Cuprins

Listă de figuri

Listă de tabele

Capitolul 1
DECIZII DE DESIGN

În acest capitol sunt prezentate cele mai importante decizii de design luate în cadrul dezvoltării componentei software.

1.1 Deciderea asupra modului de stocare a datelor

1.1.1 Subiectul problemei

Nevoia modulului de a stoca şi de a accesa date stocate anterior.

1.1.2 Factori de decizie

- Consistenţa datelor stocate
- Utilizarea RAM-ului
- Timp de accesare la pornirea aplicaţiei
- Timp de accesare în cadrul aplicaţiei
- Spaţiul ocupat pe disc
- Compatibilitate cu versiunile anterioare

1.1.3 Soluţii propuse

În următorul tabel, se presupune ca pentru fişierele binare, XML şi JSON este necesară încărcarea datelor la pornirea aplicaţiei. SQLite oferă însă soluţii de căutare inteligente, nefiind necesară încărcarea tuturor datelor la pornirea aplicaţiei.

1.1.4 Decizia

S-a decis folosirea SQLite ca format pentru baza de date deoarece îndeplinea toate criteriile specificate.

TABELA 1.1: Compararea principalelor metode de stocare a datelor pe baza factorilor de influenţare

	Binar	**XML sau JSON**	**SQLite**	**Memorare în Cloud**
Consistenţa datelor stocate	Nu	Nu	Da	Da
Utilizarea RAM-ului	Ridicat	Scăzut	Mediu	Mediu
Timp de accesare la pornirea aplicaţiei	Mediu	Ridicat	Scăzut	Scăzut
Timp de accesare în cadrul aplicaţiei	Scăzut	Scăzut	Mediu	Ridicat
Spaţiul ocupat pe disc	Scăzut	Ridicat	Scăzut	Foarte scăzut
Compatibilitate cu versiunile anterioare	Nu	Da	Da	Nu

1.2 Deciderea asupra comportamentului sincron/asincron

1.2.1 Subiectul problemei

Există operaţii care ar putea necesita mai mult timp (>100ms) şi nu este vizibil direct faptul că acestea au fost declanşate de către o cerere (exemplu: o nouă poziţie este trimisă). O cerere poate declanşată din fire de execuţie diferite. Există posibilitatea ca acest lucru sa fie realizat sincron, în afara modului, între cereri şi răspunsuri, fapt ce poate duce la deadlock.

1.2.2 Factori de decizie

- Timpul în care firul de execuţie este blocat de cerere
- Sincronizarea între operaţii

1.2.3 Soluţii propuse

- Procesul se va executa asincron folosind un fir de execuţie de lucru
- Procesul se va executa sincron, în interiorul cererilor

1.2.4 Decizia

Se vor furniza două interfeţe diferite. Funcţionalitatea va fi oferită printr-o interfaţă sincronă, ce va fi utilizată în cadrul operaţiilor ce au loc pe un singur fir de execuţie. O alta interfaţă va decupla firele de execuţie şi procesele într-o buclă de lucru. Acest fapt ne oferă libertatea utilizării principiului de multithread-ing (execuţia mai multor thread-uri în acelaşi pipeline, fiecare având propria secţiune de timp în care este menit să lucreze).

Capitolul 2
ARHITECTURA SOFTWARE

În acest capitol este prezentată partea structurală a modulului cât şi unităţile software din care acesta este format.

2.1 Unitatea software PNavServiceImpl

Unitatea PNavServiceImpl implementează interfaţa sincronă şi asincronă, şi îi oferă totodată dezvoltatorului posibilitatea de a alege ce interfaţă doreşte să folosească.

Cea asincronă are avantajul de a decupla firele de execuţie şi de a permite rularea activităţilor în paralel, însă are şi dezavantajul necesităţii de implementare unui mecanism de sincronizare în codul aplicaţiei în care va fi folosit modulul.

2.2 Unitatea software PNavImpl

Deşi funcţionalităţile de bază precum precum învăţarea şi predicţia sunt realizare de către unitatea PNavRecorder respectiv PNavPredictor, unitatea PNavImpl realizează funcţionalităţi suplimentare cum ar fi multiple profile de utilizatori, ştergerea bazelor de date.

Funcţionalitatea multiplelor profile de utilizatori permite gestionarea mai multor baze de date, ce pot fi selectate pe baza unui ID de profil. Acest ID poate cuprinde valori în intervalul 0-255. Pentru fiecare profil este creat un nou fişier în care vor fi stocate datele de utilizator. Unitatea PNavImpl implementează de asemenea şi funcţionalităţi de întreţinere a profilelor de utilizator precum ştergerea individuală, ştergerea totală, copierea, schimbarea între profile.

2.3 Unitatea software PNavRecorder

NavRecorder-ul este unitate în care întreg procesul de învăţare are loc.

Unitatea primeşte datele de geolocație şi de timp (oră - zi/lună/an) şi învaţă rutele parcurse de către dezvoltator într-un mod inteligent. Acest lucru înseamnă că waypoint-urile (punctele prin care a

trecut utilizatorul în timpul rutei sale) sunt stocate numai când autovehiculul şi-a schimbat orientarea semnificativ (valoare standard: > 15°) sau distanţa dintre waypoint-uri nu este prea scurtă (valoare standard: > 200m). Valori pot fi configurate înaintea procesului de compilare.

Când sesiunea de înregistrare este finalizată, waypoint-urile sunt trimise către unitatea software PNavDataManager pentru a fi scrise în baza de date. Totodată, unitatea are implementate funcţionalităţi de oprire-pornire, lucru ce-i acordă dezvoltatorului dreptul de opri şi porni oricând sesiunea de înregistrare.

2.4 Unitatea software PNavPredictor

Rolul unităţii PNavPredictor este acela de a calcula predicţiile.

Primul pas constă în încărcarea datelor prin unitatea PNavDataManager, care sunt mai departe prioritizate în funcţie de criterii specifice (descrise în tabela 5.2, “Criterii de prioritizare pentru predicţia bazată pe rute”). Datele pot fi de asemenea filtrate pe baza aceloraşi criterii, rezultând astfel o cantitate mai mică de date şi un timp mai scurt de încărcare a acestora.

Prioritizarea datelor este bazată atât pe datele de geolocație cât şi cele de timp, astfel încât o rută va avea o probabilitate mult mai mare de utilizare într-o anumită zi din săptămână sau la o anumită oră din zi.

Ca şi unitatea PNavRecorder, unitatea PNavPredictor are implementate funcţionalităţi de oprire-pornire.

2.5 Unitatea software PNavConfiguration

Unitatea PNavConfiguration configurează unitatea NavPredictor, prin intermediul unor funcţii ce folosesc criteriile definite în unitatea PNavCriteria.

2.6 Unitatea software PNavCriteria

Unitatea PNavCriteria conţine toate tipurile de criterii ce pot fi folosite la filtrarea sau prioritizarea datelor.

Fiecare criteriu în parte este folosit la procesarea datelor de către unitatea PNavPredictor. După procesarea tuturor criteriilor cea mai probabilă rută este creată.

2.7 Unitatea software PNavDataManager

Unitatea PNavDataManager implementează logica necesară pentru a realiza comunicarea între unitatea PNavDataStorage şi restul unităţilor.

În general, obiectele sunt stocate separat (e.g. rutele sunt stocate separat faţă de destinaţiilor lor). Cum însă pentru predicţia unei rute este nevoie de toate informaţiile, unitatea PNavDataManager le comasează. Aceasta oferă de asemenea şi alte funcţionalităţi precum adăugarea, gruparea, căutarea sau ştergerea de obiecte.

2.8 Unitatea software PNavDataStorage

Unitatea PNavDataStorage este dezvoltată pe baza structurii bazei de date.

În afară de funcţionalitatea principală de a stoca sau încărca date, aceasta asigură şi accesarea selectivă a obiectelor. Pentru realizarea acestor funcţionalităţi se execută interogări prin intermediul SQLite.

Capitolul 3
STRUCTURA BAZEI DE DATE

4.1 Tabelele de date

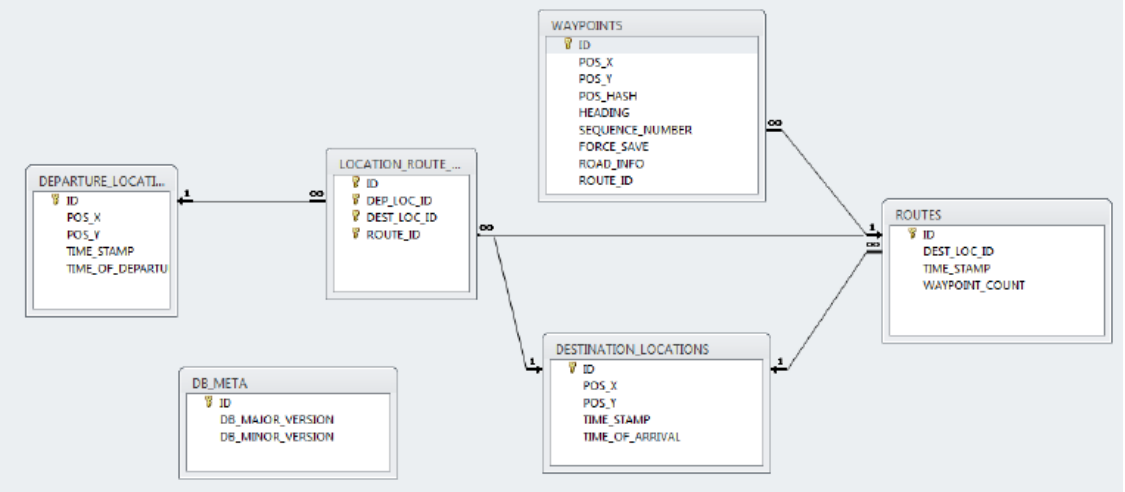

FIGURA 4.1: Structura tabelelor de date şi relaţiile dintre ele

Tabelele din figura de mai sus sunt folosite pentru realiza structura întregii baze de date.

Tabela meta este folosită la identificarea versiunii bazei de date. Acest lucru este necesar pentru a detecta compatibilitatea şi pentru a permite migrarea către o versiune mai recentă.

Datele înregistrate sunt separate în puncte de plecare, destinaţii, rute şi waypoint-uri. O rută este întotdeauna formată din mai multe waypoint-uri, unul sau mai multe puncte de plecare şi una sau mai multe destinaţii. Ruta (waypoint-urile) sunt stocate numai o singură dată, în timp ce toate punctele de plecare şi destinaţiile sunt stocate. În acest fel, numărul de destinaţii poate influenţa probabilitatea rutei.

Accesul la date se face prin SQLite. Toate datele stocate pot fi atât citite cât şi modificate.

Capitolul 4
STRUCTURA BAZEI DE DATE

4.1 Tabelele de date

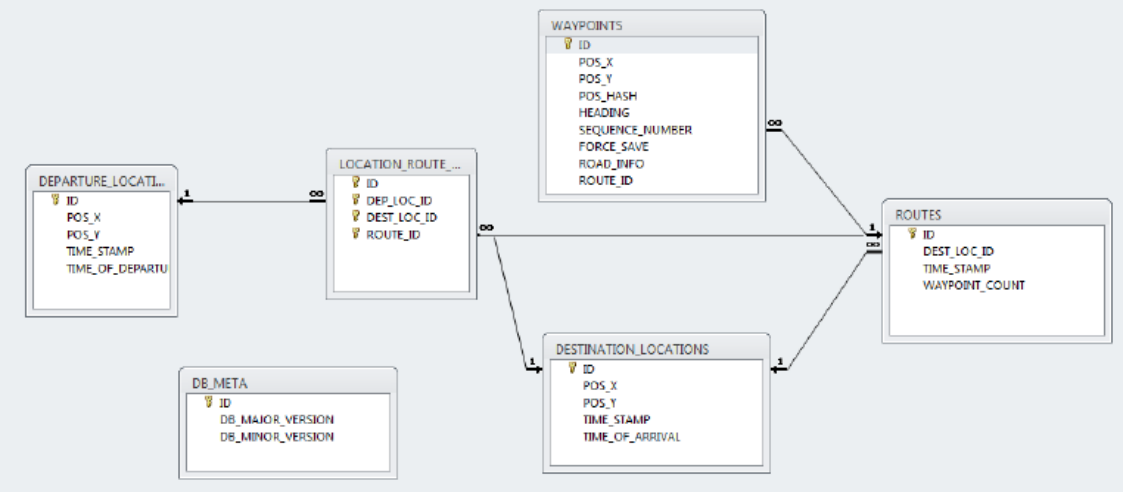

FIGURA 4.2: Structura tabelelor de date şi relaţiile dintre ele

Tabelele din figura de mai sus sunt folosite pentru realiza structura întregii baze de date.

Tabela meta este folosită la identificarea versiunii bazei de date. Acest lucru este necesar pentru a detecta compatibilitatea şi pentru a permite migrarea către o versiune mai recentă.

Datele înregistrate sunt separate în puncte de plecare, destinaţii, rute şi waypoint-uri. O rută este întotdeauna formată din mai multe waypoint-uri, unul sau mai multe puncte de plecare şi una sau mai multe destinaţii. Ruta (waypoint-urile) sunt stocate numai o singură dată, în timp ce toate punctele de plecare şi destinaţiile sunt stocate. În acest fel, numărul de destinaţii poate influenţa probabilitatea rutei.

Accesul la date se face prin SQLite. Toate datele stocate pot fi atât citite cât şi modificate.

Capitolul 5
DESCRIEREA ALGORITMILOR

5.1 Învăţarea rutelor

Rutele sunt învăţate printr-un mecanism inteligent. Acest lucru are avantajul de a reduce semnificativ baza de date şi de a accelera încărcarea datelor.

5.1.1 Reducerea waypoint-urilor

Criteriile pentru excluderea waypoint-urilor sunt:

1. Vehiculul nu şi-a schimbat orientarea semnificativ (valoare standard: $> 15°$)
2. Poziţiile sunt apropiate una de cealaltă (valoare standard: > 200m)

Valorile standard pot fi configurate înaintea procesului de compilare.

5.1.2 Verificarea rutelor duble

În cazul în care ruta curentă are o secţiune similară cu o rută deja învăţată, unitatea software PNavDataManager va detecta secţiune respectivă şi va stoca numai waypoint-urile situate după aceasta. În schimb, toate destinaţiile sunt memorate. Criteriile pentru a detecta o astfel de rută sunt:

1. Destinaţia noii rute trebuie să fie situată într-o rază de 1km faţă de locaţia vechii destinaţii
2. Punctul de start al noii rute trebuie să fie situată într-o rază de 1km faţă de punctul de plecare vechii destinaţii
3. Toate waypoint-urile noii rute trebuie:
   (a) Să fie situat într-o anumită rază faţă de punctul de start al vechii rute
   (b) Sau să fie situat într-o anumită rază faţă de destinaţia vechii rute
   (c) Sau să fie situat într-o anumită rază faţă de un waypoint al vechii rute

Pragurile pot fi configurate înaintea procesului de compilare.

5.1.3 Filtrarea rutelor inutile

În timpul învăţării, rutele prea scurte (< 2km) nu vor fi stocate deoarece acest lucru ar însemna faptul că utilizatorul este destul de aproape de destinaţie.

Totodată, rutele foarte lungi (> 200km) vor fi de asemenea exclude din stocare deoarece ele nu reprezintă rute uzuale.

Pragurile pot fi configurate înaintea procesului de compilare.

5.2 Eliberarea de spaţiu

Dacă pragul setat iniţial pentru dimensiunea maximă a bazei de date este atins, se va activa funcţia de eliberare a spaţiului. Aceasta va şterge obiectele cele mai vechi pentru a crea loc pentru obiectele noi.

5.3 Furnizarea predicţiilor

Pentru ca predicţiile să fie disponibile sunt necesari doi paşi.

Primul pas reprezintă încărcarea datelor relevante, în timp de al doilea constă în prioritizarea lor.

5.3.1 Filtrarea datelor

Pentru o încărcare selectivă şi mai rapidă a datelor din unitatea PNavDataStorage, sunt create interogări. Sunt definiţi trei paşi în filtrare, unde pasul următor se excută doar în cazul în care cel curent nu a returnat destule rezultate:

1. Filtrarea rutei în funcţie de distanţa de la poziţia curentă la waypoint-urile rutelor
2. Filtrarea rutei în funcţie de distanţa de la poziţia curentă la punctele de start ale rutelor
3. Filtrarea rutelor în funcţie de timpul scurs până la ajungerea la destinaţie

Toate rutele ce duc la o destinaţie situată la o distanţă mai mică de 1km faţă de poziţia actuală sunt ignorate deoarece utilizatorul aproape a ajuns la eventuala destinaţie.

În cazul predicţiilor baza pe timp, modulul furnizează o predicţie fără a cunoaşte ruta, ci doar destinaţia sa. Un astfel de caz ar fi cel în care utilizatorul a condus pe o rută în intervalul luni-miercuri, însă in ziua de joi a pornit de la o altă locaţie. Astfel, bazat pe timp, modulul prezice destinaţia fără a şti ruta corespunzătoare acesteia.

5.3.2 Frecvenţa predicţiei de rute

De obicei, waypoint-urile furnizate de modul de predicţie sunt transformate într-o rută ce foloseşte drumuri din hartă, fapt ce durează câteva secunde. Pentru a nu supraîncărca sistemul de navigaţie cu prea multe predicţii, dezvoltatorul poate seta timpul minim dintre două predicţii. Acestă setare se poate efectua chiar în timpul rulării.

5.3.3 Numărul maxim de predicţii

Pentru a putea suporta multiple platforme, modulul permite setarea numărului maxim de rute prezise ce apar deodată. Acestă setare se poate efectua chiar în timpul rulării.

5.3.4 Prioritizarea predicţiilor

Există mai multe criterii pentru a stabili prioritatea unei rute.

TABELA 5.2: Criterii de prioritizare pentru predicţia bazată pe rute

Criteriu	**Descriere**	**Min (0% probabilitate)**	**Max (100% probabilitate)**	**Pondere**
Frecvenţa rutei	Numărul de utilizări al unei rute	Niciodată	Folosită de mai mult de 10 ori	4
Frecvenţa rutei într-un anumit interval de timp	Numărul de utilizări al unei rute într-un anumit interval de timp (de la -1 oră la +2 ore) faţă de ora curentă	Niciodată	Folosită de mai mult de 10 ori	1
Frecvenţa rutei într-o anumită zi	Numărul de utilizări al unei rute în ziua curentă din săptămână	Niciodată	Folosită de mai mult de 10 ori	1
Frecvenţa rutei într-un anumit grup de zile	Numărul de utilizări al unei rute într-un anumit grup de zile (e.g. luni-vineri)	Niciodată	Folosită de mai mult de 10 ori	1
Ultima utilizare a unei rute	Diferenţa dintre ultima utilizare a rutei şi ora/data curentă	>4 săptămâni	<2zile	3
Distanţa până la destinaţie	Distanţa dintre poziţia actuală şi destinaţie	>100km	0km	1

5.3.5 Predicţii bazate pe filtrarea rutei în funcţie de distanţa de la poziţia curentă la waypoint-urile rutelor

Criteriile definite în tabela 5.2, "Criterii de prioritizare pentru predicţia bazată pe rute" sunt extinse prin adăugarea următoarelor criterii:

TABELA 5.3: Criterii de prioritizare pentru predicţia bazată pe rutele din jurul unei poziţii

Criteriu	**Descriere**	**Min (0% probabilitate)**	**Max (100% probabilitate)**	**Pondere**
Distanţa până la rută	Distanţa dintre poziţia actuală şi waypoint-urile rutei	> 1km	0km	1
Direcţia către rută	Diferenţa dintre orientarea waypoint-urilor şi direcţia de navigare	180°	0°	1

5.3.6 Predicţii bazate pe filtrarea rutei în funcţie de distanţa de la poziţia curentă la punctele de start ale rutelor

Criteriile definite în tabela 5.2, "Criterii de prioritizare pentru predicţia bazată pe rute" sunt extinse prin adăugarea următoarelor criterii:

TABELA 5.4: Criterii de prioritizare pentru predicţia bazată pe filtrarea rutele în funcţie de distanţa până la punctul de start al rutei

Criteriu	**Descriere**	**Min (0% probabilitate)**	**Max (100% probabilitate)**	**Pondere**
Distanţa până la punctul de start al rutei	Distanţa până la cel mai apropiat punct de start al rutei	> 3km	0km	1

5.3.7 Predicţii bazate pe filtrarea rutei în funcţie de timpul scurs până la ajungerea la destinaţie

Pentru acest caz sunt folosite criteriile definite în tabela 5.2, "Criterii de prioritizare pentru predicţia bazată pe rute".

Capitolul 6
MANAGEMENTUL ERORILOR

6.1 Tipuri de erori

- Erori de secvenţă
- Erori apărute la accesarea bazei de date
- Bază de date coruptă

6.2 Detectarea erorilor

Erorile sunt raportate pentru fiecare apel către şi dinspre interfaţa modului de predicţie. Toate funcţiile returnează un număr ce corespunde unui tip de eroare.

6.2.1 Erori de secvenţă

O maşină de stare va verifica încălcarea ordinii secvenţelor.

6.2.2 Erori la accesarea bazei de date

Trebuie evaluate rezultatele funcţiilor native de accesare ale bazei de date.

6.2.3 Bază de date coruptă

Trebuie evaluate rezultatele funcţiilor native de accesare ale bazei de date.

6.3 Tratarea erorilor

În funcţie de tipul de eroare, aceasta poate fi prevenită pe viitor sau nu de către dezvoltator.

Există erori de secvenţă precum “Înainte de apelarea funcţiei X este necesară pornirea unităţii software PNavPredictor”. Se poate întâmpla de asemenea ca atunci când dezvoltatorul porneşte procesul de predicţie de două ori consecutiv sa fie întâmpinat de eroarea “Procesul de predicţie se

află deja în curs de rulare". În astfel de cazuri este clar cum se pot preveni erorile.

Mai sunt însă şi cazuri ce nu pot fi tratate de către dezvoltator. Astfel de erori sunt cele precum "Baza de date este coruptă", ce pot să apară în cazul în care fişierul de sistem folosit pentru stocarea datelor este corupt. În aceste situaţii datele stocate anterior nu mai pot fi recuperate, toate informaţiile referitoare la rute fiind definitiv pierdute.