# 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 embereknek tervezem, akik nem tudnak gyakran beszélni egymással.

A szoftverem kialakításánál az egyszerűségre és a felhasználó felület design-nál a legjobb átláthatóságra fogok törekedni. A szoftvert C# programozási nyelven .NET keretrendszer és NAudio könyvtár segítségével fogom megvalósítani, a klienst windows formos alkalmazásban és a szervert pedig console-os formában. A szerver adminok a consoles-os ablakba beírt parancsokkal vagy vizuálisan kialakított többlet funkciókkal tudják majd a szervert használókat koordinálni. A szoftvert akármilyen operációs rendszer alatt lehet majd futtatni, amire telepíthető a .NET keretrendszer, ezzel a platform függetlenséget megoldottam a szoftverem kapcsán. 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 bejelentkezés és a regisztráció teljesen titkosítva SHA2-es titkosítással fog történni a küldés közbeni törhetőséget elkerülvén. A szerver és a kliens között TCP protokoll fogja kialakítani a kapcsolatot, ezek után az UDP és az RTP protokoll veszi át a szerepet az szöveg és az aszinkron hang küldésénél. A 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 hang küldésére viszont real-time protokollt fogok használni, hogy a felek késés nélkül tudjanak beszélgetni egymással. A kialakításnál betartom az objektum-orientált programozás és a clean code alapelveit a későbbi bővíthetőség vagy más néven skálázhatóság és átláthatóság érdekében. A szoftver több felhasználós rendszerben lesz, ami annyit jelent, hogy két felhasználó típus lesz az alkalmazáson belül, az átlagos felhasználók és az admin vagy adminisztrátorok, akik moderálják a szerveren tartózkodókat. Az adminoknak külön belépési felhasználó neve és jelszava lesz amit a rendszer érzékel és az adminok által használt funkciókkal kibővített klienst fognak kapni.

# A feladat megvalósításához használt szoftverek és külső kódok

## 2.1 Visual Studio 2017

A Visual Studio legújabb verzióját fogom használni a feladat megvalósítása során, mert ez a legátláthatóbb és legtöbb segítséget nyújtó IDE, vagyis fejlesztői környezet. A telepítés ez új verzióban egy moduláris telepítővel telepíthető, ahol ki lehet választani, hogy milyen részegységeket telepíthetünk a Visual Studiohoz. A Visual Studioban rengeteg programozási nyelv támogatott, ezért szinte akármilyen project megvalósítható benne, ezen felül a szintaktikai hibákat egyből jelzi a fejlesztő számára.

A Visual Studio legújabb verziójában lehetőség van a valós idejű egység tesztekre, ami azért jó mivel ha valaki elakad egy szemantikai hiba felderítésével akkor a hiba egyből felderíthető, mivel a szoftver egyből teszteli kód írása közben és a teszt eredmények akármikor elérhetőek lesznek.

A Visual Studioban lehetőség van külső keretrendszereket és könyvtárakat csatolni a NuGet package manager segítségével.

## 2.2 Modelio 3.6

A Modelio program egy nyílt forráskódú szoftver, amiben akármilyen UML diagramot tervezhetünk és szerkeszthetünk. A class diagramunkat még kóddá is alakíthatjuk. Mivel a szoftver nyílt forráskódú, ezért a szoftverben nagyon sok hiba van még, de a szoftverem megvalósításában ez nem hátráltat.

## 2.3 Photoshop

A logo és egyéb grafikai munkákhoz a Photoshop CS6-ot fogom használni. Minden Photoshop projectemet és képemet elmentem a késöbbi módosítás érdekében.

## 2.4 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. Jelenleg ez az egyetlen könyvtár hangok kezelésére, ami C# nyelvhez i elérhető.

## 2.5 Git verziókövető rendszer

A szakdolgozatom írása alatt GIT verziókövető rendszert fogok használni. A Git egy nyílt forráskódú verzió kezelő rendszer, amit Linus Torvalds fejlesztett ki a Linux kernel fejlesztéséhez. Nagyobb projecteknél több ágon is dolgoznak a fejlesztők, de én a projectemnél csak egy ágat fogok használni.

Azért döntöttem a Git 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, ezenfelül Pendrive-ra is hetente lementem. A verziókövetéshez bitbucket.org-ot és GitKraken használok. A munkámat privát repository-ban tárolom. A GitKraken egy vizuális interface a Git verziókezelőhöz, ami rendkívül átlátható és friss design-al rendelkezik.

## 2.6 Navicat

Az adatbázis kialakításához és megtervezéséhez a Navicat nevű szoftvert fogom használni, mert az ingyenesen letölthető verzió is több féle adatbázis motorral és a feladatom megvalósításához elegendő funkcióval rendelkezik. A navicat-el gyorsan és egyszerűen megvalósítható akármilyen adatbázis.

## 2.7 Xampp

Az adatbázis futtatásához a Xampp nevű szoftvert fogom használni. A Xampp egy webfejlesztéshez használt szoftver-csomag, ami tartalmaz, webszervert, adatbázis-kezelőt és PHP programozási nyelv értelmezőt. Én a szoftveremhez az adatbázis kezelőt fogom ezek közül használni.

## 2.8 .NET keretrendszer

A .NET egy Microsoft által fejlesztett nyílt forráskódú keretrendszer. A .NET 4.7 a jelenlegi legújabb verzió, ami elérhető. A Windows formok már a 2.0 –ás verzióban elérhetőek voltak, amit 2005-ben adtak ki. A .NET keretrendszer az előre megírt függvényekkel és osztályokkal megkönnyíti a szoftverfejlesztés folyamatát. A .NET keretrendszer alatt megírt szoftverek platform függetlenek, tehát amelyik számítógépen telepítve van a keretrendszer azokon futtathatók lesznek.

# A szoftverhez használt hálózati protokollok

A hálózati protokollok írányítják az adott adatokat a kettő vagy több fél között. Helyzettől függően mindig más-más tulajdonsággal rendelkező protokollra lehet szükségünk. Az interneten a két leggyakrabban használt protokoll a TCP/IP és az UDP. A protokollok elsősorban a küldött üzenetek felépítését adják meg. Az általam használt protokollok az OSI-modell 3.-ik rétegében helyezkedik el, ami a szállítási réteg. A szállítási rétegnek nem kell ismernie a hálózati topológiát, amin a csomag halad, csak a kommunikációban részt vevő két félnél van rá szükség.

**Transmission Control Protocol TCP/IP:** A protokoll feladata, hogy a szerver és a kliens közötti kapcsolatot kialakítsa és a bejelentkezési felhasználó nevet és jelszavat elküldje a szervernek. Az adatvesztés ennél a protokollnál teljes mértékben elkerülhető. A protokoll elsősorban a kapcsolatot alakítja ki és csak aztán kezdi meg az adat küldést. A szerver mindig küld egy acknowledge üzenetet a kliens felé, ami jelzi a kliens felé, hogy a csomag megérkezett-e. A csomag ha nem érkezik meg, akkor a protokoll újra küldi ugyan azt a csomagot, de viszont ha megkapta a csomagot a szerver akkor a kliens folytatja a további csomagok küldésével. OSI modell szerint a protokoll a szállítási rétegben helyezkedik el.

**User Datagram Protocol (UDP):** Az UDP protokoll-t a szoftveremben a szöveges chat üzenetek küldésénél fogom használni, mivel az üzenetek megérkezésénél fontos a gyorsaság. A gond az UDP protokollal, hogy nem garantálja az üzenetek megérkezését, de ezenfelül ez a legjobb módja a szöveges üzenetek elküldésére.

**Real-time Transfer Protocol (RTP):** Az RTP protokoll a leghatékonyabb módja a hang üzenetek küldésére. Az RTP az UDP port tartományából teljesen véletlenszerűen választ. A hang adat amit elküldtünk elveszne akkor azt a másik felhasználó úgy hallja mintha a partnere nem beszélt vagy szaggatva beszélt volna. Ez az adatvesztés leginkább az internet kapcsolat minőségétől függ. Az RTP protokoll a csomagokkal együtt időbélyeget, csomag típust, csomag sorszámot küld. Az UDP-vel ellentétben az RTP figyel a csomagok sorrendjére és figyel az időzítésre is. Egy RTP csomag pontosan 32 bit, ami megint csak a protokoll gyorsaságára utal.

**Real-Time Control Protocol (RTCP):** Ez a protokoll szorosan kapcsolódik az RTP protokollhoz. Ez a protokoll nem továbbít hangot, csak az elküldött adatok minőségi információit küldi vissza, tehát visszajelzést ad a az elveszett csomagokról és a csomagok közti időbeni eltérésről. A protokoll feladata, hogy a sávszélességet lehetőség szerint teljes mértékben kihasználja az RTP protokoll, ezzel is csökkentve az üzenetek késleltetését.

# A szoftver megvalósítása

## 3.1 Specifikációk

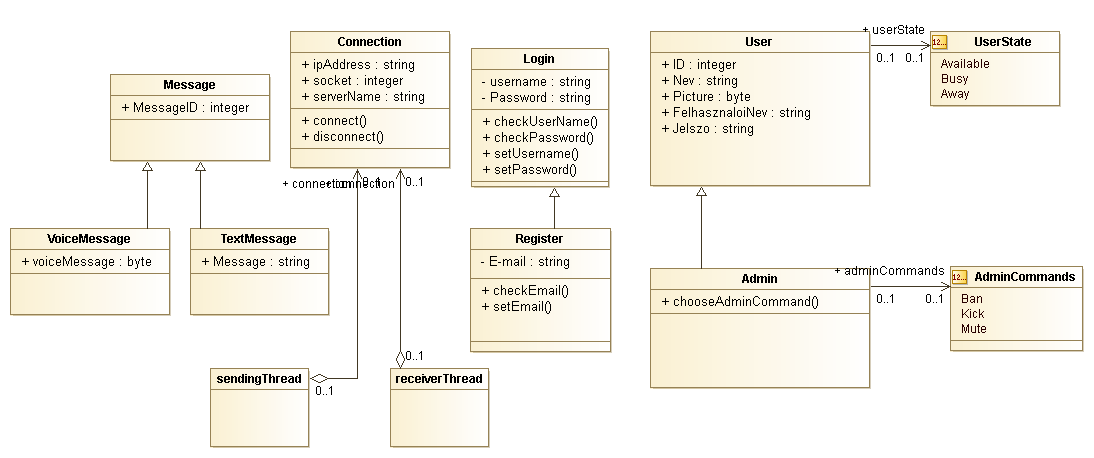
**Az alkalmazással szemben a következő funkciók várhatók el:**

* csoportos hang chat: Egy közös szerveren lévők egymással tudnak beszélgetni a kliens segítségével.
* csoportos szöveges chat: Egy közös szerveren lévők tudnak egymásnak írni, URL linket küldeni és hangulatjeleket küldeni.
* privát szöveges chat: A felhasználók egymással is tudnak szövegesen chatelni, ami teljesen privát formában történik.
* felhasználói profil teljes testreszabása
  1. megjelenítendő név: A többi felhasználó által látható név, akármikor módosítható az Options menüpont alatt.
  2. profil kép változtatása: A profilkép akármikor módosítható az Opciók menüpont alatt, a Windows fájlkezelő segítségével.
  3. mikrofon érzékenysége: A mikrofon felvételi érzékenysége az Opciók menüpont alatt változtatható decibel mértékegységben megadva
  4. kimeneti hang hangereje: A kimeneti hang hangereje nem csak a Windows-ban lesz állítható, hanem a szoftveren belül is lehet majd állítani.
  5. mikrofon állapota(némítva, nyomógombos, hang érzékelés): A mikrofon állapotát az Opciók menüpont alatt lehet módosítani. A mikrofonnak három állapota lesz, némított amikor a felhasználót egyáltalán nem hallják a társai, nyomógombos amikor a felhasználót csak akkor tud beszélni, ha az adott gombot lenyomja és hang érzékeléses amikor a felhasználó bármikor beszél azt egyből hallják a beszélgető partnerei.
* admin parancsok (kirúgás, kitiltás, felhasználó némítása): Az adminok a moderálás érdekében a felhasználókat ki tudják majd rúgni, azaz a felhasználót lecsatlakoztatja a szerverről, de a felhasználó egyből vissza tud majd jönni. Az adminok ki tudják majd tiltani a felhasználókat a szerverről, ami több a kirúgással ellentétben több időre szól. Az adminok még némítani tudják majd a felhasználót, adott időre vagy amíg az admin újra vissza nem kapcsolja.

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. A szoftvert a továbbiakban majd újabb funkciókkal lehet majd kibővíteni.

## 3.2 Class diagram

A class diagram mutatja meg a szoftver részei közötti kapcsolatot.



## 3.3 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:**

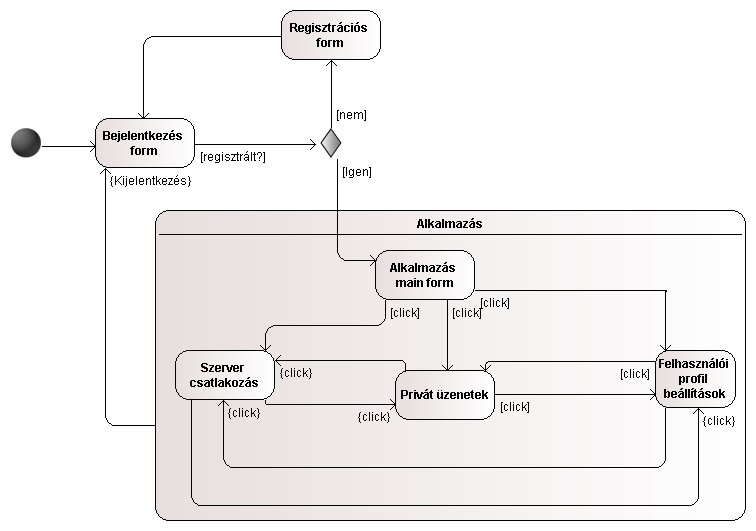
* Átlagos felhasználó
* Admin

Az aktorok a szoftver felhasználóit jelenti.

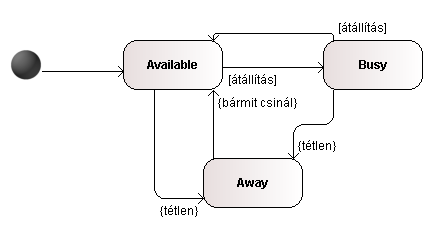
Az ábrán a specifikációk cím alatt meghatározott funkciók láthatók. Minden funkció eléréséhez bejelentkezés szükséges, amit az include kifejezés és a nyíl jelenít meg. A nyíl mindig afelé mutat, amit az adott funkció magában foglal. Az Admin aktor leszármazik az Átlagos felhasználóból, ami azt jelenti 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. A leszármazást a vastagabb fejű nyíl jelez. A nyíl mindig annak az irányába mutat, amiből leszármazik az adott aktor. Az aktor által elérhető funkciókat sima vonallal kötjük az aktorhoz.

## 3.4 Állapotgép Diagram

**Az első állapotgép diagramon** aformok 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 gomb megnyomásának hatására a szoftver a bejelentkezés form-ra irányít.



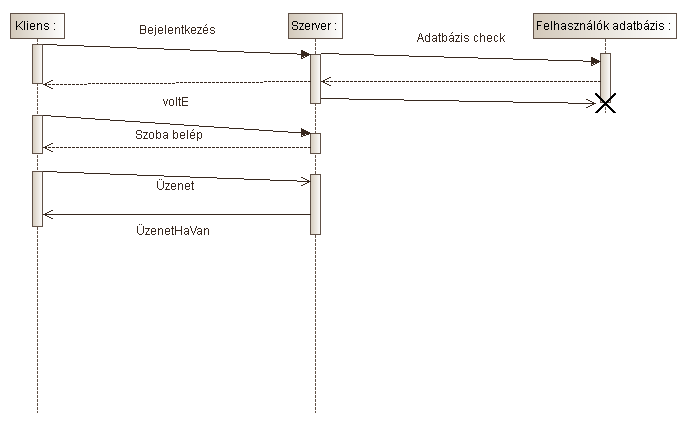
**A második állapotgép diagram** az átlagos felhasználó állapotait mutatja be. A programba belépéskor mindenfelhasználó elérhető állapotba kerül. Ha, a felhasználó elfoglalt be tudja állítani magának az elfoglalt állapotot és ezután vissza is tudja állítani, ha ismét elérhető. A szoftver attól függetlenül hogy milyen állapotban van a felhasználó egy bizonyos idő elteltével átállítja az állapotát távolira, azaz nincs a gép közelében.



## 3.5 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(felhasználói név, jelszó) 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.



## 3.6 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 ingyenesen letöltött képet körbe vágtam és transzperenssé tettem aztán a képet lekicsinyítettem 128x128 px-es méretre. Aztán <http://convertico.com/> segítségével, ami egy online konvertáló web alkalmazás 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 méret a logok tervezésénél. 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 regisztrációs és login formon a textboxok tartalmát mindig az adott állapothoz igazítottam és ezért a textboxok ha üresek és máshová kattintunk akkor egyből vissza írja az adott mező kezdő tartalmát írja vissza. A jelszó mezőknél a jelszót kitakarva mutatja, de ha üres a mező és átkattintunk máshova, akkor a Password szó vagy Confirm Password szavak vissza íródnak és már nem lesz rejtve a mező tartalma.

A main formon az állapotok közötti váltást a panelek 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 csík formában, ami a menüpontok aktív állapotát fogja jelezni. Ezt úgy alakítom ki, hogy a kis panel magasságának mindig megadom az adott menüpont gombjának a magasságát a windows formon.