

Chatex

Dokumentáció



Noszlopy Gáspár Közgazdasági Technikum

Készítette:

Szép Dániel,

Kiss Levente,

(Szabó Richárd)

Tartalomjegyzék

Tartalomjegyzék

Tartalomjegyzék.....	2
1. Bevezetés	6
1.1 A témaválasztás megindoklása	6
1.2 Célkitűzés	6
1.2.1 Főbb jellemzők:	6
1.3 Fejlesztői környezet.....	7
1.3.1 Fejlesztői eszközök	7
1.4 Elvárások a feladattal kapcsolatban.....	8
1.4.1 Frontend Technológiák:	9
Flutter	9
HTML (Hypertext Markup Language).....	10
JavaScript.....	10
1.4.2 Backend Technológiák:	12
PHP (Hypertext Preprocessor)	12
MySQL	12
Dart.....	13
2. Fejlesztői dokumentáció	14

2.1 Github környezet	14
2.2 A kialakított adatszerkezet részletes bemutatása	15
2.2.1 Az adatbázis táblái	15
2.2.2 A users tábla	16
2.2.3 A messages tábla	18
2.2.4 A friends tábla	19
2.2.5 A friend_requests tábla	20
2.2.6 A chats tábla	21
2.2.7 A chat_members tábla	22
2.2.8 A message_attachments tábla	23
2.2.9 Felhasználó regisztrálása	24
2.2.10 Felhasználó bejelentkezése	27
2.2.11 Jelszó helyreállítás	29
2.2.12 Ehhez tartozik még a FORM:	31
2.3 Fájlstruktúra	37
A PHP fájlok funkciói:	37
Auth	37
Chat	37
Friends	38
Reset password	38

Settings	38
2.4 Tesztelési Dokumentáció	40
2.4.1 Regisztráció tesztek forgatókönyve:	40
2.4.2 Tesztek pontosabb ismertetése:	41
2.4.3 Bejelentkezési teszt:	42
2.4.4 Felhasználó keresés teszt:	43
3. Felhasználói dokumentáció	44
3.1 Applikáció használatának részletes ismertetése	45
3.1.1 Elfelejtett jelszó folyamatának ismertetése	49
3.1.2 Képernyők ismertetése	50
3.1.3 Barát keresése folyamat ismertetése	53
3.1.4 Barátjelölések elfogadása	54
3.1.5 Beállítások navigálása	55
3.1.6 Fiók módosítása	56
3.2 További, még nem implementált ötleteink	57
4. Összefoglalás	58
4.1 Munkamegosztás	58
4.2 Főbb feladatok:	59
• Kiss Levente:	59
• Szép Dániel:	59

• Szabó Richárd:.....	59
5. Köszönetnyilvánítás	60

1. Bevezetés

1.1 A témaválasztás megindoklása

Alkalmazásunk a Messenger jelenlegi verziójának (a készítéskor: v485.2.0.68.111) ad egy letisztult, egyszerű, fölösleges funkciók nélküli, konzisztens dizájnnal rendelkező alternatívát!

Ez az alkalmazás azoknak az embereknek szól, akik ugyanúgy nem kedvelik a Messenger felsorolt hibáit (akár még többet is) és felesleges, nem használt funkcióit!

1.2 Célkitűzés

A Chatex alkalmazás célja, hogy egy alternatívát nyújtson a híres Messenger helyett, mégpedig úgy, hogy csak az egymás közötti csevegésre fókuszál minden olyan funkció nélkül, ami nem ezt a célt szolgálja. Más szóval, a Chatex használata egy sokkal könnyebb, gyorsabb, és felhasználó barátibb környezetet nyújt, miközben ugyanúgy megtartja a játékos, személyiséget kifejező (kép küldés, fájl küldés) funkciókat.

1.2.1 Főbb jellemzők:

1. Villámgyors üzenetküldés késleltetés nélkül.
2. Biztonságos adatkezelés és titkosítás a felhasználók adatainak védelméért.
3. Modern, letisztult felület, amely könnyen kezelhető.
4. Ellenben a Messengerrel a Chatex modern kódra épül, ami egy megbízható és remekül kezelhető Framework köré épül (Flutter) így a közel jövőben kiadott verziók tökéletesen fogják fedni a kitűzött célokat!

1.3 Fejlesztői környezet

A chat applikációnk elkészítéséhez többféle fejlesztőeszközt használtunk. A kód nagy részét Visual Studio Code-ban írtuk meg, de használtunk Android Studio-t is a fejlesztéshez. Később viszont áttértünk véglegesen a VS Code-ra mivel a nagy támogatás az extensionök-re kulcs fontosságú volt a csoportmunka és egyéb dolgok kivitelezésére (Pl.: Live Share extension).

A Flutter Framework használata úgy indokolható, hogy lehetővé teszi, hogy a programot több platformon is tudjuk futtatni egyetlen egy Dart nyelven íródott kóddal! Így remekül kibővíti az alapból backend-re szánt Dart nyelvet a Flutter különböző kiegészítései (csomagjai) amit mi is használtunk bizonyos funkciók elérése érdekében, amik a pub.dev oldal segítségével nem sikerültek volna. A hab a tortán az hogy a Google által fejlesztett ez a Framework ami azt jelenti, hogy remek támogatással rendelkezik, és tökéletesen implementálható bármilyen szolgáltatás, amit a Google fejlesztett pl.: Firebase. A következő ábrán látszik hogy az Android-ra fejlesztett alkalmazáshoz milyen rendszerkövetelmények szükségesek:

Platform	Minimum OS	Támogatott Architektúrák
Android	Android 5.0+	arm32, arm64, x86_64

1.3.1 Fejlesztői eszközök

Mind VS Code-ba, mind Android Studio-nál szükség volt a fejlesztéshez egy Android-ot futtató emulátorra (lehetett volna fizikai eszközt is használni, de mi esetünkben a lokális futtatás maradt az egyetlen járható út. Emiatt is folyamodtunk ahhoz, hogy egy jól kezelhető, átlátható szerverünk legyen, ahol jól tudjuk majd megírni a backend-et! Ez volt az XAMPP, ahol a webszerveren keresztül értük el az adatbáziskezelőnk (Apache webszerver és phpMyAdmin). Innét kiindulva a backend programozásunk PHP-on keresztül ment végbe, ahol MySQL kéréseket hajtottunk végre, amit a Chatex kezdeményezett a felhasználói bemenetekre.

1.4 Elvárások a feladattal kapcsolatban

A célkitűzésben leírtakat sikeresen implementáltuk a fejlesztés során, de akadtak olyan részek is, amik nem mindig működtek úgy ahogy vártuk (legtöbbször ez a Flutter-Dart widgetek és a csomagok által használt kiegészítések működésével volt probléma). Sajnos az olyan eseteknél rögtönöznünk kellett, néhol még teljes funkciókat kellett feladnunk pl.: csoportok, videó küldés implementálása!

A jó hír viszont az, hogy ezek a kivételek csak ritkán jelentek meg és azokat leszámítva sikeresen, elvárt módon készült el minden funkció a Chatex 1.0-ba! Ezek között volt pl.: a sikeres „fél” valós idejű chat kommunikáció fénykép és fájl küldéssel, reszponzív felhasználói felület, ami alkalmazkodik különböző mobil eszközökhöz, és még sorolhatnám...

A chat applikációnk funkcióinak megfelelő működését, mind backend technológiák és frontend technológiák együttes működésével tudtuk biztosítani!

1.4.1 Frontend Technológiák:

Frontend-nek számít minden olyan technológia, amivel a felhasználó interakcióba képes lépni, tehát minden olyan elem, ami dizájn-t, szöveget, gombokat, beviteli mezőket, ikonokat épít fel az ide tartozik! Ilyen technológia a:

Flutter

A Flutter egy szoftverfejlesztő készlet (SDK), ami ráadásul a Google által készített nyílt forráskódú Frameworknek számít. Használata megkönnyíti a natív mobil- (Android, iOS), web- és asztali alkalmazások (Windows, Linux) létrehozását egyetlen kódbázisból (Dart nyelv). Fő előnye, hogy **lehetővé teszi:**

- **Platformok közötti fejlesztést:** Lehetővé teszi Android, iOS, Windows, macOS, Linux és webes alkalmazások létrehozását egyetlen kódbázissal.
- **Hot restart/Hot reload újratöltés:** A fejlesztők valós időben láthatják a kódváltozásokat anélkül, hogy a teljes alkalmazást újra kellene fordítaniuk.
- **Rugalmas felhasználói felület:** Használata testre szabható widgeteken alapszik, amelyek lehetővé teszik, hogy minden platformon natív megjelenésű és teljesítményű alkalmazásokat tervezzen.
- **Optimalizált teljesítmény:** A natív kódösszeállításnak köszönhetően az alkalmazások gyorsak és gördülékenyek minden platformon, a megfelelő eszközökkel (ha eléri persze a minimum rendszerkövetelményt, különben gyengén fog futni).
- **Több IDE támogatása:** Az alkalmazások a Flutter segítségével fejleszthetők olyan szerkesztőkben, mint az Android Studio, a Visual Studio Code, és még az IntelliJ is!

HTML (Hypertext Markup Language)



Leíró nyelv, melyet a weboldalak készítéséhez fejlesztettek ki, és mára már internetes szabvánnyá vált. Munkánkban elég kicsi mennyiségben használtunk HTML-t, de amikor kellett (jelszó helyreállító email küldése és maga az emailben elküldött jelszó helyreállító űrlap) akkor nagyon könnyen tudtuk kezelni! Mihez szokták használni:

- A HTML-lel egy webhely szerkezetét, illetve tartalmát szokás meghatározni. Létrehozhatók vele például bekezdések, címsorok, táblázatok, elhelyezhetők képek, és így tovább.
- Összes böngésző számára egy alapkövetelmény lett, hogy felismerjék (erre épül az Internet).
- Önmagában nem tud sokat alkotni, de JavaScript-el és PHP használatával egy nagyon erős leíró nyelv tud lenni, ami látszik is a jelszó helyreállítási funkciókból (és a phpk-ből)

JavaScript

A JavaScript használata lehetővé tette számunkra, hogy dinamikusan építsük fel a jelszó helyreállítási formot pl.: speciális ablakméret, tulajdonságok különféle böngészőkhöz, és a PHP-val való adatátvitel és kommunikáció! Előnyei, amik megkönnyítették a munkánkat:

- Minden modern böngésző megfelelően tudja értelmezni a JavaScript kódot a beépített motorjával, ami nagyon hatékonyá teszi a kód értelmezést. Talán mi esetünkben ez annyira nem mérvadó mert tényleg nagyon minimális a webes rész, de akár a későbbiekben nagyon hasznunkra lesz az, hogy ilyen magas szintű nyelvről beszélünk!
- Dinamikus változó deklaráció és kezelés így nem kell bajlódni azzal, hogy milyen típusokat fognak kezelni a változók értékadásakor. A mi esetünkben ez a funkció hasznos volt a képernyő méret és egyéb változók pl.: helyreállítási link értelmezése.
- Remek kommunikáció, adat értelmezés PHP-val, mert ha php változókat kezelünk, amik lehetnek elég komplexek akkor is probléma nélkül tudja őket értelmezni!

1.4.2 Backend Technológiák:

A Backend a háttérben futó folyamatokkal foglalkozik, pl.: szerveroldali programozással, űrlapon beküldött adatok feldolgozásával. Az alkalmazásunk kulcsfontosságú része, és a téma megválasztásából fakadó egyik legfontosabb kérdés, hogy milyen technológiákat kell használnunk a „fél” valós idejű chat alkalmazás megvalósításáért! Végül az XAMPP és a Flutter Frameworkből adódóan ezek a technológiák voltak kulcs fontosságúak:

PHP (Hypertext Preprocessor)



- A **PHP** egy általános szerveroldali szkriptnyelv elsősorban dinamikus weblapok készítésére (lásd jelszó helyreállító email). Az első szkriptnyelvek egyike.
- Sok különböző platform használja különféle feladatokra, ami esetünkben relációs adatbázisok (MySQL) kezelésére szolgál SQL kérésekkel, amiket maga a Chatex küld a frontend/backend adataival.
- Remekül működik csomagkezelőkkel pl.: a Composer, ami nagy hasznunkra vált a fejlesztés során!

MySQL



- A MySQL az egyik legelterjedtebb adatbázis-kezelő rendszer, aminek egyik fő oka az lehet, hogy a teljesen nyílt forráskódú LAMP (Linux–Apache–MySQL–PHP) összeállítás részeként költséghatékony és egyszerűen beállítható megoldást ad dinamikus webhelyek szolgáltatására, vagy egyéb adatbázist igénylő projektekre.
- MySQLi segítségével lehet integrálni a PHP-val (MySQL Improved csomag)
- PhpMyAdmin segítségével könnyebb az adatbáziskezelés, mivel támogatja a GUI használatát egy jól megírt lokálisan futó weboldalként, ahol minden funkcióját elérjük pl.: táblák létrehozását, módosítását, importálását és exportálását.



A Dart célkitűzése a webböngészők fő szkriptnyelvének, a JavaScriptnek a lecserélése (kitalálásakor egy backend nyelvnek szánták). Kísérletet tesznek a JavaScript problémáinak megoldására, miközben a nyelv jobb teljesítményt nyújt, mint a JS. Könnyebben lehet fejlesztőeszközöket alkalmazni a nagyobb szabású projektekhez, és egyben biztonságosabb is.

- Típusos nyelv: a változóknak meg kell adni a típusát a létrehozásuk során (Flutter Frameworkkel nem muszáj). Ez jó, mert így minden változóról pontosan tudjuk, hogy milyen értéket vár és a Dart is, hogy milyen jellegű adatként bánjon vele. Sőt dinamikusan típusos, ami azt jelenti, hogy akár saját típusokat is kreálhatunk és használhatunk a Dart programunkon belül.
- Null-biztos: ami megköveteli, hogy minden változónak legyen értéke. Ez megelőzi az olyan jellegű hibákat, hogy bármely változó véletlenül null értéket kapjon (sok null kezelés van az alkalmazásunkban). Az alkalmazásunkban például gyakori eset volt, hogy amikor egy értéket vettünk át az adatbázisból php-n keresztül akkor null-ként értelmezte, ami exceptionöket adott (pl.: profilkép értelmezése).
- Rengeteg különböző könyvtárral (library), beépített típussal rendelkezik. (pub.dev oldalon találhatók)
- A kódot kétféleképpen futtathatjuk: natív platformon, (Android) ami a mobil és asztali eszközöket célozza meg. Web platformon, ami azt jelenti, hogy a Dart nyelv segítségével webalkalmazásokat is készíthetünk, melynek során a Dart a JavaScript-re fordítódik át.

2. Fejlesztői dokumentáció

2.1 Github környezet

A vizsgaremekhez kiírt feltételekhez és munkánk megkönnyítéséhez alakítottunk ki egy Github környezetet applikációnkhoz. egy közös email cím létrehozásával (chatexfejlesztok@gmail.com ami az alkalmazásban is szerepel) végeztük el a környezet elkészítését.

A közös emaillel csináltunk egy repository-t **Messengeres-vizsgaremek** néven és ezután hozzáadtuk egymás fiókjait, hogy mindegyikőnk fejlesztése könnyen nyomkövethető és zökkenőmentes legyen.

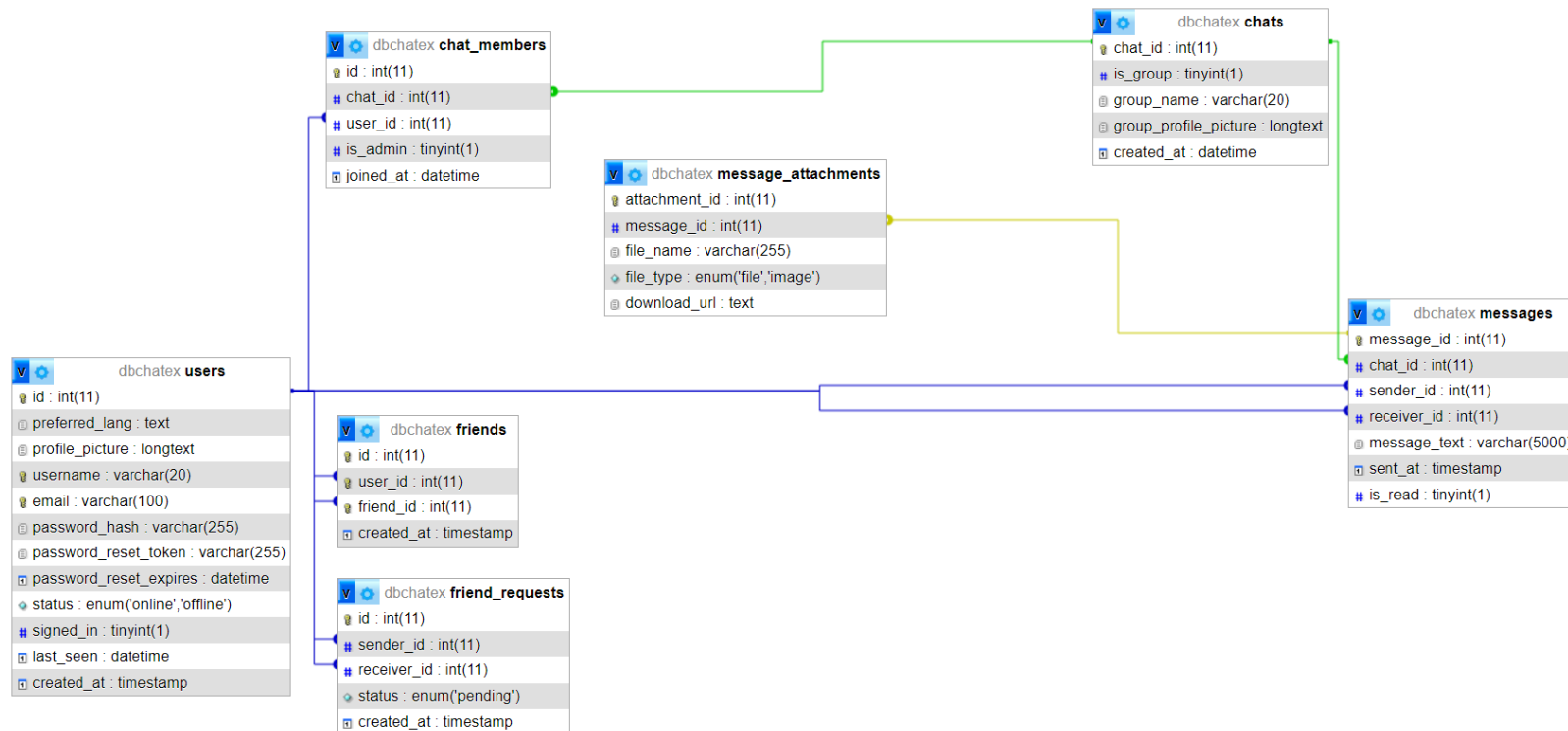
Az elkészült vizsgaremek tartalmazza az adatbázist, programkódot, dokumentációt és az előadást, amit a következő Github repositoryban érhető el: [Chatex vizsgaremek](#). itt található az összes szükséges anyag a projekt teljes megértéséhez.

(Lehet, hogy a leadott vizsagremek más repositoryban lesz, ezt még nem tudjuk, de a szerkezete attól még nem változott)

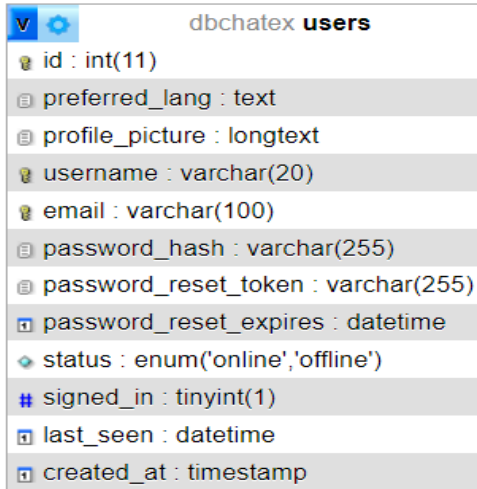
2.2 A kialakított adatszerkezet részletes bemutatása

Chat applikációnk táblái strukturáltak, szervezettek és hatékonyan, és a felhasználók biztonságukra figyelve kezelik, tárolják az adatokat, de hogyan is néznek ki ezek a táblák, hát így:

2.2.1 Az adatbázis táblái



2.2.2 A users tábla



id	: int(11)
preferred_lang	: text
profile_picture	: longtext
username	: varchar(20)
email	: varchar(100)
password_hash	: varchar(255)
password_reset_token	: varchar(255)
password_reset_expires	: datetime
status	: enum('online', 'offline')
signed_in	: tinyint(1)
last_seen	: datetime
created_at	: timestamp

Így néz ki a users táblánk, aminek a szerkezetét így lehetne jellemezni: **1. mező:** id (fő azonosító): Ez a mező az adott felhasználó azonosítója. Ez az elsődleges kulcs, szóval egyedi minden egyes felhasználónak ezért segít a megkülönböztetésben. Ehhez az azonosítóhoz kapcsolódik az összes többi azonosító (idegen kulcs), **mivel ez a fő azonosító.** **2. mező:** preferred_lang (preferált nyelv): Ez a mező a felhasználó által kiválasztott nyelvet határozza meg, Egyelőre még csak magyar és angol nyelv elérhető, de ez később bővülni fog! Ez alapján határozzuk meg hogy a szöveges tartalom (állandó) hogyan jelenjen meg a felhasználó számára! **3. mező:** profile_picture (profilkép): Ez a mező az adott felhasználó profilképe, a felhasználó állítja be egy JPEG, JPG, PNG, SVG formátumú képpel (más formátum nem ajánlott!), ha nem

állít be profilképet, az applikáció az alapértelmezettet adja a felhasználónak. **4. mező:** username (felhasználónév): Ez a mező tartalmazza az adott felhasználó által megadott felhasználónevet, ezzel a névvel tudnak a többi felhasználók rákeresni erre a személyre, ez a név jelenik meg a chatekben is! (feltétele, hogy 3-20 karakterig terjedhet csak!) **5. mező:** email: Ez a mező tartalmazza az adott felhasználó által megadott email címet, erre az email címre fogja megkapni a felhasználó az összes üzenetet a rendszertől (jelenleg csak az új jelszó kérés esetében). **6. mező:** password_hash (jelszó): Ez a mező tartalmazza az adott felhasználó által megadott jelszót kódolva (biztonsági okok miatt), ezzel tud csak a felhasználó bejelentkezni a fiókjába (nem kódolt állapotot kérünk, hanem amit megadott!), és ha elfelejti, új jelszót tud kérni a főképernyőn (a Chatex nem fogja tudni megmondani!). **7. mező:** password_reset_token (jelszó helyreállítása): Ez a mező tartalmazza az adott felhasználónak éppen függőbe lévő jelszó helyreállítási kérését, ha nincs jelenleg helyreállító tokenje, akkor NULL, értéket adunk neki. Az ID-n keresztül a token alapján tudja a jelszó helyreállító kinek kell új jelszót beállítani. **8. mező:** password_reset_expires (jelszó helyreállításának élettartama): Ez a mező tartalmazza az adott felhasználónak a jelszó helyreállításának élettartamát (15 perc), ha a felhasználó nem végezte el a „dolgát” ebben az időtartamban, a jelszó helyreállításának tokenje automatikusan törölve lesz az adatbázisból és új kérést kell nyitnia a felhasználónak.

9.mező: status (státusz) mező a felhasználó online/offline állapotát tárolja el, amit manuálisan tud váltani a belépés utáni sidebar-on. A manuális beállítás azt teszi lehetővé, hogy a nem kívánatos beszélgetéseket eltudja kerülni az ember, hogy offline állapotba rakja magát, és az addig úgy is marad míg át nem váltja (még bejelentkezéskor is megmarad!). **10. mező:** signed_in (bejelentkezve) mező a felhasználó bejelentkezésekor frissíti magát 1-re (bináris igen) és amikor kijelentkezik akkor pedig 0-ra (bináris nem). Ez azért hasznos mert csak akkor jelenítünk meg egy illetőt online állapotúnak, ha a signed_in mező 1 és a beállított státusz online állapoton van, különben, ha offline akkor semmilyen esetben jelenítjük meg onlineként! **11. mező:** last_seen (utoljára látva) mező, aminek a feladata nagyon is egyszerű, mégpedig az, hogy kijelentkezéskor egy függvény segítségével beállítjuk, hogy a jelenlegi dátumot mentse el (év-hónap-nap óra-perc-másodperc formátumban). **12. mező:** created_at (fiók készült ekkor): Ez a mező tartalmazza azt, hogy az adott felhasználó pontosan mikor regisztrálta fiókját. Ezt az értéket annyira nem kezeljük, de későbbiekben statisztikát lehet belőle készíteni!

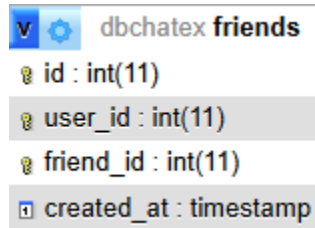
2.2.3 A messages tábla



message_id	: int(11)
# chat_id	: int(11)
# sender_id	: int(11)
# receiver_id	: int(11)
message_text	: varchar(5000)
sent_at	: timestamp
# is_read	: tinyint(1)

Így néz ki a messages táblánk, aminek a szerkezetét így lehetne jellemezni: **1. mező:** message_id (üzenet azonosító): Ez a mező tartalmazza minden egyes üzenetnek az azonosítóját, hogy miután a felhasználó kilép az applikációból, ne tűnjenek el az üzenetei mikor visszalép, ez a táblának az elsődleges kulcsa, a mező INT típusú és maximum 11 karakter hosszúságú lehet. **2. mező:** chat_id (chat azonosító). Ez a mező azonosítja hogy az üzenet melyik chat-hez tartozik úgy hogy ez egy idegen kulcshoz van kapcsolva. **3. mező:** sender_id (küldő azonosító). Ez a mező tartalmazza az üzenetet küldő felhasználónak az azonosítóját, ezzel biztosítjuk, hogy az üzenet biztos hogy a chatben lévő kettő személy között lesz elmentve, a mező INT típusú és maximum 11 karaktert fogad be (idegen kulcs a users táblából). **4. mező:** receiver_id (fogadó azonosító): Ez a mező tartalmazza az üzenetet fogadó felhasználónak az azonosítóját, ezzel biztosítjuk, hogy az üzenet biztos a kettő személy között lesz elmentve, a mező INT típusú és maximum 11 karaktert fogad be (szintén idegen kulcs a users táblára). **5. mező:** message_text (üzenet tartalma): Ez a mező tartalmazza az üzenetek tartalmát (szöveg, ha nincs akkor NULL), amihez hozzá van adva az üzenet azonosító, ezzel a kettővel nem fog elveszni se maga az üzenet, se az üzenet tartalma, a mező VARCHAR típusú és maximum 5000 karaktert fogad be. **6. mező:** sent_at (elküldve ekkor): Ez a mező tartalmazza azt az időpontot, amikor az küldő felhasználó által elküldött üzenetet a vevő felhasználó megkapja, a mező TIMESTAMP típusú. **7. mező:** is_read (láttam): Ez a mező tartalmazza azt, hogy az elküldött üzenetet a fogadó felhasználó látta-e vagy sem, a mező TINYINT típusú és csak egyetlen egy karaktert fogad be (lényegében boolean).

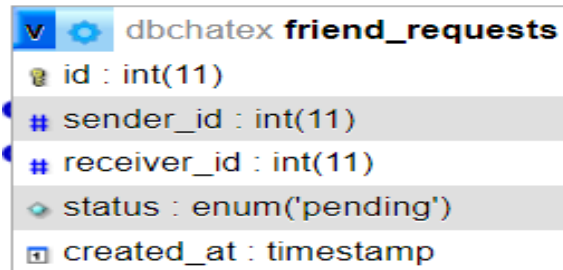
2.2.4 A friends tábla



dbchatex friends	
🔑	id : int(11)
🔑	user_id : int(11)
🔑	friend_id : int(11)
📅	created_at : timestamp

Így néz ki a friends táblánk, aminek a szerkezetét így lehetne jellemezni: **1. mező:** id (azonosító): Ez a mező tartalmazza a barátlistán lévő felhasználók azonosítóját, ez az azonosító az elsődleges kulcs és INT típusú, maximum 11 karaktert fogad be (idegen kulcs a users táblára). **2. mező:** user_id (felhasználó azonosító): Ez a mező tartalmazza a felhasználónak az azonosítóját (idegen kulcs a users táblára), ez az egyik rész a barát azonosító megalkotásához, a mező INT típusú és maximum 11 karaktert fogad be. **3. mező:** friend_id (barát azonosító): Ezzel lesz teljes az id alapján megalkotott barát kapcsolat, amit a program használata közben duplán veszünk fel. A mező INT típusú és maximum 11 karaktert fogad be. **4. mező:** created_at (barátlistához hozzáadva ekkor): Ez a mező tartalmazza azt az időpontot, amikor a két felhasználó hozzá adta egymást a barátlistájukhoz, a mező TIMESTAMP típusú (későbbiekben lehet ebből statisztikát készíteni).

2.2.5 A friend_requests tábla

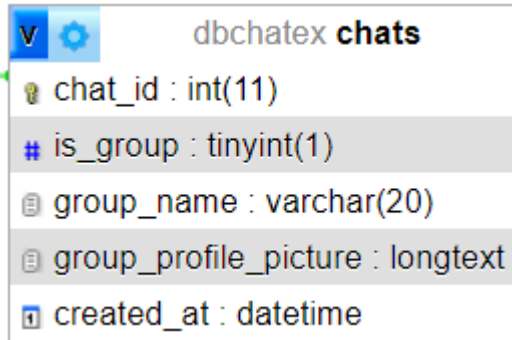


dbchatex friend_requests	
id	int(11)
sender_id	int(11)
receiver_id	int(11)
status	enum('pending')
created_at	timestamp

Így néz ki a friend_requests táblánk, aminek a szerkezetét így lehetne jellemezni: **1. mező:** id (azonosító): Ez a mező tartalmazza maga a barátkérelemnek az azonosítóját, hogy az adatbázis feljegyezze a két felhasználó között lehetséges több barátkérélmeket is, a mező INT típusú és maximum 11 karaktert fogad be. **2. mező:** sender_id (barátkérélmek küldő azonosító): Ez a mező tartalmazza a barátkérélmek elküldő felhasználó az azonosítóját (idegen kulcs a users táblára), a mező INT típusú és maximum 11 karaktert

fogad be. **3. mező:** receiver_id (barátkérélmek kapó azonosító): Ez a mező tartalmazza annak a felhasználó az azonosítóját (idegen kulcs a users táblára), aki a barátkérélmek kapta egy másik felhasználótól, aki még nincs a barátlistáján, a mező INT típusú és maximum 11 karaktert fogad be. **4. mező:** status (barátkérelem állapota): Ez a mező tartalmazza a barátkérelem állapotát, egyből az elküldés után a „pending” (függőben) attribútumot kapja, ha a vevő felhasználó elutasítja akkor szimplán töröljük a kérést (későbbiekben felhasználóknak is lesz visszajelzés), ha elfogadja akkor meg szintén töröljük a rekordot és felvesszünk egy új rekordot a friends táblába a vevő és a fogadó adataival (duplán felvesszük a barát kapcsolatot), a mező ENUM típusú, és csak „pending”-el térhet vissza. **5. mező:** created_at (barátkérelem elküldésének időpontja): Ez a mező tartalmazza azt az időpontot, amikor a barátkérélmek elküldték a felhasználóknak (nem csak statisztikára lesz jó, hanem hogy csökkenő sorrendben jelenítsük meg a kéréseket a Chatex-ben).

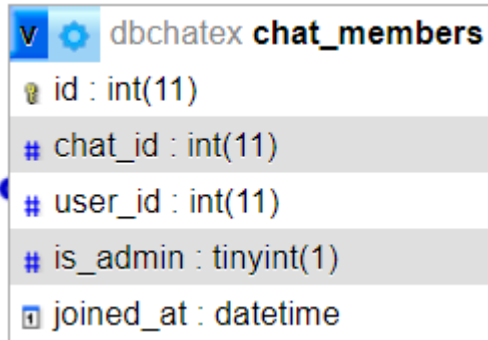
2.2.6 A chats tábla



	dbchatex	chats
🔑	chat_id	int(11)
#	is_group	tinyint(1)
📄	group_name	varchar(20)
📄	group_profile_picture	longtext
🕒	created_at	datetime

Így néz ki a chats táblánk, aminek a szerkezetét így lehetne jellemezni: **1. mező:** chat_id (chatet azonosító id): Ez az elsődleges kulcs, ami azért felel, hogy minden chat-re egyértelműen tudjunk hivatkozni. **2. mező:** is_group (csoport e) mező, ami azt azonosítja, hogy a chat egy csoport vagy sem (jelenleg minden felvételkor hamis-ként tér vissza, mert nincsen a csoport funkció implementálva. **3. mező:** group_name (csoport név) ez a mező jelenleg nincsen használatban mivel nem sikerült a csoportok funkciót implementálni így minden chat készítéskor üres. **4. mező:** group_profile_picture (csoportkép) ez a mező sincs használatban, de a csoportnak lehetett volna megadni profilképet. **5. mező:** created_at (mikor lett létrehozva a chat) ez a mező minden chat létrehozásakor eltárolja mikor lett létrehozva a megadott formátummal (jelenleg statisztikára jó).

2.2.7 A chat_members tábla

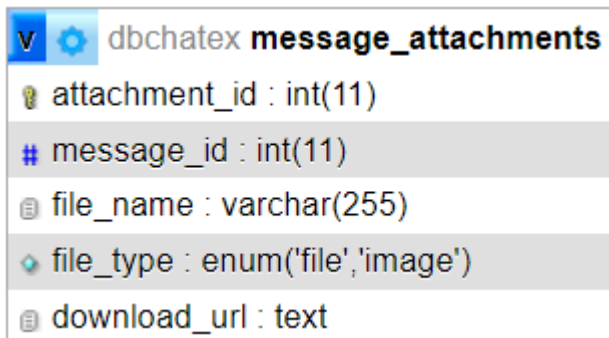


dbchatex	chat_members
id	: int(11)
# chat_id	: int(11)
# user_id	: int(11)
# is_admin	: tinyint(1)
joined_at	: datetime

Így néz ki a chat_members táblánk, aminek a szerkezetét így lehetne jellemezni: **1. mező:** id mező: egyértelműen azonosítja, hogy egy chaten belül hány user van felvéve, illetve egyértelműen azonosítja a rekordot. **2. mező:** chat_id (chat azonosítója) a chats táblára hivatkozik (egy idegen kulcsként). **3. mező:** user_id (felhasználó azonosítója) ez a mező azonosítja, hogy melyik felhasználó tagja a chatnek (idegen kulcs a users táblából). **4. mező:** is_admin (admin e felhasználó) csak csoportoknál lenne fontos, ami nem sikerült (nem használatos a tagok felvételekor). **5. mező:** joined_at (felhasználó mikor csatlakozott a

chathez) mező jelenleg semmi mérvadót nem csinál statisztikára lehetne használni.

2.2.8 A message_attachments tábla



dbchatex	message_attachments
🔑	attachment_id : int(11)
#	message_id : int(11)
📄	file_name : varchar(255)
🔹	file_type : enum('file','image')
📄	download_url : text

Így néz ki a message_attachments táblánk, aminek a szerkezetét így lehetne jellemezni: **1. mező:** attachment_id (csatolmány id) ez a mező azonosítja az üzenetek mellé küldött csatolmányt, ami lehet kép és file. **2. mező:** message_id (üzenet id) egy idegen kulcs, ami a message táblára hivatkozik azon belül is az elküldött üzenet id-ára (ha nincs szöveg akkor NULL, de attól még lehet csatolmány). **3. mező:** file_name (fájlnév) adja meg a fájlnévet, amit lokálisan a htdocs/ChatexProject/uploads/files vagy /media-ba menti és onnét is lehet letölteni a telefonról. **4. mező:** file_type (fájl típus) ami egyszerűen azonosítja a feltöltött csatolmányt kép vagy fájl típusra, ez alapján dönti el, hogy lokálisan melyik mappába tárolja (media vagy files). **5. mező:** download_url (letöltési url) ami egy text típusú mező a nagysága miatt, ez tartalmazza a localhostal kezdődő uri-t, ami a csatolmány elérési útvonalát adja meg.

2.2.9 Felhasználó regisztrálása

```
<?php
header("Content-Type: application/json; charset=UTF-8");
header("Access-Control-Allow-Origin: *");
header("Access-Control-Allow-Methods: POST");
header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With");

require_once __DIR__ . "../db.php";

//formátum stb. ellenőrzés mert ha pl.: szimuláljuk az adatokat tudjuk hol a hiba, jelszó visszaállító email küldése!

function normalizeEmail($email)
{
    // Gmail esetén: a pontokat és a + utáni részt eltávolítjuk
    if (strpos($email, '@gmail.com') !== false) {
        $emailParts = explode('@', $email);
        $localPart = str_replace('.', '', $emailParts[0]); // Pontok eltávolítása
        $localPart = explode('+', $localPart)[0]; // + utáni rész eltávolítása
        return $localPart . '@gmail.com';
    }
    return $email; // Más e-mail szolgáltatóknál nincs változás
}

$userData = json_decode(file_get_contents("php://input"), true);

// Adatok kinyerése
$username = trim($userData['username']);
$email = normalizeEmail(trim($userData['email']));
$password = trim($userData['password']);
$password_hash = password_hash($password, PASSWORD_DEFAULT);
$language = $userData['language'];

// Ellenőrizzük, hogy az e-mail formátuma helyes-e
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    http_response_code(400); // Hibás kérés
    echo json_encode(["message" => "Érvénytelen email cím!"]);
    exit();
}

// Ellenőrizzük, hogy az email cím már létezik-e az adatbázisban
$checkStmt = $conn->prepare("SELECT id FROM users WHERE email = ?");
$checkStmt->bind_param("s", $email);
$checkStmt->execute();
$checkStmt->store_result();

if ($checkStmt->num_rows > 0) {
    http_response_code(409); // Konfliktus
    echo json_encode(["message" => "Ezzel az emaillel már létezik felhasználó!"]);
    exit();
}
```

Ez a PHP szkript egy REST API végpontot biztosít a felhasználók regisztrációjához (ezt a fájlt hívja meg a Dart). Az API JSON formátumban fogadja a bemenetet, ellenőrzi az adatokat, normalizálja a Gmail-es e-mail címeket, és elmenti az adatokat egy MySQL adatbázisba.


```

// Felhasználó beszúrása az adatbázisba stmt = STATEMENT (állítás)
$stmt = $conn->prepare("INSERT INTO users (preferred_lang, username, email, password_hash, created_at) VALUES (?, ?, ?, ?, NOW())");
if ($stmt === false) {
    http_response_code(500); // Belső szerverhiba
    echo json_encode(["message" => "SQL előkészítési hiba."]);
    exit();
}

$stmt->bind_param("ssss", $language, $username, $email, $password_hash);

// Lekérdezés végrehajtása és válasz küldése
if ($stmt->execute()) {
    http_response_code(201); // Erőforrás létrehozva
    echo json_encode([
        "message" => "Sikeres regisztráció!",
        "username" => $username,
        "email" => $email,
        "password_hash" => $password_hash,
        "preferred_lang" => $language
    ]);
} else {
    http_response_code(500); // Belső szerverhiba
    echo json_encode(["message" => "Sikertelen regisztráció!"]);
}

$stmt->close();
$conn->close();

```

A header beállításoknál először beállítjuk, hogy a válasz JSON formátumban érkezzen, bárholnan engedélyezett a hozzáférés, Az API kizárólag **POST** módszerrel hívható és az API-hoz szükséges fejléceket engedélyezi.

A normalizeEmail módszer használatával a különböző használatú emaileket egy formátumra teszi, így nem kell szenvedni az összes lehetséges email lecsekkolásával.

Átveszi az adatokat és be is olvassa a tárolójukba:

Utána ellenőrzi az email formátumát, és ha helyes, akkor tovább megy és ellenőrzi, hogy létezik-e már az email az adatbázisban, utána beszúrja az adatbázisba, ha minden jól megy akkor kész a regisztráció.

Ezt a kódot az Auth.dart-ban így használjuk fel:

```
try {
  final Uri registrationUrl = Uri.parse(
    'http://10.0.2.2/ChatexProject/chatex_phps/auth/register.php');
  final response = await http.post(
    registrationUrl,
    body: jsonEncode(<String, String>{
      'username': username.text.trim(),
      'email': email.text.trim(),
      'password': password.text.trim(),
      'language': language,
    })),
  );

  final responseData = jsonDecode(response.body);

  if (responseData["message"] == "Sikeres regisztráció!") {
    final username = responseData['username'];
    final email = responseData['email'];
    final passwordHash = responseData['password_hash'];
    final preferredlang = responseData['preferred_lang'];

    await Preferences.setUsername(username);
    await Preferences.setEmail(email);
    await Preferences.setPasswordHash(passwordHash);
    await Preferences.setPreferredLanguage(preferredlang);
    ToastMessages.showToastMessages(
      language == "Magyar"
        ? "Sikeres regisztráció!"
        : "Successful registration!",
      0.1,
      Colors.green,
      Icons.check,
      Colors.black,
      const Duration(seconds: 2),
      context,
    );
    await Future.delayed(const Duration(seconds: 2));
    Navigator.pushReplacement(
      context,
      MaterialPageRoute(
        //BuildContext volt a típusa
        builder: (context) => const LoginUI(),
      ), // MaterialPageRoute
    );
  }
};
```

2.2.10 Felhasználó bejelentkezése

```
<?php
header("Content-Type: application/json; charset=UTF-8");
header("Access-Control-Allow-Origin: *");
header("Access-Control-Allow-Methods: POST");
header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With");

require_once __DIR__ . '/../db.php';
require_once __DIR__ . '/../vendor/autoload.php';

use Firebase\JWT\JWT;

$userData = json_decode(file_get_contents("php://input"), true);

$email = trim($userData['email']);
$password = trim($userData['password']);

// Lekérdezzük a felhasználót az email alapján
$stmt = $conn->prepare("SELECT id, preferred_lang, profile_picture, username, email, password_hash FROM users WHERE email = ?");
$stmt->bind_param("s", $email);
$stmt->execute();

$result = $stmt->get_result();

$user = $result->fetch_assoc();

if (!$user || !password_verify($password, $user["password_hash"])) {
    http_response_code(401); // 401 = Unauthorized
    echo json_encode(["message" => "Hibás email vagy jelszó!"]);
    exit();
}
```

Ez a PHP szkript egy bejelentkezési API végpontot valósít meg JSON Web Token (JWT) használatával. A felhasználó e-mail címe és jelszava alapján hitelesíti a bejelentkezést, és sikeres azonosítás esetén egy JWT token generál.

```

// JWT token létrehozása - hitelesítés amivel pl: nem kell újra meg újra bejelentkezni
$issued_at = time();
$expiration_time = $issued_at + (60 * 60 * 24); // 24 óra
$payload = [
    "iat" => $issued_at,
    "exp" => $expiration_time,
    "sub" => $user["id"], // Felhasználó ID-ja az adatbázisból
    "username" => $user["username"], // Adatbázisból lekért username
    "email" => $user["email"],
    "preferred_lang" => $user["preferred_lang"]
];

$secret_key = "chatex";

$jwt = JWT::encode($payload, $secret_key, 'HS256');

http_response_code(200); // 200 = OK
echo json_encode([
    "message" => "Sikeres bejelentkezés",
    "success" => true,
    "token" => $jwt,
    "id" => $user["id"],
    "username" => $user["username"],
    "preferred_lang" => $user["preferred_lang"],
    "email" => $user["email"],
    "password_hash" => $user["password_hash"],
    "profile_picture" => trim($user["profile_picture"], "\\")
]);

//TODO: használni a token!!!

$stmt->close();
$conn->close();

```

Adatok fogadása: A bejövő JSON kérés tartalmát a **php://input** segítségével olvassuk be és dekódoljuk.

Adatok ellenőrzése: Az e-mail cím és a jelszó tisztítása (trim()) után az adatbázisban ellenőrizzük a felhasználót.

Felhasználó keresése: SQL lekérdezéssel próbáljuk megkeresni az e-mail címhez tartozó felhasználót.

Jelszó ellenőrzése: Ha a felhasználó létezik, a password_verify() függvénnyel ellenőrizzük a jelszót.

JWT token generálása: Sikeres bejelentkezés esetén egy 24 órán keresztül érvényes JWT tokenet hozunk létre.

Válasz küldése: A válasz tartalmazza a JWT tokenet és a felhasználó adatait.

Hibakezelés: Hibás bejelentkezési adatok esetén 401-es státuszkódot küldünk vissza.

2.2.11 Jelszó helyreállítás

```
<?php
header("Content-Type: application/json");
header("Access-Control-Allow-Origin: *");
header("Access-Control-Allow-Methods: POST");
header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With");

use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\Exception;

require_once __DIR__ . '/../db.php'; // Adatbázis kapcsolat
require_once __DIR__ . '/../vendor/autoload.php'; // PHPMailer betöltése

// JSON adatok beolvasása
$data = json_decode(file_get_contents("php://input"), true);

if (!isset($data["email"])) {
    echo json_encode(["success" => false, "message" => "Email megadása kötelező."]);
    exit();
}

$email = $data["email"];

// Megnézzük, hogy létezik-e az e-mail az adatbázisban
$stmt = $conn->prepare("SELECT id FROM users WHERE email = ?");
$stmt->bind_param("s", $email);
$stmt->execute();
$result = $stmt->get_result();

if ($result->num_rows > 0) {
    $row = $result->fetch_assoc();
    $userId = $row["id"];

    // Token generálása és lejáratí idő beállítása
    $token = bin2hex(random_bytes(20));
    $expires = date("Y-m-d H:i:s", strtotime("+15 minutes"));

    // Token mentése az adatbázisba
    $stmt = $conn->prepare("UPDATE users SET password_reset_token = ?, password_reset_expires = ? WHERE id = ?");
    $stmt->bind_param("ssi", $token, $expires, $userId);
    $stmt->execute();

    // Jelszó visszaállító link
    $resetLink = "http://localhost/ChatexProject/chatex_phps/reset_password/open_reset_window.php?token=$token";

    // **📧 PHPMailer konfigurálása és email küldés**
    $mail = new PHPMailer(true);
```

```

try {
    // SMTP beállítások
    $mail->isSMTP();
    $mail->Host = 'smtp.gmail.com'; // SMTP szerver (pl. Gmail)
    $mail->SMTPAuth = true;
    $mail->Username = 'chatexfejlesztok@gmail.com'; // SMTP e-mail címed
    $mail->Password = 'uvatzwfcrjlcujrs'; // SMTP jelszó vagy App Password
    $mail->SMTPSecure = PHPMailer::ENCRYPTION_STARTTLS;
    $mail->Port = 587;
    $mail->CharSet = 'UTF-8';
    $mail->Encoding = 'base64';

    // Feladó és címzett
    $mail->setFrom('chatexfejlesztok@gmail.com', 'Chatex support (no reply)');
    $mail->addAddress($email);

    // E-mail tartalma
    $mail->isHTML(true);
    $mail->Subject = "Jelszó visszaállítás";
    $mail->Body = "<h1>Kattints az alábbi linkre a jelszó visszaállításához:</h1>
    <p><a href='$resetLink' target='_blank'>$resetLink</a></p>
    <h2>Ez a link 15 percig érvényes.</h2>
    <p>Chatex</p>";

    // E-mail küldés
    if ($mail->send()) {
        echo json_encode(["success" => true, "message" => "Helyreállító e-mail elküldve."]);
    } else {
        echo json_encode(["success" => false, "message" => "E-mail küldése sikertelen."]);
    }
} catch (Exception $e) {
    echo json_encode(["success" => false, "message" => "E-mail hiba: {$mail->ErrorInfo}"]);
}
} else {
    echo json_encode(["success" => false, "message" => "Nincs ilyen email című felhasználó!"]);
}

$stmt->close();
$conn->close();

```

Ez egy PHP alapú rendszer, amely segít a felhasználóknak visszaállítani az elfelejtett jelszavunkat. Ha valaki elfelejti a jelszavát, megadhatja az e-mail címét, és kap egy linket, amelyen keresztül új jelszót állíthat be.

A szkript fogad egy HTTP POST kérést, amely tartalmazza a felhasználó e-mail címét. Ellenőrzi, hogy az e-mail szerepel-e az adatbázisban. Ha igen, akkor generál egy véletlenszerű token-t, amelyet eltárol az adatbázisban egy lejáratí idővel együtt. Létrehoz egy jelszó visszaállító linket, amely ezt a token használja. Az e-mailt elküldi a felhasználónak PHPMailer segítségével. Ha az e-mail küldése sikeres, visszatér egy JSON válasszal.

2.2.12 Ehhez tartozik még a FORM:

```
<?php
require_once __DIR__ . "../db.php";

if ($_SERVER["REQUEST_METHOD"] === "POST") {
    header("Content-Type: application/json");
    header("Access-Control-Allow-Origin: *");
    header("Access-Control-Allow-Methods: POST");
    header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With");

    $token = $_POST["token"] ?? '';
    $newPassword = $_POST["new_password"] ?? '';

    if (!$token || !$newPassword) {
        echo json_encode(["success" => false, "message" => "Érvénytelen kérés."]);
        exit;
    }

    $stmt = $conn->prepare("SELECT id, email FROM users WHERE password_reset_token = ? AND password_reset_expires > NOW()");
    $stmt->bind_param("s", $token);
    $stmt->execute();
    $result = $stmt->get_result();
```

```

    if ($result->num_rows > 0) {
        $row = $result->fetch_assoc();
        $userId = $row["id"];
        $userEmail = $row["email"];

        $hashedPassword = password_hash($newPassword, PASSWORD_BCRYPT);
        $stmt = $conn->prepare("UPDATE users SET password_hash = ?, password_reset_token = NULL, password_reset_expires = NULL WHERE id = ?");
        $stmt->bind_param("si", $hashedPassword, $userId);
        $stmt->execute();

        echo json_encode(["success" => true, "message" => "Jelszó sikeresen frissítve!"]);
    } else {
        echo json_encode(["success" => false, "message" => "A jelszó helyreállító email lejárt!"]); //Érvénytelen vagy lejárt token.
    }

    $stmt->close();
    $conn->close();
    exit;
}

// Ha nem POST kérés, hanem GET (az oldal megjelenítése)
$token = $_GET["token"] ?? '';
if (!$token) {
    die("Érvénytelen token.");
}

// Lekérjük az email címet a token alapján
$stmt = $conn->prepare("SELECT email FROM users WHERE password_reset_token = ? AND password_reset_expires > NOW()");
$stmt->bind_param("s", $token);
$stmt->execute();
$result = $stmt->get_result();
$userEmail = "";

if ($result->num_rows > 0) {
    $row = $result->fetch_assoc();
    $userEmail = $row["email"];
}

$stmt->close();
$conn->close();

```


Kétféle módon működik:

POST kérés: Ha a felhasználó már kapott egy visszaállító e-mailt és beírja az új jelszavát.

GET kérés: Ha valaki rákattint a jelszó-visszaállító linkre, megnyílik az oldal, ahol beírhatja az új jelszót.

A rendszer ellenőrzi, hogy a felhasználó valóban jogosult-e a jelszó megváltoztatására.

1. Ha a felhasználó elküldi az új jelszót (POST kérés)

A program ellenőrzi, hogy mindkét adat meg van-e adva.

Megkeresi az adatbázisban, hogy a token érvényes-e.

Ha a token rendben van:

- Az új jelszót titkosítva menti el.
- A token törlésre kerül, hogy ne lehessen újra felhasználni.
- Sikeres választ küld vissza.

Ha a token hibás vagy lejárt, hibaüzenetet küld vissza.

2. Ha a felhasználó megnyitja a jelszó-visszaállító oldalt (GET kérés)

A program ellenőrzi, hogy van-e token az URL-ben.

Ha igen, megkeresi az adatbázisban az e-mail címhez tartozó adatokat.

Ha a token érvényes:

- Megnyílik egy oldal, ahol a felhasználó beírhatja az új jelszavát.

Ha a token lejárt vagy hibás:

- Egy hibaüzenetes oldal jelenik meg.

A weboldal tartalmaz:

- Két jelszó mezőt (az új jelszó és annak megerősítése).
- Ellenőrzést, hogy a jelszó megfelelő-e:
 - Legalább 8 karakter hosszú legyen.
 - Legyen benne kis- és nagybetű.
 - Tartalmazzon legalább egy számot.
- Hibaüzeneteket, ha valami nem stimmel.

Jelszó visszaállítás ➤ Beérkező levelek x



Chatex support (no reply) <chatexfejlesztok@gmail.com>

címzett: én ▼

Kattints az alábbi linkre a jelszó visszaállításához:

http://localhost/ChatexProject/chatex_phps/reset_password/open_reset_window.php?token=3e48e4658628513eee049b5cf335f9f6dba2a2a9

Ez a link 15 percig érvényes.

Chatex

Jelszó visszaállítás - Google Chrome

localhost/ChatexProject/chatex_php/reset_password/reset_passw...

Új jelszó megadása

ocsi2005levente@gmail.com címre

A jelszónak 8-20 karakter hosszúnak kell lennie, tartalmaznia kell legalább 1 kisbetűt, 1 nagybetűt és 1 számot!

A jelszavak nem egyeznek!

Jelszó helyreállítása

Jelszó visszaállítás > Beérkező levelek x

Chatex support (no reply) <chatexfejlesztok@gmail.com>
címezett: én ▾

Kattints az alábbi linkre a jelszó

http://localhost/ChatexProject/chatex_php/reset_password/open_r

Ez a link 15 percig érvényes.

Chatex

← Válasz → Továbbítás 😊

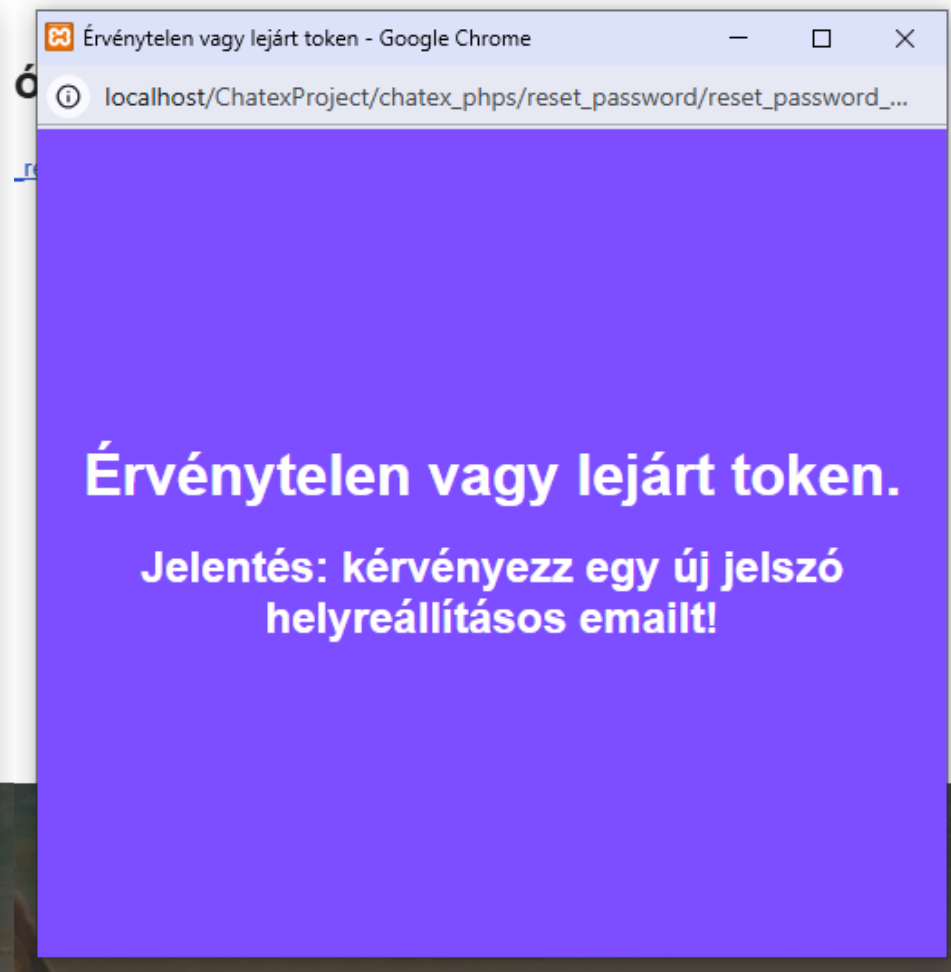
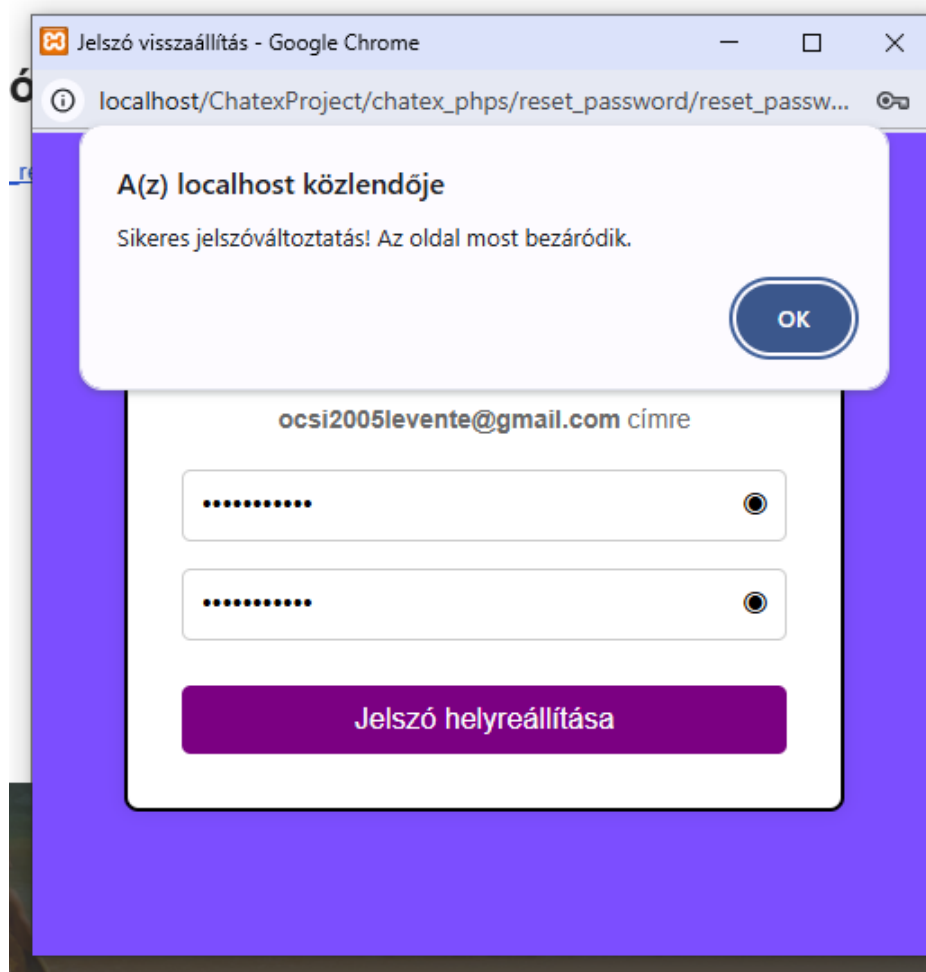
Jelszó visszaállítás - Google Chrome

localhost/ChatexProject/chatex_php/reset_password/reset_password_...

Új jelszó megadása

ocsi2005levente@gmail.com címre

Jelszó helyreállítása



2.3 Fájlstruktúra

A Chatex applikáció fájlrendszerének felépítése, ami az alkalmazás működéséhez szükséges.

A PHP fájlok funkciói:

1. Auth

- Tartalmazza az applikációnknak az összes autentikációs funkcióját
- login.php: Bejelentkezési funkció
- register.php: Regisztrációs funkció
- logout.php: Kijelentkezési funkció
- update_status.php: A felhasználó státuszának ellenőrzése (ha nyitva van a mobilon az applikáció akkor Online, különben offline)
- validate_token.php: Megnézi, hogy a token még érvényes-e vagy már elavult

2. Chat

- Tartalmazza a chathez szükséges funkciókat
- get_chats.php: Kimutatja a felhasználónak a jelenlegi csevegéseit
- get_friend_list.php: Kimutatja a barátlistát.
- get_group_chats.php: Kimutatja a felhasználó által felvett csoportokat
- get_messages.php: Kimutatja a felhasználó és egy másik személy közötti üzeneteket
- start_chat.php: Csevegés elindítása egy ismerőssel

3. Friends

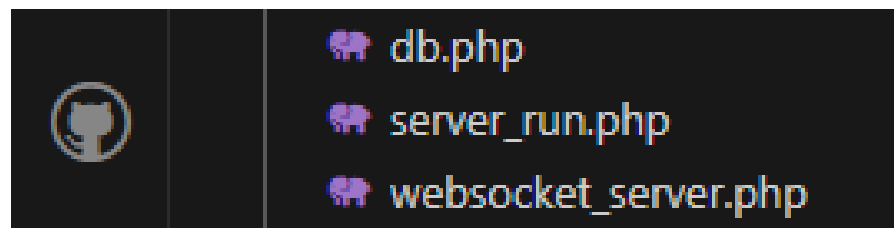
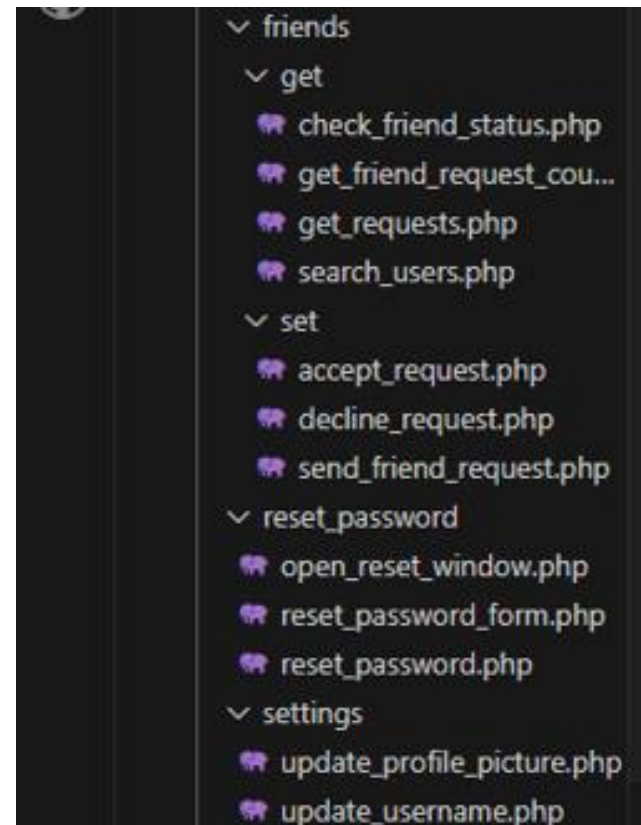
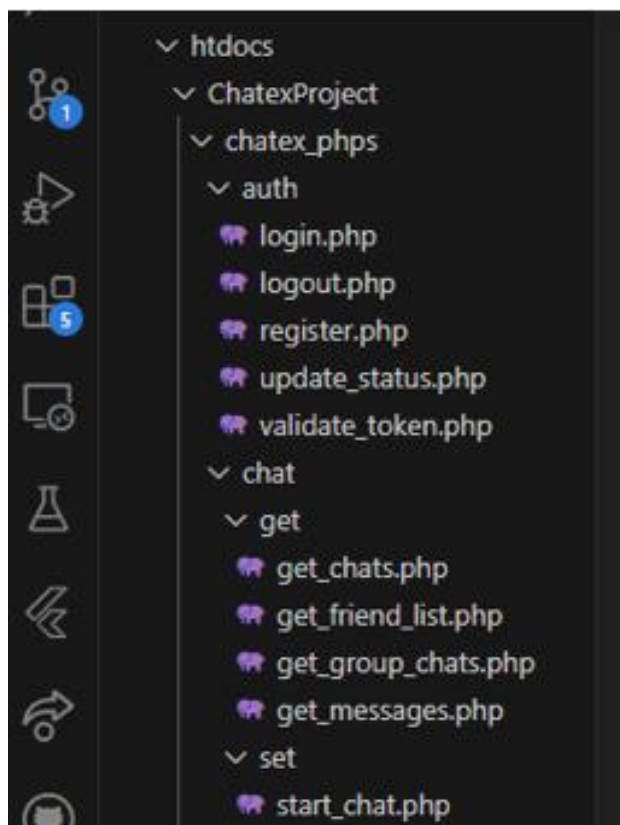
- `check_friend_status.php`: Kimutatja az ismerőseidnek a státuszát (Online – Offline)
- `get_friend_request_count.php`: Az összes még el nem fogadott barátkérelmek számát mutatja ki
- `get_requests.php`: Kapott barátkérelmek kimutatása
- `search_users.php`: Felhasználók keresése
- `accept_request.php`: barátkérelmek elfogadása
- `decline_request.php`: barátkérelmek elutasítása
- `send_friend_request.php`: barátkérelmek elküldése

4. Reset password

- `open_reset_window.php`: Megnyitja a jelszó helyreállításhoz szükséges ablakot
- `reset_password_form.php`: Egy lekicsinyített ablakban megjelenő Form a jelszó helyreállításához
- `reset_password.php`: Jelszó helyreállítása

5. Settings

- `update_profile_picture.php`: Új profilkép beszúrása a felhasználónak
- `update_username.php`: Új felhasználónév készítése



2.4 Tesztelési Dokumentáció

A tesztelés célja az, hogy biztosra menjünk, hogy az applikációnk különböző funkciói (pl.: regisztráció, bejelentkezés, jelszó helyreállítás, ismerősök hozzáadása stb.) helyesen működjenek.

- Ennek két fő része a pozitív és a negatív tesztelés:
 - A pozitív teszteléssel helyes adatokat megadva vizsgáljuk, hogyan veszi át az applikáció és mit csinál vele.
 - Negatív teszteléssel rossz adatot megadva vizsgáljuk azt, hogyan viselkedik és reagál erre az applikáció

2.4.1 Regisztráció tesztek forgatókönyve:

- Test 1: Regisztráció helyes adatokkal.
- Test 2: Regisztráció üres emaillel.
- Test 3: Regisztráció üres jelszóval.
- Test 4: Regisztráció rövid jelszóval.
- Test 5: Regisztráció különleges karakterekkel.
- Test 6: Bejelentkezés rossz adatokkal.
- Test 8: Felhasználó „valaki2” megkeresése.

A teszt helyes adatok esetén sikeresen regisztrál és az adatok megtalálhatók lesznek az adatbázisban.

Helytelen adatok esetén megjelenik a hibát jelző toastmessage.

2.4.2 Tesztek pontosabb ismertetése:

- **Test 1:**

- Azonosító: Valid registration
- Elvárt adatok:
 - Email: validEmail
 - Jelszó: validPassword123
- Eredmény: sikeresen regisztrál és nem ad ki hibaüzenetet.

- **Test 2:**

- Azonosító: Registration with empty email
- Elvárt adatok:
 - Email:
 - Jelszó: validPassword123
- Eredmény: Sikertelen regisztrálás, toastmessage megjelenik, hogy nem lehet üres emaillel regisztrálni.

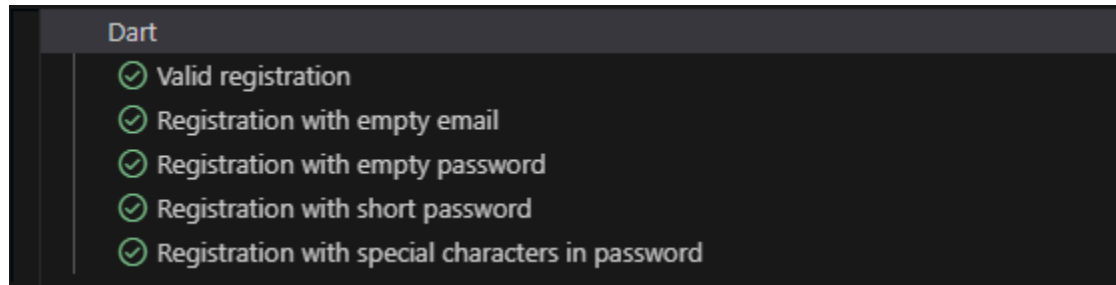
- **Test 3:**

- Azonosító: Registration with empty password
- Elvárt adatok:
 - Email: validEmail
 - Jelszó:
- Eredmény: Sikertelen regisztrálás, toastmessage megjelenik, hogy hibás a jelszó.

- **Test 4:**

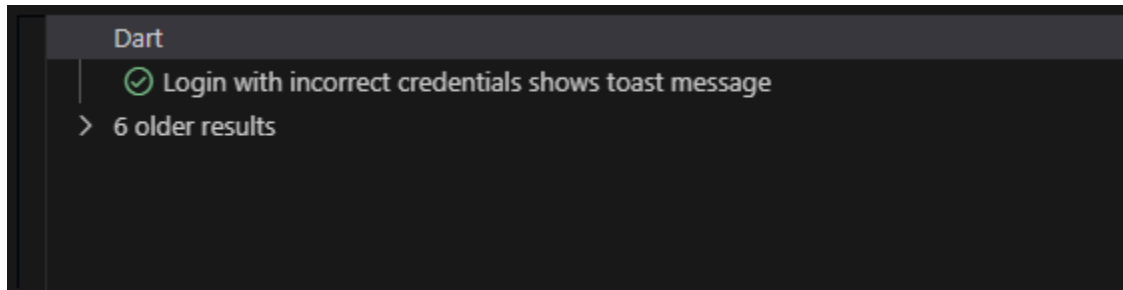
- Azonosító: Registration with short password
- Elvárt adatok:

- Email: validEmail
- Jelszó: 123
- Eredmény: Sikertelen regisztrálás, toastmessage kiírja, hogy túl rövid a jelszó.
- **Test 5:**
 - Azonosító: Registration with special characters in password
 - Elvárt adatok:
 - Email: validEmail
 - Jelszó: @nval!dPassword123)
 - Eredmény: Sikeres regisztrálás és bekerül az adatbázisba



2.4.3 Bejelentkezési teszt:

- Rossz adatok beírásával teszt
- Azonosító: Login with incorrect credentials shows toastmessage



- Elvárt adatok:

- Email: wrongemail@example.com
- Jelszó: wrongpassword

- Eredmény: A bejelentkezés gomb rányomására megjelenik a toastmessage, ami azt írja, hogy „Hibás Email vagy jelszó!” pirosban.

2.4.4 Felhasználó keresés teszt:

- Az applikáció felületén rányom az ismerősök gombra a jobb alsó sarokban, ami átviszi az ismerősök felületre, az ott lévő írható felületre beírja azt, hogy „valaki2”, a felület kimutatja a felhasználó profilját, és rányom a barát hozzáadás gombra.
- Azonosító: FindValaki test
- Eredmény: Sikeresen megtalálja a valaki2 felhasználót és sikeresen elküldi az ismerősnek jelölést



3. Felhasználói dokumentáció

A Chatex egy chat applikáció, amit arra szeretnénk fejleszteni, hogy majd jobb legyen, mint a Messenger, mert szerintünk azért, mert szétpakolnak egy chat applikációt sok haszontalan képességgel, nem lesz attól jobb, csak nehezebb lesz rajta kiigazodni, és mi ezen szeretnénk javítani!

Hardver követelmények: Modern érintőképernyős eszköz

Operációs rendszer: Legalább Android 5.0-ás verzió

Szoftver követelmények: szükséges, hogy minimum x86_64 architektúrás processzorral rendelkezzen az eszköze (legtöbb telefon), hogy az applikációnk tudjon futni telefonján.

Internetkapcsolat: legalább 5 Mbps (kötelező)

Minimum képernyőfelbontás: 360x640 pixel.

3.1 Applikáció használatának részletes ismertetése

Itt az applikáció bejelentkezési felületét látod, innét tudsz bejelentkezni a fiókodba, hogy el tudj kezdeni chatelni, regisztrálni, ha nem lenne még fiókod, megváltoztatni az alkalmazás által használt nyelvet és a jelszavadat helyreállítani, ha elfelejtetted, vagy csak újat akarsz csináltatni.

Ha van fiókod akkor az emailed és jelszavad beírásával be tudsz jelentkezni, ha elfelejtetted a jelszavad, akkor nyomj az elfelejtett jelszó gombra és ha még nincs fiókod akkor készíthetsz egyet az Új fiók létrehozása gombra nyomva.

Mivel az applikációnk egy chat alkalmazás ezért minden egyes funkcióhoz szükséges egy fiók, amit csak a regisztráción keresztül tudsz létrehozni, ahhoz, hogy hogyan kell regisztrálni, itt megtalálod:

Ezeket a lépéseket kell végig menned, hogy sikeresen regisztrálj:

Az alkalmazás megnyitása után megjelenő felület (ha a felhasználó nincs bejelentkezve).

- Főoldal (betöltés után „Új fiók létrehozása” gomb)

Miután a felhasználó rányomott a „Új fiók létrehozása” gombra átvizsgálja a regisztrációs felületre, ahol ez található:

- Felhasználónév mező (kötelező kitölteni)
- Email cím mező (kötelező kitölteni)
- Jelszó mező (kötelező kitölteni)
- Jelszó megerősítése mező (kötelező kitölteni)
- Regisztrálás gomb (ezzel a gombbal regisztrálsz, ha minden mező stimmel)
- Jelszó megjelenítése gomb (amivel megnézheted mit írtál)
- Megváltozik a mező színe, ha nem helyes adatot adnak meg (piros = hiba, lila/fehér = jó érték)
- Toast message-el jelzi, hogy sikeres volt-e a regisztráció (képernyő alján lévő üzenet buborék)

A mezők követelményei:

- A felhasználónév rész elvár legalább három karaktert és nem lehet túl 20 karakteren
- Az email cím rész elvár egy rendes email formátumot, szóval @-jel, az email nevezője (Pl.: gmail) és a domain név (Pl.: .com).
- A jelszó rész elvár minimum 8 karaktert, amiben kell lennie 1 kisbetűnek, 1 nagybetűnek és 1 számot, maximum 20 karakter
- A jelszó megerősítésnél ugyan az, mint a jelszónál, csak még meg kell egyeznie a jelszóval is.

A következő oldalon látszódik a minta a regisztráláshoz:

20:21

← Regisztráció

Felhasználónév

E-mail cím

pl: valaki@kiszolgalo.hu

Jelszó

Min. 8 karakter, Max. 20 karakter,
1 kisbetű, 1 nagybetű, és 1 szám.

Jelszó újra

Regisztrálás

Chatex

20:23

← Regisztráció

valaki

hibasemail

Megfelelo1

Megfelelo2

Regisztrálás

Chatex

20:23

← Registration

va

megfelelo@megfelelo.hu

nemmegfelelo

nemmegfelelo

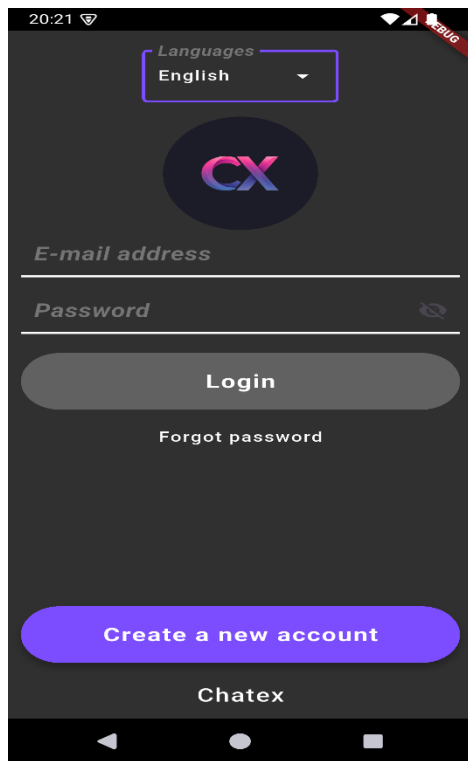
Sign up

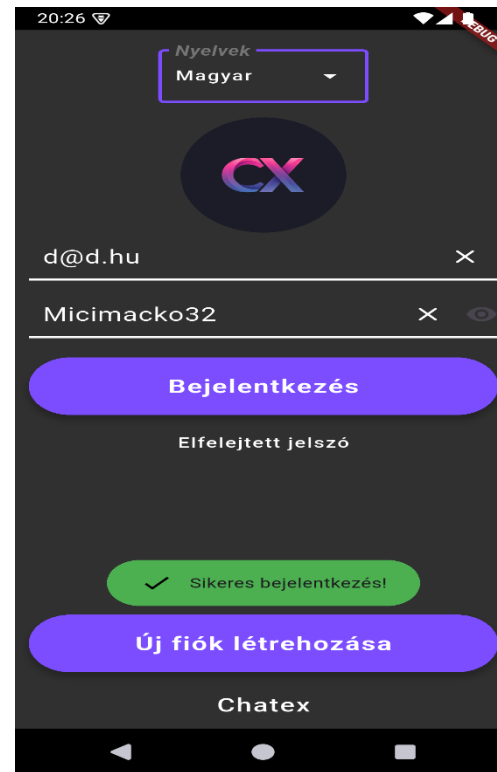
Chatex

Most a bejelentkezést fogjuk szemléltetni, és a főképernyőt, hogy hogyan épül fel:

Bejelentkezési felület tartalmazza:

- Email cím mezőt (regisztrált email címmel tud belépni csak a felhasználó!)
- Jelszó mező (regisztrált jelszóval tud belépni csak a felhasználó!)
- Bejelentkezés gomb (ami csak validált, és érvényes adatokkal lehetséges!)
- Jelszót megjelenítése (ha tudni akarjuk mit gépeltünk el akkor ez hasznos lehet)
- elfelejtett jelszó gomb (rákattintva megjelenik a jelszó helyreállítási mező, amit majd később!)
- Nyelv megváltoztatása (legördülő menü, ahol a preferált nyelvet tudja a felhasználó beállítani angol és magyar között)
- Toast message-el jelez vissza, hogy sikeres volt-e a bejelentkezés!



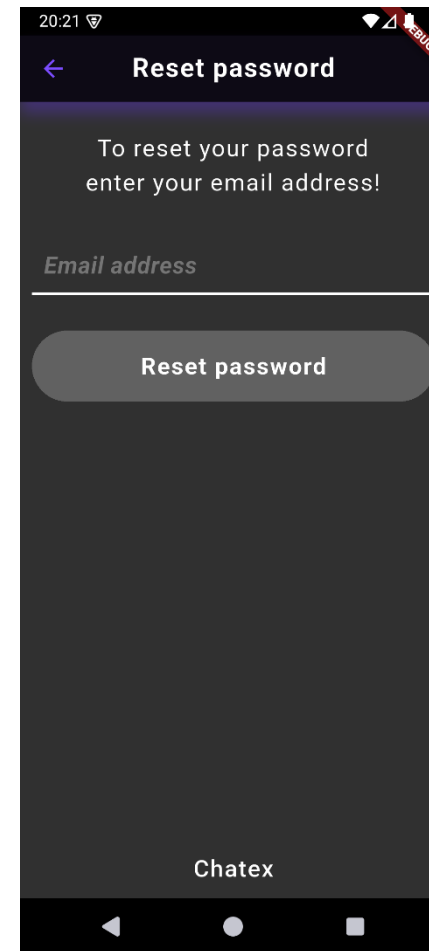
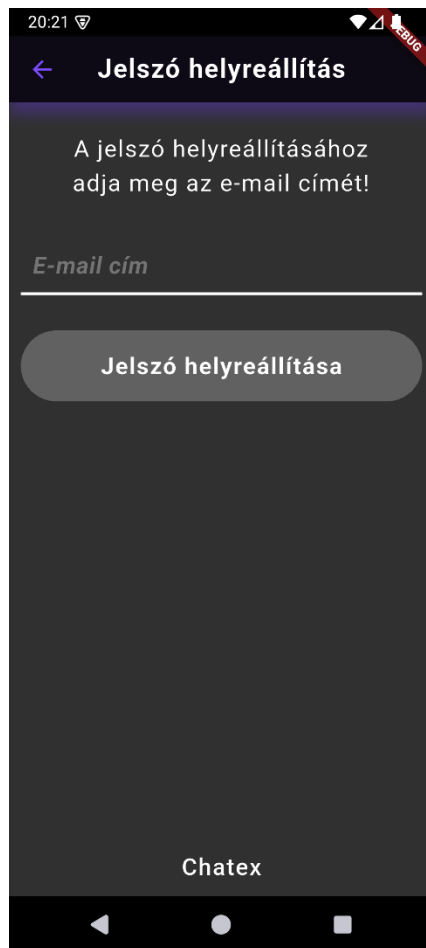


3.1.1 Elfelejtett jelszó folyamatának ismertetése

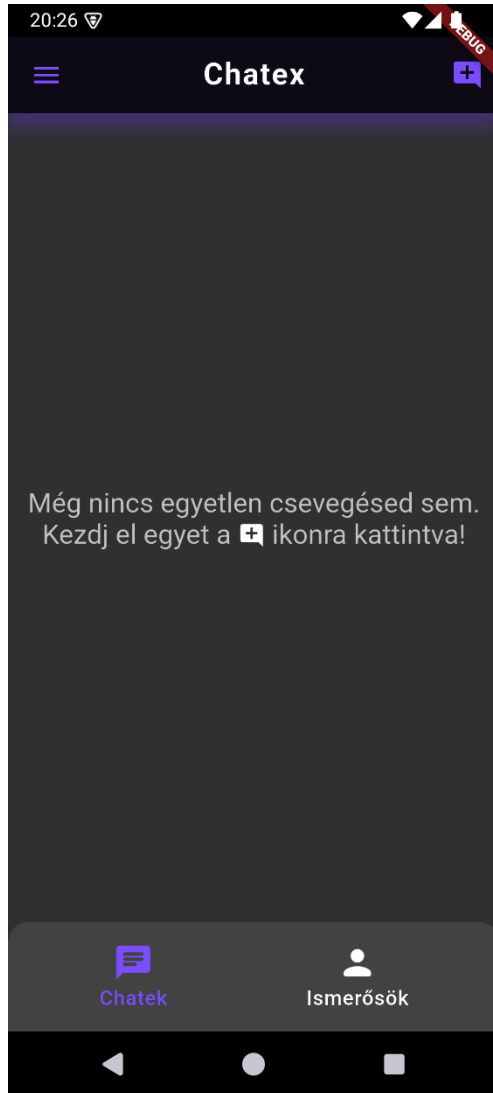
Elfelejtett jelszó felület:

1. Megadja a felhasználó az email címét
2. Megnyomja a Jelszó helyreállítása gombot

Ha helyes email címet adott meg, akkor az email fiókjában találja az üzenetet az új jelszó kéréséről.



3.1.2 Képernyők ismertetése



Bejelentkezés után a chatek képernyő fog fogadni, ahol szinte mindent lehet kezelni, most ezt szemléltetjük:

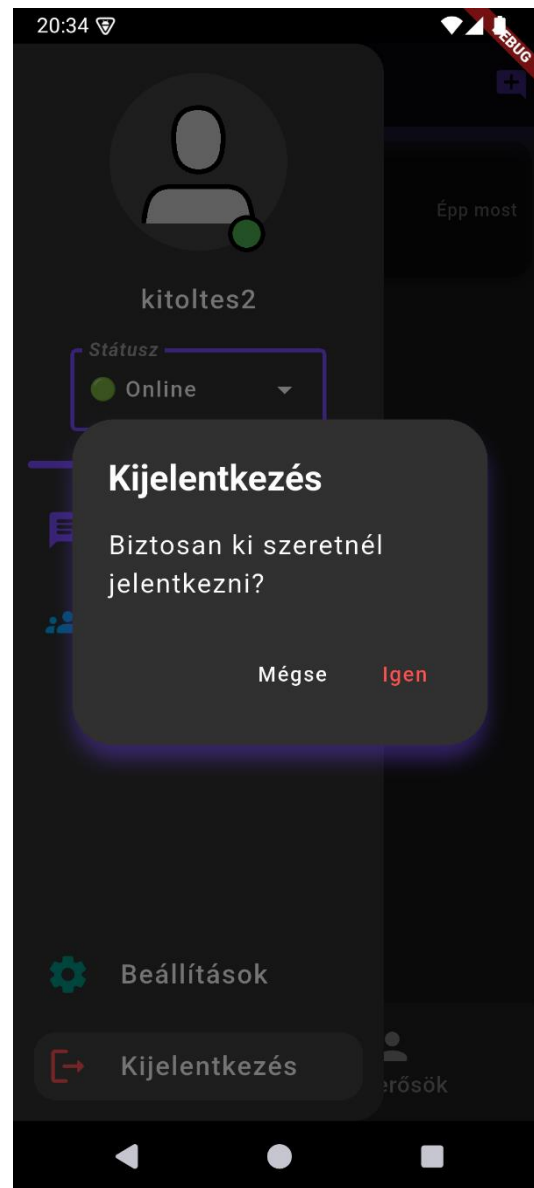
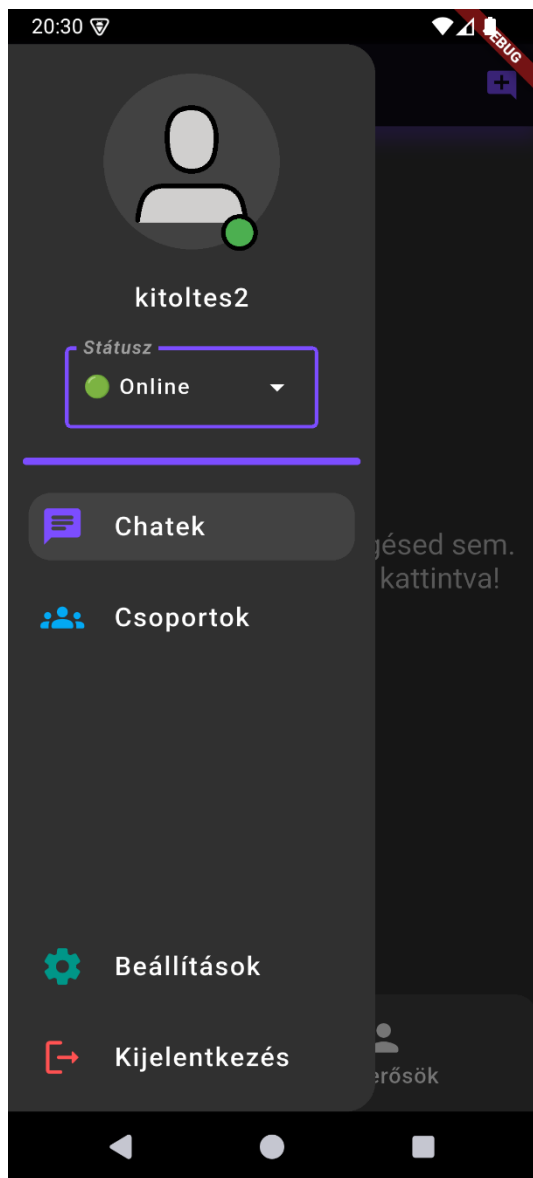
Alapértelmezett nincsen egy chat-je sem a felhasználónak, de a Chatex úgy lett megírva, hogy tanácstalan felhasználókon szövegeken keresztül segítsen, ez látszik a képernyőn:

Egy kis jelmagyarázat, hogy mi mit csinál:

- A chatek képernyő betölti a felhasználó chatjeit (ez az alapértelmezett is!)
- Az Ismerősök gombra kattintva betölti az ismerősök képernyőt, ahol közösségi funkciók vannak implementálva.
- A jobb fenti ikonra kattintva betölti a chat létrehozása képernyőt
- A bal fenti ikon pedig megnyitja az oldalsó menüt, ahol további képernyőkre navigálhatunk

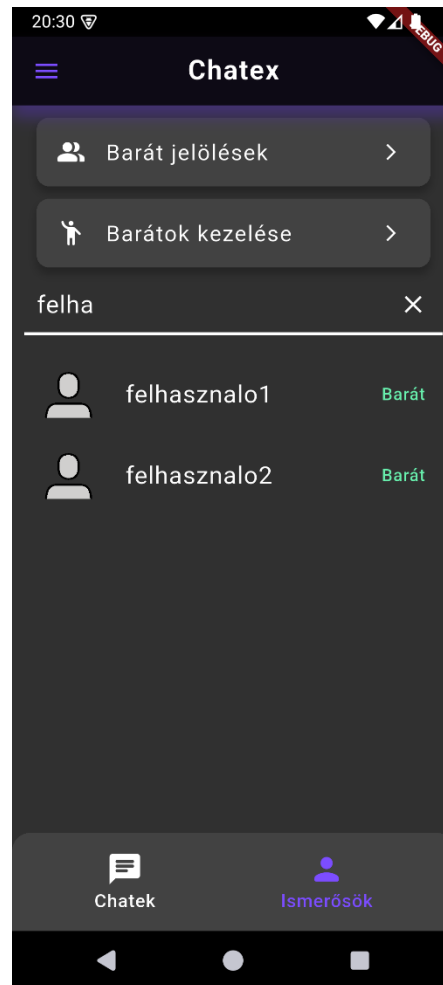
Az említett képernyők részletesen (amik nem a főképernyőn találhatók):

Az első képen az oldalsó menü látható, míg a másodikon a kijelentkezés művelete:



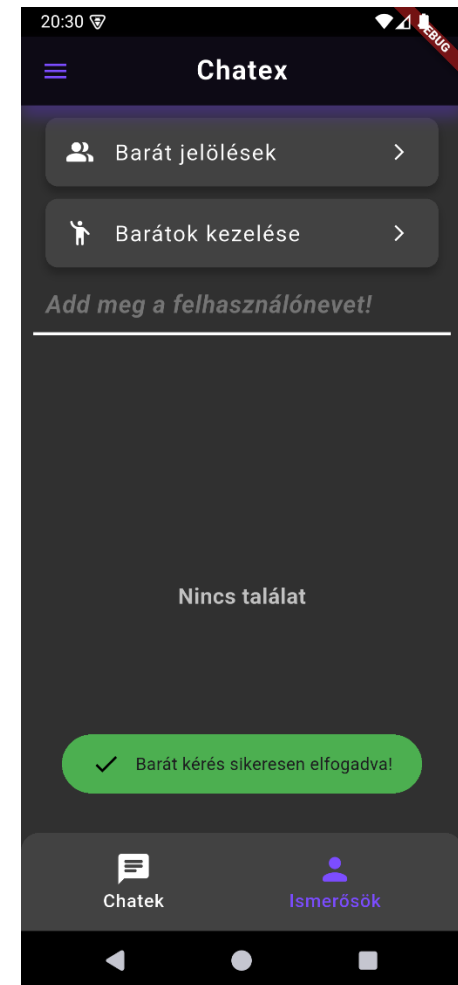
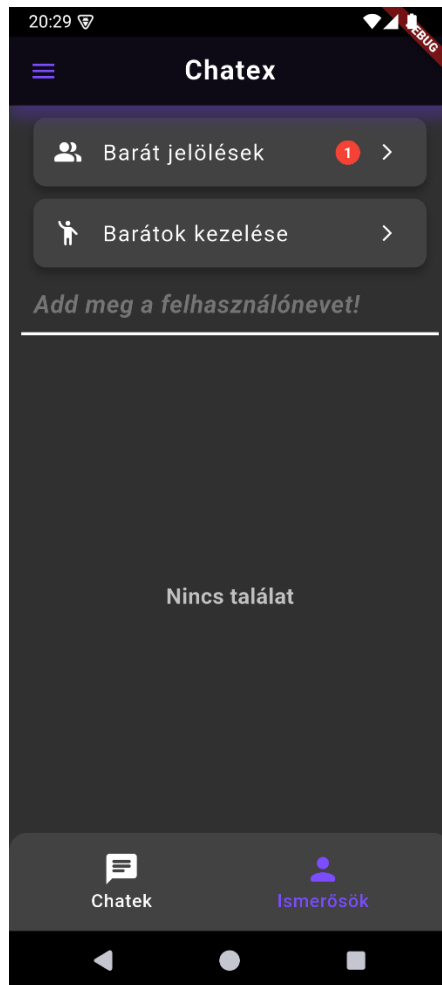
3.1.3 Barát keresése folyamat ismertetése

A fő felületen rányomsz a „Ismerősök (Friends)” oldalra, ami előhívza a bal oldali képen látható felületet, oda kell beírnod egy fióknak a felhasználónevét, ha van hasonló találat vagy teljes egyezés, akkor kimutatja azt a felhasználót, utána már csak az „Jelölés (Add)” gombra kell rányomnod és el is küldted a felhasználónak a barátkérelmed, ha a felhasználó elfogadta a kérést akkor a képen látható „Barát” felirat fog megjelenni!



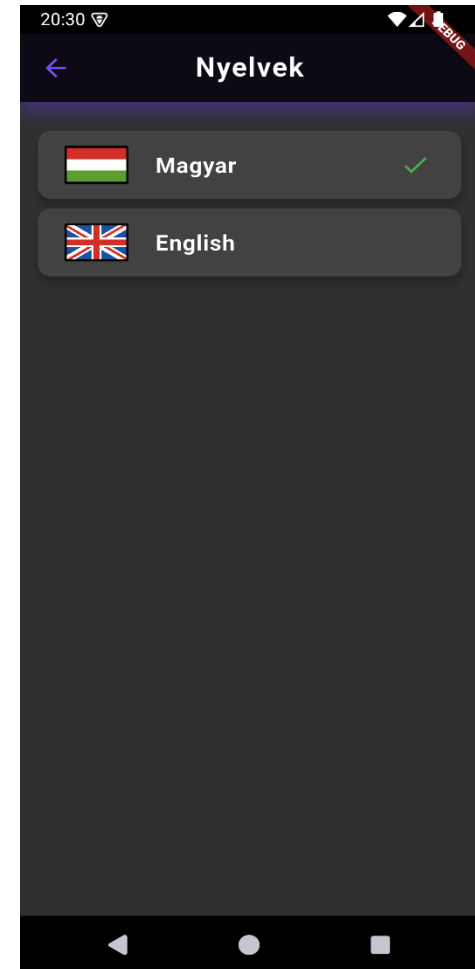
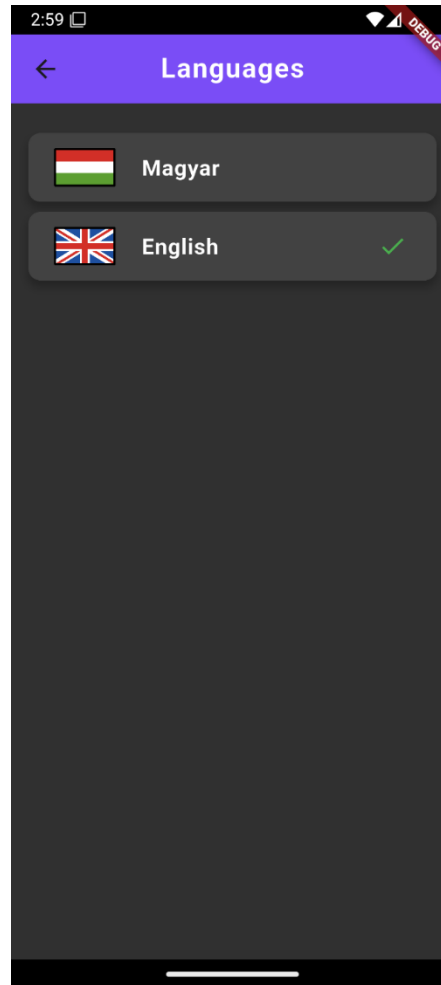
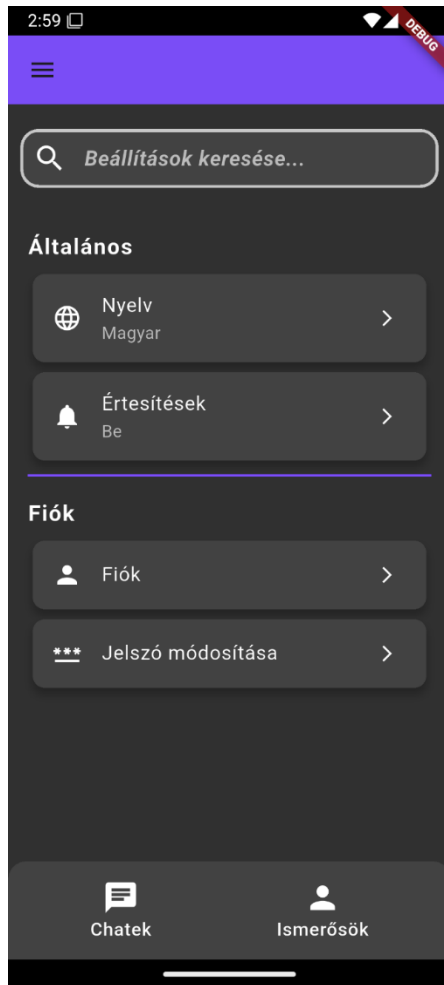
3.1.4 Barátjelölések elfogadása

Ha kapsz egy barátkérelmet, akkor bármikor amikor az Ismerősök felületre lépsz, kimutatva lesz piros számmal a „Barát jelölések” oldal, rá kattintva kifogja mutatni azokat a felhasználókat, akiknek a barátkérelmeit nem fogadtad vagy utasítottad el, ha el szeretnéd fogadni, csak a zöld pipára kell nyomnod, és ha el akarod utasítani, akkor a piros X-re.



3.1.5 Beállítások navigálása

A beállítások felületre nyomva előhúzza a bal oldali képernyőt, itt meg tudod változtatni az applikáció nyelvét, ki és be tudod kapcsolni az értesítéseket, a fiók adatait meg tudod nézni és a jelszavadat módosítani, még a Nyelv menüt bővíteni fogjuk több nyelvel.





3.1.6 Fiók módosítása


A fiók felületen meg tudod tekinteni az adataidat és néhány dolgot még módosítani is tudsz, mint például a profilképedet megváltoztatni akármilyen png, jpeg fájlal, a felhasználónevedet, jelszavadat, és még az email címedet is megváltoztathatod. Akár még törölheted is a fiókodat!


20:30

← **Fiók kezelése**


Fiók adatai


Felhasználónév  Profilkép 

Felhasználónév 


Email cím 

Email cím


Jelszó módosítása 

Jelszó módosítása 

Min. 8 karakter, Max. 20 karakter,
1 kisbetű, 1 nagybetű, és 1 szám.

Jelszó újra 

Módosítások mentése

 **Fiók törlése**

3.2.1 További, még nem implementált ötleteink

Még biztosan tovább lehetne fejleszteni ezt az applikációt, már van pár ötletünk is, Például:

1. Kétlépcsős hitelesítés, hogy biztonságosabb legyen a felhasználók fiókja.
2. Bármikor elérhető szerver kapcsolat. (online működés)
3. Több testreszabhatósági lehetőség profilok és chatekhez az egyéniség kedvéért.
4. Még több nyelv hozzáadása, hogy még több felhasználó tudja élvezni az applikációnkat.
5. Egyszerű bejelentkezés Google fiók segítségével.
6. Értesítések teljes implementálása (jelenleg csak státuszként jelenik meg)
7. Híváson, és videó híváson történő csevegés

4. Összefoglalás

4.1 Munkamegosztás

- A munka során az együttműködés Discordon történő hívásokon át és a Github verziókezelő segítségével lett biztosítva.
- **Ötlettervező, projektvezető és programozó (Kiss Levente):** A Projekt témájának kitalálója, feladatok elosztója, A többi csapattárs közötti kapcsolattartás létrehozója.
- **Fejlesztők (Kiss Levente, Szép Dániel):** A chat applikáció Frontend és Backend fejlesztése, tesztek megírása, adatbáziskezelés, funkciók megírása.
- **Designer (Kiss Levente, Szép Dániel):** A chat applikációnk kinézetének tervezése, megírása, az összes felületen lévő egységesítése.
- **Dokumentáció megírása (Szép Dániel, Kiss Levente):** Az applikáció teljes fokát átölelő dokumentáció megírása, végén utolsó korrektúraolvasás.
- **Mesteri munkakerülés (Szabó Richárd):** Hívásokba nem megjelenés, hamar lelépés más és más kifogásokkal, kapcsolattartás teljes kizárása. *Az érdektelenség megtestesítője!*

4.2 Főbb feladatok:

- **Kiss Levente:**

- Applikáció programjainak megírása
- **XAMPP szerver konfigurálása és fenntartása:** PhpMyAdmin, MySQL, Apache szerver a chat applikáció teszteléséhez.
- Az applikáció fő dizájnjának elkészítése, amit a végén egységessé teszünk az applikáción keresztül.
- Alkalmazásnak naprakészen tartása.
- Az alkalmazásnak különböző méreteken való futtatása.

- **Szép Dániel:**

- Tesztesetek megírása.
- Dokumentáció és prezentáció megalkotása
- Program nyelvhelyességének átnézése.

- **Szabó Richárd:**

Sajnos Szabó Richárd osztálytársunk nem bírta követelményeinket teljesíteni, mert el volt foglalta a vizsgán kívüli dolgokkal (Pl.: Sorozatok nézésével).

4.3 Elérhetőségeink:

Email cím: chatexfejlesztok@gmail.com

Telefon: +36 20 542 6746

5. Köszönetnyilvánítás

Ezúton szeretnénk megköszönni **Bloch Tamás** tanárúrnak, a folyamatos támogatást, szakmai iránymutatást és türelmet, amellyel végigkísérte a munkáink elkészítését. Nélküle ez a vizsgaremek nem jöhetett volna létre ilyen formában.

Külön köszönettel tartozunk **Pradalits Tibor** igazgató úrnak, aki lehetővé tette a projekt megvalósítását, valamint biztosította a szükséges feltételeket és háttérrel a munkához.

Hálásak vagyunk minden segítségért és biztatásért, amit a folyamat során kaptunk!