# Sistem de Gestiune a Flotei Auto

# Documentul de proiectare

Cuprins

[1. Introducere 1](#_Toc160527836)

[1.1 Scopul documentului 1](#_Toc160527837)

[2. Prezentare generală și abordări de proiectare 2](#_Toc160527838)

[2.1 Prezentare generală 2](#_Toc160527839)

[2.2 Presupuneri/ Constrângeri/ Riscuri 2](#_Toc160527840)

[2.2.1 Presupuneri 2](#_Toc160527841)

[2.2.2 Constrângeri 2](#_Toc160527842)

[2.2.3 Riscuri 3](#_Toc160527843)

[3. Considerațiii de proiectare 4](#_Toc160527844)

[3.1 Obiective și linii directoare (ghiduri) 4](#_Toc160527845)

[3.2 Metode de dezvoltare 4](#_Toc160527846)

[3.3 Strategii de arhitectură 4](#_Toc160527847)

[4. Arhitectura Sistemului și Proiectarea Arhitecturii 6](#_Toc160527848)

[4.1 Vedere logică 6](#_Toc160527849)

[4.2 Arhitectură hardware 6](#_Toc160527850)

[4.3 Arhitectură software 6](#_Toc160527851)

[4.4 Arhitectura informațiilor 7](#_Toc160527852)

[4.5 Arhitectura de comunicații interne 7](#_Toc160527853)

[4.6 Diagrama de arhitectură a sistemului 8](#_Toc160527854)

[5. Proiectarea sistemului 9](#_Toc160527855)

[5.1 Proiectarea bazei de date 9](#_Toc160527856)

[5.1.1 Obiecte de date și structuri de date rezultante 9](#_Toc160527857)

[5.1.2 Fișiere și baze de date 9](#_Toc160527858)

[5.2 Conversii de date 9](#_Toc160527859)

[5.3 Interfețe utilizator 10](#_Toc160527860)

[5.3.1 Intrări 10](#_Toc160527861)

[5.3.2 Ieșiri 10](#_Toc160527862)

[5.4 Proiectarea interfețelor cu utilizatorul 10](#_Toc160527863)

[6. Scenarii de utilizare 11](#_Toc160527864)

[7. Proiectare de detaliu 12](#_Toc160527865)

[7.1 Proiectare hardware de detaliu 12](#_Toc160527866)

[7.2 Proiectare software de deatliu 12](#_Toc160527867)

[7.3 Proiectare detaliată de securitate 13](#_Toc160527868)

[7.4 Proiectare de detaliu pentru performanța sistemului 14](#_Toc160527869)

[7.5 Proiectare detaliată a comunicațiilor interne (între componente) 14](#_Toc160527870)

[8. Controale pentru verificarea integrității sistemului 15](#_Toc160527871)

[Anexa A: Gestiunea modificărilor documentului 16](#_Toc160527872)

[Anexa B: Acronime 17](#_Toc160527873)

[Anexa C Documente la care se face referire 18](#_Toc160527874)

## Introducere

**Numele Proiectului:** Sistem de Gestiune a Flotei Auto

**Acronim:** SGFA

**Evoluția Documentului:** Acest Document de Proiectare a Sistemului reprezintă o etapă crucială în dezvoltarea sistemului SGFA. Acesta transformă cerințele funcționale și non-funcționale stabilite în Documentul de Cerințe în specificații de proiectare mai tehnice, care vor servi drept bază pentru construirea și implementarea sistemului.

**Considerații Referitoare la Securitate și Confidențialitate:** Având în vedere natura datelor gestionate de sistemul SGFA, se acordă o atenție deosebită securității și confidențialității. Acest document va include specificații și măsuri de securitate pentru protejarea datelor sensibile privind vehiculele și operațiunile acestora.

Documentul de Proiectare a Sistemului descrie în detaliu modul în care cerințele funcționale și non-funcționale sunt traduse în specificații de proiectare a sistemului. Se va documenta proiectarea sistemului la un nivel înalt, precum și specificațiile detaliate de proiectare care vor ghida dezvoltarea fiecărei componente a sistemului.

Se vor prezenta obiectivele și considerațiile de proiectare, arhitectura sistemului la nivel înalt, proiectarea datelor, interfața om-mașină și scenariile operaționale. De asemenea, proiectarea la nivel înalt va fi descompusă în specificații detaliate pentru fiecare componentă a sistemului, inclusiv hardware, comunicații interne, software, controalele de integritate a sistemului și interfețele externe. Acest document va servi drept ghid pentru întreaga echipă de dezvoltare în construirea și implementarea sistemului SGFA.

### Scopul documentului

Scopul acestui document este să ofere o orientare detaliată și specificații tehnice pentru proiectarea și implementarea sistemului de gestionare a flotelor auto, utilizând tehnologiile Flutter pentru frontend, Django pentru backend și PostgreSQL pentru baza de date. Acesta va servi ca ghid esențial pentru echipa de dezvoltare, furnizând informații necesare pentru a defini arhitectura și designul sistemului, asigurând în același timp conformitatea cu cerințele și obiectivele proiectului.

## Prezentare generală și abordări de proiectare

### Prezentare generală

Sistemul de management al flotei auto este conceput pentru a oferi o soluție comprehensivă și eficientă pentru monitorizarea și gestionarea mașinilor din flotă. Arhitectura sistemului este concepută pentru a integra mai multe tehnologii și platforme, inclusiv Django pentru backend-ul serverului, Flutter pentru dezvoltarea aplicației mobile și Postgres pentru baza de date. Obiectivul principal al proiectului este de a oferi o interfață intuitivă și accesibilă, care să permită utilizatorilor să acceseze și să gestioneze datele despre vehiculele din flotă în timp real.

### Presupuneri/ Constrângeri/ Riscuri

#### Presupuneri

* Există o presupunere că utilizatorii finali vor avea acces la dispozitive mobile sau desktop cu conexiune la internet pentru a accesa aplicația.
* Se presupune că vehiculele din flotă sunt echipate cu dispozitive de urmărire GPS sau alte tehnologii de monitorizare.
* Se presupune că personalul va primi instruire și suport adecvat pentru utilizarea sistemului.

#### Constrângeri

* Constrângeri de performanță: Sistemul trebuie să ofere o experiență fluidă și receptivă, chiar și în condiții de trafic intens de date sau de utilizare.
* Constrângeri de securitate: Sistemul trebuie să respecte standardele de securitate și să ofere protecție adecvată pentru datele sensibile despre vehicule și utilizatori.
* Constrângeri de interoperabilitate: Sistemul trebuie să fie capabil să se integreze cu diverse alte sisteme sau dispozitive utilizate în gestionarea flotei auto.
* Constrângeri de memorie și resurse: Aplicația trebuie să fie optimizată pentru a funcționa eficient pe diferite dispozitive, inclusiv cele cu resurse limitate.

#### Riscuri

* Riscul de nefuncționare a dispozitivelor de urmărire sau a altor echipamente de monitorizare a vehiculelor, care ar putea duce la lipsa datelor sau la date inexacte.
* Riscul de expunere la amenințări de securitate, cum ar fi atacurile cibernetice sau accesul neautorizat la datele sistemului.
* Riscul de rezistență redusă la incertitudini și schimbări în mediul de afaceri, cum ar fi modificările în legislația privind protecția datelor sau în cerințele tehnologice.

Strategiile propuse pentru reducerea acestor riscuri includ implementarea unor măsuri solide de securitate cibernetică, utilizarea redundanței și a backup-urilor pentru date critice, precum și monitorizarea constantă a performanței sistemului și a integrității datelor.

## Considerațiii de proiectare

### Obiective și linii directoare (ghiduri)

* Eficiență și performanță: Accentul se pune pe dezvoltarea unei aplicații care să ofere o experiență fluidă și receptivă, fără a compromite performanța sistemului.
* Securitate: Se acordă o atenție deosebită asigurării integrității și confidențialității datelor, implementând cele mai bune practici de securitate cibernetică.
* Ușurința în utilizare: Interfața utilizatorului trebuie să fie intuitivă și ușor de navigat, pentru a asigura o experiență plăcută utilizatorilor.
* Flexibilitate și scalabilitate: Arhitectura sistemului trebuie să fie concepută pentru a permite extinderea și adaptarea la cerințele viitoare, incluzând posibilitatea de a adăuga noi funcționalități și de a gestiona o flotă auto în creștere.

### Metode de dezvoltare

* Abordarea utilizată pentru designul sistemului și al software-ului este orientată pe obiecte, folosind framework-uri precum Django pentru backend și Flutter pentru frontend. UML (Unified Modeling Language) este folosit pentru a documenta și a comunica design-ul sistemului.
* O atenție specială este acordată adaptabilității în cazul în care condițiile de proiectare sau cerințele de sistem se schimbă pe parcursul dezvoltării. Planuri alternative și soluții de rezervă sunt luate în considerare pentru a face față eventualelor contingente.

### Strategii de arhitectură

* Utilizarea tehnologiilor existente și a standardelor de industrie: S-a optat pentru utilizarea unor tehnologii și framework-uri recunoscute în industrie, cum ar fi Django pentru dezvoltarea backend-ului și Flutter pentru frontend. Această alegere oferă beneficiile suportului comunității, actualizărilor regulate și a stabilității pe termen lung. De asemenea, adoptarea standardelor de industrie asigură interoperabilitatea și compatibilitatea cu alte sisteme și tehnologii.
* Reutilizarea și modularitatea: Componentele software sunt proiectate pentru a fi modulare și reutilizabile, pentru a reduce redundanța și a facilita dezvoltarea, testarea și întreținerea sistemului. Prin separarea funcționalităților în module distincte, fiecare cu responsabilități clare, se promovează o arhitectură curată și extensibilă.
* Flexibilitate pentru extensibilitate: Arhitectura sistemului este proiectată pentru a permite adăugarea facilă de noi funcționalități și module în viitor, în conformitate cu evoluția cerințelor și necesităților pieței. Aceasta se realizează prin adoptarea unor principii de proiectare orientate către obiecte, cum ar fi principiile SOLID, care promovează un design flexibil și extensibil.
* Paradigmele interfeței utilizatorului: Interfața utilizatorului este proiectată pentru a oferi o experiență plăcută și intuitivă utilizatorilor finali. Se aplică principii de design modern, cum ar fi Material Design pentru aplicații mobile și Responsive Design pentru adaptabilitatea la diferite dispozitive și dimensiuni de ecran. Interacțiunea cu utilizatorul este simplificată și intuitivă, pentru a facilita utilizarea aplicației și pentru a crește satisfacția utilizatorului.
* Gestionarea erorilor și a recuperării: Sistemul include mecanisme robuste pentru gestionarea erorilor și recuperarea în cazul unor situații neașteptate sau a unor condiții de eroare. Sunt implementate strategii adecvate de gestionare a erorilor, precum afișarea de mesaje de eroare clare pentru utilizatori și jurnale de evenimente pentru diagnosticarea și remedierea problemelor.
* Performanță și scalabilitate: Arhitectura sistemului este optimizată pentru a asigura performanță și scalabilitate, inclusiv gestionarea eficientă a resurselor, optimizarea interogărilor la bază de date și implementarea de tehnici de caching și de optimizare a încărcării paginilor. Aceste strategii sunt concepute pentru a asigura o experiență rapidă și receptivă utilizatorilor, chiar și în condiții de utilizare intensă sau de creștere a volumului de date.

## Arhitectura Sistemului și Proiectarea Arhitecturii

Această secțiune conturează designul arhitecturii sistemului și hardware-ului.

### Vedere logică

* **Backend-ul (Serverul):** Aceasta este componenta centrală a sistemului, responsabilă pentru gestionarea tuturor operațiunilor logice și pentru furnizarea datelor către interfața utilizatorului. Backend-ul este construit folosind framework-ul Django, care oferă o structură robustă și scalabilă pentru dezvoltarea aplicațiilor web. Aici sunt gestionate operațiile CRUD (create, read, update, delete) pentru vehicule, utilizatori, rapoarte etc.
* **Frontend-ul (Interfața utilizatorului):** Interfața utilizatorului este reprezentată de aplicația mobilă și de interfața web, care permit utilizatorilor să acceseze și să interacționeze cu datele sistemului. Aplicația mobilă este dezvoltată folosind framework-ul Flutter, în timp ce interfața web este construită utilizând tehnologii web moderne precum HTML, CSS și JavaScript. Interfața utilizatorului comunică cu backend-ul prin intermediul unor API-uri RESTful pentru a obține și a trimite date către server.
* **Baza de date (Postgres):** Baza de date este responsabilă pentru stocarea și gestionarea datelor sistemului. Postgres este utilizat ca sistem de gestionare a bazelor de date, oferind o performanță și o fiabilitate ridicate. Datele despre vehicule, utilizatori, rapoarte, permisiuni etc., sunt stocate în tabelele bazei de date și sunt accesate și manipulate de către backend-ul aplicației.
* **Servicii externe și dispozitive de urmărire:** Sistemul poate interacționa cu diverse servicii externe și dispozitive de urmărire GPS sau alte tehnologii de monitorizare a vehiculelor. Aceste date sunt integrate în sistem pentru a furniza informații actualizate despre locația, starea și alte aspecte relevante ale vehiculelor din flotă.

### Arhitectură hardware

Nu este cazul

### Arhitectură software

Componentele software ale sistemului de management al flotei auto includ:

* **Backend-ul (Serverul):** Dezvoltat folosind framework-ul Django, care oferă o structură solidă pentru dezvoltarea aplicațiilor web. Este responsabil pentru gestionarea logicii de afaceri, manipularea datelor și furnizarea de servicii prin intermediul API-urilor RESTful către frontend-ul aplicației.
* **Frontend-ul (Interfața utilizatorului):**
  + **Aplicația mobilă:** Dezvoltată folosind framework-ul Flutter, oferă o interfață utilizator modernă și receptivă pentru utilizatorii care accesează sistemul de pe dispozitive mobile.
  + **Interfața web:** Construită cu tehnologii web moderne precum HTML, CSS și JavaScript, permite utilizatorilor să acceseze și să interacționeze cu datele sistemului de pe dispozitive desktop și mobile.
* **Baza de date (Postgres):** Utilizată pentru stocarea persistentă a datelor sistemului, inclusiv informații despre vehicule, utilizatori, rapoarte etc. Postgres oferă performanță și fiabilitate, fiind alegerea ideală pentru aplicații critice.
* **Servicii externe și dispozitive de urmărire:** Interacționează cu diverse servicii externe și dispozitive de urmărire GPS pentru a colecta și a integra date despre locația și starea vehiculelor din flotă în sistem.

Arhitectura software este concepută pentru a asigura o separare clară a responsabilităților și pentru a facilita dezvoltarea, testarea și întreținerea sistemului. Utilizarea unor tehnologii moderne și standardizate, precum și adoptarea unor principii de proiectare și dezvoltare bine-cunoscute, asigură o aplicație robustă, scalabilă și ușor de întreținut.

### Arhitectura informațiilor

Sistemul va stoca o varietate de informații, inclusiv:

* Informații despre vehicule, cum ar fi marca, modelul, numărul de înmatriculare, anul fabricației, kilometrajul, starea tehnică etc.
* Informații despre utilizatori, cum ar fi nume, prenume, adresa de email, număr de telefon, roluri și permisiuni etc.
* Rapoarte despre mentenanță, revizii, reparații, programări de service etc.
* Informații despre trasee și locații ale vehiculelor în timp real, obținute prin intermediul dispozitivelor de urmărire GPS sau alte tehnologii de monitorizare.

Anumite informații, cum ar fi datele personale ale utilizatorilor, pot fi considerate informații sensibile și vor fi protejate conform standardelor și reglementărilor de securitate și confidențialitate aplicabile.

Datele sistemului pot fi furnizate de utilizatori prin intermediul interfeței utilizatorului sau pot fi obținute automat prin intermediul integrării cu servicii externe sau dispozitive de urmărire.

### Arhitectura de comunicații interne

Arhitectura de comunicații interne a sistemului se bazează pe comunicarea între componente software și baza de date. Comunicarea este realizată prin intermediul protocolului HTTP pentru cererile și răspunsurile către backend și prin intermediul protocoalelor specifice pentru interacțiunea cu baza de date (de exemplu, PostgreSQL protocol).

Componentele sistemului sunt conectate prin intermediul unei rețele locale (LAN) pentru comunicarea internă a sistemului, în timp ce pentru comunicarea externă cu serviciile externe se folosesc protocoale de comunicație standard, cum ar fi HTTP, HTTPS etc.

Diagrama de arhitectură a sistemului va ilustra în detaliu modul în care componentele software sunt conectate între ele și modul în care comunică între ele pentru a furniza funcționalitatea sistemului.

### Diagrama de arhitectură a sistemului

Diagrama de arhitectură a sistemului va oferi o imagine generală și integrată a structurii și interacțiunilor dintre componentele hardware, software, de comunicații și de informații ale sistemului. Aceasta va evidenția modul în care fiecare componentă contribuie la funcționalitatea generală a sistemului și cum acestea sunt conectate și comunică între ele. Diagrama va furniza o înțelegere vizuală a arhitecturii sistemului și a relațiilor dintre componentele sale.

## Proiectarea sistemului

### Proiectarea bazei de date

#### Obiecte de date și structuri de date rezultante

1. **Vehicule:**
   * Structura de date:
     + ID vehicul (int, cheie primară)
     + Marca (string)
     + Modelul (string)
     + Numărul de înmatriculare (string)
     + Anul fabricației (int)
     + Kilometraj (float)
     + Starea tehnică (string)
     + Data ultimei revizii (date)
   * Această structură de date stochează informațiile despre fiecare vehicul din flotă.
2. **Utilizatori:**
   * Structura de date:
     + ID utilizator (int, cheie primară)
     + Nume (string)
     + Prenume (string)
     + Email (string)
     + Parolă (string, criptată)
     + Roluri (string)
   * Această structură de date stochează informațiile despre utilizatorii sistemului și rolurile lor.
3. **Rapoarte:**
   * Structura de date:
     + ID raport (int, cheie primară)
     + ID vehicul (int, cheie externă către tabelul Vehicule)
     + ID utilizator (int, cheie externă către tabelul Utilizatori)
     + Descriere (text)
     + Data creării (date)
   * Această structură de date stochează rapoartele generate de utilizatori pentru vehiculele din flotă.
4. **Permisiuni:**
   * Structura de date:
     + ID permisiune (int, cheie primară)
     + ID utilizator (int, cheie externă către tabelul Utilizatori)
     + Permisiuni (string)
   * Această structură de date stochează permisiunile acordate utilizatorilor în cadrul sistemului.

#### Fișiere și baze de date

##### Baze de date

Proiectarea detaliată a fișierelor DBMS va include tabelele descrise mai sus, împreună cu cheile primare, cheile externe și orice constrângeri de integritate referențială necesare.

##### Fișiere non-DBMS

În cadrul sistemului de management al flotei auto, este posibil să fie utilizate și fișiere non-DBMS pentru scopuri precum jurnalele de evenimente, fișiere de configurare etc. Aceste fișiere sunt utilizate în principal pentru citirea și scrierea de către modulele de sistem, precum și pentru configurarea și administrarea sistemului. Alte fisiere vor fi imaginile cu logoul companiei.

### Conversii de date

Documentele care descriu conversiile de date necesare pot include:

* Specificații ale formatului de import/export pentru datele vehiculelor din alte sisteme sau formate de fișiere.
* Proceduri pentru transformarea și validarea datelor înainte de importul în baza de date a sistemului.
* Metode pentru exportul datelor din baza de date a sistemului în formate utilizabile de alte sisteme sau raportare.

Aceste documente pot fi stocate într-un repository de documentație asociat proiectului sau pot fi parte din documentația tehnică a sistemului.

### Interfețe utilizator

#### Intrări

Clasele de utilizatori sau rolurile asociate cu sistemul de management al flotei auto includ:

1. Utilizatori normali: Aceștia sunt angajații sau operatorii care interacționează zilnic cu sistemul pentru a introduce sau a accesa informații despre vehicule, rapoarte și programări de service.
2. Administratori: Aceștia sunt responsabili de configurarea sistemului, gestionarea utilizatorilor și a permisiunilor, precum și de monitorizarea activității sistemului.

Mijloacele de intrare folosite de utilizator/operator includ:

* Ecranele de introducere a datelor pentru adăugarea și actualizarea informațiilor despre vehicule, rapoarte și utilizatori.
* Formulare pentru programarea și gestionarea reviziilor și reparațiilor vehiculelor.
* Interfețe pentru configurarea permisiunilor și a altor setări ale sistemului.

Aspectul și detaliile ecranelor de intrare vor fi specificate în documentația de proiectare a interfeței utilizator.

#### Ieșiri

Ieșirile sistemului includ rapoarte, ecrane de afișare a datelor și interfețe grafice pentru utilizatori. Acestea pot include:

* Rapoarte despre starea vehiculelor, programări de service, revizii și reparații.
* Ecrane de vizualizare a detaliilor vehiculelor, rapoartelor și utilizatorilor.
* Interfețe pentru exportul datelor în formate utilizabile de alte sisteme sau pentru raportare externă.

Aceste ieșiri sunt destinate utilizatorilor normali și administratorilor și vor fi proiectate pentru a fi intuitive și ușor de înțeles. Restricțiile de acces și considerațiile de securitate vor fi luate în considerare în proiectarea interfețelor utilizator.

### Proiectarea interfețelor cu utilizatorul

Proiectarea interfețelor cu utilizatorul pentru sistemul de management al flotei auto include mai multe ecrane și elemente de interfață pentru a facilita interacțiunea utilizatorilor cu sistemul. Mai jos sunt enumerate și descrise câteva dintre acestea:

1. **Tabloul de bord (Dashboard)**:
   * Descriere: Acesta este primul ecran pe care utilizatorii îl văd după autentificare. Tabloul de bord oferă o vedere generală a stării actuale a flotei auto, inclusiv numărul total de vehicule, kilometrajul total al flotei, rapoartele recente și programările de service.
   * Elemente: Grafice și diagrame pentru vizualizarea datelor, statistici rapide, linkuri către paginile relevante.
2. **Ecranul de gestionare a vehiculelor**:
   * Descriere: Acest ecran permite utilizatorilor să vadă și să gestioneze informațiile despre vehiculele din flotă. Utilizatorii pot adăuga, actualiza sau șterge vehicule, pot vizualiza detalii precum kilometrajul, starea tehnică și data ultimei revizii.
   * Elemente: Formulare de introducere a datelor, liste de vehicule, filtre pentru căutare, butoane de acțiune pentru editare și ștergere.
3. **Ecranul de gestionare a rapoartelor**:
   * Descriere: Acest ecran permite utilizatorilor să genereze, vizualizeze și gestioneze rapoartele despre diverse aspecte ale flotei auto, cum ar fi rapoarte de stocare, rapoarte de utilizare și rapoarte de revizii.

## Scenarii de utilizare

Pentru sistemul de management al flotei auto, următoarele sunt câteva scenarii de utilizare care descriu funcționalitatea generală a sistemului și fluxul operațional al acestuia:

1. Adăugarea unui nou vehicul în sistem:

* Utilizatorul navighează către ecranul de gestionare a vehiculelor.
* Utilizatorul apasă butonul "Adăugare vehicul".
* Complează formularul cu informațiile despre noul vehicul: număr de înmatriculare, marcă, model, an de fabricație etc.
* Utilizatorul salvează informațiile introduse, adăugând astfel vehiculul în baza de date a sistemului.

2. Generarea unui raport de utilizare a vehiculelor pentru o anumită perioadă:

* Utilizatorul accesează ecranul de gestionare a rapoartelor și selectează opțiunea pentru raportul de utilizare a vehiculelor.
* Utilizatorul specifică intervalul de timp pentru raport și alte criterii de filtrare, cum ar fi tipul de vehicul sau departamentul.
* Sistemul generează raportul bazat pe criteriile specificate de utilizator.
* Utilizatorul vizualizează raportul generat și, dacă este necesar, îl exportă în formatul dorit.

3. Programarea unei revizii tehnice pentru un vehicul specific:

* Utilizatorul accesează ecranul de gestionare a programărilor de service.
* Utilizatorul selectează vehiculul pentru care dorește să programeze o revizie tehnică.
* Utilizatorul completează detaliile programării, cum ar fi data și ora dorite pentru revizie.
* Sistemul validează și înregistrează programarea.
* Utilizatorul primește confirmarea programării și detaliile acesteia.

Aceste scenarii de utilizare acoperă diverse aspecte ale funcționalității sistemului și ilustrează modul în care utilizatorii interacționează cu acesta pentru a îndeplini sarcini specifice legate de gestionarea flotei auto.

## Proiectare de detaliu

### Proiectare hardware de detaliu

Nu este cazul

### Proiectare software de deatliu

1. **Identificator serviciu:** Serviciul de management al flotei auto.
2. **Clasificare:** Aplicație software.
3. **Definiție:** Acest serviciu gestionează toate aspectele legate de flota de vehicule, inclusiv adăugarea, actualizarea și ștergerea vehiculelor, programarea întreținerii și reviziilor, generarea de rapoarte și gestionarea programărilor de service.
4. **Cerințe:** Satisfacerea cerințelor de performanță, scalabilitate și securitate. Interfațarea cu baza de date pentru stocarea și accesarea informațiilor despre vehicule și programări. Asigurarea unei interfețe de utilizator intuitive și ușor de folosit.
5. **Structuri de date interne:** Utilizarea de structuri de date pentru stocarea informațiilor despre vehicule, programări, utilizatori și alte date relevante.
6. **Constraingeri:** Limite de timp pentru execuția anumitor operațiuni, restricții de acces pentru anumite funcționalități, precum și limite de stocare pentru datele sistemului.
7. **Compoziție:** Utilizarea subserviciilor pentru autentificare, autorizare, gestionarea sesiunilor și alte funcționalități de bază.
8. **Utilizatori/Interacțiuni:** Interacționează cu alte servicii pentru autentificare și autorizare. Comunică cu interfața utilizatorului pentru a primi și a afișa informații despre vehicule și programări.
9. **Procesare:** Utilizarea algoritmilor de gestionare a programărilor și a reviziilor, validarea datelor introduse de utilizator și gestionarea excepțiilor.
10. **Interfețe/Exporturi:** Furnizarea de servicii pentru interfața utilizatorului, inclusiv funcționalități de adăugare, actualizare și ștergere a vehiculelor și programărilor. Exportarea datelor pentru generarea de rapoarte.

### Proiectare detaliată de securitate

* Autentificare: Utilizarea unui sistem de autentificare sigur, cum ar fi autentificarea bazată pe parole sau autentificarea cu factori multipli.
* Autorizare: Definirea permisiunilor și rolurilor utilizatorilor pentru a accesa diferitele funcționalități ale sistemului.
* Jurnalizare și auditare: Înregistrarea și monitorizarea activităților utilizatorilor pentru a identifica și a investiga activități suspecte sau neautorizate.
* Criptare: Criptarea datelor sensibile pentru a asigura confidențialitatea și integritatea acestora în timpul transmiterii și stocării.
* Utilizarea porturilor de rețea: Controlul accesului la porturile de rețea și gestionarea traficului pentru a preveni atacurile cibernetice.
* Detectare și prevenire a intruziunilor: Implementarea unor mecanisme de detectare a intruziunilor pentru a identifica și a preveni accesul neautorizat la sistem.

### Proiectare de detaliu pentru performanța sistemului

Cerințe/estimări de capacitate și volum: Determinarea numărului de utilizatori simultani, volumului de date procesate și cerințelor de performanță ale sistemului în funcție de încărcările anticipate.

Așteptări de performanță: Stabilirea timpilor de răspuns acceptabili pentru diferite operațiuni ale sistemului, cum ar fi interogările bazei de date sau generarea de rapoarte.

Cerințe de disponibilitate: Definirea nivelului minim de disponibilitate a sistemului și a funcționalităților critice.

Proiectare de performanță: Selectarea și configurarea hardware-ului și software-ului pentru a satisface cerințele de capacitate și performanță.

Proiectare de fiabilitate: Implementarea mecanismelor de backup, recuperare și arhivare pentru a asigura disponibilitatea și integritatea datelor.

Proiectare de backup, recuperare și arhivare: Definirea strategiilor de backup și recuperare a datelor, inclusiv frecvența backup-urilor, procedurile de restaurare și politica de arhivare a datelor.

### Proiectare detaliată a comunicațiilor interne (între componente)

Numărul de servere și clienți: Determinarea numărului de servere și clienți implicați în comunicațiile interne și dimensionarea rețelelor corespunzător.

Specificații pentru cerințele de sincronizare și control al busului: Definirea cerințelor de sincronizare și control al busului pentru a asigura o comunicare eficientă între componente.

Format(e) pentru datele schimbate între componente: Specificarea formatului datelor schimbate între componente pentru a asigura compatibilitatea și interoperabilitatea.

Reprezentare grafică a conectivității: Crearea unei diagrame care să arate conexiunile între componente și direcția fluxului de date, precum și distanțele aproximative între componente pentru a facilita achiziționarea hardware-ului și instalarea în locația dorită.

Topologia LAN: Definirea topologiei rețelei locale (LAN) pentru a stabili modul în care componente sunt conectate între ele și modul în care fluxul de date este gestionat în rețea.

## Controale pentru verificarea integrității sistemului

Pentru a asigura integritatea sistemului, este esențial să avem în vedere următoarele controale și proceduri:

1. Sistem de securitate internă:

- Definirea și implementarea unui sistem de autentificare și autorizare robust, care să restricționeze accesul utilizatorilor la datele critice în funcție de rolurile și privilegiile lor.

- Utilizarea principiului celui mai mic privilegiu pentru a minimiza expunerea la riscuri.

2. Implementarea unor mecanisme de auditare a activității aplicațiilor pentru a monitoriza și înregistra toate operațiunile efectuate asupra datelor critice, cum ar fi adăugarea, ștergerea sau actualizarea acestora.

3. Tabele standard pentru validarea datelor: definirea și utilizarea unor tabele standard pentru validarea datelor introduse în sistem, astfel încât să se asigure consistența și acuratețea informațiilor.

4. Izolarea unităților de cod: În unit testing, unitatea de cod este testată izolat, fără a depinde de alte componente ale sistemului. Acest lucru asigură că eventualele erori sunt localizate și ușor de identificat.

5. Automatizarea testelor: Testele unitare sunt de obicei automate, ceea ce înseamnă că pot fi rulate automat de către un sistem de testare fără intervenția umană. Acest lucru permite executarea lor rapidă și repetată, ceea ce este esențial într-un proces de dezvoltare agilă sau continuă.

6. Scopul testelor: Scopul testelor unitare este de a verifica corectitudinea unei unități de cod în diverse scenarii și condiții. Acestea verifică dacă unitatea produce rezultatele corecte pentru anumite intrări și dacă se comportă așa cum este așteptat în situații normale și extreme.

7. Detectarea erorilor timpurii: Unit testing permite detectarea erorilor într-o fază incipientă a dezvoltării, ceea ce face ca remedierea acestora să fie mai ușoară și mai puțin costisitoare decât în ​​fazele ulterioare ale procesului de dezvoltare.

Anexa A: Gestiunea modificărilor documentului

Tabel 1 – Înregistrarea modificărilor asupreaa documentului curent

| versiune | Data | Autorul/Deținătorul | Descriere |
| --- | --- | --- | --- |
| 1.0 | 10.04.2024 | Rahme Victor | Prima revizie |
| 1.1 | 11.04.2024 | Rahme Victor | Adaugarea ultimelor 2 capitole |
| 1.2 | 14.04.2024 | Rahme Victor | Mici restructurari |

Anexa B: Acronime

Tabel 2 - Acronime

| Acronim | Forma completă |
| --- | --- |
| SGFA | Sistem de gestiune a unei flote auto |

Anexa C Documente la care se face referire

Tabel 3 – Documente la care se facce referire

| Nume document | Locație sau URL | Dată emitere document |
| --- | --- | --- |
| < Nume document > | <Locație sau URL> | <ZZ/LL/AAAA> |
| < Nume document > | <Locație sau URL> | <ZZ/LL/AAAA> |
| < Nume document > | <Locație sau URL> | <ZZ/LL/AAAA> |