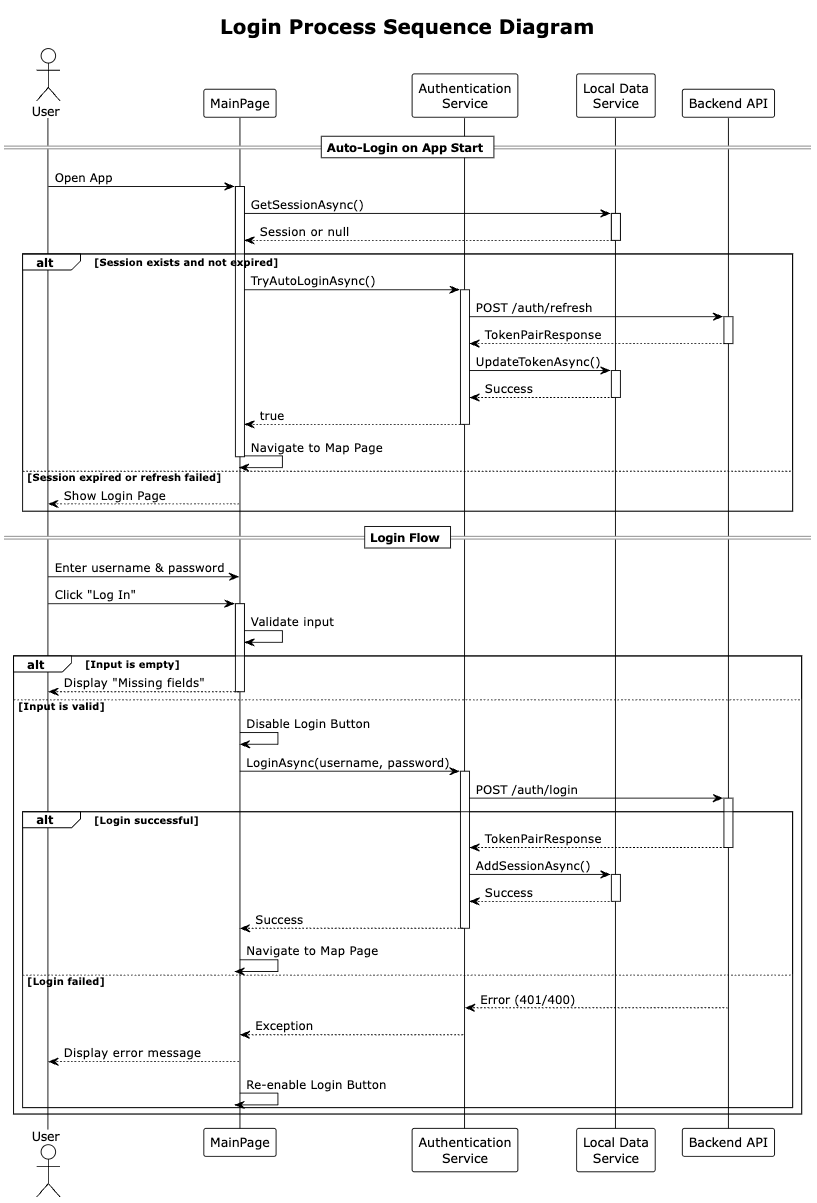
**רישום משתמש והנפקת זוג טוקנים – צד שרת POST /auth/register —**

**יצירת רכב (כולל בדיקות שם ואופציונלית היתרים) – צד שרת POST /vehicles —**

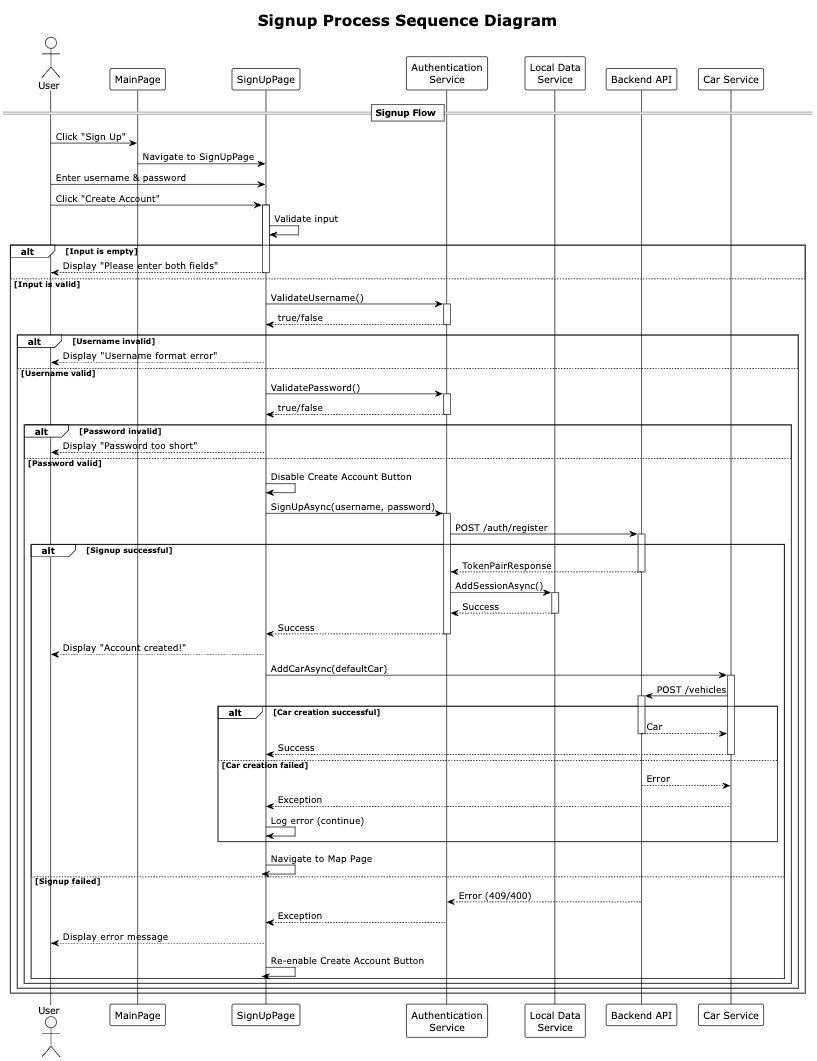
**שליפת מקטעים בbbox והערכה – צד שרת POST /map/segments —**

**פתיחת סשן חניה – צד שרת POST /parking/start —**

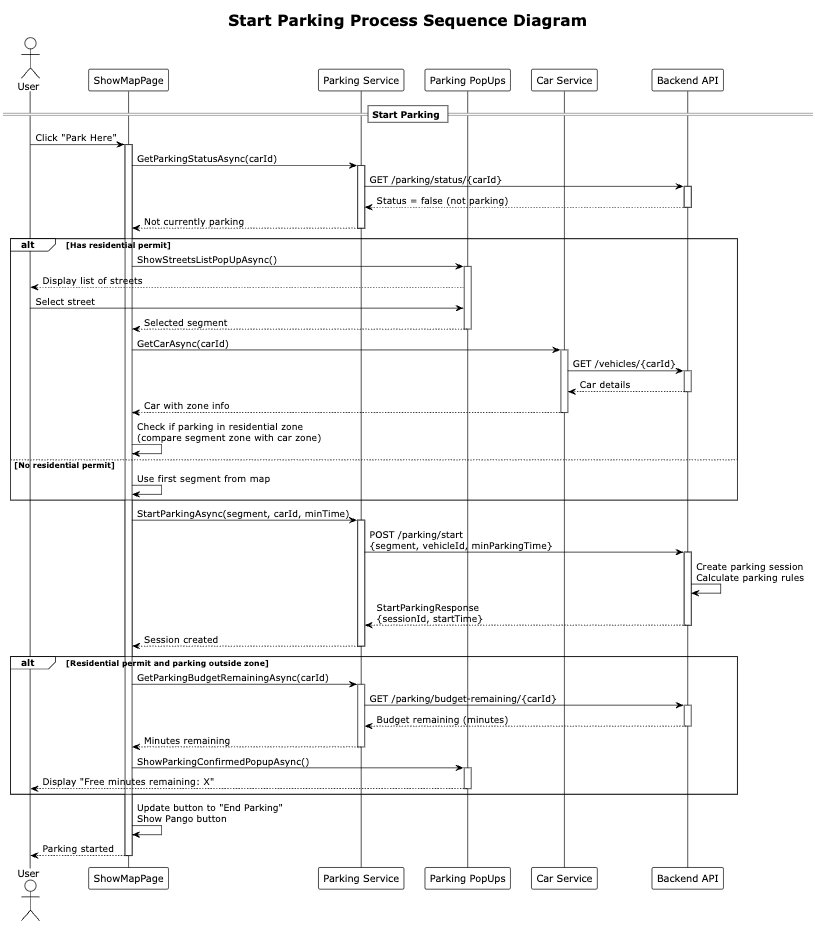
**סגירת סשן וחישוב חיובים/תקציב – צד שרת POST /parking/stop —**

**תהליך Log-in לאפליקציה – צד לקוח**

**תהליך הרשמה לאפליקציה – צד לקוח**



**תהליך התחלת חנייה – צד לקוח**



**תהליך עצירת חנייה – צד לקוח**

