Szavazóalkalmazás
Fejlesztői dokumentáció

Tartalom

1. Specifikáció

Házi feladatom témájaként egy szavazórendszert választottam. A rendszer több felhasználós, egy multiplatform Qt-s kliensalkalmazásból és egy Linux-os szerveralkalmazásból áll, közöttük a kommunikációs TCP csatornán szöveges alapon zajlik.

Funkcionális követelmények

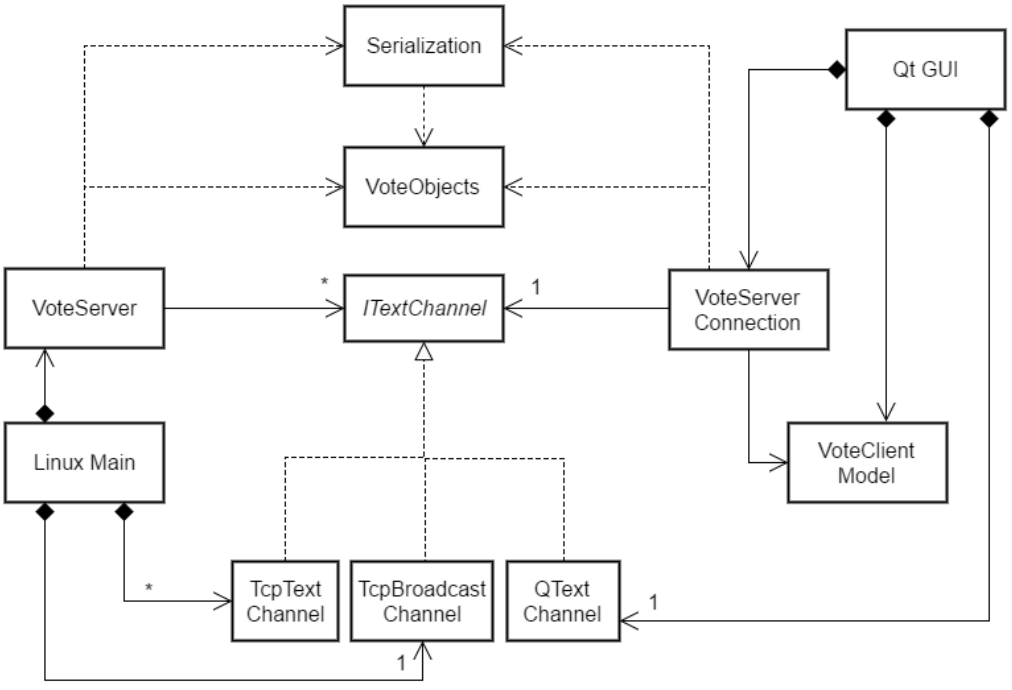
- szavazás indítása (címmel, leírással és opciókkal)
- szavazások listázása a főoldalon
- szavazás (opció kiválasztásával)
- szavazat állásának megtekintése
- szavazat törlése

Egyéb követelmények

- multiplatform kliens
- real-time interaktív működés

2. Architektúra

Az alkalmazás a következő architektúrával rendelkezik. Fontos szerepet kapott az egyes komponensek függőségeinek minimalizálása, egyszerű lecserélhetősége.



3. Kommunikáció

A szerver és a kliens alkalmazás egy megadott portú TCP socketen keresztül kommunikál egymással, egy általam definiált protokoll segítségével. A kommunikáció szöveges, egy parancsot egy sorban küldenek át. A parancs felépítése a következő:

{parancs sorszám} {parancs törzs objektum}

A c++-os törzs objektumok sorosítására egy kiterjeszthető keretrendszert dolgoztam ki. A sorosító minden objektumot sztringgé alakít oda és vissza. A következő típusokat a következőképpen sorosítom:

- int -> szám karakterenként decimális formátumban
- std::string -> az eredeti sztring idézőjelek közé fűzve
- std::vector<T> -> [a0 a1 a2 … an], ahol a0, a1, …, an a vektor elemei
- komplex objektum -> tagváltozók felsorolva szóközökkel elválasztva

Jól látszik, hogy nincsenek semmilyen típusjelölők, még a komplex objektumok szélei sincsenek jelölve, tehát a szerializált adatból nem lehet kitalálni a típusát, azt deszerializáláskor kell megadni, és a fejlesztő felelőssége, hogy helyesen adja meg.

A törzs objektum típusára a parancs sorszámából egyértelműen lehet következtetni. A következő parancsok és hozzá tartozó típusok vannak:

#	*név*	*törzs típus*	*küldő*	*leírás*
0	Login	int	user	Bejelentkezés kérése. Paramétere a felhasználói azonosító.
1	Register	-	user	Új felhasználói azonosító.
2	CreateVoting	Voting	user	Új szavazás létrehozása az átküldött objektum alapján.
3	DeleteVoting	int	user	Egy szavazás törlése id alapján.
4	VoteFor	Vote	user	Szavazás beküldése.
5	VotingChanged	VotingUpdate	server	Értesítés, hogy megváltozott a szavazás állása.
6	VotingCreated	Voting	server	Értesítés, hogy létrejött egy új szavazás.
7	VotingDeleted	int	server	Értesítés, hogy törlődött egy szavazás.
8	LoginOk	Votings	server	Válasz a bejelentkezésre. Tartalmazza az összes jelenlegi szavazást.
9	RegisterOk	Votings	server	Válasz a regisztrációra. Tartalmazza az összes jelenlegi szavazást és az új felhasználói azonosítót.

Az egyes saját struktúrák definíciói a voteobjects.h fájlban találhatóak.

4. Megosztott réteg

A szerializációért felelős komponens és a csatornán átküldött üzleti objektumok, és a csatorna absztakciója megosztottan a szerveren és a kliensen is jelen kell, hogy legyenek. Az ebben a fejezetekben leírt osztályok így mindkét helyen definiálva vannak.

itextchannel.h

Ez a fájl tartalmazza az ITextChannel interfészt. Ez az interfész felelős a hálózati szöveges kommunikációs csatorna elfedéséért. Célja, hogy a használt technológiákat könnyen cserélni lehessen. Több megvalósítása is van kliens és szerveroldalon. Az interfész a következőkből áll:

sendMessage(string data)	A data szöveget átküldi a csatornán.
messageReceived	Esemény, amely meghívódik, ha üzenet érkezett a csatornán keresztül.

voteobjects.h

Az hálózaton átküldött üzleti objektumokat és a parancsok azonosítóit tartalmazza. Az egyes objektumok felépítésére itt most nem térek ki, mert a kód alapján triviálisan megérthetőek.

serialization.h

Az itt leírt ser namespace tartalmazza az üzenetek törzsobjektumainak szerializálásához és deszerializálásához szükséges függvényeket. Az itt leírt függvények és struktúrák a következőek:

T Parser<T>:: parseMessageData(istream&)	Segédosztály, az egyes specializációi megvalósítják a streamről az objektumok deszerializálását.
T parseMessage<T>(string)	Visszaad egy T típusú deszerializált objektumot, amelyet a paraméterül kapott string-ből olvas be.
Composer<T>::composeMessageData (ostream, T)	Segédosztály, az egyes specializációi megvalósítják a paraméterül kapott objektumok stream-re szerializálását.
string composeMessage<T>(int, T)	Visszaad egy string-et, amelybe a paraméterül kapott parancskódot és törzs objektumot szerializálja.

5. Szerver

A szerver felelőssége, hogy a felhasználók csatlakozni tudjanak hozzá és az adataikat meg tudják vele osztani, illetve le tudják tőle kérdezni, így egymással is interakciót tudnak létesíteni. A szerver egy linux rendszeren kell, hogy fusson, a „sys/socket.h”-ban implementált hálózatkezelést használja. Felelőssége a modell nyílvántartása, a csatornák felépítése, és a TCP kapcsolatok kezelése.

voteserver.h

Ez az osztály felelős a szerver modelljének a megvalósításáért. Nem perzisztens, az adatokat csak a memóriában tárolja. Az alkalmazás egy ilyet kell, hogy tartalmazzon, ennek a műveletei fognak meghívódni a csatornákon érkezett üzeneteknek a hatására. Az osztály változóit az első, a függvényeit a második táblázat tartalmazza.

ITextChannel* broadcastChannel	A kommunikációs csatorna, amelyen keresztül minden felhasználónak képes üzenetet küldeni.
map<int, VotingModel> votings	A szerveren tárolt adatok nagy része. A szavazásokat tárolja id szerint. A VotingModel egy Voting objektumból (szavazás

	adatai) és egy <int, int> map-ből áll, amely az egyes felhasználók szavazatait tárolja. Tehát a változó a szavazásokat és a hozzátartozó szavazatokat tárolja.
int currentUserId	A következő regisztrációkor kiadott user id.
int currentVotingId	A következő szavazásnak kiadott id.

registerUserChannel (ITextChannel*)	Regisztrál egy kommunikációs csatornát, amelyet egy felhasználóval tart fenn. Itt kötöm be a csatornát az onMessageReceive függvényre. Az itt létrehozott lamda kifejezés tartalmazza a csatornához tartozó felhasználói azonosítót.
onMessageReceive (string&, int&, ITextChannel*)	Az beérkező üzeneteket dolgozza fel, osztja el a különböző függvények között.
loginOrRegister	Bejelentkezést vagy regisztrációt dolgoz fel, küldi el a szükséges választ. A válaszhoz összeállítja az összes szavazást és az új felhasználó saját szavazatait.
createVoting	Elmenti a rendszerbe a kapott szavazást és kiküldi mindenkinek.
deleteVoting	Törli a szavazást, és értesít mindenkit.
voteFor	Elmenti az adott szavazatot és a változásról értesít mindenkit.

tcpbroadcastchannel.h

A broadcast kommunikációt elvégző csatorna implementáció. A hivatkozott vektor socket-ein végigmegy és mindegyikre meghívja a linux-os send függvényt. Ez a csatorna adatokat nem fogad.

tcptextchannel.h

Egy felhasználóhoz tartozó TCP socket kommunikációs csatorna implementációja. Tárolja a socket azonosítóját és egy buffert (stemp), hogy soronként tudja értelmezni a beérkező adatokat a receiveMessage segédfüggvénye. Fontos, hogy ezeknek az osztályoknak a példányait nem lehet másolni, vagy módosítani.

main.cpp

A szerver alkalmazás kiindulási pontja. A következő két táblázat a szerver állapotát jelentő globális változókat és a segédvüggényeket írja le.

int ssock	A szerver socket leírója. Ebbe az állományba érkeznek az új kapcsolatok.
vector<pollfd> poll_list	Egy lista az aktív kapcsolatok állományleíróival. A poll fügvénynek lehet megadni paraméterként.
TcpBroadcastChannel tcpBroadcastChannel	A broadcast csatornát reprezentáló objektum.
map<int, TcpTextChannel> channels	Minden felhasználóhoz tartozó csatorna objektum, a kulcs az egyes socketek fájlleírója.
VoteServerModel server	A server modelljét tároló és a működést megvalósító objektum.

handle_new_connection	Fogadja az új kapcsolódási kéréseket, menti őket a leírók listájában, és létrehozza a hozzá tartozó ITextChannel-t és ezt regisztrálja a server-nek.
process_close	Lezár egy felhasználói kapcsolatot, törli a szükséges objektumokat.

process_read	Olvassa az adott kapcsolatról érkező adatot és átadja a neki megfelelő ITextChannel-nek.
main	Az alkalmazás belépési pontja. Folyanatosan polloz a felhasználó csatornák és az új kapcsolatkérések között és kezeli a bekövetkezett eseményeket.

6. Kliens

A rendszer kliens oldali részét egy Qt alkalmazásként valósítottam meg. Ennek a felelőssége eltárolni a felhasználónak szükséges adatokat és megjeleníteni azokat jól kezelhető formában. Ez a komponens három jól elkülöníthető részre bomlik: az alkalmazás modelljére (VoteClientModel), a szerverrel való kommunikáció megvalósítására (VoteServerConnection, QTextChannel) és a megjelenítésért felelős osztályokra (MainWindow, NewVotingDialog, VotingWidget).

voteclientmodel.h

Ez az osztály valósítja meg az alkalmazás modelljét. Lekérdezhetőek rajta a szükséges adatok illetve elvégezhetőek a szükséges módosítások. A felhasználói azonosítót és a felhasználónevet fájlban tárolja. Az osztály állapotát, eseményeit és interfészét a következő táblázatok írják le:

int myId	A felhasználó id-ja.
string myName	A felhasználó neve
map<int, VotingModel> votings	Id szerint rendezve az egyes szavazások, illetve a felhasználó által leadott szavazat rájuk.

votingAdded	Esemény, amely szavazás hozzáadása esetén tüzel.
votingRemoved	Esemény, amely szavazás eltávolítása esetén tüzel.
votingChanged	Esemény, amely szavazás módosítása esetén tüzel.

setUserId	Beállítja és menti a felhasználói azonosítót.
setMyName	Beállítja és menti a felhasználói nevet.
addVoting	Hozzáad egy szavazást.
removeVoting	Eltávolít egy szavazást.
updateVoting	Frissít egy szavazást.
setMyVote	Beállítja a felhasználó szavazatát.
getMyName	Visszaadja a felhasználó nevét.
getVoting	Visszaadja az id-vel megadott szavazást
getVotings	Visszaadja a rendszer összes szavazását (és a felhasználó szavazatait)-
registered	Visszaadja, hogy regisztrált e már a felhasználó.
userId	Visszaadja a regisztrált felhasználó azonosítóját.

voteserverconnection.h

Ez az osztály a szerverrel való kapcsolatot valósítja meg. A tagfüggvényei meghívásával különböző üzenetek küldhetőek, az ezekre érkezett válaszokban leírt módosításokat pedig végrehajtja a VoteVlientModel-en. Változói, eseményei és tagfüggvényei a következőek:

VoteClientModel* clientModel	A modellre referencia.
ITextChannel* serverChannel	A csatorna, amelyen keresztül a szerverrel kommunikál.

connectionCreated	Bejelentkezés vagy regisztráció után bekövetkező esemény.

startConnection	Elvégzi a bejelentkezést vagy a regisztrációt.
createVoting	Elküldi a szervernek a létrehozandó szavazást.
deleteVoting	Elküldi a szervernek a törlendő szavazás id-jét.
vote	Elküldi a szervernek a szavazatot.
onMessageReceive	A csatornáról érkező üzeneteket fogadja. Az üzenetek azonosítói alapján parse-olja a törzs objektumokat, majd elvégzi a szükséges műveleteket a modellen.
finishLogin	Segédfüggvény a bejelentkezésre és regisztrációra érkező válaszok lekezelésére.

qtextchannel.h

A tic-tac-toe hálózati alkalmazás socket kezelésének segítségével elkészített Qt-s TCP csatorna implementáció. Ezt fogja a megjelenítés létrehozni és tárolni, és ezen keresztül fognak az adatok a szerver és a VoteServerConnection között utazni.

Qt ablakok

mainwindow.h

Ez az alkalmazás főablaka. Itt tárolom változókban a kliens modellt és a szerverrel való kapcsolatot. Itt valósítom meg a menüelemek eseménykezelőit. Itt valósítom meg a modell eseménykezelőit, változás esetén a megváltozott szavazás megjelenítését frissíti, kezdetben az összes szavazást betölti. A szavazások megjelenítése az ablak közepén lévő QListWidget-ben találhatóak.

newvotingdialog.h

Egy új szavazást lehet létrehozni ezzel a dialógusablakkal. A getVoting függvénye a formjainak megfelelő szavazást adja vissza, ez az ablak modellje.

newvotingwidget.h

Ez egy saját widget, ami egy szavazás adatainak a megjelenítéséért felelős. Ilyeneket tartalmaz a főoldalon található lista. Társítható hozzá egy szavazás és egy szavazat. A VoteServerConnection társításának segítségével a szavazat változása esetén képes elküldeni a szavazat üzeneteket a szervernek.