NTP Dokumentacija

UVOD

Priloženi projekat predstavlja implementaciju sistema za prijavu i registraciju u C++ (verzija 11). Na projektu su radili: Adi Hinović, Kenan Bejtić i Abdulah Fejzić. Ovaj kod omogućava korisniku da unese detalje za kreiranje naloga, koji se zatim pohranjuju u tekstualnu datoteku radi održavanja. Također osigurava sigurnost korisničke lozinke heširanjem i slanjem prije pohranjivanja. U kodu se nalaze funkcije za rad sa korisnicima u sistemu. Prije svega, korisnik može izabrati da se prijavi, promijeni lozinku, uđe u novi profil ili pristupi administratorskom izborniku ako ima administratorske privilegije. Svaka od ovih akcija implementirana je kao zasebna funkcija koja komunicira s korisnikom i vrši izmjene u korisničkom fajlu.

Ovaj kod koristi sljedeće biblioteke:

iostream - za osnovni unos/izlaz u C++
fstream - za rad sa fajlovima
sstream - za rad sa string streamovima
string - za rad sa žicama
json - za rad sa JSON objektima
OpenSSL - za kriptografiju, posebno generiranje soli i heširanje šifre
***conio*.h** (ili cstdio u nekim slučajevima) - za unos skrivenih znakova (npr. za unos lozinki)
algoritam - za određene algoritamske operacije, kao što je pretvaranje stringova u velika slova
Da biste mogli da koristite OpenSSL, potrebno je da ga instalirate na svom računaru. Instalacija zavisi od operativnog sistema. Detalji se mogu naći na https://wiki.openssl.org/index.php/Binaries . JSON za moderni C++ je biblioteka samo za zaglavlje, što znači da samo trebate preuzeti datoteku nlohmann/json.hpp sa https://github.com/nlohmann/json i uključiti je u svoj projekat.
string.h: To je biblioteka koja se koristi za manipulaciju nizovima znakova ili stringova.
iomanip: Ovo je standardna C++ biblioteka koja postavlja parametre za ulazno/izlazni tok. Ova biblioteka se ne koristi eksplicitno u datom kodu.

FUNKCIJE

Neke od glavnih funkcija u ovom kodu su:

***provjeraKorisnickogImena*()**: Ova funkcija provjerava da li dato korisničko ime već postoji u bazi podataka (u ovom slučaju, tekstualna datoteka). Koristi nlohmann/json biblioteku za čitanje JSON stringa iz datoteke i petlju kroz nju.

***provjeraZnaka*()**: Ova funkcija provjerava da li niz e-pošte sadrži simbol '@'.

***provjeraEmail*()**: Ova funkcija provjerava da li data adresa e-pošte već postoji u bazi podataka.

***generisanjeSoli*()**: Ova funkcija generiše kriptografsku sol koristeći RAND_bytes() funkciju iz OpenSSL biblioteke.

***hashSifre*()**: Ova funkcija hashira lozinku koristeći funkciju PKCS5_PBKDF2_HMAC() iz OpenSSL biblioteke. Funkcija primjenjuje hash funkciju PBKDF2 (Password-Based Key Derivation Function 2) na lozinku, kombinirajući je sa soli za povećanu sigurnost.

***unos*()**: Ova funkcija se koristi za registraciju novih korisnika. On traži sve potrebne korisničke podatke, potvrđuje ih i pohranjuje u tekstualnu datoteku.

***ispis*() i *ispisProvjera*()**: Ove funkcije se koriste za ispis podataka o korisniku. Razlika je u tome što printVerify() ispisuje heširanu lozinku i sigurnosno pitanje/odgovor, dok print() ispisuje zvjezdice umjesto lozinke iz sigurnosnih razloga.

***promjenaSifre*()**: Ova funkcija omogućava korisniku da promijeni svoju lozinku. Prvo provjerava postoji li korisničko ime, a zatim traži od korisnika da unese odgovor na sigurnosno pitanje. Ako je odgovor tačan, korisniku je dozvoljeno da unese novu lozinku.

***login*()**: Ova funkcija omogućava korisniku da se prijavi na sistem. Provjerava da li uneseno korisničko ime postoji, a zatim traži od korisnika lozinku. Ako je kod ispravan, korisnik se uspješno prijavljuje na sistem.

***brisanjeKorisnika*()**: Ova funkcija omogućava brisanje korisničkog profila iz sistema. Provjerava da li uneseno korisničko ime postoji, a zatim briše taj profil iz JSON datoteke.

***ispisKorisnika*()**: Ova funkcija ispisuje informacije o korisniku. Provjerava se da li uneseno korisničko ime postoji, a zatim se ispisuje informacija o tom korisniku.

adminMenu(): Ova funkcija omogućava pristup administratorskom meniju.

Rad sa fajlovima u JSON formatu

Korisnički podaci se pohranjuju u JSON formatu. Ovaj format je dobro strukturiran, lak za čitanje i široko prihvaćen. Podaci se čitaju iz datoteke *users.json*, a zatim se njima manipulira pomoću biblioteke JSON za moderni C++.

Hashing passwords (Haširanje šifri)

Funkcija hashSifre koristi koncept heširanja lozinke, što je važan sigurnosni korak prilikom čuvanja lozinki u aplikaciji. Heširanje transformiše lozinku u seriju bajtova, koristeći kriptografski algoritam. U ovoj funkciji, koristi se PBKDF2 (Password-Based Key Derivation Function 2) algoritam za stvaranje heša lozinke. Ovo je standardizovana metoda za stvaranje sigurnosnog ključa iz lozinke. PBKDF2 koristi HMAC (Hash-based Message Authentication Code) kao pseudoslučajanu funkciju.

```
string hashSifre(const string& sifra, unsigned char* salt, int duzSalt) {
    const int iteracija = 10000; // Broj iteracija salt-a
    const int duzinaKljuca = 32; // SHA-256 napravi 256-bitni kljuc

    unsigned char kljuc[duzinaKljuca]; // Kljuc koji ce se generisati
    PKCS5_PBKDF2_HMAC(sifra.c_str(), sifra.length(), salt, duzSalt, iteracija, EVP_sha256(), duzinaKljuca, kljuc);

    string hash;
    char hexBuf[duzinaKljuca * 2 + 1];
    // Pretvaranje u heksadecimalni zapis
    for (int i = 0; i < duzinaKljuca; i++) {
        sprintf(hexBuf + (i * 2), "%02x", kljuc[i]);
    }
    // Postavljanje null-terminatora
    hexBuf[duzinaKljuca * 2] = '\0';
    // Konverzija u string
    hash = string(hexBuf);

    return hash;
}
```

Funkcija hashSifre prima tri argumenta: referencu na string sifra, pokazivač na salt (koji je niz bajtova) i duzSalt, koji je broj bajtova u salt nizu. Salt je dodatni niz bajtova koji se koristi da se učini heširanje lozinke sigurnijim.
Postavlja se broj iteracija na 10000. Ovo je broj puta koji algoritam treba da izvrši, dodajući salt na lozinku. Više iteracija povećava vreme potrebno za heširanje, što otežava brze napade na lozinke.
Postavlja se dužina ključa na 32. Pošto se koristi SHA-256 za HMAC, dužina ključa je 32 bajta (256 bita). Deklarisanje kljuc niza koji će sadržati generisani ključ.
Ovde se poziva OpenSSL funkcija PKCS5_PBKDF2_HMAC da generiše ključ. Argumenti su šifra (i njena dužina), salt (i njegova dužina), broj iteracija, funkcija za heširanje (SHA-256), dužina ključa i mesto gde se čuva ključ.
Deklaracija stringa hash koji će držati konačan heš, i niza hexBuf koji će se koristiti za konverziju bajtova ključa u heksadecimalni format.
Konvertovanje svakog bajta ključa u heksadecimalni zapis i postavlja ga u hexBuf.
Postavljanje null-terminatora na kraj hexBuf stringa. Konvertovanje hexBuf u string i postavljanje ga u hash, zatim vraćanje hash stringa.

Unos korisnika

Funkcija unos() koristi se za registraciju novog korisnika u sustavu. Nakon prikupljanja različitih informacija od korisnika, sve te informacije se čuvaju u datoteci u JSON formatu. Ovdje je opći pregled onoga što funkcija čini:

```
void unos() {
    // Unosimo podatke
    Podaci p;
    json jsonObj;
    int duzinaSifre=0;
    unsigned char salt[8];
    int salt_len = 8;
    cout << "Unesite korisnicko ime : ";

    getline(cin, p.korisnickoIme);
    //Ponavlja se unos ako se unese vec postojece korisnicko ime
    if (provjeraKorisnickogImena(p.korisnickoIme, file))
        do {
            cout << "Korisnicko ime vec postoji, unesite ponovo: ";
            getline(cin, p.korisnickoIme);
        } while (provjeraKorisnickogImena(p.korisnickoIme, file));

    //Ponavlja se unos ako se unese vec postojeci email
    cout << "Unesite email: ";
    getline(cin, p.email);


    if (!provjeraZnaka(p.email)){
        do{
            cout <<"Vas mail nema @ znak, pokusajte ponovo: ";
            getline(cin,p.email);
        }while(!provjeraZnaka(p.email));
    }



    if (provjeraEmail(p.email, file))
        do {
            cout << "Email vec postoji, unesite ponovo (vas mail mora da sadrzi @ znak): ";
            getline(cin, p.email);
        } while (provjeraEmail(p.email, file) || !provjeraZnaka(p.email) );

    //Predstavljanje sifre preko zvjezdica
    cout << "Unesite sifru profila (minimum 8 karaktera): ";
    char c;
    while ((c = _getch()) != '\r' || duzinaSifre<8) { // \r je Enter
        if (c == '\b' && !p.sifra.empty()) { // brisanje
            p.sifra.pop_back();
            cout << "\b \b";
            duzinaSifre--;
        }
        else if (isprint(c)) {
            p.sifra += c;
            cout << '*';
            duzinaSifre++;
        }
    }
    cout << endl;
    cout << "Potvrdite sifru profila: ";
    string potvrdaSifre;
    c = '\0';
```

```
    do {
        while ((c = _getch()) != '\r') { // \r je Enter
            if (c == '\b' && !potvrdaSifre.empty()) { // brisanje
                potvrdaSifre.pop_back();
                cout << "\b \b";
            }
            else if (isprint(c)) {
                potvrdaSifre += c;
                cout << '*';
            }
        }
    } while (p.sifra != potvrdaSifre);
    cout << endl;
    cout << "Odaberite sigurnosno pitanje: " << endl;
    for (int i = 0; i < 5; i++) {
        cout << i + 1 << ". " << sigurnosnaPitanja[i] << endl;
    }
    do {
        cout << "Broj pitanja: ";
        cin >> p.sigurnosnoPitanje;
    } while (p.sigurnosnoPitanje < 1 || p.sigurnosnoPitanje > 5);
    cout << "Unesite odgovor na sigurnosno pitanje: ";
    cin.ignore();
    getline(cin, p.odgovor);

    ifstream infile(file);
    json jsonNiz;
    if (infile.good()) {
        infile >> jsonNiz;
    }

    generisanjeSoli(p.salt, salt_len);
    string hashiranaSifra = hashSifre(p.sifra, p.salt, salt_len);

    string hashiranOdgovor=hashSifre(p.odgovor,p.salt,salt_len);

    //Pisanje u jedan json objekat
    jsonObj["korisnickoIme"] = p.korisnickoIme;
    jsonObj["email"] = p.email;
    jsonObj["sifra"] = hashiranaSifra;
    jsonObj["sigurnosnoPitanje"] = sigurnosnaPitanja[p.sigurnosnoPitanje - 1];
    jsonObj["odgovor"] = hashiranOdgovor;
    jsonObj["salt"] = p.salt;

    //dodavanje json objekta na kraj json niza
    jsonNiz.push_back(jsonObj);

    //pisanje niza u fajl
    ofstream outfile(file);
    outfile << jsonNiz.dump(4) << endl;
    outfile.close();

    cout<<endl;
}
```

Podaci o korisniku (korisničko ime, e-mail, lozinka, sigurnosno pitanje i odgovor) unose se u strukturu Podaci.
Provjerava se da li već postoji korisničko ime ili e-mail u datoteci. Ako postoji, korisnik mora ponovno unijeti informacije.
Lozinka mora imati minimalno 8 znakova, a tijekom unosa lozinke korisnik vidi samo zvjezdice (*) umjesto stvarnih znakova.
Korisnik mora potvrditi svoju lozinku.

Nakon što je lozinka potvrđena, korisnik bira jedno od 5 predloženih sigurnosnih pitanja i unosi svoj odgovor na odabrano pitanje.
Sve ove informacije pohranjuju se u JSON objekt.
Pomoću funkcije generisanjeSoli(), generira se "sol" - slučajni niz bajtova koji se koristi za povećanje sigurnosti heširanja lozinke. Lozinka i odgovor na sigurnosno pitanje se zatim heširaju koristeći funkciju hashSifre().
Sve pohranjene informacije, uključujući heširanu lozinku, heširani odgovor na sigurnosno pitanje i sol, dodaju se u JSON objekt.
Ovaj JSON objekt se zatim dodaje u JSON niz, koji sadrži sve prethodno spremljene korisnike. Ako datoteka s korisnicima već postoji, prethodno spremljeni korisnici se prvo učitavaju iz te datoteke.
Naposljetku, ovaj JSON niz se sprema u datoteku.

ZAKLJUČAK

Prikazani kod pruža osnovni, ali robustan sistem registracije. Upotreba JSON-a za pohranjivanje podataka omogućava laku manipulaciju i proširenje podataka u budućnosti. Osim toga, korištenje modernih kriptografskih tehnika poput slanja i heširanja lozinki osigurava sigurnost korisničkih podataka.
Važno je napomenuti da ovaj kod pruža osnovnu funkcionalnost i da se može dalje razvijati. Na primjer, mogu se dodati dodatne validacije unosa, bolje rukovanje greškama, evidentiranje aktivnosti itd.