<u>Készítette:</u>

Újfalvi Csongor | Kis Szilvia | Cosma Cristian

Számítástechnika III. év

Sapientia EMTE, Marosvásárhely

Tartalom

1 Bevezető

1.1 A felvetett probléma

Napjainkban már szinte minden digitális adat elérhető, megszerezhető valamilyen módszerrel, legyen az akár legálás, akár nem. Az interneten forgó adatok mind ennek a veszélynek vannak kitéve, mivel a leggyakoribb titkosítási módszerek már széles körben ismertek, s nem jelent gondot azoknak, akiknek céljuk ezek kijátszása. Bizonyos adatoknál nem elég az, ha az üzenet tartalmát titokban tartjuk, sokkal hatékonyabb az, ha magát az üzenet létezését is titkosítjuk. Erre létezik egy úgynevezett szteganográfia technika, az úgynevezett üzenettitkosítás művészete, ami lehetővé teszi, hogy az üzenet létezéséről csak a címzett tudjon.

1.2 A projekt célja

A projektünk célja egy olyan alkalmazás készítése, amely egy biztonságosabb kommunikációs csatornát biztosít, mint a legtöbb chat szolgáltatás. Az alkalmazás titkosított információcserét biztosít a felhasználók között. Ezt a titkosítást úgy oldottuk meg, hogy egy kódoló script segítségével egy képbe kódoljuk az elküldeni kívánt üzenetet, amit aztán egy dekódoló fog láthatóvá tenni a fogadó fél számára.

2 Komponensenkénti leírás

2.1 Encode/Decode modul

Az Encode/Decode modul három Python scriptből áll, amelyek parancssorból hívhatóak. A paramétereket is parancssorban kell átadni a scripteknek. A fő script neve initiate.py, ezt különböző (-e vagy -d) opciókkal futtatva vagy az encode vagy a decode scriptet indítjuk el. A decode és az encode scriptek különböző parancssori paraméterek megadásával működnek. Az encode mód esetében ki kell választani a kódoláshoz a színcsatornát (-r, -g, -b) egy opcióval, meg kell adni a bemeneti képfájl nevét, a szöveget, amit a képbe szeretnénk kódolni, a kimeneti képfájl nevét és a kimeneti config fájl nevét. Ez utóbbi tartalmazza azt, hogy milyen színcsatornára kódoltunk, a képfájl nevét és az indexet ameddig tart az elkódolt szöveg. Ez a config fájl szükséges a dekódoláshoz, ezeket az információkat használja fel a decode script. Tehát a decode mód esetében meg kell adnunk a config fájl nevét és egy szöveges fájl nevét amelybe a dekódolt szöveg kerül.

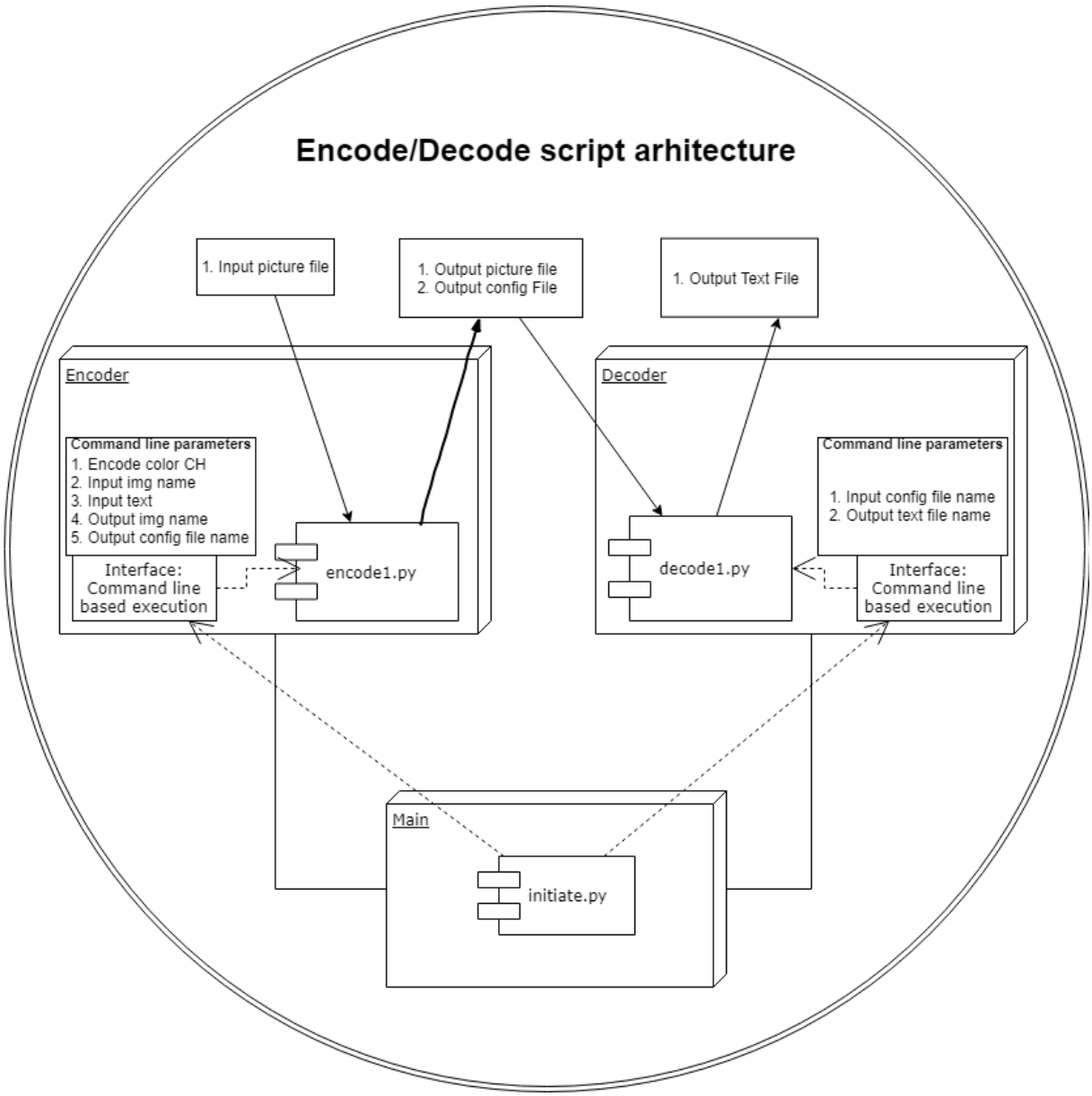
A parancssort az os.system() függvényt használva futtatja a script. Az encode meg decode mód kiválasztásához és az ezekhez tartozó további adatok megadásához az argparse könyvtár függvényeit használja a script, ezek segítenek a parancssori paraméterek egyszerű feldolgozásában és lehetővé teszik, hogy probléma nélkül több paraméterrel is dolgozhassunk vagy akár több opcióval is.

Az encode és decode scriptekben is az argparse könyvtár segít a parancssori paraméterek feldolgozásában. Az encode esetében miután megvan, hogy melyik színcsatornára kell kódolni a script feldolgozza a többi paramétert is és meghívja a függvényt amely a

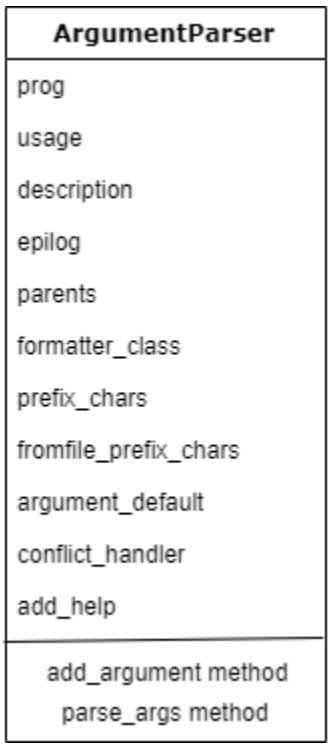
kódolást végzi. Ez a függvény opencv segítségével megnyitja a képet, majd átalakítja a kódolásra szánt szöveget karakterenként ASCII-ba. Majd ezen a karakter listán és a kép pixeleinek, mátrixának elemein iterálva minden főátlón lévő pixel adott színcsatornájú értéke átíródik a karakter listában soron következő ASCII értékre. Mindez addig megy amíg elfogynak a karakterek vagy végigjártuk a pixelmátrix főátlóját. Ez után a módosított pixelmátrixot egy kimeneti képfájlba mentjük ugyancsak Opencv segítségével. Ezt követően a függvény visszatéríti az utolsó karakter index számát, így megjegyezve azt, hogy meddig is tart az elkódolt szöveg a főátlón. Ezt az utóbbi adatot, a kimeneti fájl nevét és a színcsatornát amelyre kódoltunk egy konfigurációs fájlba írja a script. Így már készen is áll minden a decode script számára a visszafejtéshez.

A decode script először megnyitja a konfigurációs fájlt és azt feldolgozva megkapja az adatokat amelyek szükségesek a visszafejtéshez. A script ez után meghívja a decode nevű függvényt megadva a képfájl nevét, a színcsatornát és az utolsó kódolt karakter indexét. Ez a függvény megnyitja a képet opencv segítségével, majd végig iterál a pixelmátrixon. Minden a főátlón elhelyezkedő pixel megfelelő színcsatornán található értékét hozzáfűzi egy listához. Ezt a listát téríti vissza a függvény. Egy egyszerű chr függvény segítségével megtörténik ennek a listának a szöveggé alakítása. Így egy stringben újra az eredeti, kódolás előtti szöveg található. Ezt a script egy paraméterként megadott szöveges fájlba menti.

2.1.1 Encode/Decode script architektúrája



2.1.2 Encode/Decode class diagram



2.2 Kliens/felület

2.2.1 Windows form

A Windows Forms egy .NET-keretrendszerben működő, kezelt könyvtárak egy csoportja, melyeket gazdag kliensalkalmazások fejlesztésére terveztek. Ez egy grafikus API az adatok megjelenítéséhez és a felhasználói interakciók kezeléséhez a könnyebb telepítés és a nagyobb biztonság érdekében.

2.2.2 Grafikus felület

Ma már követelmény, hogy grafikus felületeken (Graphical User Interface (GUI)) történjen a felhasználó és az alkalmazás közötti párbeszéd.

A grafikus felületű alkalmazásokra jellemző, hogy a felhasználó tevékenysége irányítja a folyamatokat azáltal, hogy kattint a felület egy pontján vagy legördít egy menüt vagy bejelöl egy választógombot stb., vagyis, a felhasználó tevékenysége különböző eseményeket, és arra történő válaszokat generál.

2.2.3 Eseményvezérelt (GUI) alkalmazások

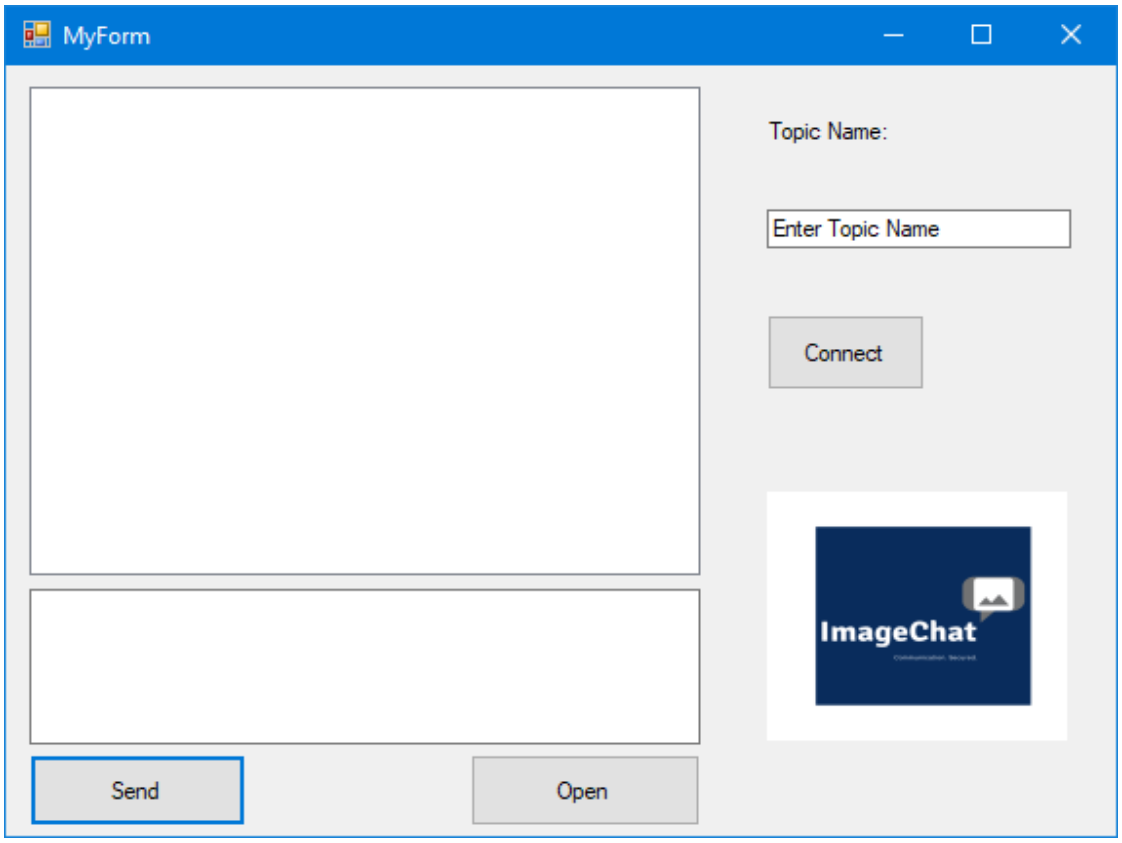
A GUI alkalmazások jellemzője, hogy a felhasználó határozza meg a program futásának menetét a beavatkozásaival. A felhasználó egy grafikus, akár több ablakból (űrlap, form) álló felületen keresztül kommunikál a programmal, s a rendelkezésére álló eszközökkel (billentyűzet, egér stb.) avatkozik be a program futásába, aminek jellemző viselkedése, hogy vár a felhasználó következő jelzésére, és arra reagál.

2.2.4 Események (Events)

Az eseményhez olyan kezelőfüggvényt vagy függvényeket kapcsolhatunk, amelyeknek meghívása az esemény kiváltásának hatására történik. A programozó feladata, hogy ilyen eseménykezelő metódust készítsen. Az eseménykezelő olyan kód, ami értesítést kap, ha egy esemény bekövetkezett. Megkapja az eseményt kiváltó objektumot (sender) és az esemény kezeléséhez szükséges valamennyi információt.

Az eseménykezelő metódusok visszatérési típusa mindig void és két paraméterük van, az első az eseményt kiváltó objektum (sender) (pl. a gomb), ez mindig Object típusú, a második pedig az EventArgs osztályból származtatott objektum, amely az esemény információit tartalmazza. Ez utóbbinak több specializált változata van (pl.: külön a billentyűzet és egér által kiváltott eseményekhez).

2.2.5 A projekt grafikus felülete:



2.2.6 Felhasználói specifikáció

- ✓ 1.lépés: A téma mezőbe be kell írni a téma nevét, majd a ***Connect*** gombra kattintva a következő függvény fog meghívódni: `ConnectClient();`
  A függvény megpróbál csatlakozni a szerverhez, átadva a kliens által beírt téma nevét, illetve amennyiben a csatlakozás sikertelen, hibát térít vissza.

- ✓ 2.lépés: Az ***Open*** gombra kattintva, az *OpenFile()*; függvény lesz meghívva, melynek következtében, ki kell választani egy fájlt, esetünkben egy képfájlt.

- ✓ 3.lépés: A ***Send*** gomb klikk eseményére meghívódik a *sendMsg(str);* függvény, paraméterként megkapja az úgynevezett üzenetmező tartalmát, majd felépítve a protokollt, “téma-üzenet”, elküldi a szervernek.

- ✓ 4.lépés: Az érkező üzenetek fogadására a getText(); függvényt használjuk. Várjuk az érkező adatcsomagot és sikeres fogadás esetén megjelenítjük a kapott üzenetet.

2.2.7 Funkcionális követelmények

- Téma megadása

Az alkalmazás felhasználójának meg kell adnia egy témakört, amit a Connect gomb lenyomása után már nem lehet módosítani.

- Csatlakozás

A témanév megadása után az első szükséges lépés a Connect gomb lenyomása, ami a szerverhez való csatlakozást teszi lehetővé.

- Fájl kiválasztása

Ahhoz, hogy az üzenetet egy képbe tudjuk kódolni, szükségünk van egy fájlra, esetünkben egy képfájl kiválasztására, melynek kiterjesztése lehet .jpg, .png, .bmp, stb.

- Üzenetküldés

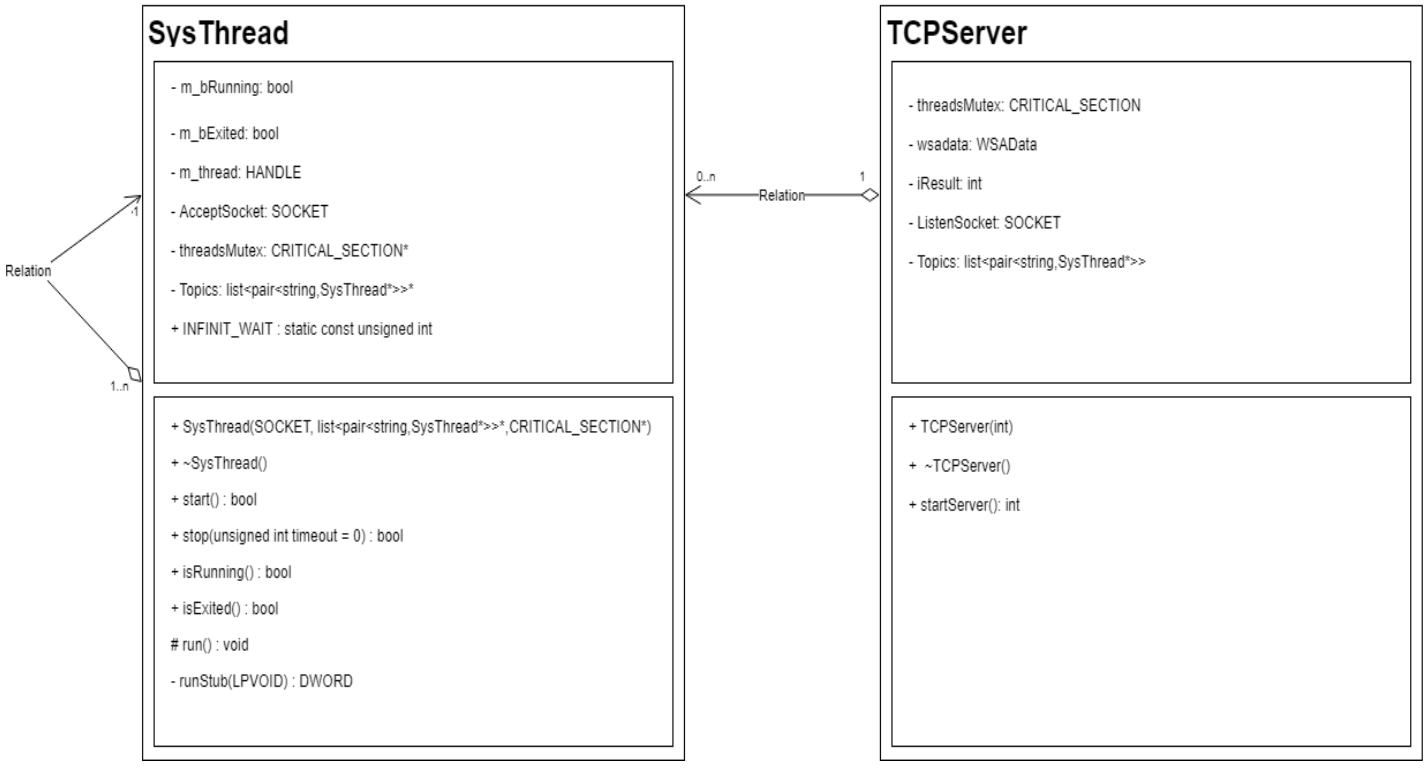
Miután megtörtént a kép kiválasztása, be kell írni az üzenetet a szövegdobozba. A küldött üzenet egy kódoló skript segítségével a képbe lesz kódolva, amit a fogadó függvényben lévő dekódoló láthatóvá tesz a kliens számára.

2.3 Szerver

A chat szerver egy konkurens TCP szerver, amely biztosítja a klienseknek a szükséges üzenetkezelést. A szerver az úgynevezett Publish/Subscribe típusú üzenetváltási architektúrára épült, melyben a klienseknek lehetőségük nyílik különböző témákra feliratkozni. A témák alatt beszélgetési csoportokat értünk, amelyen belül egyidőben több személy is küldhet képbe kódolt üzeneteket.

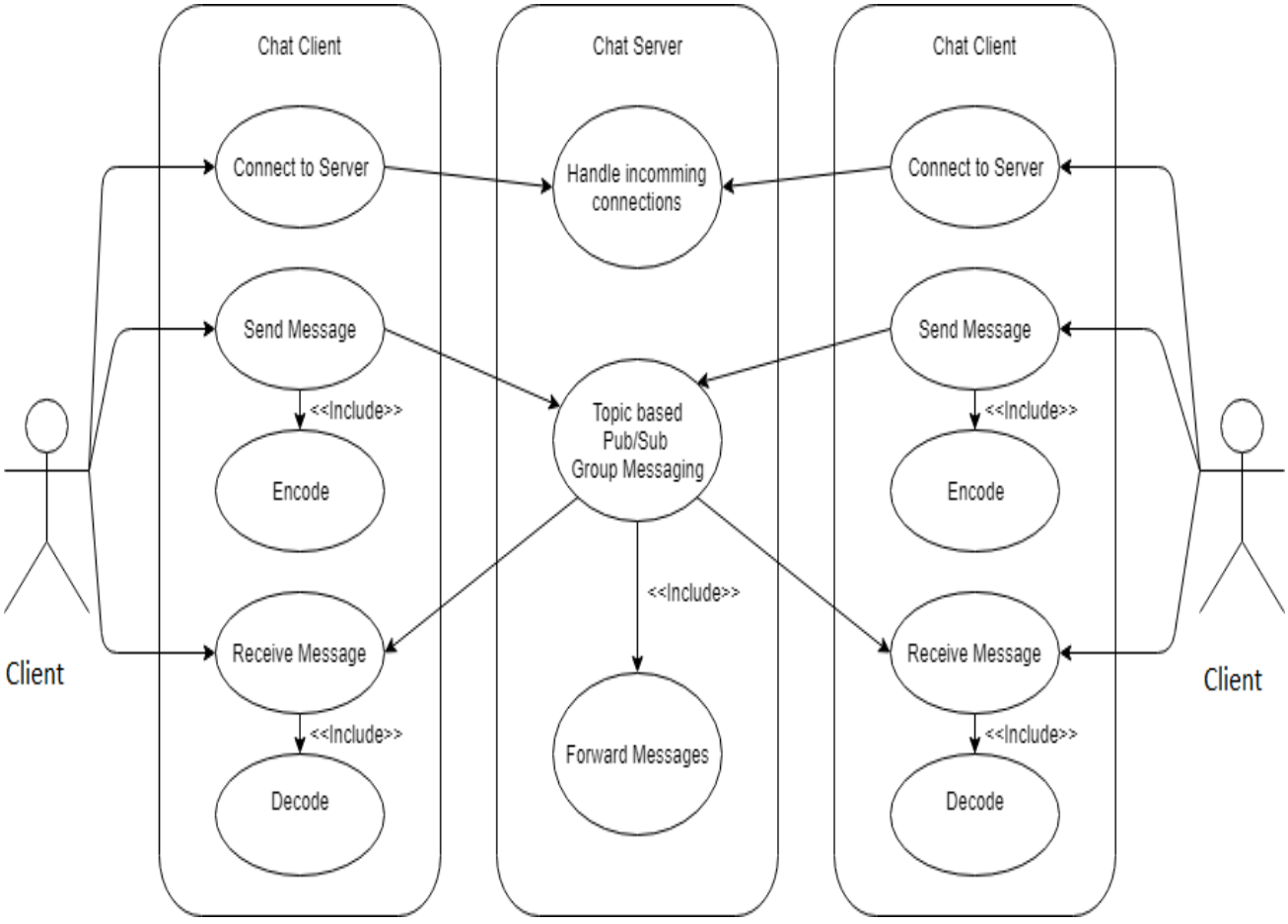
A szerver minden üzenet esetében egy képfájlt fogad a klienstől és ezt továbbítja azok számára, akik feliratkoztak a témára. Ez a fájlküldés úgy valósul meg, hogy a képfájlt adott hosszúságú darabokban fogadja a szerver.

2.3.1 Server class diagram

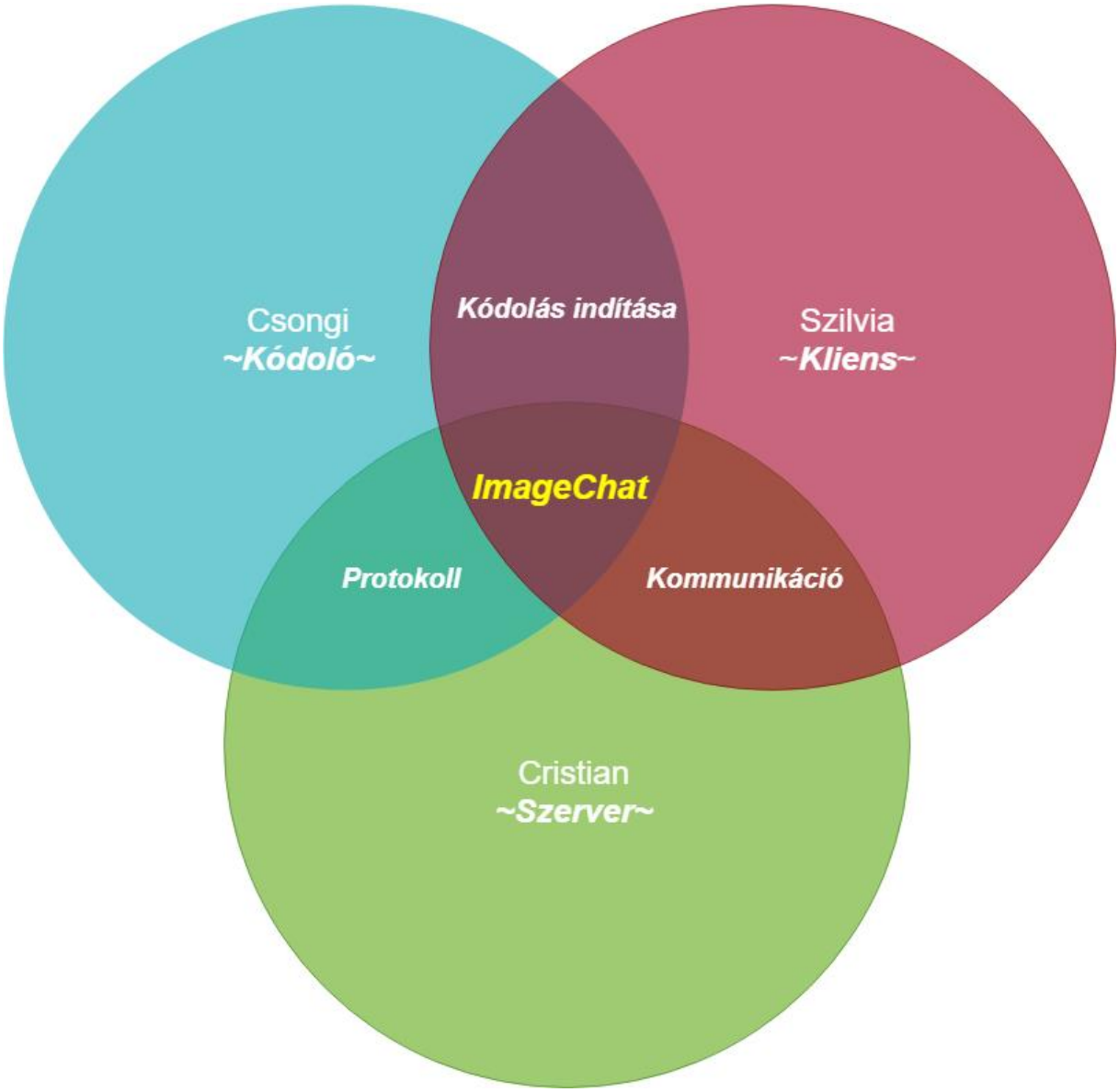


3 Szemléltető diagramok

3.1 Use case diagram



3.2 Komponens architektúra diagram



4 Nem funkcionális követelmények

- ✓ Python 3.7-es verzió.
- ✓ Python hozzáadva a rendszer PATH-hez
- ✓ Opencv 3.4.4, pip segítségével telepítve
- ✓ Microsoft Visual Studio 15.9.3
- ✓ CLR Windows Form a felülethez

5 Továbbfejlesztési lehetőségek

- ➢ A szöveg elkódolása egy adott algoritmus szerint
- ➢ Több színcsatorna használata egy szöveg kódolása esetében
- ➢ Felhasználói profil készítés a következő célokból:
  - Kódolási mód rögzítése
  - Név rögzítése
  - Egyéb felhasználói adatok rögzítése
- ➢ Többfunkciós grafikus felület

6 Bibliográfia

- https://stackoverflow.com/
- https://docs.python.org/3/
- https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_tutorials.html
- https://social.msdn.microsoft.com/Forums/vstudio/en-US/e6fbde42-d872-4ab3-8000-41ab22a4a584/visual-studio-2017-windows-forms?forum=winformsdesigner