TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

NÁVRH A REALIZACE ONLINE ŘEŠENÍ DESKOVÉ HRY (HOPE CARDGAME)

Bakalářská práce

Studijní program: B2646 – Informační technologie
Studijní obor: Informační technologie
Autor práce: **Jan Procházka**
Vedoucí práce: Ing. Pavel Tyl

Liberec 2017

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

DESIGN AND IMPLEMENTATION OF BOARD GAME INTO ONLINE SOLUTION (HOPE CARDGAME)

Bachelor thesis

Study programme: B2646 – Information Technology
Study branch: Information Technology
Author: **Jan Procházka**
Supervisor: Ing. Pavel Tyl

Liberec 2017

Technická univerzita v Liberci
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: 2016/2017

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jan Procházka**
Osobní číslo:	**M14000067**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Návrh a realizace online řešení deskové hry (HOPE Cardgame)**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Seznamte se s metodikou vývoje, potřebnými technologiemi a modelováním webových aplikací v PHP.
2. Seznamte se s projektem HOPE Cardgame online a popište ho.
3. Realizujte následné části online verze projektu HOPE Cardgame:
   a) animaci herních karet pomocí jQuery,
   b) animace pomocí CSS3 pro menší herní efekty během hry,
   c) nastylování herního prostředí v HTML5 a CSS3 (kompatibilní pro hlavní prohlížeče dle grafického návrhu),
   d) garbage collector - odstraňování duplicitních položek či položek, které již nebudou dále během hry používány,
   e) preloader - ukazatel stavu načítání potřebných prvků pro hraní,
   f) optimalizaci SQL dotazů a PHP kódu.
4. Zhodnoťte realizaci a další možnosti rozvoje aplikace.

Rozsah grafických prací: **dle potřeby dokumentace**

Rozsah pracovní zprávy: **30–40 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] **Learn php 7: object oriented modular programming using HTML5, CSS3, Javascript, XML, JSON, and MYSQL. New York, NY: Springer Science+Business Media, 2015. ISBN 9781484217290.**

[2] **FOWLER, Martin. Destilované UML. 1. vyd. Praha: Grada, 2009. Knihovna programátora (Grada). ISBN 978-80-247-2062-3.**

[3] **CHAFFER, Jonathan a Karl SWEDBERG. Mistrovství v jQuery: [kompletní průvodce vývojáře]. 1. vyd. Brno: Computer Press, 2013. Mistrovství. ISBN 978-80-251-4103-8.**

Vedoucí bakalářské práce: **Ing. Pavel Tyl**
Ústav mechatroniky a technické informatiky

Konzultant bakalářské práce: **Ing. Přemysl Svoboda**
Ústav mechatroniky a technické informatiky

Datum zadání bakalářské práce: **10. října 2016**

Termín odevzdání bakalářské práce: **15. května 2017**

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

L.S.

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2016

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem. Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 15.5 2017

Podpis: Bradáčka

Poděkování

Tímto bych chtěl poděkovat všem, kteří mi pomohli s mojí závěrečnou prací a bez kterých by tato práce nemohla vzniknout. Zejména bych chtěl poděkovat své rodině a přítelkyni za morální a hmotnou podporu v průběhu studia. Dále panu Ing. Pavlu Tylovi za jeho odborné vedení a spolupráci. V neposlední řadě bych chtěl poděkovat firmě HOPE Studio s.r.o. za možnost účastnit se projektu HOPE Cardgame Online.

Abstrakt

Cílem této bakalářské práce je provést analýzu projektu HOPE Cardgame od společnosti HOPE Studio s.r.o. a implementovat dílčí části projektu. Jedná se o rozložení herních prvků, promazávání nepotřebných grafických souborů, validace herních balíčků, preloader používané grafiky, animace pomocí JQuery.

Pro zjištění optimálního rozložení herních prvků jsou použity zkušenosti z hraní deskové hry HOPE Cardgame a zkušenosti získané z hraní online her, které jsou založeny na stejném principu. Promazávání souborů je potom řešeno rekurzivním algoritmem v jazyce PHP. Validaci herních balíčků provádí skript v PHP na straně serveru a JQuery na straně klienta. Všechny dílčí části byly úspěšně naimplementovány a nyní jsou zakomponovány v projektu.

Klíčová slova

HOPE Cardgame, projektová analýza, rozložení herních prvků, php skript, animace JQuery

Abstract

The goal of this bachelor thesis is to analyze the HOPE Cardgame project by HOPE Studio s.r.o. and implement parts of the project. These are game objects layout, greasing of unused graphics files, game pack validation, preloader of used graphics and animation using JQuery.

To determine the optimal layout of game elements, the experience of playing board game HOPE Cardgame and the experience gained from playing online games based on the same principle are used. File shuffling is then resolved by a recursive PHP algorithm. Validation of game packages is done by the server-side PHP script and JQuery on the client side. All sub-parts have been successfully implemented and are now incorporated into the project.

Keywords

HOPE Cardgame, project analysis, layout of game elements, php script, JQuery animation

Obsah

Seznam zkratek

CSS – Cascading Style Sheets

DOM – Document Object Model

HTML – HyperText Markup Language

IIS – Internet Information Services

ISO – International Organization for Standardization

MDA – Model-driven architecture

MVC – Model View Controller – název softwarové architektury

PHP – PHP: Hypertext Preprocessor, název programovacího jazyka

UML – Unified model language

W3C – World Wide Web Consortium

WWW – World Wide Web

XML – eXtensible Markup Language

1 Úvod

V současné době komplexní digitalizace je obvyklé převádět do digitalizované podoby knihy a ostatní dokumenty. Výjimkou však už nejsou ani deskové a karetní hry. Je tomu tak, protože v digitalizované podobě lze postihnout větší počet populace, a tím pádem i větší počet hráčů. Odpadá tedy nutnost chodit do heren deskových her, nebo si pořizovat vlastní krabici s hrou. Naopak otevírá se možnost zahrát si stejnou hru z pohodlného prostředí domova s lidmi z celého světa.

Cílem této práce je importování deskové a karetní hry HOPE Cardgame do webového rozhraní a to za pomoci konvenčních programovacích, analytických a kooperačních nástrojů. Jedná se tedy o komplexní souhrn technologií a nástrojů od konceptuální resp. analytické oblasti po oblast vývojářskou. Každá tato oblast má svá specifika. Je proto velice důležité, aby tyto segmenty tvořili plně spojitý a ucelený prvek. Tomu musí předcházet náležitě provedená analýza, která se následně transformuje na konceptuální úroveň. Odtud je pak již jen dílčí posun na samotnou fyzickou realizaci projektu s přihlédnutím na znuvupoužitelnost a bezpečnost aplikace. Samozřejmě je důležité tento vývojový model podpořit důsledným a procesně orientovaným projektovým managementem, který cíl této práce pomohl docílit. V neposlední řadě bylo také nutné vybrat vhodné CASE nástroje, které tento model silně a vhodně podpořily a ušetřily tak mnoho času s rutinními procesy. Tento ušetřený čas pak mohl být následně investován do algoritmizace a dalších náročných oblastí v projektu.

2 Metodiky vývoje softwaru

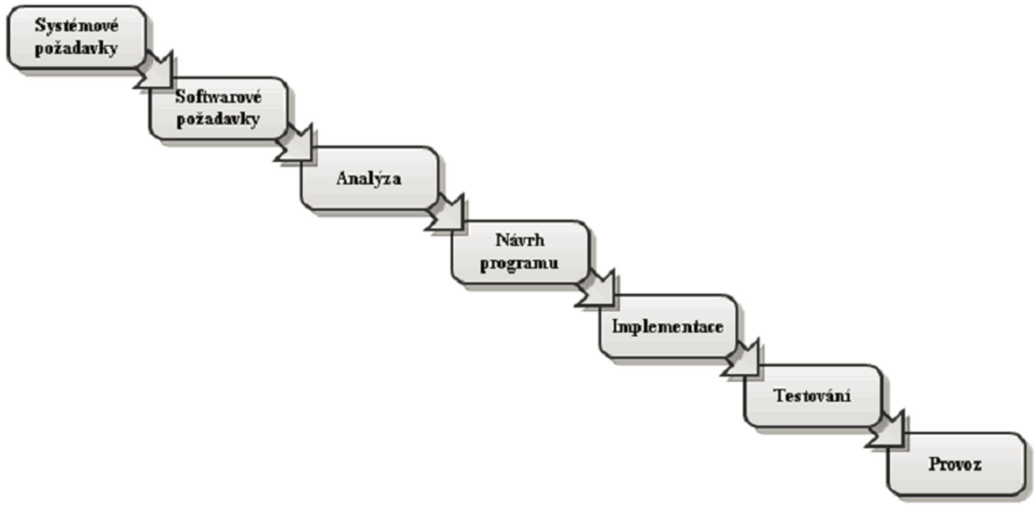
Pod pojmem metodika vývoje softwaru (MDA) je obecně označován soubor pravidel, postupů a nástrojů určených pro návrh, plánování a řízení vývoje softwaru. Stejně jako se v oboru informačních technologií vyvíjí neustále nové technologie, tak se i přizpůsobuje metodika vývoje [8, 17].

Každá metodika (model) obsahuje hlavní fáze tj. analýzu, návrh a implementaci. Jednotlivé metodiky se potom liší především rozsahem a pořadím jednotlivých fází. Rozdělují se do dvou skupin – tradiční a agilní. Agilní metody pracují na neustálé komunikaci mezi vývojářským týmem a klientem, a proto mohou snadněji upravovat zadání v průběhu vývoje.

Mezi nejzákladnější patří vodopádový, prototypový a spirálový model a mezi zástupce agilních modelů patří extrémní programování [8, 17].

2.1 Vodopádový model

Jedná se o sekvenční přístup vývoje projektu, tzn. nelze zahájit další fázi bez uzavření současné fáze (*Obrázek 1*). Tento vývojový model je tedy postavený na neustálém důkladném testování jednotlivých částí. Výhodou je úspora času a zmenšení možnosti výskytu chyby, pokud jsou kvalitně provedeny, protestovány a ukončeny předešlé fáze. V opačném případě dojde ke značnému zpomalení a prodražení projektu [25, 36].



Obrázek 1: Vodopádový model [36]

Hlavním problémem tohoto modelu je, že ve větších projektech je prakticky nepoužitelný, protože obvykle nejde definitivně uzavřít předešlou fázi. Další nevýhodou je neschopnost reagovat na změny v zadání projektu. Hlavně kvůli těmto problémům se vodopádový model téměř nevyužívá [25, 36].

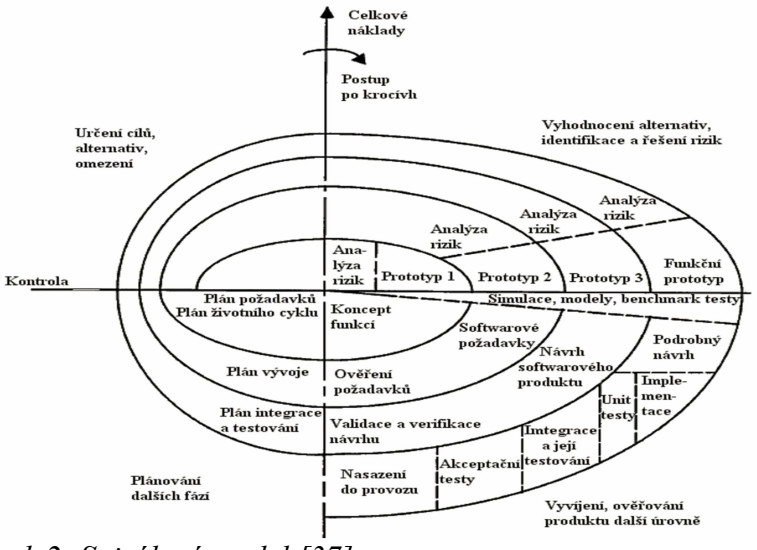
2.2 Prototypový model

Prototypový přístup je takový přístup, kde dochází k vyvíjení neúplného projektu, tzv. prototypu. Jelikož se neustále vyvíjí pouze prototyp, není tento model samostatným vývojovým modelem, ale je vhodné ho kombinovat s ostatními vývojovými modely. Celý projekt je rozdělen na menší funkční části, které se vyvíjejí odděleně, což vede ke snížení rizik při vývoji. Pro zvýšení využitelnosti celého projektu, je po celou dobu vývoje přítomen uživatel budoucího softwaru [22].

2.3 Spirálový model

Tento model je založen na interaktivním přístupu a opakované analýze. Dochází tedy ke střídání analýzy, vývoje, hodnocení a plánování dalšího postupu, jak je vidět na *Obrázek 2*. Díky tomu je schopen lepé reagovat na změny v zadání. Pokrývá tedy hlavní nedostatky vodopádového modelu [15, 37].

Tento model je závislý na provádění důkladné analýze rizik, aby nedošlo k přehlédnutí rizikových prvků, a proto je na ní v průběhu vývoje kladen větší důraz a je jí věnováno více času a prostředků. Vzhledem k tomu, že při vývoji je volnější na dodržování stanovených



Obrázek 2: Spirálový model [37]

pravidel a podmínek, bývá problém s dodržením smluvních podmínek a celkovou cenou vývoje projektu [15, 37].

2.4 Iterační model

Celý vývoj je rozdělen do iterací (period). Obvyklá délka iterace bývá v týdnech, někdy i v měsících. V případě extrémního programování je délka periody mnohem kratší (dny / hodiny). S každou takovouto periodou vzniká plně funkční program. Ten se liší od minulé funkčnosti pouze rozsahem sady funkcí [23].

Výhodou takovéhoto přístupu je naprostá kontrola nad projektem. S každým ukončením iterace je program plně protestován, zdokumentován a nové funkcionality jsou plně přijaty zákazníkem. Díky tomu je neustále připravená funkční verze k produkci. Tento vývojový model dokáže také velmi snadno reagovat na změny v požadavcích nebo technologiích. Změny jsou jednoduše zahrnuty do další iterace [23].

2.5 Inkrementální model

Jedná se o základní agilní přístup k vývoji softwaru. Tento přístup neřeší zadání jako celek, ale řeší jednotlivé části, které potom celek tvoří. Obvykle první verze bývá funkční jádro softwaru s nejzákladnější funkcionalitou. Ta je otestována zákazníkem a teprve potom dochází k dalšímu vývoji. Další komponenty se poté přidávají až poté, co jsou kompletně protestovány a zdokumentovány. Jsou pak označovány jako inkrementy nebo přírůstky. Proto tento model bývá označován občas jako přírůstkový model [24].

Výhodou tohoto přístupu je snadná, rychlá a levná reakce na změnu požadavků od klienta. Naopak občas tento postup mívá problémy s vývojem softwaru pro firmy, kde se musí striktně konkrétní postup vývoje [24].

2.6 Extrémní programování

Extrémní programování je agilní metoda vývoje softwaru. Vzniklo na počátku 90. let v USA. Principem extrémního programování jsou tradiční činnosti, které jsou ovšem dovedeny do extrému. Neustále se kód reviduje. U počítače programují 2 lidé, aby se zabránilo profesionální slepotě [6, 7].

Kód se udržuje neustále na nejnižší úrovni, tzn., nikdy se neprogramuje nic navíc. Neprogramuje se nic, co není doopravdy potřeba. Když dojde ke změně požadavků,

jsou potom minimální ztráty na provedené práci. Aby nedocházelo ke zbytečné práci, tak se kód neustále testuje, jak ze strany vývojářů, tak i ze strany klienta. Obvykle je testovací kód mnohonásobně delší než samotný software. Úpravy v návrhu kódu a architektury se dělají pořád a může je dělat kdokoli z týmu, což zaručuje objektivní přístup k projektu [6, 7].

3 Unified Modeling Language

UML nabízí standardizovaný způsob zápisu návrhů systému včetně konceptuálních prvků, jako jsou procesy a systémové funkce nebo také konkrétních prvků, například příkazy programovacího jazyků nebo databázových schémat [18].

První verzí tohoto jazyka byl standard UML ve verzi 0.9, který vznikl v roce 1994. Zatím poslední vydanou UML specifikací je verze 2.4.1, která byla i přijata v roce 2012 jako standard Mezinárodní organizací pro normalizaci (ISO) pod kódovým označením ISO/IEC 19505 [18].

3.1 Metody využívání UML

Jedním z nejpoužívanějších způsobů, jak využívat UML, je navrhování konceptů. UML se stává podpůrným nástrojem pro komunikaci mezi vývojáři a pro zaznamenání myšlenek a návrhů. Do diagramů se kreslí pouze věci podstatné pro grafické vyjádření návrhu a jeho části před zahájením implementace. Důležitá je srozumitelnost, rychlost nakreslení a snadnost změny či navržení alternativ řešení [19, 20].

Obdobou je kreslení detailních návrhů, do kterých je cílem zakreslit kompletní návrh. Takovýto diagram by měl obsahovat všechny informace nutné k naprogramování tak, aby programátor nemusel již vůbec přemýšlet nad strukturou, nebo vztahy v diagramu [19, 20].

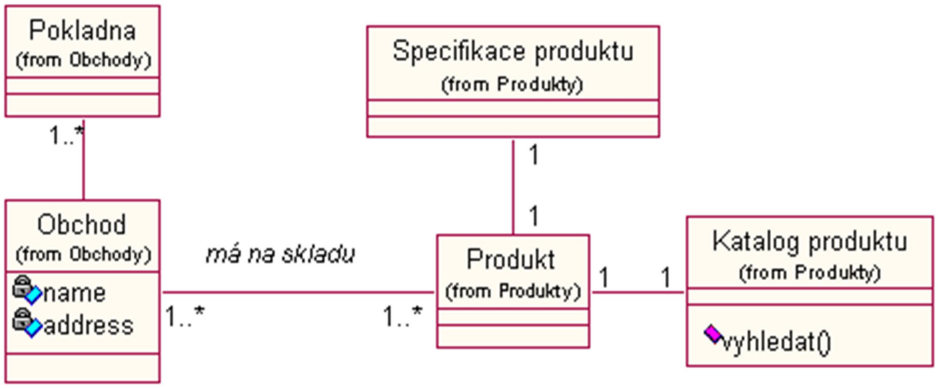
Dalším způsobem použití UML je jako programovací jazyk. Existují dnes již vývojová prostředí, která dokáží z diagramu vygenerovat spustitelný kód. Mezi takovéto nástroje patří ArgoUML nebo Astah [19, 20].

Autoři UML a autoři CASE nástrojů se nedívají na UML jako na diagramy, pro ně je základem UML metamodel, který se znázorňuje pomocí diagramů. Při tomto přístupu se často používá pojem model místo pojmu diagram. Metamodel se popisuje pomocí Meta-Object-Facility (MOF) - abstraktního jazyka pro specifikaci, vytváření a správu metamodelů. Tento pseudo jazyk tvoří další standard OMG. Pro výměnu metamodelů se používá XMI - na XML založený standard, který je součástí standardu UML [19, 20].

Přesto jsou však nejznámější a nejvíce používanou součástí standardu diagramy. Dělíme je do tří skupin a to podle struktury, chování a interakce. Mezi často používané diagramy patří diagram tříd, stavový a sekvenční diagram [19, 20].

3.2 Diagram tříd

Diagram tříd popisuje statickou strukturu systému, která znázorňuje datové struktury a objekty, jejich operace a souvislosti. Diagram tříd znázorňuje datový model systému od konceptuální úrovně až po implementaci. Datové struktury zařazuje do tříd a zobrazuje vztahy těchto tříd. Z tohoto důvodu se řadí do skupiny strukturních diagramů [26].



Obrázek 3: Diagram tříd [42]

Základním objektem diagramu (*Obrázek 3*) jsou třídy. Třídy musí obsahovat název, seznam vlastností třídy (atributy) a seznam metod a operací, které může třída vykonávat. U všech těchto vlastností a metod jsou symboly, které vyjadřují zapouzdření (přístupnost) třídy. Zapouzdření může nabývat hodnot Public, Private, Protected, ale i Derived a Package [26].

Dalším součástí diagramu tříd jsou samozřejmě vztahy, které propojují jednotlivé třídy v diagramu. Každý vztah určuje směr a počet instancí třídy, které mohou ve vztahu figurovat. Mezi základní vztahy patří Závislost (Dependency), Asociace (Association), Agregace (Aggregation) a Kompozice (Composition) [26].

Velmi důležitou vlastností třídy je dědičnost. Dědičnost je hierarchický vztah mezi třídami. Jeden ze dvou příbuzných tříd (podtříd), je považován za specializovaný druh druhého (super typu). V praxi to znamená, že každý případ potomka je také instancí rodičovské třídy. V UML jazyce je grafický symbol pro dědičnost trojúhelník [26].

3.3 Stavový diagram

Stavový diagram (*Obrázek 4*) je v UML způsob grafického zápisu vývoje systému, který má konečný počet stavů. Takovým systémem může být stavový automat či další podobné systémy, které vyjadřují stavy určitého objektu a přechody mezi nimi. Diagram poskytuje

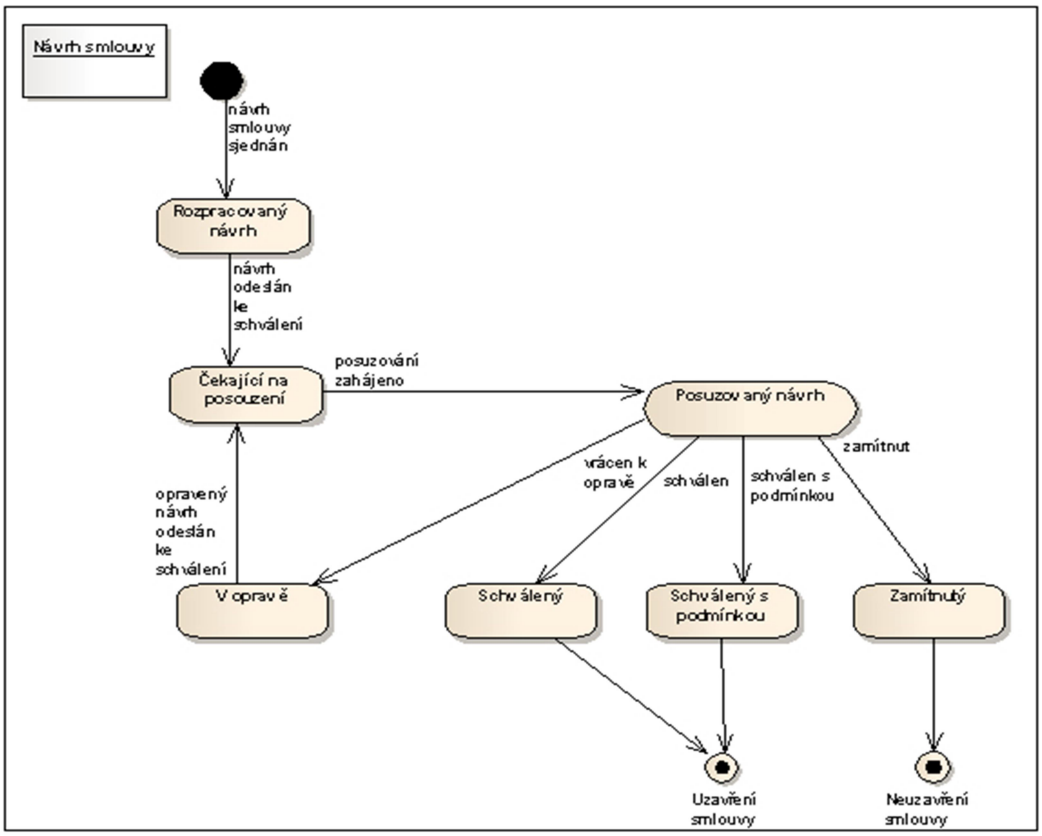
sadu elementů pro popis chování automatu, který je vyjádřen průchodem stavy a je řízen vnějším vstupem [16].

Podle specifikace modelovacího jazyka UML jsou stavové automaty důležitou pomůckou př modelování dynamického chování systému. V průběhu průchodu stavy mohou být vykonávány různé aktivity. Diagramy stavových automatů podle této definice jsou tvořeny třemi základními prvky – stav, událost, přechod [16].

Charakteristika elementů, pro které má reprezentace stavovým automatem smysl:

- element reaguje na vnější události,
- životní cyklus elementu může být vyjádřen jako řada stavů, přechodů mezi nimi a událostí,
- chování elementu je důsledkem jeho předchozího chování.

V objektově orientovaném modelování mohou stavové automaty být použity k modelování dynamického chování reaktivních objektů, jako jsou třídy, případy užití, podsystémy nebo celé systémy [16].

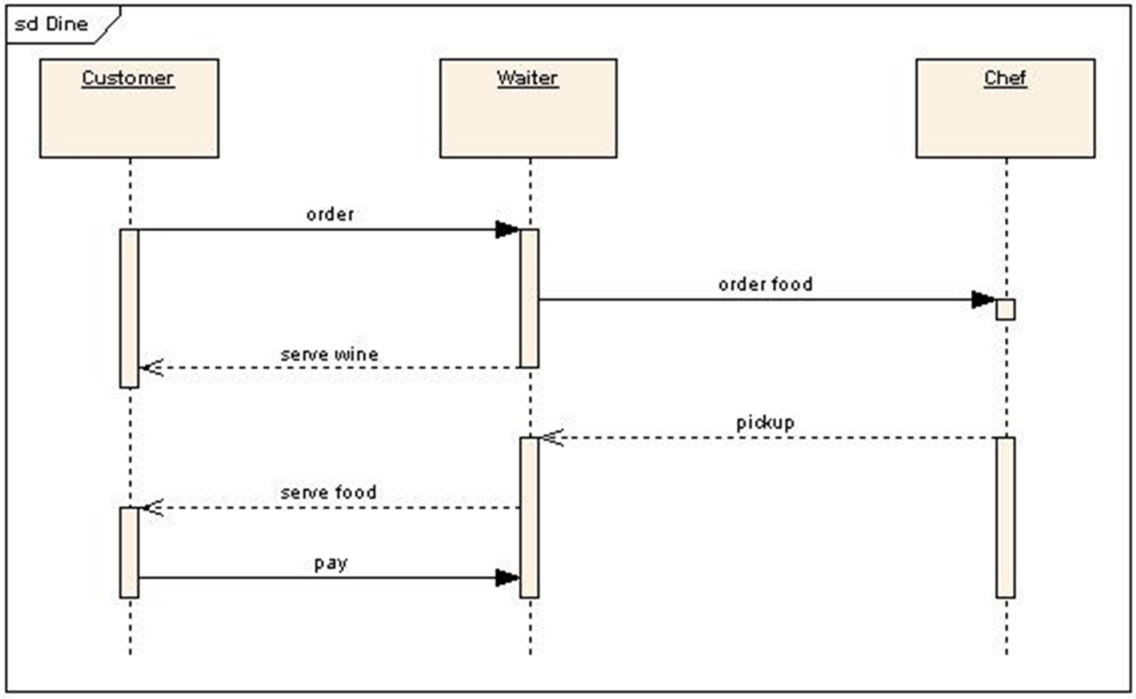


Obrázek 4: Stavový diagram [40]

3.4 Sekvenční diagram

Sekvenční diagram je jeden z diagramů interakcí jazyka UML. Zachycuje časově uspořádanou posloupnost zasílání zpráv mezi objekty. Sekvenční diagram nejčastěji znázorňuje spolupráci několika vzorových objektů v rámci jednoho případu užití [14].

Jednotlivé procesy či objekty zapojené do popisovaného případu užití jsou umístěny v horní části diagramu. Od nich pak vedou směrem dolů čáry (tzv. lifelines), které znázorňují běh času. Mezi čarami jsou pak zakresleny vodorovné šipky různých typů, které reprezentují zprávy posílané mezi objekty. Plné šipky značí volání, přerušované pak odpověď. Podlouhlé obdélníky na svislých čarách vyznačují dobu zpracovávání dané zprávy či čekání na odpověď (*Obrázek 5*)[14].



Obrázek 5: Sekvenční diagram [39]

4 Technologie webových aplikací

Základním kamenem webových technologií je jazyk HTML. Tento jazyk vyvinul Tim Bernerse-Lee v roce 1990. Zároveň HTML vyvinul i protokol HTTP a první WWW prohlížeč. HTML obsahuje značky (tagy), které mají specifické atributy. Značky a jejich atributy jsou definovány ve specifikaci HTML. V současné době (2016) je poslední vydanou verzí HTML 5.0 [27, 28].

Jelikož HTML je zobrazováno jako prostý text, který není úplně vhodný pro předávání informací, je doplněno o grafickou podobu. Ta je definována v CSS od konsorcia W3C.

Pro příjemnější interakci uživatele a HTML stránek se používá interpretovací jazyk JavaScript. Tento jazyk byl standardizován asociací ISO v roce 1998. Na rozdíl od ostatních interpretovacích jazyků běží JavaScript na straně klienta, a to po úplném stažení HTML stránek [27, 28].

Původně měl každý prohlížeč vlastní přístup ke zpracování HTML, většinou pomocí javaScriptu. Kvůli této nekompatibilitě byl zaveden DOM od konsorcia W3C. Tento přístup umožňuje pracovat s HTML obsahem jako se stromem. Definovali 5 úrovní kompatibility, které obsahují povinné a volitelné části. Aby HTML stránky mohly tvrdit, že splňují určitou úroveň DOM, musí obsahovat všechny jeho povinné části a musí splňovat předešlé úrovně. Od roku 1997 vydává W3C také specifikace HTML. Kombinace pouze HTML a CSS se časem ukázala jako nedostačující a byl vymyšlen jazyk PHP [27, 28].

4.1 PHP

Jazyk PHP je programovací jazyk, který slouží k programování dynamických stránek XML nebo HMTL. PHP bylo postupně vyvíjeno od roku 1994 a v roce 1995 vyšla první verze PHP, verze 1.0. Tuto prvotní verzi napsal programátor Rasmus Lerdorf. Jazyk se vyvíjí až do dnes. V současné době je nejpoužívanější podporovaná verze 5.3 (5.4). Nejnovější verzí je však verze 7.0. Verze 6.0 nesplňovala všechny požadavky, a proto nebyla nikdy zveřejněna [10, 21].

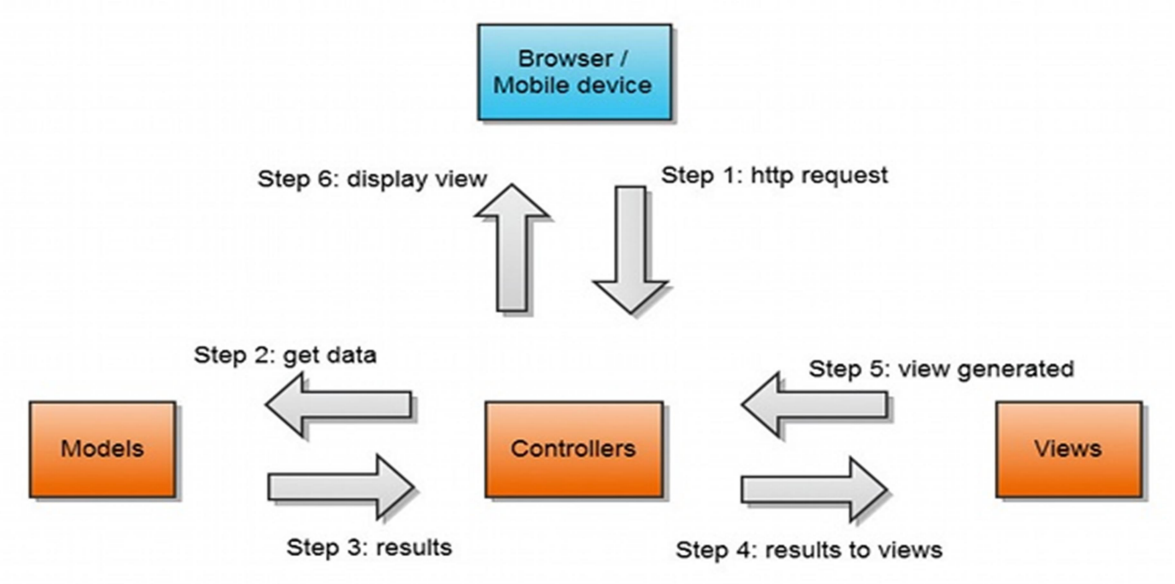
Základním principem tohoto jazyka je komunikace klient-server. Klient pošle dotaz na server, ten složí webovou stránku podle požadavků a kritérií a pošle ji zpět klientu. Ten potom dostane již zcela složenou stránku. Tím dochází pouze k minimální zátěži prohlížeče na straně klienta. Podrobněji o tomto principu v kapitole MVC [10, 21].

PHP se stalo velmi populární kvůli své jednoduchosti. Již v základní knihovně je implementováno mnoho funkcí. Jednou z výhod tohoto programovacího jazyka taky není typově orientovaný, tj. že typ proměnné není striktně definován. Typ je určen typem hodnoty v proměnné. Zásadním problémem tohoto programovacího jazyka byla absence specifikace až do roku 2014. Do té doby bylo PHP definováno pouze svou implementací. Vznikla tím nejednotnost pojmenování funkcí a nejednotnost pořadí vstupních parametrů do funkce [10, 21].

4.2 MVC

MVC je druh softwarové architektury, někdy také označované jako návrhový vzor. Nejčastěji je spojována s jazykem PHP, přestože nic nebrání použít tuto architekturu v jiných jazycích nebo situacích [4, 10, 11].

Principem této architektury je oddělení datové, grafické a řídící části aplikace, anglicky označované jako Model, View a Controller. Odsud také pochází běžně používaná zkratka MVC, která se sama stala názvem této architektury [4, 10, 11].



Obrázek 6: MVC struktura [38]

Každá z těchto tří částí má tedy své specifické vlastnosti a účel. Jako první je tedy Model. Model zpracovává datovou část aplikace. Je tedy tou částí, která má na starosti veškeré operace nad daty jako jsou výpočty, zpracování dat ze souborů a z databází apod. Komunikaci s databází, či samotné zpracování souborů by již zase příslušný model provádět neměl. Měl by místo toho používat kontejner k tomu určený. Dále by neměl obsahovat žádné

informace o grafické podobě stránky a ani žádnou komunikaci s klientem. Jednoduše lze říci, že Model dostane za úkol zpracovat určitá data a posléze předá zpracovaná data dále [4, 10, 11].

Další nedílnou součásti MVC je View, česky pohled. Jedná se o grafickou šablonu stránky. View vezme data, která získal z modelu a vloží je do správné grafické šablony. Tato šablona je obvykle v souboru typu phtml, který obsahuje HTML stránku a fragmenty PHP kódu, který umožňuje dynamicky měnit obsah. Takto složená celá stránka je předána Controlleru. Zase platí, že View již s daty jako takovými nijak nepracuje a pouze je vkládá do šablony [4, 10, 11].

Poslední nedílnou součástí architektury je samozřejmě Controller – řadič. Controller je ta část, která má na starosti komunikaci s uživatelem. Controller přijme dotaz od uživatele. Ten zpracuje, vyhodnotí a zavolá správný Model. Po získání dat z View zase odešle tyto data zpátky klientovi. Celé schéma postupu, od požadavku až po celé složení stránky, je na *Obrázek 6* [4, 10, 11].

Hlavním důvodem k rozdělení aplikace do těchto tří částí je odlišný způsob jakým pracují. S tím je spojena i odlišná algoritmizace a zpracování ze strany programátorů. Jinak se přistupuje ke zpracování dat a jinak k vytváření grafické podoby stránky. Takto rozdělenou aplikaci tedy může programovat více programátorů najednou bez komplexních znalostí celé aplikace, protože jednotlivé části jsou na sobě zcela nezávislé [4, 10, 11].

Na druhou stranu tato architektura má poté mnohem složitější adresářovou strukturu. Je složitější na organizaci souborů a na orientaci. Často bývá velmi obtížné rozlišit, co patří do Controlleru a co již patří do Modelu. Navíc to vyžaduje odlišný způsob přemýšlení oproti klasickým statickým webovým stránkám [4, 10, 11].

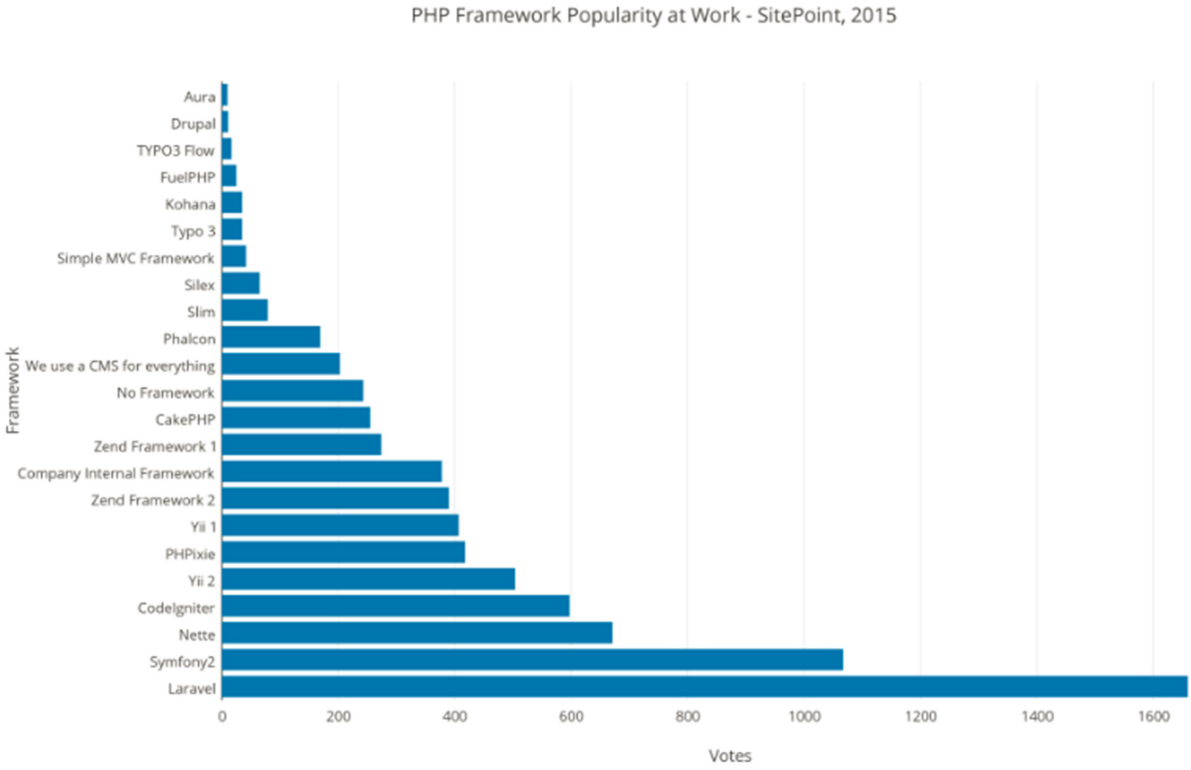
4.3 Framework

Problematiku komplexnosti projektu lze vyřešit použitím frameworku. Framework je úplný nástroj pro vytvoření MVC aplikace. Obvykle obsahuje již vytvořenou adresářovou strukturu a sady funkcí ulehčující práci. Přestože se jedná o PHP mají do určité míry odlišnou syntaxi, protože, jak bylo řečeno v kapitole 4.1, není PHP jednotné v pojmenovávání funkcí a pořadí parametrů. Díky frameworku jsme schopni docílit kratšího, jednoduššího a přehlednějšího

kódu v kratším čase. Taky jsme schopni jednodušeji reagovat na změny. Frameworky také obsahují nástroje pro přehlednější ladění aplikace [30].

Další důležitou vlastností frameworku je bezpečnost. Oproti čistému PHP obsahuje již bezpečností prvky a funkce, jako je automatická kontrola vstupů nebo sandboxy (prostor pro běh oddělených procesů, slouží jako bezpečnostní prvek), které se musí jinak dopisovat pokaždé zvlášť, což představuje velké riziko [30].

Oproti tomu jsou ovšem frameworky mnohem složitější než MVC v čistém PHP. Jsou tedy zase o něco složitější na orientaci. Kvůli vyšší komplexnosti obsahují také bezpečnostní a funkční díry. Musíme potom spoléhat na to, že vývojářský tým tyto chyby najde a opraví. Nemáme tedy nad kódem stoprocentní kontrolu [30].



Obrázek 7: Popularita frameworků (2015) [32]

Jak vyplývá používat framework tedy není nutnost, avšak přes tyto nedostatky se staly frameworky velmi populárními a dnes existuje mnoho komerčních i volně dostupných řešení.

Mezi ty nejvýznamnější celosvětově známé PHP frameworky, jak je vidět na *Obrázek 7*, patří Laravel a Symfony, mezi ty české pak Nette [1]. Každý z frameworku má samozřejmě své klady a zápory. Jejich použití je tedy pouze subjektivní rozhodnutí, ovšem větší projekty svou náročností již použití frameworku nepřímo vyžadují [30].

4.3.1 Nette

V České a Slovenské Republice jednoznačně dominuje v komerčním i personálním využívání frameworků Nette[1]. Pár z hlavních důvodů takového úspěchu je jeho jednoduchost, využitelnost, česká podrobná dokumentace, ale především velmi aktivní česká komunita. Za tímto úspěšným projektem stojí vývojář David Grudl, který se mu věnuje od roku 2004 [31].

Mezi jeho vlastnosti patří automatická eliminace bezpečnostních rizik, znovu použitelnost kódu a velmi kvalitní ladící nástroje. Pro vývoj tohoto frameworku je potřeba mít lokální webový server [31].

4.3.2 Laravel

Nejrozšířenější framework na světě je dnes Laravel[2]. Tento framework je relativně mladý, byl publikován v roce 2012. Jeho jádro je postaveno na starším frameworku jménem Symfony. Stejně jako Nette obsahuje navíc část jménem Composer. Composer je nástroj, který dokáže stahovat a instalovat všechny potřebné knihovny přímo do projektu. Composer tedy prohledá projekt, zjistí závislosti na knihovnách a ty připraví k použití. Tím odpadá tedy nutnost vyhledávat a instalovat tyto knihovny ručně. Další výhodou tohoto frameworku je, že nepotřebuje již lokální webový server, protože ten je již součástí tohoto frameworku [33].

4.4 Webový server

Jak jsem zmiňoval v kapitole 4.3, tak PHP frameworky, až na pár výjimek, potřebují ke svému vývoji lokální webový server. Samotné PHP na tom není jinak. Jelikož PHP je jazyk vytvořený pro zpracování požadavku na straně serveru, je pro jeho vývoj webový server potřeba. Nejrozšířenější jsou servery s jádrem Apache [5].

Mezi bezplatně dostupné servery patří Wamp Server[3], případně IIS, který je součástí Windows. Bez lokálního serveru je možno vyvíjet PHP na vzdáleném („klasickém“) serveru. Vývoj je ovšem poté velmi zdlouhavý a únavný, jelikož se musí po každé změně nahrávat soubory na server. Tyto servery bývají doplněny o databázové systémy, v případě jádra Apache pak nejčastěji o MySQL [5].

[1] Dostupné z: https://nette.org/
[2] Dostupné z: https://laravel.com/
[3] Dostupné z: http://www.wampserver.com/en/

Alternativou Apache je webový server Nginx[4]. Tento webový server je zaměřen na zvýšení výkonu a snížení nároků na paměť. Dále dovoluje spravovat zátěže serverů a určovat jim prioritu. Tzn., že klient pošle dotaz na server a ten je předán serveru se správnou prioritou. Tento systém je primárně vyvíjen pro OS LINUX. Dnes však existují i verze na Mac OS, či Windows [5].

4.5 HTML a CSS frameworky

Pro PHP existuje spousta frameworků, které zjednodušují práci. Ani v zobrazování webových stránek tomu není jinak. Hlavní problematikou v HTML je responzivita webových stránek. Pod pojmem responzivita rozumíme korektní zobrazení webových stránek na různých zařízení. Obvykle bývá problém s různými rozlišeními. Tyto problémy pak framework obvykle řeší za nás. Mezi nejznámější frameworky patří samozřejmě Bootstrap[5] a PureCSS[6]. Oba tyto frameworky rozdělí stránku do mřížky a umožňují potom s touto mřížkou pracovat, případně ji upravovat [30, 35].

[4] Dostupné z: https://www.nginx.com/resources/wiki/
[5] Dostupné z: http://getbootstrap.com/
[6] Dostupné z: https://purecss.io/

5 HOPE Cardgame

Obrázek 8: Desková hra HOPE Cardgame [41]

Abychom mohli vybrat správnou technologii a navrhnout dobře fungující algoritmy, musíme se důkladně seznámit s projektem a principy v něm používaným. V tomto případě se jedná o deskovou hru HOPE Cardgame (*Obrázek 8*) od společnosti HOPE Studio [12].

HOPE Cardgame je strategická stolní hra, která kombinuje prvky deskové a karetní hry. Hra je zařazena do post-apokalyptického žánru. Ve zničeném světě tedy bojují o nadvládu čtyři přeživší rasy Mimozemšťané, Mutanti, Roboti a Lidé. Dva hrající hráči, jeden proti jednomu, se potom se zvolenou rasou snaží získat převahu nad oponentem v podobě finančních zdrojů. Finanční zdroje jsou v tomto případě zároveň i životy hráčů. Hra končí ve chvíli, kdy se některý z hráčů dostane na nulu, nebo se některý z hráčů dostane na maximum, čili 30 zdrojů [12].

5.1 Herní prvky

Nejdominantnější částí celé hry je samozřejmě herní plán. Na herním plánu je znázorněna hrací plocha, tedy 5×5 polí, kam mohou hráči hrát své karty, a počítadlo zdrojů. V tomto bodě je vhodné popsat pojem *sousední políčko*, který se níže používá k vysvětlení pravidel. *Sousední políčka* jsou v této hře pouze políčka, která se sebou sousedí přes hranu karty. Karty, které se sebou sousedí rohem, nejsou karty sousední [12].

Další důležitou částí hry, jsou samozřejmě karty. Karty jsou rozděleny do tří typů: jednotky, lokace a události. Jednotky jsou potom dále rozdělovány podle ras, naopak lokace a události

nejsou přiřazeny k jednotlivým rasám. Celkem je pak ve hře 144 karet, ve 4 základních balíčcích po 36 kartách [12].

Hra obsahuje také sadu herních žetonů a krvavé kameny. Herní žetony obsahují 3 žetony pro určování cílů v útoku a 2 žetony hráčů pro signalizaci pozice na počítadle zdrojů. Krvavé kameny jsou vyrobeny ze skla a používají se pro zobrazování ztracených životů jednotek na hracím plánu [12].

5.2 Popis karet

Karty jsou jádrem této hry, proto je nutné popsat jejich vlastnosti a vzhled. Již na první pohled jsou opticky rozlišitelné, jak zobrazuje *Obrázek 9*. Pod číslem:

1 je cena karty, červená signalizuje počet zdrojů, které hráč zaplatí za zahrání karty a zelená signalizuje počet zdrojů, který hráč dostane za zahrání karty, 2 je název karty, 3 označuje typ karty, 4 je symbol - v případě jednotek určuje rasu karty a u událostí označuje, že se jedná o událost ze základní hry (v současné době existuje již jedno rozšíření této hry, to ovšem není součásti tohoto projektu), 5 je hodnota útoku – útok mají pouze jednotky, 6 je hodnota obrany – obranu mají pouze jednotky a lokace, 7 je hodnota životů – životy mají pouze jednotky (pro lepší pochopení dále popsaných pravidel si lze představit, že lokace mají život jeden), 8 je efekt karty, 9 je vzácnost karty a 10 je jméno autora použitého obrázku na kartě [12].

Obrázek 9: Karty [12]

5.3 Efekty karet

Mimo své vlastnosti mají karty také v popisku napsanou nějakou schopnost – efekt (číslo 8 na *Obrázku 9*). Efekty jsou rozděleny na pasivní a aktivní, přičemž každá karta může mít libovolný počet aktivních a pasivních efektů [12].

Pasivní efekty jsou obvykle schopnosti, které upravují jednotce jejich vlastnosti jako je pohyb, útok, cena apod. Aktivní schopnosti naopak umožňují kartě vykonávat specifický efekt, který je popsán slovně. Tento efekt lze zahrát pouze ve specifickou chvíli a ta je definována přímo na kartě před textem efektu – nejčastěji to pak bývá COMBAT [12].

5.4 Průběh hry

Každý hráč disponuje na počátku 15 zdroji, balíčkem pro dobírání karet a balíčkem pro odhazování karet. Z dobíracího balíčku si vezme 6 karet. Dle svého uvážení má možnost vyměnit si libovolný počet karet. V tuto chvíli jsou hráči připraveni zahájit hru [12].

Fáze

Každé kolo je rozděleno do 6 fází. Jsou rozdělena z prostého důvodu, a to proto, že v každé fázi mohou hráči provádět pouze některé úkony. V jednotlivých fázích se potom hráči také střídají. Hráč, který začínal kolo, začíná všechny dílčí fáze jako první. V následujícím kole potom začíná oponent. Zároveň mohou hráči kdykoli reagovat na protihráče pomocí karet typu Událost. Jednotlivé fáze jsou: Main, Move, Explore, Battle, Dominance, Draw [12].

Fáze 1. – Main

V této fázi mohou hráči vykládat jednotky z ruky na hrací desku. Po takovémto zahrání jednotky se hráči ihned odečte cena jednotky ze zdrojů. Hráči mohou vykládat jednotky pouze na volná pole ve své základně, což je první řádek hracího plánu, nebo na volná sousední pole s jejich lokací. Vykládání jednotek není limitováno počtem vyložených karet, nepřímo je tedy limitováno zdroji [12].

Fáze 2. – Move

Hráči mohou pohnout až se třemi jednotkami. Každá jednotka se může pohnout o jedno políčko, pokud není na jednotce napsáno jinak. Jednotky se mohou pohybovat pouze po sousedních políčkách [12].

Fáze 3. – Explore

Hráči mohou v této fázi vykládat z ruky Lokace. Lokace může být vyložena pouze na prázdné políčko sousedící s vlastní jednotkou. Navíc nesmí být Lokace vyložena do své základny, naopak do soupeřovy základny vyložena být může [12].

Fáze 4. – Battle

Ve fázi Battle může každý hráč provést 3 útoky. Útok provede tím, že označí 3 libovolné cíle pomocí výše zmíněných žetonů. Ve chvíli kdy označuje cíle, nemusí být schopen na cílenou jednotku útočit. Tato strategie se volí především pro druhý a třetí cíl útoku, protože se předpokládá, že předešlé útoky vytvoří možnost na následující cíl útočit. V případě, že se tak nestane a útočník útočit nemůže, tak se útok neprovede [12].

Při vyhodnocení útoky se porovnává součet hodnot útoku všech jednotek, které mohou na cíl útočit, proti hodnotě obrany cíle. Každá jednotka může útočit na sousední políčko. Jednotky označeny vlastností RANGED mohou střílet navíc na políčka, která jsou vzdálena o jedno políčko více ve vertikálním a horizontálním směru [12].

Pro vítězství v souboji proti jednotce musí být součet útoku alespoň o jedna větší. V případě že je útok alespoň dvakrát větší než obrana cílené jednotky, je jednotka přečíslena (overhelmed). Tj. prohrála souboj a ztrácí okamžitě všechny své životy. V případě, kdy nedojde k přečíslení, jednotka ztrácí jeden život, pokud není na útočících kartách zvýšené poškození. Pro vítězství v souboje, a tím pádem i zničení, proti lokaci stačí mít útok roven její obraně [12].

Ve chvíli kdy jednotka ztratí všechny své životy, nebo je jakýmkoli jiným způsobem odstraněna útočníkem ze hry, získává útočník zdroje v hodnotě její ceny. Toto pravidlo platí také pro lokace [12].

Fáze 5. – Dominance

V této fázi si hráči rozdělí maximálně deset zdrojů. Zdroje si rozdělují podle Dominance – karetní převahy na herním plánu. V každém sloupci a v každém řádku se vyhodnotí převaha hráčů, hráč který má převahu získá zdroj. Jednotka se počítá za jednu kartu, ovšem Lokace se počítá za karty 2. Vzhledem k tomuto pravidlo hráči spočítají počet karet ve sloupci či řádku. Hráč, který má součet větší, vyhrál dominanci v onom řádku nebo sloupci [12].

Fáze 6. – Draw

V této závěrečné části mohou hráči zahodit nepotřebné karty ze své ruky do odhazovacího balíčku. Následně si potom doberou karty na maximální počet – tj. 6 karet. Hráč, který v této fázi drží v ruce více než 6 karet, musí přebytečné karty odhodit, aby splňoval limit karet v ruce. V případě, že již hráči došli karty v dobíracím balíčku, dohrává pouze s kartami, které má v ruce [12].

Toto jsou základní pravidla deskové hry HOPE Cardgame, která stačí k pochopení principu hry a jejích základních principů. Jelikož se hra neustále vyvíjí, je téměř nemožné popsat v takto krátkém rozsahu všechna pravidla a výjimky, které v této hře jsou [12].

6 Analýza

Analýza projektu slouží k vytvoření projektové dokumentace. Podstatou této dokumentace je vše řádně zdokumentovat a formulovat, aby v pozdějších fázích vývoje nedocházelo k rozsáhlým nebo zbrklým změnám v požadavcích nebo neočekávaným změnám ceny, či času vyhotovení projektu. Proto by se v této dokumentaci mělo nacházet vše potřebné, pro dotažení projektu do úspěšného konce [2, 3].

Tyto dokumenty sepisuje analytik a vychází z dlouhodobé oboustranné komunikace mezi ním a zákazníkem. Analytik definuje všechny dílčí části projektu – obecnou funkčnost, dílčí části, základní grafické rozložení, databázovou strukturu, komunikaci s dalšími programy, role v projektu, analýzu rizik. Na základě této analýzy pak propočítá časovou náročnost a odhadne cenu projektu [2, 3].

6.1 Analýza rizik

Tabulka 1: Kvalitativní × Kvantitativní analýza [2]

Kvantitativní analýza	**Kvalitativní analýza**
– náročnější na výpočet	+ jednodušší na výpočet
+ transparentní	– diskutabilní
– celkově dražší	+ celkově levnější
– náročná na prog. vybavení	+ nenáročná na prog. vybavení
– náročná na lidské zdroje	+ nenáročná na lidské zdroje
– časově velice náročná	+ časově nenáročná
+ lepší kontrola nákladů	– horší kontrola nákladů
+ poměrně přesná	– méně přesná

V současné době se používají 2 postupy pro analýzu rizik. Kvantitativní postup, který vyjadřuje rizika v podobě finančních ztrát. Tento postup je mnohem komplexnější, ale náročnější a dražší. Kvalitativní postup je naopak rychlejší, jednodušší, levnější, ale není tolik přesný. Kompletní porovnání je vidět v *Tabulce 1* [2, 3].

Pro projekt HOPE Cardgame je vhodnější kvalitativní analýza, protože projekt není moc rozvětvený, lze tedy určit jednoduše jeho rizika, a velmi těžko by se odhadovala finanční ztráta. Došlo by tak k zavádějící analýze [2, 3].

Celková míra rizika se dá spočítat ze dvou faktorů – dopad rizika a pravděpodobnost jeho výskytu. Tato míra se potom rozděluje na vážná, střední a nízká rizika. Dopad rizika a pravděpodobnost výskytu se děli do tří skupin, a to podle toho, jak velký mají vliv na projekt a jak velkou mají pravděpodobnost výskytu. Počet skupin není definovaný, vytváří jej hodnotitel [2, 3].

Výpočet rizika potom probíhá podle vzorce ($\text{riziko} = \text{dopad} \times \text{pravděpodobnost}$). Rizika se potom vyhodnocují podle Tabulky 2. Červená jsou vážná rizika, oranžová jsou střední a zelná jsou nízká rizika [2, 3].

Tabulka 2: Matice rizik

	1	2	3
3	3	6	9
2	2	4	6
1	1	2	3

Vzhledem k těmto stupnicím jsem již nyní schopen srozumitelně popsat rizika, která jsou důležitá pro tento projekt. Tato rizika jsou pro nejjednodušší zpracování prezentována v následujících tabulkách – Tabulce 3 a Tabulce 4.

Tabulka 3: Tabulka rizik při vývoji

Riziko	**Dopad**	**Pravděpodobnost**	**Riziko**
Nedostatek financí	3	1-2	3-6
Odchod klíčových zaměstnanců	3	2	6

Tabulka 4: Rizika po spuštění

Riziko	Dopad	Pravděpodobnost	Riziko
Výpadek serveru	3	2	6
Neúmyslná úprava databáze	3	1	3

6.2 Popis dílčích funkcionalit

V první verzi hry pod označením Alpha, bude pouze funkční hra bez administrační vrstvy. Bude se tedy jednat pouze o menu, které zahájí duel po vybrání jedné z ras. Z toho vyplývá, že v této verzi projektu nebudeme potřebovat žádné přihlašovací prvky. Tím pádem nebudeme potřebovat ani žádné zázemí pro přihlášené hráče. Bude se tedy doopravdy jednat pouze o hru samotnou.

Pro přidání základních balíčků do hry bude třeba jejich validátor. Základní balíčky se budou validovat pouze pro jistotu, aby se tím zkontrolovalo, jestli nedošlo k chybě při jejich vytváření. Tuto funkcionalitu bude aplikace využívat i v dalších verzích, kde bude hráčům zpřístupněna možnost definovat si své vlastní herní balíčky.

První část algoritmu bude řídit celou aplikaci. Bude mít na starost spouštění jednotlivých fází ve správném pořadí. Bude pravidelně střídat začínajícího hráče a kontrolovat podmínky pro vítězství nebo prohru.

Dále bude pro každou fázi kola algoritmus, který bude volán řídícím algoritmem, a bude vykonávat úkony pro danou fázi, které jsou definovány výše. Zároveň budou sledovat, zdali protihráč nechce reagovat.

Pro každou fázi bude nastaven časový limit 45 vteřin. Reakční čas protihráče je vždy 3 vteřiny. Pokud protihráč stiskne během odpočtu této doby reakční tlačítko, hra se pozastaví a bude mít 20 vteřin na promyšlení a hraní karty události. Po uplynutí této doby se zase hra pustí a první hráč může pokračovat.

Poslední součástí projektu je zprovoznění a nastavení serverové logiky. Příprava serveru pro systém přihlašování hráčů. Aplikace bude běžet na větším počtům serverů, kvůli rozložení zátěže při připojení většího počtu lidí. Bude zde tedy jeden řídící server (MASTER), na kterém budou udržovány informace o jednotlivých duelech, jádro hry, přihlašování hráčů,

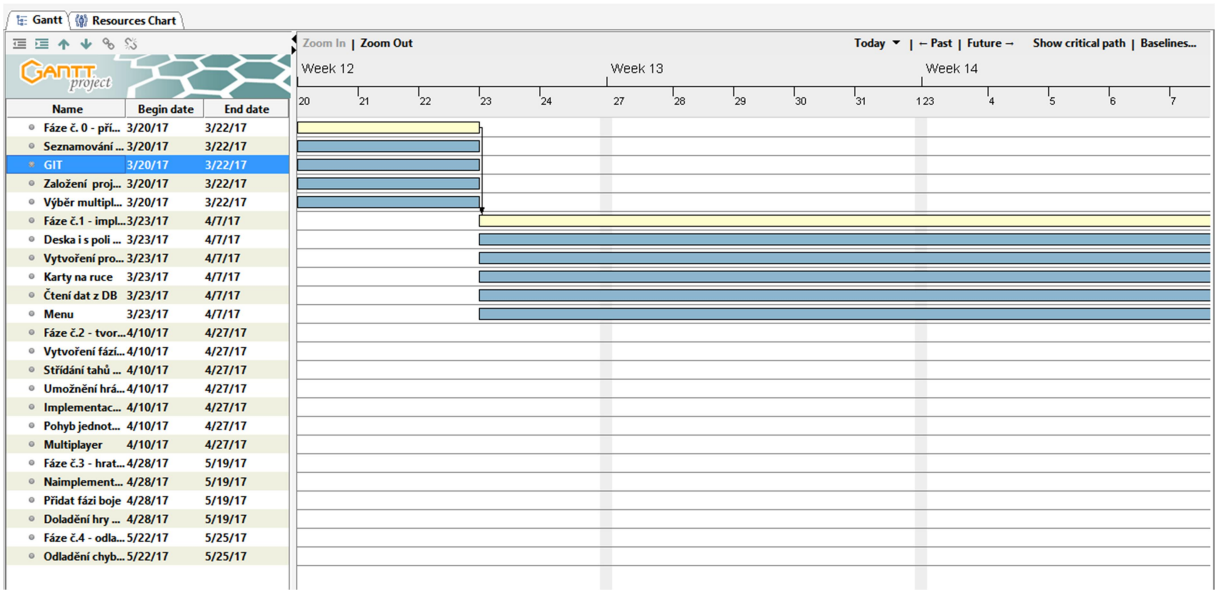
v budoucnu potom také statistiky apod. Pomocí CRONu[7] bude hra rozkopírována na herní servery (SLAVE). Na ně budou hráči přesměrováni po přihlášení z hlavního serveru a na těchto serverech poté poběží jednotlivé duely. Po skončení duelu SLAVE pouze informuje MASTER a výsledku duelu.

6.3 Ganttův diagram

Ganttův diagram se používá pro zobrazení projektu a jeho částí na časové ose. Výhodou tohoto diagramu je, že dnes již je běžně používaný a tedy i srozumitelný pro dost lidí. Další funkcí, kvůli které je tento diagram oblíbený, je možnost přidávat finanční a lidské zdroje k jednotlivým částem [34].

Přestože je Ganttův diagram velmi jednoduchý a srozumitelný, je tomu tak pouze u menších projektů. Konkrétně pak u projektů, které se vejdou na pár stránek papíru nebo na obrazovku.

V současné době je k dispozici spousta programů, které nám pomohou s tvorbou a správou těchto diagramů. Mezi nejznámější patří placený program Microsoft Project[8] a například zdarma dostupný software GanttProject[9] [34].



Obrázek 10: Ganttův diagram Zdroj: vlastní

[7] je softwarový démon, který v operačních systémech automatizovaně spouští v určitý čas nějaký příkaz resp. proces
[8] Dostupné z: https://products.office.com/cs-cz/project/project-and-portfolio-management-software?tab=tabs-1
[9] Dostupné z: http://www.ganttproject.biz/

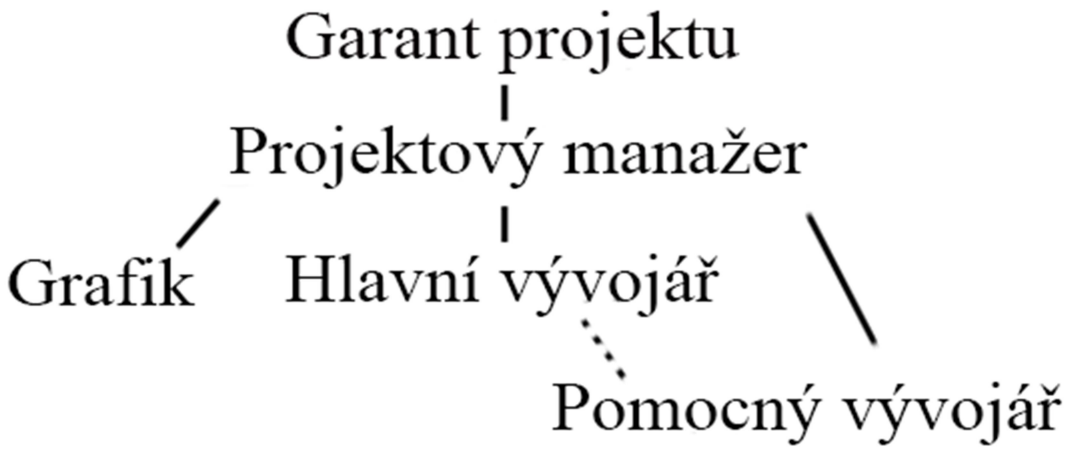
Ganttův diagram je velmi přehledný díky své grafické jednoduchosti (*Obrázek 10*). Na horizontální ose je vždy čas a na vertikální ose jsou pak jednotlivé dílčí úkoly (části projektu). Ty jsou pak v našem případě znázorněny modrými pruhy v diagramu. V levé části diagramu je přehled, který slouží k rychlé orientaci mezi úkoly a zdroji [34].

6.4 Složení týmu

Správná a rychlá realizace projektu je samozřejmě závislá i na vhodném složení pracovního týmu. Pro náš projekt je důležité mít vyváženou řídící a výkonnou rovinu. V terminologii „rolí v projektovém týmu“ se vyskytuje bezpočet definovaných rolí [13].

V prostředí firmy HOPE Studio se ale tyto role míchají, stejně jako jejich pravomoci. Je to dáno velmi malým počtem zaměstnanců. Nejedná se tak o klasický projektový tým, i když i zde jsou jednoznačně rozděleny pravomoci a role do řídící a výkonné roviny. Z týmových rolí, které popisuje *Obrázek 11*, patří do řídící roviny garant projektu a projektový manažer. Do výkonné roviny patří grafik a vývojáři [13].

Garant projektu má rozhodující slovo ve všem, co se projektu týče. V tomto konkrétním případě je to vlastník deskové hry a zadavatel onlinové verze. Manažer projektu potom, dohlíží na správné a rychlé řešení projektu a jeho dílčích částí. Grafik dělá grafiku přímo na míru pro online verzi, případně upravuje grafiku z deskové hry, aby byla dále použitelná. Hlavní vývojář má potom na starosti projekt z technické stránky (hardware). Dále dohlíží na univerzálnost, strukturovanost a znovupoužitelnost kódu a samozřejmě tvoří většinu kódu. Pomocný vývojář je podřízen v hierarchii projektovému manažeru, ovšem přijímá úkoly i od hlavního vývojáře. Celková hierarchie je znázorněna na *Obrázek 11* [13].



Obrázek 11: Struktura týmu Zdroj: vlastní

Výhodou takto malého týmu je velmi snadná komunikace. Vývojáři mohou komunikovat a úkolovat se přímo mezi sebou, což šetří čas a snižuje riziko špatného předání informací. Stejně tak se mohou obrátit v případě potřeby přímo na grafika. Obvyklá forma komunikace je email, případně telefon nebo přímá domluva v kanceláři. Zásadním problémem tohoto postupu komunikace je možnost desinformace řídící roviny. Proto se používá úkolovací systém Redmine[10]. Do tohoto systému se přidávají veškeré domluvené práce, a to vždy ke správnému úkolu. Zároveň přidávají jednotlivý účastníci vývojového procesu ke každému úkolu strávený čas a postup. Členové řídící roviny potom mohou jednoduše sledovat, kdo na čem pracuje, v jaké fázi se vývoj nachází a efektivně poté řídit a kontrolovat celý vývoj [13].

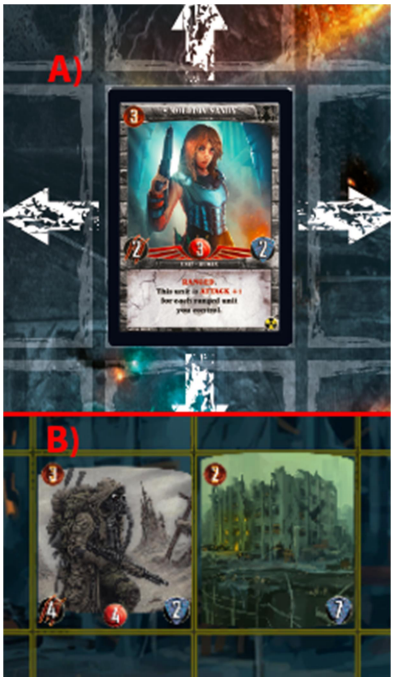
[10] Dostupné z: http://www.redmine.org/

7 Praktická část

7.1 HTML a CSS rozložení prostředí

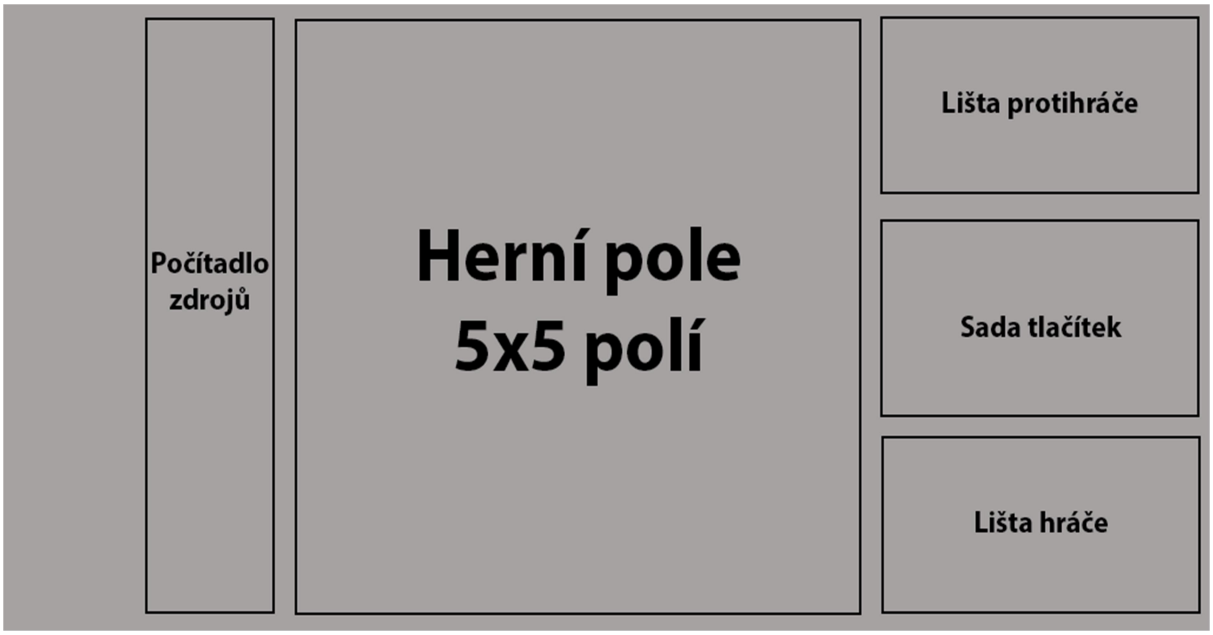
Dobrá hratelnost je založena na promyšleném a dobře navrženém herním prostředí. Nejlepší, vzhledem k autentičnosti hry, by bylo použít původní rozložení prvků. Bohužel, převodem do online verze, vznikají nové prvky, které se v deskové hře nevyskytují.

V této onlinové hře máme několik nových herních prvků. Nejdominantnější je ovšem herní pole z původní verze. V herním plánu z deskové hry mají políčka poměr 16:9 a jsou na ní pokládány celé karty. Bohužel tento poměr by se na obrazovku většiny současných notebooků a počítačů nevešel, a proto jsem se rozhodl políčka zmenšit na čtverce, v kterých se budou zobrazovat pouze miniatury karet. Tato změna konceptu je zaznamenána na *Obrázek 12*. V horní části pod písmenem A je původní velikost v deskové hře a pod písmenem B je zmenšená mřížka z online hry.

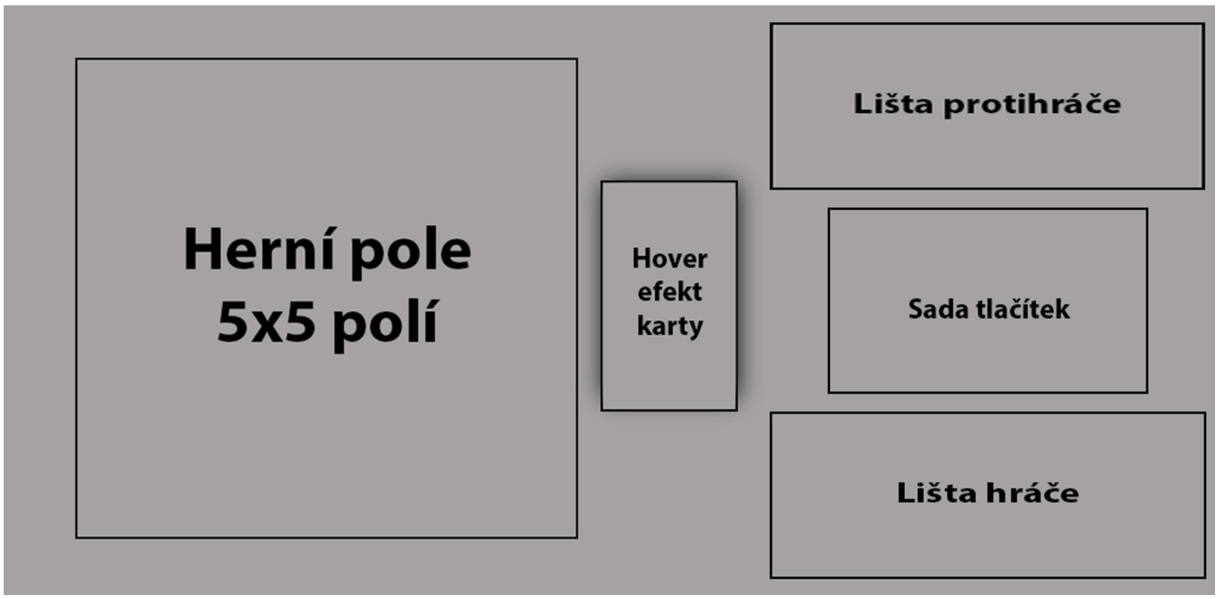


Obrázek 12: Změna velikosti mřížky (A - desková hra, B - online verze)
Zdroj: vlastní

Do zbylého prostoru musíme vměstnat ostatní prvky (přehled zdrojů, hráčovi a protihráčovi karty, hráčské ikonky (avatary) se jmény, odpočet času a sadu



Obrázek 13: Základní rozložení
Zdroj: vlastní

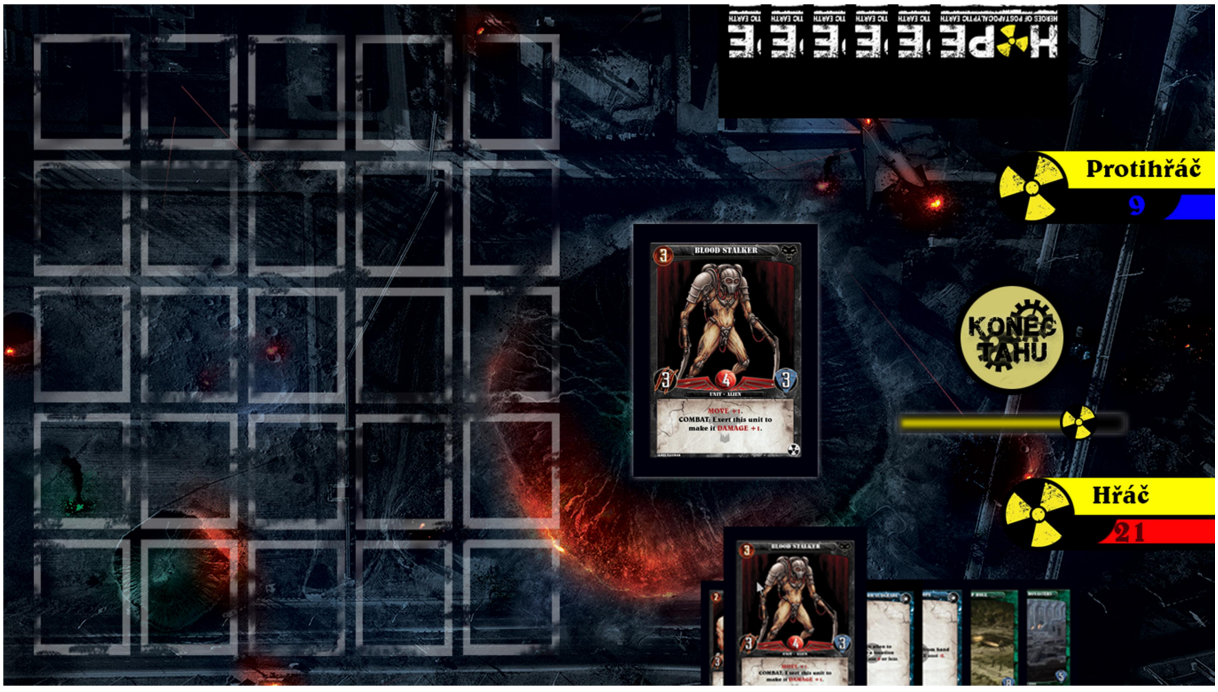


Obrázek 14: Závěrečné rozložení
Zdroj: vlastní

tlačítek na ovládání hry a prostředí).

Na *Obrázek 13* je vidět rozložení herních prvků, které zachovává originální rozložení herního plánu, jako je v deskové hře. Toto rozložení ale nevyužívá herní plochu v levé části, kdežto pravá část je přeplněná. Další nevýhodou tohoto rozložení je, že počítadlo zdrojů je osamoceno v levé části, ale ostatní herní prvky byly v části pravé. Hráč tedy neustále musel skákat pohledem z leva doprava.

Proto bylo nutné zakomponovat zobrazení zdrojů také do pravé části. Kvůli tomu jsem použil



Obrázek 15: Grafický návrh rozložení
Zdroj: vlastní

decentralizovaný koncept. Herní plán se nachází v levé části, čímž uvolnil více místa v pravé části. Toto místo jsem pak využil pro zpřehlednění prvků, které tam již byly. Počítadlo zdrojů jsem předělal na lištu pod jménem hráče. Zmenšil jsem tím jeho plochu. Došlo tím zase k zpřehlednění celého okna. Vzniklé volné místo jsem pak využil k umístění posledního prvku. Protože jsem odstranil z herního plánu celé karty a nahradil je miniaturami, je nyní potřeba náhledové okno pro karty. V tomto okně se zobrazí karta, nad kterou má hráč kurzor. Toto rozložení je zobrazeno na *Obrázek 14*. Výhodou tohoto rozložení taky je, že karty v ruce mohou být stranou a zmenšeny, protože plná velikost se zobrazuje v náhledu.

Obrázek 15 potom ukazuje kompletní grafickou šablonu pro zpracování. Odpočítávání času jsem udělal ve dvou variantách. Žlutá varianta pro zobrazení zbývajícího času ve fázi a červená varianta pro zobrazení zbývajícího času na zahájení reakce. Tlačítko na ukončení fáze/tahu jsem udělal pro každou rasu vlastní. V šabloně lze měnit podkladový obrázek, tj. pro každou hru tam bude moci být nahrána jiná podkladová grafika, aby hra nevypadala pořád stejně.

7.2 Preloader

HOPE Cardgame Online bude hra zařazená do žánru Browser Games, tj. hra se bude hrát pomocí prohlížeče. To v sobě nese několik výhod a nevýhod. Hlavní výhodou je vysoká kompatibilita napříč platformami. Hlavní nevýhodou je naopak vysoká zátěž na datové připojení. Prohlížeč se snaží bránit opakovanému načítání dat v podobě cashe. Grafiku, kterou již stahoval, nemaže a nechává si ji v paměti pro další použití. Toho budeme využívat i my v naší hře.

```
57    opts.objects.children().each(function () {
58        $(this).imagesLoaded().done(function () {
59            //alert('done');
60            $('#procento').animate({
61                width: procento+procentoobject+"%"
62              }, 100, function() {
63                procento += procentoobject;
64                if(procento>=100)
65                    $(this).animationComplete();
66              }).text(procento+procentoobject+"%");
67        });
68
```

Obrázek 16: Část algoritmu
Zdroj: vlastní

Preloader grafických prvků je tedy skript v JQuery, který při načítání hry bude ověřovat která tato grafika je již připravena (část kódu je na *Obrázek **16***) a kolik jí je ještě potřeba načíst. Skript porovnává načtenou grafiku oproti seznamu potřebné grafiky a počítá, kolik procent je již načteno. Toto procento zobrazuje pomocí progress baru na obrazovce (*Obrázek 17*) [9].

Obrázek 17: Preloader
Zdroj: vlastní

7.3 Animace pomocí jazyka JQuery

Animace jsou v dnešní době tou hlavní částí počítačových her. Úkolem tedy bylo vytvořit sadu funkcí pro základní rozpohybování herního rozložení. Jelikož jsou v šesté fázi (Draw), karty dobírány automaticky, bylo potřeba napsat animaci pohybu karet z balíčku do hráčovy ruky. Karta se během tohoto pohybu otočí lícem nahoru. Další neméně důležitou částí bylo rozpohybovat odpočítávání času [9].

7.4 Garbage collector

Úspěšnost onlinové hry je mimo jiné založena i na velmi kvalitní grafice, proto tvoří grafické podklady téměř 100% dat. Soubory se neustále duplikují z hlavního na vedlejší servery. V případě odstranění jedné reference z databáze, vznikne hned několik souborů, o kterých aplikace neví, že tam jsou. Právě kvůli tomu je potřeba napsat skript, který se bude automaticky spouštět a tato data v podobě obrázků vyhledávat a mazat.

Vzhledem k struktuře aplikace je výskyt duplicitní grafiky omezen pouze na adresář card-illustrations. Skript bude tedy vyhledávat a mazat data pouze v této složce. Ovšem, aby byl skript jednoduše použitelný v případě změny adresářové struktury, nebo rozšíření hledání mimo složku card-illustrations, je předávána cesta ke složce pomocí parametru.

```
16  $cesta = './card-illustrations/';
17
18  include_once(dirname(__FILE__).'/lib/cleanIllustration.php');
19  cleanIllustrations($cesta);
```

Obrázek 18: Volání čištění ilustrací
Zdroj: vlastní

Díky tomu, že nelze přesně definovat vnitřní strukturu složky card-illustrations, rozhodl jsem se použít rekurzivní algoritmus. K tomu se příhodně hodí možnost předat adresu ke složce pomocí parametru (*Obrázek **18***). Rekurzivní algoritmus tedy rozdělí položky v adresáři na adresáře a na soubory. Následně potom porovná soubory s databází a smaže soubory, které v databázi nejsou. Pak se rekurzivně zavolá na všechny adresáře v adresáři. V případě, že jméno souboru začíná znakem '!', soubor se nesmaže. Pokud je takto označena složka, algoritmus se na ní nezavolá. Algoritmus tímto postupem prohledá celou složku napříč všemi úrovněmi.

Součástí tohoto algoritmu je i mazání prázdných adresářů a možnost potlačit ignoraci složky, či soubor při mazání. Oba tyto parametry jsou předávány algoritmu pomocí parametrů typu *boolean*.

V případě, že si přejeme zachovat adresářovou strukturu, zvolíme pro druhý parametr hodnotu *true*, jinak zvolíme hodnotu *false*. Pro potlačení ignorování složek a souborů musíme pro třetí parametr zvolit hodnotu *true*.

Abych docílil toho, že se bude skript automaticky spouštět a bude tak zajištěna pravidelná kontrola a čištění, je skript volán z CRONu serveru.

7.5 Validace karetních balíčků

Jak bylo vysvětleno na začátku, hra obsahuje 144 karet rozdělených do 4 základních balíčků. Mimo tyto 4 základní balíčky, si hráči mohou skládat své vlastní balíčky, které se budou lépe hodit k jejich hernímu stylu.

Skládání balíčku je ovlivněno několika pravidly, které hráč musí dodržet, aby byl jeho balíček použitelný ke hře. Pravidla jsou následující:

- Balíček musí mít přesně 36 karet
- V balíčku nesmí být více než 2 stejné karty
- Balíček musí splňovat pravidla pro míchání ras

Obrázek 19: Diagram ras [13]

Libovolné míchání ras je zakázáno vzhledem k hernímu příběhu a má tedy také svá pravidla. Tato pravidla se dají převést na množinu povolených kombinací (viz *Obrázek 19*). Pro zjednodušení jejich výčtu zavedu zkratky pro jednotlivé rasy – Mimozemšťané (A), Lidé (H), Mutanti (M) a Roboti (R). Povolené kombinace tedy jsou následující:

- A + M
- M + H
- H + R

Ostatní, tedy zakázané kombinace, jsou:

- A × H
- A × R
- M × R
- všechny kombinace tří nebo 4 ras

Z těchto kombinací a výše zmíněných pravidel, pak vznikne množina podmínek. Tuto množinu jsem potom minimalizoval pomocí Karnaughovy mapy.

Algoritmus se bude muset spouštět na straně uživatele v podobě skriptu v JQuery a na straně serveru jako skript PHP. Na straně uživatele se bude automaticky spouštět při změně obsahu balíčku, aby uživatel dostal okamžitou zpětnou vazbu na změny, které provedl. Po uložení balíčku se znovu provede kontrola jeho platnosti na straně serveru. A to proto, aby uživatel nemohl hrát s balíčkem, který by nesplňoval zmíněná pravidla.

Jako zpětnou vazbu uživateli vždy napíše „Validní balíček“ a „Nevalidní balíček“ + krok, na kterém se kontrola zastavila.

8 Závěr

Cílem práce bylo seznámit se s projektem HOPE Cardgame a HOPE Cardgame Online od liberecké společnosti HOPE Studio s.r.o. Následně pak implementovat do projektu HOPE Cardgame Online garbage collector, validaci balíčků v jazyce PHP, animace a preloader v jazyce JQuery. Další částí zadání bylo navrhnout pro HOPE Cardgame Online rozložení herních prvků.

V této práci byla tedy provedena analýza projektu HOPE Cardgame. Na základě této analýzy jsem vybral vhodné postupy a technologie pro dílčí části projektu.

Pro rozložení herních prvků jsem zvolil framework Bootstrap. Výhodou frameworku je responzivnost na různých zobrazovacích zařízení. S Bootstrapem se pracuje velmi dobře. Práce s ním je rychlá a intuitivní.

Animace a preloader jsem dělal pomocí JQuery. Příjemně mě překvapila použitelnost a jednoduchost této JavaScriptové knihovny. Knihovna má velmi kvalitní dokumentaci a širokou základnu programátorů. Tuto knihovnu pro jazyk JavaScript bych, vzhledem k tomu, jak dobře se s ním pracovalo, použil na všechny další případné skripty na straně uživatele.

Poslední dílčí částí byla práce s jazykem php na straně serveru. Bohužel projekt HOPE Cardgame Online byl od začátku implementován bez použití frameworku. Udržitelnost dobré kondice projektu bez použití frameworku bude velmi náročná. Určitě bych tedy do budoucna doporučil převést tento projekt do jednoho z frameworků.

Garbage collector je skript v PHP, který hledá nepoužívanou grafiku a maže ji. Vzhledem k tomu, že adresářová struktura je neměnná, použil jsem rekurzivní algoritmus. Algoritmy, které používají rekurzi, jsou velmi rychlé. Problém těchto algoritmů je ukončovací podmínka. V případě, že není napsána správně, může dojít k zacyklení celého algoritmu. Proto jsem kladl velký důraz na její správnost a důkladné testování.

Validátor balíčků je minimalizovaná množina podmínek z pravidel hry HOPE Cardgame. Tato množina pak byla naimplementována do algoritmu a vznikla struktura velmi podobná stromu. Díky tomu je algoritmus velmi rychlý, protože nekontroluje všechny možnosti.

Tento projekt má podle mě velký potencionál k úspěchu. Nicméně se jedná v tuto chvíli pouze o Alpha verzi. Poté, co bude hotova samotná hra, bude potřeba projekt ještě rozšířit

o možnost plateb v aplikaci. Dalším vhodným rozšířením aplikace by pak byla možnost hrát proti počítačovému hráči.

Seznam použité literatury

[1] MONUS, Anna. 10 PHP Frameworks For Developers – Best of. Hongkiat [online]. Dostupné z: http://www.hongkiat.com/blog/best-php-frameworks/

[2] ČERMÁK, Josef. Analýza rizik: kvalitativní analýza rizik. Clever and Smart [online]. 2013. Dostupné z: http://www.cleverandsmart.cz/analyza-rizik-kvalitativni-analyza-rizik/

[3] ČERMÁK, Josef. Analýza rizik: kvantitativní vs. kvalitativní. Clever and Smart [online]. 2011. Dostupné z: http://www.cleverandsmart.cz/analyza-rizik-kvantitativni-vs-kvalitativni/

[4] Basic MVC Architecture: Simply Easy Learning. Tutotrialspoint [online]. Dostupné z: https://www.tutorialspoint.com/struts_2/basic_mvc_architecture.htm

[5] ROUSE, Margaret. Desktop virtualization [online]. 2011. Dostupné z: http://searchvirtualdesktop.techtarget.com/definition/desktop-virtualization

[6] Extreme programming. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001. Dostupné z: https://en.wikipedia.org/wiki/Extreme_programming

[7] Extreme Programming: A gentle introduction [online]. Dostupné z: http://www.extremeprogramming.org/

[8] BEYDEDA, Sami., Matthias. BOOK a Volker. GRUHN. Model-driven software development. New York: Springer, 2005. ISBN 9783540256137.

[9] CHAFFER, Jonathan a Karl SWEDBERG. Mistrovství v jQuery. 1. Brno: Computer Press, 2013. ISBN 8025140873.

[10] GUTMANS, Andi, Stig Sæther BAKKEN a Derick RETHANS. Mistrovství v PHP 5. Vyd. 2. Brno: Computer Press, 2008. ISBN 978-80-251-1519-0.

[11] What Are The Benefits of MVC? [online]. 2008. Dostupné z: http://blog.iandavis.com/2008/12/what-are-the-benefits-of-mvc/

[12] Pravidla HOPE Cardgame. 2. Liberec: HOPE Studio, 2016.

[13] Projektové role: Elektronický manuál Projektové kanceláře MK. Projektové řízení [online]. Dostupné z: http://projektoverizeni.mkcr.cz/projektove-role/

[14] Sequence diagrams. The Unified Modeling Language [online]. Dostupné z: http://www.uml-diagrams.org/sequence-diagrams.html

[15] BOEHM, Barry. Spiral Development: Experience, Principles, and Refinements [online]. 2000. 49. Dostupné z: http://www.sei.cmu.edu/reports/00sr008.pdf

[16] State machine diagrams. The Unified Modeling Language [online]. Dostupné z: http://www.uml-diagrams.org/state-machine-diagrams.html

[17] Tag Archives: MDA. Simplicity Through Breadth [online]. Dostupné z: https://faisalsikder.wordpress.com/tag/mda/

[18] The Unified Modeling Language. The Unified Modeling Language [online]. Dostupné z: http://www.uml-diagrams.org/

[19] UML 2.5 Diagrams Overview. The Unified Modeling Language [online]. Dostupné z: http://www.uml-diagrams.org/uml-25-diagrams.html

[20] UML Use Case Diagrams. The Unified Modeling Language [online]. Dostupné z: http://www.uml-diagrams.org/use-case-diagrams.html

[21] What can PHP do? Php [online]. Dostupné z: http://php.net/manual/en/intro-whatcando.php

[22] What is Prototype model- advantages, disadvantages and when to use it? *ISTQB EXAM CERTIFICATION* [online]. Dostupné z: http://istqbexamcertification.com/what-is-prototype-model-advantages-disadvantages-and-when-to-use-it/

[23] What is Iterative model- advantages, disadvantages and when to use it? *ISTQB EXAM CERTIFICATION* [online]. Dostupné z: http://istqbexamcertification.com/what-is-iterative-model-advantages-disadvantages-and-when-to-use-it/

[24] What is Incremental model- advantages, disadvantages and when to use it? *ISTQB EXAM CERTIFICATION* [online]. Dostupné z: http://istqbexamcertification.com/what-is-incremental-model-advantages-disadvantages-and-when-to-use-it/

[25] What is Waterfall model- advantages, disadvantages and when to use it? *ISTQB EXAM CERTIFICATION* [online]. Dostupné z: http://istqbexamcertification.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/

[26] *UML Class and Object Diagrams Overview* [online]. Dostupné také z: http://www.uml-diagrams.org/class-diagrams-overview.html

[27] HTML 5.1. W3C [online]. Dostupné z: https://www.w3.org/TR/html/

[28] *What is HTML?* [online]. Dostupné také z: http://www.yourhtmlsource.com/starthere/whatishtml.html

[29] Software framework. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-. Dostupné z: https://en.wikipedia.org/wiki/Software_framework

[30] *What is a Web Framework?* [online]. Dostupné z: https://jeffknupp.com/blog/2014/03/03/what-is-a-web-framework/

[31] Seznámení s Nette Frameworkem. Nette [online]. Dostupné z: https://doc.nette.org/cs/2.4/getting-started

[32] *The Best PHP Framework for 2015: SitePoint Survey Results* [online]. Dostupné z: https://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/

[33] Introduction to Laravel Framework. Laravelbook [online]. Dostupné z: http://laravelbook.com/laravel-introduction/

[34] *Gantt.com* [online]. Dostupné také z: http://www.gantt.com/

[35] *What are Frameworks? 22 Best Responsive CSS Frameworks for Web Design* [online]. Dostupné z: https://www.awwwards.com/what-are-frameworks-22-best-responsive-css-frameworks-for-web-design.html

[36] Vodopádový model. Testování softwaru [online]. Dostupné z: http://testovanisoftwaru.cz/manualni-testovani/modely-zivotniho-cyklu-softwaru/vodopadovy-model/

[37] Spirálový model. Testování softwaru [online]. Dostupné z: http://testovanisoftwaru.cz/manualni-testovani/modely-zivotniho-cyklu-softwaru/spiralovy-model/

[38] Tag Archives: MVC Architecture. YII FOR BEGINNERS [online]. Dostupné z: https://yii4beginners.wordpress.com/tag/mvc-architecture/

[39] Sequence Diagram Tutorial. Lucid chart [online]. Dostupné z: https://www.lucidchart.com/pages/uml/sequence-diagram

[40] Stavový diagram. Příklady použití diagramů UML 2.0 [online]. Dostupné z: http://uml.czweb.org/stavovy_diagram.htm

[41] http://eshop.hopestudio.cz/img/p/8/8/88-large_default.jpg

[42] Vytvoření diagramu tříd. Fakulta informačních technologií, VUTBR [online]. Dostupné z: http://www.fit.vutbr.cz/study/courses/MPR/public/metodika-oo/cz/d2t.htm

Seznam tabulek a obrázků