VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

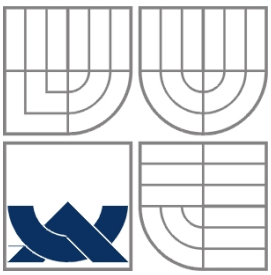
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

# PLATFORMA PRO KARETNÍ HRY NAD XMPP

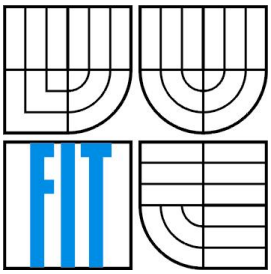
**BAKALÁŘSKÁ PRÁCE**
BACHELOR'S THESIS

AUTOR PRÁCE Ondřej Pták
AUTHOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

# PLATFORMA PRO KARETNÍ HRY NAD XMPP

PLATFORM FOR CARD GAMES OVER XMPP

**BAKALÁŘSKÁ PRÁCE**

BACHELOR'S THESIS

AUTOR PRÁCE — Ondřej Pták
AUTHOR

VEDOUCÍ PRÁCE — Ing. Schmidt Marek
SUPERVISOR

BRNO 2010

# Abstrakt

Tato práce obsahuje návrh univerzální platformy, kterou lze využít při tvorbě libovolné klasické karetní hry a její konkrétní implementaci. Součástí je klient ve formě pluginu do jabbim klienta, platforma pro tvorbu herních serverů, komunikační protokol postavený nad XMPP a jedna ukázková hra („prší“), která demonstruje možnosti platformy a snadnost implementace her.



# Abstract

This thesis includes a design of a universal platform for creation of any classic card game and it's implementation. Parts of this thesis are: a client as a jabbim plugin, a platform for creating a card game server, communication protocol built over XMPP, one card game („mau mau“). This game shows the possibilities of the platform and the easiness of developing games.



# Klíčová slova

XMPP, jabber, jabbim, jcards, karetní hry, platforma, PyQt, python, xml

# Keywords

XMPP, jabber, jabbim, jcards, card games, platform, PyQt, python, xml


# Citace

Ondřej Pták:, Platforma pro karetní hry nad XMPP, bakalářská práce, Brno, FIT VUT v Brně, 2009

# Platforma pro karetní hry nad XMPP

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Marka Schmidta.
Další informace mi poskytli vývojáři jabbim klienta.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Ondřej Pták
17. 5. 2010

## Poděkování

Děkuji vedoucímu práce, Ing. Markovi Schmidtovi, za dobré nasměrování vždy, když jsem s obtížemi hledal a volil cestu dalšího postupu. Jeho nápady a podněty mi umožnily zlepšit návrh s ohledem na specifické požadavky některých karetních her.

© Ondřej Pták, 2010

*Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.*

# Obsah

# 1 Úvod

V dnešní době se vyskytuje nepřeberné množství herního software, které láká stále více lidí. Hrát můžeme proti reálnému člověku, nebo umělé inteligenci herního programu. Dle mého názoru většina hráčů považuje za zajímavější hru proti reálnému protihráči. Existují hry pro více hráčů přes internet a některé i přes instant messaging klienty (viz kapitola 2.2), ovšem neexistuje obecná platforma pro různé karetní hry pro více hráčů.

Bylo by příjemné mít možnost si zahrát s přáteli po internetu nejen jednu konkrétní, ale libovolnou karetní hru. Je tedy třeba navrhnout platformu, která umožní hru několika hráčů přes XMPP protokol a zároveň bude snadno rozšiřitelná o definice pravidel dalších her. Použití XMPP protokolu bude výhodné z důvodu otevřenosti a snadné rozšiřitelnosti.

## 1.1 Motivace

Díky otevřenosti tohoto protokolu vznikla řada vylepšení a nadstaveb, které jeho funkcionalitu posouvají dále než je pouhá textová komunikace. Uživatelé si mohou na jabber server ukládat data, prohlížet galerie obrázků, vidět co kamarádi poslouchají, překládat mezi různými jazyky nebo používat nějakou z mnoha bran do ostatních sítí jako třeba ICQ, sms na mobilní telefony, Gadu Gadu, MSN a mnoho dalších.

Velkou výhodou je možnost skupinového rozhovoru. Myslím, že mnozí ocení online karetní hru, při níž mohou s přáteli vzájemně diskutovat.

Hrát pořád stejnou hru však rychle omrzí. Mnou navrhovaná platforma by proto měla nabízet možnost hraní karetních her a také vytváření nových. Bude možné příjemně strávit volné chvíle při partičce karetní hry s vašimi přáteli, ať se nacházejí kdekoli na světě. Stačí se připojit k internetu a mít nainstalovaný jabbim s pluginem jcards, jak se tato platforma a její klient budou jmenovat.

## 1.2 Cíl projektu

Vytyčil jsem si jako hlavní cíl vytvoření platformy, která umožní přes jabber/XMPP hraní libovolných klasických karetních her. Psaní konkrétních pravidel by mělo být co nejjednodušší a klientská část multiplatformní. Jedním z cílů je také jednoduché, pohodlné a intuitivní ovládání.

Serverovou část zamýšlím jako soubor dílčích prostředků, pomocí kterých bude možné realizovat každou klasickou karetní hru. Ty by měly obsahovat zobecněné entity jako karta, balík karet, přesun či hrací plocha. Mám představu, že programátor pouze nastaví specifické části pravidel a ostatní zařídí platforma.

# 2 Teoretická východiska

V této kapitole se seznámíme s teoretickými východisky potřebnými pro pochopení a tvorbu univerzální herní platformy přes XMPP. Jsou zde v krátkosti zmíněny karetní hry, instant messaging, informace o jazyku XML, protokolu XMPP a jeho rozšířeních, jabberu, programovacím jazyku Python a knihovně Twisted.

## 2.1 Karetní hry

Karetní hry jsou velmi staré a vznik historicky těžko dohledatelný kvůli příliš strohým informacím, které se dochovaly. Lze je rozdělit na klasické a moderní. Klasické se hrají s tradičními balíky hracích karet (whistové, rummy, kanastové a piketové karty). Tyto balíky se používají pro mnoho rozličných her. Oproti tomu moderní karetní hry (například Bang!) mají obvykle vlastní sadu hracích karet.[11]
V této práci jsem se zaměřil na tradiční karetní hry a to ze dvou hlavních důvodů. Je jich více a pro moderní karetní hry obvykle existuje specifický program s optimalizovaným ovládáním pro konkrétní hru.
Nyní zmíním obecné vlastnosti klasických karetních her (čerpáno z [1],[2],[3]).
K základní skladbě každé hry náleží především následující kroky, ale ne vždy jsou přítomny všechny:

1. rozsazení hráčů
2. snímání karet
3. rozdání
4. dražba
5. sehrávka
6. počítání konečného skóre

## 2.2 Instant messaging

Instant messaging je forma internetové služby, která umožňuje dvěma či více uživatelům komunikovat v (téměř) reálném čase. Příkladem mohou být Jabber/XMPP, ICQ/OSCAR, MSN či Gadu Gadu. Dochází také k minimalizaci časového rozdílu mezi odesláním a přijetím zprávy (řádově milisekundy až sekundy). Instant messaging je také méně vyrušující než třeba telefon, neboť uživatele nikdo nenutí, aby na zprávy odpovídal hned. Výhodou z hlediska informací je také nastavení naší přítomnosti, resp. nepřítomnosti. Více informací je možné nalézt například na wikipedii [9].

## 2.3 XMPP

XMPP je zkratka pro *Extensible Messaging and Presence Protocol*, neboli rozšiřitelný protokol pro posílání zpráv a zobrazení stavu. XMPP je standardizovaný, otevřený komunikační protokol založený na zasílání XML zpráv.

### 2.3.1 Historie

XMPP započal svůj vývoj v roce 1998 a do dnešní podoby byl poměrně složitý, proto považuji za vhodné uvést alespoň základní charakteristické prvky historie tohoto protokolu. [21]

V roce 1998 začal Jeremie Miller pracovat na projektu Jabber. V roce 2000 byla vydána první verze severu nazvaná Jabberd. Komunita jabberu se tomuto serveru začala věnovat a v roce

2004 došlo ke konečné standardizaci. Byly definovány 4 standardy : Core - RFC 3920 [16] základní prvek popisující elementární záležitosti a dále RFC 3921[17] , RFC 3922 [18] a RFC 3923 [15].

### 2.3.2 Výhody a nevýhody

Mezi hlavní výhody protokolu XMPP bezesporu patří otevřenost a decentralizace.

Otevřenost protokolu umožňuje komukoli implementovat programy využívající tento protokol, případně vytvořit vlastní rozšíření XMPP (XEP). Díky tomu je dnes k dispozici mnoho XMPP klientů. Výběr klienta je pouze osobní volbou, kdy se uživatel může rozhodnout pro některého dříve vydaného, či si naprogramovat vlastního.

Decentralizace protokolu znamená, že neexistuje jeden centrální server, ale každý si může vybrat podle nabízených služeb nebo si vytvořit vlastní. Tvorba vlastního serveru je užitečná například ve firemním prostředí, kde lze komunikaci omezit např. pouze na zaměstnance.

Další výhodou tohoto protokolu je bezpečnost. XMPP servery mohou být izolovány od veřejných XMPP sítí. Při komunikaci se serverem lze použít šifrování SSL/TLS. Je také možné využít certifikační autoritu a xmpp.net. Zabezpečení komunikace je možné zvýšit šifrováním jednotlivých zpráv PGP klíčem.

Mezi nevýhody patří velká režie na presence zprávy, kdy tyto zprávy tvoří přibližně 70% komunikace. Velké množství dat se také přenáší při posílání presence zpráv více účastníkům najednou. Druhou nevýhodou je neefektivní přenos binárních dat. To je dáno textovou formou XML jazyka. Pokud je nutné přenášet binární data, používá se kódování Base64. Tímto kódováním se objem dat navýší přibližně o třetinu.

### 2.3.3 Komunikace

Komunikace protokolu je založená na zasílání XML zpráv.

#### XML zprávy

Existují tři druhy zpráv, které však mohou obsahovat další vnořené XML elementy.

- *message* - tímto typem zpráv je přenášena běžná komunikace
- *precence* - typ zprávy nesoucí informaci o dostupnosti uživatele
- *iq* - typ zprávy vhodný pro stavovou komunikaci, podobně jako HTTP protokol. Název pochází z anglického dotaz/odpověď. Umožňuje nastavení či získání parametrů. Po iq dotazu vždy následuje odpověď nebo chybová hláška.

#### JID

JID (Jabber ID) slouží k jednoznačné identifikaci uživatele zprávy. Jabber ID je členěním podobný emailové adrese. Obsahuje dvě povinné části oddělené pomocí znaku „@“. Před „@“ se nachází jméno uživatele, za tímto znakem pak adresa serveru, ke kterému se daný uživatel připojuje. JID však může obsahovat ještě další parametr, tzv. zdroj. V Jabber ID následuje za serverem, od kterého je oddělen pomocí „/“. Tento zdroj může sloužit k vícero účelům - například identifikace připojení, pokud má uživatel spuštěno více klientů nebo může sloužit k určení místa, kde se uživatel nachází (doma, v praci,...).

### 2.3.4 XEP

XMPP Extension Protocol, zkráceně XEP [24], je rozšíření XMPP protokolu o nové funkce. Dříve se používalo označení JEP - Jabber Enhancement Proposal. Pro zajímavost zmíním několik rozšíření XEP.

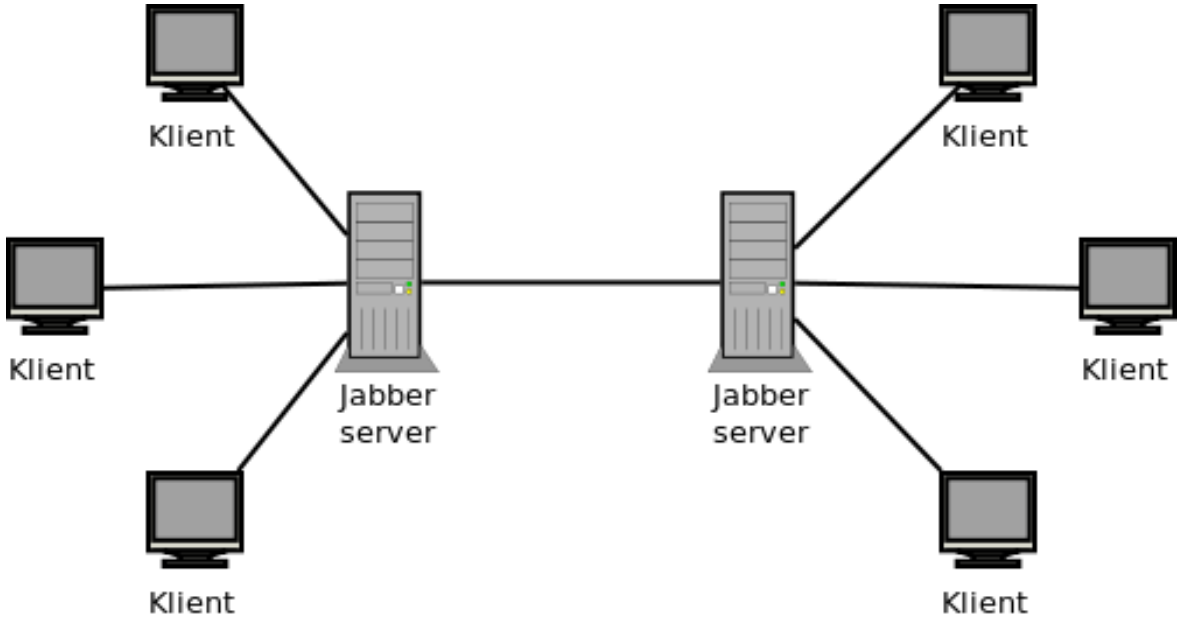
- XEP-0045 - Multi-User Chat [19]
  Definice rozhovoru pro více účastníků.
- XEP-0084 - User Avatar [20]
  Rozšíření pro výměnu avatarů (malých obrázků)
- XEP-0136 - Message Archiving [14]
  Archivace XMPP komunikace na serveru.
- XEP-0166 – Jingle [7]
  Peer-to-peer multimediální spojení. Je použitelné například pro hlasový či video rozhovor, nebo přenos souborů.

### 2.3.5 Jabber

V současné době se pojmem Jabber označují různé věci, od protokolu po komunikační síť.

Jabber wikipedia [10] definuje Jabber jako platformu využívající protokol XMPP, přičemž bývá často použit jako synonymum k XMPP protokolu rozšířeného o XEPy. [24]

Síť jabber se skládá ze serverů, které nabízejí služby a klientů, kteří se připojují k různým serverům. Dva klienti spolu komunikují přes svůj jabber server. Pokud je připojen každý uživatel k jinému serveru, tyto servery komunikují mezi sebou (obr. 2.1). Tuto komunikaci lze upravit na serveru, například zakázání určité domény, nebo naopak omezení komunikace pouze na vlastní doménu, například firemní jabber server kde mohou komunikovat s uživateli jiných serverů třeba jen někteří.




*obr 2.1: architektura jabberu*

## 2.4 Python

Tato podkapitola je volně převzata z wikipedie[13].

Python je dynamický objektově orientovaný programovací jazyk. Navrhl jej v roce 1991 Guido van Rossum. Python je vyvíjen jako open source projekt, který zdarma nabízí instalační balíky pro většinu běžných platforem (Unix, Windows, Mac OS). Ve většině distribucí systému Linux je Python součástí základní instalace.

### 2.4.1 Vlastnosti

#### Dynamický jazyk

Python je dynamický interpretovaný jazyk. Někdy bývá řazen mezi takzvané skriptovací jazyky. Jeho možnosti jsou ale větší. Python byl navržen tak, aby umožňoval tvorbu rozsáhlých, plnohodnotných aplikací (včetně grafického uživatelského rozhraní — viz například wxPython, PyQt).

#### Hybridní jazyk

Python je hybridní jazyk (nebo také víceparadigmatický), to znamená, že umožňuje při psaní programů používat nejen objektově orientované paradigma, ale i procedurální a v omezené míře i funkcionální, podle toho komu co vyhovuje nebo co se pro danou úlohu hodí nejlépe. Python má díky tomu vynikající vyjadřovací schopnosti. Kód programu je ve srovnání s jinými jazyky krátký a dobře čitelný.

#### Jednoduché učení

Bývá dokonce považován za jeden z nejvhodnějších programovacích jazyků pro začátečníky. Tato skutečnost je dána tím, že jedním z jeho silných inspiračních zdrojů byl programovací jazyk ABC, který byl jako jazyk pro výuku a pro použití začátečníky přímo vytvořen. Python ale současně boural zažitou představu, že jazyk vhodný pro výuku není vhodný pro praxi a naopak. Podstatnou měrou k tomu přispívá čistota a jednoduchost syntaxe, na kterou se při vývoji jazyka hodně dbá.

#### Produktivita

Velkou výhodou pythonu je produktivita při programování. Týká se to nejjednodušších programů i aplikací velmi rozsáhlých. U jednoduchých programů se tato vlastnost projevuje především stručností zápisu. U rozsáhlých aplikací je produktivnost podpořena rysy, které se používají při programování velmi často, jako jsou například přirozená podpora jmenných prostorů, používání výjimek, standardně dodávané prostředky pro psaní testů (unit testing) a další. S vysokou produktivností souvisí dostupnost a snadná použitelnost široké škály knihovních modulů, umožňujících snadné řešení úloh z řady oblastí.

#### Vkládání do jiných aplikací

Python se snadno vkládá do jiných aplikací (embedding), kde pak slouží jako jejich skriptovací jazyk. Tím lze aplikacím psaným v kompilovaných programovacích jazycích dodávat chybějící pružnost. Jiné aplikace nebo aplikační knihovny mohou naopak implementovat rozhraní, které umožní jejich použití v roli pythonovského modulu. Jinými slovy, pythonovský program je může využívat jako modul dostupný přímo z jazyka Python (tj. extending, viz sekce Spolupráce s jinými aplikacemi).

Programování v Pythonu klade velký důraz na produktivitu práce programátora. Myšlenky návrhu jazyka jsou shrnuty ve filosofii Pythonu.

### Výkon

Výkon aplikací napsaných v Pythonu je dobrý, protože výkonově kritické knihovny jsou implementovány v jazyce C, s kterým Python výborně spolupracuje. I samotný jazyk je na tom v porovnání s jinými interpretovanými jazyky dobře. Je např. 3 až 5 krát rychlejší než PHP. Pro Python navíc existuje snadno použitelná knihovna Psyco, která transparentně optimalizuje kód Pythonu na výkon. Některé operace jsou pomocí Psyco urychleny až řádově.

## 2.5 Twisted

Twisted [23] je knihovna napsána v jazyce python. Jejím předchůdcem je XMPPPY. Podporuje práci s mnoha internetovými protokoly, jako třeba http nebo různé IM protokoly. Poskytuje pohodlné rozhraní pro práci s XML dokumenty a streamy, čehož ve své práci využiji.

### Words

Words je modul Twisted pro podporu instant messagingu. Words podporuje mnoho protokolů jako je například XMPP, IRC, ICQ a další.

### Domish

Jedná se o součást modulu Twisted Words pro práci s XML. Umožňuje snadné vytváření, procházení a úpravu XML elementů.

### Internet

Tento modul poskytuje připojení k internetu pomocí různých protokolů.

## 2.6 Jabbim klient

Celý XMPP klient jabbim je napsán v pythonu s využitím grafické knihovny PyQt. Jabbim je určený především pro začátečníky. Je přehledný, uživatelsky přívětivý a lze jej rozšiřovat pomocí zásuvných modulů. Jedním takovým pluginem bude klientská část této práce.

# 3 Současný stav

XMPP protokol umožňuje přidávání funkcionality různými rozšířeními [24]. Existují například rozšíření k přenosu souborů, hromadnému chatu (muc), společné kreslící ploše (whiteboard) nebo VOIP (jingle).

Z dostupných informací vyplývá, že v současné době neexistuje žádný projekt určený přímo pr o karetní hry. Zajímavým podobným projektem je jgames, též plugin jabbim klienta. Pro ukázku stručný popis některých rozšíření:

### Jgames

Jgames je zatím nedokončený experiment, který v aktuálním stavu zvládá piškvorky. Klient slouží pouze k zobrazování tahu. Ke své funkci potřebuje serverovou komponentu, která se stará o pravidla. Ke komunikaci s herním serverem využívá XMLRPC přes XMPP. Jgames umožňuje sledování hry, rozhovor pomocí MUC a prohlížení probíhajících herních záznamů. Momentálně se vyvíjí pro tuto platformu hra Kasíno. V budoucnu bude jgames možná fungovat i pod javascriptovým klientem a umožní tak hru přes webové rozhraní.

### Jabber disk manager (JDM)

Jde o pohodlné rozhraní k službě jabber disk, která ukládá na serveru data pomocí přenosu souborů. Data jsou ukládána jako veřejná (pro všechny) nebo privátní (nikdo jiný je nevidí). JDM nabízí místo příkazů posílaných jabber disk serveru rozhraní s ikonami, drag and drop přesuny a potřebnými akcemi v menu.

### Whiteboard [4]

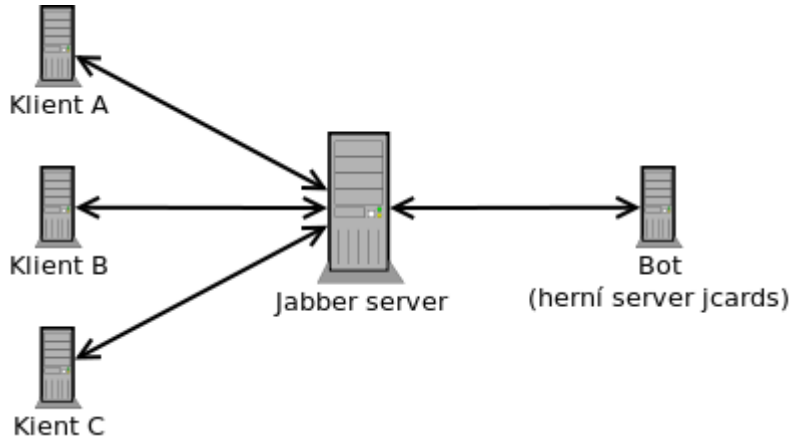
Toto rozšíření XMPP (použité např. V klientu Coccinella) nabízí kreslící plochu sdílenou dvěma a více uživateli. Každý může upravovat obrázek nezávisle na ostatních a každá změna se projeví u všech uživatelů. Grafika je převážně vektorová (využívá svg formát), ale je možné vkládat i bitmapy a text. Whiteboard ocení kreativní lidé, kteří chtějí společně tvořit, vývojáři k různým diagramům, které ihned konzultují, nebo každý komu slova nestačí.

# 4 Návrh

Obecně jsem si vytyčil za cíl vytvořit co nejobecnější platformu, multiplatformního klienta a efektivní protokol. Jcards jako celek bude mít architekturu klient – server, což umožní centrální řízení a malé nároky na klientské programy. Mezi klientem a serverem bude probíhat komunikace pomocí vlastního protokolu, založeném na XMPP. V následujících podkapitolách konkrétněji popisuji jednotlivé části jcards. Nejprve protokol, poté klient a nakonec server.

## 4.1 Protokol

Komunikace bude probíhat pomocí XMPP protokolu přes jabber server (obr 4.1). Herní server jcards bude z pohledu jabber serveru klientem, stejně jako hráči. Z pohledu platformy však jcards bude řídit hru. Bude-li si například chtít hráč vzít kartu z talonu, pošle požadavek hernímu serveru jcards, ten prověří zda to pravidla umožňují a v případě že ano, bude informovat o tahu všechny zúčastněné hráče.




*obr 4.1: komunikace jcards*

### 4.1.1 Požadavky

Komunikace mezi klientem a serverem jcards by měla být poměrně malá, aby i u klientů s pomalým připojením k internetu probíhala hra plynule. Aby protokol umožnil hraní her s rozdílnými pravidly, musí přenášet následující informace. Některé jsou jednoúčelové, například zaslání textu pravidel, jiné obecné a budou využity v různých situacích, jako třeba dotaz na uživatele.

**Identifikace**

Aby mohlo hrát více skupin najednou, musí existovat způsob jak přiřadit uživatele ke konkrétní hře. Lze se rozhodovat podle JID uživatele. Toto řešení by však zvyšovalo nároky na server z důvodu prohledávání seznamu hráčů všech her. Proto jsem se rozhodl do protokolu přidat jednoznačný identifikátor hry („gameId“), který zašle hráč v každém požadavku serveru, pokud jej již zná. Server tak bude schopen rychle přiřadit požadavek ke hře, kde se má zpracovat. Tento parametr, v téměř každé zprávě, nijak významně nezatíží přenosovou linku a ušetří výpočetní výkon serveru.

### Vytvoření hry

Každý uživatel bude moci založit vlastní hru, ke které se ostatní připojí. Je možné také nastavit ochranu hry pomocí hesla. Po úspěšném vytvoření hry dostane uživatel parametr gameId, který si ponechá pro identifikaci všech následujících požadavků.

### Připojení ke hře

K již stávající hře se budou připojovat ostatní hráči. Při hře chráněné heslem jej bude muset zadat. Při úspěšném přihlášení dostane klient parametr gameId. K tomu aby si mohli hráči vybrat, dostanou od serveru seznam dostupných her.

### Start hry

Start hry bude třeba omezit podle situace. Například, aby byl připojený správný počet uživatelů. Klient tak pošle požadavek na start a server podle situace hru spustí nebo ohlásí chybu.

### Nastavení

Při zahájení hry bude nutné přenést informace o hrací ploše a rozdaných kartách. K nastavení hrací plochy je zapotřebí znát rozměry, typ karetních listů, typ a umístění balíků a k nim přiřazené karty, případně textovou podobu pravidel.

### Pohyby karet

Snad v každé karetní hře je nutné různě pohybovat a otáčet kartami, případně s nimi provádět další operace. Bude třeba umožnit přesun a otočení karty i balíku, respektive jeho části. Požadavek od klienta bude dvojího typu. Hráč bude moci konkrétní kartu odněkud někam přenést, případně provést akci s určitou kartou. V druhém případě herní server akci vykoná nebo nabídne možnosti. Složitější operace, jako například přesun balíku, bude řešen na serveru a rozdělen na operace přesunu a otočení karty.

### Řízení hry

Klienti potřebují vědět, kdo je na řadě, jestli mohou hrát nebo čekají na ostatní.

### Chyby

Pokud hráč požaduje neproveditelný tah, špatně zadá heslo ke hře nebo nastane nějaká nestandardní událost, server zašle text chyby.

### Informace

Dále jsou potřebné informace, které nemají charakter chyby. Jedná se například o zvolenou barvu.

### Statistiky

Dalším typem přenášených dat budou statistiky. Konkrétně herní a uživatelské. Herní statistiky budou zobrazovat získané body hráčů v jednotlivých kolech, uživatelské statistiky specifické parametry pro hráče, například počet uhraných zdvihů.

### Odhlášení

Požadavek na odhlášení bude moci poslat klient i server, v obou případech dojde k odhlášení hráče a případnému zastavení rozehrané partie.

## 4.2 Klient

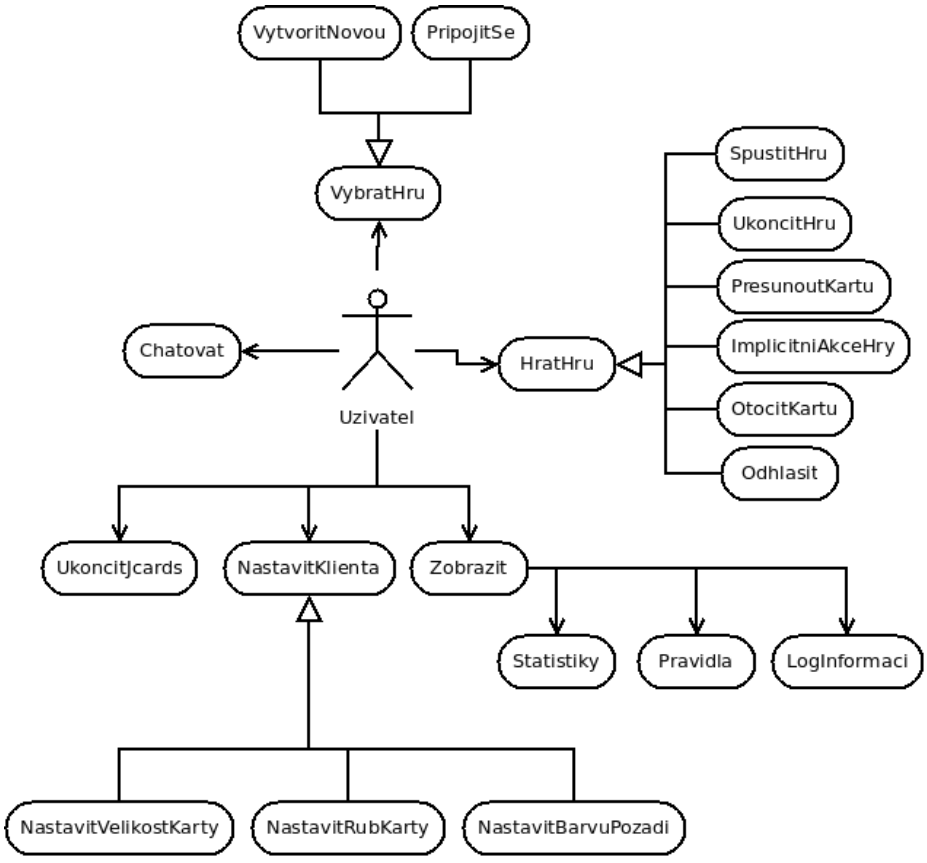
### 4.2.1 Požadavky

Klient bude zobrazovat hrací plochu a komunikovat se serverem. Proto musí umět minimálně tyto činnosti:

- komunikovat s herním serverem
- zobrazovat pravidla, log informací a statistiky
- zobrazit příchozí chyby, informace a dotazy
- přesouvat karty

Z těchto požadavků jsem vytvořil diagram případů užití (obr 4.2), který názorně specifikuje jednotlivé možné akce uživatele v systému.

Nejprve si hráč vybere hru, kterou může založit nebo se připojit k již existující. V samotné hře bude přesouvat karty, odpovídat na dotazy serveru a v libovolné chvíli se může odhlásit. Kromě toho bude moci nastavit vzhled hrací plochy, konkrétně velikost karet, barvu pozadí a motiv rubové strany karet. V neposlední řadě spolu budou hráči komunikovat pomocí MUC [19].




*obr 4.2: diagram užití klientské části*

### 4.2.2 Uživatelské rozhraní

K interakci s uživatelem bude sloužit několik oken nadefinovaných v Qt. Pro zobrazení informací a chyb si vystačím s messageBoxem, který nabízí knihovna Qt. Ostatní grafické prvky budou vytvořeny pomocí Qt Designeru jako samostatné moduly:

### Přihlašovací dialog

Tento dialog bude obsahovat seznam dostupných serverů. Po zvolení konkrétního herního serveru podle názvu hry se do dalšího seznamu načtou dostupné hry. Hráč si zvolí, zda chce hru vytvořit nebo se k již stávající připojit. Dále zde bude políčko pro zadání hesla při vytváření chráněné hry a zároveň pro zadání hesla ke hře, která jej vyžaduje.

### Hlavní okno

Zde bude většina interakce s uživatelem. Hlavní okno bude obsahovat menu, statusbar pro zobrazování informací, hrací plochu a záložky s pravidly, logem informací a statistikami.
Na hrací ploše budou vykresleny balíky a karty. Hráč bude zobrazen v podobě textu jeho JID nad balíkem typu „hand“, který bude sloužit pro karty, které má hráč v ruce. Ovládání plánuji jednoduché a intuitivní. Metodou drag and drop bude možné kartu přesunout, případně poklikáním na ni vyvolat nějakou akci. Co se uskuteční v návaznosti na poklikání bude záviset na pravidlech. Může se jednat o pouhé doplnění cílového umístění, jako v případě odhození karty na nějaký balík, nebo o nabídku možných akcí.

### Dialog nastavení

Bude možné nastavit barvu pozadí pomocí standardního Qt dialogu pro výběr barvy. Rubový motiv bude zobrazen jako řada obrázků, z kterých si hráč snadno vybere. Nastavení velikosti karty budu řešit posuvníkem. Pro lepší názornost přidám náhled, v kterém se vykreslí karta nastavené velikosti se zvoleným motivem rubu.

### Dialog čekání

Tento dialog se zobrazí ihned po připojení a poté mezi jednotlivými koly. Obsahovat bude jen dvě tlačítka: start hry a odhlášení. Start hry bude aktivní pouze pokud bude hru možné spustit.

## 4.2.3 Dekompozice

V této podkapitole popíši návrh rozdělení klienta do modulů a tříd (obr 4.3).

### jcards.py

Hlavní modul klienta jcards, v němž budou obsaženy všechny součásti kromě definice grafických oken. Bude se skládat z následujících tříd:

- **Plugin** – Navázání pluginu na jabbim klienta. Tato třída bude zděděna od PluginBase, která se nachází v programu jabbim. Také se zde bude nacházet zpracování protokolu.
- **MainWindow** – Třída hlavního okna.
- **Desktop** - Tato třída spravuje hrací plochu a zprostředkovává značnou část interakce s uživatelem. Hlavně přesuny karet pomocí myši a zobrazování změn.
- **Place** – Reprezentuje místo pro balík karet. Může být různého typu: karty v ruce hráče nebo balík, u každé varianty s různým otočením.
- **Card** – Zobrazení konkrétní karty. Tato třída umožní základní operace s kartou, jako otočení, překreslení nebo zjištění jejích parametrů.
- **MainDialog** – Dialog pro výběr hry. Zobrazí se jako první po spuštění jcards pluginu.
- **Settings** – Dialog nastavení vzhledu hrací plochy.
- **Error** – Subsystém zobrazování chyb. Pokud přijde více chyb po sobě, budou zachovány a zobrazeny ihned po zavření dialogu s aktuální chybou.
- **Info** - Subsystém zobrazování informací. Text informace zobrazí do aktuálně zobrazeného okna. Buďto do statusbaru hlavního okna, do přihlašovacího dialogu nebo samostatně pomocí messageboxu. Pokaždé se však zároveň přidá v hlavním okně do logu informací. Při příchodu více informací v krátkém čase se po určitém zpoždění zobrazí samovolně další.

- **Question** – Zpracování otázky od serveru. Při přijetí dotazu zobrazí otázku a seznam možných odpovědí, z kterých si hráč vybere a poté bude odpověď odeslána zpět serveru.
- **Wait** – Dialog čekání na spuštění hry. Obsahuje tlačítka pro start hry a odhlášení.

**mainWindow.py**

Specifikuje grafické rozhraní hlavního okna vygenerované pomocí Qt Designeru.

**wait.py**

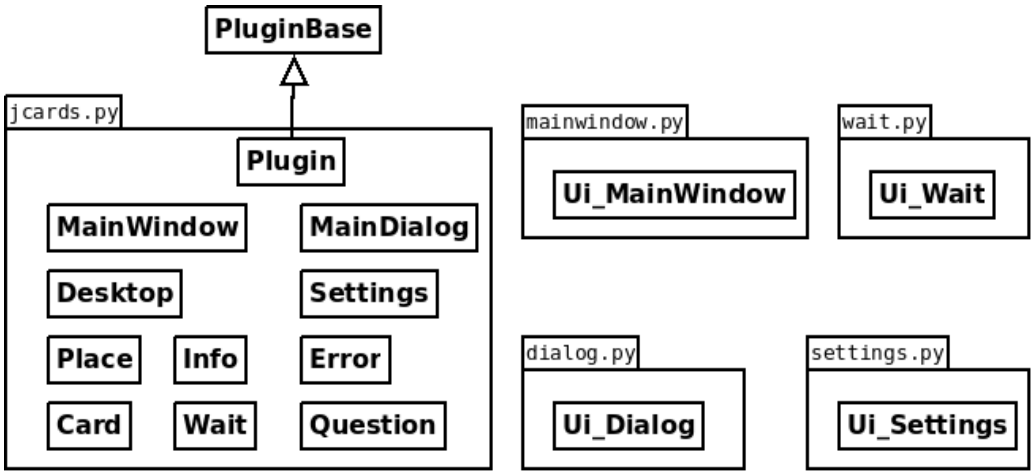
Specifikuje grafické rozhraní dialogu čekání vygenerované pomocí Qt Designeru.

**dialog.py**

Specifikuje grafické rozhraní dialogu pro přihlášení vygenerované pomocí Qt Designeru.

**settings.py**

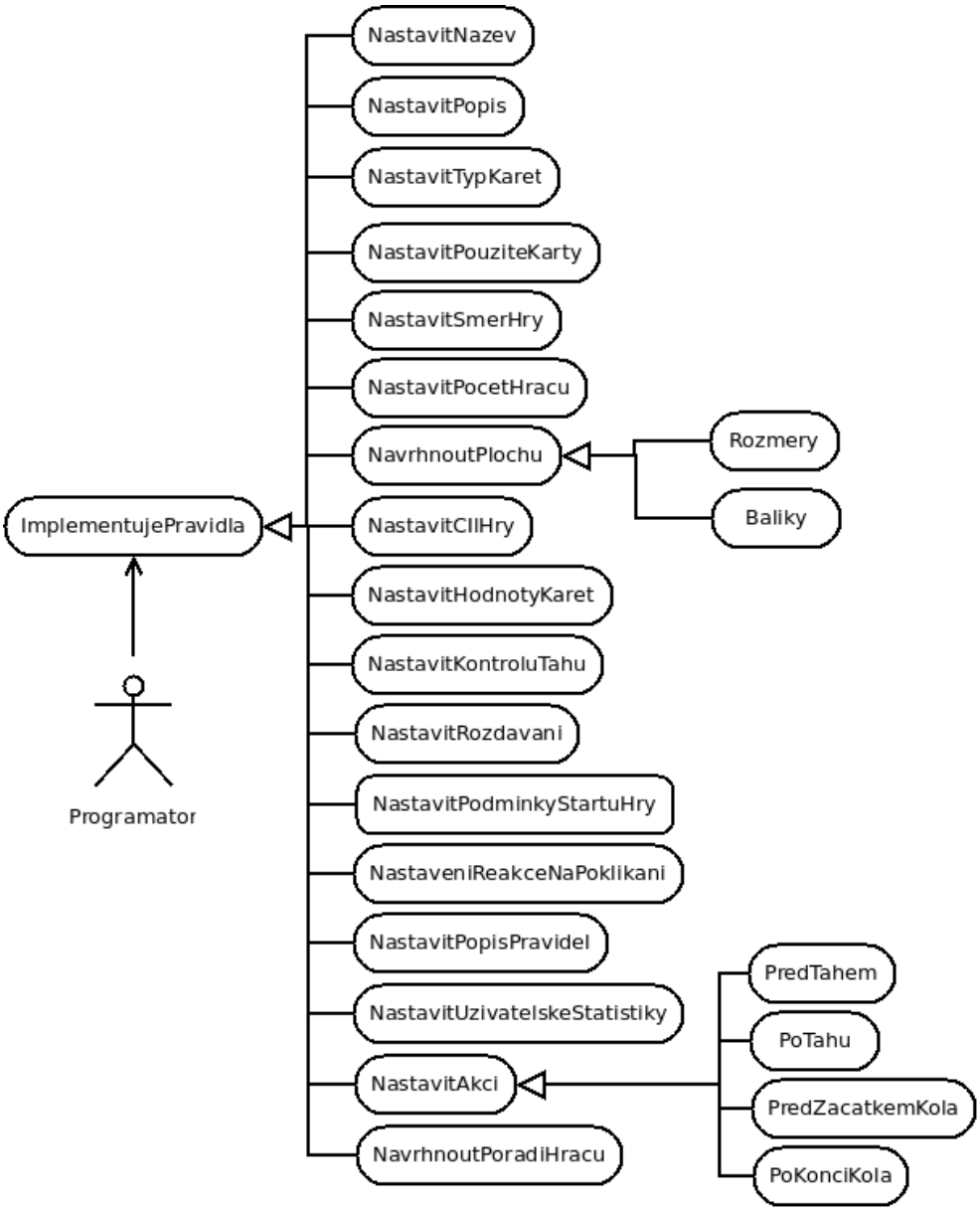
Specifikuje grafické rozhraní dialogu nastavení vygenerované pomocí Qt Designeru.




*obr 4.3: klient - struktura*

## 4.3 Server

Serverová část projektu jcards bude nabízet rozhraní k tvorbě libovolných klasických karetních her. Při tvorbě konkrétní hry programátor využije prostředky nabízené platformou jcards. Pro zjednodušení vývoje nových her bude vytvořena šablona, v které budou popsány nezbytné části kódu a okomentovaná struktura pravidel k doplnění.




*obr 4.4: diagram užití serverové části*

### 4.3.1 Požadavky

Každá klasická karetní hra se skládá z těchto kroků, které nemusí být nutně obsaženy všechny:

- rozsazení hráčů
- snímání
- rozdávání
- dražba
- sehrávka
- počítání skóre

Aby bylo možné pod jcards naprogramovat jakoukoli klasickou karetní hru co nejpohodlněji, měla by tedy nabízet následující funkce:

- generování jakéhokoli balíku z karet daného typu
- střídání hráčů po/proti směru hodinových ručiček
- rozdávaní
- dotazování hráčů
- sejmutí balíku
- přesun karty
- otočení karty
- přesun balíku, nebo jeho části
- otočení balíku, nebo jeho části
- identifikace výhry podle zadaných pravidel
- počítání skóre na konci každého kola
- generování statistik

Aby mohlo takto vytvořený herní server využívat více skupin hráčů najednou, bude každá hra spouštěna jako samostatné vlákno. Variantu řešení jednoho vlákna jsem zavrhl kvůli možnému blokování časově náročnou operací, což by znemožnilo ostatním hrát.
Každý server se bude přihlašovat k jabber serveru pod stejným JID (jcards@jabbim.cz) a lišit se jedinečným zdrojem (např. jcards@jabbim.cz/kanasta). Podle JID bez zdroje klient zjistí, jaké jsou dostupné servery a hru si vybere podle zdroje (kanasta). Obsluha XMPP komunikace poběží v samostatném vláknu a bude napsaná obecně, aby se dala opět využít, například při úpravě klienta z pluginu na samostatnou aplikaci.

Možnosti z pohledu uživatelů jcards jsou znázorněny v diagramu případů užití (obr 4.4). Uživatelem v tomto případě bude programátor tvořící vlastní karetní hru pomocí platformy jcards. Nebude muset programovat zobecnitelné prvky hry, ale pouze pravidla, v kterých se specifikují proměnné ovlivňující průběh hry a dopíše ty části, které jsou odlišné podle typu hry. Tvůrce hry nadefinuje jak se hra jmenuje a doplní text pravidel, které si budou moci číst hráči, pokud si něčím nebudou jistí. Dále nastaví typ karetních listů a směr hry (po/proti směru hodinových ručiček). Také nastaví hrací plochu, to znamená její rozměry a na ní umístěné pozice pro balíky. Jednotlivá pozice pro umístění balíku bude definovaná typem, podle nějž se bude vykreslovat, otočením a vlastníkem. Vlastník bude buďto „all“, neboli obecný balík používaný obvykle všemi hráči, nebo číslo od 0 do maximálního počtu hráčů sníženého o 1. Po rozmístění hráčů ve hře se tohoto údaje využije pro přidělení balíků jednotlivým hráčům. První hráč bude mít balíky na pozicích s vlastníkem „0“, druhý „1“ atd. Poslední, náročnější část tvorby hry bude spočívat v implementaci několika metod. Tímto způsobem se nastaví způsob rozdávání, kontrola proveditelnosti tahu, činnost po dvojkliku uživatele na kartu, akce prováděné před a po tahu či kole, za jakých podmínek nastane výhra a jakým způsobem se budou počítat hodnoty karet pro počítání skóre. Dále bude možné přenastavit standardní chování některých částí. Jedná se o generování karet z nastavených karetních listů, pokud je třeba použít jiný balík karet než kombinaci všech barev a hodnot, což udělá jcards standardně. Další nepovinné nastavení se týká ustanovení pořadí hráčů u stolu, má-li být jiné než náhodné. Poslední možností je předefinovat metodu, která se bude starat o povolování startu hry. Bez úprav bude zohledňovat pouze počet připojených hráčů.

### 4.3.2 Dekompozice

Na obrázku (obr 4.5) je znázorněn návrh rozdělení serverové části platformy jcards do modulů a jejich tříd.

### server.py

Hlavní část serverové části jcards. Tento modul bude obsahovat veškeré prostředky platformy, které můžou jednotlivé karetní hry využívat. Bude obsahovat následující třídy, řešící konkrétní prvek karetní hry:

- **Bot** – Hlavní třída serveru, bude řídit komunikaci mezi síťovým modulem a jednotlivými herními vlákny.
- **Desktop** – Správa hrací plochy. Pomocí této třídy se budou nastavovat parametry plochy a také bude generovat nastavení plochy v konkrétní podobě pro každého hráče.
- **Place** – Místo na ploše, na kterém bude umístěn balík karet a bude obsahovat data potřebná pro vykreslení u klienta.
- **Game** – Objekty této třídy budou reprezentovat jednotlivou hru a spravovat uživatele, balíky a další prvky.
- **Player** – Třída reprezentující hráče.
- **UserStats** – Uživatelské statistiky. Budou zobrazovat parametry a jejich hodnoty podle definice v pravidlech.
- **Pack** – Balík karet, tato část se bude starat o karty, jejich přidávání, mazání, vyhledávání v balíku a podobně.
- **Card** – Reprezentuje jednotlivé karty a jejich parametry.
- **Move** – Tato třída bude reprezentovat přesun karet a umožní jeho schválení či zamítnutí. Dále nabídne programátorovi karetních her pohodlný přístup ke všem informacím ohledně chystaného tahu a umožní mu přidávat další přesuny či otočení karet.
- **Question** – Subsystém pro získávání odpovědí od hráčů. Vytvoří se otázka s odpovědmi a obslužnou funkcí, která vyhodnotí reakce adresáta či adresátů.
- **GameStats** – Herní statistiky. Po každém kole se přidá bodové hodnocení jednotlivých hráčů. K tomu ještě dopočítá sumární skóre ve všech kolech pro jednotlivé hráče.
- **AbstractRules** – Obecná třída, která bude nabízet základní podporu pro pravidla. Konkrétní podobu pravidel bude definovat třída Rules.
- **Round** – Systém přidělování oprávnění hrát a uspořádání pořadí hráčů.

### networkThread.py

Tento modul bude zajišťovat práci s XMPP protokolem. Poběží v samostatném vlákně, a bude nabízet základní funkcionalitu jako připojení k síti, poslaní/příjem zpráv message,presence,iq.

### netwokt.py

V tomto modulu bude řešen komunikační protokol. Bude využívat síťových služeb modulu networkThread.py. Rozhraní k protokolu bude tvořeno pomocí Qt signálů a každý signál bude odpovídat nějakému příkazu komunikačního protokolu.

### decks.py

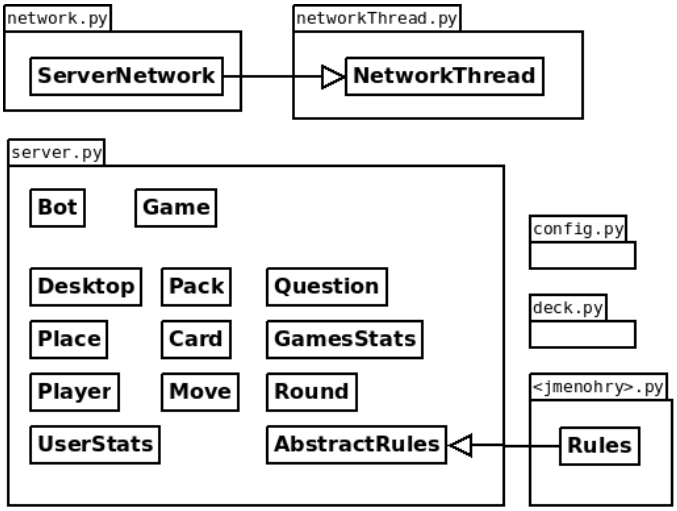
Zde se bude nacházet definice různých typů karetních balíčků. Ty budou reprezentovány názvem karetních listů a výčtem jejich barev a hodnot.

### config.py

V tomto modulu budou konstanty ovlivňující chování serveru.

### <konkretní hra>.py

V takto pojmenovaných modulech budou konkrétní pravidla vytvořených karetních her. Bude obsahovat nastavení plochy, spuštění serveru a hlavně třídu **Rules**, v které budou pravidla hry. Šablona pro tvorbu hry se bude nacházet ve složce server/games/ a jmenovat se empty.py.




*obr 4.5: server - struktura*

# 5 Implementace

V této části mé práce hodlám nastínit konkrétní implementaci jcards. Nejprve popíši protokol, poté klientskou část a následně serverovou část jcards. Detailní popis jednotlivých částí se nachází také v programové dokumentaci generované pomocí Doxygen [6]. Tam lze nalézt popis modulů, tříd, metod i proměnných včetně jejich typů.

## 5.1 Protokol

Jcards pracuje s dvěma typy XML zpráv: <presence/> a <message/>. Pomocí <presence/> se klient dozví o dostupných serverech, případně o odhlášení některého herního serveru. Pro serverovou část má <presence/> význam při výpadku klienta. Zpráva <message/> bez <body/> obsahuje všechny příkazy komunikačního protokolu. V následujících dvou kapitolách popisuji jednotlivé příkazy a ukázky komunikace při složitějších situacích.

Kromě těchto zpráv využívá jcards ještě i <iq>, ovšem pouze pro správu MUC místností, což není součástí protokolu.

### 5.1.1 Specifikace protokolu

Zde specifikuji posloupnosti jednotlivých příkazů protokolu jcards v průběhu hry. V textu „c“ značí nějakého klienta nebo konkrétně toho od něhož přišel požadavek, „C“ všechny klienty a „s“ herní server.

**Vytvoření hry**

c → s: newGame
s → c: joined
*nebo*
s → c: error

**Připojení ke hře**

c → s: listGames
s → c: gameList
c → s: joinGame
pokud se server nastaven aby se ptal ostatních hráčů:
    s → C: question
    C → s: answer
po odpovědi všech se rozhodne: pokud polovina a více stávajících hráčů hlasuje pro, je hráč přidán
    s → c: joined
*nebo*
    s → c: error

**Start hry**

c → s: startGame
pokud nelze hru spustit (například je přítomno málo hráčů):
    s → c: error
*jinak:*
    s → C: init
    s → C: cards

**Inicializace**

s → C: init
s → C: cards

**Pohyby karet**

c → s: move
s → C: moved
*nebo*
c → c: error

**Řízení hry**

s → C: turn

**Chyby**

s → c: error

**Informace**

s → c: info

**Statistiky**

s → C: userStats
s → C: gamesStats

**Odhlášení**

s → c: logout
*nebo*
c → s: logout

### 5.1.2 Popis příkazů

V této podkapitole popisuji strukturu dat, které jednotlivé příkazy přenášejí. Dále zmíním účel příkazu. Pro přehlednost jsem zvolil několik značek, které doplní formát přenášených dat.

- *** za tagem:** element se může vyskytovat 0..n-krát
- **[a,b]:** 'a', nebo 'b'
- **$JID$:** na toto místo se doplní konkrétní hodnota

Balík je definován názvem a vlastníkem, karta ještě k tomu indexem v balíku. Pokud je index karty -1, jedná se o vrchní kartu balíku, pokud -2, jde o kartu s indexem o 1 vetší, tedy umístění, kde ještě žádná karta není.
Všechny příkazy obsahují parametr xmlns, který má hodnotu „http://www.stud.fit.vutbr.cz/~xptako00/jcards/“. Pro přehlednost jej zde nebudu uvádět.
Parametr gameId identifikuje jednotlivou hru v rámci serveru.

**listGames**

Tímto příkazem žádá klient server o seznam dostupných her.

<listGames/>

**newGame**

Příkazem newGame žádá klient o vytvoření hry. Jako parametry uvede, zda má být chráněna hra heslem a jakým.

*<newGame lock='[True,False]' password='<heslo>' />*

### joinGame

JoinGame slouží k připojení k již stávající hře.

*<joinGame password="$heslo$ gameId='$identifikátor hry$' />*

### startGame

Požadavek na spuštění hry má podobu příkazu startGame.

*<startGame password='$heslo$' gameId='$identifikátor hry$' />*

### move

Pokud chce hráč přemístit nějakou kartu, pošle tento příkaz. Přenášené parametry jsou identifikace zdrojové a cílové kary.

```
<move>
        <src owner='$JID$' index='$index v balíku$'> name='$jmeno balíku$'/>
        <dst owner='$JID$' index='$index v balíku'$ name='$jmeno balíku$'/>
</move>
```

### action

Tento příkaz informuje server, že hráč chce s kartou, která je definovaná v parametrech, provést nějakou akci. Může se jednat například o otočení karty nebo odhození z ruky na nějaký balík.

```
<action gameId='$identifikátor hry$' >
        <src owner='$JID$'> index='$index v balíku$'> name='$jmeno balíku$'/>
</action>
```

### answer

Pomocí příkazu answer vrátí hráč odpověď na položenou otázku.

```
<answer gameId='$identifikátor hry$' questId='$identifikátor otázky v rámci hry$'>
        $odpověď$
</answer>
```

### logout

Příkazem logout oznamuje server hráči, že je odhlášen, případně jej může poslat hráč serveru a tím se ze hry odpojit.

*<logout/>*

### games

Server v příkazu games posílá klientům informace o dostupných hrách a jejich parametrech. Těmi jsou počet volných míst, JID zakladatele hry a informace, zda je hra chráněna heslem.

*<games>*
*<game lock=' [True,False]' gameId='$identifikátor hry$' createdBy='$JID$'*
*freePlayers='$počet volných míst$'/> **
*</games>*

### joined

Tímto server informuje hráče, že byl do hry připojen a sděluje mu identifikátor gameId, pomocí kterého dále specifikuje, v které hře je připojen.

*<joined gameId='$identifikátor hry$' />*

### error

Tento příkaz slouží ke sdělování chyb hráčům.

<error>
$text chyby$
</error>

### info

Tento příkaz slouží ke sdělování informací hráčům.

<info>
$text informace$
</info>

### question

Pomocí příkazu question získává server data od uživatelů. Pošle otázku s možnými odpovědmi, na které poté klient odpoví.

*<question questId='$identifikátor otázky v rámci hry$'>*
*<text>$text otázky$</text>*
*<answers>*
*<answer index='$index odpovědi$' type='[text,card]'>$text odpovědi$</answer> **
*</answers>*
*</question>*

### init

Init informuje klienty o počátečním stavu plochy. V parametrech přenáší text pravidel, rozměry hrací plochy a definici balíků. U nich je specifikována poloha na ploše, název, vlastník a případné otočení.

*<init listType='$typ karetních listů$'>*
*<rules>*
*$text pravidel$*
*</rules>*
*<desktop width='$šířka plochy jako násobek šířky karty$' height='$výška plochy jako*
*násobek výšky karty$'/>*
*<packs>*

```
                    <pack style='[pack,hand,line]' name='$název balíku$' posX='$pozice na ose
                        X$' posY='$pozice na ose Y$' owner='[all,$JID$]'
                        rotation='[no,left,right,reverse]'/> *
                </packs>
</init>
```

### cards

Tímto příkazem se přenáší úvodní rozmístění karet na ploše.

```
<cards cardsTotal='$celkový počet karet$'>
        <pack owner='[all,$JID$]' name='$jméno balíku$'>
                <card visible='[True,False]' index='$index karty$' value='$hodnota karty$'
                    suit='barva karty'/> *
        </pack> *
</cards>
```

### moved

Tento příkaz informuje klienty o změnách poloh karet. Může obsahovat libovolné množství elementů <move/> a <reverse/> pro přesun a otočení karty. Jednotlivé operace nad kartami se provedou v pořadí, v jakém jsou v přenášené zprávě. Při přesunu je nadefinovaný zdrojový a cílový balík. Při otočení identifikace karty, která se má otočit.

```
<moved xmlns='http://www.stud.fit.vutbr.cz/~xptako00/jcards/'>
        <move>
                <src owner='$JID$'> index='$index v balíku$'> name='$jmeno balíku$'/>
                <dst owner='$JID$' index='$index v balíku'$ name='$jmeno balíku$'/>
        </move>*
        <reverse>
                <src owner='[$JID$,all]' index='$index v balíku$' name='$jméno balíku$'/>
                <card visible='[True/False]' value='$hodnota karty$' suit='$barva karty$'/>
        </reverse>*
</moved>
```

### userStats

Uživatelské statistiky, přenášené pomocí userStats, obsahují seznam parametrů a jim odpovídajících hodnot.

```
<userStats>
        <parametr name='$jméno parametru$'>
                $hodnota parametru$
        </parametr>*
</userStats>
```

### gamesStats

Příkazem gamesStats se přenáší herní statistiky v podobě bodového hodnocení každého kráče v každém kole.

```
<gamesStats>
        <game>
                <player points='$počet bodů$'>
                        $JID hráče$
```

```
            </player>*
        </game>*
</gamesStats>
```

**wait**

Tímto sděluje server klientům, že nemohou hrát, ale čekají na start hry. Jako parametr je uveden důvod čekání.

```
<wait canStart='[True,False]'>
        <reason>
            $text důvodu čekání$
        </reason>
</wait>
```

**continue**

Tento příkaz navazuje na předchozí a oznamuje konec čekání.

```
<continue/>
```

**turn**

Příkazem turn je sdělována informace o hráči, který je aktuálně na řadě a může hrát.

```
<turn>
        $JID$
</turn>
```

# 5.2 Klient

Klientská část jcards je koncipována jako plugin do jabber klienta jabbim. Není však problém menší úpravou z něj udělat plugin do jiného klienta nebo samostatnou aplikaci. Při samostatné aplikaci by síťová část vypadala podobně jako u serveru a třídu Plugin by nahradila hlavní třída aplikace zděděná od PyQt4.QtGui.QApplication.

Grafika je vytvořená pomocí PyQt4 knihovny za pomoci Qt Designeru. Výstupem jsou soubory *.ui, které program make převede na moduly pythonu. Pro překlad grafiky na MS Windows je možné použít skript makeGui.bat. Ten vyžaduje program pyuic4 v systémové cestě.

## 5.2.1 Jabbim plugin

Do jabbimu je klientská část integrovaná tak, že hlavní třída je zděděna od PluginBase, kterou nabízí jabbim. V ní jsou specifikované vlastnosti jako název, autor, popis pluginu a další. Také je zde nastavena položka v menu, která vyvolá přihlašovací dialog a ikonka v hlavním okně se stejnou funkcí. Z jabbim klienta jcards dále využívá groupchat a obsluhu XMPP streamu.

# 5.3 Server

## 5.3.1 Tvorba karetní hry

Cílem bylo mimo jiné co nejvíce zjednodušit tvorbu dalších karetních her. Postup psaní nové hry lze rozdělit na povinné kroky a volitelné. Bez povinných hra nemůže fungovat, z volitelných může programátor využít, co potřebuje. Šablona empty.py obsahuje komentovanou strukturu kódu pravidel

karetní hry. Konkrétní popis platformy jcards obsahuje programová dokumentace na přiloženém CD. V ní programátor nalezne popis jednotlivých modulů, tříd i všech parametrů a proměnných.

V následujících odstavcích se nachází obecný návod pro tvorbu karetní hry, detailní možnosti jednotlivých částí jcards jsou popsány v již zmiňované programové dokumentaci vygenerované pomocí Doxygen[6].

## Nutné kroky:

### Návrh hrací plochy

Hrací plocha je definována jako matice políček o velikosti hrací karty. Na plochu se umisťují jednotlivé balíčky, identifikované jménem a vlastníkem. Jméno může být libovolné. Vlastník nabývá hodnot 0 až (maxPlayers-1), nebo hodnoty „all“, vhodné pro společné balíky, například talon. Balík nabývá některého z typů: hand, pack, line. Následuje tabulka doporučených rozestupů mezi balíky podle typu a otočení. Čísla znamenají počet políček, které je vhodné nechat volné v daném směru.

| Typ balíku | popis | 0°, 180° | | | | 90°, 270° | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | ← | ↑ | → | ↓ | ← | ↑ | → | ↓ |
| hand | Hráčova ruka | **2** | | **2** | | | **2** | | **2** |
| pack | Balík na sebe poskládaných karet | **1** | **1** | | | **1** | **1** | | |
| line | Do řady vyložené karty | **2** | | **2** | | | **2** | | **2** |

Po vhodném návrhu rozložení balíků a hráčů (balíky typu hand) se zadefinují rozměry hrací plochy a přidají jednotlivé balíky.

### Základní nastavení

Je potřeba nastavit jméno karetní hry a proměnné třídy Rules:

- **description** – textový popis pravidel
- **minPlayers** – minimální počet hráčů
- **maxPlayers** – maximální počet hráčů
- **listType** – typ karetních listů
- **units** – jednotka pro statistiky ($, body, ...)
- **scoreBetter** – jaké je žádané skóre: co nejmenší/největší
- **userStatsParams** – seznam parametrů uživatelských statistik
- **roundDirection** – směr hry

### Rozdávání

V metodě serv() se nachází popis rozdávání. K tomu slouží metoda deck2pack() třídy Game. Je tedy třeba jen určit, kolik karet na jaký balík rozdat a o ostatní se postará logika platformy jcards.

### Povolení tahu

Klíčovou částí pravidel v platformě jcards je metoda canStart(), jejíž návratová hodnota rozhoduje o provedení tahu nebo nahlášení chyby a zamítnutí. Jako argument dostane metoda instanci Move, která zprostředkovává tah pohodlně pro programátora. Umožňuje zjistit souřadnice zdroje a cíle, hráče, který daný tah požaduje, ukazatele na instanci Game jako základní objekt hry, balíky a přenášenou kartu. Navíc umožňuje přidat další operace, které se mají vykonat. Jsou jimi: přesun karty/balíku a otočení karty/balíku.

### Výhra

Dále je třeba v metodě checkGoal specifikovat podmínky výhry. Jako návratová hodnota je None (nikdo dosud nevyhrál) nebo hráč (instance Player), který vyhrál.

### Vyhodnocení

Počítání výsledků se děje pomocí metody eval(), která bude funkční po implementaci statistik.

## Volitelné kroky:

### Akce po kliknutí na kartu

Není nutné žádné nastavení, ovšem hráči uvítají pohodlnější poklikání na kartu. V metodě actions() se ke zdrojové kartě přiřadí cílová pozice, případně None pro žádnou akci. Speciálnějšího chování lze dosáhnout například pomocí třídy Move.

### Povolení startu

Start hry je možný, pokud je počet hráčů v rozmezí minPlayers a maxPlayers. Pokud je vhodné jiné chování, stačí předefinovat metodu canStart().

### Ustavení pořadí hráčů

Obvykle stačí při každém kole rozmístit hráče náhodně. Pokud ne, pořadí se nastaví v metodě sittingOrder().

### Generování sady karet

V základním nastavení se generují karty metodou kombinace všech barev a hodnot. Jiné chování nastavuje metoda generateDeck().

### Před začátkem kola

Metoda beforeRoundStart() se volá těsně před začátkem hry a je dobrým místem pro inicializaci proměnných.

### Po konci kola

Metoda afterRoundEnd() specifikuje akce po konci kola. Pro většinu her zřejmě nebude užitečná.

### Před tahem

V metodě beforeMove je dobré například nastavit vlastní proměnné, které budou potřeba v dalších metodách: canMove(), afterMove().

### Po tahu

Po schválení a provedení tahu se volá metoda afterMove(). Zde bude patrně velká část pravidel, jako reakce na daný tah. Například dotaz hráčů, další přesuny nebo otočení karet či balíků.

## 5.4 Prší

Jako demonstrační hru jsem si zvolil prší. Na ní ukazuji většinu možností platformy. Modul prsi.py doporučuji jako inspiraci při tvorbě dalších her.

V této hře je vidět návrh plochy a všechny ostatní základní části. V metodě afterMove se nachází většina pravidel a navíc má prší vlastní metodu pro doplnění talonu pokud dojde, z balíku pro odhazování karet. Pokud hráč odhodí sedmu či eso, zhodnotí se zda další hráč má nějaké možnosti a pokud ne, tah provede server za ně. Například poslání dvou karet z talonu, pokud nemá sedmu na přebití. Metoda action je využita tak, že pokliká-li hráč na kartu, kterou má v ruce, bere toto server jako požadavek na odhození karty. Pokud takto zvolí vrchní kartu talonu, předpokládá se líznutí karty. Stav hry se ukládá do proměnných třídy Rules. V nich je zaznamenán z pohledu speciálních karet, sedem, es a svršků.

# 6 Závěr

Výsledkem mé práce je aplikace typu klient-server (jcards). Klientská část je implementovaná jako plugin XMPP klienta jabbim s použitím grafické knihovny PyQt4 a je multiplatformní. Umožňuje všechny činnosti definované v návrhu (obr. 4.2), kromě statistik. Podle ohlasů uživatelů se podařilo splnit cíl jednoduchosti a intuitivnosti ovládání. Serverovou část tvoří herní server využívající PyQt4 knihovnu k práci s vlákny a komunikaci mezi moduly pomocí Qt signálů. Mezi klientem a serverem jcards probíhá komunikace přes XMPP pomocí vlastního protokolu. Ten umožňuje vše, co je třeba ke klasickým karetním hrám.

Jako ukázkovou hru jsem vytvořil „prší“. Tato hra je plně funkční a názorně předvádí možnosti platformy jcards. Soubor s pravidly této středně komplikované hry zabírá méně než 350 řádků v pythonu (počítáno bez příkazů pro logování). Tento výsledek je přijatelný a jedná se o zanedbatelně malý kód, který musí tvůrce hry napsat, oproti implementaci celé síťové hry bez jcards.

Při implementaci serveru jsem narazil na problém s vlákny pod PyQt4. Nakonec jsem za pomoci vedoucího bakalářské práce zjistil, že PyQt4 oproti Qt4 pro C++ nepodporuje plně vlákna. Nyní sice jednotlivé hry běží v samostatném vlákně QThread, ovšem ve skutečnosti jde jen o simulaci více vláken běžících v jednom. Doufám, že bude v budoucnu dořešena lepší podpora vláken v PyQt.

## 6.1 Další vývoj

V průběhu návrhu a implementace jcards jsem narazil na mnohé zajímavé možnosti, o které by se mohl tento projekt rozšířit. Některé subsystémy také mohou po úpravě fungovat lépe, či být ještě univerzálnější.

### Další typy karetních listů

Zatím jcards obsahuje pouze obrázky německých karetních listů. Ty jsem získal volně z wikipedie [22].

### Průvodce tvorbou hry

Průvodce by v budoucnu mohl pomoci tvůrcům karetních her s návrhem hrací plochy, nastavením parametrů a možná i s vlastním kódem pravidel. Mohl by být realizovaný v pythonu s PyQt4 knihovnou.

### Sběratelské karetní hry

Jcards je možné rozšířit, aby obsahoval prostředky i pro moderní karetní hry, jako třeba sběratelské. Proto je nutné provést minimálně následujících několik změn:

- přenos klientovi neznámých karetních listů protokolem
- propracovanější pozicování balíků a karet, například možnost natočit kartu o libovolný úhel
- uchovávání více informací o kartách – zatím se karta identifikuje podle dvojice název a vlastník, ke sběratelským karetním hrám bude třeba jednotlivé karty identifikovat jinak a uchovávat kromě obrázku i vlastnosti.

### Sloučení s jgames

Jako vhodný krok dalšího vývoje se jeví sloučením pluginu jcards s jiným pluginem jabbimu, jgames (viz kapitola 3). Jcards by poté fungoval jako nadstavba jgames. Jgames by nabízel nízkoúrovňové služby jako vzdálené volání funkcí a jcards prostředky vyšší, specifické pro karetní hry. Dosud však jgames není dokončený a vývoj těchto projektů se také může spojit.

# Literatura

### Knižní zdroje:


[1] OMASTA, Vojtěch; RAVIK, Slavomír. *Karty, hráči, karetní hry*, doplněné vydání. Pohořelice : KMa, s.r.o., 2007,.608 s. ISBN 978-80-7309-521-5.
[2] OMASTA, Vojtěch; RAVIK, Slavomír. *Velká kniha karetních her*. Praha : Regia, 2000. 474 s. ISBN 80-86367-00-2.
[3] OMASTA, Vojtěch; RAVIK, Slavomír. *100+1 karetních her*. Bratislava, Slovart : 1993. vydání první, 342 s. ISBN 80-7145-050-2.


### Zdroje z internetu:

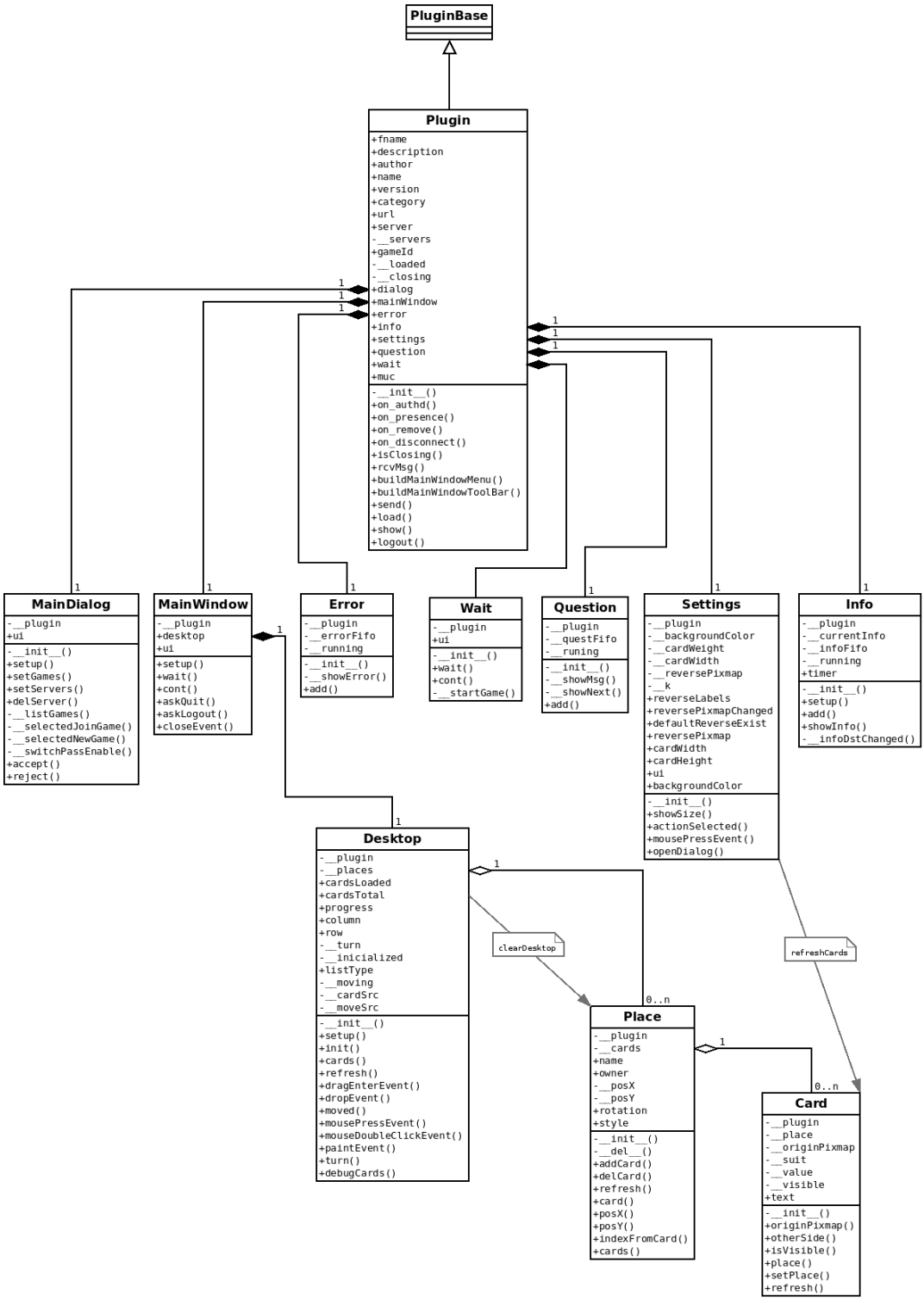

[4] BISHOP, Michael; LIRETTE, Keith. FLETCHER, Boyd: *Whiteboard* [online]. 17.7.2001 [cit. 3.5.2010]. Dostupný z WWW: <http://xmpp.org/extensions/inbox/whiteboard2.html>.
[5] HÄUSSGE, Gina; NEUMANN, Philippe. *Doxypy* [online]. 14.10.2009 [cit. 3.5.2010]. Dostupný z WWW: <http://code.foosel.org/doxypy>.
[6] HEESCH, Dimitri. *Doxygen* [online]. 24.4.2010 [cit 10.5.2010]. Dostupný z WWW: <http://www.stack.nl/~dimitri/doxygen/>.
[7] LUDWIG, Scott; BEDA, Joe; SAINT-ANDRE, Peter; MCQUEEN, Robert; EGAN, Sean; HILDEBRAND, Joe *XEP-0166: Jingle* [online]. 23.12.2009 [cit. 3.5.2010]. Dostupný z WWW: <http://xmpp.org/extensions/xep-0166.html>.
[8] KOLEKTIV AUTORŮ. *Extensible Markup Language* [online]. 4.5.2010 [cit. 5.5.2010]. Dostupný z www: <http://cs.wikipedia.org/wiki/Extensible_Markup_Language>.
[9] KOLEKTIV AUTORŮ. *Instant messaging* [online]. 10.4.2010 [cit. 3.5.2010]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Instant_messaging>.
[10] KOLEKTIV AUTORŮ. *Jabber* [online]. 11.10.2008 [cit. 3.5.2010]. Dostupný z WWW: <http://www.jabber.cz/wiki/Jabber>.
[11] KOLEKTIV AUTORŮ. *Karetní hra* [online]. 1.5.2010 [cit. 3.5.2010]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Karetní_hra>.
[12] KOLEKTIV AUTORŮ. *Kaskádové styly* [online]. 3.5.2010 [cit. 3.5.2010]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Cascading_Style_Sheets>.
[13] KOLEKTIV AUTOŮ. *Python* [online]. 12.4.2010 [cit. 5.5.2010]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Python>.
[14] PATERSON, Ian; PERLOW, Jon; SAINT-ANDRE, Peter; KARNEGES, Justin; TSVYASHCHENKO, Alexander; LEBOULANGER, Yann. *XEP-0136: Message Archiving* [online]. 6.1.2010 [cit. 3.5.2010]. Dostupný z WWW: <http://xmpp.org/extensions/xep-0136.html>.
[15] SAINT-ANDRE, Peter. *End-to-End Signing and Object Encryption for the Extensible Messaging and Presence Protocol (XMPP)* [online]. 20.2.2010 [cit. 3.5.2010]. Dostupný z WWW: <http://tools.ietf.org/html/rfc3923>.
[16] SAINT-ANDRE, Peter. *Extensible Messaging and Presence Protocol (XMPP): Core* [online]. 16.3.2010 [cit. 3.5.2010]. Dostupný z WWW: <http://tools.ietf.org/html/rfc3920>.
[17] SAINT-ANDRE, Peter. *Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence* [online]. 22.4.2010 [cit. 3..5.2010]. Dostupný z WWW: <http://tools.ietf.org/html/rfc3921>.
[18] SAINT-ANDRE, Peter. *Mapping the Extensible Messaging and Presence Protocol (XMPP) to Common Presence and Instant Messaging (CPIM)* [online]. 20.2.2010 [cit. 3.5.2010]. Dostupný z WWW: <http://tools.ietf.org/html/rfc3922>.
[19] SAINT-ANDRE, Peter. *XEP-0045: Multi-User Chat* [online]. 16.7.2008 [cit. 3.5.2010]. Dostupný z WWW: <http://xmpp.org/extensions/xep-0045.html>.

[20] SAINT-ANDRE, Peter; MILLARD, Peter; MULDOWNEY, Thomas; MISSING, Julian. *XEP-0084: User Avatar* [online]. 5.11.2008 [cit. 3.5.2010]. Dostupný z WWW: <http://xmpp.org/extensions/xep-0084.html>.
[21] *History of XMPP* [online]. 6.1.2008 [cit. 3.5.2010]. Dostupný z WWW: <http://xmpp.org/about/history.shtml>.
[22] *Soubor:Huncards.jpg* [online]. 3.6.2006 [cit. 13.4.2009]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Soubor:Huncards.jpg>.
[23] *Twisted* [online]. 3.5.2010 [cit. 3.5.2010]. Dostupný z www: <http://twistedmatrix.com/trac/>.
[24] *XMPP* Extensions [online]. 2.5.2010 [cit. 3.5.2010]. Dostupný z WWW: <http://xmpp.org/extensions/>.

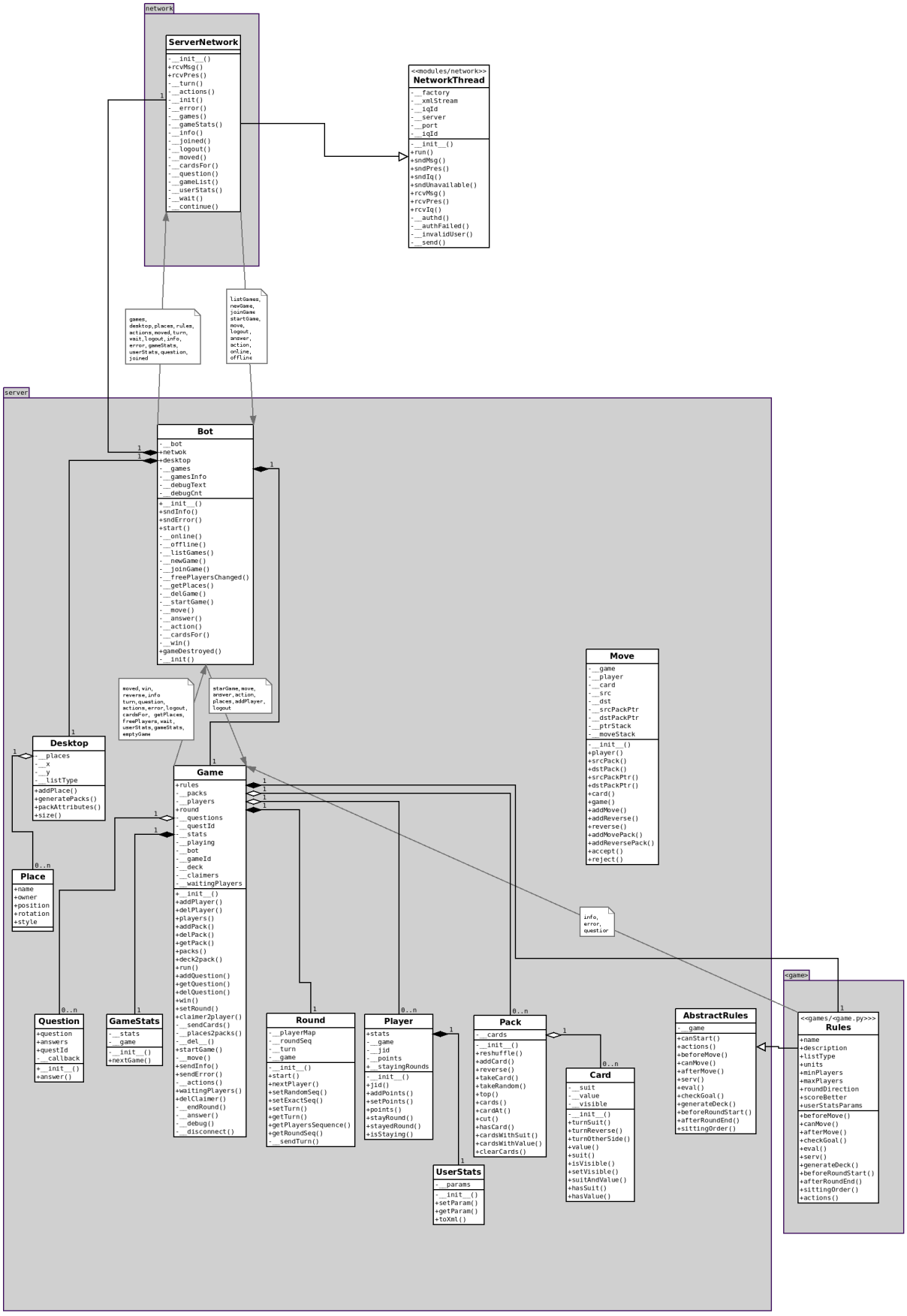
# Přílohy

Do příloh jsem umístil části popisného charakteru. Konkrétně diagramy tříd, které poskytují dobrou orientaci ve struktuře programu, uživatelskou příručku pro plugin jcards a na ukázku zdrojový kód hry prší. Dále k přílohám patří externě uložená programová dokumentace generovaná pomocí doxygen [6]. Ta se nachází na CD ve složce doc/doxygen a je ve formátu html.

# 1 Diagramy tříd




*obr 1.1: diagram tříd - klientská část*




*obr 1.2: diagram tříd - serverová část*

# 2 Uživatelská příručka

## 2.1 Instalace

### 2.1.1 Klient

Klient jcards funguje jako plugin do programu jabbim. K instalaci je třeba pouze zkopírovat složku jcards/jcards do složky plugins v adresáři s nainstalovaným jabber klientem jabbim.

### 2.1.2 Server

Před instalací serveru je potřeba mít nainstalované:

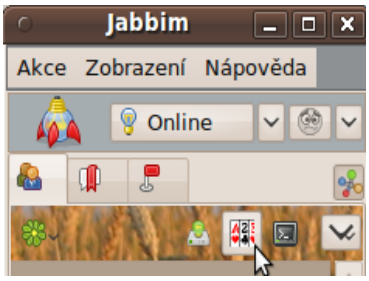
- python 2.4
- PyQt4
- twisted
  - words
  - internet

Samotná instalace spočívá v zkopírování složky server na pevný disk.

## 2.2 Spuštění

### 2.2.1 Klient

Nejprve je třeba mít povolené rozšíření jcards v nastavení jabbim, poté již stačí kliknout na ikonku karet (obr 2.1) nebo vybrat v položce menu: Akce → Rozšíření → Jcards



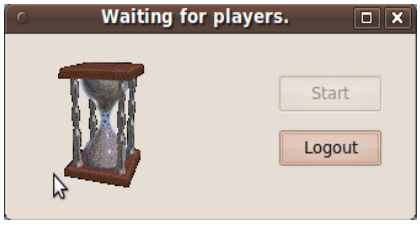
*obr 2.1: spuštění jcards*

### 2.2.2 Server

Herní server pro konkrétní hru se spustí příkazem
**python nazevhry.py**
z adresáře
**server/games**

## 2.3 Ovládání pluginu jcards

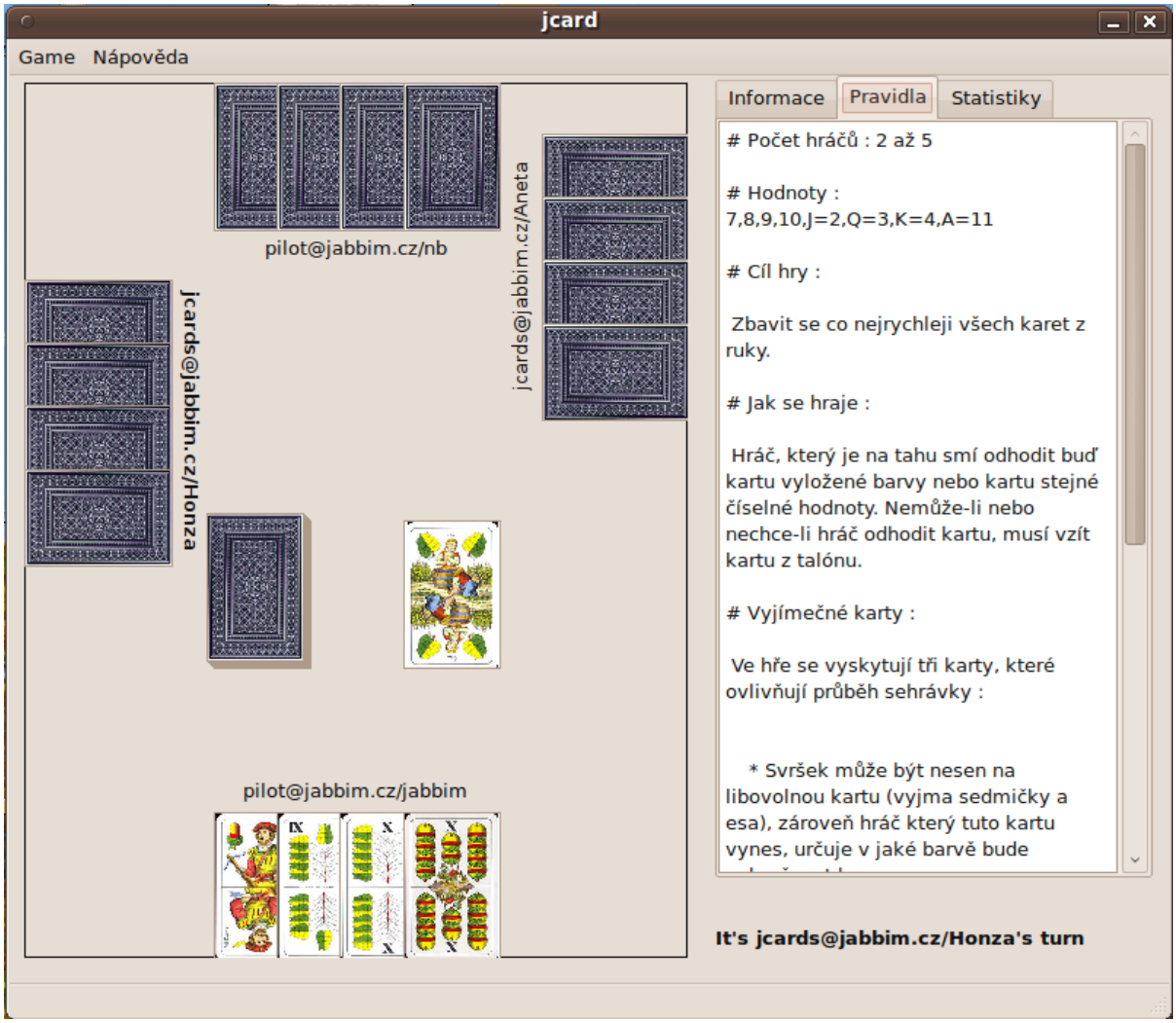
V přihlašovacím dialogu (obr 2.2) je třeba nejprve v prvním sloupci vybrat herní server (konkrétní typ hry) a poté vybrat, zda chceme hru vytvořit nebo se připojit ke stávající. Při vytváření hry je možno nastavit heslo, které ostatní budou muset při připojení zadat. Při připojování ke stávající hře je nutno vybrat konkrétní hru podle hráče, který ji založil. Pokud hra vyžaduje heslo, vyplní se do kolonky password. Tlačítko Refresh games slouží k získání obnovení seznamu her.


*obr 2.3: čekání na start hry*

*obr 2.2: přihlašovací dialog*

Po kliknutí na OK se otevře čekací dialog (obr 2.3) a okno se skupinovým rozhovorem všech do hry připojených hráčů. Pokud je hru možné spustit, je tlačítko start aktivní.




*obr 2.4: hlavní okno, hra prší*

Po startu hry se otevře hlavní okno (obr 2.4), které obsahuje:

- menu

- hrací plochou,
- pravidla,
- log informací ze serveru
- statistiky (zatím neimplementováno)
- statusbar
- jméno hráče, který je právě na řadě

Pod hlavní položkou **menu** (game) se skrývají možnosti: odhlásit se ze hry, ukončit jcards a nastavení. V dialogu nastavení lze změnit obrázek na rubové straně karet, velikost karet a barvu pozadí.

**Hrací plocha** obsahuje balíky s kartami. Karta se přenese přetažením na cíl, nebo poklikáním, což vyvolá standardní akci (třeba odhození karty) nebo nabídku s možnostmi. Přesouvat karty v ruce může hráč kdykoli, i když není na řadě.

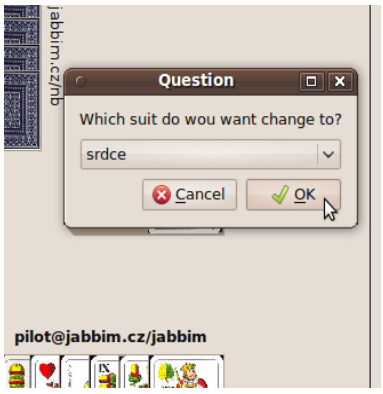
**Pravidla** obsahují text pouze pro čtení s popisem pravidel poslaných ze serveru.

Do **logu informací** se zapisují zprávy <info/>, například jakou barvu hráč zvolil.

Statistiky zobrazují (zatím neimplementováno) uživatelské a herní statistiky. Uživatelské obsahují parametry nadefinované v konkrétní karetní hře (například počet uhraných zdvihů) s hodnotami. Herní statistiky obsahují bodové hodnocení v každém kole (řádek) pro každého hráče (sloupec). Na konci potom sumu bodů.

**Jméno hráče,** který je právě **na řadě,** se zobrazuje dole vpravo. Souběžně je hráč zvýrazněn tučně na hrací ploše.

Na dotaz serveru vyskočí dialog s otázkou a výběrem z možností (obr 2.5). Chyby jsou zobrazovány běžným messageboxem s popisem chyby a informace podle momentálně viditelných oken v statusbaru hlavního okna, spodní liště přihlašovacího dialogu, nebo messageboxem v případě zobrazeného dialogu čekání. Pokaždé se však přidá i do logu informací v hlavním okně.




*obr 2.5: dotaz*

# 3 Obsah CD

Přiložené CD obsahuje následující složky:

- server/ zdrojové kódy serverové části jcards
  - games/ jednotlivé karetní hry
- jcards/ zdrojové kódy pluginu jcards
  - data/ obrázky karet
- doc/ dokumentace, technická zpráva
  - dia/ diagramy
  - doxygen/ programová dokumentace

V kořenové složce CD se nachází Makefile, soubory projektu doxypy [5] a README.

# 4 Zdrojový kód prší

```
# -*- coding: utf-8 -*-
import os
os.sys.path.append("..")
from server import *

NAME="prsi"
prsi=Bot(NAME)
# DESKTOP
prsi.desktop=Desktop(width=7, height=6)

prsi.desktop.addPlace(Place('talon', 'all', "pack", (2,3)))
prsi.desktop.addPlace(Place('odkladani', 'all', "pack", (4,3)))
prsi.desktop.addPlace(Place('hand', 0, "hand", (3,5)))
prsi.desktop.addPlace(Place('hand', 1, "hand", (0,2), "right"))
prsi.desktop.addPlace(Place('hand', 2, "hand", (3,0), "reverse"))
prsi.desktop.addPlace(Place('hand', 3, "hand", (6,1), "left"))
prsi.desktop.addPlace(Place('hand', 4, "hand", (6,4), "left"))

class Rules(AbstractRules):
        '''
        Pravidla karetní hry prší.
        '''
        # class definition for bot.game.rules
        name=""
        description=u'''# Počet hráčů : 2 až 5\n
# Hodnoty : 7,8,9,10,J=2,Q=3,K=4,A=11\n
# Cíl hry :\n
 Zbavit se co nejrychleji všech karet z ruky.\n
# Jak se hraje :\n
 Hráč, který je na tahu smí odhodit buď kartu vyložené barvy nebo kartu stejné číselné hodnoty. Nemůže-li nebo nechce-li hráč odhodit kartu, musí vzít kartu z talónu.\n
# Vyjímečné karty :\n
 Ve hře se vyskytují tři karty, které ovlivňují průběh sehrávky :\n

    * Svršek může být nesen na libovolnou kartu (vyjma sedmičky a esa), zároveň hráč který tuto kartu vynes, určuje v jaké barvě bude pokračovat hra.\n
    * Sedma, je-li právě shozena, nutí hráče který je na řadě vzít dvě karty z talónu, aniž by nějakou kartu sám odhodil. Má-li však ve svém listu také sedmu může ji shodit a následující hráč již musí brát z talónu čtyři karty.\n
    * Eso - hráč po levici hráče, který eso shodil, může toto přebít pouze jiným esem. Jestliže eso nemá, nemůže shodit žádnou jinou kartu a ani nedobírá žádnou kartu z talónu.\n
# Hodnocen\n
 Na konci hry se sečtou karty které hráči zůstali v ruce. Vyhrává ten hráč, který má po dohodnutém počtu kol nejmně bodů.
'''
        minPlayers=2
        maxPlayers=5
        listType='nemecke'
        units='b'     # points, $, ...
        scoreBetter="high"      # high/low
        userStatsParams=[]
        roundDirection="clockwise"          # clockwise/anticlockwise


        def serv(self):
                '''
                Rozdávání karet.
                '''
        for player in self.game.players().items():
                                self.game.deck2pack(player[0], 'hand', 4)
                self.game.deck2pack('all', 'odkladani', 1, visible=True)
                self.game.deck2pack('all', 'talon')    # ostatni karty

        def canMove(self,move):
                '''
                Rozhodne o proveditelnosti tahu.
                @param game ukayatel na instanci Game
                @param move tah ke kontrole
                @todo kontroluje se jestli je uživatel na řadě?
```

```
                '''
                if move.srcPack() == ("hand",move.player()) and move.dstPack() == ("odkladani","all"):
                        # VYNESENI KARTY
                        # stejna hodnota nebo barva, pripadne svrsek kdykoli
                        topCard=self.game.getPack('odkladani','all').top()
                        move.reverse()

                        sameValue=move.card().value() == topCard.value()
                        if self.j:
                                sameSuit=move.card().suit() == self.suit
                        else:
                                sameSuit=move.card().suit() == topCard.suit()
                        j=move.card().value() == 'svrsek'
                        jOnTop=topCard.value() == 'svrsek' and move.card().suit()==self.suit
                        ace=self.ace and move.card().value() == "eso"
                        seven=self.sevens>0 and move.card().value() == '7'
                        if self.sevens > 0:
                                if move.card().value() == "7":
                                        return True
                                else:
                                        return False
                        elif self.ace:
                                if move.card().value() == "eso":
                                        return True
                                else:
                                        return False
                        elif sameSuit or sameValue or j:
                                return True
                # LIZNUTI JEDNE KARTY
                elif move.srcPack() == ("talon","all") and move.dstPack() == ("hand",move.player()):
                        if not self.ace:
                                return True
                        else:
                                return False
                elif move.srcPack() == ("hand",move.player()) and move.dstPack() == move.srcPack():
                        return True

                else:
                        return False

        def beforeMove(self,move):
                '''
                Metoda volaná před tahem.
                @param game ukazatel na instanci Game
                @param move tah
                '''
                top=self.game.getPack('odkladani','all').top()
                if self.ace == True and top.value() != "eso":
                        self.ace=False
                if self.sevens > 0 and top.value() != "7":
                        self.sevens=0
                if self.j and top.value() != "svrsek":
                        self.j=False

        def afterMove(self,move):
                '''
                Metoda volaná po tahu.
                @param game ukazatel na instanci Game
                @param move tah
                '''
                if move.srcPack() == ("hand",move.player()) and move.dstPack() == move.srcPack():
                        return
                talon=self.game.getPack("talon","all").7
                odkladani=self.game.getPack("odkladani","all")
                top=odkladani.top()
                self.discarded=move.srcPack() == ("hand",move.player()) and move.dstPack() == ("odkladani","all")
                if len(talon.cards()) == 0:
                        self.reverseTalon()
                if self.discarded and top.value() == "svrsek":
                        self.j=True
                        self.suit=top.suit()
                        questText=(self.tr("Which suit do wou want change to?"))
                        suits=[self.tr("srdce"),self.tr("listy"),self.tr("zaludy"),self.tr("kule")]
```

```
                                answers=[]
                                for suit in suits:
                                        answers.append((suit,"text"))
                                def callback(question,playerFrom,answer):
                                        '''
                                        Obslužná funkce pro vyber na konci kola.
                                        '''
                                        question.rules.suit=question.suits[answer]
                                        question.game.sendInfo(question.game.players().keys(),"suit has been changed do
"+str(question.rules.suit))
                                        question.game.round.nextPlayer()
                                        self.game.delQuestion(question)

                                quest=Question(questText,answers,callback)
                                quest.rules=self
                                quest.suits=suits
                                quest.game=self.game
                                self.game.addQuestion(quest,move.player())
                                return

                        self.game.round.nextPlayer()
                        player=self.game.round.getTurn()
                        hand=self.game.getPack("hand",player)
                        if not self.discarded:
                                self.ace=False
                                self.sevens=0
                                return
                        if top.value() == "7":
                                self.sevens+=1
                                if not hand.hasCard(value="7"):
                                        # nema, lize
                                        if 2*self.sevens <= len(talon.cards()):
                                                newMove=Move(self.game,("talon","all",TOP),("hand",player,LEN),player)
                                                for i in range(2*self.sevens-1):
                                                        newMove.addMove(("talon","all",TOP),("hand",player,LEN))
                                                self.sevens=0
                                                self.beforeMove(newMove)
                                                newMove.accept()
                                                self.afterMove(newMove)
                                        else:
                                                n1=len(talon.cards())   # pocet karet ktere se presunou pred otocenim talonu
                                                n2=2*self.sevens-n1   # pocet karet ktere se presunou po otocenim talonu
                                                newMove=Move(self.game,("talon","all",TOP),("hand",player,LEN),player)
                                                for i in range(n1-1):
                                                        newMove.addMove(("talon","all",TOP),("hand",player,LEN))
                                                newMove.accept()
                                                self.reverseTalon()
                                                newMove=Move(self.game,("talon","all",TOP),("hand",player,LEN),player)
                                                for i in range(n2-12):
                                                        newMove.addMove(("talon","all",TOP),("hand",player,LEN))
                                                self.sevens=0
                                                self.beforeMove(newMove)
                                                newMove.accept()
                                                self.afterMove(newMove)


                                else:
                                        # ma nejake, dotaz zda chce vyhodit ci si liznout
                                        questText=(self.tr("What do you want to do?"))
                                        answers=[(self.tr("Take ")+str(self.sevens*2)+self.tr(" cards from talon"),"text")]
                                        sevens=hand.cardsWithValue("7")
                                        indexes=[None]
                                        for card in sevens:
                                                answers.append((self.tr("Discard ")+str(card.suitAndValue()),"text"))
                                                indexes.append(hand.cards().index(card))
                                        def callback(question,playerFrom,answer):
                                                '''
                                                Obslužná funkce pro vyber na konci kola.
                                                '''
                                                if answer == 0:
                                                        if 2*self.sevens <= len(talon.cards()):
                                                                newMove=Move(self.game,("talon","all",TOP),
("hand",playerFrom,TOP),player)
```

```
                                                                for i in range(2*question.rules.sevens-1):
                                                                        newMove.addMove(("talon","all",TOP),
("hand",playerFrom,TOP))
                                                        else:
                                                                n1=len(talon.cards())  # pocet karet ktere se presunou pred
otocenim talonu
                                                                n2=2*self.sevens-n1  # pocet karet ktere se presunou po
otocenim talonu
                                                                newMove=Move(self.game,("talon","all",TOP),
("hand",player,LEN),player)
                                                                for i in range(n1-1):
                                                                        newMove.addMove(("talon","all",TOP),
("hand",player,LEN))
                                                                newMove.accept()
                                                                self.reverseTalon()
                                                                newMove=Move(self.game,("talon","all",TOP),
("hand",player,LEN),player)
                                                                for i in range(n2-1):
                                                                        newMove.addMove(("talon","all",TOP),
("hand",player,LEN))
                                                        question.rules.sevens=0
                                                        question.game.round.nextPlayer()
                                                else:
                                                        newMove=Move(self.game,("hand",playerFrom,indexes[int(answer)]),
("odkladani","all",LEN),player)
                                                        newMove.reverse()
                                                self.beforeMove(newMove)
                                                newMove.accept()
                                                self.afterMove(newMove)
                                        quest=Question(questText,answers,callback)
                                        quest.rules=self
                                        quest.game=self.game
                                        self.game.addQuestion(quest,player)

                        elif top.value() == "eso":
                                self.ace=True
                                if not hand.hasCard(value="eso"):
                                        # nema, stoji
                                        self.game.sendInfo("You are staying this round.",player)
                                        self.game.round.nextPlayer()
                                        self.ace=False
                                else:
                                        # ma nejake, dotaz zda chce vyhodit ci stat
                                        questText=(self.tr("What do you want to do?"))
                                        answers=[(self.tr("Stay this round"),"text")]
                                        aces=hand.cardsWithValue("eso")
                                        indexes=[None]
                                        for card in aces:
                                                answers.append((self.tr("Discard ")+str(card.suitAndValue()),"text"))
                                                indexes.append(hand.cards().index(card))
                                        def callback(question,playerFrom,answer):
                                                '''
                                                Obslužná funkce pro vyber na konci kola.
                                                '''
                                                if answer == 0:
                                                        question.game.sendInfo("You are staying this round.",playerFrom)
                                                        question.game.rules.ace=False
                                                        question.game.round.nextPlayer()
                                                else:
                                                        newMove=Move(self.game,("hand",playerFrom,indexes[int(answer)]),
("odkladani","all",LEN),playerFrom)
                                                        newMove.reverse()
                                                        self.beforeMove(newMove)
                                                        newMove.accept()
                                                        self.afterMove(newMove)
                                        quest=Question(questText,answers,callback)
                                        quest.game=self.game
                                        self.game.addQuestion(quest,player)

        def reverseTalon(self):
                '''
                Otočí odkládací balík a přesune jej na práydný talon.
                Vrchní karta zůstane na odkládacím balíku.
```

```
                """
                odkladani=self.game.getPack("odkladani","all")
                talonMove=Move(self.game)
                odkladaniCount=len(odkladani.cards())
                talonMove.addReversePack(("odkladani","all"))
                talonMove.addMovePack(("odkladani","all"),("talon","all"))
                talonMove.addMove(("talon","all",0),("odkladani","all",0))
                talonMove.addReverse(("odkladani","all",0))
                talonMove.accept()

        def checkGoal(self):
                """
                Kontrola jestli již někdo vyhrál.
                @retval None Nikdo nevyhrál
                @retval jid hráč který vyhrál
                @todo da se to zprehlednit?
                """
                for pack in self.game.packs().items():
                        if pack[0][0] == "hand":
                                if len(pack[1].cards()) == 0:
                                        return pack[0][1]
                return None

        def eval(self,cards):
                """
                Vypočte součet hodnot karet.
                @param cards seznam karet (barva,hodnota)
                """
                sum=0
                for card in cards:
                        if card[1] == 'J':
                                sum+=2
                        elif card[1] == 'Q':
                                sum+=3
                        elif card[1] == 'K':
                                sum+=4
                        elif card[1] == 'eso':
                                sum+=11
                        else:
                                sum+=int(card[1])
                return sum

        def beforeRoundStart(self):
                # init variables
                self.ace=False          # odhozeno eso
                self.sevens=0           # pocet odhozenych sedmicek za ktere se berou karty]
                self.j=False# vynesen svrsek
                self.suit=""  # vynesena barva, pouziva se v pripade vyskytu svrska
                self.discard=False      # vynesena karta (True)

        def afterRoundEnd(self):
                pass

        def actions(self,playerFrom,source):
                """
                Obsluha akcí.
                Akce je kliknutí uživatele na kartu.
                @param playerFrom autor akce
                @param source zdroj jako (name,owner,index)
                @return Move instance doporučeného presunu, který se projde jako by přišlo od klienta <move/>
                """
                if source[:2] == ("hand",playerFrom):
                        #move=Move(self.game,source,("talon","all",LEN),playerFrom)
                        return ("odkladani","all",LEN)
                elif source[:2] == ("talon","all"):
                        #move=Move(self.game,source,("hand",playerFrom,LEN),playerFrom)
                        return ("hand",playerFrom,LEN)
                else:
                        return None

prsi.rules=Rules
prsi.start()
```