Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informačních technologií

Webová prezentace pacientské organizace

Bakalářská práce

Autor: Bc. Vítězslav Kaňok
Studijní obor: Aplikovaná informatika

Vedoucí práce: Mgr. Daniela Ponce, Ph.D.

Hradec Králové Duben 2020

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

vlastnoruční podpis

V Hradci Králové dne Vítězslav Kaňok

Poděkování:

Děkuji své vedoucí bakalářské práce paní Mgr. Daniela Ponce, Ph.D. za metodické vedení práce, čas a připomínky. Také bych rád poděkoval mé rodině a přátelům za pomoc a podporu během studia.

Anotace

Bakalářská práce pojednává o analýze, návrhu a implementaci webové aplikace pro spolek S-Let zabývající se pomocí lidem po operaci odstranění hrtanu neboli laryngektomii. Aplikace je naprogramovaná v programovacím jazyce Java s využitím frameworku Spring Boot ve verzi 2.0. Aplikace je vytvořena na základě požadavků zadavatele, jenž byly v prvé řadě diskuzní fórum, kde se mohou lidé, kteří se o problematiku laryngektomii zajímají, zeptat odborníků a získat tak důvěryhodné informace. Druhým důležitým požadavkem bylo, aby si mohl administrátor webu, který nemá znalost webových technologií, sám spravovat obsah webu. Tato funkcionalita mu je umožněna přes uživatelsky přívětivé rozhraní v administrační části webu. Poslední část práce popisuje implementaci celé aplikace, včetně implementace databáze PostgreSQL s pomocí rozhraní JPA a frameworku Hibernate.

Klíčová slova: Framework, Hibernate, Java, JPA, Laryngektomie, Spring Boot, Webová aplikace, WWW

Annotation

Title: Web presentation for patient organization

The bachelor thesis deals with the analysis, design and implementation of a web application for the S-Let association, which deals with helping people after laryngeal removal or laryngectomy. The application is programmed in the Java programming language using the Spring Boot framework in version 2.0. The application is created based on the requirements of the client, that includes first, a discussion forum where people who are interested in the issue of laryngectomy can ask experts and get credible information. The second important requirement was based on the ability of the web admin, who does not have knowledge of web technologies, can manage the content of the website by himself. This is possible through the user-friendly interface in the administration part of the website. The last part of the thesis describes the implementation of the entire application, including the implementation of the PostgreSQL database using the JPA interface and the Hibernate framework.

Keywords: Framework, Hibernate, Java, JPA, Laryngectomy, Spring Boot, Web Application, WWW

Obsah

Seznam obrázků

1 Úvod

Cílem této práce je vytvoření webové aplikace pro orientaci a získávání informací ohledně onemocnění laryngektomie ať již samotným pacientům nebo široké veřejnosti. Tato webová aplikace tak bude pomáhat převážně nemocným lidem s vyhledáváním informací ohledně tohoto onemocnění, dále bude možné v diskuzích mluvit s ostatními pacienty, nebo pokládat dotazy přímo odborníků z praxe, kteří budou ověřeni administrátorem webu. Dále budou na webu vystaveny základní informace o této nemoci a možnosti následné rekonvalescence, nebo způsoby jak i po operaci, alespoň v rámci možností, mluvit.

Webová aplikace bude umožňovat přístup přihlášeným i nepřihlášeným uživatelům. Samozřejmě uživatelé přihlášení budou mít více možností pro využití daného webu. Celá práce se tak bude věnovat právě vytváření této aplikace. Na začátku této práce jsou vymezeny základní funkční a nefunkční požadavky zadavatele. Následující kapitola, s názvem případy užití, se věnuje právě možnostem jednotlivých uživatelů webu. Vymezeny jsou také jejich role.

Další velmi stěžejní částí práce je analýza. V této části je popsána analýza projektu, která zahrnuje objevování a seznam funkčních a nefunkčních požadavků a analýz podobných aplikací a potenciální konkurence obecně.

Následují kapitola popisuje návrh aplikace včetně databáze. Aplikace je založena na architektuře MVC. Projekt obsahuje 2 hlavní balíčky, java a resources.

Předposlední část práce se věnuje použitým technologiím vytvořené aplikace, jedná se tak o technologie frontendové, tedy HTML, CSS, Bootstrap, Thymeleaf, JavaScript, ale i backendové, jako Java, Spring Boot, JUnit, JPA, Hibernate a PostgreSQL. U všech těchto technologií jsou uvedeny základní informace. Poslední kapitola implementace představuje ukázky kódu důležitých metod v aplikaci.

2 Požadavky zadavatele

Ve druhé kapitole této práce jsou uvedeny požadavky, které jsou ze stran zadavatele kladeny na nově vytvořenou webovou aplikaci.

V případě aplikací je velmi zásadní splnit všechny požadavky, které jsou zadavatelem po vývojáři požadovány. Na jednotlivé požadavky je následně nutné hledět ze dvou směrů, a to ze strany funkčních a nefunkčních požadavků celého webu. Zadavatel této aplikace si vyžádal webovou aplikaci pro spolek S-Let zabývající se pomoci lidem po operaci odstranění hrtanu neboli laryngektomii. Jeho hlavní požadavky byly diskuzní fórum, jeho administrace a možnost správy obsahu různých částí webu i bez znalosti webových technologií.

Fórum bude rozděleno do kategorií a v každé kategorii budou odpovídat na otázky specialisté z daného oboru. Aplikace bude obsahovat různé informace o tom, jak ses stát členem organizace, co členství přináší a informace o nadcházejících i proběhlých setkáních spolku. Jednotlivé níže uvedené požadavky jsou rozděleny na požadavky funkční, kterými jsou zabezpečení, registrace, diskuzní fórum, správa fóra, novinky a zvýšení povědomí o doméně spolku S-LET. Druhou část pak tvoří požadavky tzv. nefunkční, kde jsou zařazeny dostupnost, stabilita, přehlednost a také v neposlední řadě bezpečnost.

2.1 Funkční požadavky

Zabezpečení

Jedná se o jeden ze základních funkčních požadavků na aplikaci. Zabezpečení systému bude implementováno pomocí frameworku Spring Boot, konkrétně jeho části Spring Security. Tato část frameworku usnadňuje implementaci zabezpečení a umožňuje širokou škálu customizace, jako je nastavení rolí, autorit, práv a zabezpečení jednotlivých stránek či správu cookies. V případě potřeby umožnuje i dvoufázové ověřování a OAuth2, což je protokol pro autentizaci a autorizaci přes API Googlu, Facebooku a dalších internetových služeb.

Registrace

Požadavkem také bylo, aby všichni uživatelé navštěvující web a mají zájem se zúčastnit diskuze, se mohli registrovat do aplikace a tím jim bude umožněno například prohlížení diskuzního fóra, kde budou moct přidávat své komentáře a dotazy.

Diskuzní fórum

Zásadním požadavkem zadavatele bylo vytvoření diskuzního fóra, zejména pro lepší komunikaci s lidmi po laryngektomii, či pro ostatní osoby, například které zákrok v blízké době budou muset podstoupit, a tudíž mají mnoho otázek. Fórum bude mít několik kategorií, které budou rozděleny na základě typů otázek a odborníků, kteří na ně budou odpovídat. V každé kategorii může uživatel zadávat dotazy. Odpovědi odborníku budou odlišeny od odpovědí běžných uživatelů, čímž se získá vyšší přehlednost.

Správa fóra

Součástí fóra bude také jeho administrace. Tuto funkci vykonává uživatel, který je přihlášen v roli ADMIN a má tedy možnost spravovat jednak aplikaci jako takovou, ale bude možné díky němu zajistit správu všech součástí fóra od kategorií fóra až po jednotlivé uživatele webu.

Novinky

Jedná se o aktuality přidávané administrátorem webu. Administrátor je bude moci přidávat bez znalostí webových technologií přes uživatelské rozhraní. Byl to také velice zásadní požadavek zadavatele, protože si chtěl svůj obsah na webu spravovat sám.

Zvýšení povědomí

Dalším z požadavků zadavatele bylo rozšířit mezi lidmi povědomí o této aplikaci a jejich funkcích. Díky rozšíření povědomí a většímu počtu návštěvníků, kteří ví o této aplikaci, bude možné dosáhnout větší informovanosti populace.

2.2 Nefunkční požadavky

Hlavními nefunkčními požadavky aplikace byly její dostupnost, stabilita, bezpečnost a přehlednost.

Dostupnost

Dostupnost byl jeden ze zásadních požadavků zejména s ohledem na to, že tento web bude poskytovat důležité informace pro lidi v těžké životní situaci. Web by tak měl být dostupný pro všechny a informovanost, že takový web existuje by měla být rozšířena do povědomí i zdravých lidí v co nejvyšší míře.

Stabilita

Tento nefunkční požadavek úzce souvisí se samotnou dostupností. V případě, kdy se uživatel v nelehké životní situaci nebude moci na web dostat, například z důvodu, že web bude vypadávat, nebude chtít tyto stránky již v budoucnu využívat.

Bezpečnost

Bezpečnost aplikace by měla být ošetřena s ohledem na její funkcionalitu. Tato webová aplikace bude obsahovat diskuzní fórum, takže hlavní nebezpečí hrozí v případě útoků typu SQL injection. Tato technika napadení databáze spočívá v tom, že uživatel (hacker) vloží do formuláře nějaký SQL dotaz, kterým může změnit nebo i vymazat obsah databáze.

Přehlednost

Přehlednost byl poslední nefunkčním požadavkem pro tuto aplikaci. Vzhledem k tomu, že obsah aplikace budou využívat také starší osoby, které nemusí být ve všech případech zcela zdatné v požívání takovýchto webů, bude nutné zajistit co nejvyšší přehlednost a co nejjednodušší ovládání.

3 Případy užití

V této kapitole jsou charakterizovány jednotlivé případy užití dané aplikace a webu jako takového. Jednotlivé užití webu závisí na roli uživatele, zda se jedná o nepřihlášeného uživatele, přihlášeného uživatele, odborníka nebo administrátora webu. Všechny tyto čtyři kategorie a jejich užití webu jsou uvedeny v následující podkapitole.

3.1 Uživatelské role

V této podkapitole je uvedeno základní rozdělení jednotlivých uživatelů, kteří daný web mohou navštěvovat a využívat.

První kategorií jsou běžní návštěvníci webu, kteří nejsou přihlášeni do webové aplikace. Tito návštěvníci tedy mají jen velmi omezené možnosti využívání webu a nemohou například vstupovat do diskuzního fóra.

Druhou kategorií jsou přihlášení uživatelé. Tito uživatelé prošli procesem registrace a mají zájem se podílet na vytváření témat k diskuzi na webu a nejsou tedy až tak anonymní jako výše uvedení nepřihlášení uživatelé.

Třetí kategorii tvoří odborníci na problematiku týkající se laryngektomie. Mezi tyto odborníky lze zařadit lékaře a logopedy. Tito odborníci odpovídají na jednotlivé dotazy a vzhledem k jejich vzdělání a zkušenostem jsou jejich odpovědi označeny za odborné.

Poslední kategorií jsou administrátoři webu. Ti se starají o správný chod aplikace, přidávání a úpravu obsahu stránek. V případě potřeby zasahují do diskuzí či příspěvků, to však pouze v případech, kdy se jedná o porušení určitých pravidel na webu, jako jsou vulgarismy, výsměch nemocným lidem apod.

3.1.1 Nepřihlášený uživatel

Pokud vstoupí nepřihlášený uživatel na webovou stránku, zobrazí se mu celá nabídka možností. Uživatel si zde může vybrat mezi úvodem, komunikací, informacích o členech spolku, jak se stát členem spolku a s tím související možnost stažení přihlášky do spolku, jsou zde pro něj k nahlédnutí jednotlivé stanovy spolku,

dále také informace o činnosti spolku, jako jsou zpravodaj, novinky a pomůcky, kontakt na logopeda a také má možnost podívat se na partnery a sponzory tohoto spolku.

Velmi zajímavou část zde k nahlédnutí pro všechny uživatele tvoří kategorie komunikace. Zde jsou uvedeny jednotlivé možnosti, jak po operaci může pacient alespoň částečně komunikovat. Těmito možnostmi jsou jícnový hlas, hlasová protéza nebo elektrolarynx. U všech těchto pomocníků ke komunikaci jsou uvedeny jejich výhody a nevýhody. Pro nepřihlášeného uživatele zde není možnost navštívit fórum.

3.1.2 Přihlášený uživatel

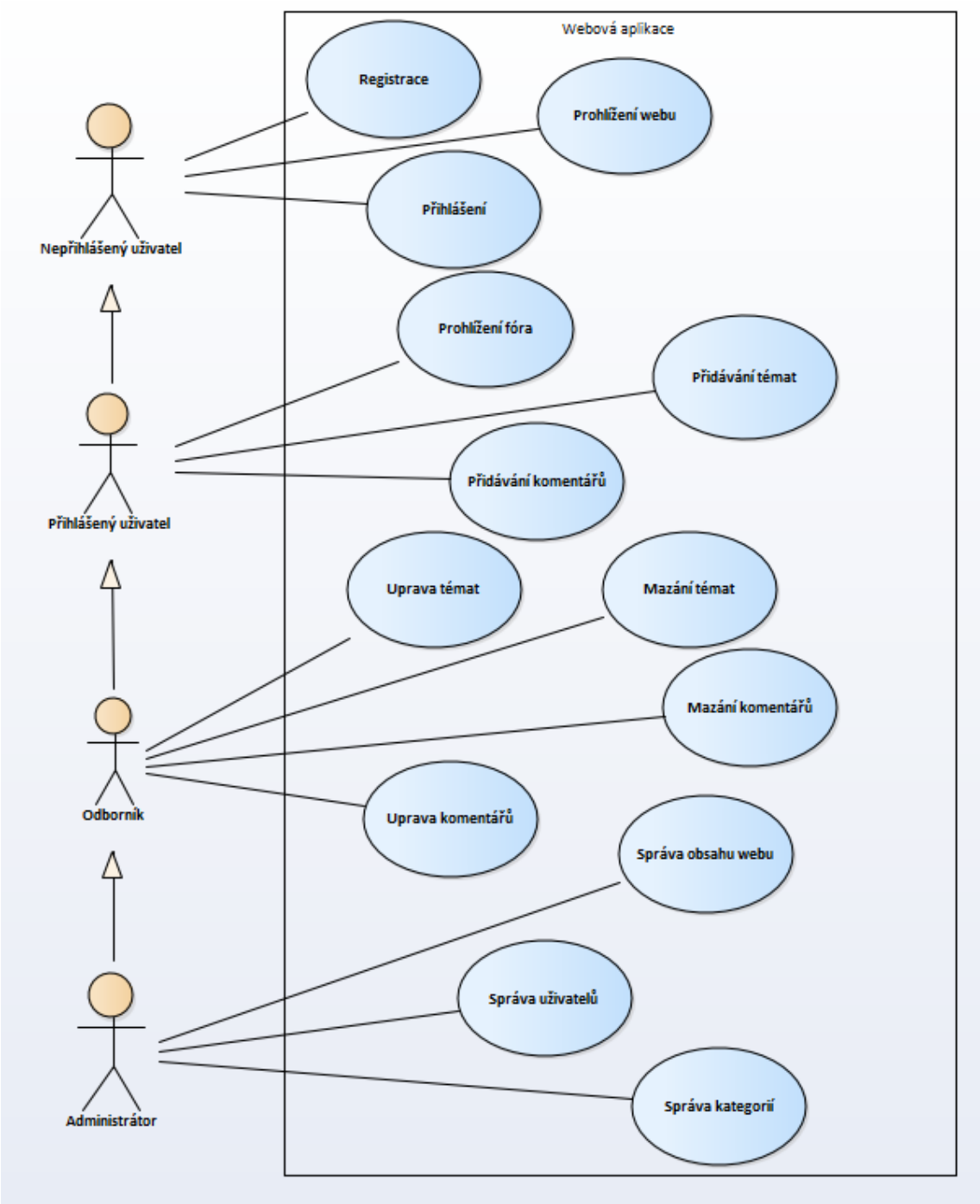
Uživatel, který prošel registrací, má možnost využívat jednotlivé stránky webu stejně jako uživatel nepřihlášený. Velkou výhodou však je návštěva fóra, kde může zakládat nová témata, ale také si může přečíst všechny předchozí dotazy a jejich odpovědi od ostatních uživatelů.

3.1.3 Odborník

U odborníka jsou opět všechny funkce stejné jako u přihlášeného uživatele. Jelikož se ve všech případech musí jednat o osobu kompetentní k odpovídání na dané otázky, je odborníkovi umožněn vstup do jednotlivých diskuzí a jeho odpovědi jsou ve fóru zvýrazněny a označeny jako odborné. Vzhledem k jeho znalostem může také mazat a upravovat jednotlivé příspěvky a témata.

3.1.4 Administrátor

Tento uživatel má opět stejné možnosti jako odborník, ale zároveň jako jediný může web upravovat z pohledu obsahu jednotlivých stránek. Jeho hlavním úkolem je pak zabezpečit, aby byl web aktuální z pohledu nových akcí spolku a užívání stránek bylo pro všechny výše uvedené uživatele komfortní a příjemné.



Obrázek 1 - Diagram případů užití. Zdroj: autor

4 Analýza

V této části je popsána analýza, která zahrnuje objevování a seznam jak funkčních, tak nefunkčních požadavků a také analýz podobných aplikací a potenciální konkurence obecně.

4.1 Analýza požadavků zadavatele

Spolek S-Let již před touto webovou aplikací disponoval statických webem. Předchozí web již nesplňoval požadavky spolku, jelikož neobsahoval diskuzní fórum a ani základní SEO. Cílem nové webové aplikace je odstranit tyto nedostatky, zvýšit povědomí o laryngektomii a vylepšit její pozici ve vyhledávání. Analýza začala sezením ve Spolku S-let, které pomohlo nastínit klíčové body aplikace. Základní funkčnost včetně různých úrovní oprávnění a krátkého shrnutí potenciálních zákazníků a způsobů, jak bude těmto uživatelům aplikace nabízena.

Při tomto sezení ve Spolku S-Let byla vytvořena tzv. mind map. Aplikace je určena převážně pro pacienty, kteří podstoupili nebo budou muset podstoupit laryngektomii. Ti se mohou registrovat a po přihlášení přidávat příspěvky a témata na diskuzním fóru. Samozřejmostí je, že všechna data, která jsou při registraci či užívání zjištěna, jsou tzv. izolovanými daty, a tedy nejsou nijak zjistitelná pro jiné uživatele. Správce webu, který tato data vidí, je bez svolení uživatele nikde nesmí uvádět a ani jinak nezneužívat. Součástí těchto standardů je také zašifrování hesel před uložením do databáze.

Základní princip aplikace se točí kolem informovanosti o laryngektomii pro všechny návštěvníky webu a pro zodpovídání dotazů v případě přihlášených uživatelů. Fórum je rozděleno do několika kategorií, a to administrátorem webu a poté jsou viditelné pro všechny přihlášené uživatele aplikace.

4.1.1 Registrace a přihlášení

Aby mohli uživatelé plně využívat celou aplikaci, musí se nejprve zaregistrovat. Registrace do této aplikace je podobná jako u většiny aplikací. Při registrování je nutné uvést uživatelské jméno, email, jméno, příjmení a heslo. Heslo je pro ověření zadáváno dvakrát, aby měl uživatel jistotu, že zadal právě takové heslo, jaké chtěl. Uživatelé také mohou přidat pár nepovinných informací, jako je

jejich fotka či krátký popis jich samotných. V případě že zapomenou heslo, mohou si ho změnit zadáním svého emailu do formuláře pro obnovu hesla a na tento email jim bude zaslán odkaz s tokenem pro změnu hesla.

4.1.2 Diskuzní fórum

Aplikace registrovanému uživateli umožňuje vytvářet nová témata a přidávat do témat komentáře. V případě, že je uživatel v roli odborník, jsou jeho odpovědi odlišeny od běžného uživatele a mají označení odborná odpověď. Odborník má na rozdíl od běžného uživatele právo mazat a upravovat komentáře a témata. Administrátor má navíc právo spravovat kategorie.

4.1.3 Správa obsahu webu

Správa obsahu webu je umožněna pouze administrátorovi přes jednoduché uživatelské rozhraní. Stěžejní části obsahu webu jsou novinky. Novinky se vykreslují do různých stránek jako její obsah, a to na základě o jaký typ novinky se jedná. Typy novinek jsou rozděleny na setkání, konference, oznámení, terapie, sezení a zpravodaj.

4.2 Podobné aplikace a služby

Jak již bylo zmíněno, v současné době existuje několik aplikací, které se snaží usnadnit komunikaci mezi nemocnými. I když žádná z aplikací, které budou uvedeny v této analýze nejsou přímými konkurenty v této oblasti.

4.2.1 Explain TB

Tato aplikace byla nalezena pouze pomocí SZÚ neboli státního zdravotního ústavu, jelikož se jedná o anglickou aplikaci. Aplikace je určena pro všechny nemocné tuberkulózou a jejich blízké. Byla vytvořena Research Center Borstel a Germn Central Commitee afainst Tuberculosis. Aplikace nabízí 41 kapitol s informacemi o tuberkulóze a může Vám pomoct odpovědět na otázky, které byste chtěli znát. Aplikace je zcela zdarma (pro IOS a Android) a na výběr máte mnoho jazyků včetně češtiny a slovenštiny. Nabízí text i audio nahrávky.

4.2.2 Parkinson – Help

Další podobnou aplikací je také Parkinson Help, která je primárně určena pro osoby trpící Parkinsonovou chorobou a pro jejich blízké. Aplikace seznamuje širokou veřejnost s obtížemi Parkinsonovy nemoci, přispívá ke zkvalitnění života lidí s touto chorobou a podporuje komunitní život nemocných v regionech, kde zakládá Parkinson-Help (P-H) kluby. Tyto kluby se nachází v současné době v Praze, Ostravě, Rožnově pod Radhoštěm, na Karlovarsku a v Českém ráji. Tyto kluby se snaží rozvíjet aktivity pro nemocné a jejich rodiny, dále pořádají nejrůznější osvětové akce, vzdělávací workshopy atd.

4.2.3 CFHero

Další aplikací pro nemocné je CFHero, která je určena pro lidi, kteří onemocní cystickou fibrózou. Aplikace učí herní formou jedince postižené cystickou fibrózou cviky, které jim mají prodloužit život. Jednotlivé úkoly pacienta/hráče učí osvojení potřebné každodenní inhalace a dechovou fyzioterapii. Jako měna ve hře slouží kyslíky. Za ty si mohou hráči pořídit pro své avatary různé doplňky, od slunečních brýlí po masku koně, ale i další věci jako například komiksy, které je seznámí se situacemi, v nichž se mohou coby nemocní ocitnout.

4.2.4 Nepanikařte

Jedná se o aplikaci, která je vytvořena za účelem pomoci lidem trpícími depresemi. Uživatelé si zde mohou zvolit mezi čtyřmi základními kategoriemi, které je zrovna postihly, a to jsou deprese, úzkost a panika, chci si ublížit a myšlenky na sebevraždu. V každé z těchto kategorií jsou pak následně rady, které nemocnému mohou pomoci. V aplikace také nechybí kontakt na pomoc. V současné době si aplikaci stáhlo přes 8 tisíc lidé.

V České republice existuje v současné době značné množství aplikací, které jsou vytvořeny za účelem usnadnit nemocným jejich stav. Nejvyužívanější z nich se specializuje na nemoci spojené s psychickými poruchami.

5 Návrh

5.1 Návrh aplikace

Aplikace je založen na architektuře MVC. Tato v dnešní době nejpoužívanější architektura webových aplikací má za cíl rozdělení tří základních modulů, kterými jsou model, view(pohled) a controller(kontroler). MVC architektura umožnuje programovat přehledné a vysoce škálovatelné aplikace, tudíž nenastává problém při přidávání dalších funkcionalit, či škálování aplikace při zvyšování počtu návštěvníků.

V části model aplikace obsahuje základní třídy, které formují její kostru a je na nich celý projekt postaven. Vzhledem k funkčním požadavkům pak tento modul obsahuje logicky vytvořené balíčky *feedback, forum, news* a *security*. Jak už název modulu sám napovídá, v jednotlivých balíčcích jsou modelové třídy, které obsahují atributy a metody potřebné pro jednotlivé části aplikace. Pro diskuzní fórum je nejpodstatnější baliček *forum* ve kterém se nacházejí třídy *Category, Post* a *Topic*. Tyto POJO třídy obsahují atributy, které jsou používány pro aplikační logiku a uloženy do databáze nebo si je aplikace z databáze načte pro vygenerování obsahu nějakého tématu na fóru. Všechny třídy jejichž instance jsou ukládány do databáze mají jako předka třídu *EntityBase*, která třídám implementuje jedinečný identifikátor, datum a čas vytvoření a také datum a čas poslední úpravy instance.

```
@MappedSuperclass
public abstract class EntityBase implements Serializable {

    @Id
    @GeneratedValue
    protected Long id;

    @CreationTimestamp
    private LocalDateTime createDateTime;

    @UpdateTimestamp
    private LocalDateTime updateDateTime;


    protected EntityBase() {
    }
```

Obrázek 2 - Ukázka entitní Java třídy *EntityBase* bez třídních metod. Zdroj: autor

```java
@Entity
public class Topic extends EntityBase {


    @NotBlank
    @Size(min = 2, max = 30, message = "Název musí mít délku 2-30 znaků.")
    private String name;

    @NotBlank
    @Size(min = 5, max = 1000, message = "Text musí mít délku 5-1000 znaků.")
    @Column(length = 1500)
    private String text;


    @ManyToOne(fetch = FetchType.EAGER, optional = false)
    @JoinColumn(name = "users_id", nullable = false)
    @OnDelete(action = OnDeleteAction.CASCADE)
    @JsonIgnore
    private User user;

    @ManyToOne(fetch = FetchType.LAZY)
    private Category category;

    @OneToMany(mappedBy = "topic", cascade = CascadeType.ALL)
    private Collection<Post> posts;
```

Obrázek 3 - Ukázka entitní Java třídy *Topic* bez třídních metod. Zdroj: autor

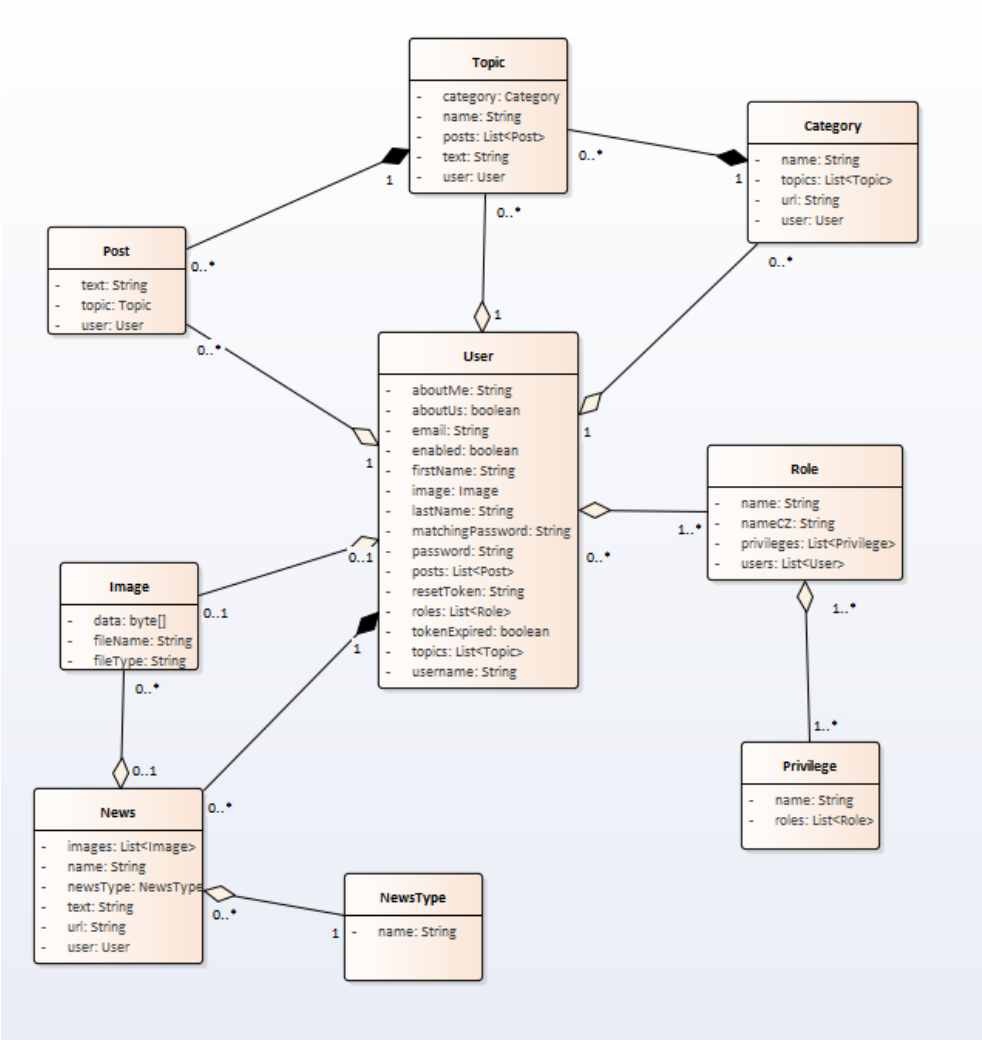
V modulu View se nachází XML dokumenty ve webovém formátu HTML obsahující prvky z Frameworku Thymeleaf, který usnadňuje zobrazování dat z výše zmíněných modelových tříd.

Poslední část architektury, tedy část Controller, propojuje modelovou část s částí pohledovou a určuje výstupy pro uživatele, může také ovlivnit výstup podle toho jakou má uživatel roli nebo jestli je přihlášený. Kontrolery přistupují k datům přes vrstvu Service (Servis), která je důležitá pro přehlednost aplikace, protože obaluje rozhraní repository a díky tomuto zapouzdření je aplikace bezpečnější a přehlednější.

Servisní část aplikace, jak již bylo výše zmíněno, je využívána kontrolery pro přístup k datům, ale také přidává vlastní funkcionalitu. V případě, kdy se

registruje nový uživatel, je nezbytně nutné, aby jeho heslo bylo uloženo v zašifrované podobě. Právě o to se stará servisní vrstva.

Repository část implementuje konkrétní dotazy do databáze.



Obrázek 4 – UML diagram tříd. Zdroj: autor

5.2 Návrh databáze

Většina aplikací potřebuje nějaký mechanismus na ukládání dat. At už se jedná o desktopovou či webovou aplikaci. Nejspolehlivějším a nejpoužívanějším způsobem ukládání dat je do databází. V této aplikaci tomu není výjimkou a data se ukládají do NoSQL databáze, konkrétně PostgreSQL.

Databázové tabulky se díky Java Persistence API generují automaticky na základě POJO tříd, což je velká výhoda využití toho API pro objektově relační mapování. API prozkoumá celý projekt a vyhledá třídy s anotací *@Entity*. Tyto třídy

pak namapuje na databázové tabulky a pomocí datových typů jednotlivých atributů nebo explicitně pomocí anotací na atributech jim přiřadí datové typy. Mezi základní anotace patří *@Id*, která určí, že daný atribut je primární klíč. *@NotNull* zabezpečí, že datová buňka nesmí byt null, ale může obsahovat prázdný řetězec. *@NotEmpty* znamená, že objekt nesmí být null a také jeho velikost musí být vetší než nula. Poslední podobná anotace je *@NotBlank*, která zajistí, že v buňce je text o délce alespoň jednoho znaku.

Datovému typu String můžeme určit jeho délku pomocí anotace *@Size*. U číselných typů jako jsou int, long atd. můžeme určit maximální či minimální hodnotu pomocí *@Min* a *@Max*.

Atributům můžeme také určit, jak se budou v tabulce jmenovat a to pomocí *@Column*. Tato anotace nabízí širokou škálu konfigurace jako přesné definování datového typu či nastavení délky textu.

V projektu mají všechny třídy, jako základ abstraktní třídu *EntityBase*, která implementuje základní redundantní atributy a metody jako jsou například id, datum a čas vytvoření a datum a čas poslední úpravy. Obsahuje taky metodu pro vrácení času vytvoření ve String formátu, ve tvaru, před jakou dobou byla instance vytvořena. Taková abstraktní třída musí mít anotaci *@MappedSuperclass*, která zajištuje, že se pro tuto samostatnou třídu nebude vytvářet tabulka. Pomocí JPA se pak vytváří rozhraní pro CRUD operace nad databází.

Každá databáze potřebuje nějaké vazby tabulek a to jak 1:1, 1:N či M:N. V JPA si můžeme vybrat, zda potřebujeme vazbu unidirectional neboli jednosměrnou nebo bidirectional neboli obousměrnou. Rozdíl mezi těmito vazbami je, zda potřebujeme přistupovat k datům jen z jedné třídy nebo chceme přistupovat k datům obou tříd navzájem.

Vazba 1:1 se v JPA provádí pomocí anotace *@OneToOne* této anotaci pak nastavíme, která třída provádí mapování *(mappedBy = „mapovacíTřída“).* Pomocí atributu *cascade* určíme, jak se mají chovat navázané objekty v případě smazání instance mapovací třídy.

```java
@Entity
@Table(name = "users")
public class User extends EntityBase {

    @OneToOne(cascade = CascadeType.ALL)
    @JoinColumn(name = "image_id", referencedColumnName = "id")
    private Image image;
```

Obrázek 5 - Ukázka anotací třídy *User* při mapování 1:1 ke třídě *Image*. Zdroj: autor

Vazba 1:N se provádí přes anotaci *@OneToMany* respektive *@ManyToOne*. Jak už název napovídá, podle toho, která třída zastává kardinalitu jedna a která kardinalitu mnoho, se přiřadí konkrétnímu atributu výše zmíněná anotace.

```java
@Entity
public class Category extends EntityBase {

    @OneToMany(mappedBy = "category", cascade = CascadeType.ALL)
    private Collection<Topic> topics;
```

Obrázek 6 - Ukázka anotací třídy *Category* při mapování 1:N ke třídě *Topic*. Zdroj: autor

```java
@Entity
public class Topic extends EntityBase {

    @ManyToOne(fetch = FetchType.LAZY)
    private Category category;
```

Obrázek 7 - Ukázka anotací třídy *Topic* při mapování N:1 ke třídě *Category*. Zdroj: autor

Mapování M ku N se provádí pomocí anotace *@ManyToMany* ve které lze nastavit i způsob načítání dat, a to buď Eager, což znamená, že jsou společně s načítanými daty z primární tabulky načteny i data ze všech tabulek obsahující cizí klíč primární tabulky. Načítání dat s atributem Lazy znamená, že se načtou pouze data z primární tabulky a data z cizích tabulek se načítají, až v případě potřeby. Tyto dva způsoby mají samozřejmě svoje výhody a nevýhody. Hlavními výhodami Eager jsou, že při potřebě dat z tabulek s cizím klíčem už nevzniká zpoždění, protože všechna data již byla načtena. Nevýhodou pak je, že prvotní načtení trvá delší dobu

a mohou být načtena i data, která nebudeme používat. U Lazy načítání je to přesně naopak. Prvotní nahrání dat je rychlejší a zabírá méně prostoru v paměti, ale v případě potřeby dat z tabulky s cizím klíčem, vzniká zpoždění z důvodu potřeby načtení dalších dat.

```
@Entity
@Table(name = "users")
public class User extends EntityBase {

    @ManyToMany(fetch = FetchType.EAGER)
    @JoinTable(
            name = "users_roles",
            joinColumns = @JoinColumn(
                    name = "users_id", referencedColumnName = "id"),
            inverseJoinColumns = @JoinColumn(
                    name = "role_id", referencedColumnName = "id"))
    private Collection<Role> roles;
```

Obrázek 8 - Ukázka anotací třídy *User* při mapování M:N ke třídě *Role*. Zdroj: autor

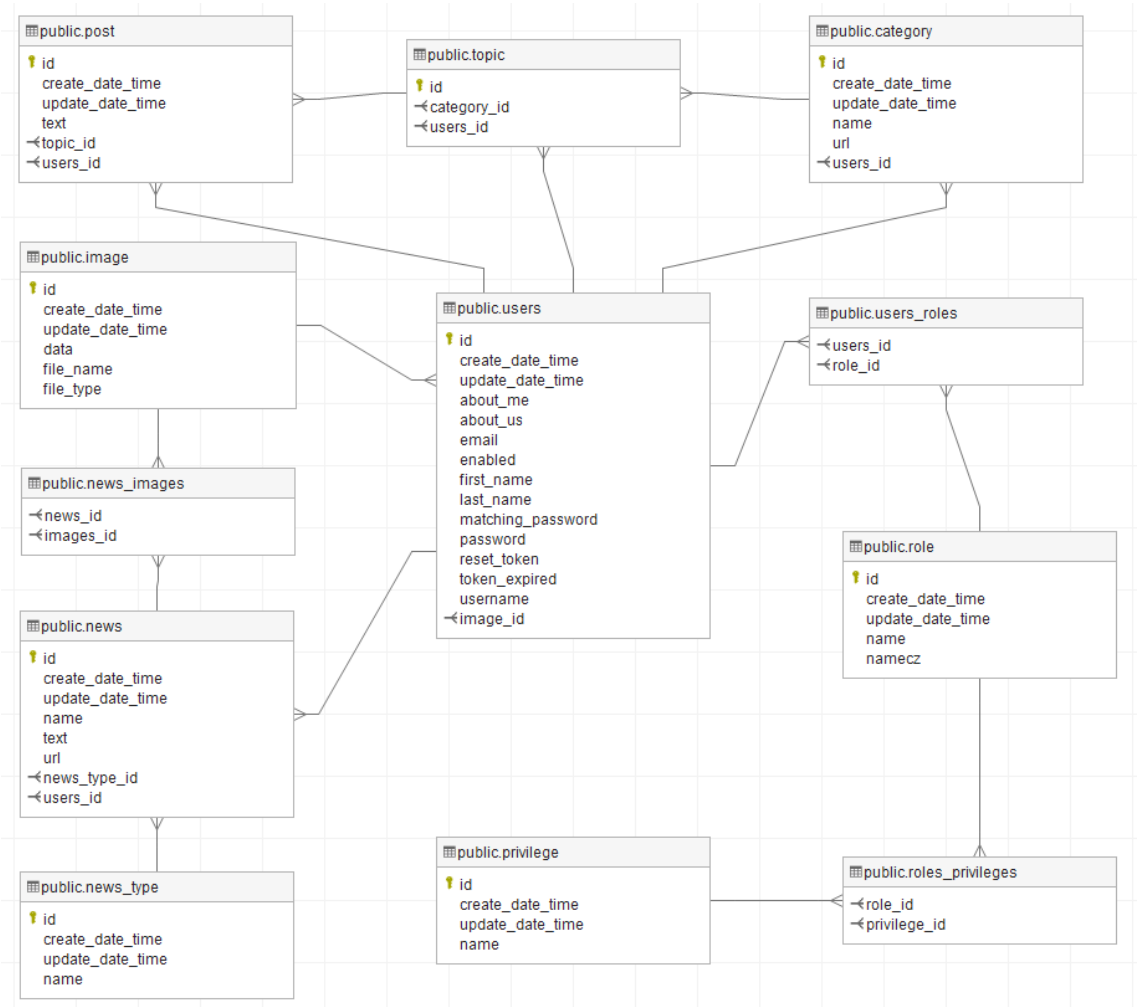
```
@Entity
public class Role extends EntityBase {

    @ManyToMany(mappedBy = "roles")
    private Collection<User> users;
```

Obrázek 9 - Ukázka anotací třídy *Role* při mapování M:N ke třídě *User*. Zdroj: autor

Databáze obsahuje následující tabulky:

(primární klíče jsou označeny žlutým klíčem a cizí klíče obrácenou šipkou)



Obrázek 10 – Diagram tabulek v databázi. Zdroj: autor

6 Použité technologie

6.1 HTML

Jazyk HTML (HyperText Markup Language; česky hypertextový značkovací jazyk) je dominantním jazykem na webu. Kdykoliv si prohlížíte nějakou webovou stránku ve svém webovém prohlížeči nebo s ní pracujete, je velmi pravděpodobné, že se jedná o dokument jazyka HTML. Tento jazyk původně vznikl, aby bylo možné formátovat a sdílet vědecké materiály. [1]

Již z názvu je patrné, že HTML umožňuje svým uživatelům vytvářet a strukturovat jednotlivé sekce, odstavce, nadpisy, odkazy a blockquotes pro webové stránky a aplikace. [14]

Nelze však HTML vnímat jako programovací jazyk. Nemá schopnost vytvářet tzv. dynamické funkce. Místo toho umožňuje organizovat a formátovat dokumenty podobně jako je tomu možné například v aplikaci Microsoft Word. [14]

Výhody HTML

- Široce používaný jazyk se spoustou zdrojů a obrovskou komunitou lidí, kteří jej využívají.
- Běží nativně v každém webovém prohlížeči.
- Open-source a zcela zdarma.
- Čisté a konzistentní značení.
- Oficiální webové standardy jsou udržovány díky W3C.
- Snadno integrovatelný s jazyky jako PHP a Node atd.

Nevýhody

- Nejčastěji je HTML využíváno pro statické webové stránky. Pro dynamickou funkčnost je lepší zvolit jazyk JavaScript.
- Nedovoluje uživateli implementovat logiku. V důsledku toho musí být všechny webové stránky vytvořeny samostatně, i když používají stejné prvky, např. záhlaví a zápatí.
- Některé prohlížeče přijímají nové funkce pomaleji.
- Chování prohlížeče je někdy obtížné předvídat (např. starší prohlížeče ne vždy obnovují novější značky). [14]

6.2 CSS

CSS je zkratka pro kaskádové styly, které kladou právě důraz na „styl“. Zatímco HTML se používá ke strukturování webového dokumentu, jako je definování věcí, zejména pak titulků a odstavců, umožňuje vkládání obrázků, videa a dalších médií, CSS prochází a určuje styl dokumentu. Toto určení stylu se týká zejména pak rozvržení stránky, barvy a písma. Opět lze pro představu uvést příklad. HTML si lze představit jako základ domu, který má každý objekt jeden, CSS je pak samostatnou estetickou volbou, tedy může následně jít o dům moderní, viktoriánského stylu atd. [4]

V současné době se používá převážně vylepšená forma tohoto stylu a to CSS3, jedná se o styl, který je proti předešlému CSS vybaven lepším uživatelským rozhraním a snadněji tak dosáhne požadovaného výsledku ve stylování jednotlivých elementů. Dále disponuje pokročilými selektory, tedy možností daleko rychleji a jednodušeji identifikovat elementy, opět dle kritérií. Díky tomu je umožněno učinit zdrojový kód mnohem přehlednější. Díky CSS3 je vývojářům umožněno vyvíjet bohatší a interaktivnější webové aplikace. [4]

6.3 Bootstrap

Bootstrap je bezplatný a open source framework pro tvorbu webových stránek a webových aplikací a lze jej považovat za světově nejpopulárnější framework tohoto druhu. V jeho zdrojovém kódu se nachází vysoce kvalitní HTML, CSS a JavaScriptový kód, který zjednodušuje počátek jakéhokoli projektu. [12]

Výhodou této sady nástrojů u Bootstrap je snadná implementace a uživatelsky přívětivý vývoj webových stránek, bez potřeby vysoké znalosti kódování v jazycích HTML a CSS. Bootstrap je kompatibilní se všemi hlavními prohlížeči na jejich nejnovějších verzích, a tak se přizpůsobuje zobrazení na každém z nich. Stejně tak je přizpůsobený pro fungování na starších či méně známějších prohlížečích. Bootstrap je sestaven z již kompilovaných CSS a JS souborů, které v případě použití stačí načíst v hlavičce webového projektu. Pro jeho správnou funkčnost je však zapotřebí i JavaScriptové knihovny jQuery. Vedle obvyklých HTML prvků, nabízí Bootstrap pokročilejší možnosti, jako je například seskupování tlačítek, vytváření navigace, záložek, štítků a dalších funkcí. Stejně tak obsahuje i

několik JavaScriptových komponent ve formě jQuery pluginů, poskytujících rozšíření uživatelského rozhraní o rozbalovací políčka s nabídkou neboli Dropdown, dále také Alerty, Tooltipy atp.

Bootstrap lze zredukovat na tři hlavní soubory: bootstrap.css - rámec CSS, bootstrap.js - rámec JavaScriptu / jQuery a glyphicons – písmo (sada ikon).

Responzivní design umožňuje webové stránce nebo aplikaci zjistit velikost a orientaci obrazovky návštěvníka a podle toho automaticky přizpůsobit zobrazení. Mobile-First přístup předpokládá, že chytré telefony, tablety a mobilní aplikace určené pro prohlížení webu jsou hlavním nástrojem návštěvníků pro získání informací a nejlepším řešením požadavků na tyto technologie v designu. [12]

6.4 Thymeleaf

Thymeleaf lze nazývat jako jakousi knihovnu Java. Jedná se o modul šablon XML / XHTML / HTML5, který je schopný aplikovat sadu transformací na soubory šablon za účelem zobrazení dat textu, jenž byl vytvořen v dané aplikaci. Je vhodnější jej využívat společně s XHTML / HTML5, a to zejména ve webových aplikacích. Dokáže ovšem zpracovat jakýkoli soubor XML, ať už na webu nebo v desktopových aplikacích. Thymeleaf tedy může pracovat jak s webovým, tak i s newebovým prostředím. [10]

Hlavním cílem Thymeleaf je poskytnout elegantní a dobře formovaný způsob vytváření šablon. Aby toho bylo dosaženo, je založen na XML značkách a atributech, které definují provedení předdefinované logiky na Document Object Model, zkráceně DOM, namísto toho, aby explicitně zapisovaly tuto logiku jako kód uvnitř šablony. Jeho architektura umožňuje rychlé zpracování šablon a spoléhá na inteligentní ukládání do paměti analyzovaných souborů, aby bylo možné během provádění využívat co nejmenší množství operací. [10]

Díky modulům pro Spring Framework a řadě integrací s jinými nástroji a schopnosti připojit se k funkčnosti jiných oblastí je Thymeleaf v současné době ideální pro vývoj webových aplikací HTML5 JVM. Thymeleaf je licencovaný pod licencí Apache License 2.0.

6.4.1 Šablony Thymeleaf

Thymeleaf umožňuje zpracovat šest druhů šablon. A to: XML, Valid XML, XHTML, Valid XHTML, HTML5 a Legacy HTML5. [10]

Všechny tyto režimy, jak jsou výše uvedené šablony v Thymleaf nazývány, se vztahují na dobře vytvořené soubory XML s výjimkou režimu Legacy HTML5, který umožňuje zpracovávat soubory HTML5 pomocí funkcí, jako jsou samostatné (neuzavřené) značky, atributy tagů bez hodnoty nebo značky, nezapisované mezi uvozovky. [10]

Za účelem zpracování souborů v tomto konkrétním režimu provede Thymeleaf nejprve transformaci, která převede soubory na správně vytvořené soubory XML, které jsou platné HTML5. Nejedná se však o jediné typy šablon, které může Thymeleaf zpracovat a uživatel je vždy schopen definovat svůj vlastní režim zadáním způsobu, jak analyzovat šablony v tomto režimu i způsobu, jak zapsat výsledky. Tímto způsobem by Thymeleaf mohl efektivně zpracovat cokoli, co lze modelovat jako strom DOM, ať už XML nebo ne. [10]

6.5 JavaScript

Nejpopulárnější programovací jazyk, který je interpretován na straně klienta a dokáže měnit obsah stránky asynchronně.

JavaScript uvidíte téměř všude, kam se na internetu dostanete, představuje jazyk webu. JavaScript je mocný, kompaktní skriptovací jazyk, který je vložen do webových stránek pro jejich oživení. JavaScript podporuje nepřeberné množství oslnivých možností pro vaše webové stránky. [2]

Stručný seznam nejzákladnějších možností JavaScriptu:

- JavaScript může reagovat na různé události, jako je pohyb myší apod.
- JavaScript si umí vyžádat od uživatele zadání vstupních hodnot.
- JavaScript dokáže ověřit platnost vstupních dat.
- JavaScript dovede vytvořit interaktivní formuláře.
- Pomocí JavaScriptu je možné jednoduše stránku opatřit heslem.
- JavaScript může změnit obsah stránky v závislosti na datu nebo hodině.
- S JavaScriptem lze snadno měnit barvu stránky.

- JavaScript může posloužit jako nástroj pro vytváření jednoduchých interaktivních her přímo na WWW stránkách
- JavaScript může fungovat jak na straně serveru, tak na straně klienta
- JavaScript umí pracovat s ovládacími prvky ActiveX a s Java Aplety.
- JavaScript může umístit na počítač klienta balíček dat, do kterého uloží potřebné informace pro pozdější použití – tzv. cookies.
- JavaScript umožňuje vytvářet dynamické stránky, které mění svoji podobu v závislosti na různých okolnostech [3]

6.6 Java

Pojem Java se používá ve dvou významech: programovací jazyk a platforma. Programovací jazyk definuje pravidla pro zápis programů. Nedílnou součástí programovacího jazyka Java je tzv. aplikační programové rozhraní (anglicky Application Programming Interface, API). Zjednodušeně řečeno je to kód, který máme k dispozici při psaní programů. Java API se skládá z tříd. Třídy jsou uspořádány do balíků. Platforma znamená prostředí, ve kterém můžeme programy spouštět. [11]

Tvorba programů pro Java platformu probíhá ve dvou krocích. V prvním kroku napíšeme tzv. zdrojový text (někdy též zdrojový kód). Zdrojový text je zápisem programu. Ve druhém kroku je zdrojový text přeložen do tzv. bajt kódu. Překlad zdrojového textu do bajt kódu zajistí překladač (kompilátor, angl. compiler). [11]

Java je používána v mnoha doménách a oblastech, jako je například:

- Bankovnictví, zde je Java určena pro správu transakcí
- Maloobchod: Fakturační aplikace, které lze vidět v obchodě / restauracích atd, jsou zcela napsány v Java.
- Informační technologie: Java je navržena pro řešení závislostí implementace.
- Android: Aplikace jsou napsány v jazyce Java nebo používají rozhraní Java API.
- Finanční služby: Používá se v aplikacích na straně serveru.

- Akciový trh: Zpracovává algoritmy, které určují, do kterých společností bychom měli investovat.
- Big Data: Hadoop MapReduce framework je psán pomocí Java.
- Vědecká a výzkumná oblast, zde Java pomáhá zpracovat velké množství informací a tvoří je tak přehlednějšími. [11]

6.6.1 Vlastnosti Java

Jednoduchost – tato vlastnost se v programovacím jazyku Java projevuje tím, že byly odstraněny všechny komplexnosti, jako jsou například ukazatele, přetížení operátora atd., jak tomu je například u výše zmíněného CSS či v jiných programovacích jazycích. Tím dle vývojářů Java ulehčil tento jazyk mnoha lidem život a jejich programování.

Nezávislost – vlastnost díky niž lze Java použít na jakoukoli aplikaci napsanou na jedné platformě a následně ji přenést na platformu zcela jinou.

Objektově orientovaná – všechno je v Java považováno za „objekt“, který má nějaký stav, chování a všechny operace jsou prováděny pomocí těchto objektů.

Zabezpečení – celý kód je po kompilaci převeden v bajt kód, který není člověkem čitelný. Java nepoužívá explicitní ukazatel a nespouští programy uvnitř jazyka, aby zabránila jakýmkoli činnostem z nedůvěryhodných zdrojů. Umožňuje vývoj systémů aplikací bez virů a neoprávněných manipulací.

Dynamika – Java má schopnost přizpůsobit se vývojovému prostředí, které je z její strany podporováno dynamickým přidělováním paměti, díky čemuž se snižuje plýtvání paměti a zvyšuje se výkon aplikace.

Distribuce – Java poskytuje funkci, která pomáhá vytvářet distribuované aplikace. Pomocí nástroje RMI (Remote Method Invocation) může program vyvolat metodu jiného programu v síti a získat výstup. K souborům tak lze přistupovat pomocí vypsání metod z jakéhokoli počítače na internetu.

Robustnost – Java disponuje silným systémem správy paměti. Pomáhá při odstraňování chyb díky tomu, že kontroluje kód během kompilace a za běhu.

Vysoký výkon – Java dosahuje vysokého výkonu pomocí bajt kódu, který lze snadno převést do nativního strojového kódu. Díky použití kompilátoru JIT (Just-In-Time) umožňuje Java vysoký výkon.

Více vláken – Java podporuje více podprocesů provádění. Díky tomu je programování pomocí vláken mnohem snazší. [15]

6.6.2 Komponenty Java

6.6.2.1 JVM

Celým názvem Java Virtual Machine, neboli virtuální stroj. Jedná se o specifikaci, která poskytuje správnou dobu běhu programu v prostředí, ve kterém lze vytvářet Java bajt kódy. V tomto případě jsou sledovány tři zápisy a to:

- Specifikace: Jedná se o dokument, který popisuje implementaci virtuálního počítače Java. Poskytuje ji společnost Sun a další společnosti.
- Implementace: Jedná se o program, který splňuje požadavky specifikace JVM.
- Runtime instance: Instance JVM je vytvořena vždy, když do příkazového řádku napíšete příkaz Java a program spustíme. [15]

6.6.2.2 JRE

Celým názvem Java Runtime Environment odkazuje opět na dobu běhu jednotlivého programu, ve kterém lze provádět opět bajt kód Java. Implementuje JVM (Java Virtual Machine) a poskytuje všechny knihovny tříd a další podpůrné soubory, které JVM používá za chodu. JRE je softwarový balíček, který obsahuje to, co je potřeba ke spuštění Java programu. [15]

6.6.2.3 JDK

Celým názvem Java Development Kit. JDK obsahuje kompletní JRE, které obsahuje nástroje pro programátory Java. Spolu s JRE zahrnuje překladač / zavaděč, kompilátor (javac), archivátor (jar), generátor dokumentace (Javadoc) a další nástroje potřebné pro vývoj v Java. Stručně řečeno, obsahuje vývojové nástroje JRE + sada Java Development Kit, a je poskytována zdarma. [15]

6.7 Spring Boot

Hlavní framework celé aplikace. Framework (aplikační rámec) je softwarová knihovna, která pomáhá vývojářům urychlit vývoj softwaru tím, že má v sobě předpřipravené nějaké často používané funkcionality, které si vývojář jenom upraví k vlastní potřebě. Příklad takovéto funkcionality může být připojení webové aplikace k databázi, ale i responzivní formulář.

Spring Boot je projekt postavený na základě Spring Frameworku. Poskytuje jednodušší a rychlejší způsob nastavení, konfigurace a spouštění desktopových i webových aplikací. [9]

6.7.1 Výhody Spring Boot

- Je velmi snadné vyvinout aplikace založené na Springu s Javou i s Groovy.
- Zrychluje vývoj a zvyšuje produktivitu
- Vyhýbá se zápisu redundantního kódu, anotací a XML konfigurace
- Je lehké do Spring Boot aplikací přidat další přídavné knihovny Spring jako jsou Spring JDBC, Spring ORM, Spring Data, Spring Security apod.
- Poskytuje základní konfiguraci, díky které nemusí vývojář všechno nastavovat hned na začátku, ale může si ji později upravit dle potřeby
- Poskytuje vnořené http servery jako jsou Tomcat, Jetty atd., pro rychlé a jednoduché testování
- Poskytuje CLI (Command Line Interface, příkazový řádek) nástroj pro vývoj a testování Spring Boot aplikací pomocí příkazového řádku jednoduše a rychle
- Poskytuje spoustu pluginů (zásuvné moduly), pro vývoj a testování Spring Boot aplikací lehce za použití nástrojů pro správu, řízení a automatizaci buildů aplikací jako jsou Maven a Gradle
- Poskytuje spoustu pluginů (zásuvné moduly), pro snadnout práci s vnořenými databázemi [8]

6.8 JUnit

Jedná se o framework pro tvorbu tzv. jednotkových testů, které jsou důležité pro správné psaní v jazyce Java. Tyto jednotkové testy lze považovat za základní kameny vývojových technik, tzv. test driven development neboli programování řízené testy (TDD) a zároveň se jedná o techniku extrémního programování. Toto extrémní programování předepisuje určité činnosti všem účastníkům vývojového procesu. Je možné říci, že obě zmíněné techniky vznikly právě pomocí frameworku JUnit. Extrémní programování i JUnit jsou oba produkty vývojáře Kenta Becka.

6.8.1 JUnit 5.0

Nejnovějším frameworkem v této oblasti je JUnit 5.0. Jeho vývoj byl od posledního JUnit 4.0 zapříčiněn zejména tím, že stále více dochází k zvyšování nároků na testovací frameworky. Mimo jednoduchých jednotkových testů se pro testování začaly využívat také jiné typy testů. Architektura frameworku JUnit 5.0 je založena na třech různých modulech. Jedná se o moduly s názvem JUnit Platform, JUnit Jupiter a JUnit Vintage. V současné době obsahují tyto moduly celkem 221 tříd.

6.8.2 JUnit Platform 1.0.0

Stěžejním modulem JUnit 5 je zejména platforma s názvem JUnit Platform. Tato platforma slouží zejména jako základ pro spuštění jakéhokoli testovacího frameworku, a to v rámci programovacího jazyku Java. JUnit Platform tak umožňuje spouštět jednotlivé testy skrze testovací frameworky, a to díky níže uvedeným balíčkům:

junit.platform.commons

Jedná se o vnitřní knihovnu frameworku JUnit. Je využívána výhradně pro JUnit 5 a využívání externími stranami není týmem vývojářů v tomto případě podporováno.

junit.platform.console

Jedná se o typ platformy obsahující tzv. ConsoleLauncher. Tento launcher představuje samostatnou aplikaci, která je ve formě příkazového řádku a slouží ke spouštění modulu JUnit Platform.

junit.platform.engine

Platforma engine představuje veřejné rozhraní, které díky tzv. interface a jeho implementaci umožní modulu JUnit Platform spouštět jednotlivé testy. Tyto testy mohou být napsané i v jiném externím testovacím frameworku. Funguje to tak, že daný testovací framework implementuje interafce TestEngine s unikátním identifikátorem, díky tomu se stane pro modul JUnit Platform přístupný a spustitelný.

junit.platform.gradle,plugin

Jedná se o rozhraní, které slouží zejména k vykonávání testů na platformě JUnit Platform pomocí nástroje Gradle.

junit.platform.launcher

Jde o veřejné rozhraní, které je využíváno vývojovým prostředím ke konfigurování a provádění vytvořených testů.

6.8.3 JUnit Jupiter 5.0.0

Modul JUnit Jupiter definuje nový programovací model, pro psaní testů a model rozšíření ve frameworku JUnit 5. Jeho základní balíčky jsou popsány níže.

junit.jupiter.api

Jedná se o rozhraní, vytvořené pro psaní jednotkových testů. Obsahuje téměř všechny typy anotací, které v JUnit 5 existují. Dále v něm je třída Assertions pro ověřování podmínek příkazem assert, třída Assumptions pro tvorbu předpokladů ke spuštění testů a třída DynamicTest pro vytváření dynamicky generovaných testů.

junit.jupiter.engine

Definuje třídu JupiterTestEngine, která je implementací rozhraní TestEngine, čímž umožňuje spouštět testy pomocí modulu JUnit Platform. [6]

6.8.4 JUnit Vintage

junit.vintage.engine

Definuje jedinou třídu, která implementuje rozhraní TestEngine a díky tomu umožňuje spouštět archivní JUnit testy, tedy testy napsané ve frameworku JUnit 3 nebo JUnit 4. [6]

6.9 JPA – Java Persistence API

Specifikace Java Persistence API, zkráceně JPA, je aplikačním rozhraním umožňujícím přístup a správu dat mezi Java objekty a databází. Namísto komunikování s databází pomocí SQL3 příkazů, JPA umožňuje Java třídě / objektu použít objektově relační mapování (ORM) pomocí anotaci nebo XML souborů, které definují to, jak bude tato třída mapována do tabulky dané databáze. Taková třída se nazývá perzistentní entitou a její instance jsou reprezentovány jednotlivými řádky databázové tabulky. Perzistentní entity mohou používat dědičnost a polymorfismus, přičemž JPA garantuje, že objekt bude při načítání z relační databáze instancí stejné třídy jako v momentě, kdy byl do databáze uložen. [5]

6.10 Hibernate

Hibernate je nástroj pro objektově relační mapování v jazyce Java. V době psaní práce se již objevuje i větev projektu Hibernate pro vývojáře v C++, ale tato práce se bude soustředit výlučně na část věnovanou Java, původnímu cílovému jazyku Hibernate.

Hibernate na programátora neklade žádné velké nároky. Jediná věc, kterou musí perzistentní třída splňovat pro spolupráci s Hibernatem, je existence bezparametrického neprivátního konstruktoru. Hibernate je nadstavbou rozhraní JDBC. Vše, co lze udělat pomocí Hibernate, lze tedy udělat i pomocí JDBC. Pokud je to potřebné, můžeme z Hibernate získat objekt typu Connection a s jeho pomocí pak můžeme pracovat jako s klasickým JDBC. Hibernate ale navíc poskytuje množství pokročilých funkcí, které zjednodušují práci. I v případě běžného použití jeho funkcí však Hibernate sám vnitřně všechnu komunikaci s databází provádí pomocí příkazů JDBC. [13]

6.11 PostgreSQL

PostgreSQL je objektově-relační databázový systém s otevřeným zdrojovým kódem. Databáze je využívána mnoha organizacemi – NOAA (National Oceanic and Atmospheric Administration), CISCO, Yahoo aj. [7]

Funguje na operačních systémech Linux, Windows a macOS. Aktualizovaná je průběžně jednou ročně. Podporuje řadu datových typů (integer, double, binary, text, date, geometry, JSON, XML, arrays atd.). [7]

7 Implementace

Implementační fáze probíhala ve vývojovém prostředí IntelliJ IDEA, které je k dispozici zdarma pro studenty, což je velká výhoda, protože se jedná o velmi profesionální nástroj, který usnadňuje vývoj webových aplikací v Java s vysokou podporou při využití Spring Boot frameworku.

Webová aplikace využívá i knihovny třetích stran, a to tedy kromě výše zmíněného frameworku Spring Boot, který obsahuje části pro zabezpečení, připojení k databázi, což jsou JPA, Hibernate a JDBC, testování, odesílání emailů a samotného spuštění a na kterém je celý projekt postaven. Dalšími knihovnami jsou Thymeleaf a Commons file upload. Všechny tyto knihovny jsou naimportovány do projektu přes nástroj pro správu, řízení a automatizaci buildů aplikací Apache Maven. Tento nástroj zajistí kompatibilnost knihoven a v případě potřeby i jejich opětovné stažení. Aby Apache Maven věděl, které knihovny má stáhnout, využívá soubor pom.xml. Existují i jiné nástroje pro správu, jako je například další velmi populární Gradle, ale v tomto projektu je využit právě Apache Maven.

Projekt, jelikož je založen na architektuře MVC, obsahuje 3 základní moduly (model, view, controller). Aplikace navíc ještě obsahuje přídavné vrstvy (helper, repository, service).

Projekt obsahuje 2 hlavní balíčky (*java, resources*). V balíčku *java* se nachází veškeré Java třídy a ty jsou dále rozděleny do *balíčků controller, helper, model, repository, security a service*. V balíčku *resources* pak nalezneme všechny sobory týkající se převážné frontendových technologií HTML, CSS, JS a také konfigurační soubor Spring Boot aplikací *application.properties*. Projekt ještě obsahuje balíček *test*, ve kterém jsou implementované třídy pro jednotkové testování.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.1.8.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>cz</groupId>
    <artifactId>laryngektomie</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>laryngektomie</name>
    <description>Web application for laryngektomie</description>

    <properties>
        <java.version>1.8</java.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-actuator</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>


        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-jdbc</artifactId>
        </dependency>

        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
        </dependency>
```

Obrázek 11 – Ukázka *pom.xml* souboru. Zdroj: autor

7.1 Persistence dat

Pro ukládání dat v této aplikaci jsou důležité tři technologie (JPA, Hibernate, PostgreSQL). JPA a jeho anotace, které byly popsány v kapitole návrh databáze, společně s Hibernate frameworkem, se starají o ukládání a nahrávání dat do a z databáze PostgreSQL. Pro funkčnost nestačí pouze výše zmíněné anotace na třídách, ale také nastavení v souboru *application.properties*. V tomto souboru se

nastavuje Connection string a další údaje viz obrázek 12. Pro funkčnost databáze je také důležité přidání JDBC ovladače pro konkrétní databázi.

```
#Database connection postgreSQL DB START
spring.main.banner-mode=off
logging.level.org.springframework=ERROR
spring.jpa.hibernate.ddl-auto=update
spring.datasource.initialization-mode=always
spring.datasource.platform=postgres
spring.datasource.url=jdbc:postgresql://localhost:5432/test9
spring.datasource.username=postgres
spring.datasource.password=testonly
spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true
```

Obrázek 12 - Ukázka nastavení připojení k PostgreSQL databázi. Zdroj: autor

Díky těmto nástrojům je velice snadné změnit typ databáze, stačí změnit nastavení v souboru *application.properties* viz obrázek 13 a naimportovat ovladač JDBC pro konkrétní databázi v tomto případě MariaDB.

```
#Database connection mariaDB
spring.datasource.url=jdbc:mariadb://localhost:3307/test3
spring.datasource.username=root
spring.datasource.password=testonly
spring.datasource.driver-class-name=org.mariadb.jdbc.Driver
spring.datasource.tomcat.test-while-idle=true
spring.datasource.tomcat.validation-query=SELECT 1
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.enable_lazy_load_no_trans=true
```

Obrázek 13 - Ukázka nastavení připojení k MariaDB. Zdroj: autor

Všechny třídy, které se ukládají do databáze, musí mít anotaci *@Entity*, proměnnou pro identifikátor s anotací *@Id* a konstruktor bez parametrů. Těmto třídám se pak vytváří rozhraní ve vrstvě repository. Tyto rozhraní musí mít anotaci *@Repository* a dědí z rozhraní *JpaRepository*, tím získají základní CRUD funkce. Další dotazy do databáze se přidávají napsáním vlastních metod. Například pokud chceme získat záznamy z databáze o posledních třech vytvořených novinkách, stačí napsat metodu *findFirst3ByOrderByCreateDateTimeDesc* a JPA rozhraní z této předložené metody vytvoří potřebný dotaz do databáze. Tyto dotazy jsou navíc ošetřeny proti SQL injection útoku.

7.2 SpringBoot Security

Jedna z nejdůležitějších částí aplikace je její zabezpečení. Administrátor může přes webové rozhraní upravovat obsah webu, a tedy by bylo nežádoucí, kdyby tuto možnost měl i jiný než ověřený uživatel. O zabezpečení se stará právě jedna z částí frameworku Spring Boot, a to ta s názvem Security. Tato knihovna potřebuje pro svou správnou funkci dvě třídy.

První třídou je *UserPrincipalDetailsService*, která implementuje rozhraní *UserDetailsService*. V této třídě především implementujeme metodu *loadUserByUsername*, ve které definujeme, jaká třída bude v naší aplikaci považována za uživatele.

```
@Override
public UserDetails loadUserByUsername(String username)
        throws UsernameNotFoundException {

    User user = userRepository.findByUsername(username);
    if (user == null) {
        throw new UsernameNotFoundException("Uživatel se zadaným uživatelským jménem nebyl nalezen.");

    }

    return new UserPrincipal(user);

}
```

Obrázek 14 - Ukázka implementace metody *loadUserByUsername*. Zdroj: autor

Druhou třídou je *SecurityConfiguration*, která dědí ze třídy *WebSecurityConfigurerAdapter* a implementuje rozhraní *ApplicationContextAware*. V této třídě implementujeme metodu *configure*, ve které jsou nastaveny oprávnění pro jednotlivé URL dotazy na aplikaci, a to podle autorizace a autentifikace uživatele.

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    http
            .cors().and().csrf().disable() HttpSecurity
            .authorizeRequests() ExpressionUrlAuthorizationConfigurer<H>.ExpressionInterceptUrlRegistry
            .antMatchers( ...antPatterns: "/").permitAll()
            .antMatchers( ...antPatterns: "/poradna/**").authenticated()
            .antMatchers( ...antPatterns: "/nastaveni/**").authenticated()
            .antMatchers( ...antPatterns: "/admin/poradna/prispevky/**").hasRole("SPECIALIST")
            .antMatchers( ...antPatterns: "/admin/poradna/temata/**").hasRole("SPECIALIST")
            .antMatchers( ...antPatterns: "/admin/**").hasRole("ADMIN")
            .anyRequest().permitAll()
            .and() HttpSecurity
            .formLogin() FormLoginConfigurer<HttpSecurity>
            .loginPage("/prihlaseni").permitAll()
            .loginProcessingUrl("/prihlasit")
            .defaultSuccessUrl( defaultSuccessUrl: "/default",  alwaysUse: true)
            .failureUrl("/prihlaseni?error=true").permitAll()
            .and() HttpSecurity
            .logout().logoutRequestMatcher(new AntPathRequestMatcher( pattern: "/odhlaseni")) LogoutConfigurer<HttpSecurity>
            .deleteCookies("JSESSIONID").logoutSuccessUrl("/prihlaseni")
            .and() HttpSecurity
            .rememberMe().tokenValiditySeconds(2592000);


}
```

Obrázek 15 - Ukázka implementace metody *configure*. Zdroj: autor

7.3 Aplikační logika

Aplikace má dvě nejdůležitější funkce a těmi jsou úprava obsahu webu přes uživatelské rozhraní a diskuzní fórum. Aplikační logika je vysvětlena s ohledem na různé případy užití v aplikaci, aby byla snadněji pochopitelná i pro čtenáře, který nedisponuje zdrojovým kódem.

7.3.1 Úprava obsahu

Úprava obsahu webu je naimplementována za pomocí tříd *User, News, NewsType* a *Image*. Uživatel s rolí *ADMIN* se přihlásí do webové aplikace a začne s přidáváním novinky na web. Vyplní základní údaje jako je název, ve WYSIWIG editoru vyplní požadovány text, vybere si typ novinky v select boxu, například setkání a nepovinně může také přidat fotky z konkrétní akce. Klikne na tlačítko vytvořit a tím odešle data do *NewsControlleru.* Tento kontroler si převezme všechny vyplněné údaje, zkontroluje jejich typovou správnost a v případě špatně vyplněného pole, vrátí uživateli předvyplněný formulář, s odezvou, kde je chyba. V případě správně vyplněných polí, prvně uloží přes servis vrstvu všechny nahrané obrázky pomocí metody *saveImages*, která si bere jako parametr *List<MultipartFile>*. V této metodě se pak postupně proberou jednotlivé soubory a v případě, že mají správný

datový formát, například image/jpeg nebo image/png, je převede na instanci třídy *Image*, kterou uloží do databáze. Následně zpátky v kontroleru přiřadí již uložené obrázky konkrétní novince, dále nastaví novince informaci o uživateli, který ji vytvořil a novinku uloží opět přes servisní vrstvu novinek, a to metodou saveOrUpdate. Tato metoda má jako jediný parametr novinku. V servisní vrstvě se jako první krok vytvoří z nadpisu novinky uživatelsky přívětivé URL a pomocí třídy *ForumHelper* a její statické metody *makeFriendlyUrl*, která má jako jediný parametr String, který chceme převést na URL. Metoda tento String vezme, převede všechny znaky na malá písmena, změní všechny písmena s diakritikou na písmena bez ní a místo mezer přidá pomlčky. V poslední řadě text, pokud je delší něž počet uvedený ve statické proměnné *TITLE_IN_URL_MAX_LENGTH*, ořízne na počet znaků uvedený v této proměnné. Návratová hodnota této metody je String s již vytvořeným textem ve tvaru URL. Poté se nastaví onen text s URL konkrétní novince a pomocí repository vrstvy se uloží do databáze.

7.3.2 Diskuzní fórum

Pro prohlížení diskuzního fóra, se uživatel nejprve musí zaregistrovat. Pokud této možnosti chce využít, stačí vyplnit klasický webový formulář pro registraci. V případě úspěšné registrace, se poté může přihlásit do diskuzního fóra.

Diskuzní fórum využívá převážně POJO třídy *User, Category, Topic* a *Post*. Chce-li uživatel vytvořit nový dotaz na fóru, vyplní formulář k tomu určený. Stejně jako při vytváření novinky, se po vyplnění údajů od uživatele a odeslání formuláře, dostanou tyto data do kontroleru, který ověří jejich správnost. Kontroluje například jestli text není příliš dlouhý. Nové téma poté přes vrstvu servis uloží do databáze a přesměruje uživatele do nově vytvořeného tématu.

7.4 Testování

Aplikace obsahuje jednotkové testy s využitím frameworku JUnit a testuje třídu *ForumHeper*, která obsahuje metody pro stránkování, vytváření datumu a času ve formátu String a vytváření popisku témat na fóru.

Testovací třída se nachází v balíčku *test.java.cz.laryngektomie.helper* a má název *ForumHelperTest*. Každá testovací metoda musí být označena anotací *@Test*. První testovací metoda testuje funkčnost metody *getListOfPageNumbers,* která podle současné stránky, na které se uživatel nachází, vygeneruje čísla stránek pro stránkování. Pomocí metody *assertEquals* z JUnit frameworku, konkrétně z balíčku *org.junit.Assert*, která si bere dva parametry, a to jako první parametr očekávaný výstup z testované metody a jako druhou konkrétní výstup z metody. Tyto hodnoty poté porovná, jsou-li totožné.

```
@Test
public void getListOfPageNumbers(){
    int totalPages = 30;
    int currentPage = 8;

    List<Integer> expectedIntegersList = Arrays.asList(1, 6, 7, 8, 9, 10, 30);

    List<Integer> listOfPageNumbers = ForumHelper.getListOfPageNumbers(totalPages, currentPage);
    assertEquals(expectedIntegersList, listOfPageNumbers);
}
```

Obrázek 16 - Ukázka testovací metody *getListOfPageNumbers*. Zdroj: autor

Ostatní testovací metody jsou implementovány v podobném duchu. Jedná se o metody *getCreatedTimeString*, která testuje vytvoření Stringu ve formátu HH:mm dd.MM.yyyy z instance třídy *DateTime*, *getDescription*, která zkracuje text pro případné náhledy novinek a *makeFriendlyUrl*, která vytváří z nadpisů novinek uživatelsky přívětivé URL. Všechny tyto třídy využívají výše zmíněnou metodu *assertEquals*.

8 Závěry a doporučení

V rámci této práce, jak již bylo avizováno v úvodu, byla vytvořena webová aplikace dle požadavků navrhovatele. Tato aplikace umožnuje svým návštěvníkům získávat důležité informace o problematice laryngektomie a zároveň klást otázky odborníkům z různých odvětví v diskuzním fóru.

V první části byly představeny požadavky zadavatele na aplikaci, a to jak funkční, tak nefunkční. Mezi funkční byly zařazeny požadavky na zabezpečení, registraci a diskuzní fórum, kde mohou klienti klást dotazy a dostanou věcné odpovědi od odborníků z řad lékařů a logopedů. Dále samozřejmě správa tohoto fóra a správa obsahu webu ve formě uživatelského rozhraní pro přidávání novinek na web. Posledním funkčním požadavkem bylo zvýšení povědomí o doméně. Nefunkčními požadavky byly dostupnost, stabilita, bezpečnost a vzhledem k cílové skupině, která je tvořena z největší části klienty ve věku od 40 let, taky velmi důležitá přehlednost. Vzhledem k těmto požadavkům pak byly vytvořeny jednotlivé role pro aplikaci a také případy užití.

V následující části byly analyzovány jednotlivé požadavky zadavatele, jejich souvislosti a nastínění budoucích funkcionalit aplikace. Také zde byly představeny některé vybrané aplikace, které stejně jako aplikace vytvořená v této práci, mají za cíl usnadňovat život lidem v případě nějakého onemocnění.

V návrhové časti je popsána architektura aplikace, která je postavena na MVC modelu. Jsou zde také uvedeny základní POJO třídy, které dávají aplikaci kostru.

Dále jsou v práci popsány všechny důležité technologie nezbytné pro tvorbu aplikace, ať už se jedná o frontendové technologie typu HTML, CSS, Bootstrap, Thymeleaf a JavaScript, nebo backendové technologie jako Java, či Spring Boot a jeho části pro persistenci dat, jmenovitě JPA, Hibernate a PostgreSQL. Nesmíme opomenout taky důležitou technologii pro tvorbu jednotkových testů JUnit.

V poslední části jsou popsány důležité implementační součásti, jako integrace databáze a její konfigurace, implementace zabezpečení a přidávání knihoven třetích stran. Na závěr jsou ještě popsány jednotlivé jednotkové testy v aplikaci.

Vývoj této aplikace mi přinesl mnoho praktických zkušeností s vývojem a s komunikací se zadavatelem. Zadavatel měl zpočátku obavy, zda bude schopen, s ohledem na to, že neovládá žádné webové technologie, spravovat si obsah webu sám. Tato funkcionalita mu je umožněna přes jednoduché uživatelské rozhraní administrace, kde může přidávat novinky na web či určovat kteří uživatelé budou zobrazeni v sekci o nás.

Aplikaci je možné rozšiřovat o další funkce, které by mohly administrátorovi pomoct při propagaci jeho domény. Například se jedná o rozesílání zpravodaje na emaily. Tento zpravodaj zatím doručují jen v tištěné podobě. Dále by aplikace mohla být rozšířena o širší správu obsahu všech stránek, a to přes jednoduchý formulář v administrační části webu.

V případě vytváření podobného projektu bych vytvářel projekt i s frontendovým frameworkem typu AngularJS nebo React. Backendovou logiku zachovat v jazyce Java a Spring Boot frameworku, ale předělal ji na REST API. Díky tomu by projekt získal ještě lepší uživatelskou přívětivost v podobě různých AJAX volání při stránkování a jiných funkcionalit na webu.

Testování aplikace probíhalo v okruhu známých a také z řad pacientů, ti především oceňovali vzhled webu a jeho jednoduchost.

9 Seznam použité literatury

9.1 Tištěné zdroje

[1] BROWN, Tiffany B., Kerry BUTTERS a Sandeep PANDA. *HTML5 okamžitě: [ovládněte HTML5 za víkend]*. Brno: Computer Press, 2014. ISBN 978-80-251-4296-7.

[2] HOLZNER, Steven. *JavaScript profesionálně: [kompletní referenční příručka]*. Praha: Mobil Media, c2003. iDnes internet knihy. ISBN 80-86593-40-1.

[3] PÍSEK, Slavoj. *JavaScript: efektní nástroj oživení www stránek*. Praha: Grada, 2001. ISBN 80-247-0014-x.

9.2 Internetové zdroje

[4] CSS Introduction. W3Schools Online Web Tutorials [online]. Dostupné z: https://www.w3schools.com/css/css_intro.asp

[5] Entity Inheritance - The Java EE 6 Tutorial. Moved [online]. Copyright © 2013, Oracle and [cit. 28.04.2020]. Dostupné z: https://docs.oracle.com/javaee/6/tutorial/doc/bnbqn.html

[6] JUnit 5 User Guide. JUnit 5. Available at: https://junit.org/junit5/docs/current/user-guide/

[7] PostgreSQL: The world's most advanced open source database. PostgreSQL: The world's most advanced open source database [online]. Copyright © 1996 [cit. 28.04.2020]. Dostupné z: https://www.postgresql.org/

[8] Spring Boot Tutorial for Beginners. *O7planning* [online]. [cit. 2019-05-08]. Dostupné z: https://o7planning.org/en/11267/spring-boot-tutorial-for-beginners

[9] Spring Boot: The Most Notable Features You Should Know - DZone Java. DZone [online]. Dostupné z: https://dzone.com/articles/what-is-spring-boot

[10] Tutorial: Using Thymeleaf. Thymeleaf [online]. Dostupné z: https://www.thymeleaf.org/doc/tutorials/2.1/usingthymeleaf.html

[11] Úvod. Programovací učebnice [online]. Dostupné z: http://programovaci-ucebnice.g6.cz/ucebnice/UcebniceJazykaJava/1_Uvod.xhtml

[12] What is Bootstrap? - Definition from WhatIs.com. Computer Glossary, Computer Terms - Technology Definitions and Cheat Sheets from WhatIs.com - The Tech Dictionary and IT Encyclopedia [online]. Dostupné z: https://whatis.techtarget.com/definition/bootstrap

[13] What is Hibernate ORM and What is it Used For? | Web, Design, SEO - FreelancingGig. Freelance Marketplace - SEO, Web, Design | FreelancingGig [online]. Copyright © 2020 FreelancingGig. All rights reserved. [cit. 15.04.2020]. Dostupné z: https://www.freelancinggig.com/blog/2018/03/15/hibernate-orm-used/

[14] What is HTML? The Basics of Hypertext Markup Language Explained. Hosting Platform - Go Online With Hostinger For Only $0.99 Now [online]. Copyright © 2004 [cit. 15.04.2020]. Dostupné z: https://www.hostinger.com/tutorials/what-is-html

[15] What is Java? A Beginner's Guide to Java and its Evolution | Edureka. Instructor-Led Online Training with 24X7 Lifetime Support | Edureka [online]. Copyright © 2020 Brain4ce Education Solutions Pvt. Ltd. All rights Reserved. [cit. 15.04.2020]. Dostupné z: https://www.edureka.co/blog/what-is-java/

10 Zkratky

API - Application Programming Interface

CLI - Command Line Interface

CRUD - Create, Read, Update, Delete

CSS - Cascading Style Sheets

DOM - Document Object Model

HTML - HyperText Markup Language

JDBC - Java Database Connectivity

JDK - Java Development Kit

JIT - Just-In-Time

JPA - Java Persistence API

JRE - Java Runtime Environment

JS – JavaScript

JSON - JavaScript Object Notation

JSP - JavaServer Pages

JVM - Java Virtual Machine

MVC - Model-view-controller

ORM - Object-relational mapping

POJO - Plain old Java object

RMI - Remote Method Invocation

SEO - Search Engine Optimization

SQL - Structured Query Language

SVG - Scalable Vector Graphics

SZÚ – Státní zdravotní ústav

TDD - Test-driven development

URL - Uniform Resource Locator

WAR - Web application Archive

XHTML - eXtensible HyperText Markup Language

XML - eXtensible Markup Language

UNIVERZITA HRADEC KRÁLOVÉ
Fakulta informatiky a managementu
Akademický rok: 2018/2019

Studijní program: Aplikovaná informatika
Forma studia: Kombinovaná
Obor/kombinace: Aplikovaná informatika (ai3-k)

Podklad pro zadání BAKALÁŘSKÉ práce studenta

Jméno a příjmení: **Bc. Vítězslav Kaňok**
Osobní číslo: **I1700645**
Adresa: Jamnická 55, Staré Město, 73801 Frýdek-Místek 1, Česká republika

Téma práce: Webová prezentace pacientské organizace
Téma práce anglicky: Web presentation for patient organization

Vedoucí práce: Mgr. Daniela Ponce, Ph.D.
Katedra informačních technologií

Zásady pro vypracování:

Cílem práce je na základě požadavků zadavatele vytvořit webovou aplikaci pacientské organizace na platformě Java s využitím relační databáze.
Osnova:
1. Úvod
2. Požadavky zadavatele
3. Případy užití
4. Analýza
5. Návrh
6. Implementace
7. Závěr

Seznam doporučené literatury:

FARRELL, Joyce. Java programming. Nineth edition. Australia: Cengage Learning, [2019]. ISBN 978-1337397070.
MARTIN, Robert C. Clean architecture: a craftsman's guide to software structure and design. London, England: Prentice Hall, [2018]. ISBN 978-0134494166.
NARDONE, Massimo, Merrick SCHINCARIOL a Mike KEITH. Pro JPA 2 in Java EE 8: an in-depth guide to Java persistence Apis. New York, NY: Springer Science+Business Media, 2018. ISBN 978-1484234198.
WALLS, Craig. Spring in action. Fifth edition. Shelter Island: Manning, [2019]. ISBN 9781617294945.

Podpis studenta: Datum:

Podpis vedoucího práce: Datum: