



# Semestrální práce z KIV/UPS

## Hra TIC-TAC-TOE

Lukáš Runt (A20B0226P)

*lrunt@students.zcu.cz*

6. ledna 2024

# Obsah

<b>Obsah</b>	<b>1</b>
<b>1 Zadání</b>	<b>2</b>
1.1 Zvolená hra, programovací jazyky a protokol . . . . .	2
<b>2 Popis hry</b>	<b>2</b>
<b>3 Programátorská dokumentace</b>	<b>3</b>
3.1 Protokol . . . . .	3
3.1.1 Formát zpráv . . . . .	3
3.1.2 Stavy uživatele . . . . .	8
3.2 Server . . . . .	10
3.2.1 Seznam zdrojových souborů . . . . .	10
3.3 Klient . . . . .	11
3.3.1 Zdrojové soubory . . . . .	11
3.3.2 Použité knihovny . . . . .	12
<b>4 Uživatelská dokumentace</b>	<b>12</b>
4.1 Překlad . . . . .	12
4.2 Spuštění serveru . . . . .	12
4.3 Spuštění a ovládání klienta . . . . .	13
<b>5 Závěr</b>	<b>16</b>
<b>Seznam obrázků</b>	<b>16</b>
<b>Seznam tabulek</b>	<b>16</b>

# 1 Zadání

Vytvořte síťovou hru pro více hráčů s architekturou server-klient (1:N) určenou pro PC. Server bude vytvořen v jazycích C/C++ nebo jiném nízko úrovněovém jazyce. Klient bude implementován ve vysokoúrovňovém jazyce jako je například Java, Kotlin, Python a jiné. Síťová komunikace bude realizována transportními protokoly TCP nebo UDP.

Složitost projektu vyžaduje vhodnou strukturovanost kódu do modulů nebo tříd, důkladnou dokumentaci a stabilitu aplikací, včetně ošetření výjimek a prevence pádu aplikací. Aplikace musí být schopny obsluhovat nevalidní síťové zprávy a udržovat nějakou formu záznamu (log). Server musí být schopen paralelně obsluhovat několik herních místností, a být nastavitelný v počtu místností, limitu hráčů a síťové adrese a portu.

Klientská část musí implementovat grafické uživatelské rozhraní, umožňující zadání adresy a portu pro připojení k serveru. Hráči jsou identifikováni přezdívkou, a aplikace musí ošetřovat všechny uživatelské vstupy a informovat o stavu hry. Klient rovněž signalizuje nedostupnost serveru a protihráče.

Celé znění zadání je dostupné na stránce [1]:

<http://home.zcu.cz/~ublm/files/PozadavkyUPS.pdf>.

## 1.1 Zvolená hra, programovací jazyky a protokol

Hra: Tic-Tac-Toe

Server: C++

Klient: Python

Síťový protokol: TCP

## 2 Popis hry

Tic-Tac-Toe je strategická tahová hra pro dva hráče, která se hraje na čtvercovém herním poli o velikosti  $3 \times 3$  viz. obrázek 1. Hráči při hraní střídavě umisťují své herní symboly (křížky a kolečka) do prázdných herních polí. Cílem hry je vytvořit vodorovnou, svislou nebo diagonální řadu stejných symbolů. Hra končí v okamžik, kdy se jednomu z hráčů podaří vytvořit tuto řadu, tento hráč se stává vítězem, nebo jsou všechna hrací pole zaplněna, v tomto případě hra končí remízou [2].

X		O
	X	O
X	O	X

Obrázek 1: Ukázka hry

## 3 Programátorská dokumentace

### 3.1 Protokol

#### 3.1.1 Formát zpráv

Komunikace mezi serverem a klientem probíhá se stanoveným formátem zpráv. Každá zpráva má svou hlavičku, pro identifikaci, jaká operace se má provést, parametry, které nesou potřebné informace k operaci a ukončovací znak zprávy, v tomto případě enter (`\n`), který značí, že zpráva přišla celá a může se zpracovat. Jednotlivé parametry a hlavička je pak oddělena oddělovacím znakem `|`. Posílané zprávy jsou ve formátu:

```
<hlavička>|<parametr1>|...|<parametrN>\n
```

V následující tabulce 1 jsou vypsány všechny zprávy, které lze očekávat v síťové komunikaci. V levém sloupci jsou zprávy, které odesílá klient. V pravém sloupci jsou pak odpovědi od serveru na předchozí zprávy. Můžeme si všimnout, že některé zprávy mohou od klienta více druhů odpovědí, protože odpověď záleží na stavu, ve kterém se klient nachází a také na správnosti odeslané zprávy.

Klient	Server
LOGIN <přezdívka>	LOGIN <číslo_kódu>
PING	LOBBY
	WAITING <čas>
	GAME <str> <sou> <při> <tah> <pole1-N>
	RESULT <výs> <odv> <čas> <pole1-N> <win1-N>
START	WAITING <čas>
	GAME <str> <sou> <při> <tah> <pole1-N>
CANCEL	LOBBY
TURN <pole>	VALID <číslo_kódu>
	GAME <str> <sou> <při> <tah> <pole1-N>
	RESULT <výs> <odv> <čas> <pole1-N> <win1-N>
REMATCH <odpověď>	RESULT <výs> <odv> <čas> <pole1-N> <win1-N>
	RESULT <výs> <odv> <čas> <pole1-N> <win1-N>
	GAME <str> <sou> <při> <tah> <pole1-N>
	LOBBY
USERS	Vypíše do konzole a logu seznam uživatelů

Tabulka 1: Tabulka odeslaných a očekávaných zpráv

V následující části jsou podrobně popsány všechny zprávy obsažené v předchozí tabulce:

- LOGIN|<přezdívka>\n
  - Požadavek na přihlášení uživatele
  - <přezdívka>: Jméno uživatele
  - Komunikace: Klient → Server
  - Stav: Disconnected
- LOGIN|<číslo\_kódu>\n
  - Odpověď serveru na pokus přihlášení uživatele
  - <číslo\_kódu>: Jak dopadlo přihlášení
    - 0 - Přihlášení OK: Nový uživatel
    - 1 - Přihlášení OK: Existující uživatel, který není připojen
    - 2 - Přihlášení ERROR: Ve hře je uživatel se stejnou přezdívkou
    - 3 - Přihlášení ERROR: Přezdívka obsahuje nepovolené znaky
    - 4 - Přihlášení ERROR: Přezdívka je příliš krátká
    - 5 - Přihlášení ERROR: Přezdívka je příliš dlouhá

6 - Přihlášení ERROR: Server je plný (maximální počet uživatelů)

- Komunikace: Server → Klient
- Stav: Disconnected, Logged
- PING\n
  - Odeslání zprávy serveru pro zjištění stavu přihlášeného uživatele
  - Komunikace: Klient → Server
  - Stav: Logged, Waiting, In\_game, Result\_screen
- LOGGED\n
  - Reakce na PING → uživatel je v lobby
  - Komunikace: Server → Klient
  - Stav: Logged
- START\n
  - Požadavek na zahájení hry
  - Komunikace: Klient → Server
  - Stav: Logged
- CANCEL\n
  - Požadavek na ukončení hledání hry
  - Komunikace: Klient → Server
  - Stav: Waiting
- WAITING|<čas>\n
  - Reakce serveru na PING a START → uživatel čeká na hru
  - <čas>: Čas který uživatel strávil ve frontě (počet zpráv PING)
  - Komunikace: Server → Klient
  - Stav: Waiting

- GAME|<strana>|<soupeř>|<připojení>|<tah>|<pole1>|...|<poleN>\n
  - Reakce serveru na PING a TURN (pokud se nejedná o tah ukončující hru) → vrací stav hry
  - <strana>: Strana (symbol) na které hráč hraje
    - 0 - Hráčovi je přidělen symbol O
    - 1 - Hráčovi je přidělen symbol X
  - <soupeř>: Přezdívka soupeře
  - <připojení>: Stav připojení soupeře
    - 0 až 30 - Čekání na soupeře → čas ve vteřinách, po který je soupeř odpojen
    - 1 - Soupeř je připojen
    - 2 - Čekání na soupeře skončilo → konec hry
  - <tah>: Číslo aktuálního tahu
  - <pole\_i>: Stav hracího pole
    - 0 - FREE: Hrací pole je volné
    - 1 - X: Hrací pole je obsazené X
    - 2 - O: Hrací pole je obsazené O
  - Komunikace: Server → Klient
  - Stav: In\_game
- TURN|<pole>\n
  - Odeslání tahu uživatele
  - <pole>: Index hracího pole, kde byl tah proveden
  - Komunikace: Klient → Server
  - Stav: In\_game

- VALID|<číslo\_kódu>\n
  - Reakce na tah hráče, zda je tah validní → byl proveden
  - <číslo\_kódu>: Značí, zda byl tah validní, případně k jaké chybě došlo
    - 0 - Tah OK: Tah je validní
    - 1 - Tah ERROR: Tah mimo hrací pole (mimo indexy pole)
    - 2 - Tah ERROR: Hrací pole není volné
    - 3 - Tah ERROR: Tah hráče, který nehraje (není na žádné straně)
    - 4 - Tah ERROR: Tah hráče, který není na tahu
  - Komunikace: Server → Klient
  - Stav: In\_game
- RESULT|<výsledek>|<odveta>|<čas>|<pole1>|...|<poleN>|<pole\_výhry>\n
  - Reakce serveru na PING, TURN (pokud se jedná o tah ukončující hru) a REMATCH → vrací stav výsledkové obrazovky
  - <výsledek>: Výsledek hry
    - 1 - Výhra: Hráč vyhrál
    - 2 - Porážka: Hráč prohrál
    - 3 - Remíza: Nikdo nevyhrál, byla zaplněna všechna pole
    - 4 - Soupeř se odpojil: Hra skončila kvůli přerušení spojení jednoho z hráčů
  - <odveta>: Stav odpovědí hráčů na odvetu
    - 0 - Nikdo z hráčů neodpověděl
    - 1 - Protihráč nechce hrát znova
    - 2 - Protihráč chce hrát znova
    - 4 - Hráč čeká na soupeřovu odpověď
  - <čas>: Čas na odpověď uživatele, který ještě neodpověděl
    - 0 až 30 - Čas [s], který zbývá při čekání na uživatele, který ještě neodpověděl
    - 1 - Žádný z uživatelů neodpověděl
    - 2 - Čekání na odpověď skončilo, odвета není možná
  - <pole\_i>: Stav hracího pole
    - 0 - FREE: Hrací pole je volné
    - 1 - X: Hrací pole je obsazené X



2 - O: Hrací pole je obsazené O

- <pole\_výhry\_i>: Tři indexy výherních polí, pokud jeden z hráčů vyhrál. Pokud nikdo nevyhrál, tak výherní pole nejsou ve zprávě obsaženy
- Komunikace: Server → Klient
- Stav: Result\_screen
- REMATCH|<odpověď>\n
  - Odpověď hráče, zda chce se soupeřem odvetu
  - <odpověď>: Opověď hráče
    - 0 - Ne: Hráč nechce odvetu se soupeřem
    - 1 - Ano: Hráč chce odvetu se soupeřem
  - Komunikace: Klient → Server
  - Stav: Result\_screen
- USERS\n
  - Požadavek na vypsání všech uložených uživatelů
  - Komunikace: Klient → Server
  - Stav: Logged, Waiting, In\_game, Result\_screen

### 3.1.2 Stavy uživatele

Uživatel (klient) se může nacházet celkem v šesti stavech, respektive pěti stavech na serveru. Stavy připojeného klienta jsou následující.

#### DISCONNECTED

Stav, při kterém uživatel není připojen k serveru. Pokud je uživatel v tomto stavu, musí se nejprve přihlásit, aby se dostal do lobby nebo do jiného stavu, pokud již byl k serveru přihlášen.

#### LOGGED

Stav, kdy je uživatel připojen k serveru. V tomto stavu může uživatel spustit hledání hry nebo se odpojit od serveru.

#### WAITING

V tomto stavu uživatel čeká, dokud se nenajde další uživatel, se kterým je možno hrát hru. V tomto stavu může uživatel též zrušit hledání hry (příkaz CANCEL).

## IN\_GAME

Je stav, ve kterém se hraje samotná hra. V tomto stavu může uživatel dělat tahy, pokud je na řadě. Je zde též kontrolováno připojení uživatele, pokud je uživatel odpojen déle než 30 vteřin hra předčasně končí. Hra též končí, pokud je zaplněno celé hrací pole nebo některý z hráčů spojí tři symboly diagonálně, horizontálně nebo vertikálně. Ve všech případech je uživateli změněn stav na výsledkovou obrazovku.

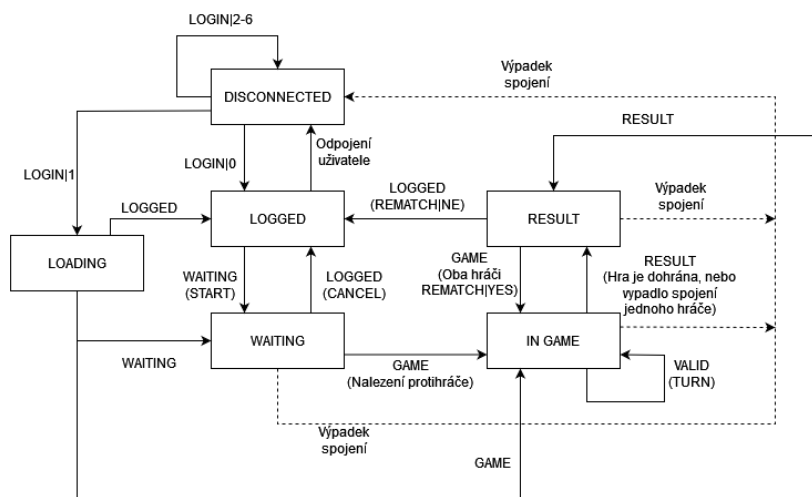
## RESULT\_SCREEN

Stav po dohrání hry. V tomto stavu se hráči rozhodují, zda chtějí odvetu se soupeřem nebo se vrátit do lobby. Pokud jeden z hráčů chce hrát znova, spustí časovač 30 vteřin, jinak se hráči vrátí do lobby.

## LOADING

Tento stav se nachází pouze na klientské straně, neboť se jedná o stav, při kterém se načítají data uživatele (doba při které uživatel čeká na odpověď serveru na stav uživatele), pokud již byl uživatel někdy přihlášen.

Následující obrázek 2 reprezentuje stavový diagram hry. Stavový diagram je z pohledu klienta. Popis přechodových hran bez závorek jsou zprávy přijímané klientem od serveru. V závorce jsou zprávy, které klient odesílá na server. Tečkovanou čarou je znázorněn nečekaný výpadek spojení.



Obrázek 2: Stavový diagram hry

## 3.2 Server

Server je naprogramován v jazyce C++. Program pracuje v jednom vlákne. Pro paralelizaci přijímání zpráv od různých klientů je použit `select`. Server je implementován staticky, to znamená, že odpovídá na zprávy uživatele, ale sám žádnou zprávu neposílá jako první.

### 3.2.1 Seznam zdrojových souborů

- **Server.cpp** - obsahuje funkci `main`, zpracovává vstupní parametry 4.2, spouští server, spravuje spojení s klienty, zpracovává příchozí zprávy od klientů, odpojuje klienty kteří se nechovají podle pravidel a odesílají nevalidní zprávy.
- **User.h/.cpp** - Třída reprezentující uživatele. Obsahuje stavový automat se stavy uživatele, stará se o provedení zpráv přijatých od klientů. Udrží informace o všech uživateli ve formě statického vektoru. Instance uživatele uchovává hodnoty o stavu, ve kterém se uživatel nachází, přiřazenému descriptoru, ukazateli na hru, přezdívce, času čekání a času poslední zprávy, podle které se určuje, zda je uživatel připojen. Po zpracování zprávy vrací třída **User.cpp** třídě **Server.cpp** zprávu, kterou má server odeslat klientovi jako odpověď. Při nevaliditě zprávy vrací třída **User.cpp** zprávu `ERROR`, která vede k odpojení uživatele od serveru.
- **Game.h/.cpp** - Třída reprezentující hru se stará o hru a stránku s výsledky, respektive o to, jestli chtějí uživatele odvetu. V instanci třídy je uložen stav hry, počet tahů a výherní pole, pokud nějaká jsou. Dále jsou uložena jména, a časy poslední zprávy uživatelů. Třída má za úkol validovat uživatelské tahy, vyhodnocovat stav hry, kontrolovat, zda jsou oba uživatelé připojení a také jestli mají zájem o odvetu. V této třídě se kontroluje připojení uživatelů pomocí časových značek. Vždy když uživatel odešle zprávu, uloží se čas zprávy. Pokud server od uživatele neobdrží zprávu déle než 1 s, vyhodnotí se, že má problém s připojením, tento stav se ukáže soupeřovi. Hráč má poté 30 s na to se vrátit do hry. Pokud se hráč nestihne připojit do 30 s od výpadku, hra končí s výsledkem `CONNECTION_LOST`. Časově je ošetřena i výsledková obrazovka s možností odvetu. Čeká se 30 s od první odpovědi jednoho z hráčů, poté odvěta není možná a hráči se musí vrátit do lobby.
- **Logger.h/.cpp** - Logger aplikace, zapisuje do konzole a do logovacího souboru, implementován s použitím návrhového vzoru `singleton`.

Umožňuje 4 úrovně logování, a to `ERROR`, `WARNING`, `INFO` a `DEBUG`. Log je ve formátu `<Úroveň><čas><zpráva>`.

- `Constants.cpp` - Obsahuje všechny konstanty, které server používá.
- `Enums.cpp` - Obsahuje všechny výčtové typy serveru.

### 3.3 Klient

Klient je napsán v jazyce Python verze 3.9.10. Pro uživatelské rozhraní byla použita knihovna `PyQt5` 3.3.2. Aplikace běží po připojení k serveru ve třech vláknech. Jedno vlákno se stará o interakci s uživatelem a zobrazení správného stavu aplikace. Druhé vlákno se stará o přijímání zpráv. Poslední vlákno má za úkol odesílání `PING` zprávy v časovém intervalu 250 ms. Pokud se server neozve 1 s, klient to vyhodnotí jako výpadek spojení, ukončí spojení se serverem, zobrazí chybovou hlášku a přesune uživatele na obrazovku s přihlášením k serveru.

#### 3.3.1 Zdrojové soubory

- `Client.py` - obsahuje funkci `main`, slouží ke spuštění aplikace
- `Constants.py` - obsahuje všechny konstanty aplikace
- `Enums.py` - obsahuje všechny výčtové typy aplikace
- `Heartbeat.py` - třída, která se stará o pravidelné zasílání `PING` zpráv. Dědí od třídy `Thread`, instance je tedy spuštěna jako samostatné vlákno, po připojení k serveru.
- `Logger.py` - Konfigurace a vytvoření instance loggeru s použitím Python knihovny `logging`
- `MainWindow.py` - Třída s oknem aplikace, stará se obsluhu hlavního okna, zpracovávání požadavků uživatele, zpracovávání příchozích zpráv a aktualizace stavu uživatele.
- `MessageBoxes.py` - Implementace metody vyskakovacího okna
- `Scenes.py` - Obsahuje definice vzhledu všech scén aplikace
- `Socket.py` - Stará se o síťovou komunikaci se serverem, obsahuje metody na připojení a odpojení od serveru, odesílání a přijímání zpráv.

### 3.3.2 Použité knihovny

#### PyQt5

Tato knihovna byla použita pro implementaci grafického uživatelského rozhraní. Tato knihovna je postavena na frameworku Qt, což znamená, že lze vytvářet moderní a efektivní aplikace s multiplatformní podporou. PyQt5 umožňuje vytvářet interaktivní a atraktivní uživatelské rozhraní pomocí widgetů, dialogů a dalších prvků. Knihovna se instaluje pomocí příkazu:

```
pip install PyQt5
```

#### PyInstaller

Knihovna pro vytváření spustitelných `.exe` souborů. Hlavní výhoda spustitelného `.exe` souboru je, že uživatel nemusí mít stažené potřebné knihovny ani Python. Tato knihovna se instaluje pomocí příkazu:

```
pip install pyinstaller
```

S pomocí knihovny byly vytvořeny skripty `Create_executable.bat` pro Windows a `Create_executable.sh` pro linux.

## 4 Uživatelská dokumentace

### 4.1 Překlad

Přeložení zdrojových souborů serveru probíhá zadáním příkazu `cmake` a `make` v kořenovém adresáři se souborem `CMakeLists.txt` a zdrojovými soubory.

Klientovi může být vytvořen spustitelný `.exe` soubor pomocí spuštění jednoho ze skriptů `Create_executable.bat` nebo `Create_executable.sh` dle operačního systému. Předpokladem je nainstalovaná knihovna `PyInstaller` viz. 3.3.2.

### 4.2 Spuštění serveru

Server se spouští příkazem:

```
./server -c<maximální_počet_hráčů> -p<port>
```

Parametry nejsou při spuštění povinné. Server lze též spustit jen s jedním nebo žádným parametrem. Při spuštění bez příslušných parametrů bude

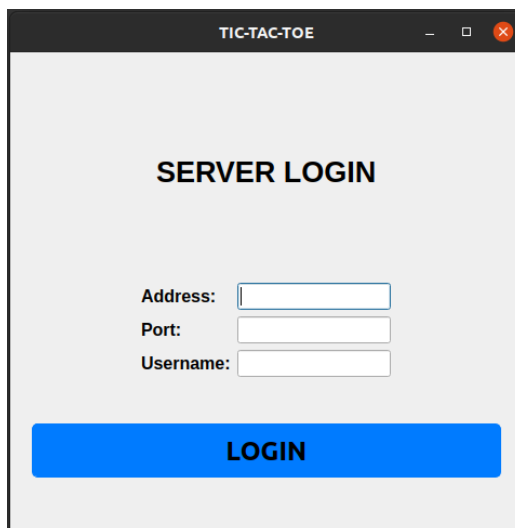
použito defaultně nastavených hodnot (port 10000 a maximální počet hráčů 10). Obě hodnoty jsou omezené, aby nedocházelo k zadání špatných hodnot. Port je omezen minimální hodnotou 1024 a maximální hodnotou 49151. Minimální počet hráčů je stanoven na 2, aby se dala odehrát hra.

### 4.3 Spuštění a ovládání klienta

Klient lze spustit dvojitým poklepáním na vytvořený `.exe` soubor, který obsahuje všechny potřebné knihovny ke spuštění, takže uživatel nemusí mít nainstalovaný Python ani jeho knihovny ve svém počítači. Aplikace se dá také spustit příkazem za předpokladu stažených potřebných knihoven 3.3.2:

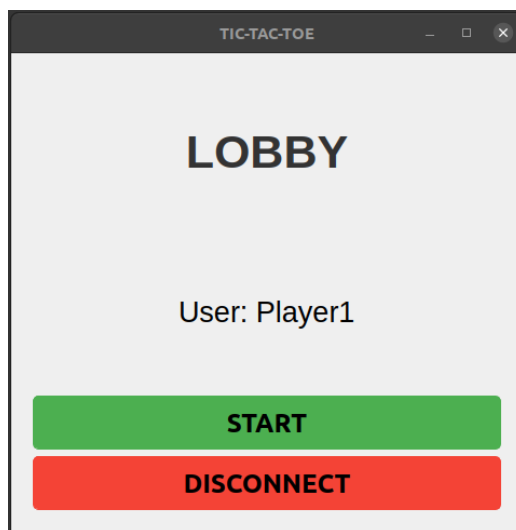
```
python Client.py
```

Při úspěšném spuštění souboru se zobrazí přihlašovací okno k serveru 3.



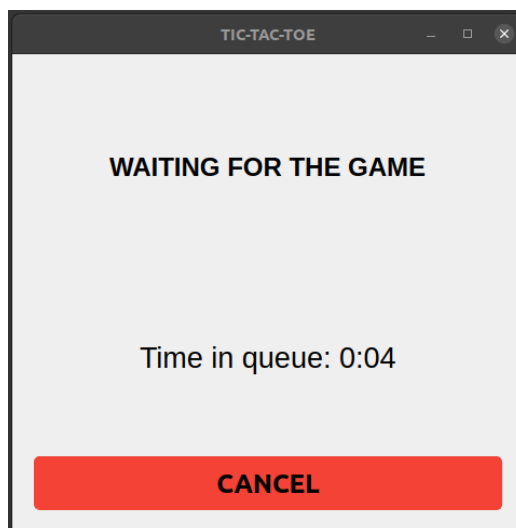
Obrázek 3: Aplikace ve stavu přihlášení k serveru

V tomto okně uživatel zadává adresu serveru. Port, na kterém server běží a své uživatelské jméno. Tlačítko `LOGIN` slouží k připojení serveru. Při úspěšném připojení je uživatel připojen do lobby aplikace 4. Při chybě je uživateli vypsána chybová hláška.



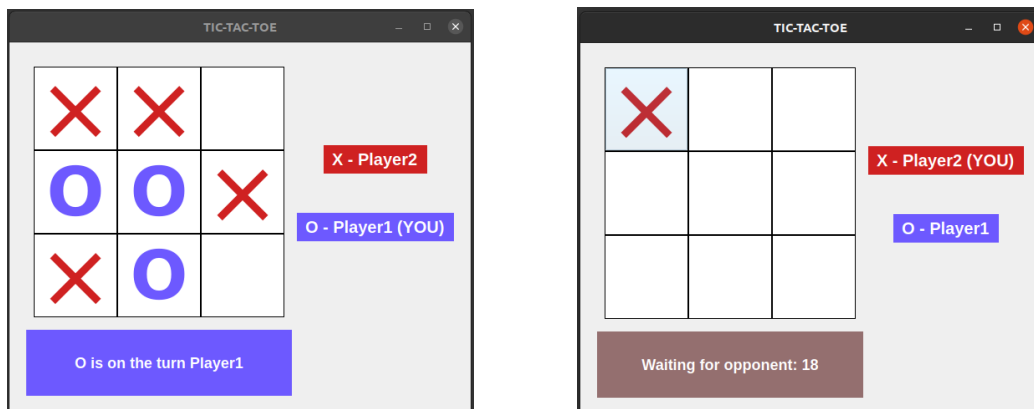
Obrázek 4: Lobby aplikace

Na této obrazovce je zobrazeno jméno přihlášeného uživatele a dvě tlačítka. Stisknutím tlačítka **START** se zahájí hledání hry 5. Odpojení do serveru nabízí tlačítko **DISCONNECT**, které vrací uživatele zpět na přihlašovací obrazovku 3.



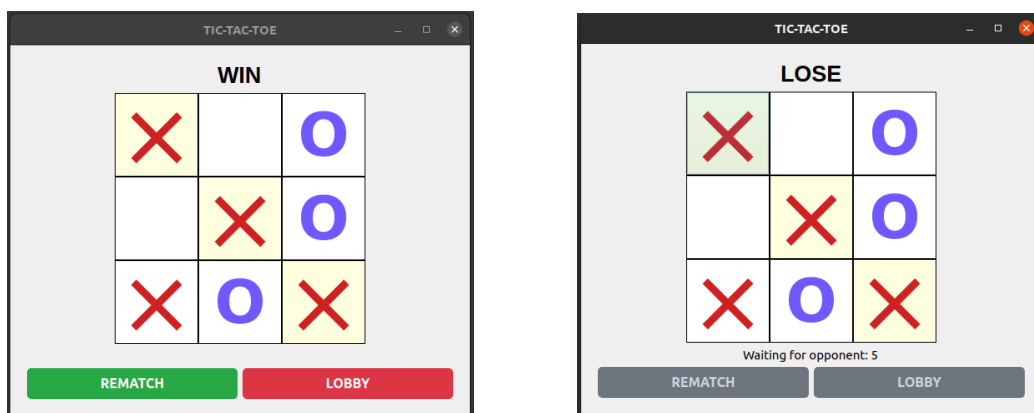
Obrázek 5: Obrazovka při čekání na hru

Obrazovka, kde uživatel čeká na hru ve frontě, zobrazuje uživateli čas strávený ve frontě. Stisknutím tlačítka **CANCEL** je hledání ukončeno a uživatel je vrácen zpět do lobby 4. Po nalezení hry je uživateli zobrazena scéna s hrou.



Obrázek 6: Zobrazení hry, vpravo ukázka reakce na odpojení protivníka

Obrazovka se hrou 6 obsahuje herní plochu  $3 \times 3$ , kde uživatel provádí své tahy kliknutím do příslušného herního pole. Pokud je tah validní zobrazí se hráčův symbol v poli, na které se kliknulo. Vedle hracího pole jsou zobrazena jména hráčů se symbolem, který příslušnému hráči patří. Pod hracím polem je napsáno, kdo je zrovna na tahu. V tomto poli se také zobrazuje odpojení protivníka se zbývajícím časem, který má protivník na vrácení do hry.



Obrázek 7: Obrazovka s výsledky, vpravo čekání na odpověď soupeře

Obrazovka s výsledky 7 se zobrazí v případě, kdy někdo vyhraje, hra skončí remízou, nebo vyprší čas na připojení odpojeného uživatele. Obsah obrazovky zahrnuje výsledný stav hry, hrací pole ve výsledném stavu se zvýrazněnými poli výherní trojice, pokud došlo k vítězství jednoho z hráčů. Pod herním polem s výsledky se nacházejí dvě tlačítka, zda uživatel chce hrát odvetu se stejným hráčem.



## 5 Závěr

V rámci semestrální úlohy byla vytvořena síťová hra pro více hráčů. Tato zkušenost mi nejen poskytla hlubší porozumění principům komunikace mezi klienty a serverem, ale také rozšířila mé dovednosti v oblasti detekce a ošetření nečekaných stavů, robustnosti a optimalizace síťových operací. Během tvorby této hry jsem se setkal s výzvami spojenými s ukládáním dočasných dat, minimalizací latence a zajištěním plynulého herního zážitku pro všechny hráče. Věřím, že získané znalosti budou cenným přínosem v budoucí kariéře v oblasti vývoje softwaru a síťových technologií. Celkově vzato mohu konstatovat, že vytváření této síťové hry bylo pro můj profesní a osobní rozvoj velmi obohacujícím zážitkem.

## Reference

- [1] Martin Úbl. *Zadání semestrální práce*. Zář. 2023. URL: <http://home.zcu.cz/~ublm/files/PozadavkyUPS.pdf>.
- [2] *Tic-Tac-Toe*. Čvn. 2021. URL: <https://en.wikipedia.org/wiki/Tic-tac-toe>.

## Seznam obrázků

1	Ukázka hry . . . . .	3
2	Stavový diagram hry . . . . .	9
3	Aplikace ve stavu přihlášení k serveru . . . . .	13
4	Lobby aplikace . . . . .	14
5	Obrazovka při čekání na hru . . . . .	14
6	Zobrazení hry, vpravo ukázka reakce na odpojení protivníka .	15
7	Obrazovka s výsledky, vpravo čekání na odpověď soupeře . . .	15

## Seznam tabulek

1	Tabulka odeslaných a očekávaných zpráv . . . . .	4
---	--	---