**Smart Parking Assistant**

 Roca Denis Dominic

2E1

Faculty of Computer Science

Univeristatea Alexexandru Ioan Cuza, Iasi  Bulevardul Carol I, nr. 11

**1. Introducere**

Proiectul SmartParkingAssistant (B) reprezintă o aplicație client-server concepută pentru a asista șoferii în găsirea locurilor de parcare disponibile în timp real. Scopul său principal este de a colecta informații despre disponibilitatea locurilor de parcare utilizând senzori și camere inteligente, și de a le furniza șoferilor prin intermediul unei interfețe client.

**2. Tehnologii Aplicate**

**2.1 TCP**

Pentru realizarea comunicării între client și server, proiectul utilizează protocolul TCP (Transmission Control Protocol). Acesta oferă o conexiune fiabilă și bidirecțională între clienții aplicației și serverul central.

**2.2 Baza de Date cu Jansson**

Datele colectate despre disponibilitatea locurilor de parcare sunt stocate într-o bază de date. Pentru manipularea și gestionarea datelor în format JSON, proiectul utilizează biblioteca Jansson. Jansson oferă funcționalități eficiente pentru serializarea și deserializarea datelor în format JSON, facilitând astfel interacțiunea între server și baza de date.

**2.3 Socket**

Comunicarea între client și server este realizată prin intermediul socket-urilor. Sockets permit transmiterea datelor între dispozitivele conectate la rețea. Serverul ascultă pe un anumit port și acceptă conexiuni de la clienți. Fiecare client și server comunică prin intermediul unui socket dedicat, asigurând astfel un flux eficient și sigur de date între părți.

**2.4 Interfața GTK**

Interfața utilizatorului este realizată folosind biblioteca GTK. GTK este o bibliotecă multi-platformă pentru crearea de interfețe grafice. Este potrivită pentru proiecte de orice dimensiune și este disponibilă pe majoritatea sistemelor de operare moderne. GTK este scrisă în C, dar a fost proiectată pentru a fi folosită cu multe limbaje de programare diferite. În acest proiect, GTK este folosit pentru a crea o interfață grafică intuitivă și ușor de utilizat pentru utilizatori.

**3. Arhitectură Generală a Proiectului**

Proiectul este structurat în două componente principale:

•           **Server:** Gestionează comunicarea cu clienții și colectează datele despre disponibilitatea locurilor de parcare. Implementează funcționalități de stocare și manipulare a datelor utilizând baza de date Jansson.

•           **Client:** Furnizează o interfață utilizator pentru șoferi, permitându-le să solicite informații despre locurile de parcare disponibile. Comunică cu serverul pentru a primi datele actualizate în timp real.

**Instrucțiuni de Utilizare**

Pentru a utiliza aplicația SmartParkingAssistant (B), urmați pașii de mai jos:

•           **Pasul 1: Porniți Serverul**

•           Asigurați-vă că serverul este pornit și ascultă pe portul specificat.

•           **Pasul 2: Deschideți Aplicația Client**

•           Porniți aplicația client pe dispozitivul dumneavoastră.

•           **Pasul 3: Interacționați cu Interfața**

•           Utilizați interfața clientului pentru a solicita informații despre locurile de parcare disponibile.

•           **Pasul 4: Primiți Actualizări în Timp Real**

•           Clientul va primi actualizări în timp real de la server cu privire la disponibilitatea locurilor de parcare. El va avea acces la următoarele comenzi :

* « Autentificare » dacă utilizatorul are deja un cont, va putea să se autentifice cu acesta introducând numele și parola
* « Creare cont » dacă utilizatorul nu are un cont în apilație, are posibilitatea de ași crea un cont

După ce utilizatorul este autentificat, va putea accesa și următoarele comenzi care sunt disponibile doar cât timp este conectat.

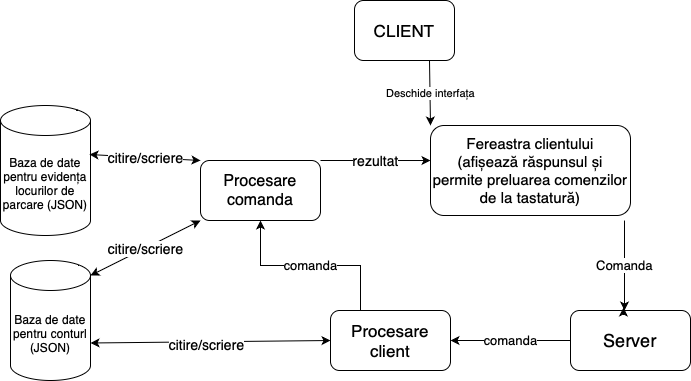
* « Cauta un loc de parcare » primește numărul de locuri libere și locul cel mai apropiat pe care poate să-l ocupe
* « Parcheaza » pentru a ocupa locul găsit.

Această comandă îi va cere clientului să selecteze metoda de taxare.

* «Abonament » se eliberează un abonament disponibil 14 zile.
* «Plata » clientul va plăti 10 lei/oră.
* « Deconectare

•           Administratorul se poate conecta la server prin acceeasi aplicatie ca si clientii si are acces în plus la următoarele comenzi:

* „Quit server” -> această comanda va închide serverul și va deconecta clientul.
* „Inițializare sistem” -> administratorul are posibitatea ca să inițializeze fișierele JSON corespunzătoare bazei de date care stochează informații despre conturile clienților și locurile de parcare.



**4. Aspecte de implementare**

Pentru gestionarea conexiunilor și a autentificării, serverul utilizează fire de execuție multiple (multi-threading) pentru a permite procesarea concurentă a mai multor clienți. Codul relevant din funcția **main()** este prezentat mai jos:

int main() {

…

while (1) {

Client \*client = (Client \*)malloc(sizeof(struct Client));

…

if (pthread\_create(&client\_thread, NULL, client\_handler, (void \*)client) < 0) {

perror("Eroare la crearea thread-ului");

free(client);

return 1;

}

}

return 0;

}

Pentru implementarea autentificării și gestionarea conturilor utilizatorilor, se utilizează funcția **client\_handler**, care este apelată pentru fiecare client conectat. Acesta primește comenzile de la client și gestionează procesul de autentificare. Codul relevant este prezentat mai jos:

void \*client\_handler(void \*arg) {

…

while (client->isAuthenticated == 0) {

…

if (strncmp(command, "Creare cont", 11) == 0) {

// ... (codul pentru crearea contului)

} else if (strncmp(command, "Autentificare", 13) == 0) {

// ... (codul pentru autentificare)

} else {

strncpy(buffer, "Comandă necunoscuta!", 22);

send(client\_sock, buffer, strlen(buffer), 0);

memset(command, 0, sizeof(command));

memset(buffer, 0, BUFFER\_SIZE);

continue;

}

}

// După autentificare, se utilizează funcția handleCommand pentru gestionarea comenzilor.

while (client->isAuthenticated == 1) {

int bytes\_received = recv(client\_sock, command, BUFFER\_SIZE, 0);

if (bytes\_received <= 0) {

perror("Eroare la primirea comenzii");

break;

} else {

handleCommand(command, client, park, client\_sock);

goto restart;

}

}

return NULL;

}

După autentificarea cu succes, serverul utilizează funcția **handleCommand** pentru a procesa comenzile ulterioare primite de la client. Această funcție este responsabilă pentru interpretarea și executarea comenzilor specifice, comunicând cu clientul și manipulând informațiile din structurile de date corespunzătoare.

**5. Concluzie**

Proiectul SmartParkingAssistant (B) reprezintă o soluție inovatoare pentru gestionarea și furnizarea informațiilor despre locurile de parcare disponibile în timp real. Implementarea utilizând TCP, Jansson și sockets asigură o comunicare eficientă și securizată între client și server, contribuind la facilitarea experienței șoferilor în găsirea locurilor de parcare.

Această documentație oferă o privire de ansamblu asupra proiectului SmartParkingAssistant (B), descriind tehnologiile aplicate, arhitectura generală și instrucțiunile de utilizare. Proiectul aduce beneficii semnificative în domeniul parcărilor inteligente și îmbunătățește experiența de parcare a șoferilor.

Pe viitor pot fi adăugate și alte îmbunătățiri precum:

* Gestionarea mai multor abonamente;
* Adaptarea aplicației în funcție de parcare și cerințele ei ;
* Ocuparea mai multor locuri de parcare pe numele aceluași client ;
* O interfață grafică mai ușor de folosit ;
* Etc…

**6. Surse**

<https://profs.info.uaic.ro/~computernetworks/cursullaboratorul.php> <https://profs.info.uaic.ro/~georgiana.calancea/laboratories.html>

<https://docs.gtk.org>