# Bevezetés

Manapság nagyon elterjedtek a hangalapú kommunikációs szoftverek, amik jelentősen az internetes kommunikáció legjobb és legegyszerűbb formája. A szakdolgozatom témáját abból az okból választottam, mivel sokat játszom számítógépes játékokkal online és nincsen olyan voice chat alkalmazás, ami megbízható és az átlag felhasználó számára is egyszerűen kezelhető lenne. A szoftvert leginkább a játékos közösségnek és azoknak a családoknak tervezem, akik nem tudnak gyakran beszélni egymással.

A szoftverem kialakításánál az egyszerűségre és a user interface design-nál a legjobb átláthatóságra fogok törekedni. A szoftvert C# programozási nyelven fogom megvalósítani, a klienst windows formos alkalmazásban és a szervert pedig console-os formában. A szerver adminok a consoles-os ablakba beírt parancsokkal is tudják majd a szervert használókat koordinálni. A szoftvert csak Windows operációs rendszer alatt lehet futtatni. A szoftverhez használni fogom a NAudio könyvtárat, amivel a hanghoz köthető funkciókat fogom kialakítani, mint például a bejövő és kimenő hang kezelése vagy a mikrofon állapota(néma, nyomógombos, hangérzékelés). A szerver és a kliens között TCP protokoll fogja kialakítani a kapcsolatot, ezek után az UDP protokoll veszi át a szerepet az aszinkron hang és szöveg küldésénél. A hang és szöveg küldésére, azért fogok UDP protokollt használni, mert az UDP egészen közel van a valós idejű adattovábbításhoz és az UDP protokoll nem vár választ a címzettől, ezért is ez a legjobb az adatküldésre. A kialakításnál betartom az objektum-orientált programozás és a clean code alapelveit a későbbi bővíthetőség vagy más néven skálázhatóság és átláthatóság érdekében.

# A szoftver megvalósítása

## Specifikációk

**Az alkalmazással szemben a következő funkciók várhatók el:**

* csoportos hang chat
* csoportos szöveges chat
* privát szöveges chat
* felhasználói profil teljes testreszabása
  1. megjelenítendő név
  2. profil kép változtatása
  3. mikrofon hangereje
  4. kimeneti hang hangereje
  5. mikrofon állapota(némítva, nyomógombos, hang érzékelés)
* admin parancsok (kirúgás, kitiltás, felhasználó némítása)

Szerintem ezek az alapvető funkciók egy voice chat alkalmazáshoz, hogy megfelelően, mindenki számára élvezhetően működjön. A hanggal kapcsolatos funkciókat a NAudio könyvtár használatával fogom megvalósítani.

## NAudio könyvtár

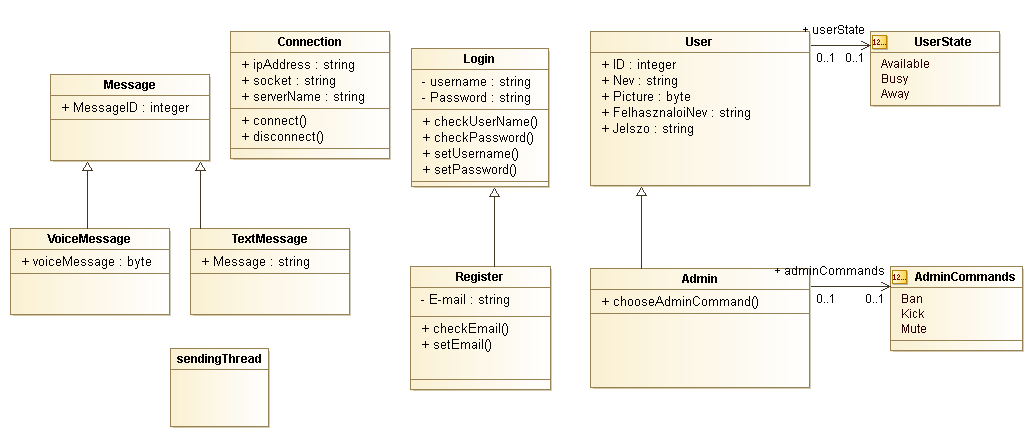
A szoftveremhez a NAudio nyílt forráskódú könyvtárat fogom használni. Ez a könyvtár elsősorban azért jött létre, mivel .NET 1.0-ás verziójában nem volt a hangok lejátszására támogatás. A nyílt forrás kód annyit jelent, hogy bárki hozzáférhet a forráskódhoz vagy akár saját igényei szerint módosíthatja azt, de módosítás után is teljesen mindenki számra elérhetővé kell tenni. A NAudio könyvtárat NuGet package-ként elérhető így tehát a telepítése könnyedén elérhető visual studio fejlesztői környezeten belül. Ez a könyvtár képes hangok felvételére és lejátszására vagy akár meglévő hangfájlok módosítására.

## Git verziókövető rendszer

A szakdolgozatom írása alatt GIT verziókövető rendszert fogok használni. Azért döntöttem a használata mellett, mert szeretem ha van egy biztonsági mentés az eddigi munkámról és ha mondjuk elrontok valamit a kódban, amit utólag nem tudok kijavítani, akkor visszatudom állítani előző verzióra. A verziókövetéshez bitbucket-et és GitKraken használok. A munkámat privát repository-ban tárolom.

## Class diagram

A class diagram mutatja meg a szoftver részei közötti kapcsolatot.



## Használati eset diagram

A használati eset vagy use case diagramot a felhasználók számára elérhető funkciókat mutatja meg. A használati eset diagramokat a szoftverfejlesztés azon fázisában használjuk, amikor az ügyfél meghatározza a specifikációkat és ezekből az ügyfél számára is érthető ábrát hozunk létre.



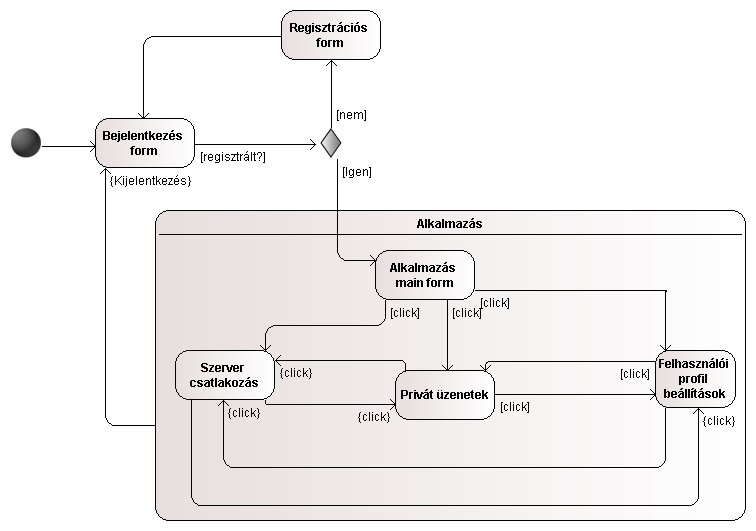
**Aktorok:**

* Átlagos felhasználó
* Admin

Az ábrán a specifikációk cím alatt meghatározott funkciók láthatók. Minden funkció eléréséhez bejelentkezés szükséges. Az Admin aktor leszármazik az Átlagos felhasználóból, ami azt jelenti hogy az Admin minden funkciót tudni fog amit az Átlagos felhasználó is tud, de viszont az átlagos felhasználó nem éri el az admin által használható funkciókat.

## Állapotgép Diagram

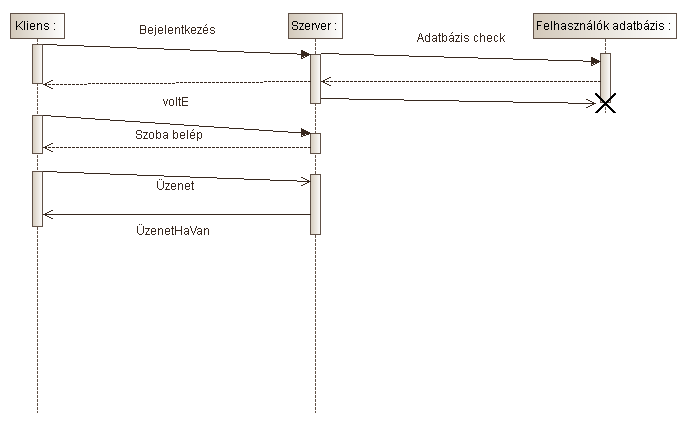
**Az első állapotgép diagramon a** formok közötti váltásokat mutatja be. Az egész szoftver a bejelentkezés formnál kezdődik el, azután ha a felhasználó még nem regisztrált, akkor a szoftver a regisztrációs formra irányítja át. Ha a felhasználó már regisztrált akkor bejelentkezés után az összes funkciót eléri a main formból a bal oldalra kihelyezett gombok megnyomásával. A main formon a kijelentkezés gombra kattintva, újra a bejelentkező form lesz az aktív.



## Szekvencia diagram

A szekvencia diagram a kliens, a szerver és az adatbázis közötti időbeni üzenet váltásokat fogja szemléltetni. A diagramon a legelső üzenet a klienstől a szerver felé egy szinkron üzenet lesz, ami bejelentkezési adatokat(username, password) fog elküldeni, amint a szerver ezeket megkapta egyből az adatbázishoz fordul egy lekérdező szinkron üzenettel, hogy az adott felhasználói név és jelszó létezik-e az adatbázisban. A szerver ezután válasz üzenetet kap és azt egyből továbbítja a kliens felé. A felhasználó, ha beakar lépni egy szobába, akkor a kliens egy szinkron üzenettel felveszi a kapcsolatot a szerverrel, ami visszaküld egy válasz üzenetet,hogy a felhasználó be tud e lépni. Miután a felhasználó belépett a szobában a kliens és a szerver is csak akkor küld üzenetet, ha éppen van beérkező, ezek jelen esetben aszinkron üzenetek lesznek.

Az ábrán teli nyilak mutatják a szinkron üzeneteket, a nem teli nyilak pedig az aszinkron üzeneteket. A folyamat futását a vékony téglalapok szemléltetik.



## Design

A design kialakításánál és megtervezésénél a legjobb átláthatóságra törekszem. A designt a mai trendnek megfelelően fogom kialakítani. Manapság a designban eléggé elterjedt a flat design, ami a letisztult kinézetéről és a 3D-s formák hiányáról ismerszik meg. A design és a felület kialakításánál a windows form elemeit fogom használni. A színek kiválasztásánál az alkalmazás fantázia nevére hagyatkozom, ami a sPeach. A Peach magyarul barackot jelent, tehát a színeket is a barack színeihez fogom igazítani.

**Login és Regisztrációs Form:**

* bigLabel:
  + Font style: Lucida Sans, félkövér
  + Font size: 32,25 pt
* username/password textbox:
  + Background color: SeaShell
  + Font style: Lucida Sans, félkövér
  + Font size: 16 pt
* Sign in gomb:
  + Background color: SeaShell
  + Font style: Lucida Sans, félkövér
  + Font size: 9pt
* Linkelt label:
  + Font style: Lucida Sans, félkövér
  + Font size: 9pt
  + Link color: 0;0;225
  + Visited link color: 128;0;128

A szoftver bal felső sarkában látható ikonhoz egy internetről letöltött képet körbe vágtam és transzperenssé tettem aztán a képet lekicsinyítettem 128x128 px-es méretre. Aztán online konvertáló segítségével png formátumból ico formátumba konvertáltam át.

A szoftver logójához ugyan azt a képet vágtam körbe csak nagyobb méretben használtam fel és a logot 250x150 px-re állítottam be, ami egy elterjedt logo méret. A logon még megtalálható a szoftverem fantázia neve is.

A regisztrációs és a login form között átváltást a Hide() és a Show() metódussal valósítottam meg, amik azt csinálják hogy a login formot a háttérben tartják a regisztrációs formot meg az előtérbe rakják.

A main formon az állapotok közötti váltást a panel közti váltással oldom meg. Mindig más gomb nyomásra más panel fog láthatóvá válni, míg a többi láthatatlan lesz.

A gombok mellett egy vékony panel fog elhelyezkedni, ami a menüpontok aktív állapotát fogja jelezni.