Digital Menu Board

Daniela Duțescu[0009−0005−5052−0658]

Universitatea Alexandru Ioan Cuza, Facultatea de Informatică, 22, Bulevardul Carol I, Iaşi RO – 700505

1 Introducere

Abstract. În Est-ul Europei de astăzi încă se mai găsesc o mulțime de restaurante și localuri care își servesc clienții cu aceleași meniuri printate și cartonate, spre deosebire de Occident, unde avem de a face cu o competiție ceva mai severă. Acest fapt nu duce la nimic altceva decât îndepărtarea, poate chiar înstrăinarea, generațiilor noi, ce devin tot mai ancorate în tehnologie. Totodată, dificultatea modificării unui meniu fizic este sporită în comparație cu cea a modificării meniurilor digitale și este, de asemena, mai scumpă. Așadar, includerea oportunităților asemenea soluției *Digital Menu Board* în rândul restaurantelor ce au un oarecare regres față de altele, va ajuta semnificativ la amploarea lor, deci implicit și la economie. Desigur, nu ne vom aștepta la un surplus de sume colosale venite datorită acestei soluții, dar cu siguranță vor avea o contribuție mai valoroasă în cele ce urmează în comparație cu starea inițială.

2 Tehnologii aplicate

Proiectul este concentrat pe structura server/client, conceput prin intermediul limbajului de programare C/C++. Pentru realizarea conexiunii dintre client și server vom utiliza protocolul TCP/IP (Transmission Control Protocol) realizat în mod concurent cu multiplexare I/O folosind select(). Motivația alegerii acestui protocol constă în caracteristica sa de bază și anume securitatea. Cum soluția de față oferă managerilor restaurantelor putere deplină în gestionarea meniurilor sale, de la prețuri la ingrediente, nu am dori ca acesta din urmă să descopere că mâncărurile și rețetele sale se regăsesc în multe alte localuri la un preț mai avantajos, fapt ce i-ar întări concurența pe piață. Desigur că nu se rezumă totul la siguranța datelor transmise de la distanță pentru actualizarea meniului, protocolul ales va garanta, de asemenea, și siguranța procesului de logare a utilizatorilor. Astfel protocolul TCP/IP ne oferă toate "ingredientele" necesare pentru a ne construi meniul ideal și într-o manieră mai sigură.

3 Structura soluției

Avem de-a face cu o structură compactă ce ne garantează o mentenanță mai ușoară în comparație cu multe alte idei. Astfel, vom avea de-a face cu o primă entitate, server-ul, ce va avea ca rol principal procesarea comenzilor (și a parametrilor

acestora unde este cazul) și efectuarea operațiilor ce țin strict de comanda curentă introdusă de către cea de-a doua entitate a structurii, clientul. Clientul este sursa acțiunilor, acesta va solicita server-ul cu mai multe comenzi deja predefinite, în scopul acoperirii nevoilor sale. Mai multe detalii puteți vizualiza în Fig. 1 de mai jos.

Așadar:

Server-ul va avea în vedere următoarele:

1. Acceptarea și verificarea conexiunilor realizate de către client;
2. Procesarea comenzilor/acțiunilor primite de la client;
3. Oferirea resurselor și/sau respectarea comenzilor clientului;
4. Garantarea securității informațiilor clientului;

Clientul va avea în vedere următoarele:

1. Conectarea la server;
2. Trimiterea diverselor acțiuni server-ului în vederea prelucrării meniului restaurantului: Conectare, Afișare, Export, Modificare, Deconectare.

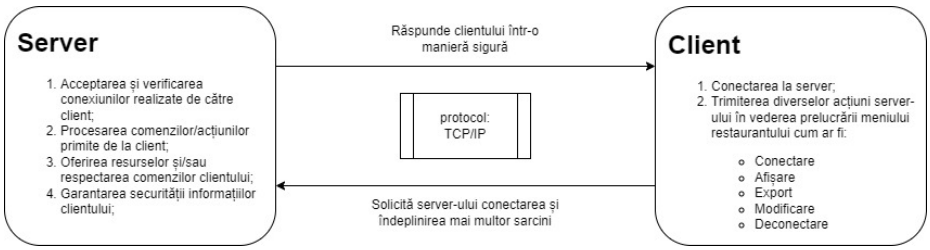

Fig. 1: Diagramă sugestivă pentru structura soluției.

4 Aspecte de Implementare

În contextul aplicației *Digital Menu Board* vom proiecta două programe, reprezentând clientul, respectiv server-ul, pe baza protocolului TCP/IP pentru a garanta securitatea conexiunii clientului cu server-ul, dar și integralitatea informațiilor din baza de date a clientului. În cadrul programului client se realizează protocolul de comunicare, unde solicităm conectarea securizată la server prin crearea socket-ului și inițializarea cu adresa IP și portul server-ului din linia de comandă.(A se observa în Fig.2). Clientul face conexiunea la server cu ajutorul instanței *connect()* pentru ca mai departe, să transmită la server intrucțiunea introdusă de managerul restaurantului.(Fig. 3) După procesare, server-ul trimite datele aferente/confirmarea conform instrucțiunii primite. De exemplu, pentru instrucțiunea de conectare, clientul va introduce comanda alături de numele acestuia și va primi o confirmare de la server ca s-a conectat, iar, pentru instrucțiunea de afișare, clientul va primi în terminal afișarea meniului curent.

```c
/* cream socketul */
if ((sd = socket(AF_INET, SOCK_STREAM, 0)) == -1)
{
  perror("[client] Eroare la socket().\n");
  return errno;
}

/* umplem structura folosita pentru realizarea conexiunii cu serverul */
/* familia socket-ului */
server.sin_family = AF_INET;
/* adresa IP a serverului */
server.sin_addr.s_addr = inet_addr(argv[1]);
/* portul de conectare */
server.sin_port = htons(port);
```

Fig. 2: Crearea socket-ului și inițializarea cu adresa IP și portul server-ului

```c
/* ne conectam la server */
if (connect(sd, (struct sockaddr *)&server, sizeof(struct sockaddr)) == -1)
{
  perror("[client]Eroare la connect().\n");
  return errno;
}
```

Fig. 3: Conectarea clientului la server

În structura arhitecturii server-ului, se monitorizează acceptarea conexiunilor cu clienții și servirea concurentă a acestora. Inițial, server-ul ascultă la port noi conexiuni(a se vedea Fig.4), utilizează *select()*(Fig.5) cu scopul de a controla mai mulți descriptori în același timp, adică stabilește conexiunea cu clienții prin intermediul multiplexării, pentru ca ulterior, să accepte clienți noi(Fig.6), să primească instrucțiuni de la aceștia și să le ofere informațiile corespunzătoare comenzilor. De exemplu, pentru instrucțiunea de conectare, server-ul va primi comanda împreună cu numele restaurantului, va verifica dacă acesta există cu un meniu în baza de date, și îi va confirma/infirma accesul în urma verificării. În continuare, voi prezenta interacțiunea așteptată între client și server după cum urmează: Managerul restaurantului se va conecta la server, va putea să vizualizeze meniul, să insereze și să șteargă produse, să aducă modificări asupra meniului(denumirea produsului, ingrediente, preț), să se deconecteze și să repete aceste aceste instrucțiuni de oricâte ori dorește și este conectat într-o sesiune cu server-ul. Server-ul va procesa comenzile primite de la client și îi va răspunde conform implementării fiecărei instrucțiuni; în plus, îi va trimite de fiecare dată un mesaj de confirmare cu ce operațiune/modificare a făcut.

```c
/* atasam socketul */
if (bind(sd, (struct sockaddr *)&server, sizeof(struct sockaddr)) == -1)
{
  perror("[server] Eroare la bind().\n");
  return errno;
}

/* punem serverul sa asculte daca vin clienti sa se conecteze */
if (listen(sd, 10) == -1)
{
  perror("[server] Eroare la listen().\n");
  return errno;
}
```

Fig. 4: Atașarea socketului și ascultarea noilor conexiuni

```
/* apelul select() */
if (select(nfds + 1, &readfds, NULL, NULL, &tv) < 0)
{
  perror("[server] Eroare la select().\n");
  return errno;
}
```

Fig. 5: Multiplexarea evidențiată în cadrul select()

```
/* a venit un client, acceptam conexiunea */
client = accept(sd, (struct sockaddr *)&from, &len);
```

Fig. 6: Acceptarea unui client dacă acesta este pregătit

5 Concluzii

Pentru moment, aplicația *Digital Menu Board* acoperă cerințele prezentului prin interacțiunea directă a proprietarului de restaurant cu meniul acestuia, dar, având în vedere faptul că ne aflăm într-o eră a inteligenței artificiale, vom dori automatizarea acestei soluții prin prisma IA. Așadar, o versiune supremă a soluției *Digital Menu Board* ar consta în conceperea unui algoritm de învățare automată care să fie capabil să colecteze suficiente recenzii din partea clienților la adresa meniului și să modifice și/sau să actualizeze conținutul acestuia în mod optim.

References

1. Pagina cursului de Rețele de Calculatoare, https://profs.info.uaic.ro/~computernetworks/cursullaboratorul.php.
2. https://en.cppreference.com/.