PENGUCHAT

Anastasiu Alexandru-Ioan, clasa a XII-a C

Profesor coordonator Smîntînă Rodica

Colegiul Național "Gheorghe Șincai"
An școlar 2020-2021

Cuprins

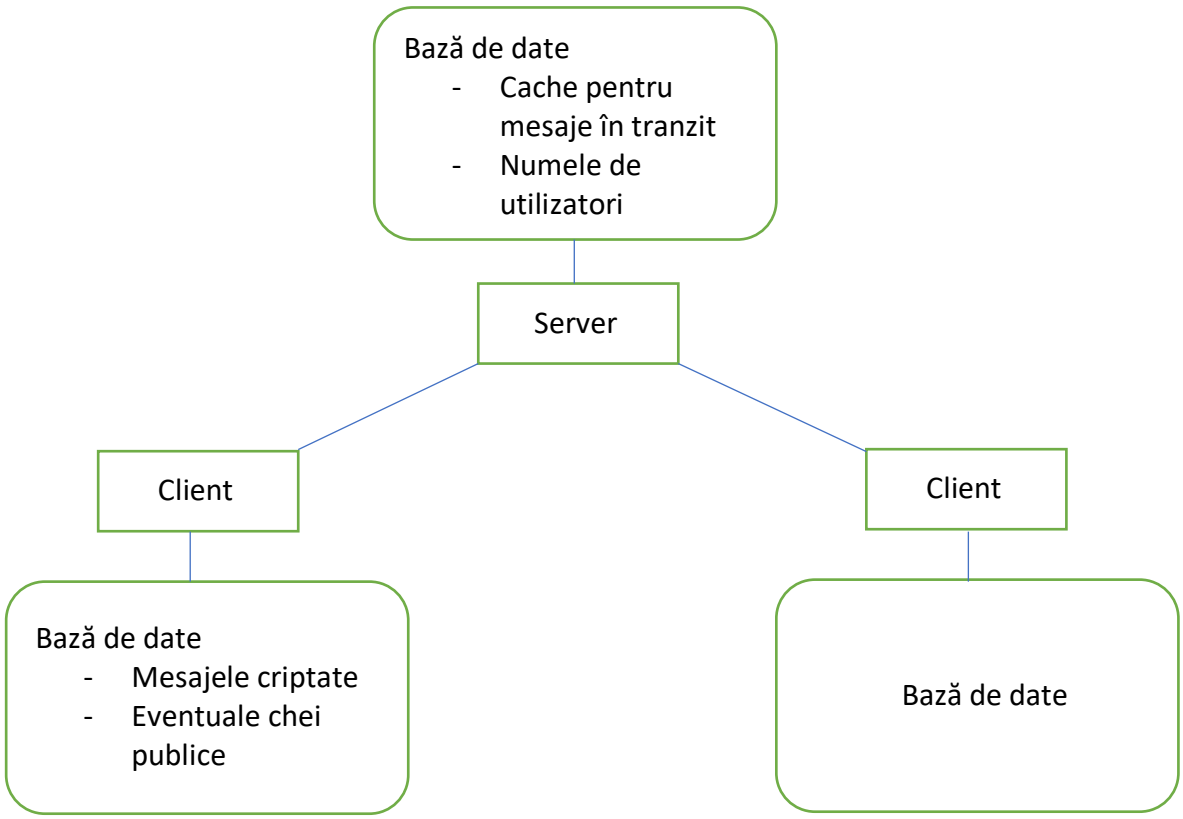
Prezentare generală

Proiectul este o aplicație de chat criptată end-to-end, precum Whatsapp sau Telegram, permițând utilizatorilor să aibă conversații private. Este prevăzut cu un sistem de autentificare de tip username-parolă și destinat utilizării în interiorul unui intranet. În plus, permite nu doar trimiterea de mesaje text, ci și de fișiere. Programul este scris în limbajul de programare Python, interfața folosește framework-ul Kivy, iar anumite module sunt scrise în Cython, pentru eficientizare.

Tehnologii utilizate

PenguChat este un ecosistem de aplicații, ce se compune dintr-o aplicație tip server (care rulează pe o mașină centrală, fiind un releu, „poștașul” ce transmite mesajele) și una de tip client, ce are o interfață grafică, fiind modul în care utilizatorii trimit mesaje.

Pentru a garanta securitatea, mesajele trimise nu sunt stocate pe server, ci pe mașinile fiecăruia dintre clienți (serverul le salvează doar atunci când sunt în tranzit și destinatarul nu este on-line). Astfel, fiecare client are o bază de date (SQLite) salvată local, populată cu mesajele criptate. Decriptarea se face doar atunci când aplicația este pornită și utilizatorul autentificat.



Pentru comunicare, este folosit protocolul TCP, mesajele fiind segmentate în pachete. Acestea sunt transmise în plaintext, formatate ca JSON-uri, având o secțiune vizibilă (header, denumirea fiecărui câmp, de exemplu „command” sau „timestamp”) și o secțiune cifrată (content).

Criptarea se realizează prin intermediul algoritmului AES, unul socotit a fi extrem de sigur. End-to-end se referă la faptul că doi clienți, la momentul în care încep o conversație, vor negocia o cheie comună de criptare, astfel încât nici măcar serverul să nu poată decripta conținutul mesajelor.

Pentru a face acest lucru, are loc un schimb de chei, folosind algoritmul Diffie-Hellman: primul client, trimițând o cerere de prietenie celui de-al doilea, își oferă cheia publică. Al doilea client, acceptă cererea, moment în care două lucruri se întâmplă:

- Al doilea client generează cu ajutorul cheii publice primite și propriei chei private o cheie comună, folosită de aici încolo pentru comunicare
- Finalizând acest proces, trimite primului propria sa cheie publică, pentru ca acesta să repete procesul.

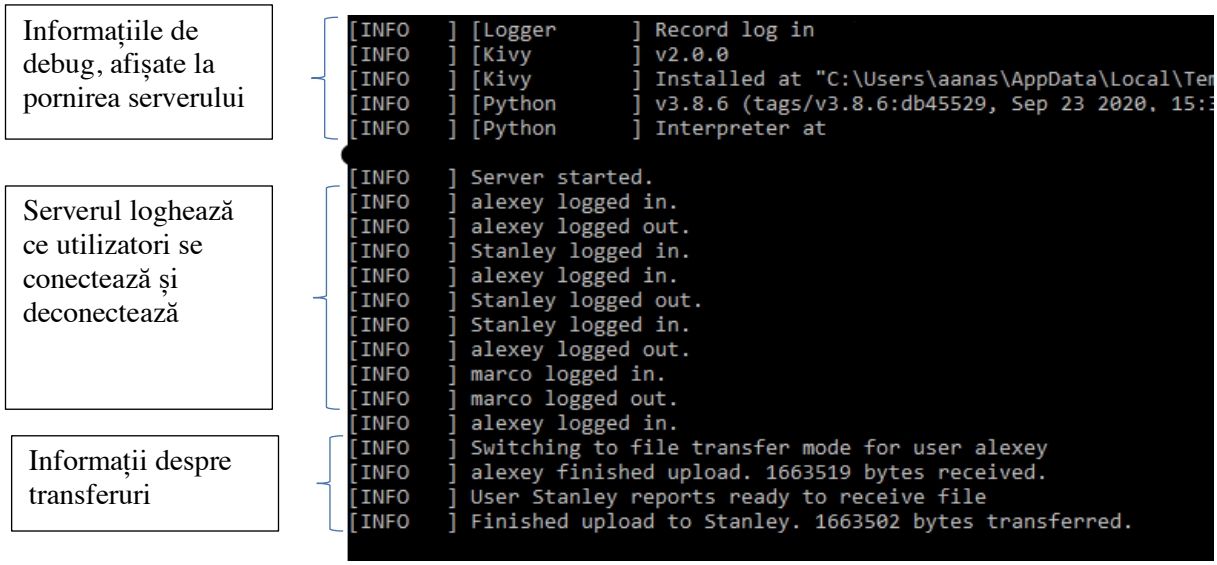
Rezultatul este că, printr-un canal presupus nesigur, are loc un schimb de chei, ce duce la generarea unei chei comune. Un atacator nu ar putea folosi cheile publice pentru a-și genera singur această cheie necesară.

Transferul de fișiere este tratat similar cu cel al mesajelor text. Un „pachet de intenție” atenționează serverul că urmează un transfer de informație binară, și că nu ar trebui să încerce să parseze informația primită. Serverul trece, astfel, pentru trimițător în modul de transfer de fișiere. Clientul va lua datele citite din fișier, le va cripta, apoi le va împărți în blocuri, astfel încât să se respecte limita de 64KB permisă unui pachet TCP. Este treaba destinatarului să reasambleze fișierul, apoi să îl decripteze.

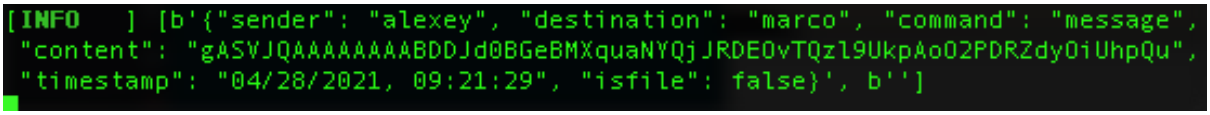
Mod de funcționare

Serverul

Serverul rulează pe o mașină dedicată, un computer central. Este os-agnostic (poate rula cu succes pe orice sistem de operare), dar preferabil este să funcționeze sub un sistem de tip UNIX-like, deoarece nu are o interfață. Consola este modul prin care utilizatorul (administratorul de sistem) interacționează cu aplicația.



La momentul conectării la server, fiecărui username de client îi este asociat un ip (adresa publică) și un port. Astfel, atunci când serverul primește cererea să trimită un mesaj la un utilizator, el va verifica dacă are salvată în memorie o adresă pentru el. În caz contrar, marchează mesajul ca fiind „în tranzit” și îl salvează în baza de date (cache), de unde urmează să fie trimis la momentul când destinatarul se conectează.



Deasupra este prezentată o radiografie a unui pachet. Acesta conține trimițătorul, destinatarul, o comandă pentru server (în acest caz „trimite mesaj”), și conținutul criptat al mesajului.

Clientul

Clientul rulează pe fiecare dintre mașinile clienților, conectându-se la serverul central. Ca și backend, el se constituie din trei module:

- Modulul principal se ocupă de comunicare și schimb de informații, dar și inițializează interfața grafică.
- Modulul numit DBHandler, ce se ocupă de interfațarea cu baza de date. Pe lângă procesarea de date, acesta se ocupă și de securitate, prevenind expunerea directă a bazei de date către rețeaua exterioară. Toate apelurile către ea se fac prin intermediul API-ului aplicației, prevenind astfel metode de atac precum SQL injection (executarea de cod arbitrar prin introducerea de input malformat).
- Modulul numit UIElements, împreună cu regulile definite în fișierul PenguChat.kv conțin instrucțiunile pentru framework-ul ce se ocupă de rendering. Aici sunt definite fiecare ecran, fiecare element (bulele de chat, de exemplu). Mai jos, poate fi văzută declararea unui buton.

```
<MenuButton>:
    background_normal: 'assets/fallback.png'
    background_color: 0, 0.413, 0.586
    canvas.before:
        Color:
            rgb: 0, 0.213, 0.286
        Line:
            width: 2
            rectangle: self.x, self.y, self.width, self.height
```

Backend-ul clientului

Din punct de vedere al backend-ului, procesul de login și cel de signup se desfășoară similar. Între server și client are loc un handshake, o schimbare de chei, fiind creat astfel un tunel criptat. Username-ul și parola sunt transmise serverului, care ori autentifică utilizatorul (trimite un cod de răspuns tip „succes”, apoi asociază ip-ul cu userul în memoria internă), ori permite o nouă încercare (trimite cod de răspuns tip „eșec”).

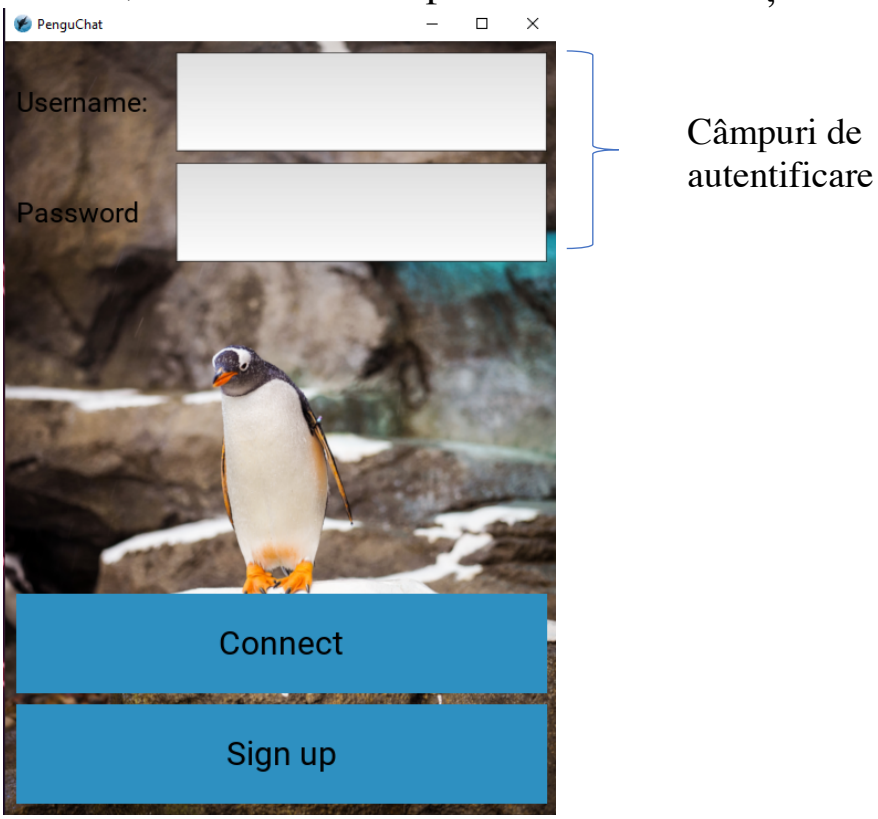
```python
if packet:
    command = loads((packet + '}').encode())
    if command['command'] == 'secure':        # serverul și-a trimis cheia publică, se generează cheia
        application.secure_server(command)    # comună pentru comunicarea cu acesta
    elif command['command'] == '200':         # credențialele trimise sunt corecte
        application.login_ok()
    elif command['command'] == '201':         # utilizatorul a fost creat cu succes.
        application.signup_ok()
    elif command['command'] == 'friend_key':  # un alt utilizator și-a trimis cheia publică
        application.got_friend_key(command)
    elif command['command'] == '406':         # numele de utilizator a fost deja folosit, se afiseaza
        application.username_taken()          # o eroare utilizatorului.
    elif command['command'] == '401':         # numele de utilizator sau parola sunt incorecte
        application.login_failed()
    elif command['command'] == 'friend_request':
        add_request(command)                  # un alt utilizator a trimis o cerere de prietenie
```

Deasupra, o privire asupra răspunsurilor serverului și cum reacționează aplicația.

Frontend-ul clientului

Interfața clientului este una intuitivă, fiecare buton având ori text ce îi descrie funcționalitatea, ori o pictogramă semnificativă. Astfel, aplicația are două mari ecrane, cel de login (și signup, în cazul în care utilizatorul nu are cont), și cel de chat, unde utilizatorii pot accesa conversațiile.

Ea se compune, în principiu, din două ecrane (momente în care fereastra aplicației își schimbă dimensiunile): fereastra de login și cea de chat, unde utilizatorii pot accesa conversațiile.

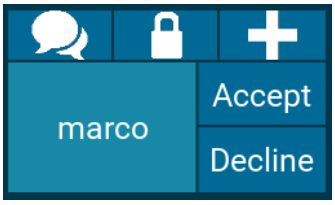


Diferitele opțiuni:
- Inbox, locul unde ajung cererile de prietenie
- Butonul de logout
- Butonul pentru cerere nouă de prietenie

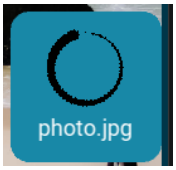
Selectarea partenerului de discuție

Butoane de trimitere și atașare fișier

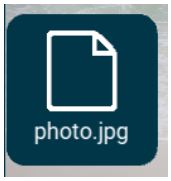
Interfața se actualizează pentru a semnala primirea unei cereri de prietenie



Un exemplu de cerere de prietenie.



Un fișier în curs de trimitere



Același fișier, ajuns la destinație

Mod de utilizare

Indiferent de platformă, instalarea celor două componente ale ecosistemului este foarte facilă. În cazul în care este necesară editarea de configurări (de exemplu adresa serverului sau anumite chei unice de securitate), se recomandă rularea programului direct cu ajutorul Python. În directorul de sursă al aplicației, se deschide o consolă și se instalează cu ajutorul modulului Pip dependențele necesare. Apoi, atât serverul, cât și clientul pot fi rulate:

```
python3 -m pip install -r requirements.txt
Server/server.py    # pornește serverul
Client/client.py    # pornește clientul
```

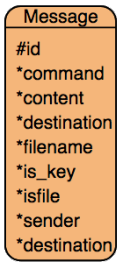
Pe partea serverului, utilizatorul nu poate încă executa comenzi, terminalul este folosit doar ca output.

Baza de date

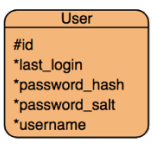
Bazele de date ale serverului și clientului au o structură simplă, ușor accesibilă.

Server:

- Conține două tabele:
  - Messagecache : acționează ca un cache, un loc unde sunt stocate mesajele în tranzit, atunci când utilizatorii nu sunt conectați.

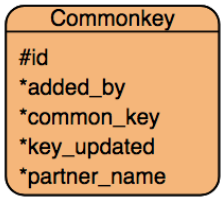


  - User: stochează utilizatorii înregistrați, hash-urile parolelor (parolele în sine nu sunt înregistrate), ultima dată când s-au conectat.

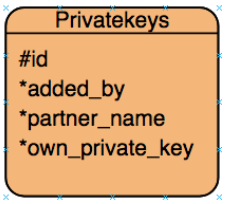


Client:

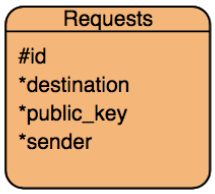
- Conține patru tabele:
  - Commonkeys: stochează cheile publice pentru cererile de prietenie trimise.



  - Privatekeys: conține cheile private, necesare decriptării.



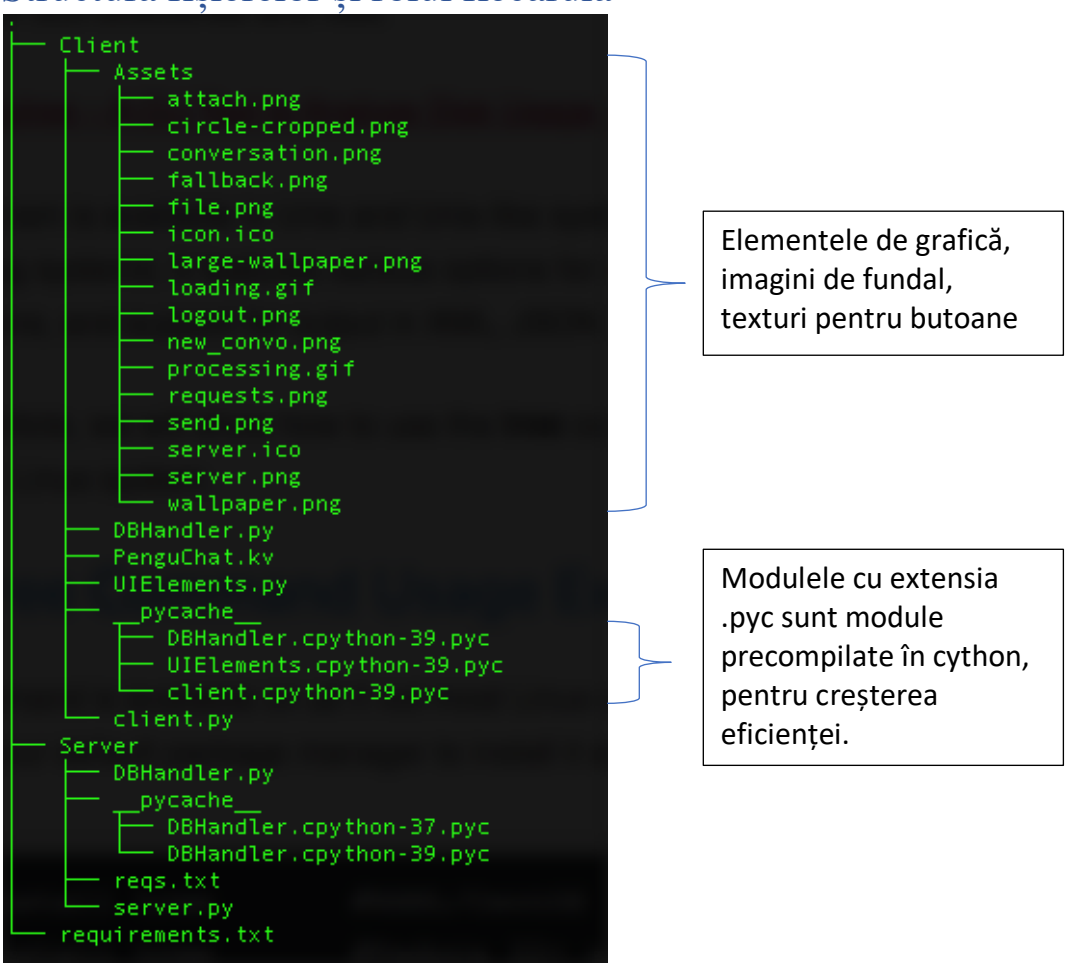
  - Requests: conține cheile publice ale cererilor de prietenie primite.



  - Messages: identic cu cel din Server, stochează mesajele.

Ca tipuri de date folosite, pentru nume s-au folosit câmpuri de tip VARCHAR, pentru momente DATETIME, pentru valori criptate (dar și pentru cheile de criptare, care sunt numere cu 64 de cifre), s-a folosit câmpul BLOB (binary large object).

Structura fișierelor și rolul fiecăruia



Fiecare parte a acestei aplicații (serverul și clientul) au câte un folder dedicat.
În folderul clientului, Assets conține toate imaginile, texturile butoanelor, animațiile, tot ceea ce este necesar elementelor grafice. DBHandler (DataBase Handler) este modulul cu ajutorul căruia este accesată baza de date. Client.py este fișierul care conține aplicația. UIElements și PenguChat.kv sunt modulele ajutătoare pentru grafică.

În folderul serverului, fișierele urmează aceeași metodă de notare ca cele din Client. Singura excepție este reqs.txt, care conține doar dependențele necesare serverului (reducând dimensiunea instalării în cazul în care se dorește doar instalarea acestuia).

Posibilități de îmbunătățire

- Pentru fișiere, poate fi adăugat un preview. Momentan este nevoie ca utilizatorul să apese pe mesaj pentru a-i vizualiza conținutul.
- Folosind TCP pentru transmiterea tuturor datelor, viteza de transfer a informației este limitată la 2-3 mbps în cazuri bune. Implementarea unui protocol FTP pentru acest lucru ar crește semnificativ viteza.
- O cunoscută limitare a sistemului de comunicare este faptul că, în cazul în care multe mesaje sunt în coada de așteptare, acestea pot ajunge la destinație fragmentate. Un mecanism de flow-control ar putea preveni acest lucru.

Bibliografie

- În cadrul realizării proiectului, am folosit ca surse de documentare următoarele.
  - https://www.comparitech.com/blog/information-security/diffie-hellman-key-exchange/ : pentru documentarea care a dus la decizia de a folosi acest algoritm de schimbare de chei.
  - https://kivy.org/doc/stable/ : documentația pentru kivy și partea grafică a programului
  - https://www.w3.org/People/Frystyk/thesis/TcpIp.html : pentru înțelegerea modului în care informația este transportată, m-am folosit de articole precum acesta.
  - https://en.wikipedia.org/wiki/Advanced_Encryption_Standard : Diffie-Hellman este doar un mod de a schimba chei de securitate, printr-un canal nesecurizat. Pentru a cripta efectiv datele, PenguChat folosește algoritmul AES.
  - https://stackoverflow.com/ : documentația universală, forum unde programatori pot schimba opinii și idei pentru implementări.