DOCUMENTAȚIA PROIECTULUI

Car Tracking API

STUDENȚI

Socaci Victor

Teulea Ioan-Octavian

**Cuprins**

1. Prezentarea proiectului, ce isi propune, ce probleme rezolva

2. Tehnologiile folosite

3. Baza de date: diagrama bazei de date + scurta prezentare a tabelelor si a relatiilor dintre ele

4. Prezentarea API-ului: screenshot la Swagger pentru a putea vedea toate endpoint-ruile + scurta descriere a CRUD-urilor

5. Prezentare despre cum poate fi utilizata aplicatia: tipuri de utilizatori, ce vede fiecare, autentificare etc

6. Concluzii si contributii (cum v-ati impartit task-urile, ce ati invatat in urma acestui proiect etc)

7. Link GIT catre codul proiectului

Descrierea Proiectului

#### Tematica: Car Tracking API

Proiectul "Car Tracking API" este o aplicație web care oferă funcționalități pentru gestionarea și monitorizarea mașinilor, întreținerilor și centrelor de service asociate. Aceasta permite utilizatorilor să înregistreze și să urmărească datele referitoare la vehiculele lor, iar administratorilor să gestioneze toate datele din sistem.

#### Obiectivele Proiectului

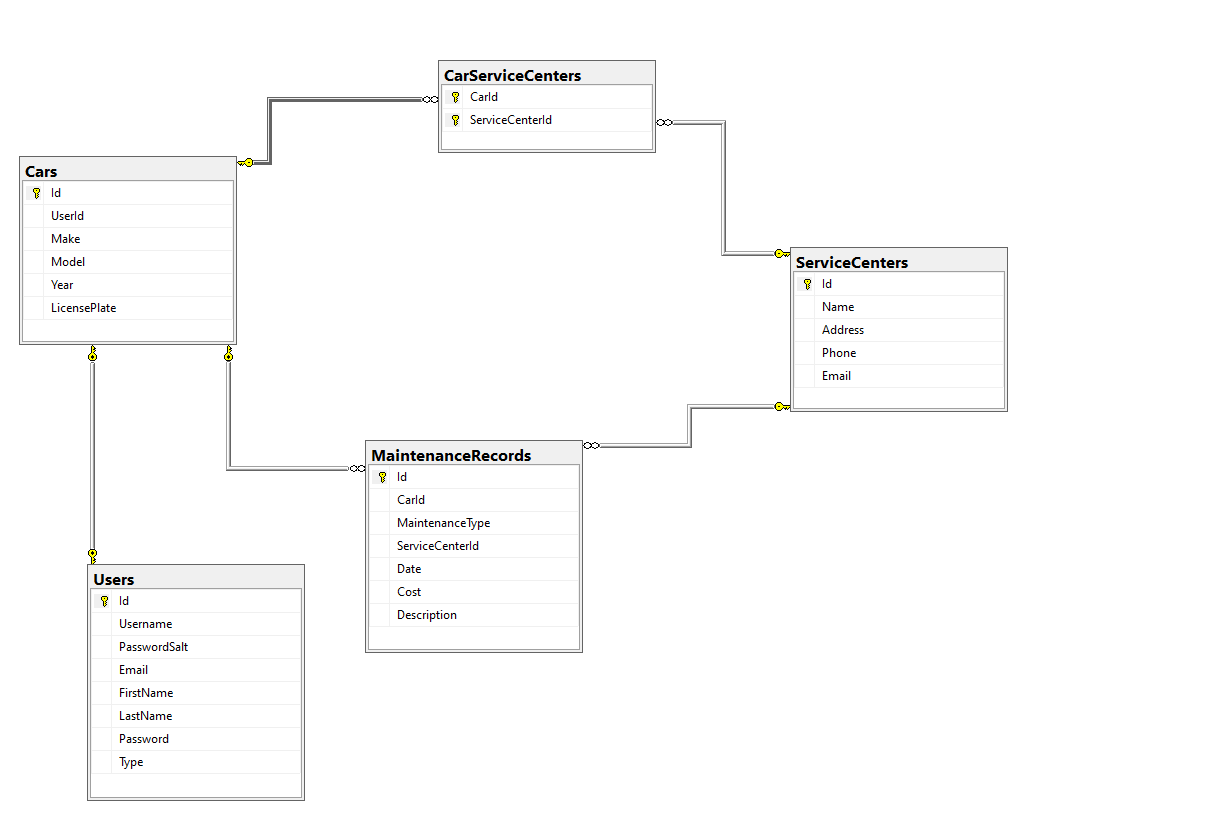
1. **Gestionarea Mașinilor:** Permite administratorilor să adauge, să editeze și să vizualizeze toate mașinile înregistrate. Utilizatorii obișnuiți pot vizualiza doar mașinile pe care le dețin.
2. **Monitorizarea Întreținerilor:** Administratorii pot adăuga, edita și șterge înregistrările de întreținere pentru toate mașinile. Utilizatorii obișnuiți pot vizualiza doar întreținerile mașinilor pe care le dețin.
3. **Gestionarea Centrelor de Service:** Administratorii pot adăuga și vizualiza toate centrele de service. Utilizatorii obișnuiți pot vizualiza centrele de service la care mașinile lor au fost întreținute.
4. **Autentificare și Autorizare:** Sistemul permite înregistrarea și autentificarea utilizatorilor, precum și atribuirea de roluri pentru a delimita accesul la anumite funcționalități.

#### Probleme Rezolvate

* **Centralizarea datelor:** Toate informațiile despre mașini, întrețineri și centre de service sunt centralizate într-o singură aplicație, facilitând gestionarea și accesul la date.
* **Siguranța datelor:** Utilizatorii au acces doar la datele care le aparțin, iar administratorii au acces complet pentru a asigura gestionarea eficientă a sistemului.
* **Ușurința gestionării întreținerilor:** Permite urmărirea întreținerilor efectuate asupra mașinilor, ajutând proprietarii să mențină mașinile în stare bună.
* **Vizualizarea centrelor de service:** Oferă informații despre centrele de service, inclusiv locație și date de contact, pentru a facilita planificarea întreținerilor.

**Tehnologiile folosite:**

1. **ASP.NET Core**
2. **Entity Framework Core**
3. **SQL Server**
4. **Swagger/OpenAPI**
5. **Authentication și Authorization**
6. **Dependecy Injection(DI)**
7. **Middleware**

**Diagrama bazei de date:**

Prezentare tabele si relatii:

1. **Users**

* **Atribute:**
  + Id: Identificator unic al utilizatorului.
  + Username: Numele de utilizator.
  + PasswordSalt: Salt-ul utilizat pentru hash-ul parolei.
  + Email: Adresa de email a utilizatorului.
  + FirstName: Prenumele utilizatorului.
  + LastName: Numele de familie al utilizatorului.
  + Password: Parola utilizatorului (hash-uită).
  + Type: Tipul utilizatorului (0 pentru Admin, 1 pentru Utilizator obișnuit).
* **Relații:**
  + Un utilizator poate deține una sau mai multe mașini (relație de 1 la N cu entitatea **Cars**).

1. **Cars**

* **Atribute:**
  + Id: Identificator unic al mașinii.
  + UserId: Identificatorul utilizatorului care deține mașina (cheie externă către **Users**).
  + Make: Marca mașinii.
  + Model: Modelul mașinii.
  + Year: Anul fabricației.
  + LicensePlate: Numărul de înmatriculare.
* **Relații:**
  + O mașină poate avea mai multe înregistrări de întreținere (relație de 1 la N cu entitatea **MaintenanceRecords**).
  + O mașină poate fi întreținută la mai multe centre de service (relație de M la N cu entitatea **ServiceCenters** prin entitatea intermediară **CarServiceCenters**).

1. **MaintenanceRecords**

* **Atribute:**
  + Id: Identificator unic al înregistrării de întreținere.
  + CarId: Identificatorul mașinii (cheie externă către **Cars**).
  + MaintenanceType: Tipul întreținerii efectuate.
  + ServiceCenterId: Identificatorul centrului de service (cheie externă către **ServiceCenters**).
  + Date: Data la care a fost efectuată întreținerea.
  + Cost: Costul întreținerii.
  + Description: Descrierea întreținerii.
* **Relații:**
  + Fiecare înregistrare de întreținere este asociată cu o mașină (relație de N la 1 cu entitatea **Cars**).
  + Fiecare înregistrare de întreținere este asociată cu un centru de service (relație de N la 1 cu entitatea **ServiceCenters**).

1. **ServiceCenters**

* **Atribute:**
  + Id: Identificator unic al centrului de service.
  + Name: Numele centrului de service.
  + Address: Adresa centrului de service.
  + Phone: Numărul de telefon al centrului de service.
  + Email: Adresa de email a centrului de service.
* **Relații:**
  + Un centru de service poate avea mai multe înregistrări de întreținere (relație de 1 la N cu entitatea **MaintenanceRecords**).
  + Un centru de service poate întreține mai multe mașini (relație de M la N cu entitatea **Cars** prin entitatea intermediară **CarServiceCenters**).

1. **CarServiceCenters**

* **Atribute:**
  + CarId: Identificatorul mașinii (cheie externă către **Cars**).
  + ServiceCenterId: Identificatorul centrului de service (cheie externă către **ServiceCenters**).
* **Relații:**
  + Aceasta este o entitate intermediară care permite relația de M la N între **Cars** și **ServiceCenters**. Fiecare pereche (CarId, ServiceCenterId) reprezintă o asociere între o mașină și un centru de service.

### Operatii CRUD în API



#### A screenshot of a computer Description automatically generated

#### Tipuri de Utilizatori

1. **Administrator (Admin)**
   * Rolul: 0
   * Permisiuni: Are acces complet la toate funcționalitățile aplicației. Acesta poate gestiona utilizatorii, mașinile, înregistrările de întreținere și centrele de service.
2. **Utilizator Obișnuit (User)**
   * Rolul: 1
   * Permisiuni: Are acces restricționat, limitat la gestionarea și vizualizarea propriilor mașini și a înregistrărilor de întreținere asociate.

#### Funcționalități și Permisiuni

##### Pentru entitatea **Car**:

1. **Get (Admin)**
   * Endpoint: GET /api/Cars/GetCars
   * Descriere: Returnează toate mașinile din baza de date.
   * Permisiuni: Doar admin (Rol 0).
2. **GetByCarOfOwner**
   * Endpoint: GET /api/Cars/GetCarOfOwner
   * Descriere: Returnează mașinile aparținând utilizatorului autentificat.
   * Permisiuni: Disponibil atât pentru admin (Rol 0), cât și pentru utilizatorii obișnuiți (Rol 1).
3. **AddCar (Admin)**
   * Endpoint: POST /api/Cars/AddCar
   * Descriere: Adaugă o nouă mașină în baza de date.
   * Permisiuni: Doar admin (Rol 0).
4. **EditCar (Admin)**
   * Endpoint: PUT /api/Cars/{car-id}/edit-car
   * Descriere: Modifică detaliile unei mașini existente.
   * Permisiuni: Doar admin (Rol 0).

##### Pentru entitatea **MaintenanceRecords**:

1. **GetMaintenances (Admin)**
   * Endpoint: GET /api/MaintenancesRecords/GetMaintenancesRecords
   * Descriere: Returnează toate înregistrările de întreținere.
   * Permisiuni: Doar admin (Rol 0).
2. **GetMaintenancesOfOwner**
   * Endpoint: GET /api/MaintenancesRecords/GetMaintenanceOfOwner
   * Descriere: Returnează înregistrările de întreținere pentru mașinile aparținând utilizatorului autentificat.
   * Permisiuni: Disponibil atât pentru admin (Rol 0), cât și pentru utilizatorii obișnuiți (Rol 1).
3. **AddMaintenance (Admin)**
   * Endpoint: POST /api/MaintenancesRecords/AddMaintenances
   * Descriere: Adaugă o nouă înregistrare de întreținere.
   * Permisiuni: Doar admin (Rol 0).
4. **EditMaintenance (Admin)**
   * Endpoint: PUT /api/MaintenancesRecords/{maintenanceId}/edit-maintenance
   * Descriere: Modifică detaliile unei înregistrări de întreținere existente.
   * Permisiuni: Doar admin (Rol 0).
5. **DeleteMaintenance (Admin)**
   * Endpoint: DELETE /api/MaintenancesRecords/delete-maintenance
   * Descriere: Șterge o înregistrare de întreținere existentă.
   * Permisiuni: Doar admin (Rol 0).

##### Pentru entitatea **ServiceCenters**:

1. **Get (Admin)**
   * Endpoint: GET /api/ServicesCenters/GetServicesCenters
   * Descriere: Returnează toate centrele de service.
   * Permisiuni: Doar admin (Rol 0).
2. **AddServiceCenter (Admin)**
   * Endpoint: POST /api/ServicesCenters/AddServiceCenter
   * Descriere: Adaugă un nou centru de service în baza de date.
   * Permisiuni: Doar admin (Rol 0).
3. **GetServiceOwner**
   * Endpoint: GET /api/ServicesCenters/GetServiceCenterOfOwner
   * Descriere: Returnează toate centrele de service la care a fost întreținută mașina utilizatorului autentificat.
   * Permisiuni: Disponibil atât pentru admin (Rol 0), cât și pentru utilizatorii obișnuiți (Rol 1).

##### Pentru entitatea **Users**:

1. **Register**
   * Endpoint: POST /api/User/register
   * Descriere: Înregistrează un nou utilizator în sistem.
   * Permisiuni: Disponibil pentru toți utilizatorii.
2. **Login**
   * Endpoint: POST /api/Users/login
   * Descriere: Autentifică un utilizator și returnează un token JWT.
   * Permisiuni: Disponibil pentru toți utilizatorii.
3. **GetAllUsers (Admin)**
   * Endpoint: GET /api/Users/GetUsers
   * Descriere: Returnează toți utilizatorii înregistrați în sistem.
   * Permisiuni: Doar admin (Rol 0).

### Autentificare și Autorizare

Autentificarea și autorizarea în aplicația "Car Tracking API" sunt gestionate prin JWT (JSON Web Tokens).

1. **Autentificare**
   * Utilizatorii se autentifică folosind endpoint-ul de login. După autentificare, primesc un token JWT care trebuie inclus în header-ul fiecărei cereri ulterioare pentru a accesa resursele protejate.
2. **Autorizare**
   * Fiecare endpoint are definită o politică de autorizare bazată pe roluri. Doar utilizatorii cu rolurile specificate pot accesa anumite resurse și efectua operații CRUD.

### Exemplu de Flux de Utilizare

1. **Admin**:
   * Se autentifică folosind endpoint-ul /api/users/login.
   * Adaugă o nouă mașină folosind endpoint-ul /api/cars.
   * Vizualizează toate mașinile folosind endpoint-ul /api/cars.
   * Modifică detaliile unei mașini folosind endpoint-ul /api/cars/{id}.
   * Adaugă o înregistrare de întreținere folosind endpoint-ul /api/maintenances.
   * Șterge o înregistrare de întreținere folosind endpoint-ul /api/maintenances/{id}.
   * Adaugă un nou centru de service folosind endpoint-ul /api/servicecenters.
2. **Utilizator Obișnuit**:
   * Se autentifică folosind endpoint-ul /api/users/login.
   * Vizualizează mașinile deținute folosind endpoint-ul /api/cars/owner.
   * Vizualizează înregistrările de întreținere pentru mașinile deținute folosind endpoint-ul /api/maintenances/owner.
   * Vizualizează centrele de service la care au fost întreținute mașinile folosind endpoint-ul /api/servicecenters/owner.

Concluzii si contributii

Fiecare dintre noi a contribuit in mod egal in realizarea task-urilor pentru aceasta aplicatie de tip API. Amandoi ne-am implicat in implementarea de operatii CRUD, implementarea repository-urilor, implementarea serviciilor, implementarea metodelor de mapping și implementarea de controller-e, atat si in creea entitatilor folosind EntityFramework. Prin acest proiect am aprofundat cunostiintele de ASP.NET și RESTful API prezentate la laborator.

GITHUB: <https://github.com/SocaciVictor/CarMaintenanceTrackerAPI.git>