VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2016 Bc. Vladimír Hůla

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

HERNÍ WEBOVÝ PORTÁL - OBJEKTOVÉ PROGRAMOVÁNÍ

GAME WEB PORTAL - OBJECT-ORIENTED PROGRAMMING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Vladimír Hůla

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Lukáš Povoda

BRNO 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

Student: Bc. Vladimír Hůla ***ID:*** 134502
Ročník: 2 ***Akademický rok:*** 2015/16

NÁZEV TÉMATU:

Herní webový portál - objektové programování

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je navrhnout a implementovat programovou strukturu pro herní webový portál, která pokryje potřeby hráčů a správců stránek (podrobnosti konzultujte s vedoucím práce). Návrh student odůvodní a k práci doloží UML diagramy vybraných struktur (upřesní vedoucí práce). Dále implementuje systém DKP (dragon kill points) a zaměří se na komfort návštěvníků portálu implementací aplikačních rozhraní jiných služeb (přihlášení pomocí protokolu OAuth, načítaní statistik hráče). Návrh zaměří tak, aby bylo možno všechny časti webu lokalizovat do jakéhokoliv jazyka. Student zároveň zpracuje rešerši o použitých technologiích a návrhových vzorech vhodných pro řešení tohoto problému.

DOPORUČENÁ LITERATURA:

[1] VAN DUYNE, Douglas K, James A LANDAY a Jason I HONG. The design of sites: patterns for creating winning web sites. 2nd ed. Upper Saddle River, N.J.: Prentice-Hall, c2007, xli, 981 s.

[2] FUNK, Tom. Web 2.0 and beyond: understanding the new online business models, trends, and technologies. Westport, Conn.: Praeger, 2009, xviii, 172 p. ISBN 0313351872.

Termín zadání: 1.2.2016 ***Termín odevzdání:*** 25.5.2016

Vedoucí práce: Ing. Lukáš Povoda
Konzultant diplomové práce:

doc. Ing. Jiří Mišurec, CSc., *předseda oborové rady*

ABSTRAKT

Tato práce se zabývá návrhem a implementací programové struktury herního webového portálu. Tento webový portál bude sloužit jako komunikační a informační centrum, které hráčům usnadní vzájemnou koordinaci a umožní jim získávat nové zkušenosti. V práci je proveden rozbor potřeb hráčů, již existujících řešení a jejich nedostatků. Z výsledků toho rozboru jsou navrženy jednotlivé funkce portálu. Realizace nejdůležitějších částí portálu byla popsána, implementace těchto částí byla zhodnocena a případně bylo navrženo, jak danou část vylepšit. V závěru práce bylo navrženo několik funkcionalit o které by se mohl portál v budoucnu rozšířit.

KLÍČOVÁ SLOVA

Herní webový portál, programová struktura, objektové programování, návrhové vzory, API, protokol OAuth, lokalizace, DKP, PHP, Nette framework, Doctrine

ABSTRACT

This thesis deals with design and implementation of programming structure of game web portal. The web portal will serve as a communication and information center to simplify coordination among players and allow them to gain new experience. In the thesis are analyzed the needs of players, existing solutions and their drawbacks. The results of this analysis are used to design individual functions of the portal. Implementation of the most important parts of the website has been described, the implementation of these parts were evaluated and some enhancemets were eventually suggested. In the end of this thesis several functionalities were suggested, which could extend the portal in the future.

KEYWORDS

Game web portal, program structure, object-oriented programming, design patterns, API, OAuth protocol, localization, DKP, PHP, Nette framework, Doctrine

HŮLA, Vladimír *Herní webový portál - objektové programování*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2016. 69 s. Vedoucí práce byl Ing. Lukáš Povoda

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Herní webový portál - objektové programování“ jsem vypracoval(a) samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Lukáši Povodovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora(-ky)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .

podpis autora(-ky)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OP Výzkum a vývoj
pro inovace

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM VÝPISŮ

ÚVOD

Cílem této práce je navrhnout a implementovat programovou strukturu pro herní webový portál, který bude sloužit herní komunitě jako zdroj snadno přístupných informací a jako místo pro vzájemnou komunikaci.

Většina her v sobě neimplementuje nástroje pro snadnou organizaci událostí a nepřímou komunikaci uvnitř herních komunit. Hráči poté musí tyto chybějící nástroje řešit sami pomocí externích aplikací v podobě diskusních fór, blogů, organizačních kalendářů nebo vlastních programových řešení. V mnohých případech jsou tato řešení nepřehledná nebo neobsahují informace o jejich aktuálnosti, což se týká převážně návodů a článků.

Úvodní část práce se zabývá rozborem potřeb hráčů, možnými způsoby řešení popsaných problémů, již existujícími řešeními a jejich nedostatky. Dále jsou stručně popsány technologie, které jsou použity pro realizaci herního portálu. Jedná se o frameworky Nette, Doctrine a Bootstrap. Součástí této práce je rovněž rešerše o návrhových vzorech vhodných pro návrh systému. Samotný návrh systému herního portálu se zabývá jednotlivými komponentami, lokalizací a základním rozvržením front-endu. V kapitole 5 jsou popsána použitá aplikační rozhraní, která jsou implementována do systému pro zvýšení komfortu uživatelů portálu. V závěru práce byly popsány nejdůležitější realizované části portálu mezi které patří programová struktura modelu, implementace řízení přístupových práv, znovupoužitelné formuláře, vlastní komponenty pro vykreslování formulářů, drobečkové navigace a hlavní navigace, implementace lokalizace a systému DKP.

1 TEORETICKÁ ČÁST PRÁCE

Rozvoje internetu a informačních technologií využil i herní průmysl a drtivá většina dnešních her podporuje hraní více hráčů po internetu. Jedná se o klasické hry s podporou více hráčů, ale i o tzv. MMO hry, kdy na jednom serveru hrají i tisíce hráčů najednou. Pravděpodobně nejznámějším a největším zástupcem MMO her je World of Warcraft, který po deseti letech provozu má stále více než 5 milionů předplatitelů, viz obr. 1.1.

Online hry hrají miliony lidí po celém světě. Účelem těchto her je mimo plnění různých úkolů a cílů i vzájemná interakce hráčů. Drtivá většina těchto hráčů se sdružuje do různých skupin za účelem snadnějšího dosažení svých cílů. Tyto skupiny mají různá pojmenování v závislosti na konkrétní hře, či jejím žánru, například guilda, klan nebo aliance. Jejich funkce však zůstává stejná.

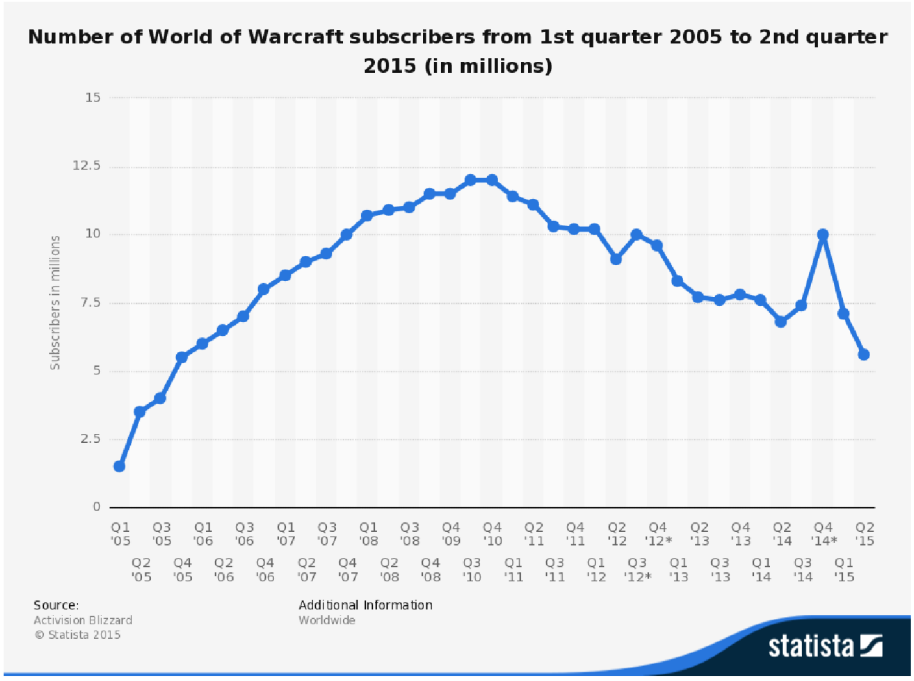
Komunikace mezi hráči je v jednotlivých hrách realizována pomocí chatu, případně jednoduché pošty nebo VoiP. Tento způsob komunikace je dostačující pouze v případě, že jsou hráči připojeni přímo ve hře. Pro organizaci různých událostí, které se mají konat v určitý čas, např. tréningy nebo zápasy, je tento způsob komunikace nevhodný. Udržování různých textových informací v podobě návodů, vzkazů a informací ve většině her nelze řešit vůbec, případně jen velmi obtížně.

Tyto nedostatky se projevují mnohem více v případě MMORPG her. Zde dosahují guildy počtu i stovek členů a samotné hry jsou koncipovány tak, že pro splnění některých úkolů je třeba kooperace většího počtu hráčů. U takto velkých uskupení hráčů je prakticky nutné používat externí řešení a doplnit tak funkcionality, které ve hrách chybí. Ideální platformou pro tato řešení je web, protože je nezávislý na použitém zařízení a operačním systému. Pro přístup vyžaduje pouze prohlížeč, což je další nesporná výhoda tohoto řešení.

Většina guild využívá nástroje, které jsou volně dostupné a zdarma. Z tohoto důvodu jsou většinou všechny textové informace řešeny pomocí diskusního fóra, kde jsou vedle diskusí vytvořena témata s návody a různými informacemi. Diskusní fórum však pro tento účel není vhodné řešení a velice často se stává nepřehledným. Zásadní nevýhodou je nekompatibilita mezi použitými nástroji. Například diskusní fórum je realizováno pomocí systému A, kalendář pro plánování událostí je na systému B a správa *dragon kill points* - DKP (více informací v kapitole 4.3) na systému C. Největší nevýhodou tohoto řešení je, že nelze (nebo jen velice těžko) v těchto systémech využívat jen jeden uživatelský účet. Všichni členové guildy se pak musí registrovat do třech různých systémů, čímž se správa těchto systémů stává velice obtížná.

Cílem této práce je vytvořit webový portál, který bude řešit výše zmíněné problémy a umožní pohodlnou koordinaci všech hráčů. Hráči budou mít k dispozici sadu

nástrojů vytvořených na míru jejich potřebám, jako jsou články, návody, diskusní fóra, galerie nebo kalendář událostí. Zároveň se bude jednat o informační centrum, kde všichni hráči mohou uveřejňovat své články, návody a dělit se o své zkušenosti s ostatními. Při návrhu byl kladen velký důraz na hráče MMO/MMORPG her, kteří jednotlivé funkce portálu využijí nejvíce. Z tohoto důvodu je celá práce koncipována pro pokrytí potřeb těchto uživatelů portálu.



Obr. 1.1: Vývoj počtu předplatitelů World of Warcraft [19].

1.1 Existující řešení

Již existujících řešení lze nalézt mnoho. Mezi nejúspěšnější z nich lze zařadit projekty Enjin [8] a iClan Websites [10].

Oba výše zmíněné projekty podporují velké množství her a nabízejí širokou škálu různých doplňků a komponent vytvořených na míru dle potřeb hráčů. Pro odlišení svých stránek mohou uživatelé využít velké množství předpřipravených šablon, které mohou dále přizpůsobovat podle svých potřeb. Dostupnost těchto doplňků a šablon je omezena. V základním balíčku, který je zdarma jsou dostupné jen ty nejdůležitější funkce. Další funkce jsou již zařazeny do zpoplatněných balíčků, které jsou nabízeny formou měsíčního předplatného. Nejdražší balíčky obsahují i možnost využití VoiP v podobě služeb Ventrilo, TeamSpeak nebo Mumble.

1.1.1 iClan Websites

IClan Websites se specializuje na guild hosting a nabízí uživatelům mnoho propracovaných funkcí, jako je správce zápasů nebo propracovaná editace šablon. Každý guild web je tedy nezávislý web hostovaný touto službou. Předplatné se u této služby pohybuje v rozmezí 5-33 dolarů za měsíc. Největší nevýhodou této služby je dle mého názoru absence globálního účtu, pomocí kterého by bylo možné se přihlašovat do více guild webů, a omezení počtu vytvořených stránek s obsahem na webu, které je uplatněno i v případě základní varianty předplatného.

1.1.2 Enjin

Projekt Enjin se nezabývá pouhým guild hostingem, ale tvoří jakousi herní sociální síť, kde každý registrovaný uživatel má svůj profil a seznam přátel/kontaktů. Do jednotlivých stránek guild a herních komunit se poté uživatelé přihlašují pomocí účtu na Enjinu. Jednotlivé nabízené funkce jsou zde označovány jako moduly. Stejně jako projekt iClan Websites, je i zde použití jednotlivých modulů omezeno pomocí různých plánů měsíčního předplatného. Zde se ceny pohybují od 9 do 30 dolarů za měsíc. Největší nevýhodu opět vidím v omezení celkového počtu modulů, které lze na guild webu zdarma použít a rovněž v absenci modulu pro správu událostí a DKP v bezplatné variantě hostingu.

1.2 Výhody navrhovaného řešení

Obě výše zmíněná řešení jsou velice propracovaná a nabízí uživatelům velké množství funkcí. Jejich nedostatkem je však celková provázanost uživatelských účtů a absence druhé vrstvy portálu, která by sdružovala informace, články a návody o jednotlivých hrách. V případě projektu Enjin lze toto částečně vytvořit pomocí komunitního webu. Tento komunitní web by bylo třeba složitě dohledávat v databázi Enjinu. Těchto komunitních webů navíc může vznikat neomezené množství, čímž se stane nalezení dobře fungujícího komunitního webu o dané hře pro každého uživatele ještě složitější. V případě použití druhé vrstvy portálu tato situace nenastává, protože každá hra bude zastoupena pouze jednou a její správci budou pečlivě vybíráni, aby bylo dosaženo co nejvyšší kvality obsahu.

Lokalizace je další nespornou výhodou navrhovaného řešení. Prakticky všechny existující řešení jsou v angličtině, případně jiném jazyce, a lokalizaci nepodporují. Díky implementaci lokalizace obsahu vytvářeného komunitou se tak stane tento obsah dostupný mnohem většímu okruhu uživatelů.

Výhodou navrhovaného herního portálu je dostupnost všech funkcí pro správce guild webů zcela zdarma. Správce guild webu je ve většině případů rovněž správce guildy (tzv. guild master) i v samotné hře, kde má na starost mnohdy až stovky hráčů. Tento uživatel tedy přivede do webové aplikace desítky až stovky nových uživatelů a dle mého názoru není vhodné ho zatížit nějakými měsíčními poplatky, které by ho mohly odradit od používání aplikace. Vhodnějším řešením by bylo zpoplatnit některé funkce pro jednotlivé uživatele, např. skrytí reklam, dostupnost mobilních aplikací, atd... Tento projekt není komerční a proto se tato práce ekonomickou stránkou projektu dále nezabývá.

Vytvoření responzivních šablon pro jednotlivé části portálu zpřístupní jeho obsah širší škále uživatelů. V případě projektů Enjin a iClan Websites nejsou jednotlivé guild weby responzivní a pro přístup na tyto weby je potřeba použít k tomuto účelu speciálně vytvořené mobilní aplikace.

2 POUŽITÉ TECHNOLOGIE

K vytvoření webového portálu o tak velkém rozsahu, jako je tento, jsem se rozhodl použít frameworky se kterými mám již nějaké zkušenosti a znám jejich funkcionality a nedostatky. Použití jiných nástrojů by mohlo vést k jejich nevhodnému použití nebo vytvoření chybného návrhu, což by mohlo způsobit komplikace při případných úpravách či rozšiřování portálu v budoucnu.

Celý webový portál je postaven na PHP frameworku Nette [18]. Jedná se o český framework, který se u nás stal poměrně populárním a komunita okolo tohoto frameworku je velice aktivní. Nette framework používá softwarovou architekturu Model-View-Presenter [16], která aplikaci rozděluje na tři oddělené části, které velice zpřehledňují její kód a podporují znovupoužitelnost kódu. Na internetu je k dispozici mnoho doplňků k tomuto frameworku, z nichž některé jsou na tomto projektu použity. Jedná se například o vizuální komponentu ublaboo/datagrid [4]. Datagrid vykreslovaná data umí řadit, stránkovat, editovat, filtrovat, přidávat nové záznamy a případně nad nimi provádět volání různých akcí, jako je smazání záznamu či přesměrování na jeho detail.

Pro práci s databází Nette obsahuje databázovou knihovnu Nette\Database, případně lze použít knihovnu dibi, vytvořenou tvůrci Nette frameworku. Tyto knihovny však nebyly použity a byly nahrazeny ORM frameworkem Doctrine [7], který pracuje na jiném principu než tyto dvě knihovny. ORM frameworky používají návrhový vzor mapper, viz 3.2.1. Díky tomu, že je databáze namapována do objektů, je její obsluha velice zjednodušena. Tvůrce aplikace ve většině případů pak pracuje pouze s objekty a SQL dotazy píše spíše jen v případech, kdy je potřeba generované dotazy optimalizovat.

O front-endovou část se stará framework Bootstrap [3], který zajišťuje responzivitu webu a obsahuje mnoho předpřipravených komponent. Pro snadné přizpůsobení bootstrapu jsou k dispozici jeho zdrojové kódy ve formě preprocessorů Less a Sass (použit byl preprocessor Less). Tyto preprocessory rozšiřují kaskádové styly o dynamické prvky jako jsou proměnné, funkce nebo základní aritmetické operace. Tento kód je potřeba před použitím ještě zkompilovat do CSS. K tomuto účelu byl použit task runner Grunt [9], který kompilování less na css provádí automaticky po každé změně souboru.

3 NÁVRHOVÉ VZORY

Návrhový vzor lze popsat jako způsob řešení daného problému při vytváření softwaru. Jedná se o jakýsi návod jak postupovat. Tento postup je obecný a návrhové vzory jsou tedy nezávislé na použitém programovacím jazyku. Existuje velké množství návrhových vzorů, které se hodí pro různé situace. Tvůrce není nijak omezen v počtu použitých vzorů. Při psaní software by však nikdy nemělo dojít k situaci, kdy bude aplikace za každou cenu používat vzor XY.

Důvodů k použití návrhových vzorů je hned několik. Jejich správné použití zvyšuje efektivitu kódu, jeho přehlednost a znovupoužitelnost. Každý programátor by měl návrhové vzory znát, takže při práci s cizím kódem se v něm lépe orientuje. Návrhové vzory dále řeší mnoho obecně známých problémů, které se často v aplikacích vyskytují a bývá pravidlem, že snižují pravděpodobnost výskytu chyb, než je tomu při použití vlastního řešení. Implementace rozšíření a dalších pozdějších úprav kódu, je rovněž o mnoho snazší, protože návrhové vzory s tímto počítají a jsou již na tuto situaci navrženy. Návrhové vzory se dělí na 3 základní kategorie: creational patterns, structural patterns a behavioral patterns [5, 16].

3.1 Creational patterns

Tyto vzory jsou určeny k vytváření objektů a to převážně za běhu programu. Jejich úkolem je zjednodušit či zautomatizovat vytváření požadovaných objektů, zajistit jejich správný počet a instancovat správnou třídu.

3.1.1 Factory

Továrních vzorů existuje několik druhů, např. Simple factory, Static factory nebo Abstract factory. Tyto vzory obecně mají stejný účel, tj. vytvářet objekty dle potřeb uživatele. Řídí způsob získávání závislostí objektu a jeho vytvoření. Stejně jako většina návrhových vzorů i tento vzor zlepšuje znovupoužitelnost a usnadňuje implementaci nových funkcionalit.

Výpis kódu 3.1 znázorňuje příklad implementace tovární třídy, která má za úkol vytvořit objekt uživatele a naplnit jej daty. V tomto příkladu jsou data získávána ze session. Pokud by bylo třeba z nějakého důvodu uživatelská data získávat například z databáze, tak stačí upravit pouze kód tovární třídy a zbytek aplikace zůstává nezměněn.

Výpis 3.1: Příklad implementace továrny.

```
class User
{
    private $data;

    public function __construct($data)
    {
        $this->data = $data;
    }
}

class UserFactory
{
    private $sessionManager;

    public function __construct($sessionManager)
    {
        $this->sessionManager = $sessionManager;
    }

    public function createUser()
    {
        $data = $this->getData();
        return new User($data);
    }

    public function getData()
    {
        return $this->sessionManager->getUserData();
    }
}
```

3.1.2 Lazy initialization

Opožděná inicializace je návrhový vzor, který se používá převážně při optimalizaci výkonu aplikace. Při použití tohoto vzoru je odloženo vytváření objektů, získávání dat z databáze nebo provádění náročných výpočtů až do okamžiku, kdy je to opravdu potřeba. Nedochází tak k provádění zbytečných operací, které nejsou

potřeba a zrychlí se tak běh aplikace.

Nevýhodou je riziko vytváření prodlev za běhu aplikace při provádění těchto operací, které by se jinak provedly při startu aplikace. V případě jazyka PHP však tento problém nemůže nastat vzhledem k povaze způsobu zpracovávání požadavků v tomto jazyce.

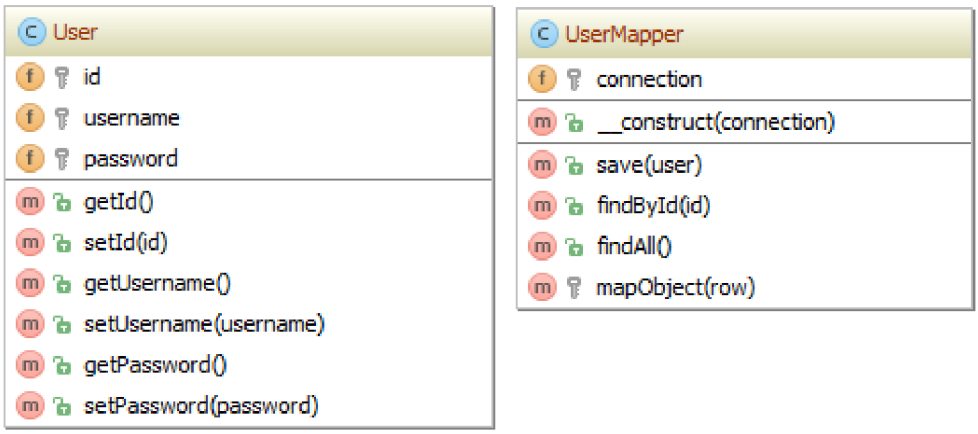
3.2 Structural patterns

Strukturální vzory určují způsob uspořádání jednotlivých tříd a jejich vzájemný vztah v aplikaci. Tím zpřehledňují kód a zlepšují jeho znovupoužitelnost.

3.2.1 Data mapper

Data mapper zajišťuje obousměrný přenos dat mezi datovým uložištěm (obvykle databází) a programem. Při čtení dat z uložiště mapper načte požadovaná data a naplní jimi objekt (entitu) reprezentující požadovaná data v uložišti. Při zápisu mapper analyzuje změny v entitě a tyto změny přenese zpět do uložiště.

Příklad mapování entity uživatele je na obr. 3.1. Objektu *UserMapper* je v konstruktoru předáno připojení k databázi. Při volání metod *findById* nebo *findAll* je pro každý vrácený řádek záznamu z databáze volána metoda *mapObject*, která získanými daty naplní objekt *User* a vrátí jej. Metoda *findById* tedy vrací objekt typu *User* a metoda *findAll* vrací pole těchto objektů, které reprezentují všechny uživatele uložené v databázi. Voláním metody *save* se naopak data z objektu *User* uloží do databáze pomocí SQL dotazů *INSERT* nebo *UPDATE*.



Obr. 3.1: Příklad mapování entity uživatele.

3.2.2 Dependency injection

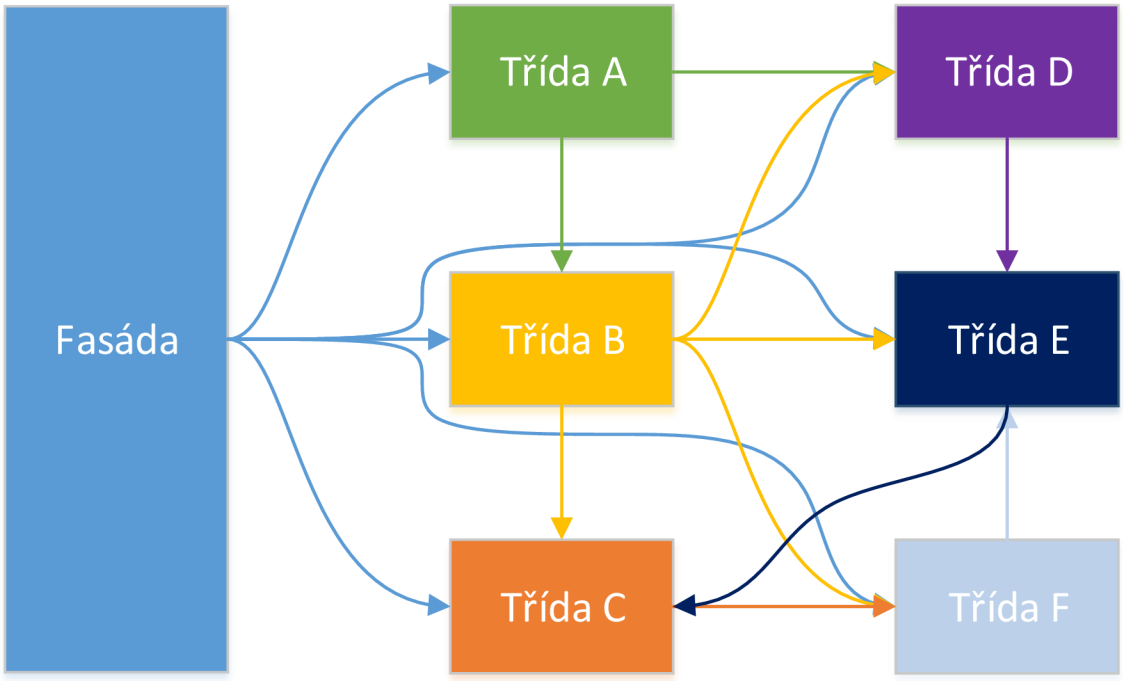
Vzor dependency injection (dále jen DI) řeší předávání závislostí mezi objekty. Jednotlivým objektům jsou jejich závislosti předány zvenčí (například v konstruktoru nebo setteru), namísto toho, aby si objekty tyto závislosti řešily uvnitř samy. Příklad použití DI je znázorněn na výpisu kódu 3.2. Výhodou předání závislostí pomocí DI v tomto příkladu je vytvoření pouze jedné instance třídy *Connection*. Toto řešení navíc zlepšuje čitelnost kódu, kde je jasně vidět, že třída Article pracuje s databází. Vše je tedy transparentní a nic se neprovádí skrytě uvnitř. DI se často používá v kombinaci s továrnami, viz 3.1.1.

Výpis 3.2: Příklad použití dependency injection.

```
class Article
{
    private $connection;

    // Vyreseni zavislosti uvnitr objektu
    public function __construct()
    {
        $this->connection = new Connection();
    }

    // Predani zavislosti zvenci pomoci DI
    public function __construct($connection)
    {
        $this->connection = $connection;
    }

    public function publish()
    {
        // Zverejni clanek
    }
}

$connection = new Connection();
$article1 = new Article($connection);
$article1->publish();
$article2 = new Article($connection);
$article2->publish();
```

3.2.3 Facade

Fasády sdružují několik spolu souvisejících tříd do jedné a vytvoří tím subsystém, se kterým se komunikuje pomocí fasády. Mimo tento subsystém se pracuje pouze s metodami fasády a podstatně se tím zjednoduší udržování závislostí, kdy se předává pouze závislost na fasádu místo všech závislostí na třídy, které fasády sdružuje. Použití fasády je vhodné v případě práce s komplexnějším systémem, který je tvořen větším množstvím tříd, které jsou na sobě závislé, viz obr. 3.2.



Obr. 3.2: Příklad použití fasády.

3.2.4 Crate

Přepravka, někdy označována také jako Messenger, slouží k přenášení hodnot. Použití tohoto vzoru je vhodné v případě, že je potřeba, aby metoda vracela více hodnot současně. Docílí se tím lepší přehlednosti kódu než při použití pole jako návratové hodnoty a zároveň je zajištěna typová kontrola návratových hodnot. Nemůže tedy nastat situace, kdy by byly například zaměněny rozměry 2D objektu za souřadnice bodu, i když se v obou případech jedná o 2 číselné hodnoty. Příklad použití přepravky je na výpisu kódu 3.3.

Výpis 3.3: Příklad použití přepravky.

```
class Gps
{
    private $data;

    public function getCoords()
    {
        return new Coords($data['latit'], $data['longtit']);
    }
}

class Coords
{
    private $latitude;
    private $longitude;

    public function __construct($latitude, $longitude)
    {
        $this->latitude = $latitude;
        $this->longitude = $longitude;
    }

    public function getLatitude()
    {
        return $this->latitude;
    }

    public function getLongitude()
    {
        return $this->longitude;
    }
}
```

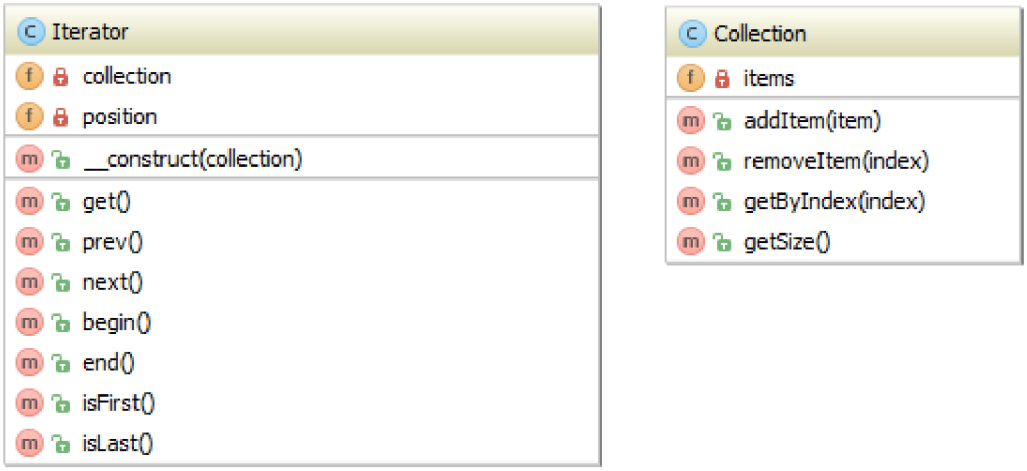
3.3 Behavioral patterns

Tyto návrhové vzory určují chování systému. Definují způsob interakce mezi jednotlivými objekty a jakým způsobem bude mezi ně rozděleno zpracovávání úlohy.

3.3.1 Iterator

Iterátor je návrhový vzor, který definuje objekt, který je schopný procházet kolekce objektů. Tyto kolekce je možné procházet nezávisle na jejich implementaci.

Příklad implementace iterátoru je zobrazen na obr. 3.3. Iterátor obsahuje ukazatel na aktuální položku v kolekci, která je mu předána v konstruktoru. Pomocí metod *prev()* a *next()* je měněna hodnota ukazatele aktuální položky. Metoda *get()* poté vrátí položku z kolekce podle hodnoty ukazatele. Metody *begin()* a *end()* posunují ukazatel na začátek/konec kolekce. Je vhodné rovněž implementovat metody *isFirst()* a *isLast()*, které zjišťují, zda je aktuální položka v kolekci na prvním/posledním místě.



Obr. 3.3: Příklad implementace iterátoru.

3.3.2 Null Object

Tento vzor definuje výchozí stav objektu a nahrazuje tak prázdný ukazatel *NULL*. Jedním ze způsobů implementace je společné rozhraní pro nulový a nenulový objekt. Místo hodnoty *NULL* je pak používán ukazatel na nulový objekt, který obsahuje vlastní implementace metod společného rozhraní. Tento přístup přináší několik výhod:

- Zjednodušuje kód - při volání metod není nutné kontrolovat, zda se nejedná o prázdný ukazatel *NULL*.
- Snižuje pravděpodobnost pádu aplikace na null pointer exceptions.
- Snižuje počet případů, které je nutné testovat.

Například přihlášený uživatel na webu bude zastupován třídou *User*. Po odhlášení uživatele bude nepřihlášený uživatel zastupován třídou *NullUser*, která má vlastní implementaci metody *getName()*, viz výpis kódu 3.4. Tato metoda bude implicitně

vracet hodnotu Guest. Toto řešení zjednodušuje vypisování jména uživatele, kdy pro přihlášeného uživatele bude vypsáno jeho jméno. V případě nepřihlášeného uživatele se voláním stejné metody vypíše Guest.

Výpis 3.4: Příklad použití Null Object.

```
interface IUser
{
    public function getName();
}

// pouziti Null Object
echo $user->getName();

// bez pouziti Null Object
if (!is_null($user)) {
    echo $user->getName();
} else {
    echo 'Guest';
}
```

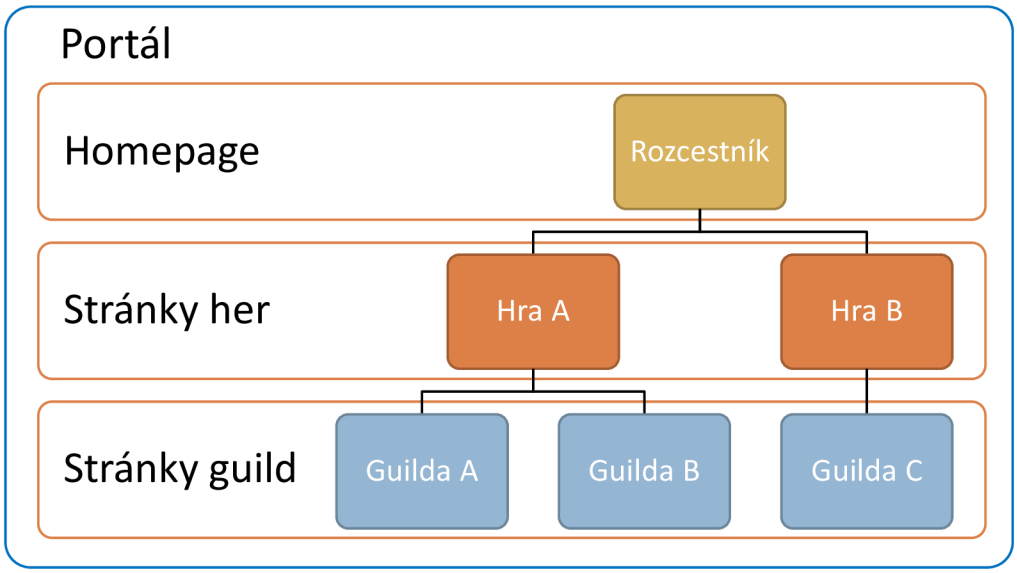
4 NÁVRH SYSTÉMU

Webový portál je rozvržen do tří částí: domovské stránky, stránek jednotlivých her a stránek guild. Viz obr. 4.1.

Domovská stránka portálu slouží jako informační centrum a rozcestník na dílčí stránky zabývající se jednotlivými hrami. Nepřihlášený uživatel zde nalezne přehled aktualit, nadcházejících událostí a seznam podporovaných her. Po přihlášení se uživateli navíc zobrazí jeho konkrétní události, přehled jeho aktivity a seznam jeho oblíbených položek, které budou sloužit jako záložky pro rychlý přesun napříč portálem. Tyto položky bude moci každý uživatel skrývat či přesouvat dle svých potřeb.

Druhou část portálu tvoří stránky jednotlivých her. Zde se zobrazují aktuality, články, galerie a návody týkající se konkrétních her. Rovněž je zde k nalezení seznam stránek guild a klanů, které mohou registrovaní uživatelé zakládat.

Stránky guild tvoří poslední část portálu. Tyto stránky si mohou jejich správci přizpůsobit dle svých potřeb. Mimo aktualit, článků a přístupu do databáze návodů mají možnost využít další funkce, jako je diskusní fórum nebo kalendář událostí, kde se mohou členové jednotlivých guild přihlašovat například na různé raidy, arény, atd...



Obr. 4.1: Struktura jednotlivých částí portálu.

4.1 Lokalizace

Anglický jazyk se v herním průmyslu stal prakticky standardem. Přestože v dnešní době již většina hráčů má alespoň základní znalosti angličtiny, tak hledání různých informací a čtení návodů (které mnohdy mají i několik stran) pro mnoho z nich může být velice obtížné. Pro zvýšení komfortu uživatelů a zlepšení dostupnosti informací je tedy lokalizace důležitou součástí webového portálu.

Většina obsahu, která bude na portálu k nalezení, bude tvořena komunitou. Tomu je potřeba přizpůsobit i celý systém lokalizace a dopředu počítat s tím, že některé překlady mohou být nekvalitní či nedodělané.

Jednotné řešení lokalizace uživatelského rozhraní a obsahu by bylo velice neefektivní a uživatelsky nepohodlné. Z tohoto důvodu je lokalizace ve všech částech portálu řešena dvěma způsoby.

4.1.1 Lokalizace uživatelského rozhraní

Lokalizaci uživatelského rozhraní mají na starost uživatelé, kteří dostali od administrátorů jednotlivých stránek oprávnění k překladu. Tímto je zajištěno, že se o překlad jednotlivých prvků budou starat předem vybraní uživatelé, kteří by měli zajistit jistou úroveň překladu. Při překládání ovládacích prvků a navigace není tedy důvod k vytváření více překladů pro jeden jazyk. Tato funkce bude zpřístupněna ve všech třech částech portálu, avšak v případě stránek guild nebude pravděpodobně moc využívána, protože ve všech guildách bývá určen jeden jazyk, kterým mezi sebou hráči komunikují a k lokalizaci jejich stránek tedy není důvod.

4.1.2 Lokalizace dokumentů

Lokalizace dokumentů je naopak určena převážně pro stránky her a guild. Jedná se o překlady všech textových dokumentů jako jsou návody, novinky, články, atd. Například guilda hrající na francouzkých serverech bude mít s největší pravděpodobností jako svůj hlavní jazyk francouzštinu. Správci guildy budou moci na své stránky umisťovat články a návody, aby je její členové nemuseli hledat. Většina těchto článků bude napsána v angličtině. Lokalizace dokumentů ovšem umožní tento článek vložit ve francouzštině a podstatně tím tak zvýší komfort uživatelů.

V tomto případě bude moci překládat dokumenty kdokoli a je tedy nutné řešit možnost, kdy uživatel svůj překlad nedokončí nebo je kvalita jeho verze překladu velmi nízká. Z těchto důvodů není možné pro každý dokument mít pouze jeden překlad do konkrétního jazyka. Velice často by docházelo k situacím, kdy by dokumenty neměly žádný překlad, protože nebyl dokončen a nebylo by možné vytvořit nový bez zásahu administrátora.

Řešením tohoto problému je implementovat možnost vytvořit několik překladů konkrétního dokumentu a k těmto překladům vytvořit systém hodnocení. Jako výchozí překlad se poté zobrazí ten s nejlepším hodnocením. Tímto způsobem se eliminují nekvalitní překlady a nedokončené překlady nebudou blokovat další překlady dokumentu.

4.2 Komponenty

Pro vytváření obsahu v jednotlivých částech portálu je k dispozici několik různých komponent pro pokrytí potřeb uživatelů. Každá část portálu má přiřazenou sadu komponent, která je poté dostupná administrátorům jednotlivých stránek. Některé komponenty mají více variant, které jsou odvozeny ze společného předka a rozšiřují ho o další funkcionality.

4.2.1 Dokumenty

Pomocí dokumentů je tvořen veškerý textový obsah portálu a jsou tak nejdůležitější komponentou ze všech. Tato komponenta je předkem specifičtějších komponent, které tvoří již konkrétní obsah. Jedná se o články, návody a novinky. Všechny tyto komponenty mají mnoho společných prvků: název, jazyk, samotný obsah, seznam zdrojů, odkud autor čerpal, možnost připojení diskuse, atd.

Články

Články tvoří většinu obsahu druhé úrovně portálu, tj. stránek her. Pomocí článků mohou jednotliví redaktoři vytvořit recenze, popisovat změny plánované vývojáři, či psát vlastní příběhy zasazené do světa těchto her. Správci stránek guild mohou články rovněž využít k vytváření informativních sekcí, například sekce s pravidly nebo neveřejné sekce s informacemi pro připojení na VoIP službu.

Návody

Návody jsou tvořeny prakticky totožným způsobem jako články. Navíc obsahují informaci o verzi hry, pro kterou je daný návod napsán. Všechny návody jsou řazeny do různých kategorií, jako jsou návody týkající se postav, profesí nebo instancí. Publikované návody, tj dokončené návody ve stavu *Published*, jsou vždy veřejné a nelze omezit jejich viditelnost. Účelem portálu je poskytovat informace herní komunitě a skrývání návodů je tedy nežádoucí.

Novinky

Novinky jsou kratší zprávy informativního charakteru, které jsou chronologicky řazeny. Každá novinka musí být přiřazena k nějakému kontejneru, který k sobě všechny novinky váže. Narozdíl od článku nebo návodu nemůže být novinka přímo obsažena v navigaci.

4.2.2 Diskuse

U většiny komponent je správcům stránek umožněno založit diskusi k danému obsahu. Do každé vytvořené diskuse je umožněno přispívat pouze registrovaným uživatelům. Každý příspěvek pak obsahuje mimo samotného textu i jeho autora a datum vytvoření. Na jednotlivé příspěvky je možné reagovat. Diskuse je tedy řazena do stromové struktury.

Každou diskusi je rovněž možno moderovat uživateli s dostatečným oprávněním. Tito moderátoři mají možnost mazat nevhodné příspěvky a případně trestat autory těchto příspěvků zákazem psaní do diskusí na dané stránce, případně na celém portálu.

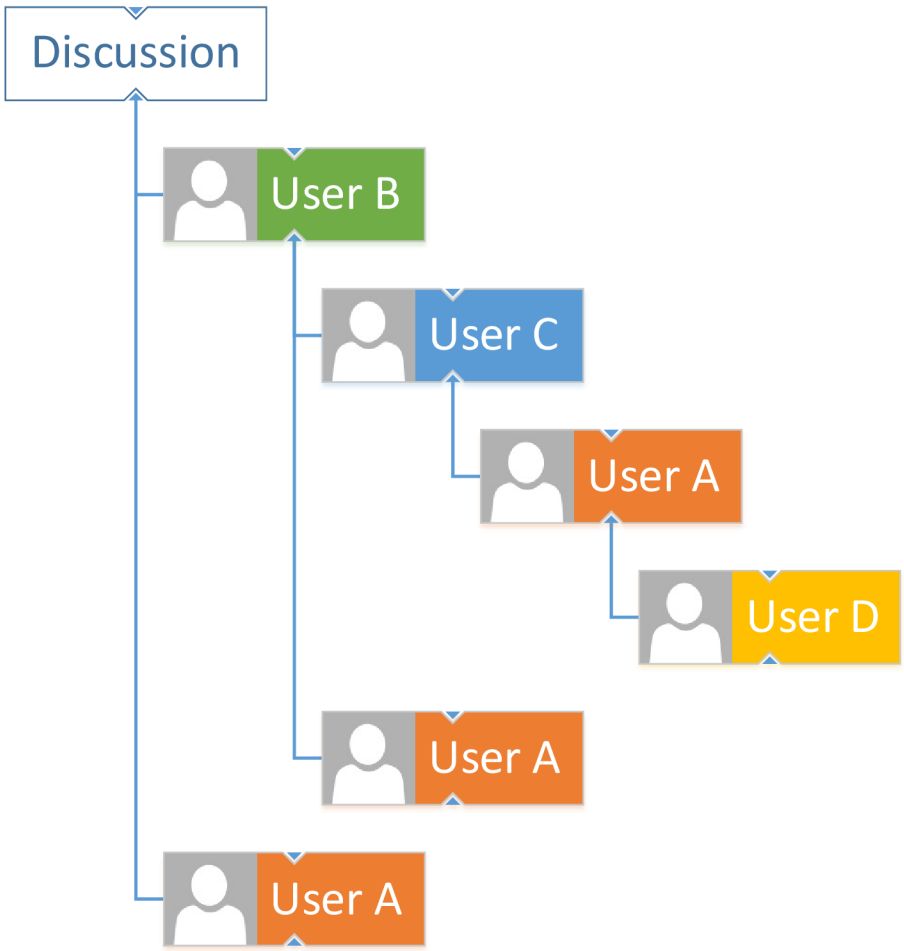
Pro omezení takzvaných „spammerů“ lze v rámci dané herní stránky nastavit maximální povolený počet po sobě jdoucích příspěvků. Toto nastavení hlídá počet příspěvků uživatele v rámci diskuse a dané větve odpovědí. Po překročení nastaveného limitu není uživateli umožněno vložit další příspěvek do diskuse dokud se v diskusi neobjeví příspěvek od jiného uživatele.

Na obr. 4.2 je znázorněna struktura diskuse, ve které je uživatel A velice aktivní. Při zapnuté ochraně diskusí a nastavení maximálního počtu příspěvků na jeden, má v této diskusi uživatel A možnost reagovat pouze na uživatele D. V ostatních případech mu systém nedovolí vložit nový příspěvek.

4.2.3 Diskusní fórum

Diskusní fórum tvoří kontejner, který v sobě uchovává jednotlivá témata. Jednotlivá fóra lze do sebe zanořovat a vytvořit tím stromovou strukturu. Ke každému fóru lze nastavit přístupová práva a vytvořit tak privátní sekce přístupné pouze registrovaným uživatelům nebo administrátorská fóra, do kterých mají přístup pouze správci jednotlivých stránek.

Každé téma je reprezentováno diskusí, která je totožná s komponentou diskusí používaných v jiných částech portálu. Liší se pouze ve způsobu vykreslování a možností vkládání formátovaného textu pomocí editoru Tiny MCE.



Obr. 4.2: Struktura diskuse.

4.2.4 Galerie

Galerie mají podobnou strukturu jako diskusní fórum. Opět se jedná o kontejner, který v sobě uchovává jednotlivé fotografie. Ke každé galerii lze řídit přístupová práva a rovněž je lze do sebe zanořovat. V nastavení galerie lze zvolit výchozí řazení fotografií od nejnovější/nejstarší nebo povolit ruční řazení fotografií. V případě, že není nastavena výchozí fotografie, která reprezentuje danou galerii, tak je jako výchozí fotografie použita první zobrazovaná fotografie v galerii.

4.2.5 Kalendář, Raid planner

Kalendář je komponenta do které lze zadávat různé události jako jsou porady nebo tréningy. Uživatelé portálu se zde mohou na tyto události libovolně přihlašovat, případně je jen sledovat. Viditelnost událostí lze omezit pomocí systému přístupových práv. Tato komponenta je primárně určena pro nejnižší vrstvu portálu, tj. stránky

guild. Každá stránka může mít vytvořen pouze jeden kalendář, kterému lze nastavit jeho umístění a způsob vykreslování.

Způsoby vykreslování jsou celkem tři. Prvním je klasický měsíční přehled. Tato varianta nemůže být vzhledem ke své velikosti umístěna v postranním panelu. Druhou variantou je zobrazování událostí v týdenních intervalech. Poslední variantou je zobrazení určitého počtu nadcházejících událostí.

Raid planner

Tato komponenta je rozšířením komponenty kalendáře o nové funkcionality. Je určena pro MMORPG hry (např. World of Warcraft), kdy každý hráč vlastní několik herních postav.

Při vytváření herní události má autor možnost nastavit instanci, které se to týká. Systém pak automaticky doplní potřebný počet hráčů a odhadovaný čas trvání události. Rovněž umožňuje nastavit požadované rozložení povolání postav jednotlivých postav a hlídat jejich požadovanou úroveň. Všechna tato nastavení bude možné před vytvořením události ještě změnit, či uložit jako šablonu pro příští použití. Každý uživatel se poté na danou událost přihlašuje pomocí své postavy.

Některé hry podporují zadávání textových příkazů do herního chatu pro zjednodušení některých úkonů. Těchto textových příkazů je využito k vygenerování makra pro vytvoření skupiny ve hře. Toto makro stačí pouze zkopírovat a zadat do herního chatu.

4.2.6 Ankety

Hlasování v anketách je umožněno pouze registrovaným uživatelům. Tím je zajištěna objektivnost anket, které nepůjdou tak snadno zmanipulovat například pomocí proxy serverů. Viditelnost ankety a hlasování je opět řízeno pomocí systému přístupových práv. Každé anketě lze nastavit doba od které je možné hlasovat, případně datum a čas ukončení hlasování.

Každé anketě je možné před jejím zveřejněním nastavit tři režimy zpracovávání hlasů. Po zveřejnění ankety již není možné tento režim změnit. Prvním je veřejný režim, kdy je ihned vidět, kteří uživatelé hlasovali pro danou možnost. Druhou variantou je tajné hlasování, kdy jednotlivé uživatelské hlasy vidí jen uživatelé s definovanými rolemi. Poslední možností je anonymní hlasování, kdy nebude možné zobrazit kdo a jak hlasoval.

4.3 DKP

DKP neboli dragon kill points je velice rozšířený systém přerozdělování získaných předmětů v MMORPG hrách. V drtivé většině her se předměty automaticky přerozdělují náhodně mezi členy skupiny. Tento způsob přerozdělování není úplně spravedlivý k hráčům, co se guildovních akcí účastní pravidelně, ale nemají moc štěstí při generování čísla, podle kterého se předměty přerozdělují. Hráči si proto vymysleli systém DKP.

Tento systém spočívá v tom, že každý člen skupiny dostane za zabití monstra určitý počet bodů, za které pak nakupuje věci z kořisti. Hráč, který se akcí účastní pravidelně by pak měl mít dostatek bodů na to, aby si koupil předmět o který má zájem. Méně aktivní hráči pak musí počkat než na ně přijde řada, protože mají málo bodů. Nakupování je zpravidla realizováno dvěma způsoby. Prvním z nich je dražba, kdy hráči daný předmět mezi sebou draží jako na aukci. Druhá metoda je fixní částka, kdy je za daný předmět předem definována jeho cena a hráč s nejvyšším počtem bodů předmět získá a cena daného předmětu je mu odečtena.

Systém DKP by tedy měl být spravedlivější než náhodné přerozdělování a ve většině případů jeho použití guildám prospívá. Zkušenější hráči, kteří hrají pravidelně, tak získají rychleji lepší vybavení a mohou tak lépe pomáhat méně zkušeným hráčům, kteří by toto vybavení nedokázali tak efektivně využít.

Jelikož systém DKP není součástí her, je nutné jej řešit pomocí externích řešení. Tento systém je velice důležitý pro fungování guild a proto je implementován mezi funkce portálu. Pro jeho snadnější ovládání je DKP systém napojen na raid planner (viz 4.2.5). Pomocí raid planneru je možné udělit DKP za zabití monstra celé skupině jedním kliknutím automaticky. Pro tuto funkci je třeba v administraci přiřadit jednotlivým monstrům počet DKP, které lze za jejich zabití získat. Při probíhajícím raidu lze založit DKP „místnost“, kam jsou automaticky pozvání všichni přihlášení členové. Hráč, který má na starosti DKP vybere monstrum, které bylo zabito a tím přidělí všem členům party body (nebo je přidělí ručně). Poté vyhlásí dražbu předmětu. Tato dražba se zobrazí uživatelům v místnosti a po předem definovaný čas má každý z nich možnost se dražby zúčastnit. Po ukončení dražby je vítězi stržena nabídnutá částka z jeho konta. Obsluha DKP je navržena pro ovládání z mobilního zařízení a proto je maximálně optimalizována pro zobrazování na těchto zařízeních. Vytvoření místnosti a pozvání jednotlivých uživatelů je umožněno i bez napojení na raid planner.

4.4 Front-end

Jak již bylo dříve uvedeno, celý portál je rozdělen na tři části. Každá část má generický obsah, který je tvořen jednotlivými komponentami. Layout stránky je tvořen nastavitelným „gridem“, ve kterém je možné vytvářet buňky o různé šířce. Každá buňka může obsahovat pouze jednu komponentu. U každé komponenty je definována minimální požadovaná šířka buňky pro zajištění jejího správného vykreslení.

4.4.1 Domovská stránka

Domovská stránka portálu má dva režimy. První z nich je zobrazen nepřihlášenému uživateli. Zde je uživateli stručně představen portál a je vyzván k registraci/přihlášení. Dále mu jsou zobrazeny různé statistiky, novinky a stránky her, které by ho mohly zajímat.

Přihlášenému uživateli jsou zobrazeny informace o jeho účtu, novinky z her, které hraje, přehled nadcházejících událostí a přehled jeho oblíbených položek. Tuto stránku si může libovolně upravovat přeskládáním zobrazovaných prvků, případně některé z nich úplně skrýt. Návrh domovské obrazovky přihlášeného uživatele je zobrazen na obr. B.1.

4.4.2 Stránky her

Stránky her tvoří druhou vrstvu herního portálu. Každá stránka se věnuje jedné konkrétní hře. Tyto stránky jsou již tvořeny komunitou a záleží pouze na správcích dané stránky, jakým způsobem využijí funkce portálu, které komponenty použijí a jakým způsobem poskládají layout stránky. K dispozici budou mít několik přednastavených šablon, které budou definovat doporučené rozložení stránky a obsah, který by měla každá herní stránka obsahovat. Povinností každé herní stránky bude zobrazení seznamu guild a návodů. Jedna z možných podob herní stránky je zobrazena na obr. B.2.

4.4.3 Stránky guild

Stránky guild jsou stejně jako herní stránky tvořeny komunitou a jejich vzhled záleží pouze na potřebách dané guildy. Některé mohou mít pouze domovskou stránku s několika komponentami pro zobrazení těch nejnutnějších věcí, jiné guildy mohou vytvořit komplexní web s mnoha funkcemi a velkým množstvím obsahu. Pomocí nástroje pro vkládání návodů z databáze portálu je možné zobrazovat velké množství

potřebných informacích pro hráče přímo na stránkách guildy a ušetřit jim tak starosti s vyhledávání těchto dokumentů v databázi portálu. Koncept guildovní stránky je zobrazen na obr. B.3.

5 APLIKAČNÍ ROZHRANÍ

Pro zvýšení komfortu uživatelů jsou v portálu implementována různá aplikační rozhraní. Tyto rozhraní umožňují přístup k datům jednotlivých uživatelů z aplikacích třetích stran. Většina těchto dat lze získat veřejně, tj. bez souhlasu uživatele. Pro přístup k citlivějším datům je však vyžadováno přihlášení uživatele do aplikace poskytující data a povolení přístupu třetí strany k těmto datům. K tomuto účelu je využíván protokol OAuth.

5.1 Protokol OAuth 2

Protokol OAuth 2 [20, 15] umožňuje přihlášení uživatele v aplikaci třetí strany a řízení přístupu této aplikace k datům uživatele. Přihlašování je navrženo tak, aby uživatel nemusel poskytovat své přihlašovací údaje třetí straně a snížilo se tak riziko úniku těchto údajů.

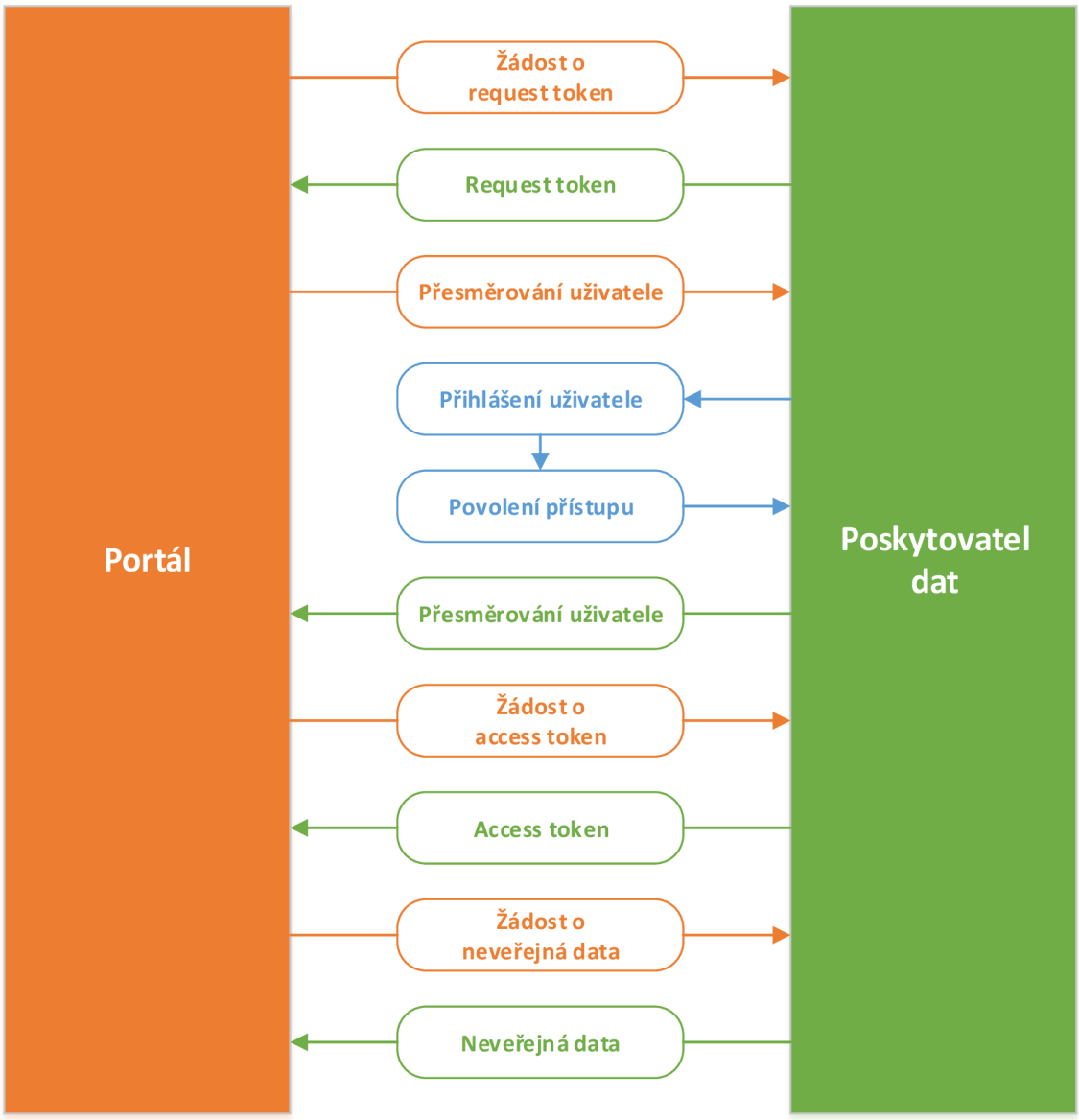
Na obr. 5.1 je znázorněn průběh autentifikace a autorizace získávání dat pomocí protokolu OAuth 2. Aplikace třetí strany (v tomto případě portál) zažádá poskytovatele dat (např. Battle.net) o tzv. request token. Pokud je daná aplikace registrována u poskytovatele, tak je jí tento token přidělen. Uživatel je poté přesměrován k poskytovateli, kde se přihlásí, povolí aplikaci přístup k jeho datům a poté je přesměrován zpět do aplikace. Aplikace poté může požádat o přístupový token (access token). Pokud daná aplikace dostala od uživatele práva přistupovat k jeho datům, tak server poskytovatele dat tento token přidělí aplikaci, která poté pomocí přístupového tokenu může žádat o uživatelská data ze serveru poskytovatele.

5.2 Steam Web API

Aplikační rozhraní Steam Web API [22] je jedním z nástrojů pro komunitu poskytovaných společností Valve Corporation. Veškerá data dostupná pomocí Steam Web API jsou veřejně dostupná a nevyžadují autorizaci od uživatelů služby Steam. K získání některých dat je potřeba pouze *API key*, který slouží pro identifikaci aplikací třetích stran. Http request k získání dat má následující podobu:

http://api.steampowered.com/<*interface name*>/<*method name*>/v<version>/?**key**=<api **key**>&format=<format>

Pomocí <interface name>, <method name> a <version> je určeno použité rozhraní, konkrétní metoda a její verze. Pro různé požadavky se mohou parametry



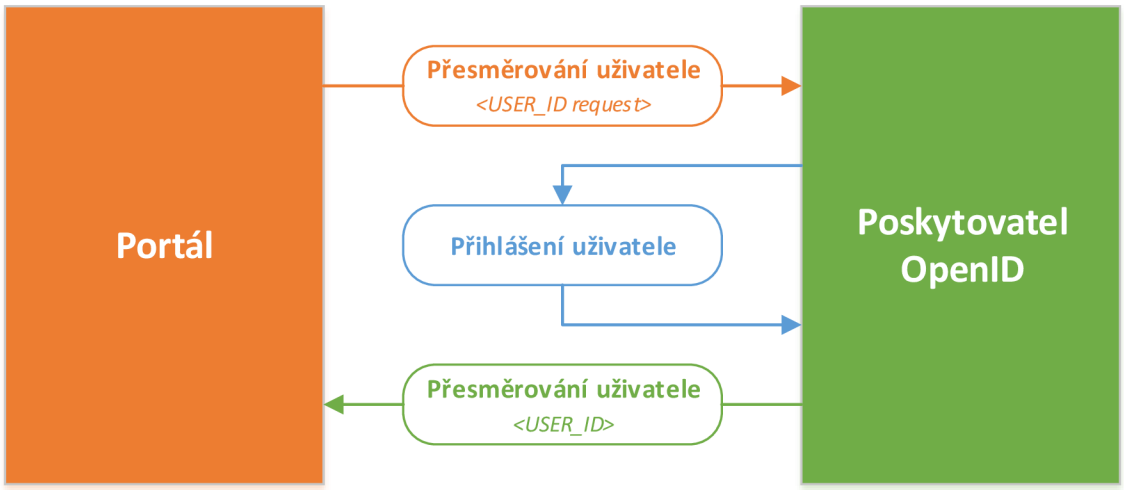
Obr. 5.1: Průběh autentifikace a autorizace pomocí protokolu OAuth 2

v adrese lišit v závislosti na datech, která má request získat. Například některé metody vyžadují mimo klíče i ID uživatele nebo ID hry. Parametr <format> určuje formát vrácených dat. Podporovanými formáty jsou JSON, XML CSV a VDF.

Steam je rovněž poskytovatelem služby OpenID. Podobně jako je tomu o protokolu OAuth 2 i OpenID umožňuje přihlašování uživatelů do aplikací třetích stran bez nutnosti poskytování přihlašovacích údajů. Na rozdíl od protokolu OAuth 2 však OpenID slouží pouze k přihlašování uživatelů, tj. neřeší přístup k privátním datům.

Přihlašování pomocí OpenID je znázorněno na obr. 5.2. Pro přihlášení do aplikace je uživatel přesměrován na službu Steam, která zde slouží jako poskytovatel OpenID. Zde se uživatel přihlásí a je přesměrován zpět na portál. V URL adrese je uvedeno

jeho *SteamID*, které slouží jako unikátní identifikátor uživatele. Pomocí *SteamID* je v aplikaci nalezen příslušný uživatel a jsou získána jeho uživatelská data z vlastní databáze aplikace.



Obr. 5.2: Průběh přihlašování pomocí OpenID

5.3 Battle.net game API

Společnost Blizzard Entertainment Inc. poskytuje možnost získání dat pomocí *Battle.net game API* [2]. Pomocí toho API lze získat data z her World of warcraft, Diablo III a Star Craft II, které byly vydány touto společností. Podobně jako je tomu u Steam Web API, je i zde většina dat veřejně přístupná. Pro získání dat je však striktně vyžadován *API key*, který opět slouží jako identifikátor aplikací třetích stran. Každá aplikace má omezen počet požadavků na 100 volání za sekundu a 36 000 za hodinu. Požadovaná data jsou vrácena ve formátu JSON, viz 5.1.

Pomocí Battle.net API lze rovněž získat informace o uživatelských účtech Battle.net a uživatelských profilech z her Star Craft II a World of Warcraft. Tyto informace již nejsou veřejné a pro přístup k těmto datům musí mít aplikace souhlas uživatelů. Data z těchto profilů jsou na portálu použita k automatickému vyplnění informací v profilu uživatele a vytvoření herních postav. Pro použití funkcionality automatického vyplnění profilu a založení herních postav je uživatel vyzván k přihlášení pomocí protokolu OAuth 2 a povolení přístupu portálu k požadovaným údajům.

Přihlášení pomocí Battle.net účtu není explicitně vyžadováno. Každý uživatel má možnost přihlášení odmítnout a poté vyplnit profil a herní postavy ručně. Díky tomu, že seznam členů guild je veřejně dostupný, tak tímto nebude narušena funkce automatické kontroly členství v guildě.

Výpis 5.1: Získání informací o herní postavě.

```
https://eu.api.battle.net/wow/character/drakthul/sankacoffee
?locale=en_GB&apikey=API_KEY

{
    "lastModified": 1426859288000,
    "name": "Sankacoffee",
    "realm": "Drak'thul",
    "battlegroup": "Reckoning / Abrechnung",
    "class": 5,
    "race": 5,
    "gender": 0,
    "level": 100,
    "achievementPoints": 1865,
    "thumbnail": "drakthul/168/112071080-avatar.jpg",
    "calcClass": "X",
    "faction": 1,
    "totalHonorableKills": 232
}
```

6 REALIZACE

V současném stavu se portál skládá z více než 300 tříd a rozhraní i přes to, že samotná realizace byla zaměřena převážně na model a před nasazením do ostrého provozu bude třeba doplnit ještě velké množství šablon a nástrojů pro správu obsahu. Níže jsou proto uvedeny ty nejdůležitější části portálu s popisem jejich implementace a případnými návrhy na jejich vylepšení.

6.1 Struktura modelu

Model je implementován pomocí fasád, viz kapitola 3.2.3. Jednotlivé třídy fasád implementují veškeré nezbytné metody pro získávání a perzistování dat. Tyto třídy komunikují s ostatními třídami modelu a získaná data předávají presenteru. Schema implementace modelu je znázorněno na obr. 6.1.

Vzhledem k tomu, že počet metod fasád velice rychle narůstal, a i přes použití propracovaného IDE se jejich použití stalo velice nepohodlným, byly tyto metody „zabaleny“ do několika samostatných tříd sloužících pro zápis a čtení. Samotná třída fasády poté obsahuje pouze gettery na tyto třídy. Tato implementace mimo zvýšení přehlednosti při psaní kódu umožňuje použít rozdílné předky těchto tříd a tím zjednodušit jejich vlastní implementaci.

Třídy pro zápis

Třídy určené k zápisu dat mají přístup k *entityManageru* pro možnost řízení transakcí. Díky tomu, že framework Doctrine umožňuje zanořování transakcí, jsou tyto metody vhodným místem pro jejich použití a v drtivé většině případů tak není nutné se transakcemi zabývat uvnitř komponent a presenterů. Při vícenásobném zápisu dat je však možné pomocí DI *entityManager* předat do příslušné komponenty nebo presenteru a celý zápis dat tak volat v jedné „nadřazené“ transakci, kdy se o správné volání transakcí postará framework Doctrine.

Zápis dat je prováděn pomocí tříd služeb, kdy jejich metody striktně přijímají parametry požadovaných typů. Metody tříd určených pro zápis dat jsou uzpůsobeny tak, že přijímají parametry různých typů, které jsou před voláním metod služeb validovány. Například je možné jako parametr předat celou entitu záznamu nebo jen jeho primární klíč. V případě primárního klíče se metoda sama postará o získání entity, která je poté předána dané službě.

Třídy pro čtení

Třídy určené ke čtení záznamů naopak nemají přístup k *entityManageru* a obsahují pouze metody pro čtení dat a jejich validaci. Čtení dat je implementováno takovým způsobem, že samotné vyhledání záznamu se provede v třídách služeb a získaný výsledek je poté zkontrolován, zda byl záznam skutečně nalezen. V případě prázdného výsledku je vyhozena výjimka. Metody těchto tříd tedy vždy vracejí požadovaný záznam. Nenalezení záznamu bývá nestandardní situace a je tedy vhodné tuto situaci ošetřit pomocí výjimek.

Výhodou tohoto rozdělení je možnost implementovat čtení záznamů jiným způsobem než pomocí Doctrine, aniž by zbytek modelu byl narušen. V případě velkého množství uživatelů může v některých případech být lepší použití jiného způsobu získávání dat pro zvýšení výkonu aplikace.

Zhodnocení implementace fasád

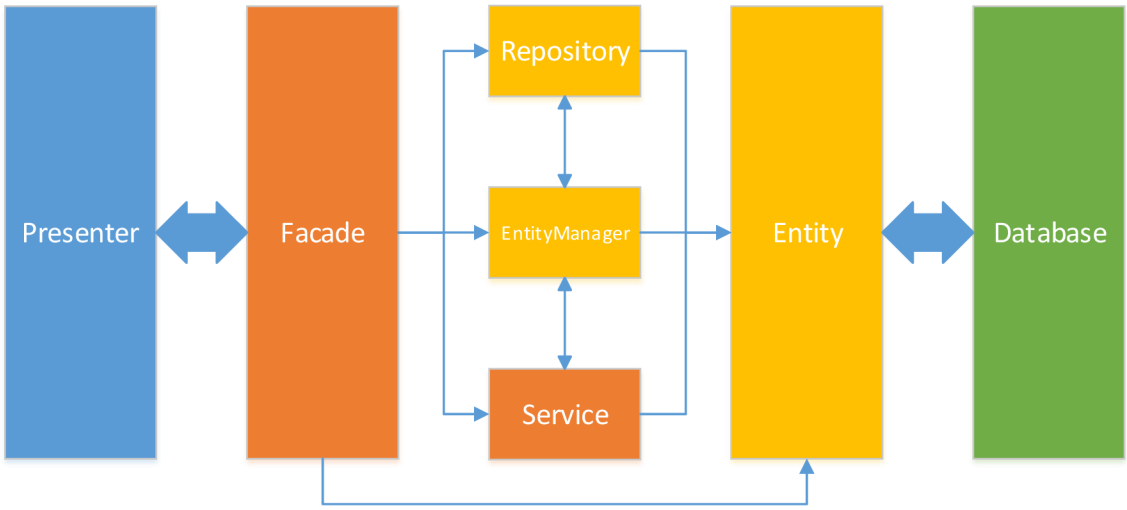
Tento netradiční způsob implementace fasád velice zjednodušil a zrychlil psaní kódu, protože požadovanou metodu bylo mnohem snazší najít. Výhodou této implementace je rovněž snížení rizika vzniku kruhové závislosti, která může vzniknout při použití DI pomocí konstruktorů. Třídy pro zápis pro validaci vstupních dat využívají třídy pro čtení záznamů. Tyto třídy však pro svou funkci třídy pro zápis nepotřebují a kruhová závislost zde tak nemůže vzniknout.

Při testování pomocí unit testů (jednotkových testů) by tato implementace pravděpodobně vše naopak zkomplikovala. Vhodnější a návrhově čistější by tedy bylo rozdělit třídy služeb na třídy pro zápis a čtení a k těmto třídám poté přistupovat přímo z fasád, kde byly použity transakce a probíhala validace vstupních dat, tj. parametry metod.

Pro snížení rizika kruhové závislosti je vhodnější výstupní data, tj. získaná data z databáze validovat uvnitř služeb. Služby by tedy vždy obsahovaly dvojice metod, kdy jedna metoda by získaná data nijak nevalidovala, zatímco druhá metoda by uvnitř volala metodu první, ale její výstup by byl validován a v případě chybných hodnot nebo nenalezení záznamu vyhazovala výjimky.

6.2 Model lokalizace a verzování dokumentů

Články, novinky a návody mají prakticky stejnou strukturu, tj. autor, datum vytvoření/editace, jazyk, název, popis a vlastní text, proto jsou zastoupeny obecnými objekty *TextEntry* a *TextEntryData*, které implementují všechny potřebné vlastnosti těchto komponent. Na obr. A.2 je zobrazen UML diagram celkového řešení verzování a lokalizace dokumentů.



Obr. 6.1: Schema implementace modelu.

Překlady dokumentů jsou tvořeny komunitou, proto je třeba zvolit jiný přístup při realizaci lokalizace než je tomu u systémové lokalizace, kdy je jeden překlad plně dostačující. Není tedy možné mít pouze jeden překlad konkrétního dokumentu a to hned z několika důvodů. Hlavním důvodem je to, že dotyčný překladatel nemusí překlad dokončit a daný dokument by tak zůstal bez překladu. Další situace, která musela být vyřešena je kvalita překladu, kdy daný překlad může obsahovat chyby, nevhodně použité výrazy nebo byl překlad vytvořen pomocí překladače.

Tento problém je řešen pomocí více překladů jednoho dokumentu. Překlady je možné hodnotit a jako výchozí překlad se zobrazuje překlad s nejlepším hodnocením. Zároveň je počet překladů kontrolován a při dosažení určitého počtu kladně hodnocených překladů se uzamkne možnost vytváření dalších. Rovněž je zablokována možnost vytvářet nové překlady pro výchozí jazyk článku, kde není důvod pro jejich vytváření.

Všechny vytvořené dokumenty mají vedenou historii úprav. Autor dokumentu tak může snadno vracet změny, které v daném dokumentu provedl. Při každém ukládání změn je dotázán, zda jsou úpravy které provedl jen opravy překlepů nebo se jedná o rozsáhlejší úpravu textu. V případě, že se jedná jen o opravy překlepů není vytvářena nová verze a změny se uloží do aktuálního záznamu. Pro rozsáhlejší úpravy textu je vytvořen nový záznam, ten může být poté uveřejněn jako nová verze dokumentu. Při uveřejnění nové verze jsou zároveň upozorněni všichni překladatelé dokumentu na nové změny a jsou vyzváni k aktualizaci překladu. Všichni uživatelé webu poté mohou mezi jednotlivými verzemi přepínat.

Tato funkcionalita je důležitá hlavně v případě návodů. Pokud je vydán patch

nebo update hry, je občas nutno návod aktualizovat. Z tohoto důvodu je u návodů veden i záznam o verzi hry, pro kterou byl daný návod napsán. V případě vydání aktualizace hry jsou všechny články, kterých se to týká, označeny jako zastaralé, jejich autoři jsou upozorněni na tuto skutečnost a jsou vyzvání k aktualizaci svých návodů.

6.3 Řízení přístupových práv

Téměř každá webová aplikace musí řešit přístupová práva jednotlivých uživatelů. Zde je třeba přístup řešit velice podrobně. Pro realizaci přístupových práv byla využita upravená implementace autorizátoru z Nette frameworku *Permission ACL*, který je v nette implementován pomocí třídy *Nette\Security\Permission*. Řízení přístupu je realizováno nadefinováním rolí uživatele, jednotlivých zdrojů, ke kterým je třeba řídit přístup a nadefinováním jednotlivých oprávnění. Implementace řízení přístupových práv v Nette frameworku dále umožňuje dědění uživatelských rolí, kdy každá role může dědit z N jiných rolí a dědění zdrojů, kdy každý zdroj může dědit z jednoho jiného zdroje.

Toto řešení bylo rozšířeno o možnost řízení přístupu k jednotlivým zdrojům dle jejich aktuálního stavu, kdy se každý nadefinovaný zdroj zavádí do systému společně s jeho stavem a rozpadne se tak na několik jiných zdrojů, které jsou tvořeny ze zdroje původního a jeho stavu. Zdroj Article je tedy do systému zaveden například pod názvem Article_CREATED nebo Article_PUBLISHED.

Dalším rozšířením původní implementace je automatické načítání rolí uživatele včetně vyřešení dědění rolí, automatické načítání zdrojů a jejich pravidel. Zdroje se do systému zavádí až při volání metody *isAllowed($role, $resource, $privilege)* a pouze s takovým stavem ve kterém se zdroj aktuálně nachází. Tím je docíleno načítání zdrojů, které jsou v danou chvíli opravdu potřeba a minimalizuje se tak počet dotazů do databáze. Vzhledem k tomu, že daný zdroj se vždy nachází pouze v jednom stavu, není nutné načítat pravidla pro všechny stavy, ale stačí načítat pravidla jen pro jeden konkrétní stav. Výjimku tvoří situace, kdy administrátor testuje přístupová práva jako je tomu na obr. 6.2. Tato situace nastává jen zřídka a proto pro tento případ nebylo implementováno speciální řešení načítání všech pravidel.

Díky rozlišování stavů je dědění zdrojů omezeno pouze na dědění od zdrojů stejného typu. Například není možné, aby diskusní fórum dědilo práva od kontejneru stránky, ve které se nachází, protože kontejner má jiné stavy než fórum a pro některé z nich by pravidla nefungovala. Zdroje jsou v modelu zastoupeny šablonami, které mají různé typy, např. ArticleTemplate a ForumTemplate. Každá z nich pou-

žívá stejné stavy jako entity, ke kterým mohou být přiřazeny. Proto každá entita, ke které je možné řídit přístup implementuje rozhraní *IResource*, čímž je zajištěno získávání zdroje (šablony) z entity pro automatické načítání práv, avšak implementace této metody se v jednotlivých entitách liší. V entitách, kde není možné dědit práva od nadřazeného prvku je vždy vrácena přiřazená šablona. V případě entit, které se do sebe mohou zanořovat, např. diskusní fóra nebo galerie, je možnost šablonu nenastavit. V tomto případě se tato metoda pokusí získat šablonu od rodičovské entity. Toto volání je opakováno dokud není v hierarchii nalezena entita s přiřazenou šablonou. Entita na nejvyšší úrovní hierarchie musí mít šablonu přiřazenou vždy. Databázová struktura řízení přístupových práv je zobrazena na obr. A.1.

Guest | Author | Registered | Moderator | **Admin** | Webmaster

ID	Name	State	Created by	View	Edit	Block	Delete
1	Article 1	CREATED	Author	No	No	No	No
2	Article 2	PUBLISHED	Author	Yes	No	Yes	Yes
3	Article 3	BLOCKED	Author	Yes	No	Yes	Yes
4	Article 4	DELETED	Author	Yes	No	No	Yes

Obr. 6.2: Příklad použití ACL s implementací stavů.

6.4 Formuláře

Formuláře jsou v Nette frameworku velice dobře implementovány. Z hlediska znovupoužitelnosti však není vhodné je vytvářet přímo v presenterech, ale pomocí továrních tříd. Pro další zvýšení znovupoužitelnosti jsou formuláře vytvářeny uvnitř komponenty, což umožňuje využití ajaxu a načítání dodatečných dat do formuláře nezávisle na presenteru, kde bude formulář použit.

Vzhledem k tomu, že každá komponenta má vlastní šablonu a je možné implementovat několik vykreslovacích metod s různými šablonami, tak je možné velice snadno stejný formulář vykreslit různými způsoby pouhým přiřazením vykreslovací šablony opět nezávisle na presenteru.

Zpracování formuláře probíhá rovněž uvnitř komponenty. Při zpracování se vyvolají události *onSave*, případně *onError*, které umožňují v každém presenteru, kde bude formulář použit provádět další úkony, například vyvolání přesměrování nebo zobrazení vlastní chybové zprávy v případě selhání zpracování dat.

Tyto formulářové komponenty jsou vytvářené pomocí generovaných továrniček nette frameworku. Generované továrničky jsou tvořeny rozhraním, které implementuje jedinou metodu *create* s anotací pro typ návratové hodnoty. Tomuto zápisu nette framework rozumí a potřebnou tovární třídu vygeneruje sám včetně předání závislostí pomocí DI.

Příklad implementace formulářů

Na obr. A.3 je znázorněn UML diagram implementace programové struktury pro formuláře novinek.

Abstraktní třída *BaseForm* je společný předek pro všechny formuláře. V této třídě je instancován formulář *BootstrapForm* 6.4.1, je mu předána instance překladače a nastavena ochrana proti Cross-Site Request Forgery. Zároveň je zde nastaveno zpracování formuláře metodou *formSucceeded*. Tato metoda je abstraktní, což zajišťuje, že zde nemusí být definována její implementace, ale potomci této třídy tuto metodu musí již implementovat musí.

Metoda *render* zajišťuje nastavení příslušné šablony a vykreslení formuláře. Ve výchozím stavu se bere hodnota proměnné *templateName*, která je defaultně nastavena na automatické vykreslování pomocí doplňku BootstrapForms 6.4.1. V případě potřeby je kdykoliv možné nastavit jinou šablonu, případně vytvořit další vykreslovací metodu s jinou šablonou.

Třída *NewsForm* je rovněž abstraktní, čímž je zajištěno, že sama o sobě nemůže být instancována. V této třídě jsou definovány jednotlivé formulářové položky v metodě *setupForm*.

Pomocí třídy *NewsCreateForm* je realizováno vytváření nových záznamů. Pro vytvoření novinky je třeba znát kontejner, ke kterému má být novinka přiřazena. Jeho instance je předána pomocí metody *setNewsContainer*. Tato metoda je volána uvnitř generované továrničky, které je předána hodnota z parametru metody *create*. V metodě *createComponentForm* je vytvořen vlastní formulář, který je nastaven pomocí metody *setupForm* z rodičovské třídy. Protože tato třída není abstraktní, tak je zde již nutné implementovat zpracování formuláře metodou *formSucceeded*.

Třída *NewsEditForm* je implementována podobně jako třída *NewsCreateForm*. Místo kontejneru je jí však předána instance editované novinky pomocí metody *setNews*, která je využita pro nastavení výchozích hodnot formuláře v metodě *createComponentForm* zároveň je využita i v metodě *formSucceeded*, kde jsou do příslušného záznamu uloženy provedené úpravy.

Tato programová struktura umožňuje co možná největší znovupoužitelnost a flexibilitu, díky které lze s minimem opakování kódu záznamy vytvářet, editovat a vykreslovat různými způsoby.

6.4.1 BootstrapForms

Tento projekt vyžaduje použití velkého množství formulářů. Nette framework podporuje automatické vykreslování formulářů voláním makra v latte šabloně. Toto vykreslování je však realizováno zastaralým řešením formátování pomocí tabulky, které není responzivní, viz obr. B.5. Při použití Bootstrapu je navíc toto vykreslování nekompatibilní s nastavením CSS Bootstrapu a stane se tak prakticky nepoužitelným.

Pro nahrazení výchozího vykreslování vznikl v rámci tohoto projektu doplněk BootstrapForms, který formátování pomocí tabulky nahrazuje řešením implementovaném ve frameworku Bootstrap, viz obr. B.6. Toto řešení je responzivní a je plně kompatibilní s Bootstrapem. Podporuje 3 různé způsoby vykreslení formuláře. Jedná se o výchozí rozložení, které je použito na příkladu s registračním formulářem (obr. B.6), dále pak široké rozložení, kdy je formulář rozložen na celou šířku stránky a o jednořádkový formulář, kdy je formulář vykreslen do jednoho řádku.

Standardní funkce formulářů byly rozšířeny o téměř všechny funkcionality z frameworku Bootstrap. Jedná se například o nastavování velikostí inputů a tlačítek, podporu ikonek nebo prefixy/suffixy pro textové prvky.

6.5 Drobečková navigace

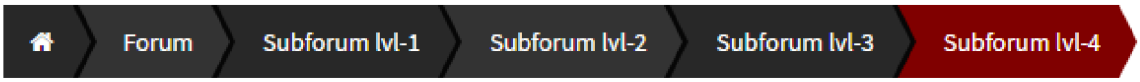
U složitější struktury webu, jako je tento je vhodné použít drobečkovou navigaci, která výrazně zvýší pohodlí návštěvníků a usnadňuje jejich orientaci. Pro Nette framework existuje hned několik doplňků, které drobečkovou navigaci řeší. Bohužel žádný z nich není uzpůsoben pro pohodlné vkládání zanořených položek jakými jsou například diskusní fóra, která mohou mít několik zanořených úrovní jejich potomků.

Pro vyřešení tohoto problému byla vytvořena vlastní komponenta, která jednotlivé položky uchovává pomocí obousměrně vázaného lineárního seznamu. Tato datová struktura umožňuje vkládání položek na libovolné místo v seznamu a tedy i snadné vkládání zanořených prvků do navigace. Její UML diagram je zobrazen na obr. A.4.

Třída *Breadcrumb* tvoří samotnou komponentu drobečkové navigace. Uchovává si ukazatel na první a poslední prvek seznamu a na poslední vložený prvek. Díky těmto ukazatelům lze velice efektivně vkládat jednotlivé položky navigace, které jsou tvořeny objekty typu *BreadcrumbNode*. Mimo metod pro vkládání položek na různé pozice v navigaci obsahuje třída *Breadcrumb* metodu *setHomepage* pro nastavení domovské stránky, která je v navigaci vložena při vykreslování vždy na první místo. Metoda *resetLastInsertedNode* nastaví ukazatel na poslední vložený prvek na hodnotu *null*, což způsobí, že následné volání metody *prependLastInsertedNode* nebo *appendLastInsertedNode* vloží prvek na poslední pozici v navigaci.

Užitečnou vlastností třídy *BreadcrumbNode* je možnost ovlivnit nastavení aktivního prvku v drobečkové navigaci. Aktivní prvek je automaticky detekován při vložení prvku do seznamu, ovšem poté je možné pomocí metody *setIsActive* toto zvýraznění zrušit a stejnou metodou nastavit zvýraznění prvku jinému.

Na obr. 6.3 je vidět příklad drobečkové navigace diskusního fóra, kterou lze snadno vytvořit pomocí jednoho cyklu, aniž by byl programátor nucen si nadřazené fóra ukládat například do pomocného pole a poté prvky vkládat v obráceném pořadí.



Obr. 6.3: Drobečková navigace zanořeného diskusního fóra.

6.6 Komponenta pro hlavní navigaci

Pro snadnější vytvoření uživatelského rozhraní byla v projektu použita šablona Admin LTE [1] postavená na frameworku Bootstrap. Mezi její velké přednosti patří propracovaná navigace. Tuto navigaci by ovšem bylo složité udržovat a proto byla napsána komponenta, které se stará o sestavení a vykreslení hlavní navigace.

Z UML diagramu na obr. A.5 je patrné, že všechny prvky navigace jsou potomky třídy *Aplication\UI\Control*. Každý prvek má tak vlastní šablonu a je možné jej vykreslit voláním metody *render*. Výhodou tohoto řešení je snadnější vykreslování, kdy stačí získat z instance třídy *SidebarMenu* všechny k ní přiřazené komponenty a ty poté vykreslit voláním metody *render* v cyklu foreach.

Takováto implementace komponent se v Nette frameworku běžně nepoužívá. Standardní struktura bývá taková, že samotná komponenta je tvořena potomkem třídy *Control* a tvoří tak jakýsi vykreslitelný kontejner, zatímco její prvky jsou potomky třídy *Component*. Při takovéto implementaci se však vykreslování prvků stává složitější, kdy je nutné v šabloně kontejneru zjišťovat datové typy jednotlivých prvků a podle toho je vykreslovat pomocí různých podmínek.

Pro flexibilnější sestavování navigace obsahuje třída *LinkGroup* metodu *expand*, která rozbalovací menu „rozbalí“ a vykreslí v podobě nadpisu a jednotlivých odkazů. Tato metoda je použita v seznamu guild uživatele, kdy se uživatel nachází v nejvyšší vrstvě portálu. Tento seznam je zde „rozbalen“ pro pohodlnější přechod uživatele do jeho guild. Navigaci tedy není třeba sestavovat dvojím způsobem a voláním této metody lze snadno ovlivnit strukturu navigace.

6.7 Použití a implementace API

Při tvorbě portálu bylo použito několik aplikačních rozhraní. Jedná se o Steam Web API, Battle.net game API a Microsoft Translator text API. Fungování Steam API a Battlenet API bylo již popsáno v kapitole 5.

Steam Web API

Steam Web API bylo použito k přihlašování a registraci uživatelů pomocí standardu OpenID. Implementace toho API je realizována pomocí třídy SteamApiService. Samotné přihlášení uživatele pomocí OpenID je realizováno knihovnou LightOpenID [13], pomocí které se získává *SteamId*, což je unikátní identifikátor uživatele na službě Steam. Pro získání informací o steam profilu uživatele je použita knihovna bitbang/http [23], která je určena k vytváření HTTP požadavků a zpracovávání příchozích odpovědí. Získaná data jsou v konstruktoru předána třídě přepravky, která má implementovanou sadu getterů informací ze získaných dat.

Battle.net game API

Battle.net game API bylo rovněž použito pro přihlašování a registraci uživatelů pomocí protokolu OAuth2. Implementace API je opět realizována pomocí samostatné služby. Třída *BattleNetApiService* komunikuje se vzdálenými servery opět pomocí knihovny bitbang/http. Toto API je mimo přihlašování uživatelů využito i k získání výchozích dat pro hru World of Warcraft. Jedná se například o seznam serverů, herních instancí, herních povolání nebo ras.

Veškerá získaná data jsou vždy vracena v přepravce. Tento způsob předávání dat se v průběhu programování aplikace velice osvědčil. Vrácený objekt má v IDE plně funkční napovídání dostupných getterů, což zlepšuje efektivitu psaní kódu. V případě změn ve struktuře získaných dat ze strany poskytovatele má navíc přepravka nespornou výhodu v tom, že stačí upravit příslušný getter a zbytek aplikace zůstane změnou neovlivněn.

Této výhody bylo využito při zpracování dostupných herních ras ve hře. Každá herní rasa má své identifikační číslo, jméno a číslo herní frakce, ke které patří. Výjimku tvoří rasa zvaná *pandaren*. Tato rasa je v získaných datech zastoupena 3x a pokaždé s jiným identifikačním číslem - jednou pro frakci aliance, jednou pro frakci hordy a jednou jako neutrální rasa. Tato vrácená data nejsou kompatibilní s implementovaným datovým modelem portálu, který počítá vždy jen s jednou rasou s unikátním názvem, kdy dostupnost rasy pro různé frakce je zaznamenána ve vazební tabulce reprezentující vazbu M:N mezi herní frakcí a rasou. Pro tuto rasu

bylo tedy vybráno jen jedno ze 3 možných identifikačních čísel a v getteru přepravky jsou zbylá dvě identifikační čísla přepsána.

Microsoft Translator Text API

Microsoft Translator Text API [17] je použito k získávání výchozích hodnot překladů pro jazyky zavedené v systému. Stejně jako dvě předešlá API je i toto implementováno vlastní službou *TranslatorApiService*. Pro získání překladu musí být daná aplikace autorizována pomocí protokolu Oauth2. O autorizaci a samotné získání překladu se stará knihovna matthiasnoback/microsoft-translator [21].

6.8 Uživatelské účty

Každý uživatel se do systému může registrovat a přihlašovat několika způsoby. Pomocí služby Steam, která je poskytovatelem standardu OpenID, pomocí Battle.net game API využívajícím protokol OAuth2 a nebo standardním způsobem pomocí hesla a emailové adresy. Při registraci vždy probíhá kontrola, zda již není daný email nebo účet třetí strany v systému zaveden.

Na obr. A.6 je znázorněn ERD diagram uživatelských účtů, kde jsou k uživateli přidruženy různé účty pomocí vazby 1:1. Tento datový model umožňuje praktický neomezený počet způsobů registrace a přihlašování uživatele. Pro přidání další přihlašovací služby stačí vytvořit příslušnou tabulku s potřebnými daty a tu navázat k entitě uživatele. Na způsobu přihlašování uživatele nezáleží, protože při procesu přihlašování se vždy získá ID uživatele z hlavní tabulky *user* a uživatel je přihlášený v systému právě pod tímto ID. Uživatelům je umožněno mít připojených více různých způsobů přihlašování najednou. Připojení dalších způsobů přihlašování je možné v profilu uživatele.

6.9 Lokalizace

Ovládací prvky portálu je možné lokalizovat do jakéhokoli jazyka. Samotný překlad je prováděn třídou *Translator*, která implementuje rozhraní *ITranslator* - viz A.7. Pokud v aplikaci není nastavený jazyk, tak se jej aplikace pokusí rozpoznat. Pokud se rozpoznání jazyka nepodaří nebo podporovaný jazyk není v seznamu dostupných jazyků, tak je nastavena angličtina. Nastavený jazyk je poté při inicializaci aplikace předán instanci překladače, který si z databáze načte všechny dostupné překlady pro daný jazyk.

Voláním metody *translate* se překladač pokusí nalézt požadovaný překlad. Pokud daný překlad neexistuje nebo se nachází ve stavu *PENDING*, tak se překladač

pokusí nalézt překlad pro výchozí jazyk, kterým je angličtina. Když překlad není nalezen ani ve výchozím jazyce, tak jsou do databáze automaticky zaneseny požadavky na překlad daného řetězce pro všechny jazyky zavedené v systému a pro každý z nich je získán strojový překlad pomocí Microsoft Translatoru [17].

Datový model lokalizace je tvořen třemi tabulkami, jak je vidět na obr. A.8. Každý překlad je uchováván společně s jeho jazykem, identifikátorem a uživatelem, který ho naposledy upravoval. V tabulce *translation* je pro zajištění unikátnosti překladu nastaven složený unikátní klíč ze sloupců *identifier* a *language_id*.

Při realizaci překladače byla prozkoumána možnost překladu slov jak v jednotném, tak v množném čísle. Tento překlad je realizován protected metodou *translatePlurals*. Tato funkce však nebyla nikde potřeba, takže tato metoda zůstala v překladači přítomná jen v testovací formě. Metoda využívá k překladům výrazy pro určování tvaru plurálu z knihovny Gettext [14]. Tyto výrazy jsou poté upraveny pro možnost zpracování funkcí *eval*. Tato funkce představuje bezpečnostní riziko, protože řetězec v parametru funkce zpracuje jako PHP kód. Toto riziko je eliminováno validací výrazu pro určení tvaru plurálu pomocí regulárního výrazu, který v případě nalezení jiného znaku, než je číslo, znak n, znaky pro porovnávání a logické operátory *AND* a *OR* okamžitě vyhazuje výjimku a ukončuje zpracovávání překladu.

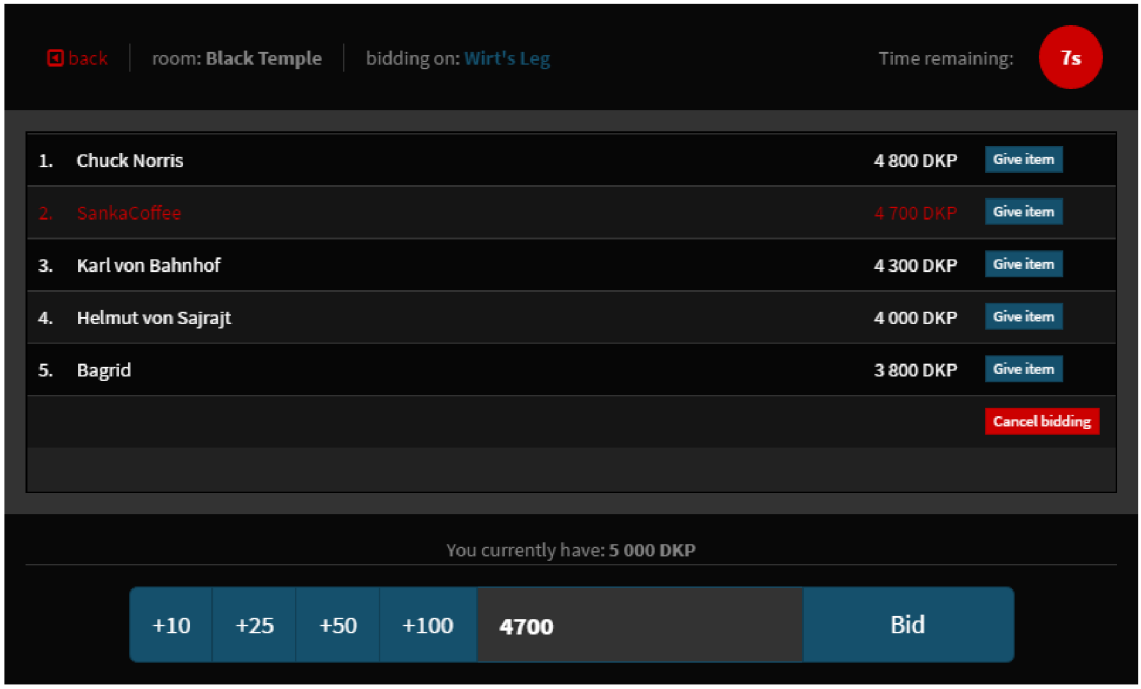
6.10 Implementace systému DKP

Systém DKP popsaný v kapitole 4.3 je úzce provázán s událostmi. Každá událost má možnost vygenerovat místnost pro správu DKP a dražení získaných předmětů. Uživatel, který založil událost a má práva pro správu DKP, může poté v dané místnosti přidělovat/odebírat DKP uživatelům přihlášeným na danou událost. Dále má možnost vyhlásit dražbu předmětu po určitý čas. Všichni uživatelé v místnosti poté mohou daný předmět dražit. Správce DKP poté daný předmět přidělí některému z dražitelů. Správci je umožněno přidělit předmět jakékoli nabídce. Předmět tedy nemusí být nutně přidělen jen hráči s nejvyšší nabídkou, pokud správce rozhodně jinak. Dále má možnost dražbu zrušit nebo po jejím ukončení opakovat.

Datový model je zobrazen pomocí ERD diagramu na obr. A.9. DKP jsou vázány na uživatele a guildu, ve které byla událost založena. Body tedy nejsou přenosné mezi guildami. Informacemi o místnosti a daty z průběhu dražby není třeba zatěžovat databázi. Všechna tyto data jsou ukládána do souboru v cachi a po opuštění místnosti vymazána.

Všichni uživatelé v místnosti potřebují vidět její aktuální stav. Toho je docíleno periodickým voláním ajaxového požadavku každou sekundu, který zajistí překreslování částí místnosti tak, aby zobrazovaná data byla vždy aktuální.

Do DKP místnosti budou hráči nejspíše přistupovat z mobilních zařízení, aby nemuseli opouštět hru. Z tohoto důvodu byl při návrhu kladen co největší důraz na responzivitu a pohodlné ovládání z mobilních zařízení. Na obr. 6.4 je zobrazen průběh dražby předmětu na obrazovce s rozlišením 800x480px. Každý uživatel zde vidí celkový počet svých DKP, jednotlivé příhozy seřazené dle jejich hodnoty, jméno předmětu který je dražen a čas zbývající do ukončení přihazování.



Obr. 6.4: Průběh dražby předmětu na obrazovce s rozlišením 800x480px.

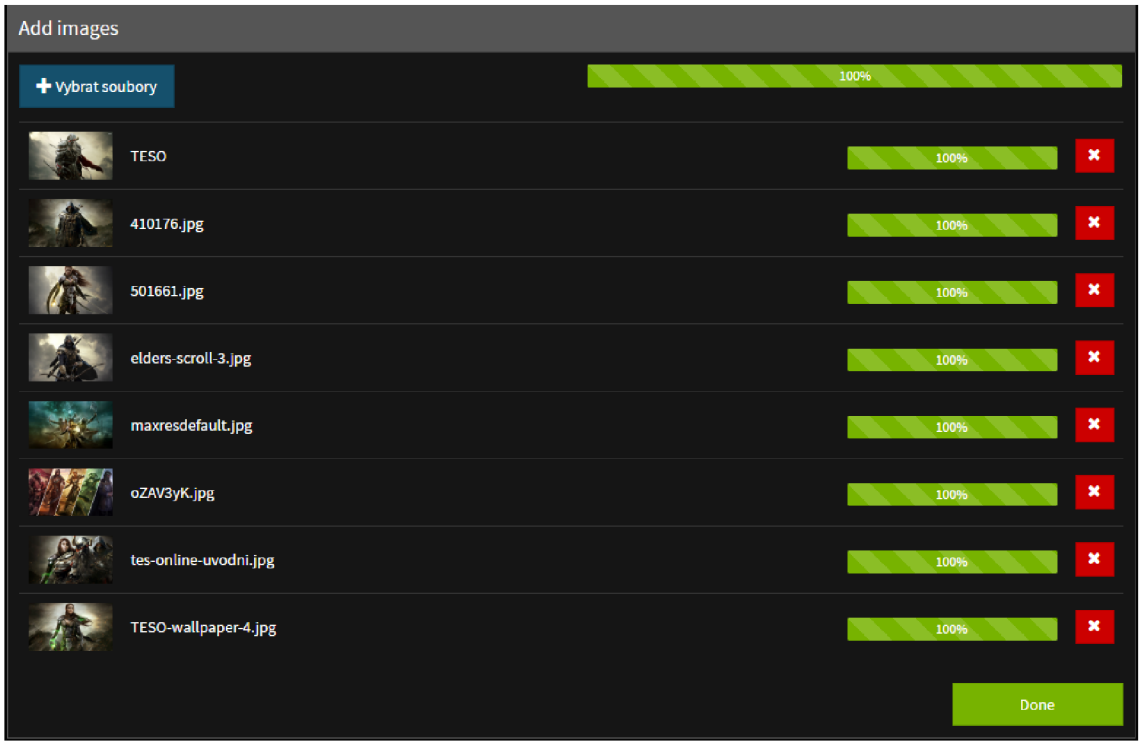
6.11 Galerie

Praktický žádný herní web se neobejde bez galerie. Implementované galerie jsou plně responzivní, čehož je dosaženo kombinací grid systému a miniatur obrázků z frameworku Bootstrap. Pro zobrazování zvětšených obrázků byl použit jQuery plugin Viewer [25], který je velice propracovaný a na rozdíl od většiny používaných doplňků pro zobrazování obrázků, se kterými se lze běžně na internetu setkat, má ovládací prvky vždy na jednom místě. Toto bylo hlavní kriterium při výběru vhodného doplňku, protože neustálé přesouvání ovládacích prvků při různých rozměrech obrázků může být pro uživatele velice frustrující. Navíc má tento doplněk několik zajímavých funkcí, jako je otáčení obrázků, jejich zrcadlení a prezentační mód.

Pro snížení datové zátěže je tento doplněk schopný načítat zvětšené obrázky až ve chvíli jejich zobrazování, čímž se značně sníží přenášený objem dat. Generování

různých velikostí obrázků je na serveru rovněž prováděno až v případě potřeby pomocí doplňku WebChemistry/Images [11], čímž se ušetří značná kapacita úložiště serveru.

Pro uživatele komfortnější nahrávání obrázků do galerie je použit doplněk jQuery-FileUpload [12]. Tento doplněk obaluje klasický upload do uživatelsky přívětivějšího grafického rozhraní, viz obr. 6.5. Nahrané obrázky lze ještě před odesláním formuláře přejmenovat, případně odstranit.



Obr. 6.5: Grafické rozhraní pro nahrávání obrázků pomocí doplňku jQuery-FileUpload.

Použití tohoto doplňku s sebou přináší problém se zpracováváním obrázků na serveru. Každý obrázek je nahráván a zpracováván samostatně okamžitě po vybrání souborů na lokálním disku uživatele. Při odeslání formuláře jsou získána data, která jsou vrácena v návratové hodnotě metody *save* zpracovávající jednotlivé obrázky během jejich nahrávání. Dočasný soubor obrázku na serveru již samozřejmě není, takže je nutné jej zpracovat ihned v metodě *save*. Vzhledem k tomu, že data z databáze nejsou fyzicky mazána, ale jsou mazána pouze pomocí stavu, tak by v případě odstranění souboru uživatelem nebo opuštění stránky, čímž by proces přidávání nových obrázků nebyl dokončen, byla databáze i úložiště serveru zbytečně zanášeny nepotřebnými daty a soubory.

Tento problém je vyřešen dočasným uložením obrázku do cache, který je pře-

veden na textový řetězec a uložen do dočasného souboru. Dočasným souborům je nastavena expirace na 10 minut a všechny dočasné soubory starší než 10 minut jsou poté nevalidní a při příštím použití doplňku smazána. Po odeslání formuláře jsou příslušné soubory uloženy do složky spravované doplňkem WebChemistry/Images [11] a v databázi je obrázek přiřazen k příslušné galerii. Zpracované soubory z cache jsou při ukládání vymazány. Tímto řešením jsou do databáze vloženy pouze relevantní data a je docíleno minimálního zatížení úložiště serveru, kdy jedinými „nežádoucími“ soubory jsou dočasné soubory z posledního nedokončeného nahrávání.

6.12 Layout

Oproti původnímu návrhu, který je na obr. B.1, B.2 a B.3 byly jednotlivé části portálu sjednoceny a mají jeden společný layout. Jeho obsah se liší pouze prohlíženou sekcí a přístupovými právy uživatele. Ukázka současného vzhledu portálu je na obr. B.4.

Layout stránky je tvořen šablonou Admin LTE [1], která byla přestylována do tmavých barev. Tato šablona je postavena na frameworku Bootstrap a obohacuje jen o několik užitečných komponent. Jedná se například o efektní postranní panel pro navigaci nebo časovou osu, která byla využita pro zobrazení nadcházejících událostí.

I přes to, že zdrojové kódy šablony jsou k dispozici ve formě preprocessoru LESS, tak nastavení jednotlivých parametrů jako jsou zaoblené rohy prvků nebo barvy textu a pozadí jsou v drtivé většině zapsány ve zdrojových kódech přímo a nikoli pomocí proměnných, takže úprava vzhledu šablony se tím stala zbytečně komplikovanou.

Dalším problémem této šablony, který se projevil při jejím užívání je implementace některých prvků, kdy jejich responzivita je zajištěna pomocí javascriptu. Jde zejména o pravý postranní panel, který by byl v budoucnu vhodný na umístění chatu a centra notifikací. Při změně velikosti okna je nad ním vyvolána javascriptová funkce, která při změně tažením zahltí prohlížeč a je třeba danou stránku načíst znovu.

V budoucnu bude tedy pravděpodobně vhodnější tuto šablonu nahradit. Jako ideální volba se zatím jeví nová verze frameworku Bootstrap, která se momentálně nachází v alfaverzi. Bootstrap 4 nově implementuje některé komponenty, kvůli kterým byla použita šablona Admin LTE a volitelně umožňuje použití css vlastnosti flexbox. Flexbox umožňuje snadné vertikální zarovnávání obsahu, vytvoření stejně vysokých prvků na stránce, či změnu pořadí vložených prvků. Tyto vlastnosti jsou v současné době často řešeny pomocí javascriptu, což mívá za následek snížení výkonu aplikace.

7 ZÁVĚR

Cílem této práce bylo navrhnout a implementovat programovou strukturu pro herní webový portál, která by byla dostatečně obsáhlá na to, aby pokryla potřeby hráčů a správců obsahu portálu. Dále měl být implementován systém dragon kill poins, který hráčům usnadní přerozdělování získaných herních předmětů. Celý portál měl podporovat lokalizaci do jiných jazyků a pro zvýšení komfortu uživatelů implementovat aplikační rozhraní jiných služeb.

V teoretické části práce byly určeny potřeby hráčů online her. Na základě těchto potřeb byly navrženy komponenty, které budou schopny tyto potřeby dostatečně pokrýt. Dále zde byly prozkoumány již existující řešení této problematiky, určena jejich slabá místa a navrženy mechanismy, které by měly tyto nedostatky řešit. Jedná se o lokalizaci a implementaci druhé vrstvy portálu, která má za úkol udržovat databáze informací týkající se konkrétních her.

V rámci této práce byl navržen kompletní datový model celého portálu, který je tvořen více než 60 databázovými tabulkami. Tento datový model lze využít pro tvorbu mnoha komponent, pro vytváření nového obsahu webu a jeho správy. Jedná se například o ankety, lokalizované návody, či rozšíření událostí o možnost přihlašování se pomocí herních postav. Zároveň byla navržena a připravena programová struktura pro snadnou správu současných a budoucích komponent pomocí fasád. Funkčnost návrhu byla ověřena implementací komponent pro splnění základních potřeb hráčů. Jedná se o novinky, galerie, diskusní fóra a kalendář událostí s možností využití systému DKP.

Aplikačních rozhraní byla implementována v podobě přihlašování/registrace uživatelů a automatickém načítání výchozích dat pro hru World of Warcraft. Dále byly prozkoumány data, která jsou možná pomocí nich získat a možnosti jejich pozdější implementace. Pro usnadnění lokalizace do jiných jazyků bylo implementováno automatické překládání obsahu pomocí Microsoft translatoru.

Při tvorbě portálu bylo použito velké množství formulářů. Z tohoto důvodu byl vyvinut doplněk BootstrapForms, který zajišťuje jejich automatické vykreslování a v současné době je uvolněn jako open-source na GitHubu.

Pro další rozšiřování tohoto projektu by bylo vhodné využít potenciálu navrženého modelu a implementovat další funkcionality jako jsou již navržené ankety, lokalizované návody, automatické načítání postav hráčů pomocí Battle.net API nebo implementovat ochranu guild ve hře World of Warcraft, kdy by guildu mohl založit pouze její skutečný majitel a vstup do ní by byl umožněn jen jejím skutečným členům. Dalším užitečným rozšířením je implementace chatu pomocí websocketu.

LITERATURA

[1] Almsaeed Studio [online]. [cit. 2016-03-03]. Abdullah Almsaeed, 2014 Dostupné z: <https://almsaeedstudio.com/>.

[2] Battle.net API [online]. [cit. 2015-12-03]. Dostupné z: <https://dev.battle.net/>.

[3] Bootstrap: *The world's most popular mobile-first and responsive front-end framework.* [online]. [cit. 2015-11-25]. Dostupné z: <http://getbootstrap.com/>.

[4] Datagrid | Ublaboo [online]. Pavel Janda, 2016 [cit. 2016-02-25]. Dostupné z: <http://ublaboo.paveljanda.com/datagrid/>.

[5] DesignPatternsPHP. *DesignPatternsPHP - DesignPatternsPHP 1.0 documentation* [online]. [cit. 2015-11-17]. Dostupné z: <http://designpatternsphp.readthedocs.org/>.

[6] DOSTÁL, Radek. *DateTimePicker for Nette Framework* [online]. Github Repository [cit. 2015-11-25]. Dostupné z: <https://github.com/radekdostal/Nette-DateTimePicker>.

[7] Doctrine [online]. [cit. 2015-11-25]. Dostupné z: <http://www.doctrine-project.org/>.

[8] Enjin [online]. [cit. 2015-11-10]. Dostupné z: <http://www.enjin.com/>.

[9] GRUNT: *The JavaScript Task Runner* [online]. [cit. 2015-11-25]. Dostupné z: <http://gruntjs.com/>.

[10] IClan Websites [online]. [cit. 2015-11-10]. Dostupné z: <https://www.iclanwebsites.com/>.

[11] Image storage for Nette Framework [online]. MartkCz [cit. 2016-04-19]. Dostupné z: <https://github.com/WebChemistry/Images>.

[12] JQuery-FileUpload: *Nette UploadControl rozšířené o blueimp/jQuery FileUpload* [online]. Jan Zechovský [cit. 2016-04-14]. Dostupné z: <https://github.com/JZechy/jQuery-FileUpload>.

[13] Lightweight PHP5 library for easy OpenID authentication [online]. Ignat Ignatov [cit. 2015-11-15]. Dostupné z: <https://github.com/iignatov/LightOpenID>.

[14] Localization Guide: *Plural Forms* [online]. [cit. 2016-03-16]. Dostupné z: <http://localization-guide.readthedocs.io/en/latest/l10n/pluralforms.html>.

[15] MALÝ, Martin. *OAuth – nový protokol pro autentizaci k vašemu API* [online]. [cit. 2015-12-01]. Dostupné z: <https://www.zdrojak.cz/clanky/oauth-novy-protokol-pro-autentizaci-k-vasemu-api/>.

[16] PECINOVSKÝ, Rudolf. *Návrhové vzory: [33 vzorových postupů pro objektové programování]* Návrhové vzory: [33 vzorových postupů pro objektové programování]. Vyd. 1. Brno: Computer Press, 2007, 527 s. ISBN 978-80-251-1582-4.

[17] Microsoft Translator Text API [online]. Microsoft, 2016 [cit. 2016-05-25]. Dostupné z: <https://www.microsoft.com/en-us/translator/translatorapi.aspx>.

[18] Nette framework [online]. 2008 [cit. 2015-11-10]. Dostupné z: <http://nette.org/>.

[19] Number of World of Warcraft subscribers Q1 2005-Q2 2015. *Statistia* [online]. [cit. 2015-12-08]. Dostupné z: <http://www.statista.com/statistics/276601/number-of-world-of-warcraft-subscribers-by-quarter/>.

[20] OAuth [online]. [cit. 2015-11-25]. Dostupné z: <http://oauth.net/>.

[21] PHP library for the Microsoft Translator V2 API [online]. Matthias Noback [cit. 2016-05-14]. Dostupné z: <https://github.com/matthiasnoback/microsoft-translator>.

[22] Steam Web API *Valve developer community* [online]. [cit. 2015-12-01]. Dostupné z: <https://developer.valvesoftware.com/wiki/Steam_Web_API>.

[23] The straightforward HTTP client [online]. Miloslav Hůla [cit. 2015-11-14]. Dostupné z: <https://github.com/bitbang/http>.

[24] Tiny MCE: *The Most Advanced WYSIWYG HTML Editor* [online]. [cit. 2015-11-25]. Dostupné z: <https://www.tinymce.com/>.

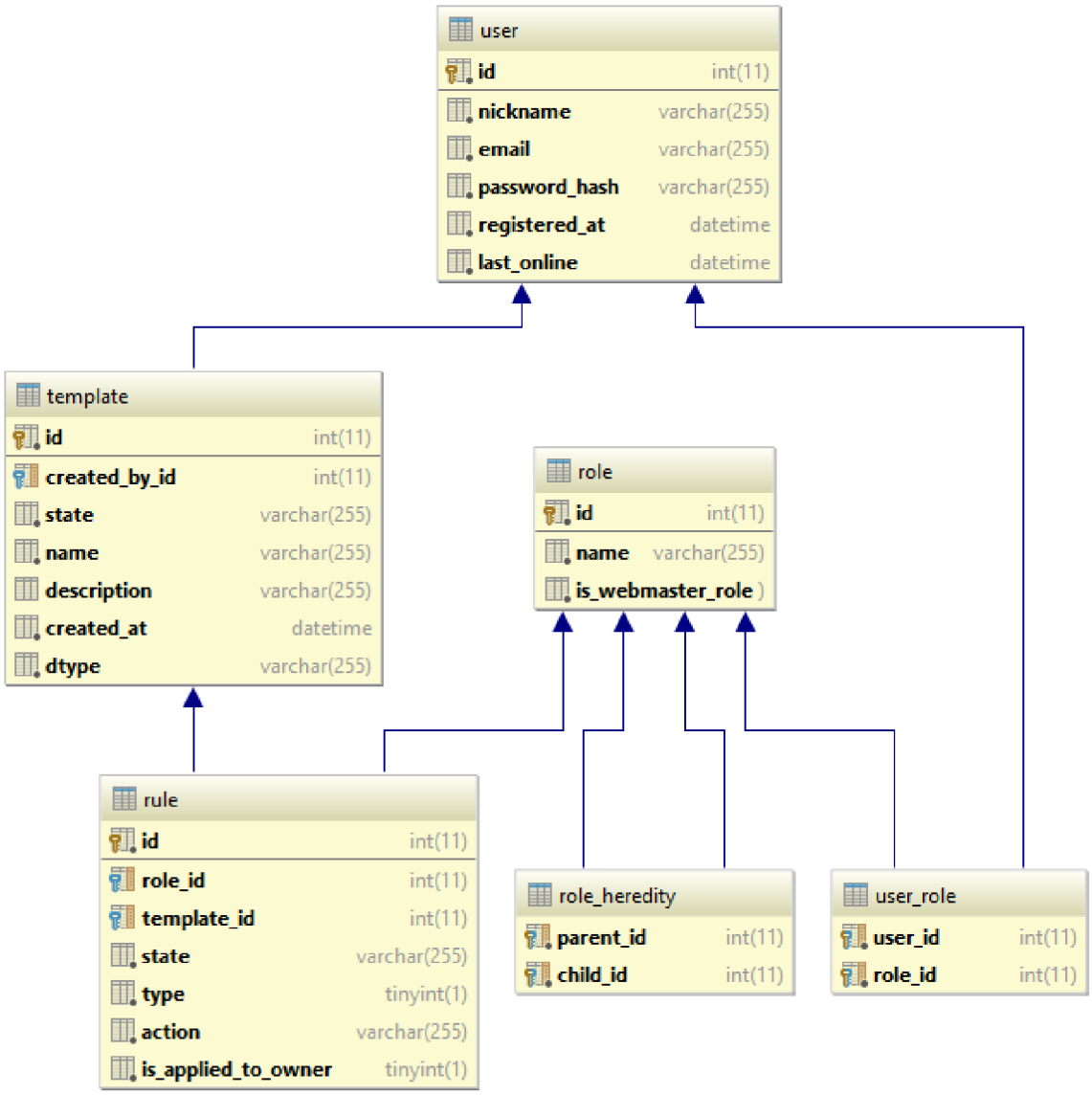
[25] Viewer: *A simple jQuery image viewing plugin* [online]. Fengyuan Chen, 2015 [cit. 2016-03-22]. Dostupné z: <https://fengyuanchen.github.io/viewer/>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

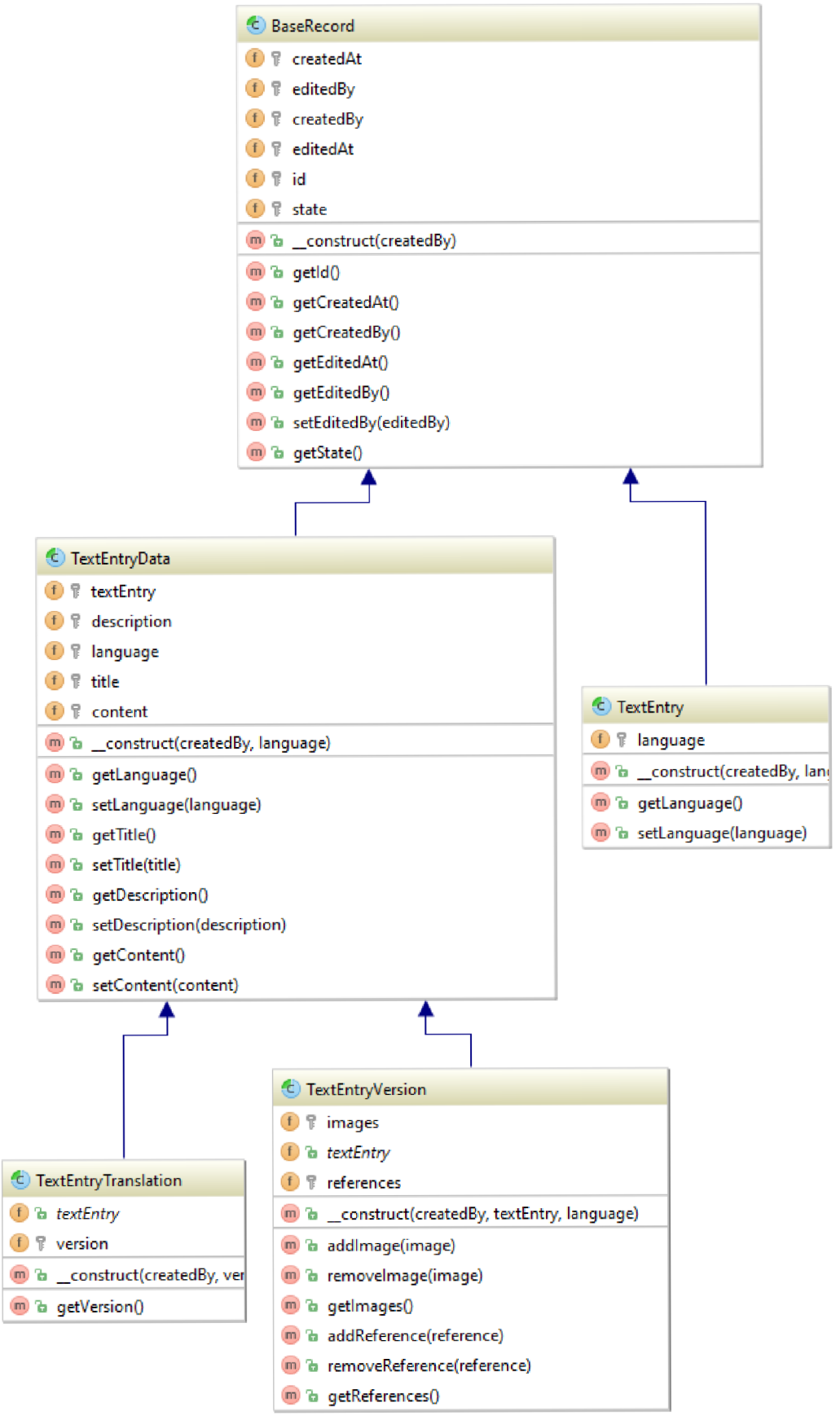
ACL	Access control list
API	Application programming interface
CSS	Cascading style sheets
CSV	Comma-separated values
DI	Dependency injection
DKP	Dragon kill poins
ERD	Entity–relationship diagram
JSON	JavaScript object notation
MMO	Massive multiplayer online
MMORPG	Massive multiplayer online role-playing game
ORM	Object-relational mapping
PHP	Hypertext preprocessor
SQL	Structured query language
UML	Unified modeling language
URL	Uniform resource locator
VDF	Valve data format
VoiP	Voice over internet protocol
WYSIWYG	What you see is what you get
XML	Extensible markup language

SEZNAM PŘÍLOH

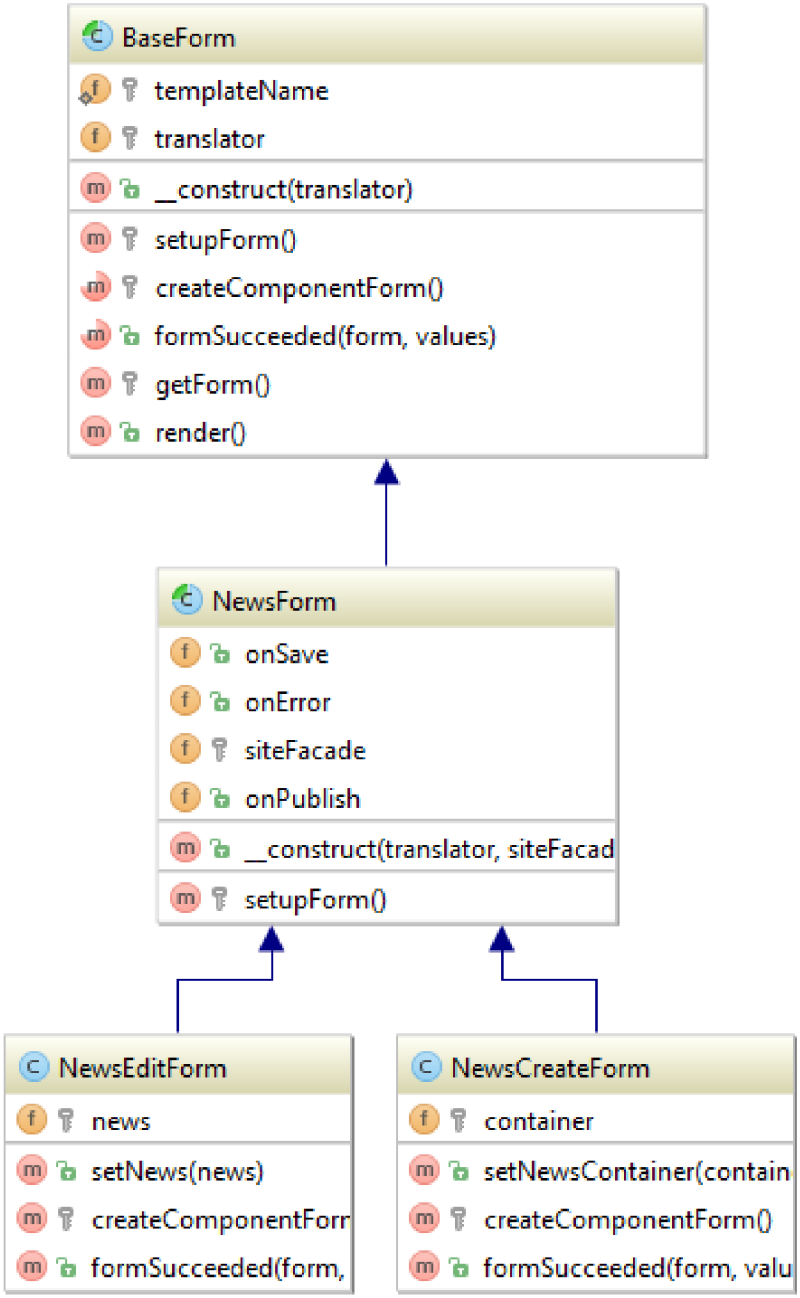
A UML A ERD DIAGRAMY



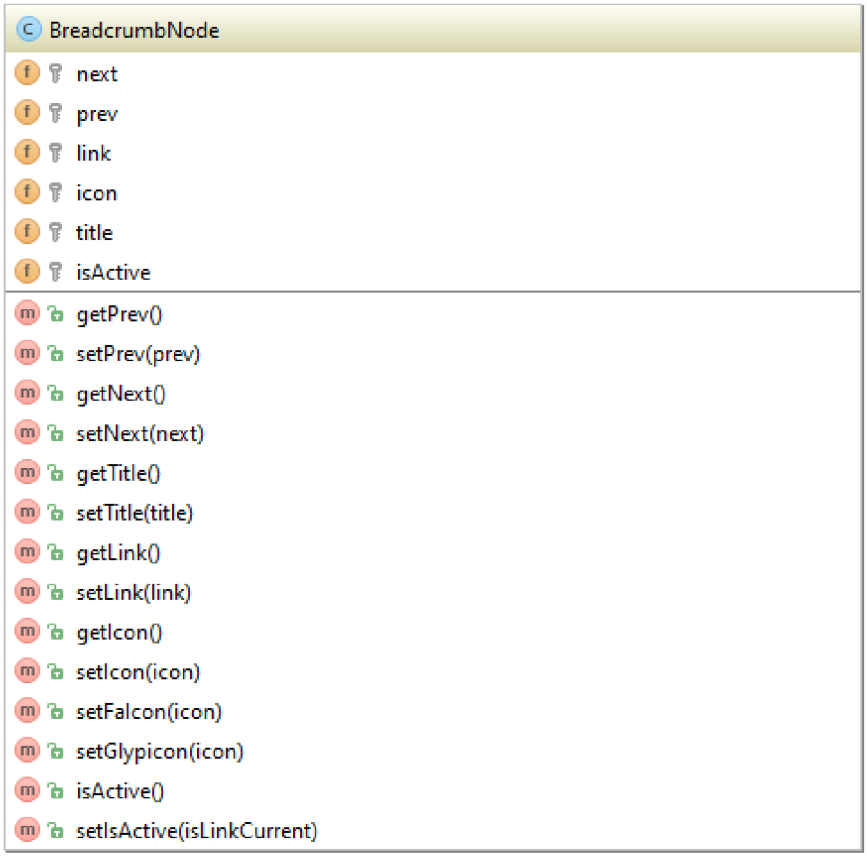
Obr. A.1: ERD diagram řešení přístupových práv.

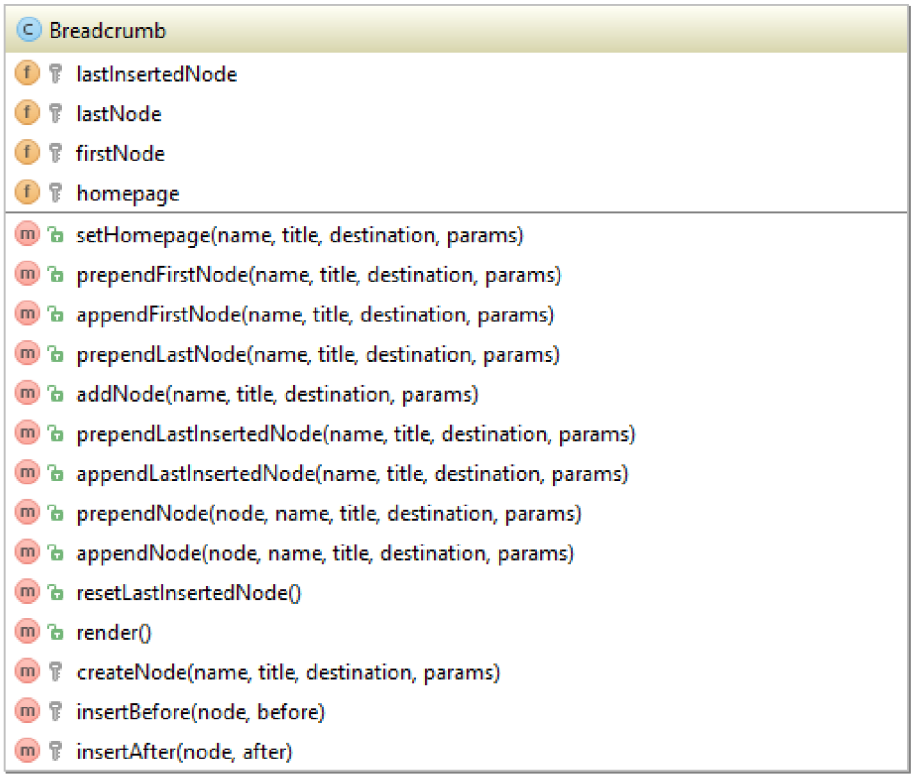


Obr. A.2: UML diagram řešení verzování a lokalizace dokumentů.

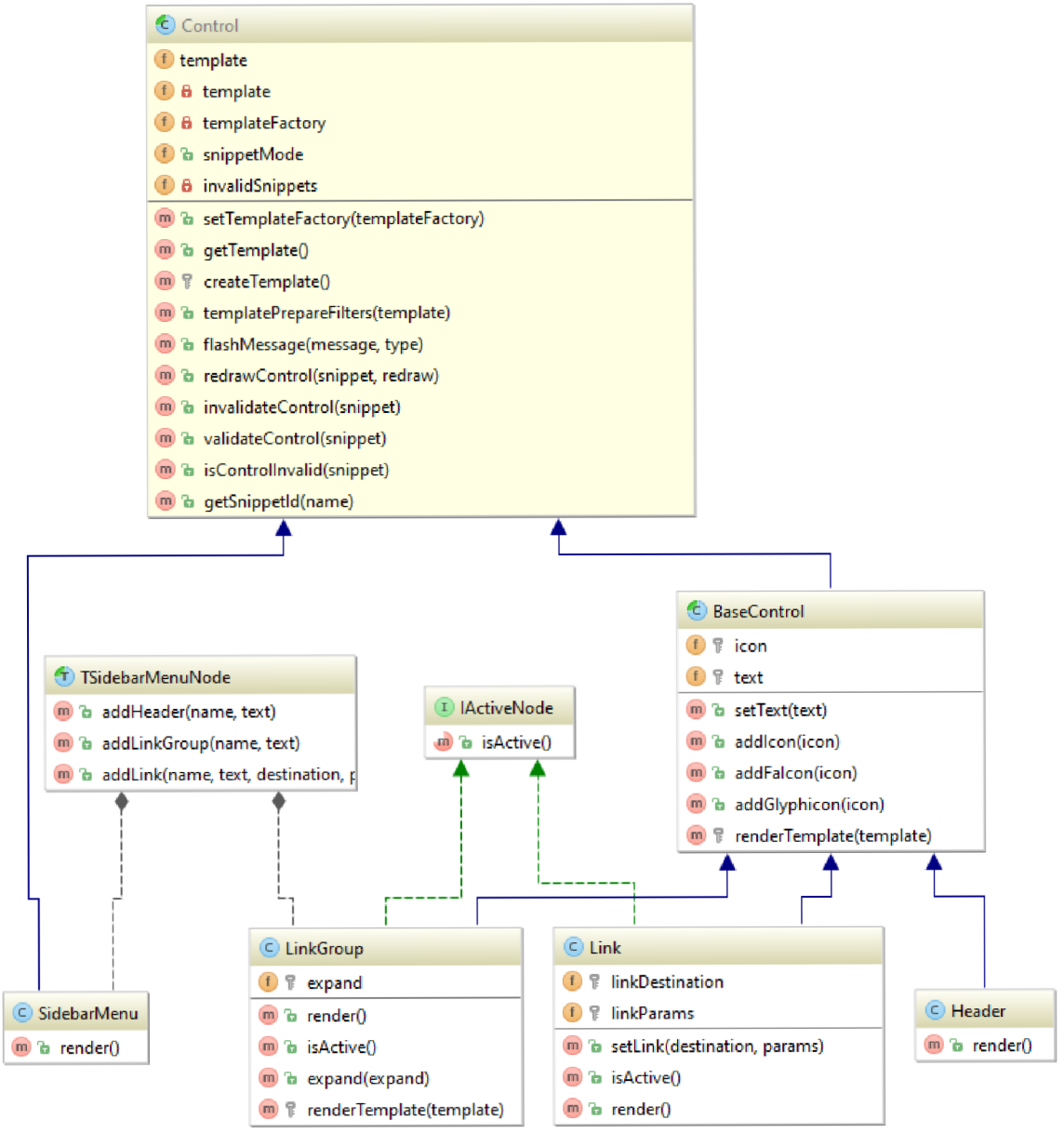


Obr. A.3: UML diagram implementace formulářů pro správu novinek.

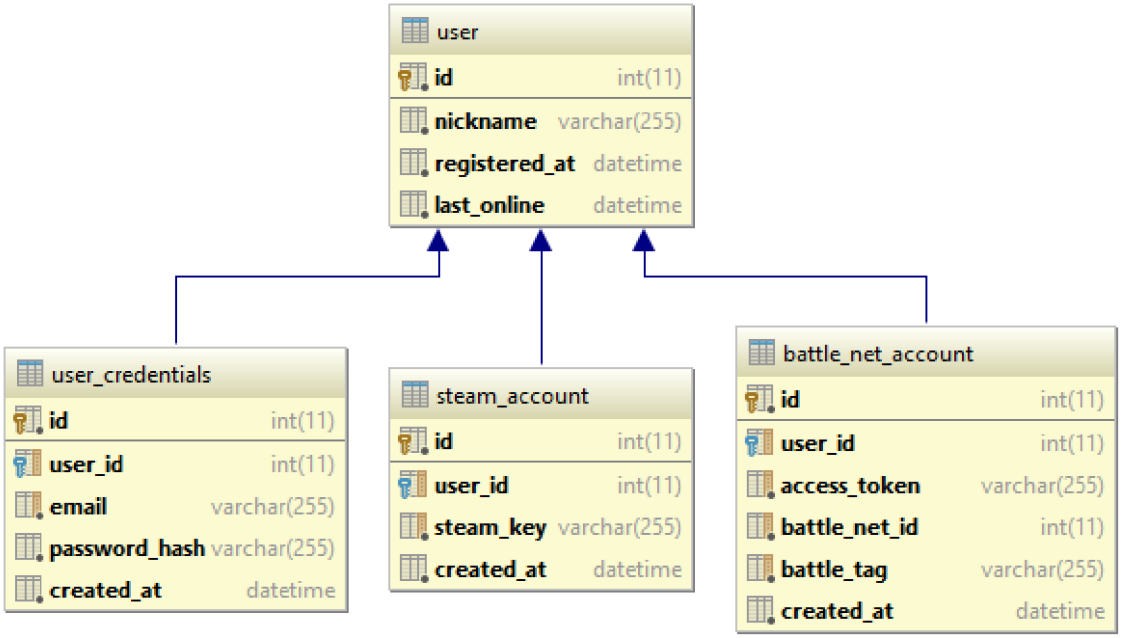




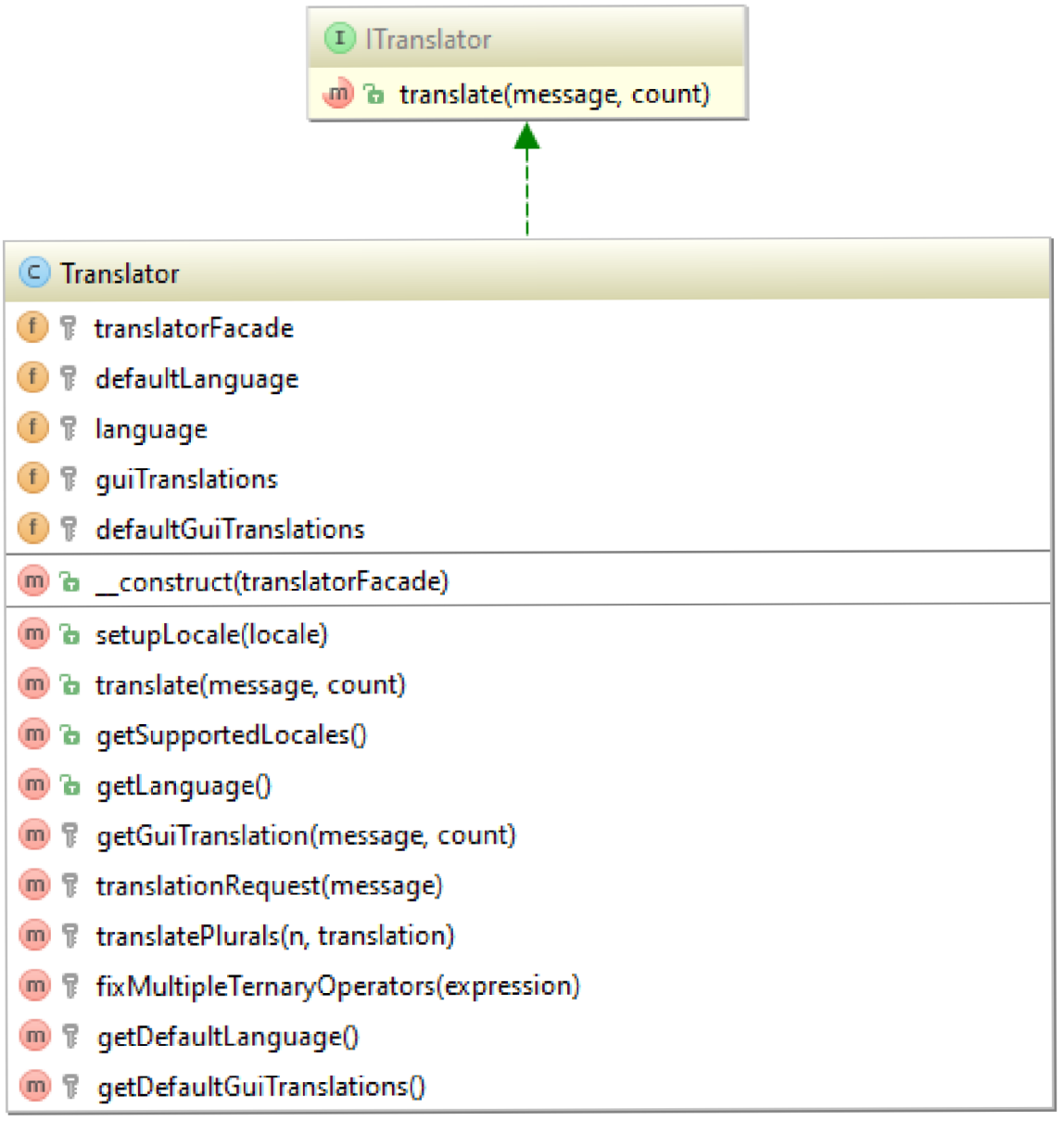
Obr. A.4: UML diagram implementace komponenty drobečkové navigace.



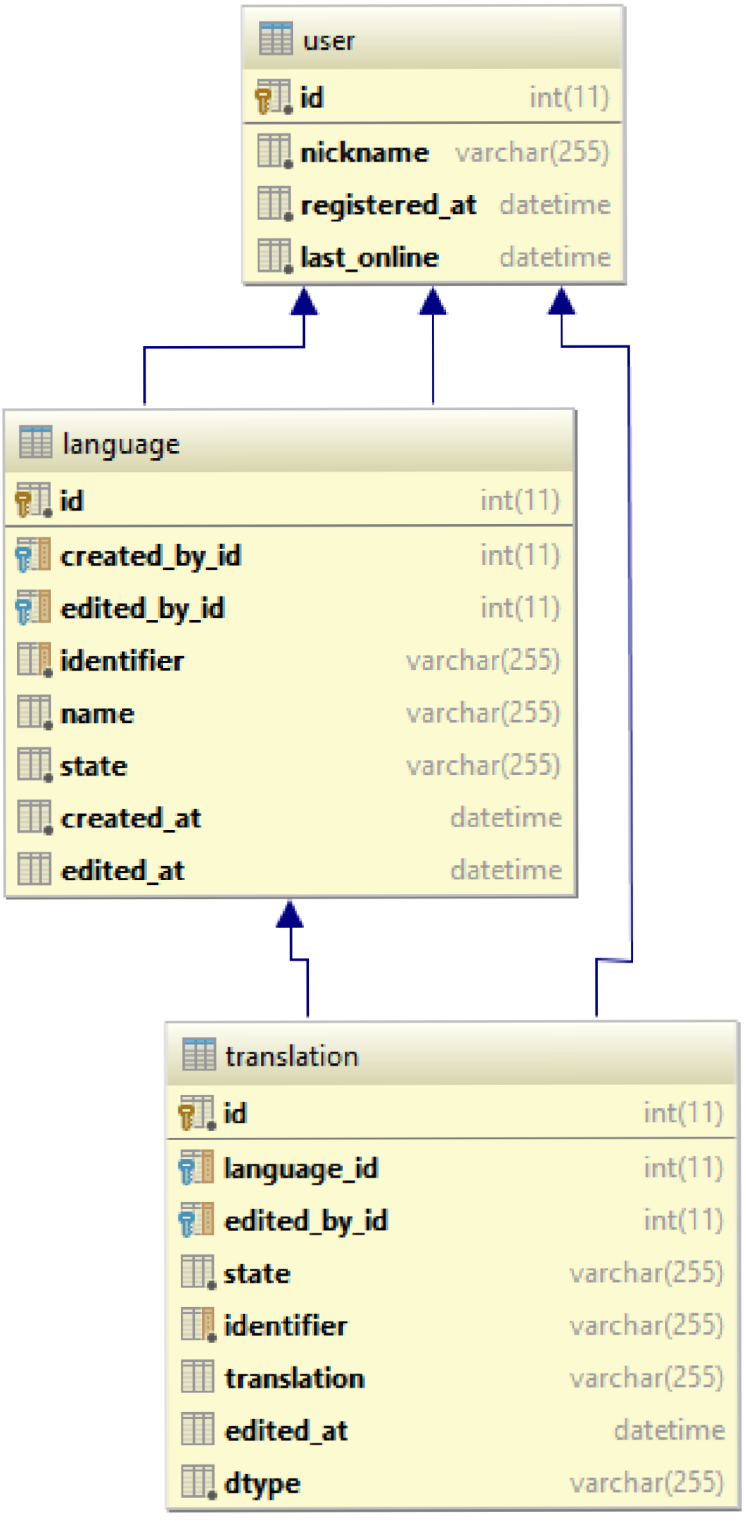
Obr. A.5: UML diagram implementace komponenty navigace Admin LTE.



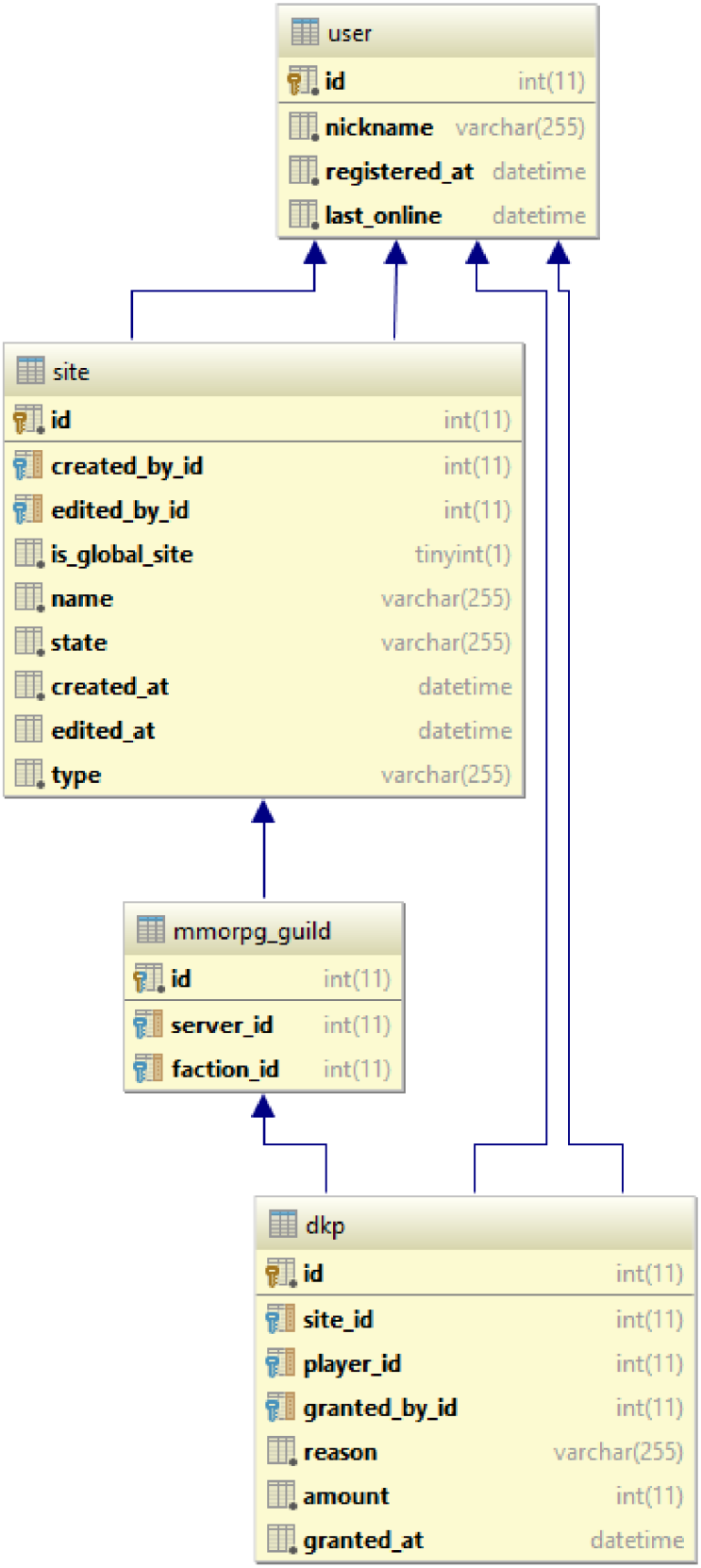
Obr. A.6: ERD diagram uživatelského účtu a přidružených služeb.



Obr. A.7: UML diagram implementace překladače.

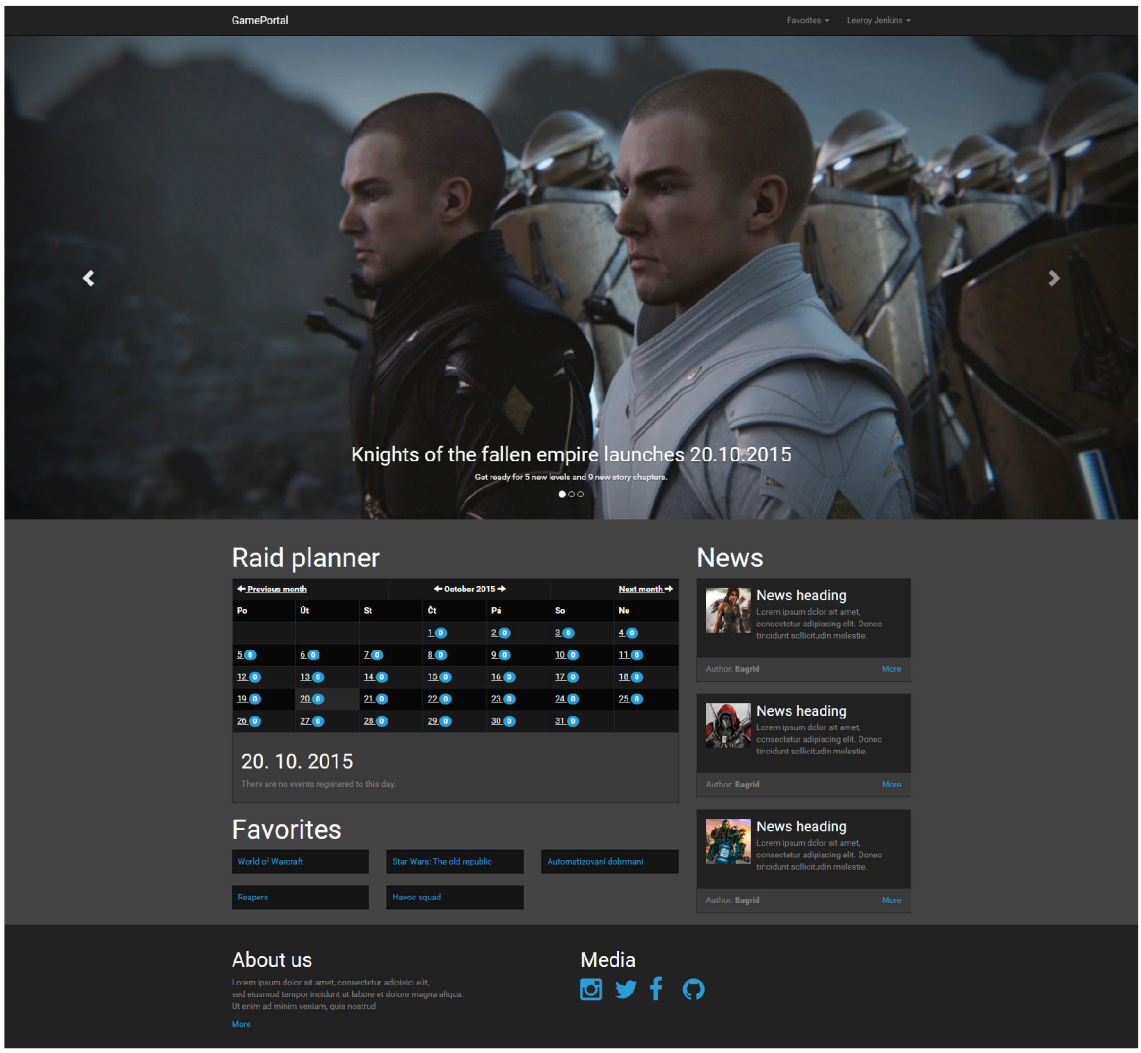


Obr. A.8: ERD diagram datového modelu lokalizace.

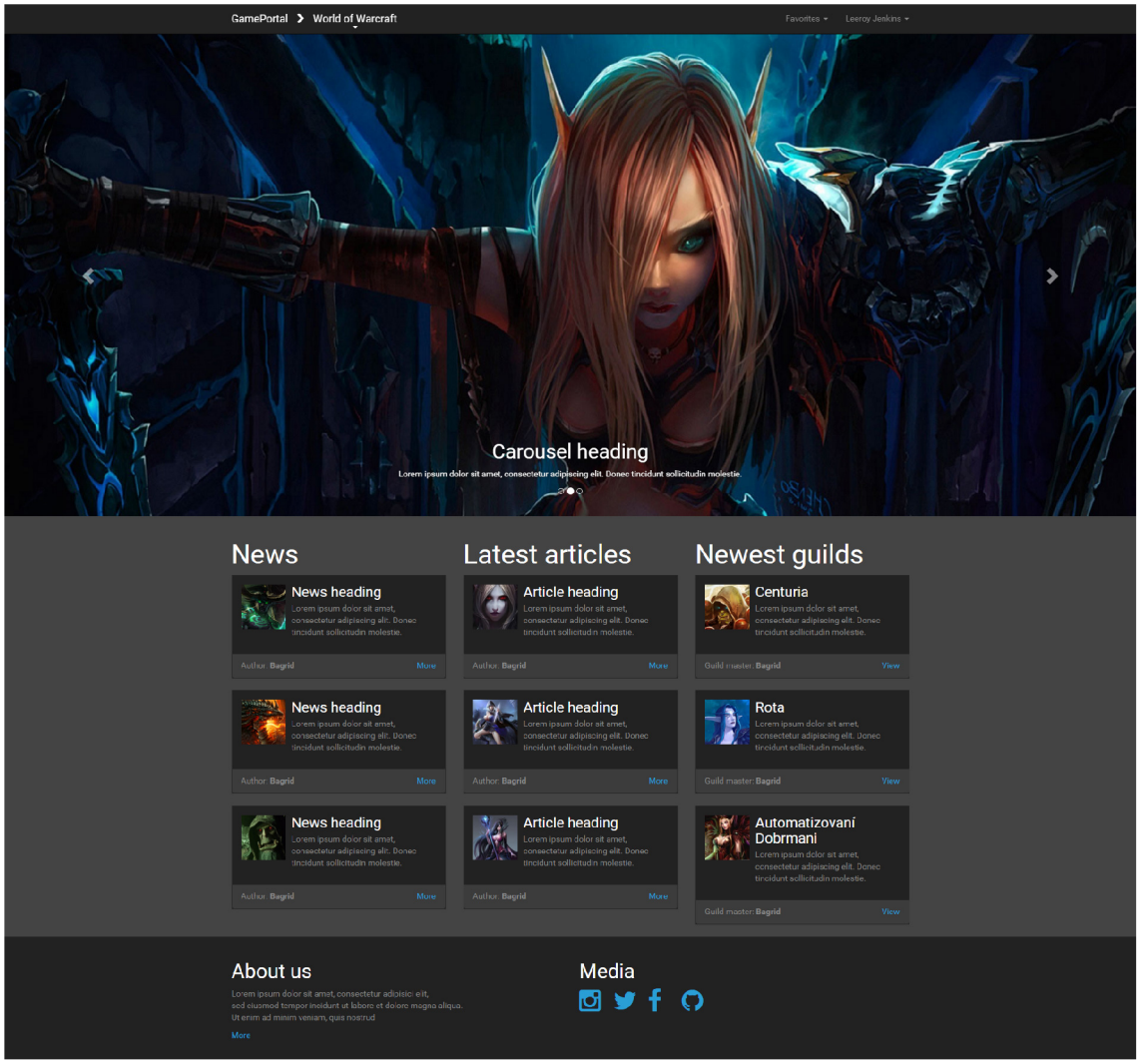


Obr. A.9: ERD diagram systému DKP.

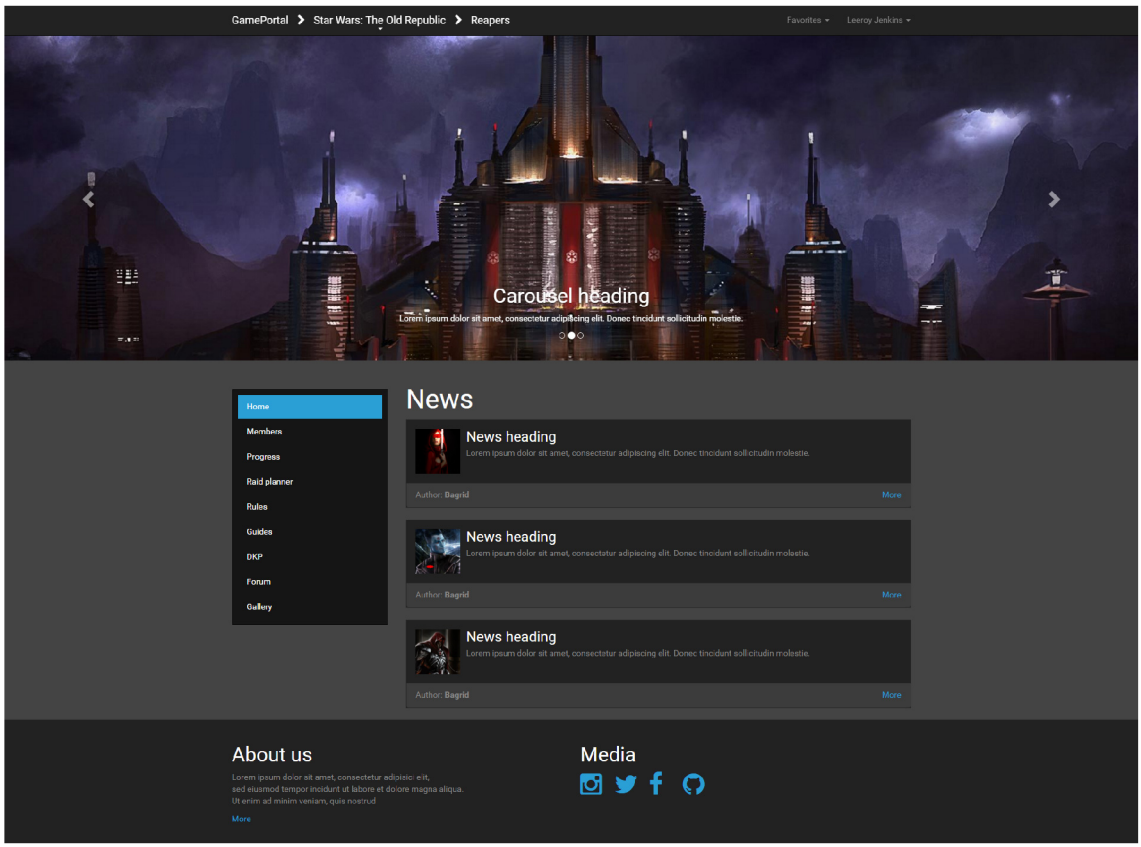
B SNÍMKY OBRAZOVEK A KOMPONENT



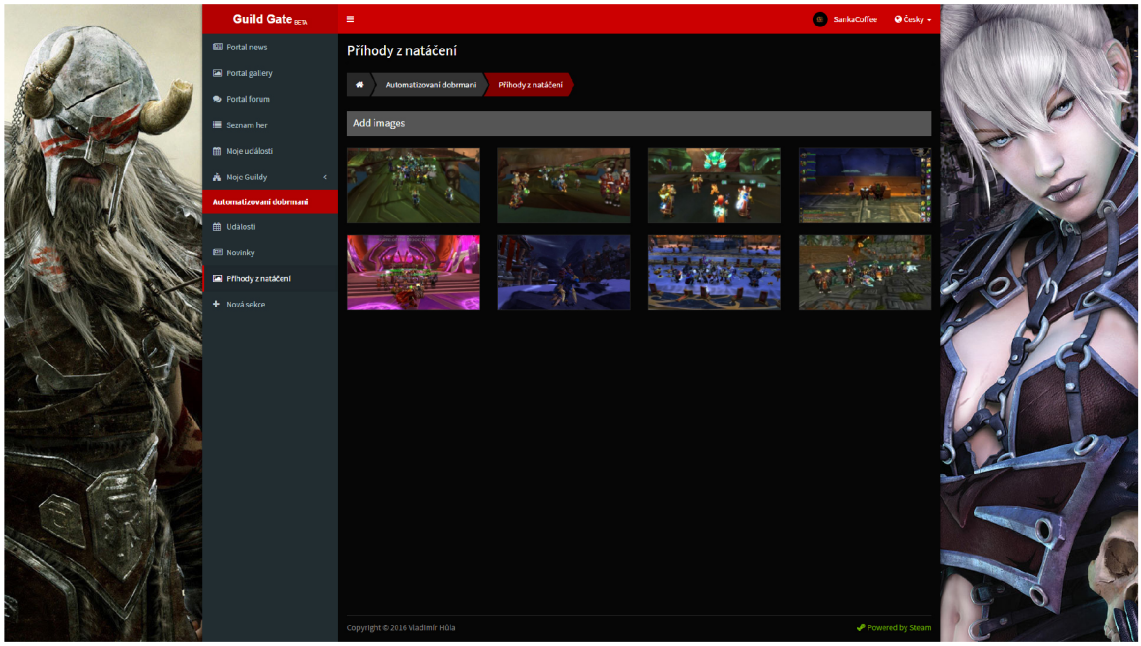
Obr. B.1: Návrh obrazovky přihlášeného uživatele.



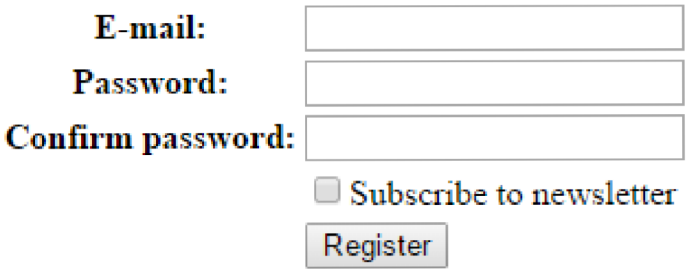
Obr. B.2: Návrh možné podoby herní stránky.



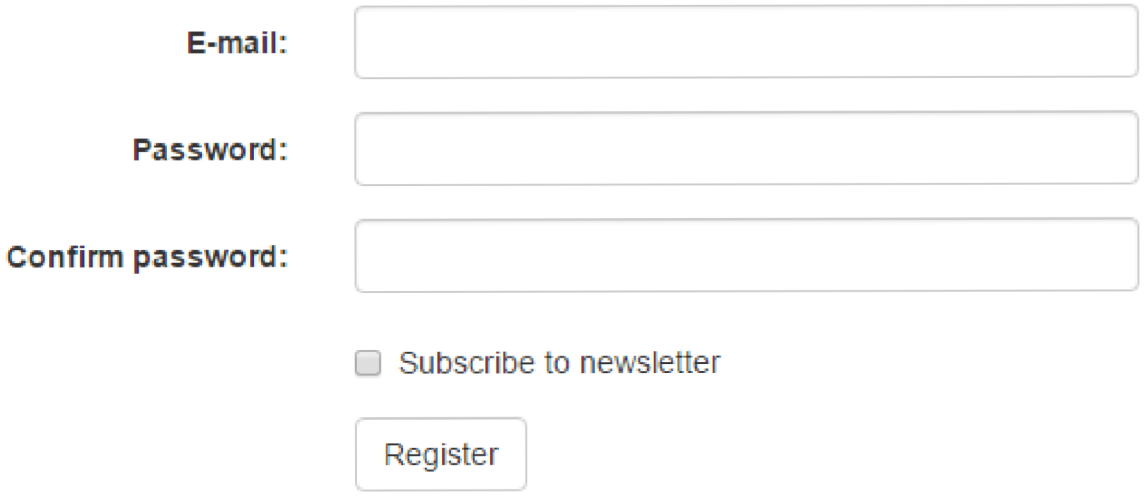
Obr. B.3: Návrh jedné z možných podob guildovní stránky.



Obr. B.4: Současná podoba portálu.



Obr. B.5: Výchozí vykreslení registračního formuláře.



Obr. B.6: Vykreslení registračního formuláře pomocí BootstrapForms.