Česká zemědělská univerzita v Praze
Provozně ekonomická fakulta
Katedra informačních technologií


Diplomová práce

# Webová aplikace pro podporu korektur textu

***Bc. Radek Bernátek***

# ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Katedra informačních technologií

Provozně ekonomická fakulta

# ZADÁNÍ DIPLOMOVÉ PRÁCE

Bernátek Radek

Informatika

Název práce

**Webová aplikace pro podporu korektur textu**

Anglický název

**Web application for support of a text correction**

**Cíle práce**

Diplomová práce je tematicky zaměřena na problematiku online korektur. Hlavním cílem práce je analýza požadavků uživatelů na online korektorské práce, její nabídky a poptávky včetně existujících řešení.

Dílčí cíle práce jsou:

- navrhnout systém dle požadavků uživatelů,
- realizace systému včetně implementace vybraných technologií podle jeho návrhu,
- zvolit vhodné testy a podrobit jim vytvořený systém.

**Metodika**

Metodika řešené problematiky diplomové práce je založena na studiu a analýze odborných informačních zdrojů. Praktická část práce je zaměřena na návrh a realizaci systému podle analyzovaných uživatelských požadavků a jeho následné otestování. Na základě syntézy teoretických poznatků a výsledků praktické části práce budou formulovány závěry diplomové práce.

**Harmonogram zpracování**

1. Studium odborných informačních zdrojů, stanovení dílčích cílů a postupů řešení: 06/2012
2. Zpracování teoretických východisek práce (přehledu řešené problematiky): 07/2012 - 09/2012
3. Vypracování vlastního řešení, diskuze, doporučení a závěry: 10/2012 - 02/2013
4. Tvorba finálního dokumentu práce: 02/2013 - 03/2013
5. Odevzdání diplomové práce a tezí: 03/2013

**Rozsah textové části**

70 - 80 stran

**Klíčová slova**

korektury, typografie, gramatika, stylistika, online kontrola, php, mysql, javascript

**Doporučené zdroje informací**

GUTMANS, Andi. Mistrovství v PHP 5. Vyd. 2. Brno: Computer Press, 2007. ISBN 978-802-5115-190.

STŘÍŽ, Pavel a Martin STŘÍŽ. Dokumentace: Velké makro, aneb, Nový způsob zpracování korektur. Vyd. 1. Bučovice: Martin Stříž, 2008. ISBN 978-808-7106-143.

KOFLER, Michael. Mistrovství v MySQL 5. Vyd. 1. Překlad Jan Svoboda, Ondřej Baše, Jaroslav Černý. Brno: Computer Press, 2007. ISBN 978-802-5115-022.

Komunita JQuery. JQuery: kuchařka programátora. Vyd. 1. Brno: Computer Press, 2010. ISBN 978-802-5131-527.

KRUG, Steve. Nenuťte uživatele přemýšlet!: praktický průvodce testováním a opravou chyb použitelnost [sic] webu. Vyd. 1. Brno: Computer Press, 2010. ISBN 978-802-5129-234.

**Vedoucí práce**

Šimek Pavel, Ing., Ph.D.

**Termín odevzdání**

březen 2013

**doc. Ing. Zdeněk Havlíček, CSc.**
Vedoucí katedry

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE
Provozně ekonomická fakulta

**prof. Ing. Jan Hron, DrSc., dr.h.c.**
Děkan fakulty

V Praze dne 15.1.2013

# Prohlášení

Prohlašuji, že svou diplomovou práci „Webová aplikace pro podporu korektur textu“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 27. listopadu 2013 .....

# Poděkování

Tímto bych chtěl poděkovat vedoucímu práce Ing. Pavlovi Šimkovi, Ph.D. za ochotu, trpělivost a odborné vedení. Dále všem dobrovolným i nedobrovolným účastníkům testování a všem, kteří mě při práci podporovali.

# Webová aplikace pro podporu korektur textu

**Souhrn**
Diplomová práce je tematicky zaměřena na problematiku online korektur. Cílem práce je analýza požadavků uživatelů na online korektorské práce, její nabídky a poptávky včetně existujících řešení. Součástí je i implementace systému dle vytvořeného návrhu za použití vhodných technologií a její řádné otestování.

**Klíčová slova:** korektury, typografie, gramatika, stylistika, online kontrola, PHP, MySQL, JavaScript

# Web application for support of a text correction

**Summary**
This master's thesis is focused on the issue of online text correction. The aim of this thesis is the analysis of user requirements for online proofreading work, its supply and demand including existing solutions research. Thesis also describes implementation of system according to design using appropriate technologies and its proper testing.

**Keywords:** proofreading, typography, grammar, stylistics, online checking, PHP, MySQL, JavaScript

# Obsah

# Seznam obrázků

# Seznam tabulek

# Úvod

V dnešní době nabízí stále se rozrůstající internet nové a nové služby podpořené rozvíjejícími se technologiemi a snaží se pokrýt veškeré požadavky lidí od nejzákladnějších až po ty nejpokročilejší.

Jedním z požadavků, který se zatím na internetu příliš neprosadil, je možnost poptávky a nabídky korektur textů jak z hlediska gramatiky, stylistiky, typografie, tak i správného citování a kvality výsledného textu. Existují určité formy, které tento požadavek řeší, ale nejsou efektivní a specializované. Jedná se především o nabídku typografických prací pomocí inzertních portálů a jednoduchých webových prezentací. Tyto systémy ale nenabízejí téměř žádnou interakci uživatelů a správu souvisejících dokumentů – jako hlavní komunikační kanál slouží elektronická pošta a jiné kanály, pro tento účel ne zcela dostačující. Moderní webové technologie a služby ale dokážou nabídnout uživatelům mnohem sofistikovanější funkce, které je nutné pouze správně využít.

Se stále zvyšujícím se počtem a rozvojem zpravodajských serverů by měla vzrůstat i poptávka po zkušených korektorech. Korektorské práce jsou na zpravodajských portálech často podceňovány a mnohokrát se uživateli zobrazí obsah v podobě, ve které by ho žádný korektor nepublikoval. Nehledě na to, že redaktoři mají na internetu možnost rychlé opravy, je velkou chybou, že tyto překlepy, gramatické či stylistické chyby a mnohé typografické prohřešky jsou odhaleny až samotnými čtenáři. Východiskem pro tato řešení může být systém, který bude umožňovat poptávat a nabízet korektorské práce.

Tento neziskový projekt by tedy měl nabídnout možnosti realizace korektorských prací jak nabízejícím, tak i poptávajícím uživatelům a to především k maximální spokojenosti všech zúčastněných stran.

Kapitola 1

# Cíl práce a metodika

## 1.1 Cíl práce

Diplomová práce je tematicky zaměřena na problematiku online korektur. Hlavním cílem práce je analýza požadavků uživatelů na online korektorské práce, její nabídky a poptávky včetně existujících řešení.

Dílčí cíle práce jsou:

- navrhnout systém dle požadavků uživatelů,
- realizace systému včetně implementace vybraných technologií podle jeho návrhu,
- zvolit vhodné testy a podrobit jim vytvořený systém.

## 1.2 Metodika

Metodika řešené problematiky této diplomové práce je založena na studiu a analýze odborných informačních zdrojů týkajících se korektur textu. Po analýze odborné literatury, rešerši existujících řešení a jejich zhodnocení, kterým je věnována druhá kapitola, bude následovat teoretická část práce, ze které vzejde analýza a podklady pro praktickou část – implementaci webové aplikace, na kterou naváže popis a výsledky testování, a nakonec i zhodnocení celého výsledku spolu s doporučeními pro další vývoj.

Ve třetí kapitole budou definovány a analyzovány požadavky funkční i obecné, na jejichž základě bude popsána interakce uživatelů a systému pomocí případů užití. Na tuto sekci navazuje prezentace doménového modelu zachycujícího statický pohled na modelovaný systém, analýza a návrh uživatelského rozhraní, popis použitých technologií,

zabezpečení a nakonec diagram nasazení znázorňující rozložení a vzájemnou komunikaci použitých technologií na hardwarových zdrojích.

Čtvrtá kapitola se věnuje už samotné realizaci – popisu vytvořeného kódu a struktury nejdůležitějších komponent. Dále se věnuje detailnímu popisu použitých technologií na backendu, jejichž základ tvoří Nette Framework spolu s ORM, ale i na frontendu, kde převažuje Twitter Bootstrap s LESS a souvisejícími JavaScriptovými komponentami. Dále zde bude vysvětlen postup, jakým způsobem bude systém testován pomocí Selenia a PHPUnit testů. Bude popsána množina nástrojů (verzovací systém Git, databázové migrace, issue tracking, průběžná integrace, ...) umožňujících rychlý a efektivní vývoj webových aplikací spolu s postupem, jak budou tyto nástroje nastaveny a efektivně využívány.

Poslední kapitola bude zaměřena na důkladné otestování výsledné aplikace. Jsou zde popsány různé metody testování a dále konkrétní postupy jednotlivých testů a jejich výsledků spolu s dalšími doporučeními. Systém bude podroben jak průběžnému automatickému testování funkčnosti, tak i testování uživatelskému, zobrazení v prohlížečích, použitelnosti a přístupnosti dle zvolené metodiky WCAG 2.0. Souhrn výsledků všech testování spolu s návrhem na odstranění možných nedostatků bude popsán v závěru této kapitoly.

Na základě syntézy teoretických poznatků a výsledku praktické části práce budou formulovány závěry diplomové práce spolu s návrhy pro další vývoj a rozšiřování aplikace.

Kapitola 2

# Přehled řešené problematiky

## 2.1 Korektury

Slovo vychází z latinského *„corriegere“*, což v překladu znamená opravovat. Korektura je zpravidla opakující se proces zjištění a vyznačení chyb, nepřesností a nesrovnalostí textů. Jedná se o souběh dvou typů oprav:

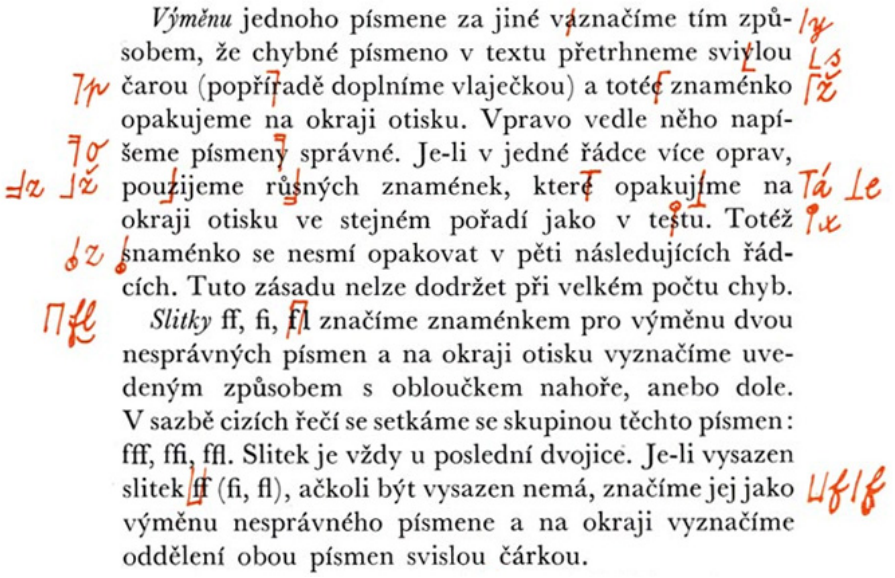
- odstranění objektivních chyb (pravopis, překlepy a chybná sazba),
- věcná či stylistická revize textu.

Samotná korektura textu vyžaduje výbornou znalost jazyka použitého v textu. Proces stylistické úpravy textu se skládá ze standardizace větných vazeb, úpravy slovosledu, předložkových vazeb, sjednocení použitých jazykových prostředků v celém kontrolovaném textu a eliminace pleonasmů – významově nadbytečných slov a tvarů apod.

Při provádění korektur bylo v minulosti hojně využíváno porovnávání kontrolovaného textu s rukopisem. Další osvědčenou formou korektury je takzvaná náslechová metoda. Ta spočívá v kooperaci dvou korektorů, kdy jeden z nich text četl a druhý opravoval. Tuto metodu může jednotlivec samozřejmě praktikovat za pomoci diktafonu. Texty psané rukou pomalu nahrazují elektronické dokumenty převedené do tiskové podoby, a proto se náslechová korektura nebo korektura porovnávání textu a rukopisu dnes již užívají minimálně. (14)

Zmíněné opravy se provádějí pomocí korekturních znamének, které jsou dány normou ČSN 88 0410 Korekturní znaménka a jejich používání. (27)

Ukázka takto opraveného dokumentu za pomoci korekturních znamének je znázorněna na obrázku 2.1.



*Výměnu* jednoho písmene za jiné vyznačíme tím způsobem, že chybné písmeno v textu přetrhneme svislou čarou (popřípadě doplníme vlaječkou) a totéž znaménko opakujeme na okraji otisku. Vpravo vedle něho napíšeme písmeny správné. Je-li v jedné řádce více oprav, použijeme různých znamének, které opakujeme na okraji otisku ve stejném pořadí jako v textu. Totéž znaménko se nesmí opakovat v pěti následujících řádcích. Tuto zásadu nelze dodržet při velkém počtu chyb.

*Slitky* ff, fi, fl značíme znaménkem pro výměnu dvou nesprávných písmen a na okraji otisku vyznačíme uvedeným způsobem s obloučkem nahoře, anebo dole. V sazbě cizích řečí se setkáme se skupinou těchto písmen: fff, ffi, ffl. Slitek je vždy u poslední dvojice. Je-li vysazen slitek ff (fi, fl), ačkoli být vysazen nemá, značíme jej jako výměnu nesprávného písmene a na okraji vyznačíme oddělení obou písmen svislou čárkou.

Obrázek 2.1: Ukázka dokumentu opraveného pomocí korektorských znamének (12)

**Základní dělení korektur (3):**

1. **Jazyková korektura** – zahrnuje opravu lexikálních, morfologických a syntaktických chyb. Je základní součástí překladů a korektor při tomto druhu korektury zkoumá, zda překlad splňuje svůj účel. To znamená, že přeložený text je terminologicky jednotný a jazykový rejstřík a styl je adekvátní. Dále kontroluje celkovou kvalitu textu – srozumitelnost, gramatiku, úplnost překladu a dodržení grafické úpravy. Pokud se strany nedomluví jinak, odstraňuje ihned překlepy a drobné chyby, v opačném případě je vyznačuje pomocí znamének.

2. **Odborná korektura** – zachycuje chyby v použití odborných termínů pro daný účel textu a provádí jednoznačnou kontrolu textu. Klient by měl upozornit na speciální terminologii a popřípadě dodat i související terminologický slovník či referenční texty vysvětlující například použití neznámých zkratek a upozornit na nepřekládání některých výrazů.

3. **Stylistická korektura** – upravuje volbu slov, pořadí a vzájemnou souvislost slov ve větě. Cílem stylistické korektury je zlepšení čitelnosti a srozumitelnosti textu pro daný účel.

4. **Předtisková korektura** – je závěrečná korektura před tiskem nebo například před publikováním na webu. Má za úkol odstranit typografické chyby jako jsou například přetékající texty a obrázky, špatné zalamování slov na konci řádků, diakritická znaménka a nevhodně použité znaky a symboly.

### 2.1.1 Počítačové programy

Korekturu textu lze provádět mnoha způsoby – jedním z nich je využití textového editoru. Moderní textové editory nabízí mnoho způsobů jak zvýrazňovat chyby, eventuálně je dokážou automaticky opravit. Je k tomu potřeba, aby byl editor především vybaven plnohodnotným slovníkem. Následující seznam popisuje možnosti několika z nich.

1. **Microsoft Word, LibreOffice/OpenOffice Writer** – nabízí kontrolu a automatickou opravu chyb jako jsou překlepy, interpunkce, shoda podmětu s přísudkem atd. Dokáže podchytit základní chyby a na ostatní se snaží upozornit různě barevnými podtrženími.

2. **WinEdt** – editor určený především pro kontrolu zdrojových textů pro LaTeX včetně kontroly použitých šablon, pravopisu, vložených grafických symbolů a interpunkce. Stejně jako Word obsahuje český slovník o obsahu asi 18000 hesel.

3. **Lingea Grammaticon** – kromě kontroly překlepů v textu je schopen vyhledat chyby ve shodě podmětu s přísudkem, ve jmenné skupině, chybnou interpunkci, nesprávné použití předložek či zájmen, chybějící nebo nadbytečné čárky ve větách. Navíc dokáže odhalit prohřešky proti správnému stylu zvoleného dokumentu, jako použití první osoby v technických textech, zmnožení větných členů, opakování slov ve větě, přítomnost slovesa ve větě a oslovení v dopise velkým písmenem.

4. **Google Drive** – online nástroj, který nabízí podporu mnoha typů dokumentů (doc, odt, pdf, . . . ). Jeho hlavní výhodou je právě to, že je online a proto umožňuje snadné sdílení napříč uživateli. Dále nabízí možnost verzování úprav, vkládání komentářů na libovolné úseky textu atd.

5. **Texmaker, LyX, TeXnicCenter, . . .** – většina z TeXových editorů v dnešní době již podporuje základní zvýraznění překlepů a na rozdíl od běžných editorů, dokáží při překladu upozornit na chyby, jako jsou například přetékající obrázky a texty. Hlavní nevýhodou je, že práce s nimi vyžaduje více zkušeností a slovníky pro automatickou kontrolu pravopisu bývají jen anglické a velice jednoduché. Editor LyX patřící do rodiny WYSIWYG editorů pro TeX tak může být optimálním řešením pro běžné uživatele.

(2)

Největším nedostatkem editorů jako jsou Word či Writer a jiných WYSIWYG editorů je nedodržování mnoha typografických pravidel, na které jsou naopak zaměřeny

TEXové editory. Na druhou stranu editace a formátování textů v těchto editorech probíhá ve formě zdrojového kódu a vyžaduje určitou dávku zkušeností.

### 2.1.2 Korektor

Korektor je osoba zodpovědná za chyby gramatiky, pravopisu a stylistiky v textech jiných autorů, pracující zpravidla v redakcích novin, vydavatelských domech nebo internetových zpravodajských serverech. Zodpovědností korektora je navíc i kontrola nesrovnalostí výtisku sazby a rukopisu a také odborná stránka kontrolovaného textu, tj. vhodnost použití cizích slov, technických výrazů, správná citace zdrojů a mnoho jiných. (14)

### 2.1.3 Gramatika

Gramatika neboli mluvnice je v lingvistice soubor logických a strukturálních pravidel, které udávají správnou stavbu vět, větných členů a slov v daném přirozeném jazyce. Stejným způsobem je pojmenováno i studium těchto předpisů, které se skládá z nauky o tvarosloví (morfologii), skladby (syntax), fonetiky a fonologie zkoumajících zvukovou stránku jazyka, sémantiky a pragmatiky. (5)

Každý jazyk včetně dialektů má svou vlastní gramatiku.

### 2.1.4 Pravopis

Ortografie neboli pravopis, je ustálený způsob záznamu zvukové podoby přirozeného spisovného jazyka za pomoci grafických znaků. Zpravidla bývá dán příslušnou státní nebo vědeckou institucí, popřípadě odborníky či používáním významnou skupinou uživatelů daného jazyka. V České republice je to například Ústav pro jazyk český (ÚJČ) Akademie věd. Proti pravidlům pravopisu je nejen psaní s chybami, ale například i použití pravopisu jiného jazyka.

Termíny pravopisná a gramatická chyba bývají často zaměňovány. Gramatickou chybou je chyba odporující systému jazyka a stává se především cizincům („On studovala na ČZU v Praze“). Pravopisná chyba je chyba proti pravidlům pravopisu – shoda podmětu s přísudkem, interpunkci, psaní mě/mně, příčestích apod. (30)

### 2.1.5 Stylistika

Stylistika neboli nauka o slohu, je jazykovědná disciplína pojednávající o jeho podstatě, jeho druzích, o slohové výstavbě jazykových projevů (textů) a o slohovém rozvrstvení jazykových prostředků.

Zkoumá hlavně způsob výstavby jazykového projevu (komunikátu), tj. způsob zpracování obsahu a využití jazykových prostředků (výběr, uspořádání v celek). Po stránce jazykové závisí slohová vytříbenost zejména na vhodném výběru slov, na plynulé stavbě vět a na jejich spojení v souvislý text.

Stylistika se zaměřuje na zkoumání subjektivních rozdílů mezi stylem psaní jednotlivých pisatelů. (19)

## 2.2 Typografie

Tento umělecko-technický obor se zaměřuje na vizuální stránku obsahu. Zkoumá především písmo, jeho správný výběr, užití a sazbu. Cílem je snadno čitelný a dobře vnímatelný text, který má jednoznačný význam.

### 2.2.1 Základní typografická pravidla

Už při psaní textu je vhodné dodržovat základní typografická pravidla, aby byl výsledný text dobře čitelný a měl správnou sazbu. Mezi tato pravidla patří:

- Volba písma – správná volba písma ovlivňuje zejména čitelnost textu. U kratších textů, například v prezentacích, je možné použít bezpatková písma (např. Arial). Naopak u delších je vhodné použít písma patková z toho důvodu, že patky vedou oči čtenáře po řádku a podporují spojité myšlení.
- Mezery – mezi slovy je vždy maximálně jedna mezera, nikdy více (odsazení a sloupce vytváříme za pomoci jiných nástrojů, jako jsou zarážky, tabulky a tabulátory). Mezislovní mezery se tedy píší jedním stiskem mezerníku. Jiným případem jsou mezislovní nedělitelné mezery (též zvané nezlomitelné, tvrdé nebo podle normativního názvosloví nepřerušující). Nedělitelná mezera zaručuje, že v daném místě nenastane řádkový zlom a užívá se v následujících případech:
  1. mezi číslem a jednotkou (1 m);
  2. jako znak oddělující skupiny číslic (25 100);
  3. ve vícedílných zkratkách (s. r. o.);
  4. před pomlčkou oddělenou mezerami;
  5. v datech;
  6. při spojení číslovky následující po výrazu (Sušice 1146);
  7. mezi jednopísmennou neslabičnou předložkou a následujícím slovem.

- Vodorovné čárky – spojovník bez mezer, pomlčka „–“ větná s mezerami, rozsahová bez mezer; u znaménka mínus je rozdíl především v umístění, kdy symbol mínus je výše než spojovník nebo pomlčka. Odděluje se mezerami.
- Větná interpunkce – slouží k členění vět a souvětí. Tečky se nepoužívají za nadpisy a popisky. Před znaky .,:;!? se nepíší mezery, za nimi ano. Výjimkou jsou jen shluky těchto znaků.
- Uvozovky – existuje více druhů, ale vždy využíváme ty, jaké jsou dané pro jazyk textu. V českých dokumentech se používají zásadně české uvozovky („“).
- Závorky – při psaném textu se využívají kulaté závorky, jejich obsah není oddělen mezerami.
- Výpustka – samotný znak tří teček. Využívá se při nedokončení věty či myšlenky. Píše se bez mezery mezi nedokončenou větou. Pokud je využita jako pokračování výčtu, pak je mezerou oddělena.
- Procenta, stupně a jednotky – má-li zápis význam jednoho slova, pak se píší bez mezery za číslem. Pokud znamená dvě slova, pak je oddělena nedělitelnou mezerou:

  10% – „desetiprocentní“ vs. 10 % – „deset procent“.

- Lomítka – obyčejně se sází bez mezer, ovšem pokud je využito ve smyslu výčtu, pak se jednotlivé položky oddělují lomítkem s mezerami na obou stranách.
- Slovo „viz“ – jedná se české slovo (významově sloveso „podívej se“), nikoliv zkratku, a proto se za ním nepíše tečka.
- Násobení, paragraf a ampresand – pro znak násobení se správně nepoužívá malé písmeno „x“ ale symbol ×. Paragraf „§“, využívající se v právnických textech, se používá zásadně s číslem odděleným nedělitelnou mezerou. Znak „&“ se využívá jako spojka ve smyslu „a“ a je obklopena nedělitelnými mezerami.
- Datum a čas – za řadovou číslovkou data se píše čárka a nedělitelná mezera, rozsah se odděluje pomocí pomlčky bez mezer (jednoslovné výrazy) nebo s mezerami (víceslovné výrazy). Hodiny a minuty se oddělují buďto dvojtečkou (dle ČSN 01 6910) nebo tečkou (PČP), minuty a vteřiny dvojtečkou, desetiny a setiny vteřiny desetinnou čárkou.

Typografická pravidla se mohou samozřejmě lišit stát od státu, kde mohou být upravovány různými normami. (14)

Nutno podotknout, že uvedená pravidla mají charakter spíše doporučení a nemají žádnou oporu v legislativě. V současnosti v České republice neplatí žádná norma, zabývající se pravidly typografie. Poslední platnou normou zaměřenou na typografii byla ON 88 2503:1974, která byla k 1. 1. 1994 zrušena. Existující norma ČSN 01 6910:2007 se zabývá úpravou písemností textovými editory, nikoliv typografií. (29)

Typografická pravidla by měla být dodržována i na internetu. Bohužel ani moderní internetové prohlížeče nepodporují všechna typografická pravidla, například dělení slov. Z tohoto důvodu nelze použít zarovnávání do bloku u užších sloupců.

## 2.3 Polygrafie

Polygrafie je výrobní obor, který zpracovává a tiskem rozmnožuje textové a obrazové předlohy. Zahrnuje tyto základní výrobní fáze:

1. zhotovování tiskových forem;
2. tisk;
3. dokončovací výroba a přidružené odborné činnosti.

Výsledkem činnosti polygrafického průmyslu jsou polygrafické výrobky. [1]

Polygrafickými výrobky jsou periodický tisk (noviny, časopisy), neperiodické publikace (knihy, učebnice, mapy, atlasy, atd.) a ostatní výrobky, tzv. merkantilie (katalogy, prospekty, kalendáře, pohlednice, ...).

Oproti typografii má polygrafie v české legislativě zastoupení, polygrafický průmysl je definován třídou ČSN 88. Většina z jednotlivých norem je ale již také bez náhrady zrušena nebo je jen českou verzí mezinárodní normy ISO. (29)

## 2.4 Normostrana

Normostrana neboli NS, je standardizovaná strana „strojopisného“ textu o určitém počtu znaků. V českém prostředí je její délka 1800 znaků včetně mezer, což odpovídá třiceti řádkům o šedesáti znacích a přibližně 250 slovům běžného textu.

[1] Dle normy ON 88 0128 (29)

Vzhledem k tomu, že dnešní textové editory podporují mnoho druhů písem, většina tisknutých stran nemá parametry normostrany. To by vyžadovalo použití konkrétního druhu písma, například Courier New a další nastavení.

V dnešní době se tedy pojem normostrana používá především jako pomůcka pro přepočet a stanovení odměny za určitou činnost aplikovatelnou na normostranu. Jde například o cenu za opis textu, překlad či korekturu. Počet normostran se tedy vypočítá jednoduše pomocí vzorce 2.1.

$$\sum NS = \frac{\sum n}{1800}, \tag{2.1}$$

kde $n$ je počet znaků včetně mezer.

Normostrany se samozřejmě mohou lišit v různých zemích dle místní legislativy či tradice, stejně jako je tomu u polygrafie. (6)

## 2.5 Rešerše existujících řešení

Ačkoliv nové technologie nabízí pokročilejší funkce, stále se na českém internetu nenachází aplikace, která by se specializovala na online podporu korektur textu. Existuje mnoho online nástrojů poskytujících elementární kontrolu pravopisu a typografie. Tyto pomůcky ale neslouží ke komplexní analýze textu. Pro nalezení existujících řešení bylo využito nejpoužívanějších českých vyhledávačů dle studie společnosti Effectix publikované na serveru zive.cz (31) a to konkrétně Google (.cz, .com) a Seznam.cz. Pro vyhledání relevantních výsledků byla využita klíčová slova a jejich kombinace, např. „Korektury online“.

Valnou většinu z relevantních výsledků hledání tvoří webové stránky společností nabízejících korektorské práce v mnoha úrovních. Návštěvníkům jsou nabízeny různé služby (kontrola typografie, stylistiky, gramatiky, expresní vyhotovení), které jsou podloženy neověřenými referencemi. Stránky budou hodnoceny z hlediska kvality (validita, uživatelská přívětivost, obsah), šíře nabízených služeb a důvěryhodnosti (uvedení podložených referencí a kontaktů).

### Očeštině.cz

Webová stránka nejednoznačného zaměření, nabízející jazykové korektury včetně online kalkulace a informace týkající se českého jazyka (testy, informace o maturitách z ČJ,

gramatická pravidla, doučování, ...). V době provádění rešerše byl na webu vyhlášený dlouhodobý „stop-stav“ přijímání nových zakázek. Téměř čtvrtinu obsahu webu zabírají zpětné odkazy na nesouvisející webové stránky.

### Českyhezky.cz

Na severu www.ceskyhezky.cz má návštěvník k dispozici online kalkulaci na základě nahraného souboru. Je zde nabízena možnost gramatické i stylistické korektury českých textů. Pro výpočet kalkulace je nutné nahrát soubor s textem. Uživatel však nemá možnost zjistit, kde je nahraný soubor uložen a jak je s ním dále nakládáno z hlediska autorských práv. Na stránkách se nenachází prohlášení, jakým způsobem jsou autorská práva chráněna. Tento problém je bohužel u většiny ze zmiňovaných webů. Dalším problémem, který se netýká pouze tohoto projektu, je nevalidita stránek a nedodržování zásad přístupného webu, o čemž svědčí především na první pohled textové menu, které je realizováno pomocí obrázků neopatřených atributy title ani alt.

### Liteera.cz

Zajímavý nekomerční projekt původně vyvíjený jako bakalářská práce studenta Jakuba Fialy na Fakultě informatiky Masarykovy univerzity v Brně. Jedná se o online typografický korektor česky psaných elektronických textů a slovník interpunkčních znaků a symbolů včetně jejich typografických zásad. Systém bohužel nenabízí automatickou kontrolu gramatiky analyzovaného textu – ačkoliv by šlo využít některého z hotových řešení. Pro základní otestování typografické korektnosti je systém dostatečný. Stejně jako předchozí systémy nenabízí možnost nabídky a poptávky. Oproti ostatním projektům nabízí povedené, uživatelsky přívětivé prostředí a validní HTML kód.

### Korektury-online.cz

Webová prezentace těžící především z vhodně zvolené domény. Doména je bohužel jedinou věcí, která je na tomto webu správně. Kvalitu snižují především zastaralé tabulkové rozložení a nevalidní kód. Na webu jsou prezentovány nabízené služby včetně ceníku a podmínek. Objednávku je možné zaslat pomocí e-mailu.

### Opravitchyby.cz

Jednoduchá webová prezentace nabízející korektorské a redaktorské práce. Stránky jsou postaveny na redakčním systému Wordpress a je využita responzivní šablona, které lze

vytknout jen několik maličkostí. Poptávající má možnost využít také okamžité odeslání objednávky.

Nedostatkem, který vzbuzuje nedůvěru vůči osobě provádějící korekturu, je neuvedení referencí. Nabízena je také neurčitá sleva pro studenty a dále možnost „zkrocení“ citací za nedefinovaný „mírný“ příplatek. Akceptovány jsou pouze dokumenty editorů MS Word a Open Office Write a soubory PDF, jejichž zpracování je o deset korun za normostranu dražší.

### Proofreading.cz

Komplexní, přístupná a uživatelsky přívětivá webová prezentace, nabízející jazykovou a stylistickou korekturu, dále služby copywritingu, tvorby názvů a zpracování informačních analýz a rešerší. Chybí nabídka odborné a předtiskové korektury. Na webu jsou k dispozici i reference, uživatel ale nemá možnost zjistit, jaká ze služeb byla dané společnosti poskytnuta a s jakým výsledkem. Ceník je podrobný a přehledný. V sekci kontaktů jsou uvedena jména, včetně adres a e-mailů, což výrazně zvyšuje důvěryhodnost.

Pokud korektoři neodhalí některou z chyb, je dána záruka vrácení částky za normostranu, která byla zaplacena. Tato záruka je platná pouze jeden týden, což je ve většině případů nedostačující.

Web neobsahuje pouze nabídku služeb, ale i články týkající se korektur, českého jazyka a podobných.

### Grammarly.com (anglicky)

Webová služba nabízející online kontrolu gramatiky, plagiátorství, interpunkce aj. Pro zobrazení výsledků kontroly je požadována placená registrace.

### Závěr

Největšími nedostatky uvedených webů, přestože se jedná o společnosti nabízející korektury, jsou:

- Nesplňují kritéria ani nejnižší úrovně A dle metodiky WCAG 2.0 o přístupnosti webu a mají celkově nepřívětivé uživatelské rozhraní.
- Nevalidní HTML kód.
- Zastaralé nebo nevyhovující uživatelské prostředí (například tabulkový layout).

- Některé z prezentací nesplňují základní typografická pravidla.
- Chybějící reference a obchodní podmínky.
- Krátkodobé záruky bez postihů.

Na internetu existuje velké množství nabídek korektorských prací – jedná se především o nabídky fyzických osob. Tyto služby nenabízejí žádnou formu propojení nabídek a poptávek a jde především o jednoduché webové prezentace s kontakty bez pokročilejší možnosti interakce s uživateli.

Další možností pro člověka poptávajícího či nabízejícího korektorské práce je využití různých diskusních fór nebo inzerátů. U obou druhů nabízených služeb chybí uživatelům především reference, bližší kontaktní údaje a většinou i obchodní podmínky. Uživatel poptávající korekturu je tedy odkázán na důvěru ve společnost či jednotlivce, který tuto službu nabízí. Dojde-li pak k odhalení chyb korektury, musí být nahlášeny v rámci několika dní, maximálně týdnů a většina z korektorů nabízí jen minimální finanční odškodnění.

Jediným řešením pro český jazyk, které se snaží nabídnout možnost online korektury, je projekt liteera.cz. Jak již bylo řečeno, byl vyvíjen jako bakalářská práce a zdá se, že další rozvoj tohoto projektu je v nedohlednu.

Kapitola 3

# Vlastní řešení

## 3.1 Analýza a návrh aplikace

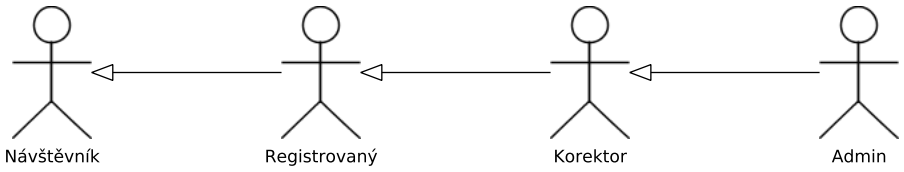
Cílem této části je podrobná specifikace a následná analýza požadavků – funkčních i nefunkčních. Chování systému bude popsáno za pomoci diagramů případů užití a dalších UML nástrojů. (25)

### 3.1.1 Aktéři

Vytvořená aplikace umožňuje rozdělení uživatelů do následujících rolí:

- Návštěvník – výchozí role všech návštěvníků webových stránek bez možnosti přihlášení a s přístupem pouze k veřejné části webu;
- Registrovaný uživatel – autentizovaný a autorizovaný uživatel oprávněný využívat příslušné funkce aplikace, v aplikaci bude vystupovat především jako zadavatel;
- Korektor – specializovaný přihlášený uživatel;
- Administrátor – pověřený přihlášený uživatel s maximálními možnými právy.

Jednotlivé role získávají práva podřízených, jak zachycuje diagram 3.1.




Obrázek 3.1: Aktéři systému

### 3.1.2 Požadavky

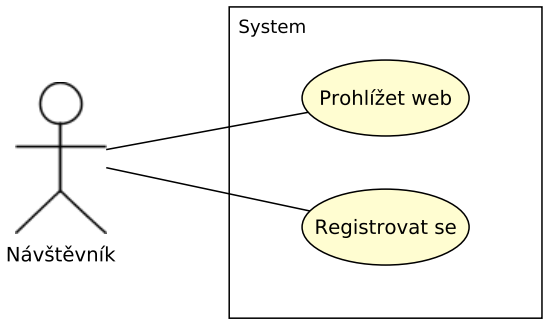
Pro správný návrh aplikace je nutné definovat požadavky na systém, které vycházejí z potřeb korektorů a zadavatelů. Požadavky jsou seřazeny chronologicky podle důležitosti v systému.

#### 3.1.2.1 Funkční požadavky

Nepřihlášený uživatel může libovolně procházet webové stránky, přečíst si podmínky i další veřejně přístupné texty. Dále může využít možnost bezplatné registrace 3.2.

Pokud uživatel není zaregistrovaný, má možnost využít maximálně jednoduché registrace, kde je vyžadováno zadání jedinečné přezdívky (přihlašovacího jména) a e-mailu, který slouží k následné autorizaci (ověření) uživatele a je především základním komunikačním kanálem uživatele se systémem. Pro odlišení skutečného uživatele od robota bude využit mechanizmus CAPTCHA či ověřovací otázka, ale to až v případě, pokud se takové útoky objeví. Pokud ne, není nutné uživatele zatěžovat dalším zadáváním.

Přihlašovací formulář vyžaduje zadání unikátního přihlašovacího jména a hesla. V případě selhání autentizace je uživateli nabídnuta možnost obnovy hesla.



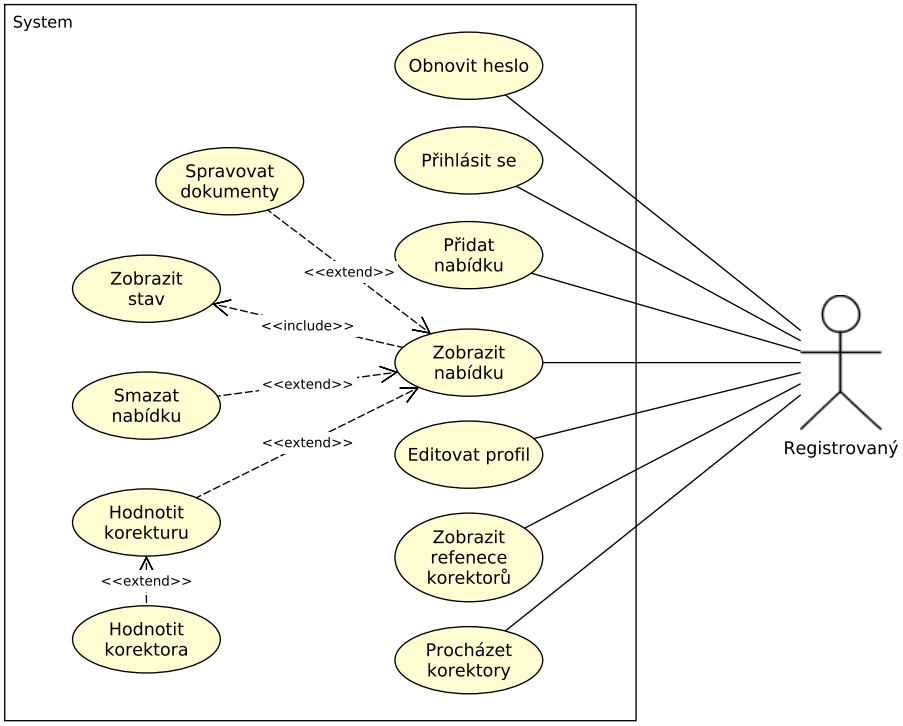

Obrázek 3.2: Diagram užití – návštěvník

Registrovaný přihlášený uživatel má možnost zakládat nové nabídky – tzn. vytvořit novou nabídku s popisem, termíny a požadavky, ke které připojí související dokumenty. Zároveň má možnost své, již vytvořené nabídky, procházet, upravovat a mazat. Součástí úprav je správa souvisejících dokumentů (přidávání, mazání), změna stavu aj. Navíc má zadavatel možnost komunikace s korektorem zpracovávajícím jeho nabídky.

Registrovaný, ale nepřihlášený uživatel, může využít možnosti přihlásit se, eventuálně si nechat zaslat nové přístupové heslo, pokud jej zapomněl. Po přihlášení má uživatel samozřejmě možnost odhlásit se ze systému. Další nezbytnou funkcí je správa

vlastního profilu. Zde uživatel vyplňuje základní identifikační a kontaktní údaje. Jediným neměnným údajem je přihlašovací jméno – e-mail. Kontaktní e-mail lze změnit.

Přihlášený uživatel má dále možnost zobrazit a procházet reference korektorů registrovaných v systému. Nemůže měnit hodnocení korektora, pokud mu nezadával práci. Diagram užití registrovaného uživatele je na obrázku 3.3.

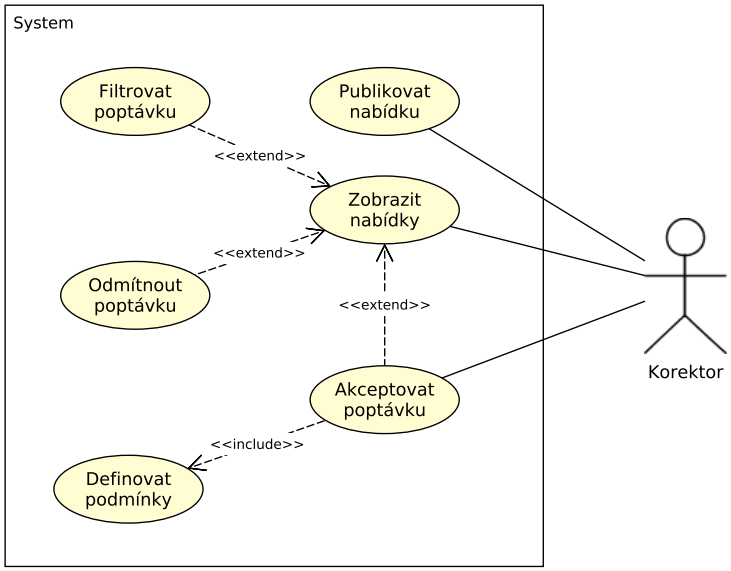



Obrázek 3.3: Diagram užití – registrovaný

Korektor je specializací registrovaného uživatele. Sám může vystupovat jako zadavatel a navíc má možnosti publikovat nabídku korektorských prací a zobrazit si aktuálně publikované poptávky registrovaných uživatelů. Poptávky si může korektor filtrovat dle určitých kritérií (náročnost, druh korektury, termín, ...). Pokud jej osloví poptávající, může poptávku akceptovat spolu s definicí podmínek, za kterých je poptávku ochoten přijmout či ji může odmítnout.

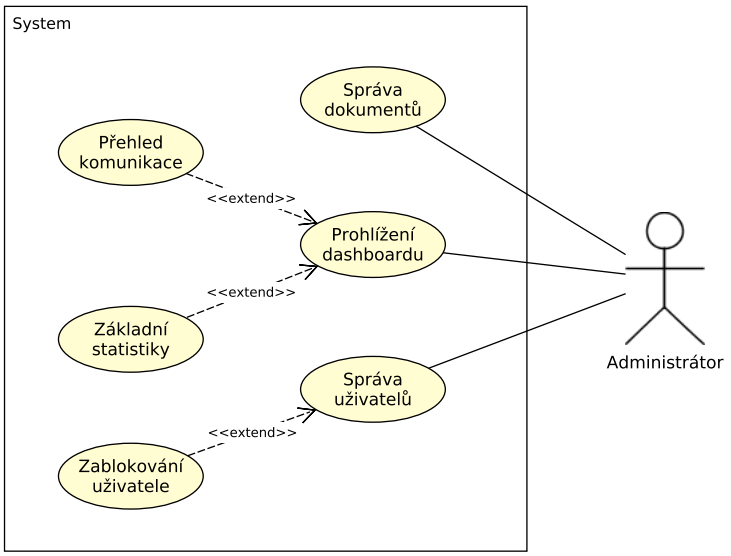
Stejně jako registrovaný uživatel může kontaktovat korektora, tak i korektor může kontaktovat uživatele, jehož dokumenty zpracovává. Veškerá komunikace je zaznamenávána. Diagram užití korektora je na obrázku 3.4.

Administrátor má veškerou kontrolu a přehled nad děním v systému. Může mazat a přidávat dokumenty. Dále má možnost správy uživatelů včetně možnosti zablokování uživatelského účtu například pro případ porušení podmínek.




Obrázek 3.4: Diagram užití – korektor

Administrátor má k dispozici také takzvaný dashboard. Jedná se o stránku s přehledem a základními statistikami systému. Cílem tohoto nástroje je možnost rychlého zobrazení dění v systému a celkového přehledu. Uživatelů s právy administrátora musí být minimum, aby se zamezilo možnosti zneužití dat. Diagram užití administrátora je znázorněn na obrázku 3.5.




Obrázek 3.5: Diagram užití – administrátor

#### 3.1.2.2 Nefunkční požadavky

Nefunkční neboli doplňkové požadavky systému jsou vlastnosti a omezení služeb, jenž systém poskytuje. Tyto požadavky budou vztaženy na systém jako celek. (8)

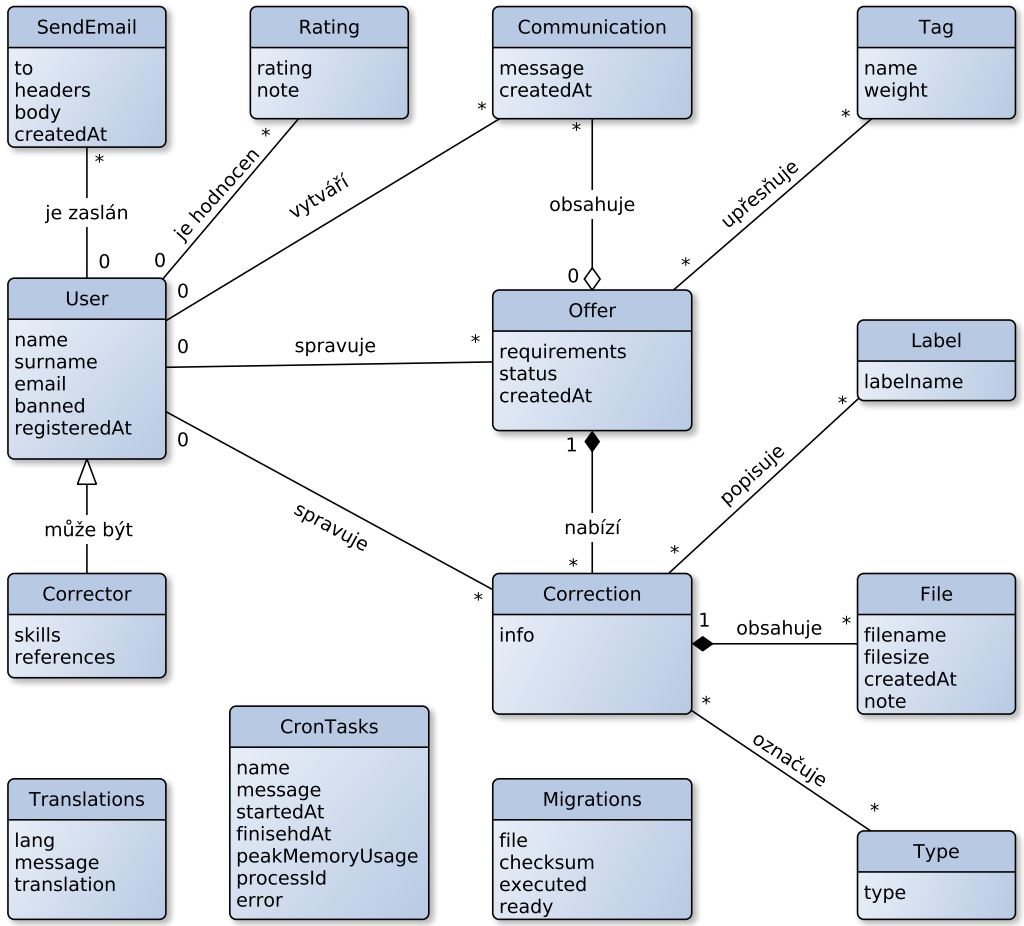
1. Dostupnost z webového prohlížeče – systém musí být dostupný ze všech moderních webových prohlížečů. Vzhled se se musí přizpůsobit zařízení.
2. Dostupnost určitých funkcí přes API – systém musí nabízet rozhraní pro další možné platformy v podobě API.
3. Snadná udržovatelnost a rychlý vývoj – systém musí být intuitivní a umožňovat rychlou naučitelnost i pro nově příchozí vývojáře.
4. Možnost využití různých databázových systémů – systém musí být schopen adaptovat se na změnu databázového systému.
5. Rozšiřitelný a modifikovatelný – systém musí být naprogramován tak, aby bylo možné doplňovat novou funkcionalitu nebo upravovat stávající bez toho, aby byl ovlivněn existující systém.
6. Bezpečný – systém musí být dostatečně zabezpečen za pomoci autentizace a autorizace uživatelů, důvěrná data musí být při přenosu a uchovávání šifrována. Důležitá je také odolnost proti útokům.

### 3.1.3 Doménový model

Doménový model je formou diagramu tříd, který se vytváří spolu s diagramy případů užití v rané fázi vývoje. Doménový model je nezávislý na platformě a oproti diagramu tříd i značně zjednodušený – jedná se o formu náčrtu základních entit a vztahů mezi nimi. Entity jsou definovány pouze nejdůležitějšími atributy. (1) Doménový model je zachycen na obrázku 3.6.

### 3.1.4 Struktura

Systém je rozdělen do tří základních částí. Veřejně přístupná část aplikace (Front), společná administrace (Admin) pro registrované, korektory i administrátory a API. Pro sekce Front a Admin je nutné vytvořit uživatelské rozhraní a definovat strukturu stránek a podstránek. Struktura stránek je zobrazena pomocí jednoduchých grafických znázornění, takzvaných mindmap.



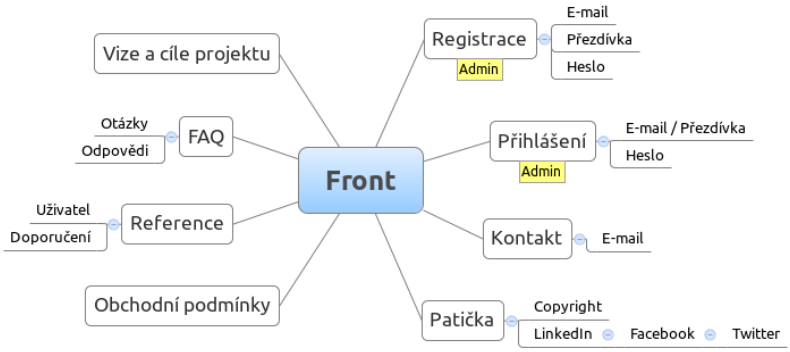

Obrázek 3.6: Doménový model

#### 3.1.4.1 Front

Veřejně přístupná část (Front), která je přístupná každému návštěvníkovi webu, má za cíl upoutat pozornost a poskytnout nezbytné informace, mezi které patří:

- vize a cíle projektu;
- často kladené dotazy (FAQ);
- reference;
- obchodní podmínky;
- kontakt.

Tato část aplikace bude realizována formou single-page, což znamená, že všechny tyto informace budou na jedné stránce. Pouze možnost registrace a přihlášení bude již přesměrovávat na samostatné stránky administrace.




Obrázek 3.7: Struktura stránek – Front

### 3.1.4.2 Admin

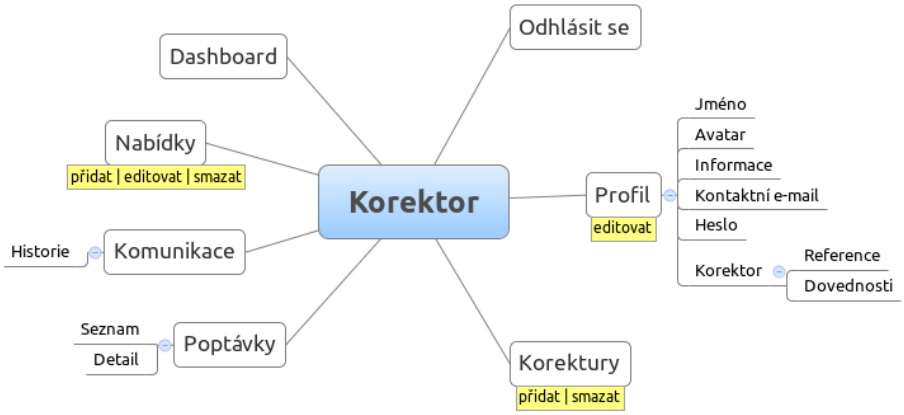
Administrace bude mít jednotné uživatelské rozhraní. Položky v menu a funkce budou dostupné uživatelům dle jejich přístupových práv. Administrátor bude mít například k dispozici celý seznam registrovaných uživatelů, oproti tomu, registrovaní uživatelé budou mít pouze seznam korektorů, kteří jsou momentálně k dispozici. Jednotlivé mindmapy jsou zachyceny na obrázcích 3.8, 3.8, 3.9 a 3.10.


Obrázek 3.8: Struktura stránek – registrovaný uživatel

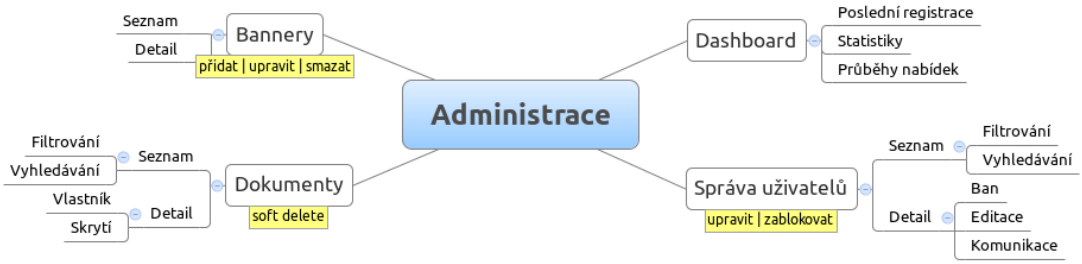
Administrátor bude mít možnost spravovat reklamní plochy umístěné v administraci, které se zobrazují registrovaným uživatelům. Tyto reklamy vkládá administrátor osobně na základě požadavků od inzerentů, se kterými komunikuje prostřednictvím jiných kanálů (e-mail, telefon, ...).

### 3.1.4.3 API

Jednou z nezbytných funkcí webové aplikace je nabízet rozhraní i pro jiné technologie. Tento požadavek narůstá spolu s rozmachem mobilních aplikací, kdy je výhodné poskytnout rozhraní i pro aplikace třetích stran a tím nabídnout možnost využívat sys-




Obrázek 3.9: Struktura stránek – korektor




Obrázek 3.10: Struktura stránek – Administrátor

tém lidem i pomocí aplikací třetích stran. API bude postaveno na architektuře REST. Komunikace probíhá pomocí HTTP požadavků a odpovědí mezi zařízením a serverem. Pro výměnu dat slouží formát JSON, který je vhodnější pro toto použití než formát XML obsahující navíc značky, které zvětšují velikost přenášených dat. (11)

Struktura URL požadavku vypadá následovně:

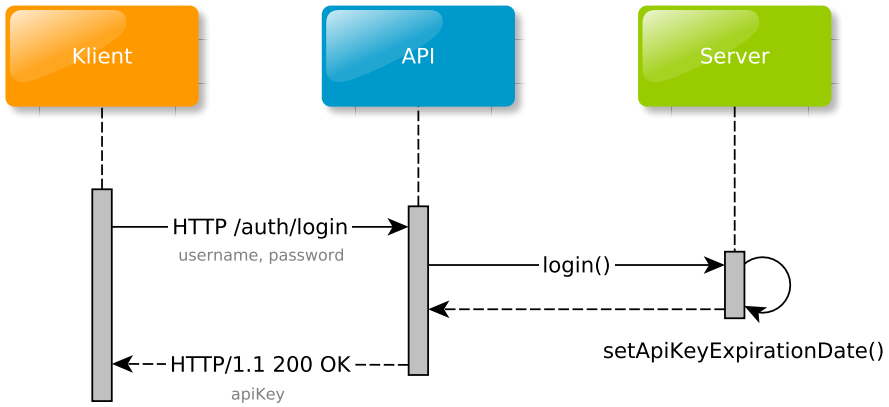
https://www.example.com/api/:verze_api:/:oblast:/:akce:

a konkrétní požadavek například pro odhlášení může vypadat takto:

https://www.example.com/api/1/auth/logout

Prvotním požadavkem komunikace s API je nutnost autentizace uživatele. Pokud se podaří uživatele identifikovat na základě zaslaných parametrů username (či e-mail) a password, pak je uživateli zpět zaslána odpověď obsahující klíč k API, který dále identifikuje přihlášeného uživatele při dalších požadavcích. Tento autorizační klíč má navíc nastavenou expirační dobu. Pokud uživatel při dalším požadavku nezašle X-Api-Key hlavičku (nebo jeho platnost již vypršela), pak není možné uživateli zaslat

vyžadované informace a je informován o problému formou HTTP odpovědi s patřičným chybovým kódem. Průběh přihlášení je znázorněn graficky na obrázku 3.11.




Obrázek 3.11: API – autentizace uživatele

Ukázka HTTP požadavku pro přihlášení uživatele:

**POST** /api/auth/login HTTP/1.1
**Accept:** application/json
**Accept-Language:** cs-CZ,cs;q=0.9,en;q=0.8
**X-Region:** CZ
**X-UUID:** 103d6d50761811e2bcfd0800200c9a66
**X-Api-Version:** 1

{"data":{"username":"test","password":"testuser"}}

Odpověď:

**HTTP/1.1 200 OK**
**Date:** Sun, 5 Oct 2013 13:22:16 GMT
**Server:** Apache/2.4.6 (Ubuntu)
**Cache-Control:** s-maxage=0, max-age=0, must-revalidate
**Expires:** Mon, 23 Jan 1978 10:00:00 GMT
**Content-Length:** 54
**Content-Type:** application/json

{"data":{"apiKey":"fz2o5mvll29tijcw9o9ygmfmhu9le67v"}}

Stejně tak jako celý web, bude i API v průběhu vývoje procházet různými změnami. Je tedy nutné upozorňovat uživatele na změnu. Proto uživatel při každém dotazu musí uvádět verzi, které se dotazuje. Vzhledem ke změnám, které mohou měnit funkcionalitu některých dat, je definována i minimální podporovaná verze API. Pokud se uživatel dotazuje nepodporované verze, je o tom náležitě informován chybovým kódem.

Mezi základní funkce, které bude API v první verzi nabízet, patří především operace umožňující autentizaci uživatele, úpravy profilu aj. Dále pak funkce pro jednoduché získávání dat z aplikace týkající se korektur, nabídek a s nimi souvisejících komponent. Počet těchto funkcí bude pravděpodobně s časem a požadavky uživatelů narůstat.

Kompletní dokumentace všech funkcí bude vedena na serveru Apiary.io.

Všechny příchozí API požadavky jsou zaznamenávány do souborového logu aplikace a zároveň do databáze, kde je k požadavku zaznamenána i hlavička odpovědi. Tyto logy budou sloužit především pro odhalování možných chyb a pro zpracování dotazů přímo od uživatelů API.

### 3.1.5 Uživatelské rozhraní

S rozmachem mobilních zařízení vzrůstá také fragmentace velikosti displeje, na kterém uživatelé zobrazují webové stránky. A zde nastává problém, jak efektivně prezentovat data například i na malých displejích. Tímto se zabývá responzivní web design (více v kapitole věnující se použitým technologiím 3.2.2). Toto bude nutné zohlednit i v návrhu uživatelského rozhraní jak administrace, tak i veřejně přístupné části webu.

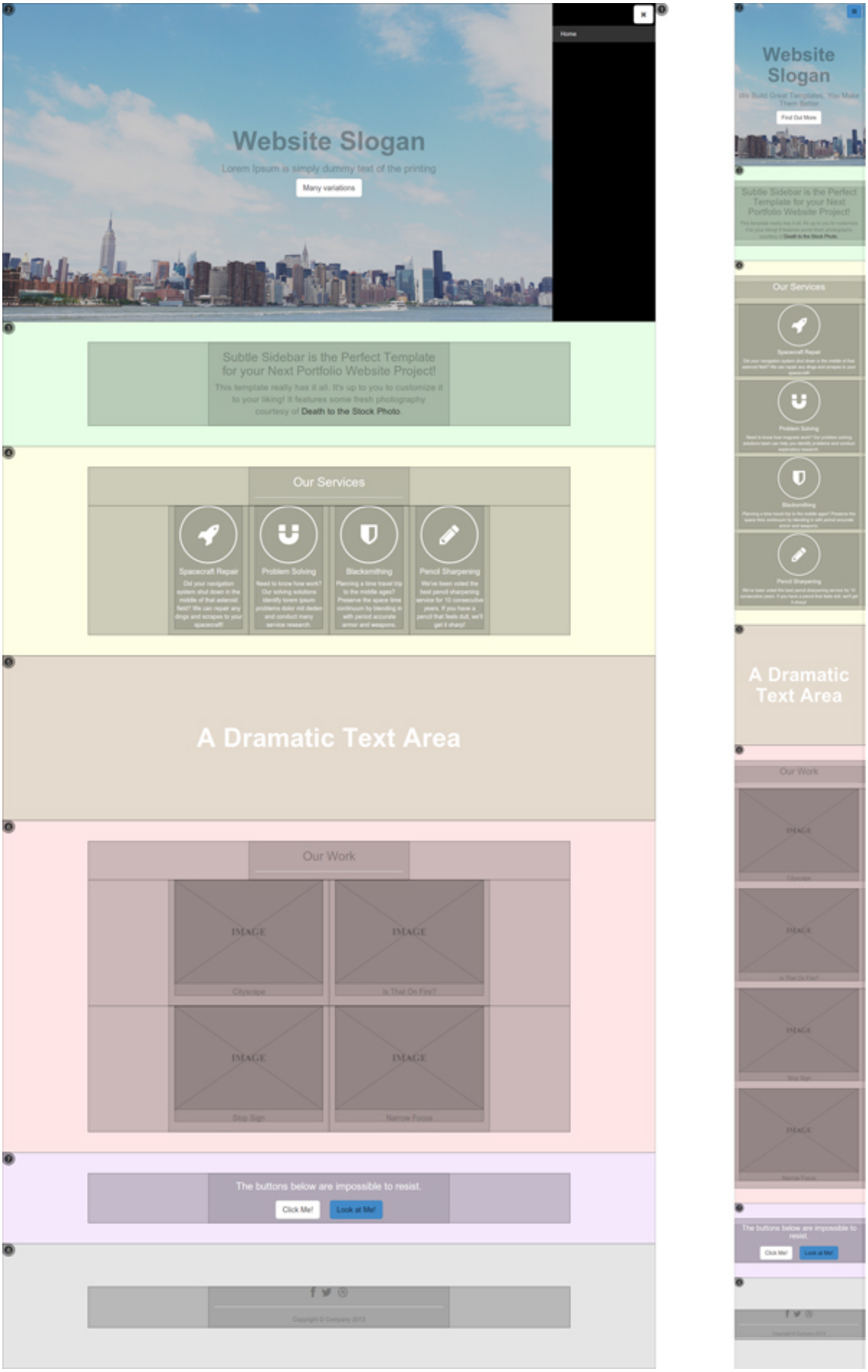
#### Front

Jak již bylo zmíněno, veřejně přístupná část bude realizována pomocí jednoduchého SPI – single-page Interface. Cílem hlavní stránky je především zaujmout návštěvníka a poskytnout mu nejdůležitější informace. Vzhledem k tomu, že hlavní stránka bude obsahovat všechny důležité informace, není nutné web dále strukturovat.

Ukázka layoutu včetně reakce na zobrazení na mobilním zařízení s menším rozlišením je znázorněna na obrázku 3.12.

#### Admin

Pro administrační rozhraní bude vhodné vybrat některé z již hotových řešení, které bude možné upravit přesně pro potřeby administrace tohoto systému, viz 3.13. Administrace musí uživateli zprostředkovat informace co možná nejpřívětivěji a nejsrozumitelněji. K tomu poslouží rozložení s hlavní obsahovou částí, která je doplněna o hlavní menu v levé části a hlavičku s upozorněními a uživatelským nastavením. V administraci bude důležité dbát na efektivní přesuny a změnu velikosti jednotlivých elementů uživatelského rozhraní v závislosti na rozměrech zařízení.




Obrázek 3.12: Grafické rozvržení – Front


Obrázek 3.13: Grafické rozvržení – Administrace

### 3.1.6 SEO

Optimalizace webových stránek pro vyhledávače je základním požadavkem moderních webových prezentací. Tato metodika pro úpravu obsahu a formy webové prezentace má za cíl dosažení co nejlepší pozice v internetovém vyhledávači vzhledem k jejímu zaměření na cílové návštěvníky. Je mnoho způsobů, jak získat co nejlepší pozici. Ve vyvíjeném systému se zaměříme především na základní metody.

- Obsah – obsah stránky by měl být kvalitní – přiměřeně dlouhý, správně strukturovaný a zvýrazněný a odpovídající zaměření webu. Netextové elementy musí mít alternativní obsah.
- Validní – stránka by měla být validní dle standardů HTML či XHTML, tzn. sémanticky i syntakticky správné používání zápisu značek.
- Struktura – všechny nadpisy by měly být vhodně strukturovány, text rozdělen do odstavců.
- URL – je dobré používat Clean URL umožňující lépe využívat klíčová slova přímo v URL.
- Přístupný web – web je obecně lépe hodnocen, pokud splňuje pravidla přístupného webu.
- Další metody – kanonizační problém (.htaccess), budování zpětných odkazů, soubor robots.txt, meta tagy Description a Keywords, ...

Většinu z uvedených požadavků řeší použití správného obsahu a použité rozložení stránky.

### 3.1.7 Zabezpečení

Zabezpečení webové aplikace je jednou z nejdůležitějších součástí vývoje. V této části budou popsány a analyzovány nejčastější a nejnebezpečnější útoky, kterými je možné tento systém napadnout, a dále popsány postupy, jakými těmto útokům předejít nebo se jim bránit.

#### 3.1.7.1 Full Path Disclosure

Odhalení skutečné cesty ke skriptu nemusí být samo o sobě bezpečnostní chybou, dává však útočníkovi k dispozici velké množství informací, které mu útok značně zjednoduší.

Pokud se na produkčním webu zobrazí chybová hláška, varování nebo zpráva zobrazující například, že na určitém řádku daného skriptu nastal problém, dáváme tím útočníkovi mnoho užitečných informací, které by jinak složitě zjišťoval.

1. Jaký operační systém je na serveru použit (Unix, Windows);
2. technologie, které pohání web (např. php) včetně možného prozrazení konkrétní verze, ke které se poté dají jednoduše dohledat chyby zabezpečení či jiné bugy;
3. nastavení serveru (zapnuté zobrazování chybových hlášek);
4. zda je web postaven na nějakém frameworku či redakčním systému (opět, stejně jako u technologií, je snadné dohledat bezpečnostní chyby dané verze frameworku či CMS).

Nejzávažnějším prohřeškem je pak publikování celého výstupu funkce `phpinfo()` (typicky na example.com/info.php či example.com/phpinfo.php), které dává útočníkovi téměř veškeré informace o použitých knihovnách včetně verzí. Všechny tyto informace mohou být klíčem k úspěšnému útoku na server.

#### 3.1.7.2 Cross-Site Scripting

Cross-site scripting je metoda narušení webových stránek zneužívající bezpečnostní chyby ve skriptech, zejména neošetřené vstupy a výstupy. Útočník pak dokáže do stránky podstrčit svůj vlastní JavaScriptový kód, kterým může stránku pozměnit, poškodit nebo dokonce získat citlivé údaje o návštěvnících. Proti XSS se lze bránit jen důsledným a korektním ošetřením všech vstupních a výstupních proměnných.

Příkladem útoku může být podstrčení upravené URL uživateli, pomocí které injektujeme do stránky svůj kód. Pokud aplikace nebude vstupy a výstupy řádně ošetřovat, spustí se skript v prohlížeči uživatele. Tímto způsobem je například možné zcizit identitu. Jednoduchou ukázkou útoku je podstrčení JavaScriptového alertu, které samo o sobě nemusí být nebezpečné, ale pokud se podstrčený kód zobrazí na webu, bude to pro ostatní návštěvníky nepříjemné.

http://example.com/?search=<script>alert('Toto je XSS útok.');</script>

Účinnou obranou proti této formě útoku je především důsledné escapování a sanitizace dat.

#### 3.1.7.3 Escapování

Escapování je metoda pro surjekci znaků vyskytujících se v řetězci do znaků povolený v daném jazyku. Ecapování je striktně závislé na kontextu a je tedy nutné rozlišovat, zda je použito na HTML, JavaScript, SQL, CSS aj., a respektovat pravidla daného kontextu. Není proto možné definovat přesný postup pro obecné escapování znaků.

#### 3.1.7.4 Cross-Site Request Forgery

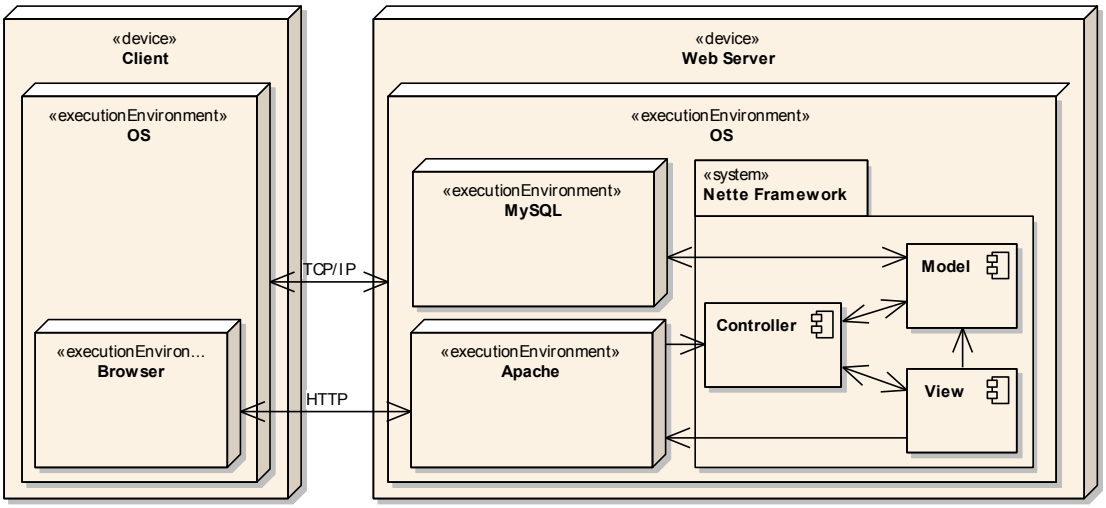
Jeden z nejhorších a nejhůř ošetřitelných útoků, jelikož se jedná o nedostatek v HTTP protokolu. Jakákoliv obrana proti tomuto útoku má většinou svou negativní stránku a řeší pouze jak obejít nedostatek HTTP protokolu. Cross-site request forgery je útok spočívající v tom, že přimějeme uživatele navštívit stránku, která skrytě vykoná útok na webovou aplikaci (pomocí HTTP požadavku), kde je uživatel zrovna přihlášen (má nastavenu cookie). Lze takto například pozměnit nebo smazat článek pod identitou jiného uživatele, aniž by si toho sám všiml. Proti útoku se lze bránit generováním a ověřováním autorizačního tokenu s určitou platností.

#### 3.1.7.5 Ukládání hesel

Hesla by měla být ukládána zahešovaná a to včetně náhodného řetězce, tzv. soli. Ideálním řešením je získávat hash pomocí funkce `crypt()`. Tato funkce je použitelná až od verze PHP 5.3.7, protože ve starších verzích obsahuje tato funkce bezpečnostní zranitelnost – problém nastává, pokud heslo obsahuje ne-ASCII znaky, pak je generovaný hash zranitelný. Od verze PHP 5.5 je navíc možné používat funkcí `password_hash()` a `password_verify()`.

### 3.1.8 Diagram nasazení

Tento diagram zobrazuje způsob, jakým bude rozmístěna architektura software na architekturu hardware. (1) Serverovou stranu zajišťuje Apache s PHP a databází MySQL spolu s Nette Framework.




Obrázek 3.14: Diagram nasazení

## 3.2 Použité technologie

Při výběru použitých technologií je nutné zohlednit nefunkční požadavky na systém.

### 3.2.1 LAMP

Základ aplikace bude využívat kombinace LAMP – operační systém Linux, softwarový webový server Apache2 s podporou skriptovacího jazyka PHP a databází MySQL. Požadované verze a nastavení jednotlivých součástí budou specifikovány v instalační příručce.

- Linux – jedná se o nejrozšířenější operační systém na webových serverech. Linux je založený na principu unixových systémů, které jsou snadno spravovatelné, rozšiřitelné a především volně šiřitelné.
- Apache2 – multiplatformní OSS softwarový webový server.

- MySQL – multiplatformní databázový systém, který je v současnosti vyvíjený společností Sun Microsystems. Je postaven na jazyce SQL a nabízí bezplatnou GPL licenci i placenou komerční variantu. (15)
- PHP – je skriptovací programovací jazyk běžící na straně serveru, využívaný především k tvorbě dynamických webových prezentací a aplikací. V našem případě je pro běh nutné využít verzi minimálně 5.3.7 a to zejména kvůli podpoře jmenných prostorů. Aplikaci by šlo upravit i pro nižší verze, bylo by ale nutné využít starších knihoven.

### 3.2.2 HTML, CSS, JS, Twitter Bootstrap

Systém by měl generovat stránky v podobě validních HTML dokumentů formátovaných pomocí kaskádových stylů a podporovaný JavaScriptem. HTML je značkovací jazyk pro tvorbu hypertextových dokumentů vytvořený konsorciem W3C. Kaskádové styly (CSS) je jazyk pro popis způsobu zobrazování stránek. JavaScript je multiplatformní objektově orientovaný skriptovací jazyk.

Požadavkem, jehož důležitost se zvětšuje s rozmachem mobilních zařízení, je potřeba optimalizace webových stránek pro různá zařízení, resp. jejich rozlišení obrazovky. Základy responzivního designu, jak se této optimalizaci říká, jsou: (13)

1. Flexibilní layout – je nutné vyhnout se hodnotám zadaných v absolutních jednotkách nebo v pixelech, ale využívat jejich přepočet do procent, popřípadě relativních jednotek (em, ex, rem). Šířka prvků musí být tedy vždy zadána flexibilně, výška zůstává neměnná.
2. Flexibilní velikost obrázků – zajistit proměnnou velikost obrázků lze stejným způsobem jako layout. Problémem tohoto řešení je fyzická velikost obrázku – tím, že jej zmenšíme vizuálně, nedojde ke zmenšení velikosti souboru, a je tedy při zobrazování obrázku stahováno stejné množství dat nezávisle na velikosti rozlišení. To je nevhodné především u mobilních zařízení využívajících zatím pomalé mobilní připojení. Možností, jak se vyhnout použití některých obrázků například na ikony, je využití ikonových fontů.
3. Media queries – základ responzivního designu je poskytnout vzhled nejlepším možným způsobem pro dané zařízení. Ze strany kaskádových stylů se o to starají právě Media queries, které umožňují definovat styly pro různá rozlišení obrazovky.
4. Minimalizace počtu HTTP požadavků a přenášených dat – dalším omezením mobilních sítí je většinou pomalá odezva. Proto je dobré minimalizovat počet dotazů

na server, a to například za pomoci použití tzv. obrázkových spritů, slučováním JavaScriptových a CSS souborů.

Pro zrychlení a zefektivnění vývoje bude využito souboru nástrojů z frontendového balíku Twitter Bootstrap. Tato sada obsahuje základní podporu webových technologií a mnoho dalších prvků. Je postaven na HTML5, CSS3 a skládá se z mnoha komponent, které podporují responzivní webdesign a lze je využít ve veřejně přístupné části aplikace i v administraci. Mezi jeho největší přednosti patří soubor nadefinovaných tříd pro bloky, tabulky, tlačítka aj., u kterých není nutné dále řešit vzhled a jejich chování.

Další výhodou použití Twitter Bootstrap je integrace technologie LESS. LESS je CSS preprocesor napsaný v Node.js, který umožňuje ze zdrojového kódu zapsaného ve vlastní syntaxi vygenerovat CSS pro prohlížeč. Mezi podobně fungující technologie patří například i SASS či Stylus. Cílem CSS preprocesorů je zachovat čitelnost a srozumitelnost u stylopisu větších rozměrů a umožnit použití vnořených definic, proměnných, mixinů a matematických výrazů, které běžně kaskádové styly nepodporují. (10) Výsledkem tohoto je snadná znovupoužitelnost kódu a lepší udržitelnost. Naopak nevýhodou je nutnost kompilace.

### 3.2.3 Grunt

Grunt je nástroj napsaný nad Node.js pro tvorbu úkolů, který bývá využíván ke kompilaci (například LESS či SASS do CSS), minifikaci a spojování souborů (jak JavaScriptových tak CSS) a sledování změn.

### 3.2.4 Bower

Bower je balíčkovací systém od Twitteru, který lze zjednodušeně definovat jako composer pro externí knihovny u klientského JavaScriptu jako je například jQuery či Font Awesome a mnoho dalších, které lze přes Bower nainstalovat.[2] Jedním souborem ve formátu JSON lze definovat závislosti aplikace. Ukázka takového souboru je v bloku 3.1.

```
1  {
2      "name": "Korektit",
3      "version": "0.0.0",
4      "ignore": [
5          "**/.*",
6          "node_modules",
```

[2]Seznam knihoven, které lze přes Bower nainstalovat je dostupný na adrese http://sindresorhus.com/bower-components/

```
            "bower_components",
            "test",
            "tests"
        ],
        "dependencies": {
            "jquery": "~1.10.2",
            "font-awesome": "~3.2.1"
        },
        "resolutions": {
            "jquery": "~1.10.2"
        }
    }
```

Kód 3.1: Ukázka definice souboru bower.json

### 3.2.5 Nette Framework

Nette Framework je open source framework pro tvorbu webových aplikací v PHP 5 s plným využitím objektů (OOP). Je postaven s důrazem na použitelnost, vstřícnost a vedení k dobrým návykům – neopakovat se, zachovávat jednoduchost, ... Mezi jeho hlavní výhody patří:

- eliminace možných bezpečnostních rizik a zranitelností;
- využití pokročilého šablonovacího systému Latte;
- ladící nástroje;
- podpora HTML5, AJAX/AJAJ, SEO;
- dokumentace, velká podpora komunity v ČR a s ní spojené velké množství pluginů a rozšíření;
- čistý a ověřený objektový návrh;
- vysoká rychlost.

Tento framework je využíván v mnoha významných českých společnostech, z nichž některé působí i na mezinárodním poli. Příkladem je GE Money, Mladá fronta, Slevomat, Socialbakers, Ulož.to a mnoho jiných. (20)

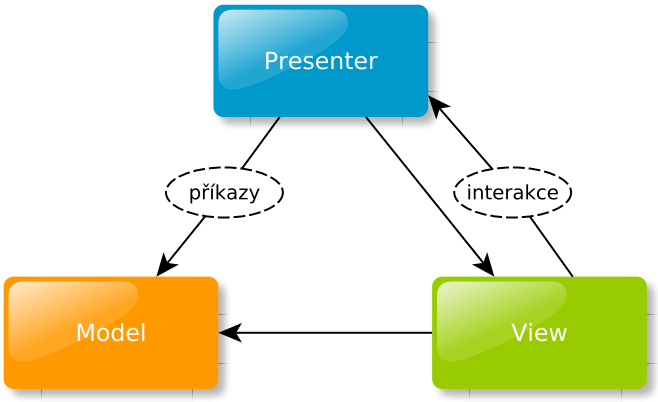
#### Architektura

Architektura Nette je postavena dle návrhového vzoru MVC. Ten usnadňuje rozdělení do třech základních vrstev Model-View-Controller, které spolu komunikují a jeho cílem je oddělit kód uživatelského rozhraní od kódu aplikační logiky. Toto řešení, oddělující tři

relativně samostatné části, zjednodušuje vývoj a údržbu aplikace a umožňuje testování jednotlivých částí zvlášť. V případě Nette se jedná o variantu MVP (9, 22):

- Model – obsahuje aplikační logiku a stará se o ukládání a načítání dat, obvykle tedy komunikuje s databází. Jakákoliv akce uživatele představuje akci modelu, který si zároveň spravuje svůj vnitřní stav a nabízí pevně dané rozhraní. O existenci view nebo presenteru model neví.
- View – Pohled se stará o prezentaci dat uživateli. K tomu je v Nette využito Latte šablon.
- Presenter – řadič, jehož hlavním úkolem je komunikovat s modelem a *prezentovat* jej v lidsky přívětivé formě. Dále udržuje stav persistentních proměnných.

Grafické znázornění MVP zachycuje obrázek 3.15.

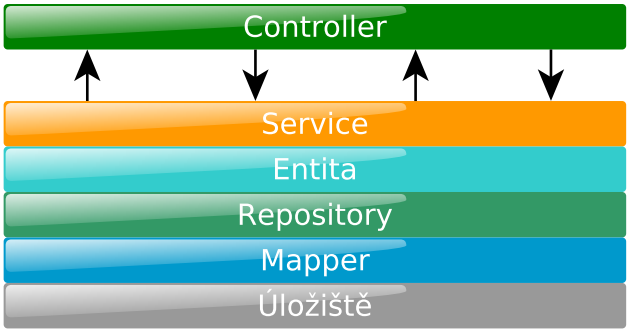



Obrázek 3.15: Architektura MVP v Nette Framework

### 3.2.6 ORM

Knihovna ORM nabízí vyšší míru abstrakce nad datovými úložišti. Z pohledu aplikace se o implementaci těchto úložišť nemusíme starat a jednoduše používáme třídy, které zajistí načítání dat včetně vazeb a jejich ukládání včetně kontroly datového typu. (32)

Obecně je objektově relační mapování programovací technika, která zajišťuje automatickou konverzi dat mezi relační databází a objektově orientovaným programovacím jazykem a má podporu ve všech moderních programovacích jazycích. Tento model se skládá z pěti vrstev:




Obrázek 3.16: Obecná struktura pětivrstvého modelu

**Entita** reprezentuje jakýsi objekt reálného světa. Tou může být například student, kniha, škola, aj. Na entitě jsou definovány také její vlastnosti včetně datových typů a vazby na ostatní entity s jejich násobnostmi.

**Repository** – jedná se o variantu návrhového vzoru DAO, tedy o vrstvu odstiňující nás od konkrétního použitého mapperu. V kontroleru se pak obracíme na repository, které dále spolupracuje (ukládá, načítá a maže entity) s konkrétním mapperem. Výhodou této konstrukce je možnost využití kombinace různých úložišť. Pomocí této konstrukce si mohu například před uložení do databáze předsunout uložení do memcache.

**Mapper** zajišťuje práci s konkrétním úložištěm. Typicky tedy načítání, ukládání a mazání záznamu – jeden mapper je pro daný typ entity odpovědný za veškerou obsluhu jednoho typu úložiště. Mapper nemusí spolupracovat pouze s databází, ale je možné jako úložiště využít například memcache, souborový systém či některou ze vzdálených služeb.

**Servisy** umožňují spouštět více událostí entity pomocí jednoho zavolání. Příkladem může být publikování jednoho článku:

```
1  if ($article->published !== Article::IS_PUBLISHED)
2  {
3      $article->published = TRUE;
4      $article->publishedTime = new DateTime;
5      $articleRepository->flush();
6  }
```

které lze pomocí servisy zavolat jednoduše například:

```
1  $article->publish();
```

Hlavními výhodami použití ORM frameworku je vytvoření abstraktní databázové vrstvy, zapouzdření do objektů, možnost testování pomocí Unit testů, validace hodnot (vstupních i výstupních) a v neposlední řadě i již předdefinované základní operace (CRUD). Nevýhodou je především větší paměťová a procesorová náročnost. (28) Pro implementaci bude využita ORM z https://github.com/PetrP/Orm, jehož výhodou je snadné napojení na Nette Framework a dále, že je postaven nad dibi. Dibi je databázová vrstva, snažící se o zjednodušení zápisu SQL příkazů, snadný přístup k metodám, eliminaci výskytu chyb a především o maximální možnou přenositelnost mezi databázovými systémy. Nabízí celou řadu modifikátorů, jejichž správné použití snižuje riziko napadení databáze, jelikož validuje vstupy. (32)

### 3.2.7 REST API

REST (Representational State Transfer) je architektura pro webové API vyvinutá souběžně s protokolem HTTP/1.1. Na rozdíl od známějších XML-RPC či SOAP je orientován datově, nikoliv procedurálně. Všechny zdroje mají vlastní identifikátor URI a REST definuje čtyři základní metody pro přístup k nim:

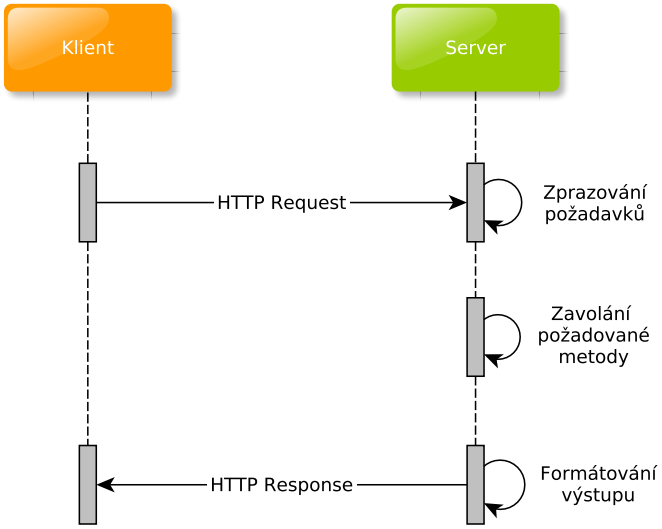
| Zdroj | Kolekce URI (http://e.com/documents/) | Element URI (http://e.com/documents/123) |
|---|---|---|
| GET | Seznam | Entita |
| PUT | Nahradí kolekci jinou kolekcí | Upraví entitu |
| POST | Vytvoří nový záznam v kolekci | Vytvoří entitu |
| DELETE | Smaže celou kolekci | Smaže entitu |

Tabulka 3.1: Metody pro přístup ke zdrojům REST

Základní principy REST jsou:

1. stav aplikace a chování se vyjadřují resourcem;
2. každý resource je identifikován pomocí URL, URN;
3. jsou definovány jednotné přístupy pro práci s resourcem v podobě čtyř základních operací;
4. resource může mít různé reprezentace (XML, HTML, JSON, ...).

REST nabízí různé standardizované formáty výměny dat, mezi něž patří ATOM-/RSS, XML a JSON. Posledně zmiňovaný poslouží pro účely této aplikace. Výsledné schéma bude zdokumentováno a publikováno pomocí apiary.io. (18)




Obrázek 3.17: Schéma REST komunikace

### 3.2.8 Testování

Automatické testování zajišťuje, že při úpravě nebylo narušeno chování systému. Vývojář nemusí při každé změně testovat, zda se úpravou nepoškodila jiná součást systému, ale jednoduše spustí automatické testování, které pokud projde, má uživatel jistotu, že testy pokrytá funkčnost systému zůstala zachována.

Webové aplikace lze testovat na několika úrovních. Jedním ze způsobů je testovat chování jednotlivých tříd a jejich funkcí k čemuž je určena knihovna PHPUnit. Chování aplikace přímo v prohlížeči včetně zapnutého JavaScriptu lze testovat například pomocí Selenia.

#### PHPUnit

Pro testování PHP aplikací se nejčastěji využívá knihovna PHPUnit. Ta umožňuje jednoduše popsat chování určité části aplikace, které je poté možno rychle otestovat.

```
1  class StackTest extends PHPUnit_Framework_TestCase
2  {
3      public function testPushAndPop()
4      {
5          $stack = array();
6          $this->assertEquals(0, count($stack));
7
8          array_push($stack, 'foo');
9          $this->assertEquals('foo', $stack[count($stack)-1]);
10         $this->assertEquals(1, count($stack));
11
12         $this->assertEquals('foo', array_pop($stack));
```

```
13            $this->assertEquals(0, count($stack));
14        }
15    }
```

Kód 3.2: Ukázka testování polí pomocí PHPUnit (4)

Takto napsaný test pak můžeme jednoduše spustit příkazem z konzole

```
phpunit --verbose StackTest.
```

#### Selenium

Selenium je multiplatformní nástroj pro automatické testování webových aplikací. Selenium testuje přes webový prohlížeč (např. Firefox) a umožňuje tento způsob testování i na serveru. Toto řešení se skládá z několika spolupracujících komponent, které dohromady dávají vývojářům nesmírně užitečný nástroj.

Pro možnost spouštění seleniových testů je nutné mít nainstalovány a spuštěny následující nástroje:

- Java;
- Selenium Server;
- Selenium IDE – plugin pro Firefox (popřípadě drivery pro jiné prohlížeče).

### 3.2.9 Plupload a Dropzone

Vzhledem k tomu, že uživatelé budou pracovat s dokumenty, je nutné zvolit vhodný nástroj pro jejich nahrávání na server. Nahrávání jakýchkoliv souborů na server je samo o sobě bezpečnostním rizikem. Ideálním řešením je uživateli tuto funkci vůbec neumožnit, což v tomto případě nepřipadá v úvahu. Bude tedy důležité nahrávání souborů dostatečně zabezpečit. Plupload je komponenta, která nabízí upload různých souborů v různém množství. Poskytuje technologie, které jsou pro daného uživatele optimální, ať už užívá moderní či zastaralý prohlížeč či využívá nebo nevyužívá flash. Dropzone je open source knihovna poskytující podporu drag and drop při nahrávání souborů včetně vytváření náhledů.

### 3.2.10 PhoneGap

PhoneGap je nástroj pro vytváření aplikací pro chytré mobilní telefony nezávisle na platformě. Je tedy možné, za jeho pomoci vytvořit aplikaci, která bude funkční jak pro Android, tak i pro iOS či mobilní platformy Windows. Je to velice účinný nástroj,

pro jehož použití je ale nutná pokročilá znalost JavaScriptu. Nevýhodou použití této technologie je malá rychlost oproti nativním aplikacím.

### 3.2.11 Composer

Composer je nástroj na správu závislostí v PHP. Umožňuje deklarovat libovolně složité závislosti jednotlivých knihoven, které následně nainstaluje do projektu.

Hlavní výhodou Composeru je možnost rychlé a snadné aktualizace všech použitých knihoven bez nutnosti manuálně ověřovat aktuální verze.

### 3.2.12 Git

Je distribuovaný systém správy verzí, původně určený pro vývoj jádra Linuxu, ale v dnešní době rozšířený napříč všemi odvětvími programování. Git bude využit jako primární verzovací systém pro vývoj a správu zdrojového kódu. Bude tak možné v budoucnu spolupracovat na vývoji ve větším týmu a jednoduše rozšiřovat a modifikovat jednotlivé verze za pomoci větvení, tagů a slučování.

| Technologie | Verze | Composer |
|---|---|---|
| Nette Framework | 2.0.12 pro PHP 5.3.7 | ✓ |
| Orm | 0.4.0-RC6 | ✓ |
| Texy! | 2.3 | ✓ |
| Dibi | 2.1.0 | ✓ |

Tabulka 3.2: Verze použitých technologií a možnost aktualizace přes Composer

### 3.2.13 DeployHQ a CI

DeployHQ je online nástroj pro nasazování systémů na různé servery. Umožňuje interakci s GITem, resp. se službou Bitbucket za pomoci tzv. POST Hooks. Za pomoci tohoto nástroje je možné nasazovat změny rychle a efektivně. Je možné nasměrovat různé větve repozitáře na odlišné servery, čehož bude využito pro rozlišení produkčního a beta serveru.

GitLab CI je open-source server pro průběžnou integraci. Jeho hlavním úkolem bude automatické spouštění testů před samotným nasazením změn na server a případné zastavení nasazování, pokud některý z testů selže.

### 3.2.14 Texy!

Pro vkládání a základní formátování obsahu textu poslouží program Texy, díky němuž lze snadno a bez odborných znalostí psát texty na webové stránky. Jeho hlavní výhodou je to, že vždy generuje validní HTML či XHTML kód. Při jeho použití je nutné dbát na správné nastavení, které zabrání uživatelům využít tento nástroj pro napadnutí systému.

Bude využit například pro vkládání detailního popisu uživatele.

### 3.2.15 Ostatní technologie

Dostupnost a jiné požadavky budou monitorovány pomocí logování chyb jak na straně aplikace, tak na straně serveru a dále pomocí nástroje New Relic či DataDog. Nástroje jsou závislé na operačním systému, který bude použit na produkčním serveru. Ideálním řešením by bylo využít New Relic, který však není vhodné používat na serverech s operačním systémem BSD.

## 3.3 Implementace

Pro projekt byla vybrána doména www.korektit.cz, na které bude provozován produkční systém. Na subdoméně beta.korektit.cz by pak měla být dostupná beta verze. Hosting byl vybrán především s ohledem na finanční možnosti začínajícího projektu a to od společnosti Gigaserver.cz. Tento hosting bez problémů splňuje požadavky pro běh Nette Framework a navíc má na rozdíl od většiny serverů nastaven memory_limit na 128MB. Nevýhodou je verze PHP 5.3.3, která bude ale podle informací od podpory do konce roku navýšena na 5.4. Pokud by zvolený hosting měl v budoucnu problémy s výkonem či jiné nedostatky, je možné celý projekt rychle přesunout na výkonnější stroj a to především díky verzovacímu systému Git, kde je zdrojový kód uchováván.

### 3.3.1 Postup řešení

Vývoj systému bude probíhat dle agilní metodiky Scrum. Zpočátku vývoje, kdy bude jediným vývojářem autor této práce, bude mít tato metodika především startovací a dokumentační charakter. Cílem tohoto postupu je zavést do projektu jistou techniku vývoje a usnadnit příchod a zaučení dalších členů do týmu.

Základem vývoje budou týdenní sprinty, které zaručí určitou funkčnost v daný čas. Na konci sprintu dojde k určení další práce na následující týden. Jednotlivé úkoly budou rozděleny a obodovány (časově odhadnuty). Bodování je v této fázi především

pro informační hodnotu – kolik času se vývojem prozatím strávilo, může být podkladem pro výpočet přibližné hodnoty projektu pro případ prodeje.

Úkoly budou vedeny a spravovány na serveru Bitbucket.org. Tato služba bude využita také pro hostování repozitáře Git. Její výhodou je možnost zapojení celého týmu, podpora code review, issue (bug) tracking, grafické zobrazení větví repozitáře a v neposlední řadě i vedení Wiki stránky. Další nespornou výhodou oproti konkurenční službě GitHub je cena. Na druhou stranu GitHub je více rozšířený a je podporován více službami (např. Apiary.io nabízí automatickou kompilaci blueprintu API přímo z repozitáře GitHub a její zveřejnění, na Bitbucket je nutné tyto akce provádět manuálně).

#### 3.3.1.1 CI a Deployment

Průběžná integrace a automatické nahrávání je nutnou součástí vývoje moderních webových aplikací.

Pro vývoj a testování budou v repozitáři vedeny dvě hlavní větve.

- **`DEPLOY_BETA`** – tato větev se bude automaticky nahrávat na beta server, který bude sloužit primárně k otestování provedených úprav před nasazením na produkční server.
- **`DEPLOY_LIVE`** – větev, jejíž obsah je nahráván na produkční server. Oproti `BETA` větvi není nahráván automaticky po pushnutí změn do repozitáře, ale je nutné jej spustit manuálně na DeployHq.com. Automatický deploy `LIVE` je zakázán záměrně, aby se předešlo nechtěnému publikování neotestovaných změn (k tomu by dojít nemělo ani při automatickém nahrávání díky automatickému testování, jedná se jen o další z opatřeních).

Každý vývojář v týmu by měl ukládat do repozitáře pouze změny a nové funkčnosti, které jsou již řádně otestovány na lokálním serveru vývojáře. Po nahrání těchto změn následuje sada procesů, které by měly vyústit ve zveřejnění otestované stabilní a optimalizované verze.

1. Push změn do repozitáře[3];
2. „Zamknutí“ webu – jednoduše pomocí souborového zámku `.isDeploy`;
3. Nahrání nové verze pomocí služby DeployHq;
4. Smazání cache;

[3]Platí pouze pro větev `DEPLOY_BETA`.

5. Spuštění migrací – pouze nově nahraných;
6. Spuštění testů;
7. Revert (vrácení) změn[4];
8. „Odemknutí“ webu.

V ideálním případě by mělo během tohoto procesu dojít také na zbuildování projektu (viz 3.2.3 – Grunt), upgrade knihoven pomocí `composer update` a zazálohování databáze. Tyto věci v současné podobě obstarává osoba zodpovědná za nasazení.

### 3.3.2 Struktura systému

Systém je rozčleněn do modulů a komponent dle filosofie použitého Nette Framework. Je navíc využito knihoven, doplňků, pluginů a komponent třetích stran pro pokrytí požadavků, které základ frameworku nenabízí. Všechny tyto použité nadstavby jsou distribuovány pod licencí MIT či novou BSD licencí a je tedy možné je využít.

Základní struktura aplikace je zobrazena a popsána v následující stromové struktuře.

```
/
├── app/ ..... adresář s aplikací
│   ├── config/ ..... konfigurační soubory
│   │   ├── config.neon ..... hlavní konfigurační soubor
│   │   ├── config.local.neon ..... lokální konfigurační soubor
│   │   ├── Configurator.php ..... konfigurátor vytvářející kontejner
│   │   └── Router.php ..... hlavní router
│   ├── model/ ..... modelová vrstva
│   │   ├── orm/ ..... entity, repozitáře a mappery
│   │   └── services/ ..... autentifikace a autorizace, translator, atd.
│   ├── AdminModule/ ..... modul Admin
│   ├── ApiModule/ ..... modul Api
│   ├── FrontModule/ ..... modul Front
│   ├── bootstrap.php ..... zaváděcí soubor aplikace
│   ├── exceptions.php ..... výjimky aplikace
│   └── .htaccess
├── libs/ ..... knihovny a rozšíření
├── log/ ..... logy
├── migrations/ ..... soubory migrací
├── temp/ ..... cache a dočasné soubory
└── tests/ ..... testy
```

[4]Pouze pokud selže některý z testů.

```
├── www/ ..... soubory frontendu
├── bower.json ..... nastavení závislostí pro Bower
├── composer.json ..... definice závislostí pro composer
├── Gruntfile.js ..... nastavení úkolů pro Grunt
└── package.json ..... nastavení závislostí Gruntu
```

Není zde popsána celková struktura, protože je mnohem složitější. Podrobněji budou dále popsány obsahy složek `migrations`, `tests` a `www`.

Pro zamezení kolizí mezi různými částmi aplikace bude využito základního jmenného prostoru `Korektit`.

### 3.3.3 Frontend

Frontendové soubory jsou uloženy v adresáři `www/`. Jeho struktura není nijak složitá. Základem jsou soubory `.htaccess` pro změny nastavení serveru, hlavní ikona portálu `favicon.ico`, `robots.txt` pro řízení přístupu robotů a `index.php`.

```
www/
├── adminer/ ..... nástroj adminer pro snadný přístup k databázi
├── assets/ ..... CSS, JS, fonty a související grafické podklady rozhraní
├── covers/ ..... hlavní obrázky pro webovou prezentaci
├── files/ ..... soubory korektur a uživatelů
├── .htaccess
├── favicon.ico
├── index.php
└── robots.txt
```

Dále je zde nástroj adminer, který umožňuje za pomoci jednoho souboru spravovat celou databázi. Jeho hlavní výhodou oproti například phpMyAdmin je rychlost práce. V adresáři `assets/` jsou nejdůležitější soubory frontendu, především JavaScriptové knihovny a soubory kaskádových stylů. Na produkční a na beta server nejsou nasazovány soubory LESS, ale pouze zbuildované CSS soubory.

### 3.3.4 Základní nastavení

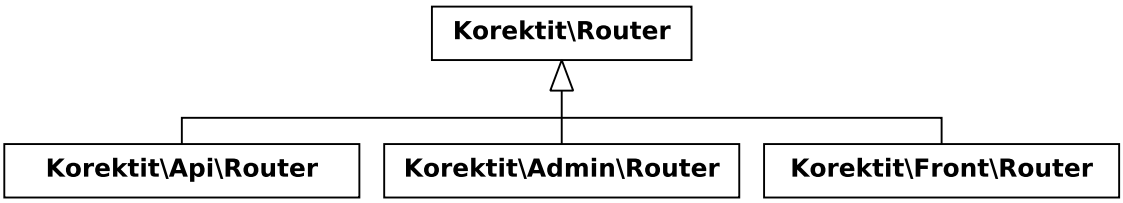
Pro běh aplikace je nezbytný systémový kontejner (statický Dependency Injection kontejner (21)), ve kterém se nacházejí všechny služby a parametry.

Vytvoření kontejneru obstarává třída `Korektit\Configurator`, která je potomkem `Nette\Config\Configurator`. Je možné se bez této třídy obejít a použít přímo konfigurátor Nette s možností rozšiřování v souboru `bootstrap.php`, což je ale nepřehlednější, a tím pádem hůře udržovatelné.

Nastavení Nette Framework, potažmo jiných knihoven systému, se zapisuje do konfiguračního souboru `config.neon`, respektive `config.local.neon` pro nastavení lokálních hodnot. Tyto konfigurační soubory umožňují nastavení základních parametrů aplikace (appNamespace, defaultLanguage, ...), dále nastavení PHP (date.timezone), Nette Framework, přístupových údajů k databázi a mnoho dalších. Konfigurační soubor je také místem, kde je umístěna definice vlastních služeb.

Routování, což je obousměrný překlad mezi URL a akcí presenteru, představuje samostatnou vrstvu aplikace. Tento přístup nám umožňuje bez problémů definovat podobu URL až po vytvoření celé aplikace a přitom zachovávat původní adresu platnou.

Na obrázku 3.18 je znázorněno schéma, jak jsou definovány Routery v této aplikaci. Jednotlivé routery mohou obsahovat kolekce rout. Toto řešení zvyšuje udržovatelnost aplikace a umožňuje nám snadno nadefinovat nové funkce.




Obrázek 3.18: Routery aplikace

Dalším nezbytným krokem je správné nastavení závislostí jednotlivých knihoven v souboru `composer.json`. Tento soubor má formát standardního souboru `json`. Jsou zde nadefinovány závislosti mezi jednotlivými knihovnami, jejich repozitáře a další kroky, například po provedení aktualizace knihoven (smazání cache, spuštění migrací a testů, aj.).

### 3.3.5 Moduly

Složky s presentery, šablonami a komponentami je možné rozdělit do podadresářů, které definují moduly. Tyto moduly mají svůj vlastní jmenný podprostor v rámci systému. V našem případě se jedná o moduly administrace (`Korektit\Admin`), prezentační části webu (`Korektit\Front`) a API (`Korektit\Api`).

#### 3.3.5.1 Front

Modul Front je nejjednodušší z modulů. Jeho hlavním úkolem je vygenerovat stránku za pomoci daného layoutu. Struktura modulu je nastavena tak, aby bylo možné v budoucnu, pokud bude již nedostačující rozvržení single-page, jednoduše přidávat další

stránky. Bude k tomu stačit upravit `Router.php` a vytvořit příslušný presenter, šablonu a eventuálně doplnit další komponenty, například pro generování navigace.

Základní struktura je zobrazena v následujícím bloku.

```
FrontModule/
├── presenters/
│   ├── BasePresenter.php
│   └── HomepagePresenter.php
├── templates/
│   ├── Homepage/
│   │   └── default.latte
│   └── @layout.latte
└── Router.php
```

Nejdůležitějším souborem je `Router.php`, který se stará o směrování v tomto modulu. Jedná se o třídu rozšiřující `Nette\Application\Routers\RouteList` a má nastaveno výchozí směrování na `HomepagePresenter`. Vzhledem k tomu, že se jedná o single-page, takto definovaný router je plně dostačující.

```
1  namespace Korektit\Front;
2
3  use Nette\Application\Routers\Route,
4  use Nette\Application\Routers\RouteList;
5
6  class Router extends RouteList
7  {
8      public function __construct($module = 'Front')
9      {
10         parent::__construct($module);
11
12         $this[] = new Route('[<lang=cs (cs|en|de)>/]<presenter>/<action
               >[/<id>]', 'Homepage:default');
13     }
14 }
```

Kód 3.3: Ukázka nastavení routeru pro modul Front

Router je navíc doplněn o možnost nasazení jazykových mutací (konkrétně české, anglické a německé) s výchozí českou mutací. Nette Framework je schopen detekovat a poskytnout mutaci v podobě, jakou vyžaduje přímo návštěvník, respektive dle jazyka, který je nastaven v jeho webovém prohlížeči.

Obsah zobrazující se na hlavní stránce je definován v souboru `@layout.latte`. Jak napovídá obsah tohoto souboru je možné při pozdějším rozšiřování webu jednoduše tento soubor rozšiřovat pomocí dalších bloků (`{include #block-name}`).

#### 3.3.5.2 Admin

Struktura modulu administrace je velice podobná modulu Front. Liší se množstvím presenterů a souvisejících šablon, kterých je vzhledem ke složitosti modulu více. Dále obsahuje adresář s rozšiřujícími komponentami, respektive upravené potomky komponent z `libs/`.

```
AdminModule/
├── components/
│   └── FilesUpload/
├── presenters/
│   ├── BasePresenter.php
│   ├── ⋮
│   └── UsersPresenter.php
├── templates/
│   ├── Users/
│   │   ├── add.latte
│   │   └── default.latte
│   ├── ⋮
│   └── @layout.latte
└── Router.php
```

V souboru `@layout.latte` je definována základní struktura webu, která obsahuje záhlaví, zápatí, obsahovou část a menu. Mění se především obsahová část, která je načtena pomocí makra `block` z aktuálně používaného presenteru, resp. šablony.

Položky v menu jsou zobrazovány dle role přihlášeného uživatele. K ověření slouží takzvané n makro `n:if`, které zabrání zobrazení položky uživateli, který nesplňuje interní podmínku makra `$user->isInRole('role')`.

U jednotlivých akcí a signálů v presenteru jsou kontrolována přístupová práva aktuálně přihlášeného uživatele, respektive jeho role. Tato práva je možné ověřovat za pomoci autorizační metody `checkAccess('zdroj', 'funkce')` v presenteru nebo využitím makra `n:if="$user->isAllowed('zdroj', 'funkce')"` v šabloně. Přístupová práva jsou nastavena v `Korektit\Authentication\Authorizator`.

Všechny presentery tohoto modulu mají stejný namespace `Korektit\Admin` a rozšiřují `BasePresenter` modulu, který rozšiřuje `BasePresenter` celé aplikace. Tato dědičnost nám umožňuje přidávat konstrukce, které mohou využívat všichni její potomci, a zároveň možnost odkazovat na daný presenter odkudkoliv z aplikace.

Nejpoužívanější částí administrace jsou samozřejmě formuláře. Je zbytečné zde popisovat funkčnost formulářů v Nette Framework, kterou je možné prostudovat v oficiální dokumentaci. Všechny vytvořené formuláře jsou ošetřeny proti CSRF útokům,

jednotlivé položky pak filtrují data při vstupu. Není tedy možné vložit data v jiném formátu než v tom, který je povolen. Tato kontrola probíhá na straně klienta (JavaScript `netteFroms.js`), i na straně serveru. Na serverové straně jsou data validována jak na úrovni jednotlivých políček, tak i z pohledu formuláře jako celku. K tomu je využíván callback validační funkce, který se k formuláři naváže jednoduše pomocí `$form->onValidate[] = callback($this, 'validateForm')`.

V administrační části je často využíván datagrid pro snadnou manipulaci s daty. Pro tento účel byl vybrán jQuery plugin jqGrid. Ten dokáže pomocí technologie AJAX vygenerovat tabulku a dále manipulovat s daty v tabulce – řadit, vyhledávat, stránkovat. Komunikace probíhá pomocí dat ve formátu JSON, který je generován funkcí `handleFillDatagird()` 3.4. Výsledná data jsou JavaScriptově zpracována a poté zobrazena.

```
1  public function handleFillDatagrid()
2  {
3      foreach ($corrections as $correction)
4      {
5          $data[] = array(
6              'id' => $correction->id,
7              'name' => $correction->title,
8              'filesCount' => $correction->files->count(),
9              'createdAt' => $correction->createdAt->format("d.m.Y"),
10             'actions' => array(
11                 $this->link('//Corrections:edit', $correction->id),
12                 $user->isInRole('admin') ?
13                     $this->link('//delete', $correction->id) : NULL,
14             ),
15         );
16     }
17     $response = new JsonResponse(
18         array('aaData' => $data),
19         'application/json; charset=utf-8'
20     );
21     $this->sendResponse($response);
22 }
```

Kód 3.4: Ukázka funkce generující JSON pro datagrid

Administrace je postavena na filosofii Nette Framework. Presenterům odpovídají šablony, které slouží především pro základní zobrazení (`default.latte`) a pro editaci a vytváření nových záznamů (`edit.latte`). Funkce presenterů jsou snadno pochopitelné a v případě složitějších je jejich funkcionalita řádně popsána v záhlaví funkce.

Jedinou rozšiřující komponentou je `FilesUploadControl`, která je více popsána v kapitole 3.3.6.

#### 3.3.5.3 API

Modul API bude sloužit především pro budoucí napojení např. na mobilní aplikace. V první verzi bude k dispozici jen nejnutnější sada funkcí.

```
ApiModule/
├── components/
│   ├── ApiResponse.php
│   ├── UserDetailBuilder.php
│   └── CorrectionDetailBuilder.php
├── presenters/
├── schemas/
└── templates/
```

Rekvesty na API nejsou definovány jako klasické routy. Pro jejich routování je využito `RestRoute`. Jde o nadstavbu výchozí routy, která navíc zajistí ověření validity HTTP požadavku, protože je obohacena o definovaný překladový slovník pro jednotlivé REST požadavky. Tato konstrukce zabraňuje možnosti použití jiné metody, než jakou daná funkce implementuje. Ukázka použití takové routy je v bloku 3.5.

```
1  $this[] = new RestRoute(
2      $prefix . '/me/password',
3      'Users:updatePassword',
4      RestRoute::METHOD_POST | $secured
5  );
```

Kód 3.5: Ukázka nastavení routeru API

Jak je z ukázky 3.5 vidět, na HTTP požadavek `/me/password` je zavolána funkce `updatePassword()` třídy `Korektit\Api\Users`.

```
1   /**
2    * Update user password
3    */
4   public function renderUpdatePassword()
5   {
6       if (isset($this->data->password))
7       {
8           $this->user->setPassword($this->data->password);
9           $this->orm->users->persistAndFlush($this->user);
10      }
11      else
12      {
13          $this->sendErrorResponse(ApiResponse::S401_3_WRONG_CREDENTIALS)
                ;
14      }
15      $this->sendSuccessResponse();
16  }
```

Kód 3.6: Ukázka API funkce zpracovávající /me/password

Tato funkce se pokusí nastavit nové heslo uživateli. Pokud se tato akce podaří, je odeslána odpověď o úspěšném dokončení. V případě neúspěchu je odeslán chybový stav. Odesílání zpráv včetně chybových kódů obstarává `Korektit\Api\BasePresenter`, a to konkrétně funkce `sendSuccessResponse()` a `sendErrorResponse()`. Tato třída zároveň zaznamenává všechny požadavky i odpovědi do příslušných databázových tabulek a v případě špatného požadavku je vytvořen i záznam v souborovém logu `api.log`.

Jednou z nejdůležitějších komponent modulu API je ApiResponse. Tato komponenta je postavena na třídě `Nette\Application\Responses\JsonResponse`, kterou doplňuje o seznam konstant definujících kódy odpovědí a formátování a sestavení odpovědi.

Seznam základních funkcí implementovaných v první verzi API je uvedený v následující tabulce 3.3.

| **Operace**[5]**: URL** | **Popis** |
|---|---|
| C: /api/auth/signup | Registrce uživatele |
| C: /api/auth/login | Přihlášení uživatele |
| C: /api/auth/logout | Odhlášení uživatele |
| U: /api/auth/profile | Úprava profilu |
| CRUD: /api/offers | Nabídky |
| CRUD: /api/offers/{offerId}/tags | Tagy |
| CRUD: /api/corrections | Korektury |
| CRUD: /api/corrections/{correctionId}/labels | Štítky |
| CRUD: /api/corrections/{correctionId}/types | Typy |
| CR: /api/messages | Notifikace |

Tabulka 3.3: Základní funkce API

Dokumentace API bude k dispozici na serveru Apiary.io, konkrétně na adrese http://korektit.apiary.io/.

### 3.3.6 Komponenty

#### Mailing

Komponenta obstarávající odchozí e-maily. Zprávy lze jednoduše vytvářet stejně jako šablony, do kterých se při zpracovávání doplní nastavené hodnoty. Stejně tak jako základní šablony, i šablony e-mailů lze převádět translátorem. Odeslané e-maily jsou ukládány do databáze.

[5]Operace jsou popsány zkratkou CRUD a jejími kombinacemi, kde C – create odpovídá HTTP POST; R – read, HTTP GET; U – update, HTTP PUT; D – delete, HTTP DELETE.

### FileUpload

Formulářová komponenta pro upload souborů. Umožňuje odesílání souborů dvěma způsoby:

- klasicky formulářem,
- pomocí Ajaxu.

Při zpracování formuláře vrací komponenta jako hodnotu prvku pole entit reprezentující jednotlivé soubory. Napojení na modelovou vrstvu probíhá následovně: konstruktoru komponenty se předá objekt `IFilesRepository`, který je zodpovědný za CRUD entit typu `IFileEntity`. Repository je také zodpovědné za přesun souboru mimo temp.

Komponenta poskytuje i ochranu proti CSRF. Pokud je formulář chráněný pomocí `Form::addProtection()`, tak jsou chráněné i ajaxové signály této komponenty. Metody `getUploadLink()` a `getDeleteLink()` k URL signálu přidají token a ten se při zpracování signálu zkontroluje.

### Cronner

Tato komponenta usnadňuje správu cronových úloh. Nastavení se provádí jednoduše pomocí anotací @cronner-task, @cronner-period, @cronner-days a @cornner-time. Ukázka nastavení úkolu opakujícího se každý den v 9 hodin je uvedena v bloku 3.7.

```
1  /**
2   * Daily user notifications
3   *
4   * @cronner-task User notifications
5   * @cronner-period 1 day
6   * @cronner-time 09:00
7   * @return CronTask
8   */
```

Kód 3.7: Nastavení cronu pomocí anotace

Komponenta vyžaduje PHP 5.3.3 a vyšší, což současný hosting splňuje. V systému bude využita pro pravidelné notifikace uživatelů, pro kontrolu termínů (například připomenutí konce lhůty pro vypracování korektury) a pro pravidelnou údržbu systému (promazávání starých souborů, atd.).

Veškeré úlohy jsou zaznamenávány v databázi, kde jsou k dispozici také informace, kdy úloha začala, kdy skončila a s jakým stavem, jaké bylo její ID procesu a jak velikou

část paměti si vyžádala. Pokud úloha skončí chybou, je také zaznamenána (výjimka, kód chyby, hláška, soubor a řádek).

#### PasswordHashCalculator

Slouží k vytvoření a verifikaci otisku hesla. Pro vytvoření otisku je využito funkce `crypt()` spolu s dynamickou solí hesla, kterou tvoří e-mailová adresa registrovaného uživatele. Verifikace hesla poté umožňuje porovnávat otisky vytvořené jak pomocí funkce `crypt()`, tak i `sha1()` kvůli kompatibilitě s mobilními aplikacemi. Počet opakování hashe $2^n$ (časovou náročnost) nastavuje proměnná `$rounds`.

Otisky vytvořené pomocí této komponenty by měly být dostatečně silné a neprolomitelné hrubou silou v reálném čase.

### 3.3.7 Model

V adresáři model se nacházejí třídy, reprezentující jednotlivé entity, jejich repozitáře a mappery. Dále pak servisy, které spolupracují s modelem, respektive s entitami.

```
model/
├── orm/
│   ├── Users/
│   │   ├── User.php
│   │   ├── UsersMapper.php
│   │   └── UsersRepository.php
│   └── ⋮
└── services/
    ├── Authentication/
    │   ├── Authenticator.php
    │   ├── Authorizator.php
    │   └── Registrator.php
    └── DatabaseTranslator.php
```

V adresáři `orm/` jsou vytvořeny podadresáře obsahující jednotlivé entity, mappery a repozitáře.

Entity jsou definovány za pomoci anotací @property. V bloku 3.8 je ukázka, jakým způsobem se definují vlastnosti entity User – její proměnné a vazby na ostatní entity.

```
1  /**
2   * Entity representing one user
3   *
4   * @property string $email
5   * @property string $username
6   * @property string $passwordHash
```

```
7     * @property string|NULL $name
8     * @property string $role {enum self::getRoles()}
9     * @property DateTime $signUpDate {default now}
10    * @property bool $banned {default false}
11    *
12    * @property Orm\OneToMany $offers {1:m OffersRepository $createdBy}
13    * @property Orm\OneToMany $corrs {1:m CorrectionsRepository $owner}
14    *
15    * @property Orm\ManyToMany $foos {m:m FoosRepository $users map}
16    */
```

Kód 3.8: Ukázka anotací entity User

Proměnné se automaticky mapují na sloupce databáze podle názvu proměnné. Proměnná $email tak odpovídá databázovému sloupci email. Pro zachování jmenné konvence (SQL vs. PHP) se názvy sloupců s podtržítkem (underscore_separated) mapují na související proměnné v podobě camelCase názvů. Například databázový sloupec password_hash bude mapován na proměnnou $passwordHash. U jednotlivých proměnných je pak nutné deklarovat jejich datový typ (nebo výčet), který je pomocí ORM neustále kontrolován. Dále je možné doplnit výchozí hodnoty a to buď klasickým výčtem, nebo lze využít funkcí entity.

Jak již bylo zmíněno, dále se v anotacích popisují vazby na ostatní entity sytému. Vazby je možné definovat jako OneToOne, OneToMany a ManyToMany. Tímto způsobem je možné mít vazby mezi tabulkami bez použití cizích klíčů, je ale určitě lepší tyto klíče využívat. U vazeb ManyToMany je vyžadována vazební tabulka pojmenovaná dle názvů entit, které svazuje, rozdělených písmem „x“. Název vazební tabulky pak může vypadat `users_x_foos`, kde mapovací entita je uváděna v názvu jako první a sloupce jsou pojmenovány jako proměnné vazby s příponou `_id`, tedy `user_id` a `foo_id`.

Velkou výhodou použitého ORM je možnost volání magických metod, vracejících entity či kolekce. Tyto metody se definují jako anotace. Ukázka definice takové metody v repozitáři je uvedena v bloku 3.9.

```
1   /**
2    * Repository containing articles
3    *
4    * @method Orm\DibiCollection findAll()
5    * @method Orm\DibiCollection findPublished()
6    * @method Orm\DibiCollection findByAuthorAndTime(User $user, DateTime
         $time)
7    */
```

Kód 3.9: Ukázka definice magických metod v Repository

Tato definice bez jakéhokoliv dalšího doplňování vrátí kolekci entit, které jsou svázány s daným autorem. Klíčovým slovem je `findXxx`, které vrací kolekci. Dále

je možné použít `getByXxx` vracející konkrétní entitu. Selektovat je možné i nad více sloupci za pomoci zřetězení. Při volání z presenteru může mít tvar:

```
$articles = $this->orm->articles->findByAuthorAndTime($author, $time);
```

Nevyhovuje-li podoba vráceného výsledku, je možné funkci upravit či doplnit v mapperu (toto je vhodné použít například při nutnosti jiného seřazení kolekce, než jaké je nastaveno jako výchozí v databázi či u pre/postfixů názvu sloupců).

Tímto způsobem jsou nadefinovány veškeré entity v systému.

#### 3.3.7.1 Servisy modelu

Adresář `services/` obsahuje služby spolupracující s modelem. V našem případě se jedná o `DatabaseTranslator` sloužící pro překlad statických textů a hlavně o servisy pro podporu práce s uživateli (přihlašování, registrace, obnova hesla, ...) – `Authenticator`, `Authorizator`, `Registrator` a `TokenGenerator`.

**Authenticator** slouží pro autentizaci uživatele v průběhu přihlašovacího procesu. Funkce `authenticate()` zkusí nejprve načíst uživatele dle zadané e-mailové adresy. Pokud je tato část úspěšná, ověří se, zda je již uživatel aktivní (ověřen e-mailem) a není zablokován. Pokud některá z těchto podmínek není splněna, je vyhozena odpovídající výjimka. Pokud je uživatel identifikován, je aktivní a nezablokován, pak se verifikuje odeslané heslo s uloženým otiskem. Pokud se otisky shodují, je proces autentizace ukončen a je vrácena identita uživatele.

**Authorizator** je rozšířením třídy `Nette\Security\Permission`. Jsou zde nastavena veškerá přístupová práva ke zdrojům a jejich operacím dle uživatelských rolí. Nastavení je poměrně jednoduchou záležitostí jak ukazuje kód v bloku 3.10.

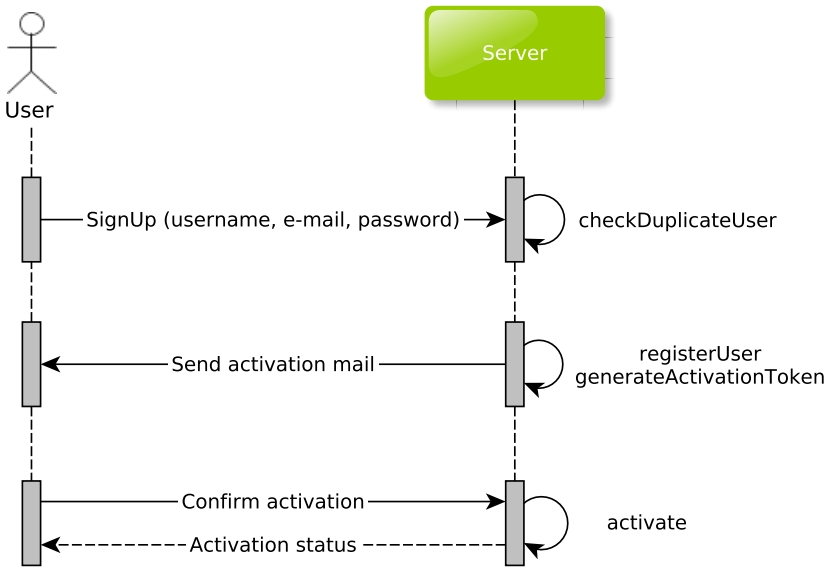
```
$this->addRole('editor');
$this->addRole('admin');

$this->addResource('article');
$this->addResource('user');

$this->allow('editor', 'article', 'add');
$this->allow('editor', 'article', 'edit', callback($this, 'isOwned'));

$this->allow('admin', 'article', self::ALL);
$this->allow('admin', 'user', self::ALL);
```

Kód 3.10: Ukázka definice přístupových práv

Je zde možné nastavit dědičnost práv rolí a mnoho dalších autorizačních pravidel, které nebudeme dále popisovat.

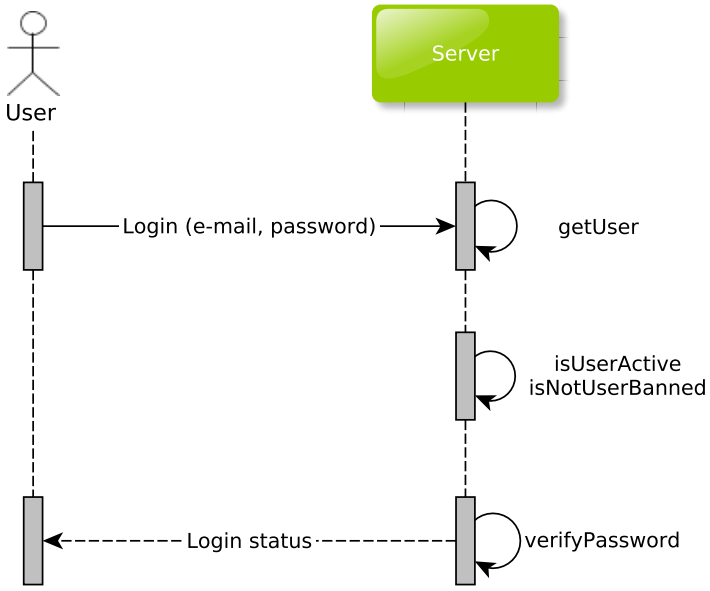
Registraci uživatele obstarává třída **Registrator**. Tato třída registruje uživatele včetně přiřazení role, a to jen v případě, že uživatel není ještě registrován. Při registraci je uživateli přiřazen aktivační token, který mu je zaslán na uvedený e-mail a je možné jednoduše přes odkaz provést aktivaci uživatelského účtu. Tento token je generován za pomoci služby **TokenGenerator**, která využívá pro generování statické funkce `Nette\Utils\Strings::random()`.

Schéma postupu při registraci a přihlašování uživatele je znázorněno v grafech 3.19 a 3.20




Obrázek 3.19: Registrace uživatele

Další službou úzce spolupracující s modelem je DatabaseTranslator. Tato služba obstarává překlady a podporuje pluralizaci. Je implementací rozhraní pro překlad `Nette\Localization\ITranslator`, díky němuž je možné jednoduše zachycovat fráze pro překlad a to například pomocí maker v šabloně `_{'Zprava'}` či jejich párových variant `{_}Zprava{/_}`. Pomocí napojení na model jsou nepřeložené fráze zaznamenávány do databáze, kde je pak možné je postupně doplňovat. Pokud by se stalo, že fráze zatím neexistuje, je zaznamenána a je administrátorovi zaslán informační e-mail s požadavkem na doplnění překladu.




Obrázek 3.20: Přihlášení uživatele

### 3.3.8 Migrace

Při vývoji aplikace je nutné zachovávat integritu databáze a zároveň snadno upravovat strukturu aby odpovídala změnám v aplikaci. K tomu bude sloužit nástroj Migrace. Tato knihovna se nainstaluje jednoduše pomocí composeru nebo je možné ji zkopírovat přímo do složky `migrations/` a v ní založit podadresáře `struct/` a `data/`.

```
migrations/
├── data/
├── struct/
├── index.html ..... statická stránka s odkazy pro spouštění migrací
└── run.php ..... obdoba bootstrap.php pro migrace
```

Složky `struct/` a `data/` mají následující účel:

- ve složce `struct/` jsou migrace, které budou součástí výsledné aplikace a nahrají se po dokončení na produkční server. Jde o všechny změny struktur, ale patří sem i různé číselníky, které budou ve výsledné aplikaci;
- ve složce `struct/` jsou migrace s testovacími daty potřebné pro běh integračních testů. Nebudou tedy ve výsledné aplikaci a nebudou nahrávány na produkci a i v případě, že by se tam dostaly, pak není možné je spustit.

Důležitým upozorněním je, že vytvořené migrace se nesmí za žádnou cenu mazat ani editovat – požadované změny lze opravit vytvořením další migrace.

Přidání nové migrace je velice jednoduché. Stačí vytvořit nový soubor s příponou `*.sql` ve formátu `YYYY-MM-DD[-N][-description].sql`, kde description je volitelný, ale je doporučeno jej psát pro lepší přehlednost. N je číslo pro určení pořadí v případě, že migrace mají stejný datum (jinak se spouštějí v abecedním pořadí), například `2011-12-29.sql`, `2011-12-30-1-foo.sql`, `2011-12-31-boo.sql`.

Dotazy migrací jsou spouštěny ve výše popsaném pořadí. Migrace jsou zaznamenávány v databázi v tabulce `migrations`. Tabulku tvoří sloupec `file`, obsahující jména souborů jednotlivých migrací, `executed`, datum a čas kdy byla daná migrace spuštěna, `checksum`, ověřující neměnnost souboru migrace a nakonec `ready` informující o tom, zda migrace proběhla v pořádku.

### 3.3.9 Testování

Do složky `tests/cases/` jsou ukládány unit testy a integrační testy včetně seleniových. Všechny testy používají knihovnu PHPUnit. Testovány jsou i komponenty API a Cron. Základní struktura je zachycena v následujícím bloku.

```
tests/
├── cases
│   ├── Selenium/ ..... seleniové testy
│   └── Unit/ ..... PHPUnit testy
├── inc/
│   ├── Selenium/ ..... definice Pages, Router, TemplateFactory, Configurator
│   ├── Unit/ ..... podpůrné skripty PHPUnit testů
│   ├── TestsConfigurator.php ..... Nette Configurator pro testy
│   └── tests.neon ..... konfigurace testů
├── test-www/
│   └── tmp/ ..... cache, sessions, aj.
├── boot.php ..... obdoba bootstrap.php
└── index.php
```

#### PHPUnit a HttpPHPUnit

Pro testování pomocí PHPUnit bude využita nadstavba HttpPHPUnit, která umožňuje spouštění testů jednoduše přes webové rozhraní.

Unit testy a neseleniové integrační testy dědí od `Tests\TestCase` a jsou umístěny ve jmenném prostoru `Tests`, popřípadě ve jmenném prostoru, který pod něj spadá.

Ukázka unit testu kontrolujícího ErrorPresenter, konkrétně funkci pro kontrolu stavového kódu 404 – Not Found, je uvedena v bloku 3.11


Obrázek 3.21: Webové rozhraní PHPUnit testů

```php
namespace Tests\Unit;

/**
 * Kontrola error presenteru
 */
class ErrorsTest extends TestCase
{
    /**
     * Kontroluje not found #404
     */
    public function test404()
    {
        if ($this->context->parameters['runExtraTests'] == FALSE)
        {
            $this->markTestSkipped("Extra testy nejsou vyzadovany.");
        }

        $baseUrl = $this->context->parameters['http']['host'] . $this->
            context->parameters['http']['path'];
        $url = $baseUrl . 'tahlestrankate2merjist1eneexistuje';

        $handle = curl_init($url);
        curl_setopt($handle,  CURLOPT_RETURNTRANSFER, TRUE);

        $response = curl_exec($handle);
        $httpCode = curl_getinfo($handle, CURLINFO_HTTP_CODE);

        $this->assertSame($httpCode, 404);
        curl_close($handle);
    }
}
```

Kód 3.11: Ukázka PHPUnit testu

### Selenium

Selenium je nástroj pro automatické testování webových aplikací. Testování probíhá přímo přes prohlížeč (např. Firefox) a umožňuje tento způsob testování i na serveru.

Seleniové testy dědí od `Tests\Selenium\TestCase` a jsou umístěny ve jmenném prostoru `Tests\Selenium`, eventuálně jemu podřízených.

```
1  namespace Tests\Selenium;
2
3  /**
4   * SignIn Test
5   */
6  class SignInTest extends SeleniumTestCase
7  {
8
9      /**
10      * Tests signing in
11      * Using Authentication feature.
12      *
13      */
14     public function testSignIn()
15     {
16         $result = $this->auth->login(
17             'Frantisek',
18             'Dobrota'
19         );
20         $this->assertFalse($result);
21
22         $result = $this->auth->login(
23          $this->context->parameters['selenium']['testUser']['email'],
24          $this->context->parameters['selenium']['testUser']['password']
25         );
26         $this->assertTrue($result);
27     }
28 }
```

Kód 3.12: Ukázka seleniového testu

Seleniové testy využívají strukturu takzvaných Pages. Jedná se o třídy postavené na návrhovém vzoru Page Object reprezentující vždy jednu konkrétní stránku, kde jsou definovány funkce a elementy, které stránka obsahuje či vykonává. Funkce se zapisují jako standardní PHP funkce. K elementům stránek lze přistupovat pomocí anotace @property-read, kde je možné odkazovat buď na samostatný element či na pole elementů – to je možné využít například u tabulek a jejích řádků. Pomocí anotace @method je pak možné definovat akce nad elementy, například přechody pomocí odkazů mezi jednotlivými stránkami.

Elementy jsou načítány ze stránky pomocí jazyka XPath, který umožňuje adresaci v XML, jehož podmnožinou je i jazyk HTML.

```
/**
 * Users page.
 *
 * @property-read Element[] $datagridRows xpath="//table[@id='users-datagrid']//tbody//tr"
 * @property-read Element $addNewUser xpath="//a[contains(text(), 'Novy clanek')]"
 * @method ArticlesAdd clickAddNewArticle()
 *
 * @property-read Element $profile link text='Zmena profilu'
 * @method Profile clickProfile()
 * @property-read Element $logout link text='Odhlasit se', a
 *
 */
```

Kód 3.13: Ukázka anotace Users PageObject

S takto popsanou stránkou je pak už možné jednoduše pracovat přímo v seleniovém testu:

```
$users = new Pages\Admin\Users($this->session);
$profile = $users->clickProfile();
```

Kód 3.14: Ukázka práce s PageObject v seleniovém testu

Jak seleniové tak PHPUnit testy je možné spouštět pomocí rozhraní HttpPHPUnit a mělo by se tak stávat před každým nasazováním provedených změn na server. Pokud se jedná o novou funkcionalitu, pak by k ní měl vždy být doplněn související test.

### 3.3.10 Zabezpečení

Systém musí být dostatečně zabezpečen proti útokům popsaných v kapitole věnující se zabezpeční 3.1.7. Proto byla přijata následující opatření:

- Důkladně sanitizovat data – obzvláště data vypisovaná do atributů.
- Dibi – používat modifikátory a používat je správně (pozor na vedlejší efekty [%i umí i pole]).
- CSRF – používat `$form->addProtection()` a zabezpečené signály.
- Nepoužívat superglobální proměnné ale proměnné již odfiltrované a zpracované frameworkem (GET, POST).
- Pro hešování hesel používat `crypt()` (od PHP 5.3.7) nebo eventuálně `password_hash()` (od php 5.5).

- Náhodné řetězce generovat metodou `Nette\Utils\String::random()`, jež dokáže generovat unikátní náhodné řetězce, které nejsou odhadnutelné jako jiné nativní „rand“ funkce php či jejich kombinace (`uniqid()`, `rand()`, ...). Další kryptograficky bezpečnou funkcí, jejíž hodnotu nelze teoreticky odhadnout a je tedy možné ji využít, je funkce `openssl_random_pseudo_bytes()`.
- Zabezpečení souborů .neon – pomocí .htaccess, index.php, změnou jména.
- Správně nastavit produkční server – vypnutí zobrazování chyb, nastavení systému na produkčním serveru na Debugger::PRODUCTION – chyby se nezobrazují, ale zaznamenávají do souborového a databázového logu, nezobrazuje se Laděnka, správné nastavení a skrytí pomocí .htaccess.
- Pro zaznamenávání chyb na produkčním serveru využít souborový log a e-mailové upozornění podpory, že došlo k chybě.

Kapitola 4

# Výsledky a diskuse

## 4.1 Testování

Stejně jako u všech softwarových i jiných produktů, tak i u webových aplikací je nutné výsledek důkladně otestovat. Tato kapitola je zaměřena na popis způsobů, jakými byla aplikace testována v průběhu vývoje a po jeho dokončení.

### 4.1.1 Selenium a PHPUnit testy

Testování pomocí Selenia a PHPUnit testů je prováděno pravidelně před každým nasazením na produkční server – pokud testy odhalí problém, nasazení by se mělo zastavit. Není tedy nutné tyto testy spouštět na produkčním serveru a to zejména kvůli vysoké výpočetní náročnosti testů, které by mohly server zahltit. Jak již bylo popsáno v kapitole věnované implementaci testů, testy by měly být spouštěny samotnými vývojáři na jejich lokálních serverech a navíc jsou spouštěny během deployovacího procesu.

### 4.1.2 Zobrazení v prohlížečích

Stejné a bezchybné zobrazení stránky v různých prohlížečích je důležitou vlastností každé webové prezentace. Ke stejnému zobrazení napomáhá validní HTML a CSS kód. Ani ten ale bohužel nezaručuje stejnou interpretaci a dochází tak k rozdílům ve vzhledu, především pak ve starších prohlížečích Internet Explorer.

Pro testování zobrazení bylo použito webu www.browsershots.org, který umožňuje automatické vygenerování náhledů stránek v různých prohlížečích a na různých platformách. Jeho hlavní nevýhodou je nemožnost generování náhledů stránek vyžadujících přihlášení uživatele. Stránky vyžadující autentizaci byly testovány lokálně na

virtuálních strojích a dále pomocí služby http://www.browserstack.com. Tato služba umožňuje vytvoření webového tunelu na lokální server. Tímto způsobem je možné zdarma otestovat zobrazení v prohlížečích IE6+, FF3+, Safari4+, Chrome14+ a Opera 10+ a to na platformách Microsoft Windows XP, 7 a 8.

Zobrazení bude testováno v nejpoužívanějších prohlížečích dle veřejně dostupných statistik z října 2013 na webu w3schools.com (24) a to především v rozlišení větším než 1024×178 pixelů. Dále bude otestováno na zařízení iPod (iOS 6.1.3) s fyzickým rozlišením 480×320 pixelů, Google Nexus 7 (Android 4.1, 1280×800). V prohlížečích umožňujících testování responzivního zobrazení budou otestována i menší rozlišení.

| **Prohlížeč/zařízení, OS** | **BrowserShots** | **BrowserStack** | **Zobrazení přímo v prohlížeči** |
|---|---|---|---|
| **MS Internet Explorer 10, Windows XP** | bez chyb | bez chyb | – |
| **Mozilla Firefox 25, Linux Mint 15 a Windows XP** | problém s fonty | problém s fonty | problém s fonty |
| **Google Chrome 30, Linux Mint 15 a Windows XP** | bez chyb | bez chyb | bez chyb |
| **Apple Safari 5, Windows XP** | bez chyb | bez chyb | bez chyb |
| **Norwegian Opera 17 Windows XP** | bez chyb | bez chyb | bez chyb |
| **iPod, iOS 6.1.3** | – | bez chyb | bez chyb |
| **Google Nexus 7, Android 4.1** | – | bez chyb | bez chyb |

Tabulka 4.1: Výsledky testu zobrazení v prohlížečích

### 4.1.3 Zabezpečení

Systém byl otestován proti základním typům útoků a to pomocí volně šiřitelných doplňků prohlížeče Firefox.

- SQL Injection – doplněk SQL Inject ME;
- Cross-site scripting (XSS) – doplněk XSS ME;
- XSS, CSRF, ... – doplněk NoScript Security Suite.

Provedené testy neodhalily žádná bezpečnostní rizika.

### 4.1.4 Testování použitelnosti

Testování použitelnosti je soustředěno především na uživatele. Jeho cílem je zjistit použitelnost systému v praxi a odhalit chyby ve funkčnosti a uživatelském rozhraní. (17)

#### 4.1.4.1 Návrh testování

Jako místo testování by měl primárně sloužit Usability lab (34), kde je možnost sledovat uživatele během plnění testovacích úkolů a následně tyto poznatky zahrnout do výsledků testování. Pro vyvíjený systém bude dostačující testování pomocí dotazníků a to ve třech fázích.

1. Tester vyplní dotazník týkající se osobních údajů a zkušeností s prací na PC a internetu.
2. Tester postupuje dle předem daného testovacího scénáře složeného ze sady úkolů. Jednotlivé úkoly byly vybrány s ohledem na co nejširší otestování celého systému.
3. Posledním krokem je vyplnění dotazníku, umožňujícího testerovi vyjádřit subjektivní pocity a dojmy týkající se systému (přehlednost, složitost, prezentace informací).

#### 4.1.4.2 Testovací skupiny

Vzhledem k povaze webových stránek byly zvoleny následující testovací skupiny – běžní návštěvníci webu, studenti, korektoři.

#### 4.1.4.3 Definování uživatelů

1. skupina
   - Lidé 20 – 30 let bez rozdílu pohlaví.
   - Vysokoškolští studenti – zpracovávající závěrečnou práci.
2. skupina
   - Lidé 6 – 65 let bez rozdílu pohlaví.
   - Korektoři – znalí problematiky a požadavků na korektorské práce.
3. skupina
   - Lidé 6 – 65 let bez rozdílu pohlaví.
   - Běžní uživatelé – běžní návštěvníci stránek.

#### 4.1.4.4 Testující osoby

Počet testujících osob byl 11.

| Číslo testujícího | Pohlaví | Věk | Vzdělání | Zkušenosti s prací na internetu a PC | Zkušenosti se zpracováváním korektur |
|---|---|---|---|---|---|
| 1 | muž | 26 | SŠ | Velmi dobré | Velmi dobré |
| 2 | muž | 28 | VŠ | Velmi dobré | Žádné |
| 3 | muž | 56 | VŠ | Dostatečné | Žádné |
| 4 | muž | 20 | ZŠ | Dobré | Žádné |
| 5 | muž | 14 | ZŠ | Velmi dobré | Dobré |
| 6 | žena | 30 | VŠ | Dobré | Dobré |
| 7 | žena | 42 | SŠ | Dostatečné | Žádné |
| 8 | žena | 31 | VŠ | Velmi dobré | Velmi dobré |
| 9 | muž | 27 | VŠ | Velmi dobré | Dobré |
| 10 | žena | 24 | VŠ | Velmi dobré | Dobré |
| 11 | muž | 36 | VŠ | Dobré | Velmi dobré |

Tabulka 4.2: Přehled účastníků testu

#### 4.1.4.5 Úkoly

**1. Úkoly běžného uživatele**

- Zaregistrujte se.
- Nechte si zaslat zapomenuté heslo.
- Kontaktuje provozovatele pomocí e-mailu.
- Odhlaste se.

**2. Úkoly pro nabízející**

- Zaregistrujte se.
- Doplňte uživatelské údaje.
- Vytvořte novou korekturu.
- Vytvořte nabídku.
- Stornujte nabídku.

**3. Úkoly pro korektory**

- Zaregistrujte se.
- Vyplňte své schopnosti.
- Vyhledejte a vyberte si z nabídky korekturu.
- Kontaktujte zadavatele korektury.
- Přijměte nabídku.

#### 4.1.4.6 Výsledky

Při zpracovávání výsledků bylo použito hodnocení dle následující stupnice.

1. Dokončil bez jakýchkoliv problémů.
2. Dokončil s menšími problémy.
3. Pro dokončení potřeboval delší dobu.
4. Dokončil s velkými problémy.
5. Nedokončil.

| | Číslo úkolu | | |
|---|---|---|---|
| **Číslo testujícího** | **1** | **2** | **3** |
| 1 | 1 | 1 | 1 |
| 2 | 1 | 1 | 1 |
| 3 | 2 | 4 | 3 |
| 4 | 2 | 1 | 2 |
| 5 | 1 | 3 | 1 |
| 6 | 1 | 2 | 1 |
| 7 | 2 | 3 | 2 |
| 8 | 1 | 2 | 1 |
| 9 | 1 | 1 | 1 |
| 10 | 1 | 1 | 2 |
| 11 | 1 | 1 | 1 |
| **Průměr** | **1.3** | **1.8** | **1.5** |

Tabulka 4.3: Tabulka hodnocení testovaných úkolů

Jak znázorňuje tabulka 4.3, úspěšnost dokončení jednotlivých úkolů se pohybuje v průměru od 1.3 do 1.8, což znamená, že většina uživatelů dokončila úkoly bez problémů nebo jen s drobnými obtížemi.

Testování nebyla podrobena veškerá aktuální funkcionalita. Některé části systému byly doplněny až po testování použitelnosti s uživateli a jsou tedy otestovány pouze kognitivně.

Z dotazníků, které uživatelé vyplňovali po testu je patrné, že testující byli s aplikací až na drobné nedostatky spokojeni. Uživatelé by ocenili především podrobné vyhledávání napříč nabídkami – například možnost zobrazovat nabídky dle vybraných tagů. Na Frontu aplikace pak bylo menším problémem menu, které je ve výchozím nastavení skryto. To činilo komplikace uživatelům s menšími zkušenostmi s prací na internetu a PC, kteří ho nevyužívali pro navigaci po stránce. V administraci by bylo vhodné doplnit upřesňující popisky ke štítkům nebo vytvořit znalostní bázi (uživatelé často nevěděli, jaký je například rozdíl mezi různými druhy korektur).

### 4.1.5 Analýza přístupnosti podle WCAG 2.0

WCAG jsou první pravidla zabývající se přístupností webových stránek, která vytvořila skupina WAI spadající pod konsorcium W3C. V současné době je platná verze WCAG 2.0, která podobu oficiálního doporučení W3C dostala 11. 12. 2008. (33). Kompletní znění metodiky zde nebude uváděno. Český překlad metodiky je veřejně dostupný. (26)

Webová přístupnost je vlastnost, která zaručuje návštěvníkovi, že webové stránky nekladou žádné překážky v jejich používání. Obecně lze přístupnost charakterizovat následujícími body:

- *Přístupnost je vlastností webových stránek.*
- *Přístupnost spadá pod obecnější vlastnost, kterou je použitelnost.*
- *Přístupnost je vždy nutné chápat ve vztahu ke konkrétnímu uživateli a jeho potřebám.*
- *Přístupný web umožňuje plnohodnotné používání všem svým uživatelům bez ohledu na jejich hendikep – ať už trvalý či dočasný.*
- *Přístupný web umožňuje plnohodnotné používání všem svým uživatelům bez ohledu na používané technické vybavení.*
- *Přístupný web umožňuje plnohodnotné používání všem svým uživatelům bez ohledu na jejich znalosti a dovednosti. (7)*

Aktuálně neexistuje žádný zákon, který by nařizoval dodržování pravidel přístupnosti pro běžné webové prezentace. V platnosti je pouze zákon 356/2000 Sb., který se však týká jen prezentací státní správy (16). Bohužel ani v oblasti SEO a SEM se neklade takový důraz na dodržování pravidel WCAG. SEO zahrnuje několik faktorů (atributy alt, title, atd.), ne však všechna pravidla. Vytvářet přístupný web je tedy důležité zejména kvůli hendikepovaným uživatelům, zvyšuje se tím především důvěryhodnost a sociální zodpovědnost organizace provozující stránky.

Analýzu bude prováděna jen pro vybrané stránky systému:

1. Hlavní strana;
2. Registrace;
3. Dashboard.

Analýzu bude prováděna dle kritérií nejnižší úrovně A.

#### 4.1.5.1 Vnímatelnost

- Textové alternativy – *Opatřete každý netextový obsah textovými alternativami, které je možné podle potřeby převést do jiných formátů jako například zvětšené písmo, bodové písmo, fonetický přepis či zjednodušený jazyk.* Viz tabulka 4.4

  Analyzované stránky splnily všechny podmínky pro splnění tohoto kritéria. Formulářům v Nette se jednoduše přidává ke každému prvku makro {label}, jež generuje titulek v podobě tagu label včetně odkazu na související prvek.

| **Kontrolní kritérium** | **1** | **2** | **3** |
|---|---|---|---|
| 1.1.1 | ANO | ANO | ANO |

Tabulka 4.4: Analýza WCAG 2.0 – 1.1 Textové alternativy

- Multimediální prvky závisející na čase – *Opatřete multimediální prvky závisející na čase alternativami.*

  Stránky neobsahují multimediální obsah.

- Přizpůsobitelné – *Vytvořte obsah, který lze prezentovat více způsoby, například zjednodušený vzhled, aniž by přitom došlo ke ztrátě informací či narušení struktury.* Viz tabulka 4.5

| **Kontrolní kritérium** | **1** | **2** | **3** |
|---|---|---|---|
| 1.3.1 | ANO | ANO | ANO |
| 1.3.2 | ANO | ANO | ANO |
| 1.3.3 | ANO | ANO | ANO |

Tabulka 4.5: Analýza WCAG 2.0 – 1.3 Přizpůsobitelné

- Rozlišitelné – *Usnadněte uživatelům slyšet a vidět obsah a odlište popředí od pozadí.* Viz tabulka 4.6

  Všechny stránky splnily bez problémů úroveň A.

| **Kontrolní kritérium** | **1** | **2** | **3** |
|---|---|---|---|
| 1.4.1 | ANO | ANO | ANO |
| 1.4.2 | Stránky neobsahují audio obsah | | |

Tabulka 4.6: Analýza WCAG 2.0 – 1.4 Rozlišitelné

#### 4.1.5.2 Ovladatelnost

- Přístupnost z klávesnice – *Ujistěte se, že všechny funkce jsou dostupné z klávesnice.* Viz tabulka 4.7

  Veškeré funkce jsou přístupné z klávesnice pomocí klávesy `Tab`.

| **Kontrolní kritérium** | **1** | **2** | **3** |
|---|---|---|---|
| 2.1.1 | ANO | ANO | ANO |
| 2.1.2 | ANO | ANO | ANO |

Tabulka 4.7: Analýza WCAG 2.0 – 2.1 Přístupnost z klávesnice

- Dostatek času – *Poskytněte uživateli dostatek času k přečtení a k práci s obsahem.* Viz tabulka 4.8

  Uživatel je limitován pouze délkou přihlášení v systému, které platí do vypršení dané session. Tuto délku není možné upravovat, ale prodlužuje se automaticky načtením jakékoliv stránky. O automatickém odhlášení je uživatel informován pomocí flashMessage.

| **Kontrolní kritérium** | **1** | **2** | **3** |
|---|---|---|---|
| 2.2.1 | ANO | ANO | ANO |
| 2.2.2 | ANO | ANO | ANO |

Tabulka 4.8: Analýza WCAG 2.0 – 2.2 Dostatek času

- Záchvaty – *Vynechte z prezentace takové prvky, u nichž je známo, že mohou vyvolat záchvat.* Viz tabulka 4.9

  Ani jedna z analyzovaných částí webových stránek neobsahuje prvky, které by mohly vyvolat záchvat.

| **Kontrolní kritérium** | **1** | **2** | **3** |
|---|---|---|---|
| 2.3.1 | ANO | ANO | ANO |

Tabulka 4.9: Analýza WCAG 2.0 – 2.3 Záchvaty

- Snadná navigace – *Usnadněte uživatelům navigaci, hledání konkrétního obsahu a určování aktuální pozice.* Viz tabulka 4.10

  Ani jedna z šablon (Front, Admin), které byly podrobeny analýze, neobsahovala odkaz umožňující přeskočení prvků opakujících se na každé stránce.

| **Kontrolní kritérium** | **1** | **2** | **3** |
|---|---|---|---|
| 2.4.1 | NE | NE | NE |
| 2.4.2 | ANO | ANO | ANO |
| 2.4.3 | ANO | ANO | ANO |
| 2.4.4 | ANO | ANO | ANO |

Tabulka 4.10: Analýza WCAG 2.0 – 2.4 Snadná navigace

#### 4.1.5.3 Srozumitelnost

- Čitelné – *Ujistěte se, že textový obsah je čitelný a srozumitelný.* Viz tabulka 4.11

- Intuitivní – *Ujistěte se, že vzhled a ovládání vašich stránek je intuitivní.* Viz tabulka 4.12

| **Kontrolní kritérium** | **1** | **2** | **3** |
|---|---|---|---|
| 3.1.1 | ANO | ANO | ANO |

Tabulka 4.11: Analýza WCAG 2.0 – 3.1 Čitelné

| **Kontrolní kritérium** | **1** | **2** | **3** |
|---|---|---|---|
| 3.2.1 | ANO | ANO | ANO |
| 3.2.2 | ANO | ANO | ANO |

Tabulka 4.12: Analýza WCAG 2.0 – 3.2 Intuitivní

- Pomoc při zadávání – *Pomozte uživatelům vyvarovat se chyb nebo chyby opravit.*

  Kritéria 3.3.1 a 3.3.2 byla analyzována na stránkách přihlášení a registrace obsahující formuláře.

| **Kontrolní kritérium** | **1** | **2** | **3** |
|---|---|---|---|
| 3.3.1 | ANO | ANO | ANO |
| 3.3.2 | ANO | ANO | ANO |

Tabulka 4.13: Analýza WCAG 2.0 – 3.3 Pomoc při zadávání

#### 4.1.5.4 Robustnost

- Kompatibilní – *Snažte se o maximální kompatibilitu se současnými i budoucími přístupovými zařízeními včetně asistivních technologií.* Viz tabulka 4.14

  Splnění kritéria 4.1.1 a 4.1.2 závisí především na validitě, aplikace byla shledána validní dle HTML5 doctype.

| **Kontrolní kritérium** | **1** | **2** | **3** |
|---|---|---|---|
| 4.1.1 | ANO | ANO | ANO |
| 4.1.2 | ANO | ANO | ANO |

Tabulka 4.14: Analýza WCAG 2.0 – 4.1 Kompatibilní

#### 4.1.5.5 Shrnutí

Analýza systému dle metodiky WCAG 2.0 neodhalila žádné závažné nedostatky, které by nebylo možné jednoduše odstranit. Dalším krokem pro celkové zlepšení přístupnosti vytvořené aplikace by mohla být implementace specifikace ARIA. WAI-ARIA si klade za cíl zpřístupnit dynamicky tvořený obsah a na rozdíl od nových HTML5 tagů je široce podporována i asistivními technologiemi (JAWS, NVDA, ZoomTex, ...). (23)

## 4.2 Další vývoj

Další vývoj aplikace závisí především na spokojenosti uživatelů. V první řadě je nutné zapracovat poznatky z výsledků testování.

V první fázi je nutné zavést schéma podpory, to znamená pověřit člověka odpovědného za rychlé odstraňování vzniklých chyb zachycených aplikací či reportovaných uživateli.

Před oficiálním spuštěním či propagací je nezbytné provést code review všech součástí aplikace, především pak frontendové části.

Pokud bude aplikace stabilní a bude získávat na popularitě (bude se zvyšovat návštěvnost a především počet aktivních uživatelů), je možné přemýšlet o dalším rozvoji. Především propagaci a rozšíření funkčnosti API pro aplikace třetích stran.

Další možností je vytvoření vlastní mobilní aplikace. K tomu lze využít buď technologie PhoneGap, nebo je možné pustit se do vývoje nativních aplikací pro Android a iOS.

Nebude-li aplikace mezi uživateli úspěšná, pak je možné zvolit některou z exitových strategií. Pro eventuálního kupce či následovníka vývoje by mohla být zajímavá možnost získání registrovaných uživatelů a možnost zpoplatnění některých funkcí (platba za větší počet souborů, korektur či nabídek; cena přiřazených tagů k nabídce; zvýrazňování nabídek; aj.). Bude-li aplikace úspěšná, pak bude možné zachovat její původní neziskovou koncepci.

# Závěr

Cílem této práce bylo zpracování analýzy a návrhu webové aplikace pro podporu korektur textu. V úvodní části byl popsán přehled řešené problematiky a to prostřednictvím definicí a vysvětlením termínů souvisejících s korekturami textu – dělení korektur, možnosti využití počítačových programů, gramatika, pravopis, stylistika, typografie, polygrafie a jiné. Dále byla provedena rešerše existujících řešení, která poukázala na nedostatky v nabídce a poptávce na současném internetu.

Vlastní řešení začalo analýzou a návrhem aplikace. Byli určeni základní aktéři systému, dále byly specifikovány požadavky nefunkční i funkční, které byly zaznamenány v podobě diagramů případů užití pro jednotlivé role. Součástí návrhu bylo i vytvoření struktury systému a popis uživatelského rozhraní. Nezbytnou součástí je také analýza požadavků na zabezpečení webové aplikace.

Na základě analýzy požadavků byl vytvořen seznam použitých technologií, které jsou podrobně popsány v příslušné kapitole.

Implementace popisuje samotný vytvořený systém spolu s postupem, jakým vývoj probíhal a jakým by měl i nadále pokračovat. Je zde popsána struktura backendu aplikace, jednotlivé vytvořené moduly, použité komponenty Nette Framework, nastavení modelu aplikace pomocí ORM a v neposlední řadě i způsob zpracování frontendové části aplikace.

Objektivní zhodnocení aplikace popisují výsledky kapitoly zaměřené na testování. Aplikace byla průběžně testována během vývoje. Závěrečné testování bylo zaměřeno z funkčního hlediska na správné zobrazení ve webových prohlížečích, zabezpečení proti útokům a analýzu přístupnosti dle metodiky WCAG 2.0. Uživatelské testování proběhlo ve formě testování použitelnosti s reálnými uživateli. Výsledky testování jsou pozitivní, odhalené nedostatky nejsou závažné a byly popsány v závěru spolu s postupy, jak tyto problémy odstranit.

Aplikaci je tedy možné označit za pokrývající požadavky uživatelů na online nabídku a poptávku korektorských prací. Nedostatky, odhalené během testování, byly vzhledem k zavedeným procesům vývoje rychle odstraněny.

Závěrem je dobré říci, že ačkoliv vytvořená aplikace nabízí řadu možností jak si vybrat kvalitního a kvalifikovaného korektora, tak i korektor je jen chybující člověk (jako každý jiný) a není od věci rčení „Důvěřuj, ale prověřuj“.

# Literatura


(1) ARLOW, J.; NEUSTADT, I.: *UML a unifikovaný proces vývoje aplikací*. Brno: Computer Press, první vydání, 2003, ISBN 80-722-6947-X.

(2) BAAROVÁ, H.; PTÁČKOVÁ, P.: Korektura textu: Programy využitelné pro provedení korektur v textových editorech. [online], 2009, [cit. 2013-11-16]. Dostupné z WWW: <http://ikorektura.estranky.cz/clanky/programy-vyuzitelne-pro-provedeni-korektur-v-textovych-editorech.html>

(3) BAAROVÁ, H.; PTÁČKOVÁ, P.; GUBÁNIOVÁ, H.; aj.: LaTeX– Korektura textu. [online], 5 2010, [cit. 2013-11-16]. Dostupné z WWW: <http://geo3.fsv.cvut.cz/vyuka/kapr/sp/2010/varys/baarova_ptackova_gubaniova_klimek.pdf>

(4) BERGMANN, S.: Chapter 4. Writing Tests for PHPUnit. [online], 2013, [cit. 2013-11-16]. Dostupné z WWW: <http://phpunit.de/manual/current/en/writing-tests-for-phpunit.html>

(5) Biglot s. r. o.: [online], [cit. 2013-11-16]. Dostupné z WWW: <http://www.jazykovalaborator.cz/gramatika/>

(6) Česká republika: Zákon o znalcích a tlumočnících. [online], § 24, odst. 2 vyhlášky 37/1967 Sb. Dostupné z WWW: <http://www.zakonyprolidi.cz/cs/1967-36>

(7) Dobrý web, s. r. o: Webová přístupnost. [online], [cit. 2013-11-16]. Dostupné z WWW: <http://www.pristupnost.cz/o-pristupnosti/>

(8) GORTON, I.: *Essential software architecture*. Berlin: Springer, druhé vydání, 2006, ISBN 35-402-8713-2.

(9) GRUDL, D.: Nette Framework: MVC & MVP. [online], 3 2009, [cit. 2013-11-16]. Dostupné z WWW: <http://www.zdrojak.cz/clanky/nette-framework-mvc-mvp/>

(10) GRUDL, D.: SASS, LESS, Stylus nebo čisté CSS? (1). [online], 4 2012, [cit. 2013-11-16]. Dostupné z WWW: <http://phpfashion.com/sass-less-stylus-nebo-ciste-css-1>

(11) GUTMANS, A.: *Mistrovství v PHP 5*. Brno: Computer Press, druhé vydání, 2007, ISBN 978-80-251-1519-0.

(12) Karolinum: [online], 2013, [cit. 2013-11-16]. Dostupné z WWW: <http://www.cupress.cuni.cz/ink2_stat/pages/files/korekturni_znacky.pdf>

(13) KELNAR, M.: Responzivní webdesign komplexně. [online], 4 2012, [cit. 2012-10-13]. Dostupné z WWW: <http://blog.martinkelnar.cz/responsivni-webdesign-komplexne/>

(14) KOČIČKA, P.; BLAŽEK, F.: *Praktická typografie*. Brno: Computer Press, druhé vydání, 2004, ISBN 80-722-6385-4.

(15) KOFLER, M.: *Mistrovství v MySQL 5*. Brno: Computer Press, první vydání, 2007, ISBN 978-80-251-1502-2.

(16) KOPTA, M.: Zákon o informačních systémech veřejné správy. [online], 2006, [cit. 2013-11-16]. Dostupné z WWW: <http://i.iinfo.cz/urs-att/dobryweb-pristupnost-invex-2006-116066709524072.ppt>

(17) KRUG, S.: *Nenuťte uživatele přemýšlet!* Computer Press, první vydání, 2003, ISBN 80-722-6892-9.

(18) MALÝ, M.: REST: architektura pro webové API. [online], 8 2009, [cit. 2013-11-16]. Dostupné z WWW: <http://www.zdrojak.cz/clanky/rest-architektura-pro-webove-api/>

(19) MLČOCH, M.: Stylistika – nauka o slohu. [online], 2009, [cit. 2013-11-16]. Dostupné z WWW: <http://kcjl2.upol.cz/mlcoch/P-23.11.doc>

(20) Nette Foundation: Kdo používá Nette Framework? [online], 2008, [cit. 2013-11-16]. Dostupné z WWW: <http://nette.org/cs/kdo-pouziva-nette-framework>

(21) Nette Foundation: Konfigurace. [online], 2008, [cit. 2013-11-16]. Dostupné z WWW: <http://doc.nette.org/cs/configuring>

(22) Nette Foundation: MVC aplikace & presentery. [online], 2008, [cit. 2013-11-16]. Dostupné z WWW: <http://doc.nette.org/cs/configuring>

(23) PAVLÍČEK, R.: Přístupnost dynamických webových aplikací. [online], 5 2009, [cit. 2013-11-16]. Dostupné z WWW: <http://www.zdrojak.cz/clanky/pristupnost-dynamickych-webovych-aplikaci-uvod/>

(24) Refsnes Data: Browser Statistics. [online], 10 2013, [cit. 2013-11-16]. Dostupné z WWW: <http://www.w3schools.com/browsers/browsers_stats.asp>

(25) REJNKOVÁ, P.: Diagram případů užití. [online], 2009, [cit. 2013-11-16]. Dostupné z WWW: <http://uml.czweb.org/pripad_uziti.htm>

(26) RYBÁK, Z.; PAVLÍČEK, R.: WCAG - český překlad metodiky WCAG 2.0. [online], 2009, [cit. 2013-11-16]. Dostupné z WWW: <http://www.blindfriendly.cz/wcag20/>

(27) STŘÍŽ, P.; STŘÍŽ, M.: *Dokumentace: Velké makro, aneb, Nový způsob zpracování korektur*. Bučovice: Martin Stříž, první vydání, 2008, ISBN 978-808-7106-143.

(28) TICHÝ, J.: Pět vrstev modelu. [online], 4 2010, [cit. 2013-11-16]. Dostupné z WWW: <http://www.phpguru.cz/clanky/pet-vrstev-modelu>

(29) Úřad pro technickou normalizaci, metrologii a státní zkušebnictví: Normy průmyslu polygrafického. [online], 2013, [cit. 2013-11-16]. Dostupné z WWW: <http://www.unmz.cz/urad/unmz>

(30) Ústav pro jazyk český Akademie věd ČR, v. v. i.: Internetová jazyková příručka. [online], 2013, [cit. 2013-11-16]. Dostupné z WWW: <http://prirucka.ujc.cas.cz/>

(31) VINŠ, M.; ZLATNÍKOVÁ, L.: GOOGLE vs. SEZNAM.CZ. [online], 2012, [cit. 2013-11-16]. Dostupné z WWW: <http://www.doba-webova.com/cs/google-vs-seznam>

(32) VRÁNA, J.: *1001 tipů a triků pro PHP*. Brno: Computer Press, první vydání, 2010, ISBN 978-80-251-2940-1.

(33) W3C: Web Content Accessibility Guidelines (WCAG) 2.0. [online], 2008, [cit. 2013-11-16]. Dostupné z WWW: <http://www.w3.org/TR/WCAG20/>

(34) ZIMBER, C.: Lab Usability Testing. [online], 2013, [cit. 2013-11-16]. Dostupné z WWW: <http://teced.com/services/usability-testing-and-evaluation/lab-usability-testing/>

Příloha A

# Seznam použitých zkratek

**AJAJ** Asynchronous JavaScript and JSON

**AJAX** Asynchronous JavaScript and XML

**API** Application Programming Interface

**ARIA** Accessible Rich Internet Applications

**BSD** Berkeley Software Distribution

**CI** Continuous Integration

**CMS** Content Management System

**CRUD** Create, Read, Update, Delete

**CSRF** Cross-Site Request Forgery

**CSS** Cascading Style Sheets

**ČSN** Česká státní norma

**DAO** Data Access Object

**FAQ** Frequently Asked Questions

**FPD** Full Path Disclosure

**GPL** General Public License

**GUI** Graphical User Interface

**HTTP** Hypertext Transfer Protocol

**IDE** Integrated Development Environment

**ISO** International Organization for Standardization

**JAWS** Job Access With Speech

**JS** JavaScript

**JSON** JavaScript Object Notation

**LAMP** Linux Apache MySQL PHP

**LESS** Leaner CSS

**MIT** Massachusetts Institute of Technology

**MVC** Model-View-Controller

**MVP** Model-View-Presenter

**NVDA** NonVisual Desktop Access

**ON** Oborová norma

**OOP** Object-Oriented Programming

**ORM** Object-Relational Mapping

**OS** Operating System

**OSS** Open-Source Software

**PČP** Pravidla českého pravopisu

**PHP** Hypertext Preprocessor

**REST** Representational State Transfer

**RPC** Remote Procedure Call

**RSS** Rich Site Summary

**SASS** Syntactically Awesome Style Sheets

**SEM** Search Engine Marketing

**SEO** Search Engine Optimization

**SOAP** Simple Object Access Protocol

**SQL** Structured Query Language

**UI** User Interface

**UML** Unified Modeling Language

**URI** Uniform Resource Identifier

**URL** Uniform Resource Locator

**URN** Uniform Resource Name

**UX** User Experience

**W3C** World Wide Web Consortium

**WAI** Web Accessibility Initiative

**WCAG** Web Content Accessibility Guidelines

**WYSIWYG** What You See Is What You Get

**XHTML** Extensible HyperText Markup Language

**XML** Extensible Markup Language

**XSS** Cross-Site Scripting

Příloha B

# Instalační příručka

Tato příručka by měla sloužit k bezproblémové instalaci systému včetně všech modulů.

## B.1 Minimální požadavky

Minimální požadavky vycházejí z požadavků použitých technologií a to především z aktuálně použité verze Nette Framework 2.1.0. Požadavky a doporučení jsou uvedeny v tabulce B.1.

Všechny tyto uvedené požadavky pokrývají i požadavky Orm, migrací i testů. Orm navíc vyžaduje MySQL 4.1 a vyšší.

## B.2 Instalace

1. Zkopírujte obsah repozitáře na server;
2. Pokud server podporuje práci s composerem, spusťte příkaz `composer update -pref-dist`, v opačném případě je nutné spustit tento příkaz na lokálním serveru, který má nainstalovaný composer.
3. Spustit Grunt (`grunt` pro produkční build, `grunt dev` pro vývojový režim), který aktualizuje frontendové knihovny pomocí Boweru a pro produkční build minifikuje a sloučí JavaScriptové soubory a kaskádové styly.
4. Nastavte přístupová práva pro zápis do složek `temp` a `log` a to ve složce aplikace a ve složce testů. Pro změnu můžete použít příkaz `chmod -R a+rwX temp log`.
5. Spusťte migrační skripty pomocí http://www.example.com/migrations/run.php.

| | |
|---|---|
| PHP verze | 5.3.7 |
| .htaccess | Povolen |
| mod_rewrite | Povolen |
| Funkce ini_set() | Povolena |
| Fukce error_reporting() | Povolena |
| Function flock() | Vyžaduje cache |
| Register_globals | Nebezpečná konfigurační direktiva PHP, která musí být vypnutá |
| Zend.ze1_compatibility_mode | Kompatibilita s PHP 4, musí být vypnutá |
| Session auto-start | Z bezpečnostních důvodů je doporučeno nepoužívat |
| Reflection extension | Extenze PHP vyžadovaná frameworkem |
| SPL extension | Extenze PHP vyžadovaná frameworkem |
| PCRE extension | Extenze PHP vyžadovaná frameworkem |
| ICONV extension | Extenze PHP vyžadovaná frameworkem |
| PHP tokenizer | Extenze PHP vyžadovaná frameworkem |
| PDO extension | Extenze PHP vyžadovaná Nette\Database |
| Multibyte String extension | Extenze PHP vyžadovaná funkcemi Strings::lower() a upper() |
| Multibyte String function overloading | Nebezpečná konfigurační direktiva PHP, musí být vypnutá |
| Memcache extension | Extenze PHP podporovaná úložištěm cache |
| GD extension | Extenze PHP vyžadovaná Nette\Image |
| Bundled GD extension | Extenze PHP vyžadovaná metodami Nette\Image::filter() a rotate() |
| Fileinfo extension or mime_content_type() | Funkce používané k MIME-type detekci uploadovaných souborů |

Tabulka B.1: Minimální požadavky Nette Framework 2.1.0

6. Spusťte testy aplikace na adrese http://www.example.com/tests/

Příloha C

# Obsah přiloženého CD