Žízeň

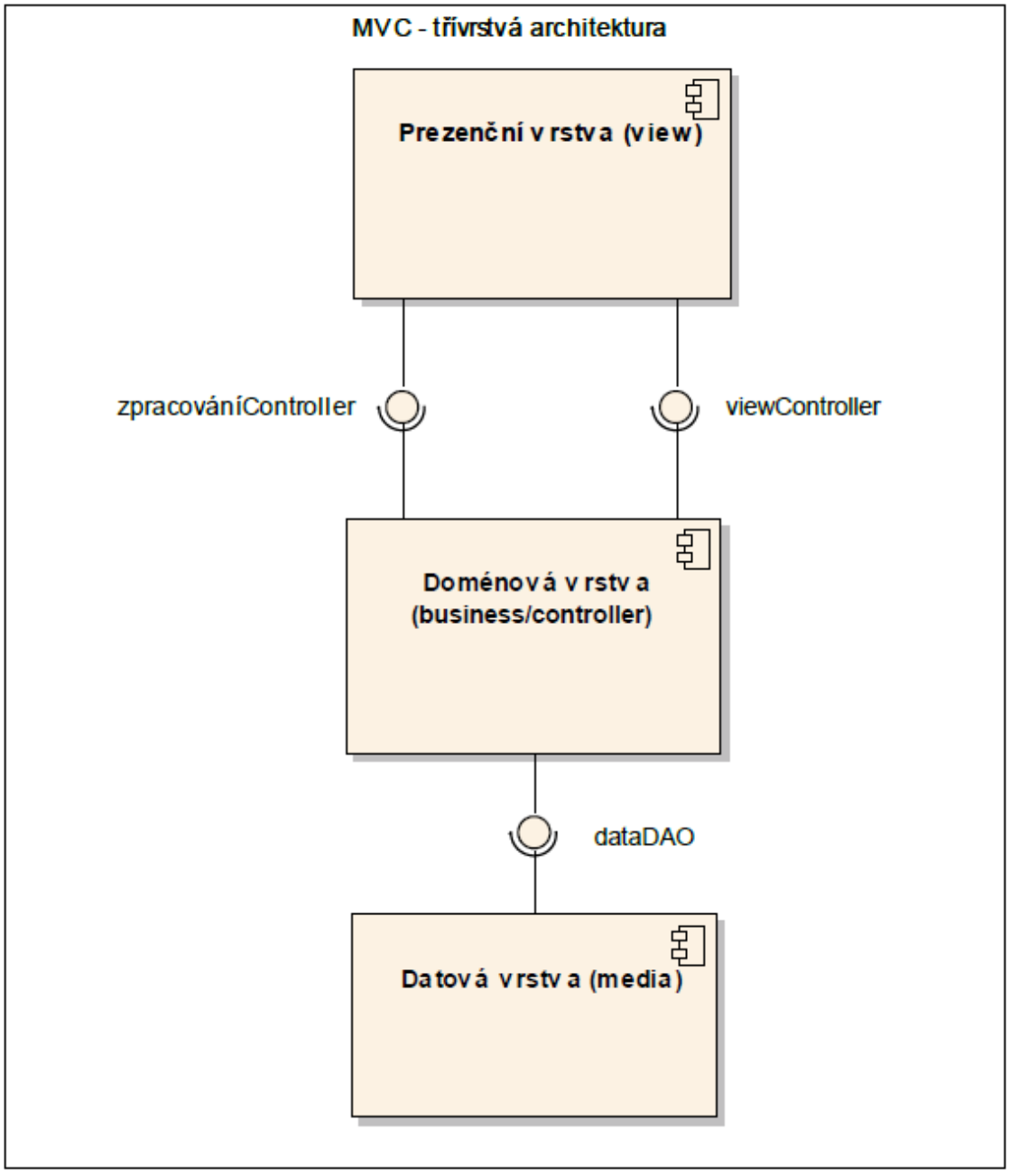
Semestrální projekt BI-ZSI
Návrhová dokumentace

Spolupracovali:
David Tošner
Martin Troup
David Kocík
Miro Hrončok

Model architektury

Architektura aplikace je navržena jako třívrstvá striktní. Tato architektura je již součástí frameworku Symfony 2.0, který jsme se rozhodli pro implementaci použít. Model používaný frameworkem se nazývá MVC (media – view – controller). Architektura rozčleňuje implementaci aplikace na tři části, což přináší mnohé výhody. Umožňuje především větší flexibilitu při změně databáze, či webového rozhraní, kdy stačí předělat pouze jednu z uvedených vrstev, ne tedy celou aplikaci. Další výhodou je možnost pracovat na aplikaci a testovat ji odděleně podle uvedených vrstev.

Diagram komponent



Prezenční vrstva

Prezenční vrstva obsahuje třídy, které se starají o prezentování informací uživateli. Informace, které od uživatele dostává dále posílá ke zpracování doménové vrstvě (viz. níže).

V této vrstvě dochází k další separaci kódu. Jelikož naše uživatelské GUI představuje webové rozhraní, které s jakoukoliv změnou zůstávají konzistentní (název aplikace, menu a další...) a informací, které se v interakci s rozhraním mění podle požadavků uživatele (info o aplikaci, zobrazení vlastních údajů a další...), máme k dispozici další dvě podvrstvy.

- layout = slouží k zobrazování konzistentních informací
- template = slouží k zobrazování nekonzistentních informací.

V aplikaci je možné použít více layout vrstev, nicméně je logické, že každá z nich bude obsahovat nějakou podmnožinu template vrstev.

Doménová vrstva

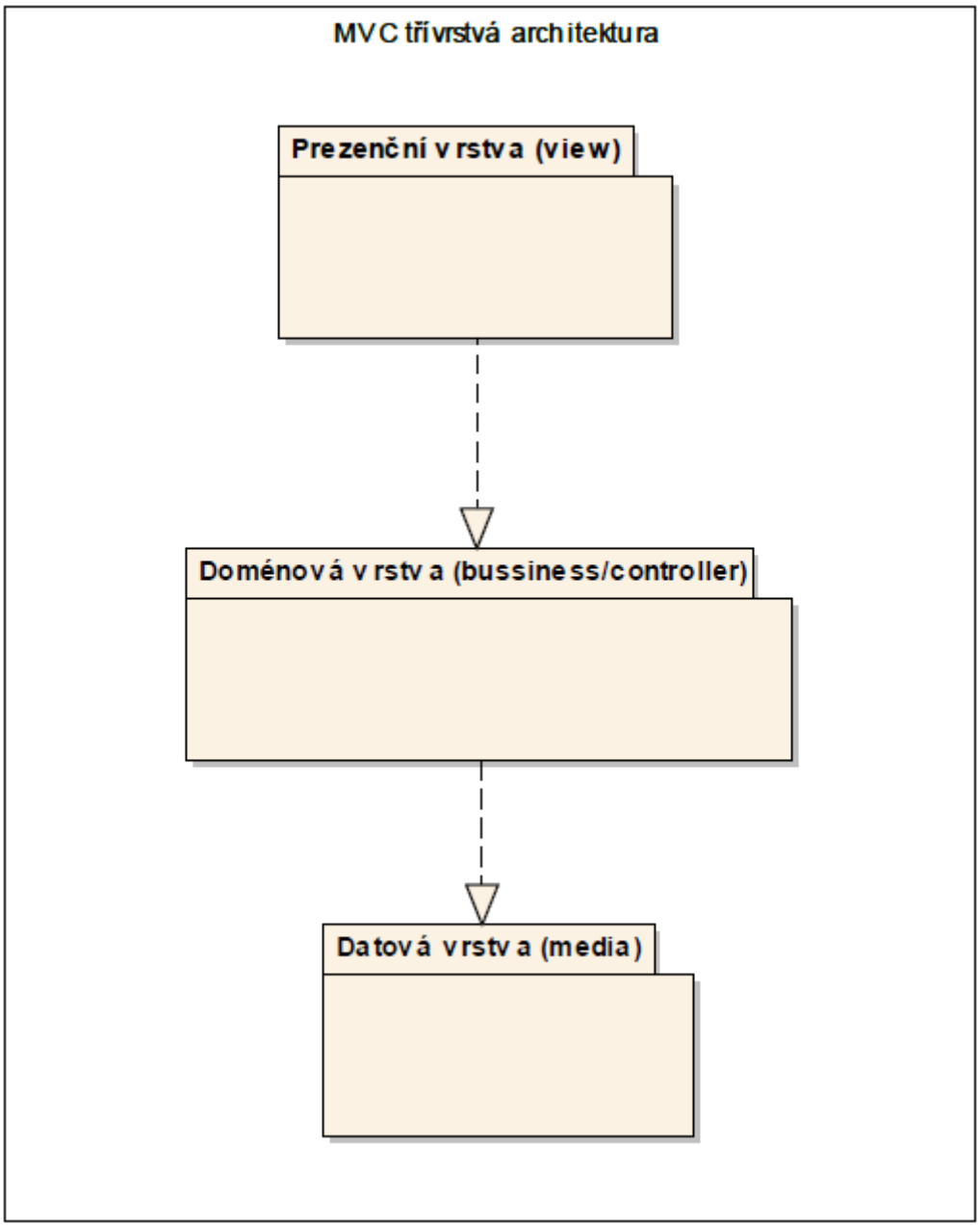
Doménová vrstva obsahuje třídy, které zpracovávají informace od prezenční vrstvy, podle kterých realizuje logiku aplikace. Pracuje jako prostředník mezi vrstvou prezenční a datovou (viz. níže).
Poté co přijme informace od vrstvy datové, pošle je zpět do vrstvy prezenční.

Datová vrstva

Datová vrstva obsahuje třídy, které zařizují komunikaci s databází. Přijímá tedy příkazy od vrstvy doménové a vrací jí požadované informace z databáze. Aby byla aplikace flexibilní na změnu databáze, je tato vrstva dále dělena do dvou podvrstev.

- **database abstraction layer** = zajistí připojení ke konkrétní databázi
- **data acces menu =** zajistí komunikaci nezávisle na konkrétní databázi

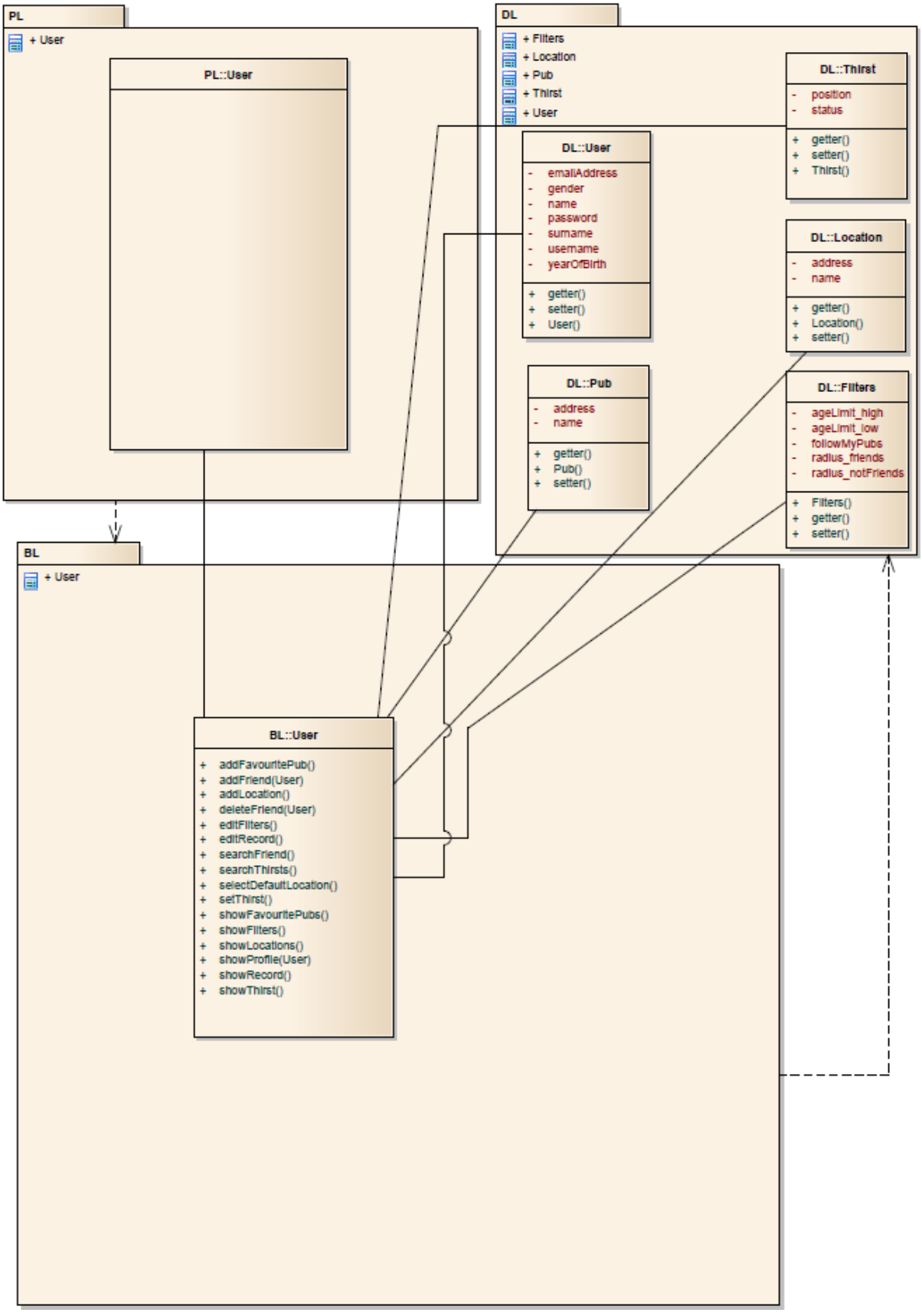
Diagram balíčků



Návrhový model

Model je vytvořen nad architekturou MVC. Třída User v prezentační vrstvě je závislá na množině tříd ve vrstvě doménové a ty sou dále závislé na množinu tříd z vrstvy datové. Množina tříd ve vrstvě doménové tvoří celou logiku. Veškeré důležité metody pro chod aplikace jsou uvedeny ve třídě User, která je tedy pro implementaci klíčová. Třídy v datové vrstvě zajišťují komunikaci s databází a předávají data vrstvě doménové. V prezentační vrstvě bude implementováno GUI pro uživatele, čili funkce, které budou moci využívat. Pro implementaci není prezentační vrstva klíčová, proto budou metody její třídy User definovány později.

Celkový diagram



Popis funkcí jednotlivých tříd

Class BL::User

Třída implementující akce uživatele. Třída umožňuje přístup k datové vrstvě a provádí činnosti zadané uživatelem. Je to nejdůležitější třída celé aplikace.

Atribut	Popis
AddFavouritePub()	Funkce zobrazí zadávací formulář v prezenční vrstvě, následně přidá hospodu do seznamu přihlášeného uživatele v datové vrstvě a zobrazí potvrzení do prezenční vrstvy. **Parametry:**
addFriend(User)	Přidá vybraného uživatele do seznamu přátel v datové vrstvě a zobrazí potvrzení do prezenční vrstvy. **Parametry:** • **User:** Instance uživatele, který má být přidán.
addLocation()	Funkce zobrazí zadávací formulář v prezenční vrstvě, následně přidá lokaci do seznamu přihlášeného uživatele v datové vrstvě a zobrazí potvrzení do prezenční vrstvy. **Parametry:**
deleteFriend(User)	Odstraní vybraného uživatele ze seznamu přátel v datové vrstvě a zobrazí potvrzení do prezenční vrstvy. **Parametry:** • **User:** Instance uživatele, který má být odstraněn.
editFilters()	Funkce umožní přihlášenému uživateli upravit filtry pro vyhledávání ostatních sezení žízně. Nová data jsou poslána do datové vrstvy k uložení a poté jsou zobrazena v prezenční vrstvě. **Parametry:**
editRecord()	Funkce umožní přihlášenému uživateli upravit údaje o vlastní osobě. Nová data jsou poslána do datové vrstvy k uložení a poté jsou zobrazena v prezenční vrstvě. **Parametry:**
searchFriends(name, surname)	Pomocí zadaných parametrů name a surname funkce vyhledá relevantní uživatele a zobrazí jejich seznam. **Parametry:** • **name:** Obsahuje jméno hledaného uživatele. • **surname:** Obsahuje příjmení hledaného uživatele.
searchThirsts()	Pomocí atributů třídy Filters funkce vyhledá relevantní žízně ostatních uživatelů a zobrazí jejich seznam. **Parametry:**
selectDefaultLocation()	Funkce umožní přihlášenému uživateli vybrat základní lokaci a pozmění údaje v datové vrstvě. **Parametry:**
setThirst()	Funkce umožní přihlášenému uživateli nastavit sezení žízně.

	Nastaví údaje v datové vrstvě a zobrazí je v prezenční vrstvě. **Parametry:**
showFavouritePubs()	Funkce zobrazí přihlášenému uživateli instance všech jeho oblíbených hospod. **Parametry:**
showFilter()	Funkce zobrazí přihlášenému uživateli filtry pro vyhledávání ostatních sezení žízně. **Parametry:**
showLocations()	Funkce zobrazí přihlášenému uživateli instance všech jeho vytvořených lokací. **Parametry:**
showProfile(User)	Zobrazí přihlášenému uživateli údaje o vybrané instanci uživatele. **Parametry:** • **User:** Instance uživatele, který má být zobrazen.
showRecord()	Funkce zobrazí přihlášenému uživateli údaje o vlastní osobě. **Parametry:**
showThirst()	Funkce zobrazí přihlášenému uživateli údaje o vlastní žízni. **Parametry:**
User()	Implicitní konstruktor. **Parametry:**

Class DL::Location

Třída reprezetující lokace. Představuje instance lokací, které si uživatel může přidat do svého profilu. Všechny členské proměnné jsou nastavovány a získávány funkcemi viz. getAdress a setAdress.

Atribut	**Popis**
getAddress()	Vrátí hodnotu třídní proměnné address. **Parametry:**
setAddress()	Nastaví hodnotu třídní proměnné address. **Parametry:** • **param1:** data, která se uloží do parametru.
Location()	Implicitní konstruktor. **Parametry:**

Class DL::Pub

Třída reprezetující hospodu. Představuje instance hospod, které si uživatel může přidat do svého profilu. Všechny členské proměnné jsou nastavovány a získávány funkcemi viz. getAdress a setAdress.

Atribut	Popis
getAddress()	Vrátí hodnotu třídní proměnné address. **Parametry:**
setAddress()	Nastaví hodnotu třídní proměnné address. **Parametry:** • **param1:** data, která se uloží do parametru.
Pub()	Konstruktor nastaví hodnoty třídních proměnných. **Parametry:**

Class DL::Thirst

Třída reprezentující žízeň. Instance žízně obsahuje údaje o žízni. Všechny členské proměnné jsou nastavovány a získávány funkcemi viz. getPosition a setPosition.

Atribut	Popis
getPosition()	Vrátí hodnotu třídní proměnné position. **Parametry:**
setPosition()	Nastaví hodnotu třídní proměnné position. **Parametry:** • **param1:** data, která se uloží do parametru.
Thirst()	Konstruktor nastaví hodnoty třídních proměnných. **Parametry:**

Class DL::Filters

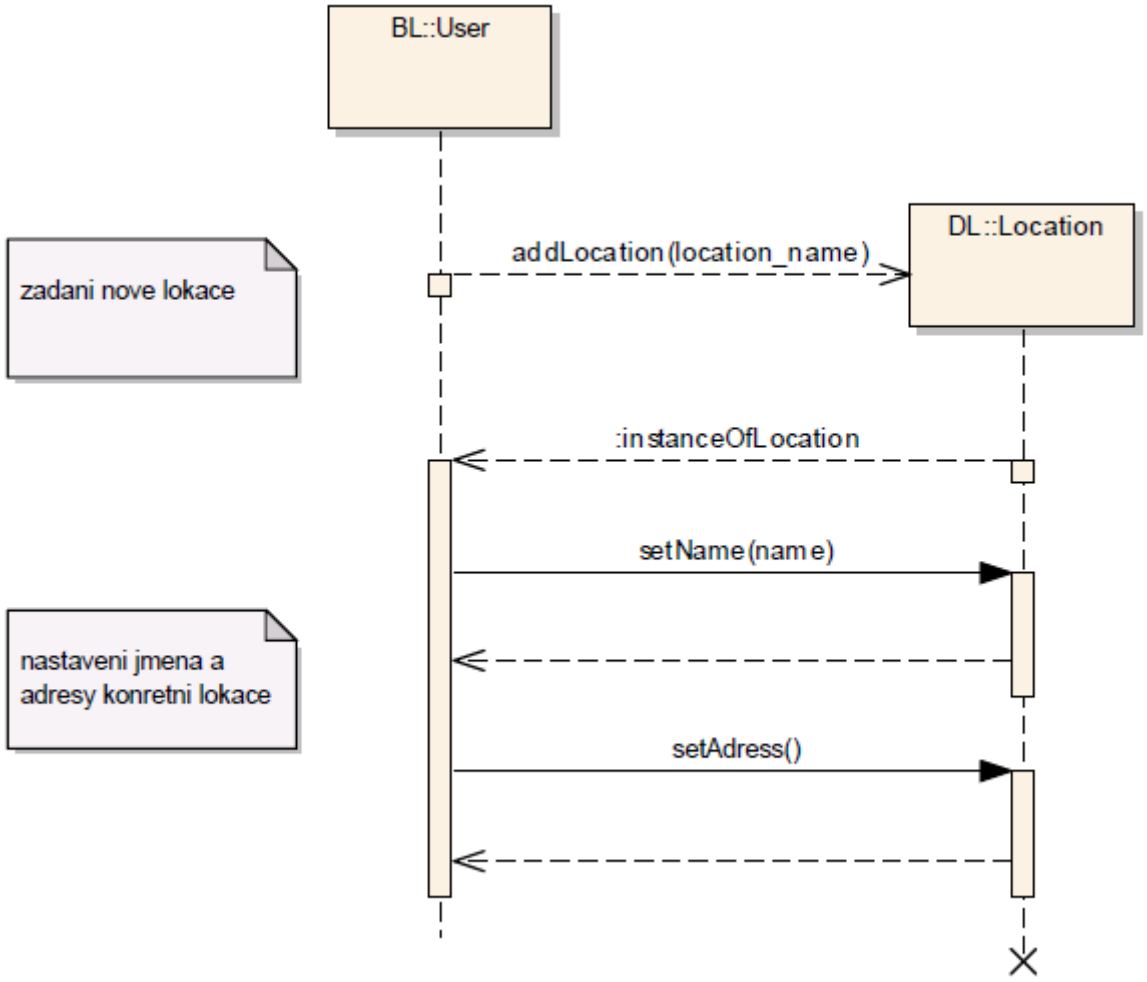
Třída reprezetující filtry k vyhledávání instancí žízní. Všechny členské proměnné jsou nastavovány a získávány funkcemi viz. getAgeLimit_high a setAgeLimit_high.

Atribut	Popis
getAgeLimit_high	Vrátí hodnotu třídní proměnné ageLimit_high. **Parametry:**
setAgeLimit_high	Nastaví hodnotu třídní proměnné ageLimit_high. **Parametry:** • **param1:** data, která se uloží do parametru.
Filters()	Konstruktor nastaví hodnoty třídních proměnných. **Parametry:**

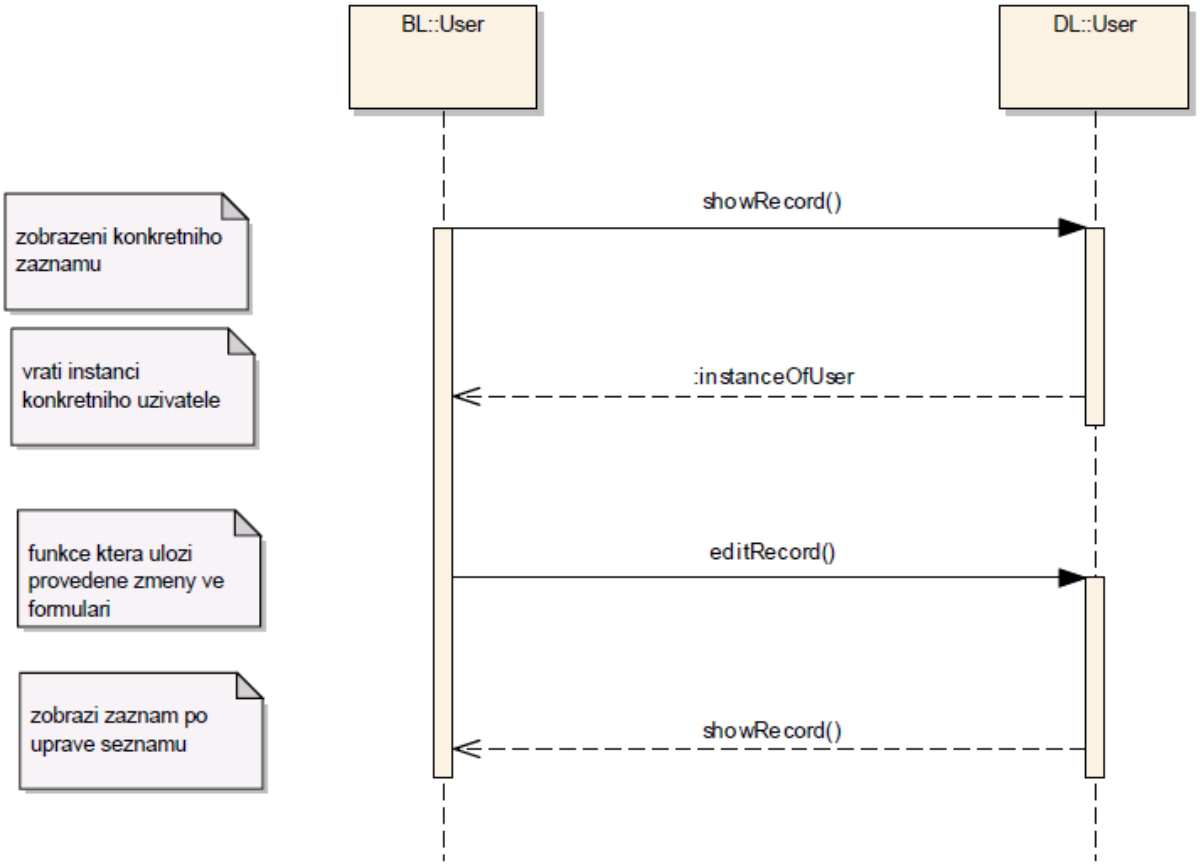
Model komunikace

Kapitola popisuje přiřazení zodpovědnosti tříd při realizaci požadovaných funkčností. Jednotlivé třídy byli popsány výše. V této kapitole jsou obsaženy pouze popisy spolupráce tříd, které jsou z pohledu objektově orientovaného navrhu zajímavé. Jsou zachycené pomocí sekvenčních diagramů. Jejich je obsažen přímo v poznámkách u grafů

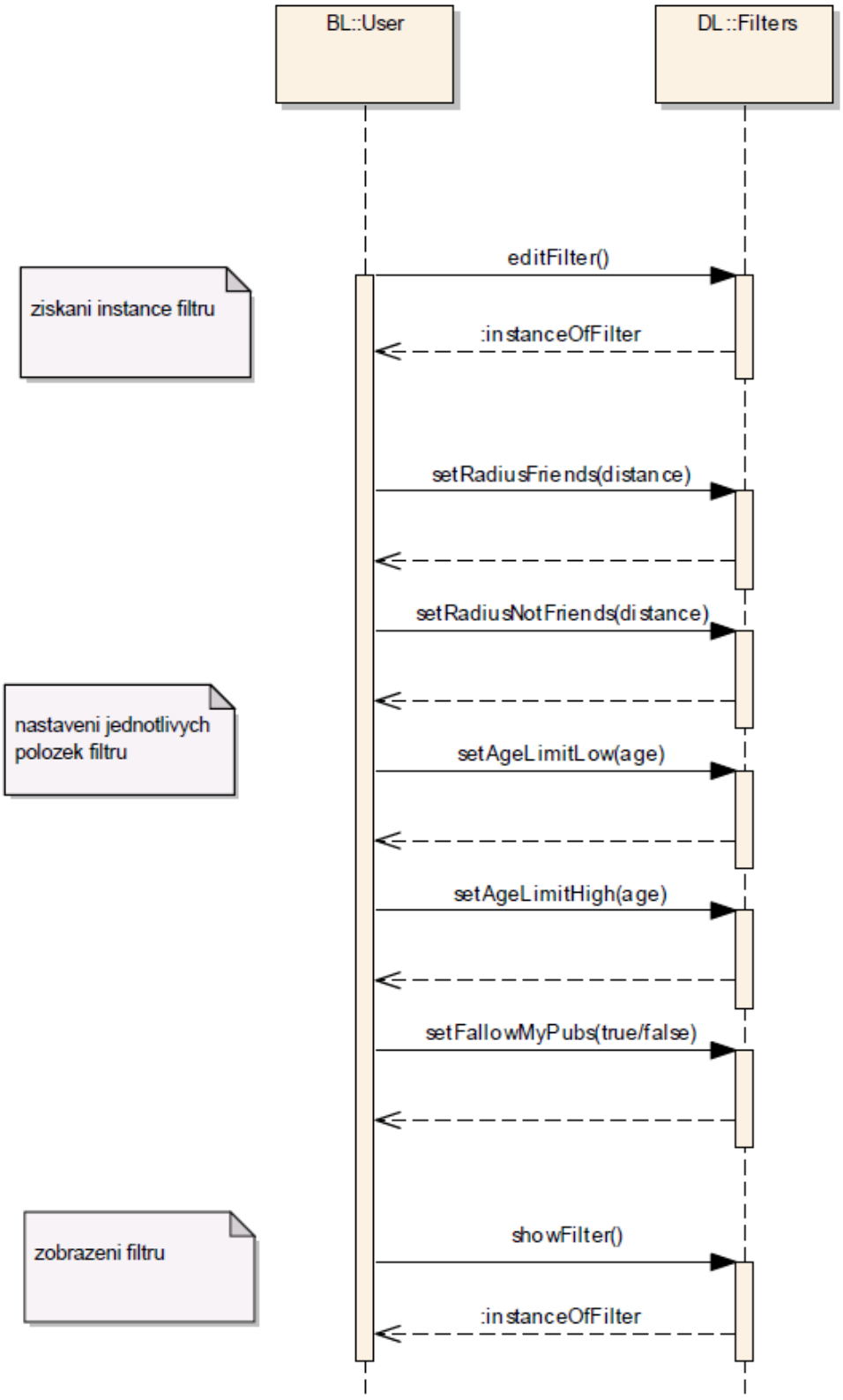
Zadání záznamu lokace



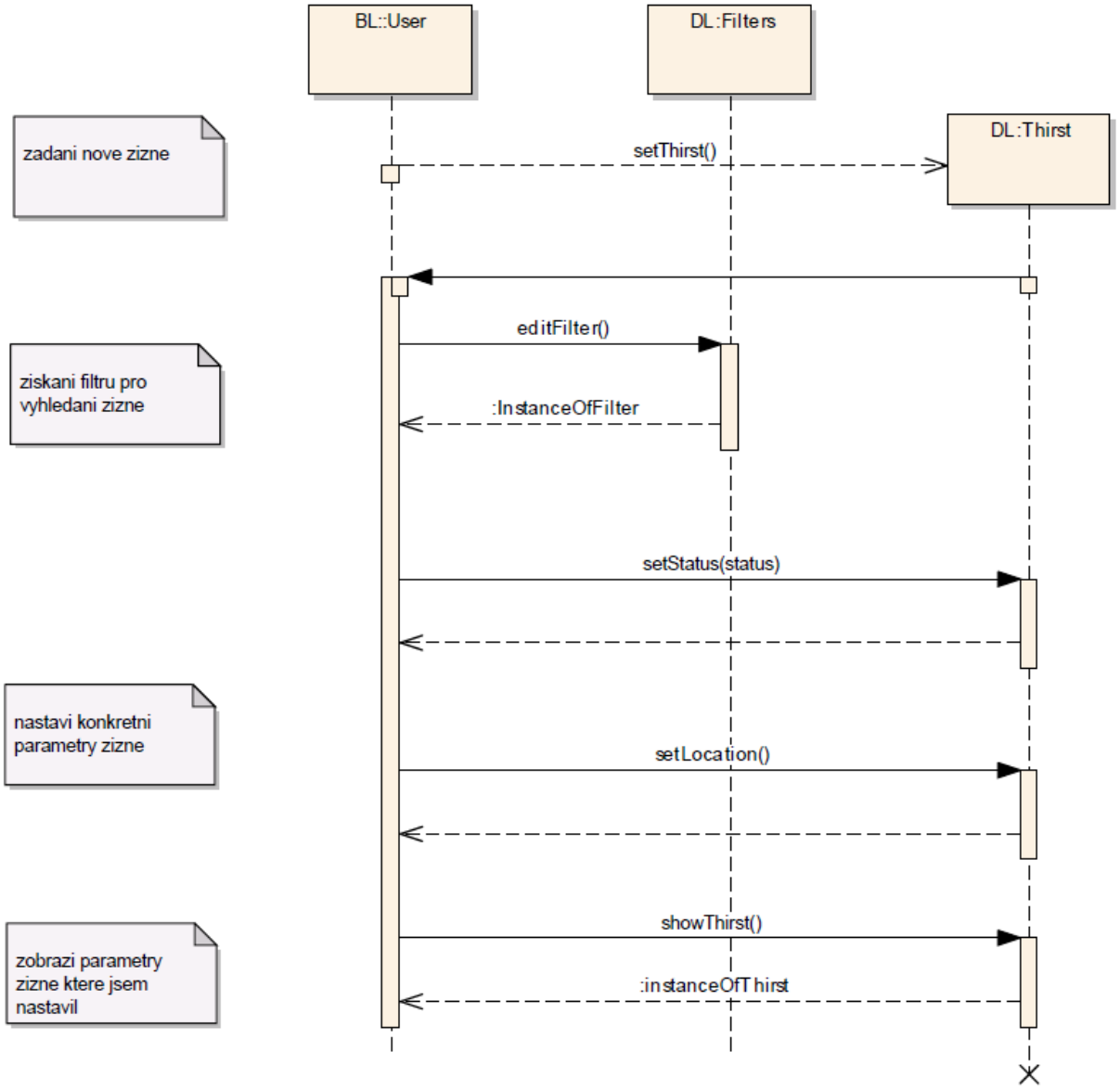
Upravení osobních údajů



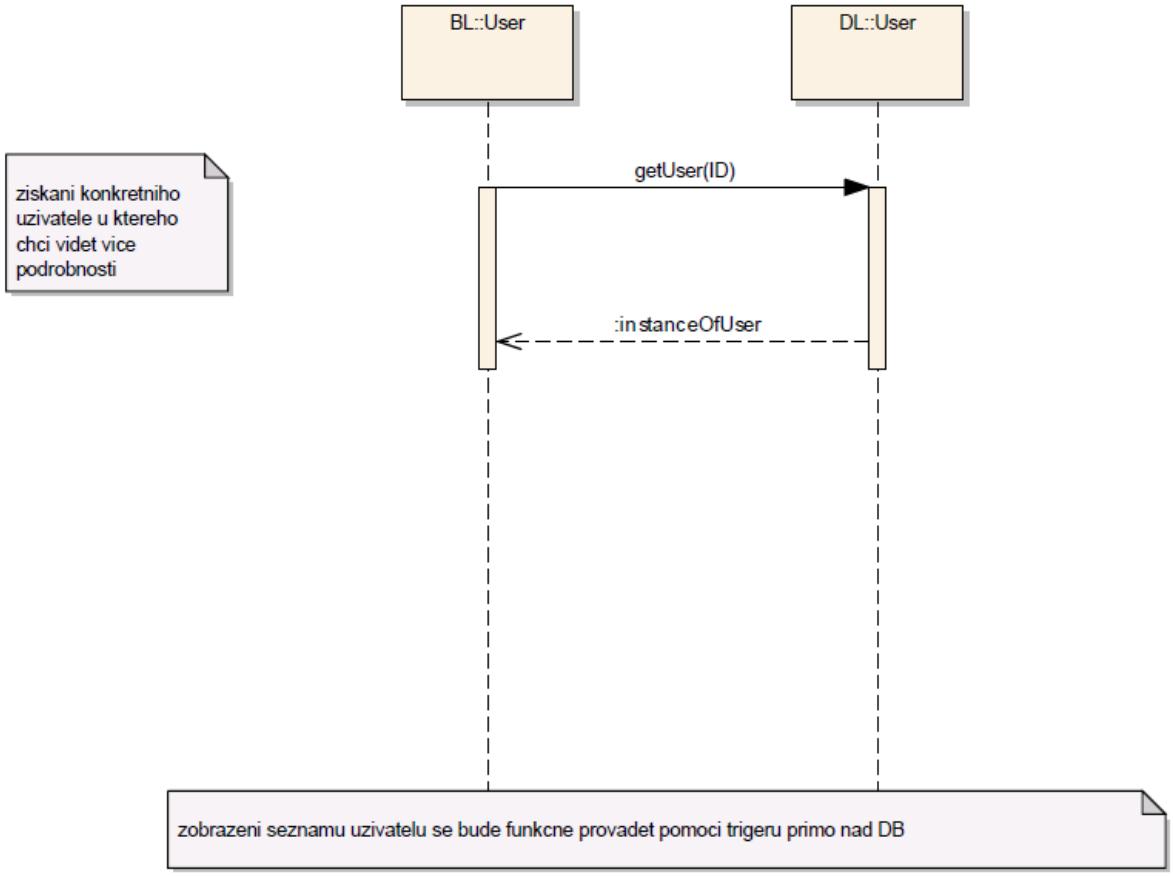
Úprava filtrů pro hledání žízně



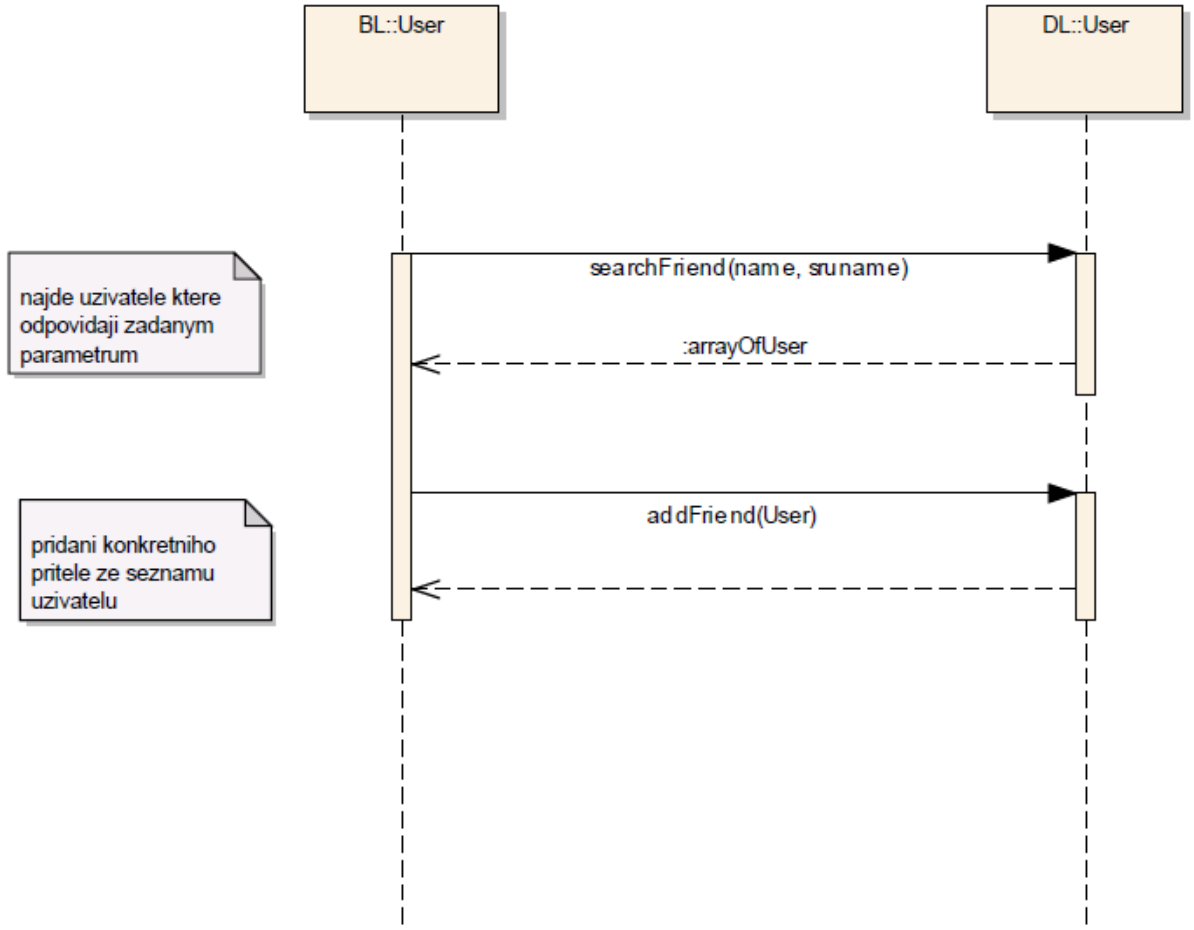
Zadání požadavku žízně



Zobrazení podrobností o uživateli



Přidání přítele do seznamu

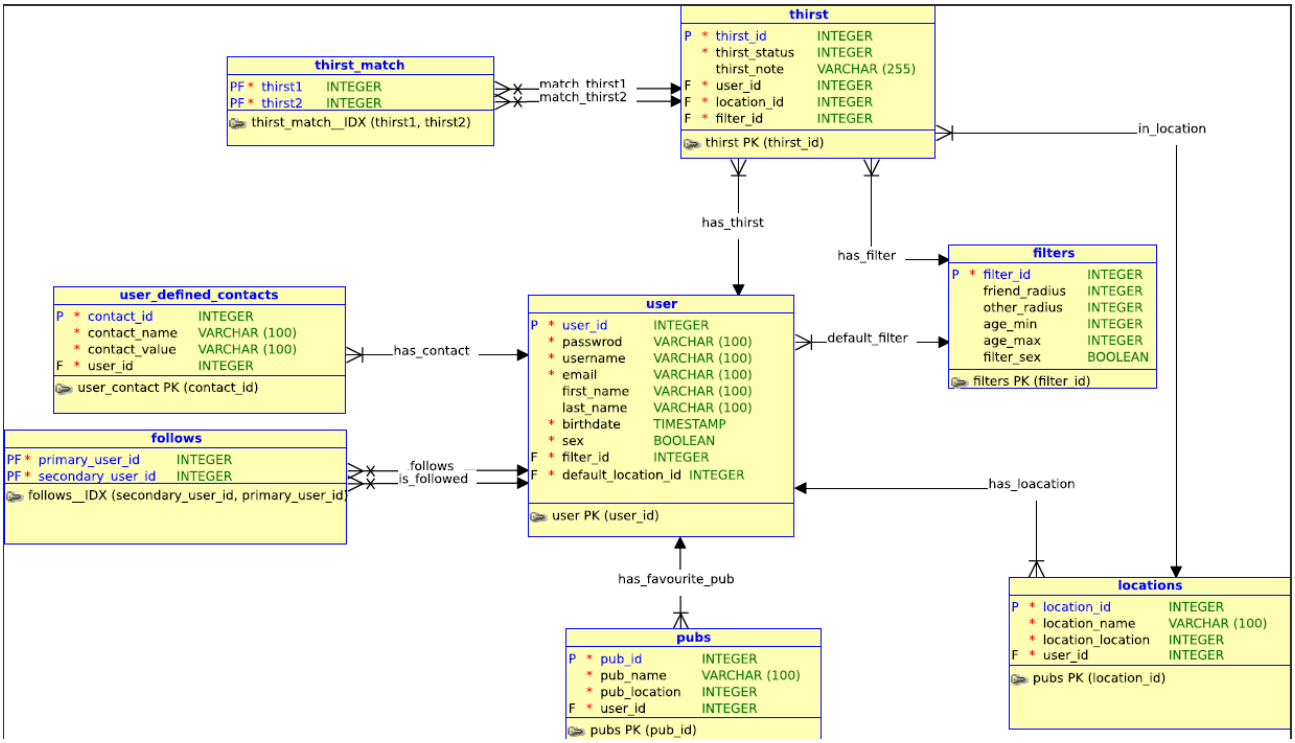


Relační databázový model

Nad databází bude napsán trigger, který po změně obsahu tabulky thirst zajistí aktualizaci obsahu tabulky thirst_match

Více uživatelů může mít stejný filtr -- po vytvoření bude uživatel mít nastaven jako výchozí filtr filtr základní, který se zkopíruje jako nový při vytváření nové žízně

Hospoda má přiřazeného uživatele, z toho vyplývá, že pokud dva uživatelé budou mít stejnou oblíbenou hospodu, bude v databázi více záznamů s hospodou. Umožní to každému uživateli kdykoli změnit název hospody, nebo její pozici, aniž by poškodil ostatní uživatele.



Jednotlive tabulky a jejich atributy

Tabulka user

user		Tabulka uživatelů
user_id	INTEGER	Číselné ID uživatele
password	VARCHAR(100)	Zahashované heslo
username	VARCHAR(100)	Přihlašovací jméno uživatele
email	VARCHAR(100)	E-mailová adresa uživatele
first_name	VARCHAR(100)	Křestní jméno uživatele
last_name	VARCHAR(100)	Příjmení uživatele
birthdate	TIMESTAMP	Narozeniny uživatele
sex	BOOLEAN	Pohlaví uživatele
filter_id	INTEGER	Číselné ID výchozího filtru
Default_location_id	INTEGER	ID výchozí lokace

Tabulka follows

follows		Tabulka sledování mezi uživateli
primary_user_id	INTEGER	Číselné ID uživatele, který sleduje druhého uživatele
secondary_user_id	INTEGER	Číselné ID uživatele, který je sledován prvním uživatelem

Tabulka user_defined_contacts

user_defined_contacts		Tabulka uživateli definovaných kontaktních informací
contact_id	INTEGER	Číselné ID uživatelem definované kontaktní informace
contact_name	VARCHAR(100)	Pojmenování uživatelem definované kontaktní informace
contact_value	VARCHAR(100)	Hodnota uživatelem definované kontaktní informace
user_id	INTEGER	Číselné ID uživatele, ke kterému tento kontakt patří

Tabulka pubs

pubs		Tabulka oblíbených hospod uživatelů
pub_id	INTEGER	Číselné ID hospody
pub_name	VARCHAR(100)	Název hopsody
pub_location	INTEGER	Souřadnice hospody (datový typ bude upraven/rozdělen podle konkrétní implemetace)
user_id	INTEGER	Číselné ID uživatele, který tuto hospodu má ve svých oblíbených (stejné hospody budou uloženy vícekrát)

Tabulka locations

locations		Tabulka uložených lokací uživatelů
location_id	INTEGER	Číselné ID lokace
location_name	VARCHAR(100)	Názve lokace definovaný uživatelem
location_location	INTEGER	Souřadnice lokace (datový typ bude upraven/rozdělen podle konkrétní implemetace)
is_default	BOOLEAN	TRUE pokud je tato lokace výchozí pro uživatele (integritní omezení: jej jedna)
user_id	INTEGER	Číselné ID uživatele, ke kterému patří tato lokace

Tabulka filters

filters		Tabulka filtrů
filter_id	INTEGER	Číselné ID filtru
friend_radius	INTEGER	Poloměr hledání jiných žíznivců, kamarádů (stačí když jeden uživatel "sleduje" druhého) v metrech, NULL pro zakázání kamarádů
other_radius	INTEGER	DTTO u ostatních uživatelů
age_max	INTEGER	Věkový strop (včetně), NULL horní limit vypne
age_min	INTEGER	Věkový spodní limit, NULL spodní limit vypne, případně aplikuje systémový limit (např. plnoletost)
filter_sex	BOOLEAN	Filtr na pohlaví, hodnota určuje pohlaví, NULL vypíná filtr

Tabulka thirst

thirst		Tabulka žízní
thirst_id	INTEGER	Číselné ID žízně
thirst_status	INTEGER	Číselné určení stavu (aktivní, neaktvní, v hospodě, ...)
thirst_note	VARCHAR(255)	Nepovinný text u žízně
user_id	INTEGER	Číselné ID uživatele, který žízeň vytvořil
location_id	INTEGER	Číselné ID lokace, kterou uživatel žízni přiřadil
filter_id	INTEGER	Číselné ID filtru, který byl použit na žízeň

Tabulka thirst_match

thirst_match		Tabulka obsahuje páry pasujících žízní, plní se triggerem
thirst1	INETGER	Číselné ID žízně učastnící se páru
thirst2	INETGER	DTTO, hodnoty jsou rovnocené, tabulku je třeba prohledávat postupně podle obou sloupců