LearNet

Cozma Damian-Constantin

Universitatea Alexandru Ioan Cuza Iași, Facultatea de Informatică
damian.cozma@student.uaic.ro

1 Introducere

Proiectul LearNet propune dezvoltarea unui sistem cu arhitectură client-server, destinat facilității invățării noțiunilor fundamentale din domeniul Rețelelor de calculatoare. Sistemul îmbină gestionarea eficientă a informațiilor pe partea de server cu o interfață grafică intuitivă pe partea de client, pentru a oferi utilizatorilor o experiență educațională modernă.

2 Tehnologii Aplicate

Pentru implementarea proiectului, am utilizat tehnologii precum:

- **GTK**: pentru realizarea unei interfețe grafice; [1] [2]
- **CSS**: pentru stilizarea interfeței grafice și îmbunătățirea experienței utilizatorilor; [3]
- **TCP**: pentru conexiuni fiabile și transfer de date securizat; [4]
- **Multiplexare I/O folosind select()**: pentru gestionarea eficientă a comunicării concurente între mai mulți clienți; [5]
- **Fișiere JSON**: pentru gestionarea dicționarului, reținerea informațiilor despre utilizatori (cereri de prietenie, listă de prieteni, bookmark-uri) și salvarea discuțiilor de pe diferitele subiecte din secțiunea de Discuții; [6] [7]
- **Autentificare**: Deși autentificarea nu este menționată explicit în enunțul proiectului, aceasta reprezintă o componentă esențială pentru gestionarea funcționalităților, cum ar fi păstrarea datelor utilizatorilor, administrarea listei de prieteni sau organizarea forumului. În implementarea curentă, autentificarea se realizează prin intermediul unui fișier de tip '.config'. Deși această metodă nu este una sigură, a fost aleasă pentru a demonstra funcționalitățile proiectului. În eventualitatea dezvoltării ulterioare a aplicației, voi considera integrarea unor tehnologii moderne de autentificare, precum OpenSSL sau alte soluții criptografice avansate.

3 Structura Aplicației

Serverul aplică un model centralizat pentru managementul datelor și comunicarea între clienți. Acesta ascultă conexiuni printr-un socket TCP/IP și gestionează cererile multiple folosind multiplexarea. Funcționalitățile principale ale

serverului includ autentificarea utilizatorilor, distribuția informațiilor solicitate și gestionarea mesajelor între utilizatori. Serverul procesează cererile conform unui protocol definit și răspunde în timp real.

Clientul utilizează biblioteca GTK pentru a oferi o interfață grafică intuitivă, cu ferestre și componente interactive, precum butoane, câmpuri de text și panouri de afișare. Fluxul aplicației pe partea de client începe cu autentificarea utilizatorului, urmată de accesarea diferitelor funcționalități, inclusiv căutarea de informații, gestionarea discuțiilor și interacțiunea cu lista de prieteni. Conexiunea clientului cu serverul este asigurată prin sockets, iar răspunsurile serverului sunt afișate în timp real în interfața grafică.

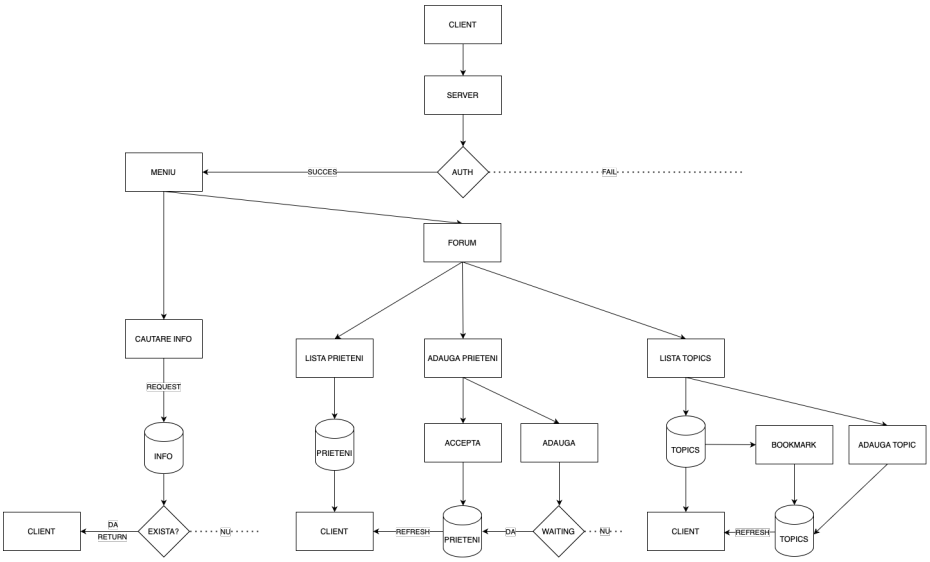

Fig. 1. Diagrama

4 Aspecte de Implementare

4.1 GTK (Interfața Grafică)

Inițializarea GTK :

Inițializarea aplicației și configurarea principală a interfeței grafice se realizează astfel:

```
1 gtk_init(&argc, &argv); //initializeaza GTK
```

```
CSS(); //aplica stiluri CSS
window_login(); /*afiseaza fereastra de login*/
gtk_main(); /*esentiala in GTK, intra intr-un loop
de evenimente */
```

Gestionarea Semnalelor și Evenimentelor :

Fiecare acțiune a utilizatorului, cum ar fi apăsarea unui buton, este captată prin semnale. Conectarea semnalelor se realizează astfel:

```
g_signal_connect(button, "clicked", G_CALLBACK(on_button_click)
    , NULL);
```

Funcția din G CALLBACK este apelată la apăsarea butonului.

Actualizare în timp real :

Pentru a asigura actualizarea în timp real a mesajelor din diferitele discuții de pe forum, am implementat o funcție de refresh automat utilizând 'g-timeout-add' [8]. Aceasta elimină necesitatea ca utilizatorul să închidă și să redeschidă aplicația sau să navigheze între meniul principal și secțiunea de forum pentru a vedea modificările din fișierul JSON. Astfel, orice schimbare realizată este afișată imediat, îmbunătățind considerabil experiența utilizatorului.

```
refresh_timer_id = g_timeout_add(3000, refresh_messages,
    text_view);
```

4.2 Baza de Date și Interacțiunea cu Serverul

Prefixarea Mesajelor de către Client :
Serverul primește mesaje de la client prefixate în funcție de acțiunea dorită. Exemple:

```
// Exemplu 1: Adaugarea unui topic
if (strncmp(msg, "serverTOPIC", 12) == 0) {
    const char *topic = msg + 12;
    sprintf(msgrasp, "Topicul␣'%s'␣a␣fost␣adaugat.\n", topic);
}

// Exemplu 2: Adaugarea unui bookmark
else if (strncmp(msg, "serverBOOKMARK:", 14) == 0) {
    const char *topic = msg + 14;
    sprintf(msgrasp, "Topic-ul␣'%s'␣a␣fost␣adugat␣la␣bookmark!"
    , topic_name);
}
```

4.3 JSON

Cererile de Prietenie :

Atunci când un utilizator adaugă un prieten, serverul actualizează fişierul JSON pentru a reflecta cererea de prietenie:

```
"pending_requests": {
    "Ioana": ["Damian"]
}
```

Daca cererea este acceptată:

```
"friends": {
    "Damian": ["Ioana"],
    "Ioana": ["Damian"]
}
```

Discuțiile de pe Forum :

Serverul înregistrează automat noile topicuri:

```
{
    "HTTP sau HTTPS": []
}
```

Numele utilizatorilor sunt prefixate automat:

```
{
    "HTTP sau HTTPS": [
        "[IoanaR]: Care este diferenta dintre HTTP si HTTPS?"
    ]
}
```

Dicționarul și Căutarea Informațiilor :

Structura fișierului JSON asociat dicționarului:

```
"TCP": {
    "description": "Protocol utilizat pentru conexiuni fiabile
    si transferuri de date.",
    "advantages": [
        "Fiabilitate in livrarea datelor.",
        "Transmiterea datelor in ordinea corecta."
    ],
    "examples": [
        "Navigare pe web prin HTTP/HTTPS.",
```

```
        "Transfer de fisiere prin FTP."
    ]
}
```

Utilizatorul poate introduce ”Help” pentru a vedea toate cuvintele-cheie disponibile.

4.4 CSS

Îmbunătățirea Interfeței Grafice :

În cadrul acestui proiect, CSS a fost utilizat pentru personalizarea interfeței grafice:

```
void CSS() {
    GtkCssProvider *provider = gtk_css_provider_new();
    GdkDisplay *display = gdk_display_get_default();
    GdkScreen *screen = gdk_display_get_default_screen(display)
    ;

    gtk_css_provider_load_from_path(provider, "style.css", NULL
    );
    gtk_style_context_add_provider_for_screen(screen,
    GTK_STYLE_PROVIDER(provider),
    GTK_STYLE_PROVIDER_PRIORITY_USER);
}
```

Exemplu de stiluri definite în ‘styles.css‘:

```
/*adaugarea imaginilor de fundal realizate pe Canva*/
#window-topic {
    background-image: url(’background-main.png’);
    background-size: cover;
    background-position: center;
}

/*design imbunatatit pentru butoane*/ /
button {
    font-family: "Avenir", sans-serif;
    background-color: #ffffff;
    border: 1px solid #e6e6e6;
    border-radius: 5px;
    padding: 6px 12px;
    font-size: 14px;
    transition: background-color 0.3s ease;
}
```

Aspecte buton CSS, [9] materiale realizate pe Canva. [10]

5 Concluzii

La început, lucrul cu GTK a fost o provocare considerabilă, în special din perspectiva realizării interfeței grafice. Cu toate acestea, prin studiu și practică, am reușit să dobândesc o înțelegere solidă a principalelor concepte și a modului în care GTK poate fi utilizat eficient.

Un alt aspect important al acestui proiect a fost lucrul cu fișiere JSON. Acesta a fost o oportunitate excelentă de a învăța să manipulez și să structurez datele într-un mod eficient și bine organizat. Am înțeles mai bine cum să utilizez JSON pentru a gestiona informații complexe, cum ar fi listele de prieteni, discuțiile și datele din dicționar.

În plus, acest proiect mi-a permis să îmi solidific cunoștințele în CSS. Am utilizat CSS nu doar pentru stilizarea interfeței, ci și pentru a adăuga un strat de profesionalism și estetică aplicației. Personalizarea butoanelor, gestionarea spațierii și crearea unui design unitar și prietenos cu utilizatorul au fost aspecte deosebit de satisfăcătoare.

Am ales să lucrez la acest proiect deoarece m-a atras ideea dezvoltării unei aplicații cu o interfață grafică, care este, pentru mine, cea mai captivantă parte a dezvoltării unei aplicații. Am petrecut o mare parte din timp perfecționând aspectul UI, ajustând elementele astfel încât să creez o interacțiune plăcută și eficientă pentru utilizator. Uneori, modificările realizate în UI au dus la mici probleme sau comportamente neașteptate ale altor elementelor grafice și trebuia să mă întorc la ele fără să afectez și mai multe, însă procesul de soluționare a acestor provocări a fost extrem de satisfăcător. În final, rezultatul a fost o interfață grafică care nu doar funcționează bine, dar reflectă și atenția mea pentru detalii și dorința de a crea o experiență captivantă pentru utilizatori.

Acest proiect m-a ajutat să îmi consolidez abilitățile tehnice, să descopăr noi aspecte ale dezvoltării aplicațiilor și să apreciez și mai mult complexitatea procesului de realizare a unei aplicații complete.

6 Referințe Bibliografice

References

1. GTK3 Documentation.
2. GTK Getting Started.
3. GTK/CSS Overview.
4. TCP - Retele de Calculatoare, FII.
5. TCP Concurent - Retele de Calculatoare, FII.
6. JSON - Introduction.
7. JSON - C.
8. GTK GLib, timeOutAdd.
9. CSS Buttons.
10. Canva.