Návrh a vytvoření Multi–Tenant aplikace

Daniel Barbořík

Bakalářská práce
2023

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Daniel Barbořík**
Osobní číslo:	**A20001**
Studijní program:	**B0613A140020 Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Návrh a vytvoření Multi-Tenant aplikace**
Téma práce anglicky:	**Design and Development of Multi-Tenant Applications**

Zásady pro vypracování

1. Proveďte rešerši existujících řešení.
2. Vypracujte stručný rozbor technologií, které budou použity k návrhu.
3. Proveďte rozbor a analýzu požadavků na zvolené řešení.
4. Zpracujte aplikaci na základě výsledků analýzy.
5. Věnujte pozornost zabezpečení aplikace.

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

Seznam doporučené literatury:

1. GORETTI, Anthony. Introducing ASP. NET Core 6. In: *Beginning gRPC with ASP. NET Core 6*. Apress, Berkeley, CA, 2022. p. 33-81.
2. JOHNSON, Glenn. Programming in HTML5 with JavaScript and CSS3: training guide. Redmond, Wash.: Microsoft, 2013. ISBN 978-0735674387.
3. UNHELKAR, Bhuvan. *Software engineering with uml*. Auerbach Publications, 2017.
4. LETT, Jacob. *Bootstrap 4 Quick Start: A Beginners Guide to Building Responsive Layouts with Bootstrap 4*. Bootstrap Creative, 2018.
5. JAKOBUS, Benjamin. *Mastering Bootstrap 4: Master the latest version of Bootstrap 4 to build highly customized responsive web apps*. Packt Publishing Ltd, 2018.
6. BEN-GAN, Itzik; DAVIDSON, Louis; VARGA, Stacia. *MCSA SQL Server 2016 Database Development Exam Ref 2-pack: Exam Refs 70-761 and 70-762*. Microsoft Press, 2017.

Vedoucí bakalářské práce: **doc. Ing. Petr Šilhavý, Ph.D.**
Ústav počítačových a komunikačních systémů

Datum zadání bakalářské práce: **2. prosince 2022**
Termín odevzdání bakalářské práce: **26. května 2023**

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

L.S.

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 7. prosince 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne …….

podpis studenta

ABSTRAKT

Hlavním cílem této práce je vytvoření aplikace, která umožní správu projektů a úkolů pro více nájemců (tenantů) v rámci jednoho systému. Aplikace poskytne uživatelům možnost vytvářet projekty, přidávat úkoly a spravovat jejich stav. Každý nájemce bude mít přístup pouze k projektům a úkolům přiřazeným jemu samotnému. Aplikace bude založena na frameworku Nette a využije objektově-relačního mapování Doctrine pro práci s databází. Hlavním zaměřením práce je navrhnout a implementovat datový model, vytvořit uživatelské rozhraní a implementovat funkcionalitu pro správu projektů a úkolů. Tato aplikace přinese efektivnější organizaci práce a usnadní spolupráci mezi uživateli ve sdíleném prostředí.

Klíčová slova: Nette, Internet, WWW, PHP, HTML, CSS, MySQL, Bootstrap

ABSTRACT

The main objective of this work is to create an application that enables the management of projects and tasks for multiple tenants within a single system. The application will provide users with the ability to create projects, add tasks, and manage their status. Each tenant will have access only to the projects and tasks assigned to them. The application will be based on the Nette framework and will utilize the Doctrine object-relational mapping for database operations. The main focus of this work is to design and implement the data model, create a user interface, and implement functionality for project and task management. This application will bring about a more efficient organization of work and facilitate collaboration among users in a shared environment.

Keywords: Nette, Internet, WWW, PHP, HTML, CSS, MySQL, Bootstrap

Poděkování

Chtěl bych upřímně poděkovat panu doc. Ing. Petru Šilhavému, Ph.D. za jeho podporu a cenné rady při psaní mé bakalářské práce. Jeho rady a podněty byly během procesu tvorby práce neocenitelné.

OBSAH

ÚVOD

Webové stránky se v současné době staly nezbytným nástrojem pro efektivní fungování jakékoliv firmy nebo organizace. Nejenže poskytují snadný přístup k informacím, ale také slouží jako prostředek pro komunikaci s klienty. Tyto online platformy nejen prezentují detaily o produktech, službách nebo samotné organizaci, ale také umožňují zákazníkům a uživatelům vzájemnou interakci s danou firmou.

Díky neustále se rozvíjející technologii se webové stránky stávají stále pokročilejšími a nabízí širší škálu funkcí a možností. Tato zahrnují interaktivní prvky, jako jsou formuláře, animace a multimédia, a také vyspělé funkce pro správu obsahu.

Tato aplikace je navržena pro správu projektů a úkolů v prostředí s více nájemci (tenanty). Jejím hlavním cílem je poskytnout uživatelům efektivní nástroj pro organizaci a sledování jejich pracovních aktivit. Aplikace umožňuje vytváření projektů, přidávání a sledování úkolů, a to vše s důrazem na individuální přístup k datům pro každého nájemce. Každý nájemce má přístup pouze ke svým vlastním projektům a úkolům, což zajišťuje bezpečnost a soukromí dat. S využitím moderních technologií, jako je framework Nette a objektově-relační mapování Doctrine, je aplikace robustní, spolehlivá a snadno rozšiřitelná. S touto aplikací mohou uživatelé lépe organizovat svou práci, spolupracovat se svými kolegy a dosáhnout vyšší produktivity ve sdíleném pracovním prostředí.

I. TEORETICKÁ ČÁST

1 INTERNET

Internet je obrovská počítačová síť na globální úrovni, která propojuje menší sítě prostřednictvím specifických pravidel, zvaných IP (Internet Protocol). Tyto protokoly definují způsob komunikace v digitálním světě. Slovo "internet" vychází z anglického "network" (síť) - standardní termín pro americké počítačové sítě jako například Arpa-net, a z předpony "inter" (mezi), což signalizuje propojení různých dřívějších, specializovaných lokálních sítí.

Internet byl původně vytvořen jako počítačová síť spojující klíčové, vojenské, vládní a akademické počítače s cílem přežít potenciální jaderný útok nebo jiné hrozby. Jeho designéři chtěli, aby tato síť byla co nejméně zranitelná, a tak byla navržena bez centrálního bodu řízení a sestávala z mnoha rovnocenných uzlů. Data se po síti přenášejí ve formě paketů samostatných informačních jednotek, které putují k příjemci nezávisle na ostatních paketech. Pokud je některá přenosová cesta přerušena, paket může dorazit k cíli jinou cestou, přes stále fungující uzly. Tato architektura stojí v jádru dnešního internetu. [1]

1.1 Protokol IP

Internet Protocol (IP) je protokol pro přenos dat, který se využívá v paketových sítích. IP umožňuje odesílání datových souborů v jednotkách nazývaných datagramy, které cestují po síti nezávisle bez potřeby předchozího vytváření spojení nebo speciální přípravy datové trasy. IP protokol nabízí nespolehlivou doručovací službu, často označovanou jako služba založená na nejlepším možném úsilí. To znamená, že všechny zařízení na cestě datagramu se snaží dostat ho co nejbližší k cíli, avšak nezaručují jeho úspěšné doručení. Datagram může dorazit vícekrát, jednou, nebo vůbec nemusí dorazit a IP protokol také nezaručuje pořadí, v jakém budou datagramy doručeny. [1]

1.2 Protokol HTTP

Hyper Text Transfer Protocol, známý jako HTTP, je internetový protokol vytvořený pro sdílení hypertextových dokumentů ve formátu HTML. Tento protokol je, společně s e-mailem, jedním z nejfrekventovanějších a napomohl významnému rozvoji internetu v nedávných letech. HTTP využívá URL univerzální adresář zdrojů, který definuje jedinečnou lokalitu konkrétního zdroje na internetu. Existuje bezpečnější verze tohoto protokolu, HTTPS, která poskytuje šifrování dat a ochranu proti odposlechu nebo jiným nežádoucím zásahům. Tento protokol funguje na principu dotaz-odpověď mezi uživatelem

(prostřednictvím webového prohlížeče) a serverem. Uživatel odesílá serveru dotaz v jednoduchém textovém formátu, obsahující detaily o požadovaném dokumentu a funkcích prohlížeče. Server reaguje několika textovými řádky, které popisují výsledek dotazu a typ dokumentu, a poté poskytuje data daného dokumentu. [1]

1.3 Domény

Doménová jména představují textové identifikátory, které slouží jako náhrada za IP adresy. Jména počítačových domén jsou tvořena více doménami oddělenými tečkou, jako je http://mail.volny.cz. Nejvyšší úroveň domény se nachází vpravo (cz), zatímco druhá úroveň je umístěna nalevo od ní (volny). Tato úroveň označuje název organizace a třetí úroveň pojmenovává konkrétní počítač v rámci této organizace a tak dále. První úroveň domény specifikuje skupinu na základě země nebo kategorie (jako např. cz pro Českou republiku, us pro USA, com pro komerční společnosti, edu pro vzdělávací instituce, gov pro vládní weby, mil pro vojenské stránky atd.). [1]

1.4 FTP

FTP, neboli File Transfer Protocol, je standardní síťový protokol využívaný k přenosu souborů mezi dvěma počítači připojenými do TCP-based sítě, jako je třeba internet. FTP vytváří pro komunikaci mezi počítači dvě spojení, z nichž jedno slouží pro instrukce a odpovědi, zatímco druhé pro samotný přenos dat. Během přenosu FTP využívá čtyři hlavní příkazy: „odeslat“, „stáhnout“, „změnit složku“ a „přenést“. FTP také může operovat v různých módech, jako je blokový, proudový a komprimovaný. V proudovém módu FTP zpracovává data bez mezer, blokový mód dělí data na bloky a komprimovaný mód používá Lempel-Ziv algoritmus pro kompresi dat.

FTP přináší významné výhody moderním společnostem i jednotlivcům, zejména schopnost přenášet velmi velké soubory. Pro odesílání menších souborů, jako je například dokument v Wordu, lze použít různé metody, ale s FTP je možné přenášet stovky gigabajtů dat současně, přičemž se zachovává hladký přenos.

FTP zjednodušuje přenos velkých objemů dat. Díky tomu, že umožňuje poslat více souborů najednou, můžete jednoduše vybrat a odeslat všechny najednou. Bez FTP by bylo nutné soubory přenášet jednotlivě, což by bylo časově náročné.

Pokud například potřebujete přesunout velký počet důležitých dokumentů z centrální kanceláře do vedlejší pobočky a máte jen několik minut, můžete využít FTP pro odeslání všech souborů současně. I když by přenos mohl trvat 15 minut, FTP by s tím zvládlo pracovat a vy byste měli dostatek času na své setkání. [2]

2 POUŽITÉ TECHNOLOGIE A PROGRAMOVACÍ JAZYKY

2.1 HTML

HyperText Markup Language, známý také jako HTML, je jazyk používaný pro vytváření a publikaci hypertextových dokumentů na internetu prostřednictvím World Wide Web. HTML je postaven na souboru značek a jejich atributů, které jsou pro každou verzi tohoto jazyka specificky definovány. Tyto značky jsou využívány k označování různých částí textu dokumentu a k přidání sémantické hodnoty danému textu. Každá značka je umístěna mezi úhlovými závorkami <a>. Jednotlivé prvky dokumentu se skládají z otevírací značky, jejího obsahu a odpovídající zavírací značky. Příkladem může být značka , která je využívána pro zdůraznění textu. Například, červená karkulka je prvek, který obsahuje text, který má být zdůrazněn (zobrazen tučně). Prvky mohou také obsahovat další prvky ve svém obsahu. [1]

```
<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <title></title>
</head>
<body>

</body>
</html>
```

Obrázek 1: Ukázka kódu HTML [3]

2.2 CSS

Cascading Style Sheets, zkráceně CSS, je stylizační jazyk určený pro definování vzhledu webových stránek vytvořených v HTML, XHTML nebo XML. Jeho základní funkcí je oddělit vizuální prezentaci webových stránek od jejich obsahu a struktury, což návrhářům usnadňuje úpravy designu stránek bez ovlivnění jejich obsahu. Přestože HTML mělo původně zajišťovat tuto funkci, kvůli nedostatku standardizace a soutěži mezi vývojáři webových prohlížečů se tato úloha stala neuskutečnitelnou. Staré verze HTML zahrnují mnoho prvků, které se nesoustředí na strukturu a obsah dokumentu, ale na jeho estetiku. Tento trend je nežádoucí z hlediska zpracování dokumentů a vyhledávání informací. [1]

```
34  /* A reference to a type */
35  span.ts span.type-ref {
36      color: rgb(175, 0, 219) !important;
37  }
38
39  /* Signature details */
40  div.signature > table {
41      border-collapse: collapse;
42      border: thin darkgray solid;
43      width: 60%;
44  }
```

Obrázek 2: Ukázka kódu CSS [4]

2.3 Javascript

Při vytváření HTML dokumentů je téměř vždy nezbytné přidat prvky, které vylepšují uživatelskou interakci a přinášejí větší radost z používání. Tyto prvky mohou zahrnovat různé animace nebo tlačítka, která vyvolávají akce. Při psaní textu do textového pole se také často vyžaduje načítání dat ze serveru. Pro tyto úkoly je běžné využívat JavaScript. [5]

Byl navržen tak, aby poskytoval lepší uživatelské zážitky prostřednictvím animací a 2D a 3D grafiky. Původně byl JavaScript zaměřen na tvorbu klientské části webových aplikací, ale s příchodem technologií jako Node.js se stal univerzálním jazykem i pro vývoj serverových částí aplikací. Některé osoby mylně považují JavaScript za součást rodiny jazyka Java, ale kromě podobnosti syntaxe mezi nimi neexistuje žádná přímá vazba. Název "JavaScript" byl výsledkem marketingové strategie, která chtěla využít popularitu Java a původního názvu ECMAScript.

Programy napsané v JavaScriptu jsou nazývány skripty a jsou integrovány přímo do HTML kódu, což umožňuje nováčkům snadno se seznámit s jazykem. V minulosti se JavaScript setkával s problémy kompatibility mezi různými prohlížeči, což mohlo vést k nesprávnému zobrazení stránek. Nicméně, dnes je JavaScript standardem pro většinu webových stránek a jeho popularita stále roste, zejména díky využití frameworků jako React, Vue a Angular. Navíc se jazyk rozšířil i do oblasti mobilních aplikací, kde umožňuje vývoj aplikací pro různé platformy, jako jsou Android a iOS, díky technologiím jako React Native. [6]

```
<HTML>
<script type="text/javascript">
/**
 * This is a multiple-
 * line comment
 */
var index = 0;
var arr = [];

function push(elem) {
    // This comment may span only this line
    arr[index++] = elem;
}
</script>
</HTML>
```

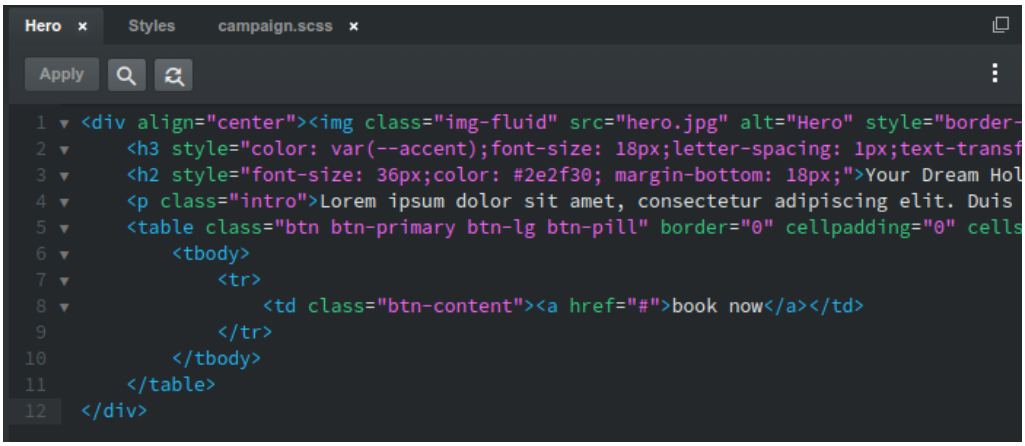
Obrázek 3: Ukázka kódu v javascriptu [7]

2.4 Bootstrap

Bootstrap je otevřený CSS a JavaScriptový framework, který vývojářům umožňuje efektivně vytvářet responzivní webové stránky. Tento nástroj byl původně navržen Markem Ottou a Jacobem Thorntonem, kteří se na jeho tvorbě podíleli během své práce u Twitteru v roce 2010. [8]

Přináší sadu šablon, syntaxí a základních elementů pro responzivní vývoj, umožňující vývojářům stavět webové stránky efektivněji, aniž by museli ztrácet čas řešením základních příkazů a funkcí. Založený na HTML, CSS a JavaScriptu, Bootstrap vývojářům umožňuje integraci kódu do již předdefinovaného mřížkového systému.

Bootstrap je nástroj, který umožňuje vytvářet webové stránky nebo aplikace schopné přizpůsobit se rozměru obrazovky, na které jsou zobrazovány. Tento framework umožňuje tvorbu webových stránek s optimalizací pro mobilní přístroje, jako jsou smartphony, tablety a mobilní aplikace. Bootstrap obsahuje všechny klíčové prvky potřebné pro design webových stránek, včetně uživatelských komponent, layoutů, JavaScriptových nástrojů a vývojového frameworku. Kromě toho je k dispozici ve dvou variantách jako předkompilovaný balíček nebo v podobě zdrojového kódu. [9]



Obrázek 4: Ukázka kódu HTML s použitím technologie Bootstrap [10]

2.5 PHP

PHP je jazyk pro vývoj programů, který se primárně používá pro tvorbu dynamických webových stránek. Jeho hlavní vlastností je možnost integrace přímo do HTML, XHTML nebo XML kódu, což podporuje vytváření webových aplikací. PHP lze však také využít pro vývoj konzolových a desktopových aplikací. Díky jednoduchosti, široké nabídce funkcí a schopnosti spojit prvky různých programovacích jazyků je PHP jedním z nejpopulárnějších skriptovacích jazyků pro web, který nabízí vývojářům určitou flexibilitu v syntaxi. [11]

```php
<?php

// Random PHP code snippet!

function create_category_feeds($categories = NULL) {

    global $wpdb, $title, $headcomments;

    if ($categories == NULL) {
        $sort_column = 'term_id';
        $query = "SELECT * FROM $wpdb->term_taxonomy
                  JOIN $wpdb->terms ON ( $wpdb->term_taxonomy.term_id = $wpdb->terms.term_id )
                  WHERE $wpdb->term_taxonomy.taxonomy = 'category' AND $wpdb->terms.term_id > 0 AND count
                  ORDER BY $wpdb->terms.name ASC";
        $categories = $wpdb->get_results($query);
    }

    $catsnum = count($categories);

    foreach ($categories as $category) {
        $link = '<link rel="alternate" type="application/rss+xml" title="';
        $link = $link . $title . ': ' . $category->name;
        $link = $link . '" href="' . get_category_rss_link(0, $category->term_id, $category->name) . '" /
        echo "\t" . $link . "\n";
    }

    $hcomlink = '<link rel="alternate" type="application/rss+xml" title="';
    $hcomlink = $hcomlink . $title . ': Comments';
```

Obrázek 5: Ukázka kódu PHP [12]

2.6 SQL

SQL neboli Structured Query Language, je programovací jazyk určený k manipulaci s informacemi v relačních databázích. Relační databáze ukládají data do tabulkové struktury, přičemž každý sloupec a řádek reprezentuje specifický atribut dat a mezi daty jsou definovány určité vztahy. SQL umožňuje provádět různé operace s daty, jako je vložení, úprava, odstranění, vyhledávání a extrakce informací z databáze. Kromě toho nabízí nástroje pro zlepšení výkonnosti a správu databází.

SQL je široce využívaný dotazovací jazyk v mnoha druzích aplikací. Díky jeho snadné integraci s různými programovacími jazyky a efektivnímu zpracování dat v hlavních SQL databázových systémech, jako jsou Oracle nebo MS SQL Server, se stává klíčovou dovedností pro data analytiky a vývojáře. Syntax SQL je založena na běžných anglických slovech v dotazech, což usnadňuje jeho pochopení. [13]

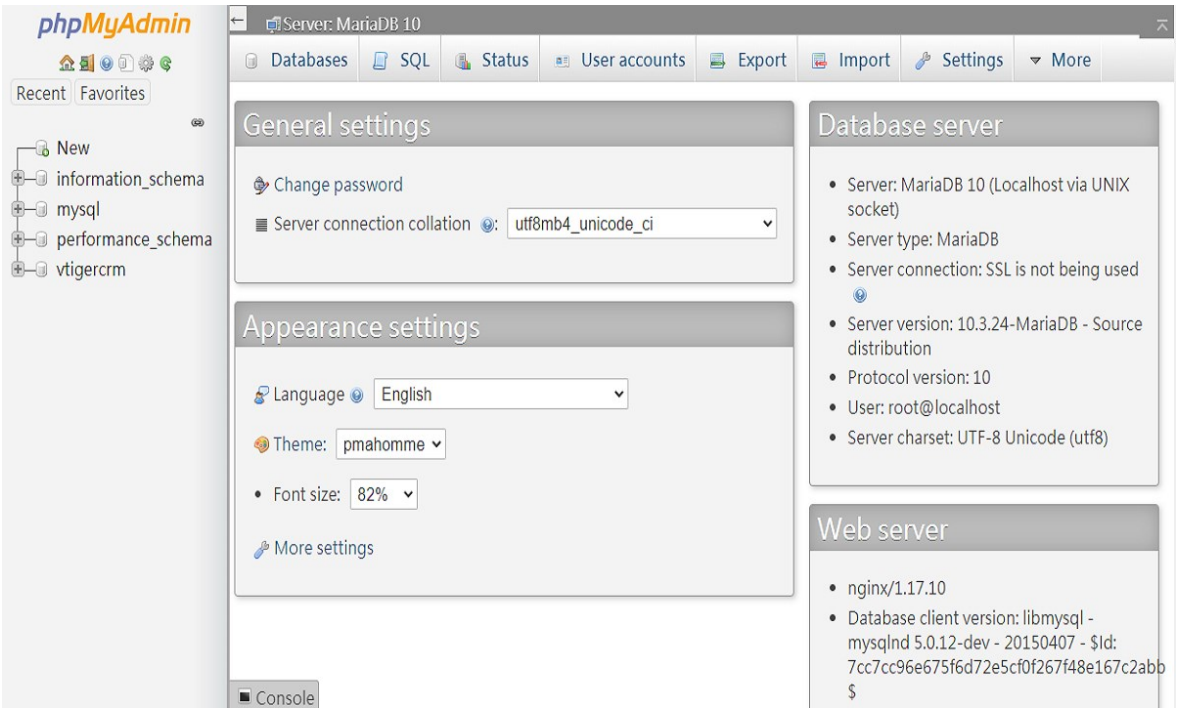
Začátek vývoje a implementace databáze pro SQL Server spočívá v pochopení procesu designu databáze a základních struktur, které tvoří její jádro. Pro vývojáře SQL Server je naprosto klíčové mít solidní znalosti těchto základů. [14]

2.7 PhpMyAdmin

PhpMyAdmin, vytvořený v roce 1998 a psaný v PHP, je softwarový nástroj navržený pro manipulaci s MySQL databázemi prostřednictvím webového rozhraní. Tato aplikace poskytuje možnost provádět širokou škálu operací s databázemi MariaDB a MySQL. Mezi

tyto operace patří vytváření, aktualizace, mazání, úpravy, import a export tabulek. Tyto činnosti lze provést buď prostřednictvím uživatelského rozhraní nebo přímo pomocí SQL příkazů.

PhpMyAdmin také nabízí funkce pro správu databází, jako je provádění dotazů, řízení databází, relací, tabulek, sloupců, indexů, oprávnění a uživatelů na platformách MySQL a MariaDB. S podporou 72 jazyků, tato aplikace může být implementována na jakýkoliv server. Uživatelé ocení jeho grafické rozhraní, které usnadňuje vytváření a úpravy databází a tabulek. Jelikož je phpMyAdmin webový nástroj, je přístupný z jakéhokoli počítače s připojením k internetu. [15]



Obrázek 6: Ukázka rozhraní PhpMyAdmin [16]

2.8 PHP Framework Nette

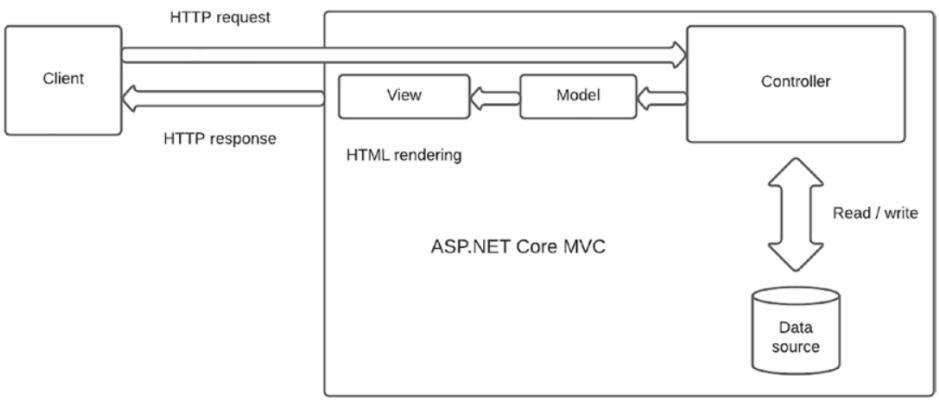
Nette Framework je otevřený vývojový nástroj pro PHP, který je k dispozici zdarma. Jeho hlavními cíli jsou zvýšení bezpečnosti, podpora technologií jako AJAX, DRY, KISS, MVC a možnost opakovaného použití kódu. Tento framework se zaměřuje na využití komponent a programování řízené událostmi. David Grudl je jeho tvůrcem a v současnosti je jeho vývojem pověřena Nette Foundation. Tento open-source software je dostupný pod licencemi GNU GPL a Nette License, což je podobné jako původní BSD licence.

V průběhu jeho vývoje byly přidávány různé nové funkce, doplňky a komponenty, což vedlo k nárůstu velikosti zdrojového kódu. Tento stav vedl k tomu, že pokud vývojář chtěl využít jen určité funkce, například jenom formuláře, bylo nutné stáhnout celý framework. Aby bylo možné používat pouze určité části, byl Nette Framework rozdělen do více samostatných repozitářů. [17]

2.9 MVC Model

Architektonický model zvaný Model-View-Controller (MVC) dělí aplikaci do tří klíčových částí: modelů, pohledů (views) a kontrolérů. Cílem tohoto vzoru je oddělit různé aspekty aplikace pro lepší organizaci a udržitelnost. Kontroléry jsou prvky, které přijímají uživatelské požadavky a provádějí akce, například manipulace s modelem nebo získávání dat z databází. Kontrolér poté určí, jaký pohled (view) má být zobrazen a poskytne mu potřebná data získaná z modelu.

Následující diagram ilustruje hlavní komponenty MVC a ukazuje, jak jsou vzájemně propojeny a odkazují na sebe: [18]



Obrázek 7: Diagram Asp.net Core MVC [19]

2.10 Doctrine PHP

Doctrine je nástroj pro mapování objektů na relační databázi (ORM) pro PHP 7.1 a vyšší. Tento nástroj používá koncept Data Mapper a je zaměřen na oddělení logiky domény od trvalého ukládání dat v relační databázi.

Doctrine umožňuje vývojářům soustředit se hlavně na objektově orientovaný design a nemusí být primárně zaměřeni na perzistenci dat. Přesto Doctrine 2 stále upřednostňuje správné zacházení s perzistencí, protože se domnívá, že oddělení perzistence od entit přináší přínosy pro objektově orientované programování.

Entity v Doctrine jsou PHP objekty, které lze jednoznačně identifikovat pomocí identifikátoru nebo primárního klíče. Entity nemusí být odvozeny od žádné konkrétní základní třídy nebo implementovat specifické rozhraní.

Tyto entity mají perzistentní vlastnosti, jsou to proměnné objektu, které Doctrine ukládá a načítá z databáze. Je třeba poznamenat, že třída entity nemůže být definována jako "final" nebo "read-only", může však obsahovat metody nebo vlastnosti definované jako "final" nebo "read-only". [20]

3 MULTI-TENANT

Oproti jedno-nájemnické architektuře, kde má každý klient svůj vlastní exemplář aplikace, databáze a infrastruktury, nabízí multi-nájemnická architektura řadu klíčových výhod jak pro poskytovatele softwaru, tak pro jejich klienty, včetně následujících:

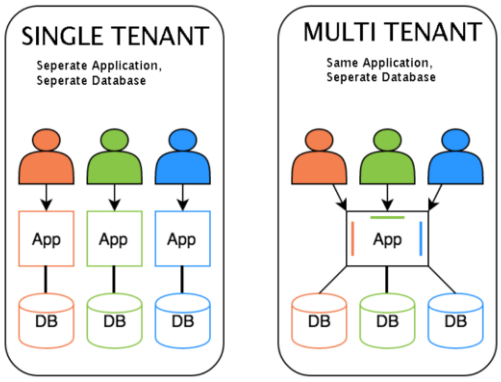
Redukce nákladů: Protože poskytovatel softwaru může služit mnoha klientům z jedné instance aplikace a infrastruktury (a proto náklady na údržbu softwaru, infrastruktury a provozu datového centra sdílí více klientů), jsou provozní náklady obvykle nižší než v jedno-nájemnickém modelu. Software jako služba (SaaS) je obvykle nabízen za fixní měsíční nebo roční poplatek, který je kalkulován na základě počtu uživatelů, míry využití nebo objemu dat, které aplikace spravuje.

Škálovatelnost: Klienti mohou na požádání škálovat své služby, noví uživatelé se připojí ke stávající instanci softwaru, což obvykle vede ke zvýšení předplatného.

Bezkódová přizpůsobitelnost: Multi-nájemnické SaaS řešení jsou vysoce konfigurovatelné, což umožňuje každému klientovi přizpůsobit aplikaci svým specifickým obchodním potřebám bez nákladů a časové náročnosti vlastního vývoje.

Kontinuální a jednotné aktualizace a údržba: Poskytovatel softwaru v multi-nájemnickém prostředí má na starosti aktualizace a opravy. Nové funkce jsou přidány a/nebo opravy jsou implementovány bez jakéhokoli úsilí ze strany klienta, a to jen jednou (na rozdíl od jedno-nájemnické architektury, kde musí být každá instance softwaru aktualizována samostatně).

Zvýšená produktivita klientů: Klienti, kteří nemusí spravovat infrastrukturu nebo software, se mohou soustředit na své klíčové úkoly. [21]



Obrázek 8: Schéma Multi-tenant aplikace [22]

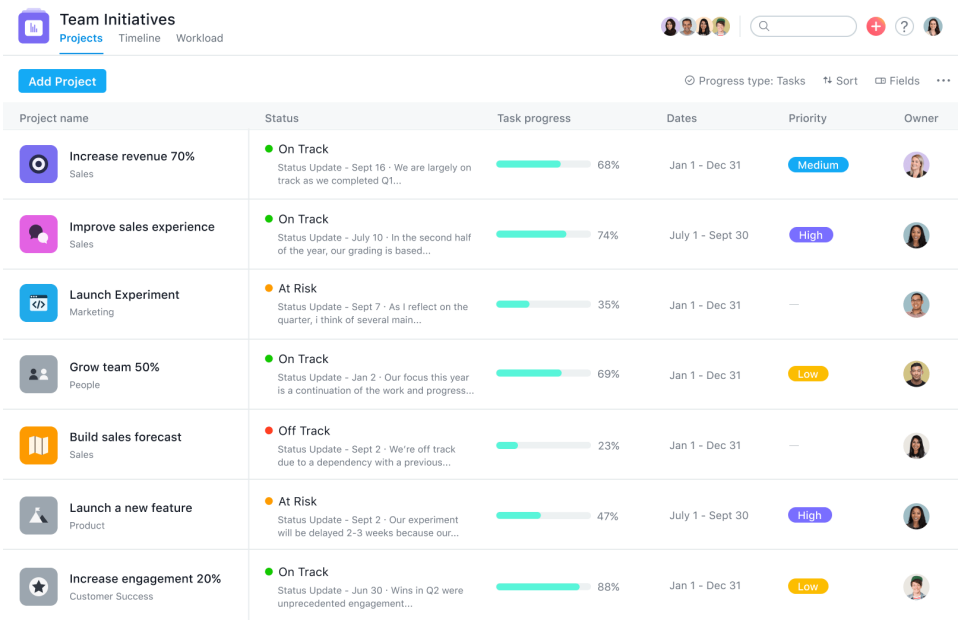
4 POROVNÁNÍ EXISTUJÍCÍCH ŘEŠENÍ

4.1 Asana

Nástroj pro správu úkolů a projektů, který byl původně vytvořen v roce 2008 týmem Facebooku. Aplikace je navržena tak, aby usnadnila týmovou spolupráci a organizaci úkolů a projektů.

Umožňuje uživatelům vytvářet úkoly, přiřazovat je členům týmu, nastavovat lhůty a přidávat podrobnosti k úkolům. Úkoly lze organizovat do projektů pro lepší přehled. Poskytuje nástroje pro sledování pokroku na úkolech a projektech. To zahrnuje vizualizace, jako jsou Ganttovy grafy, a schopnost vytvářet vlastní reporty. Uživatelé mohou komentovat úkoly a projekty, což usnadňuje týmovou komunikaci a spolupráci. Toto také pomáhá udržovat veškerou komunikaci související s projektem na jednom místě. Může se integrovat s mnoha dalšími nástroji, včetně Google Drive, Dropbox, Slack, Microsoft Teams a mnoha dalšími. To umožňuje uživatelům pracovat se svými oblíbenými nástroji přímo v Asaně.

Je to velmi přizpůsobitelný systém, umožňuje uživatelům vytvářet vlastní šablony pro projekty a nastavit vlastní pole pro úkoly. Nástroj je pro týmy různých velikostí a z různých průmyslových odvětví. Flexibilita a široká škála funkcí ho činí silným nástrojem pro správu úkolů a projektů.

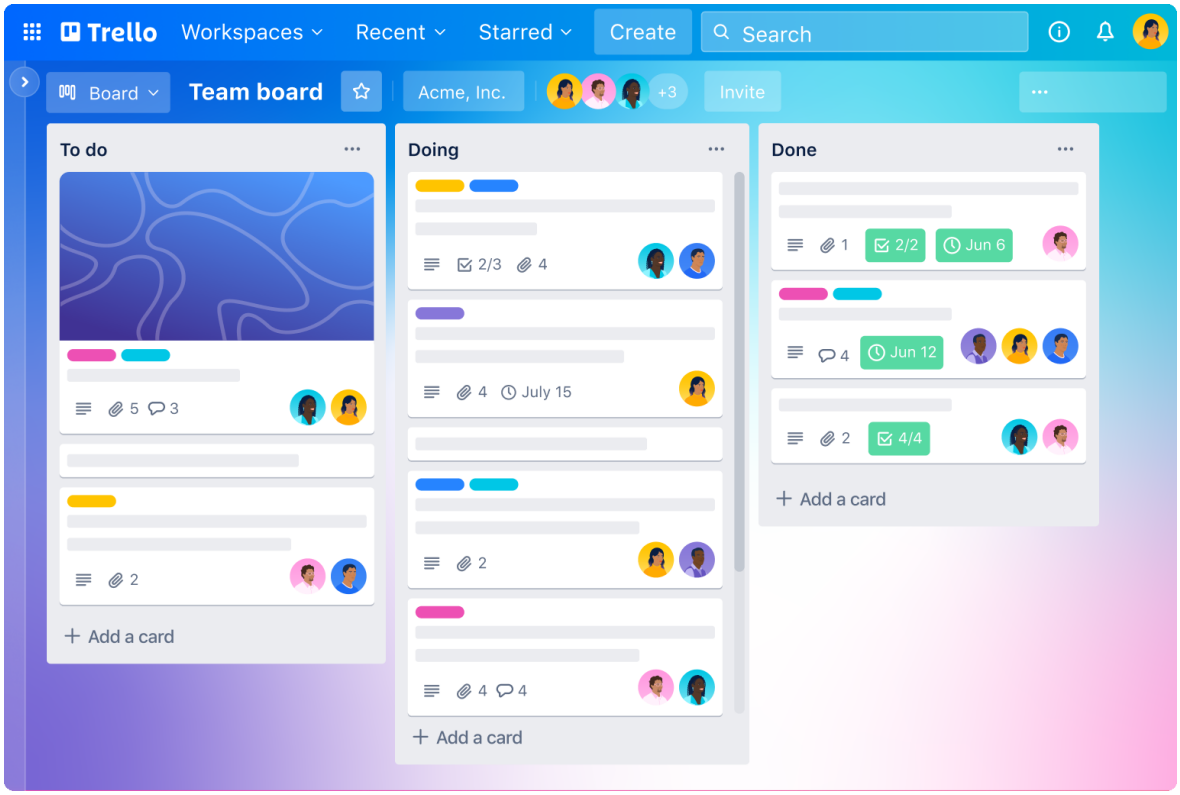


Obrázek 9: Ukázka nástroje Asana [23]

4.2 Trello

Trello je nástroj pro správu úkolů a projektů založený na metodice Kanban. Využívá vizuální tabuli, která je rozdělena na sloupce představující různé stádia pracovního postupu. Každý úkol je zastoupen kartou, která se přesouvá po tabuli podle toho, jak postupuje od jednoho stadia k dalšímu. Nástroj byl vytvořen v roce 2011 a je nyní součástí společnosti Atlassian.

Je velmi flexibilní a může být přizpůsoben pro různé druhy projektů a týmů. Desky a karty lze snadno přizpůsobit podle potřeb uživatele. Je jednoduchý a intuitivní. Je ideální pro týmy hledající vizuální způsob správy úkolů a projektů. Jeho flexibilita a přizpůsobitelnost znamenají, že je vhodný pro širokou škálu účelů a průmyslových odvětví.



Obrázek 10: Ukázka nástroje Trello [24]

II. PRAKTICKÁ ČÁST

5 ÚVOD DO PRAKTICKÉ ČÁSTI

V této praktické části je podrobně popsán proces vývoje multitenantní aplikace pro správu úkolů, vytvořené v PHP frameworku Nette. Cílem této části je poskytnout detailní pohled na architekturu aplikace, její klíčové komponenty a funkcionalitu.

Prezentovaná aplikace je navržena tak, aby poskytovala efektivní a přehledné řešení pro správu úkolů v různých organizacích. S využitím konceptu multitenancy umožňuje aplikace oddělení dat jednotlivých uživatelů na základě jejich příslušnosti k organizaci, přičemž všichni uživatelé mohou sdílet stejnou aplikaci.

V průběhu této práce je procházen celý proces vývoje, od architektury a designu aplikace, přes implementaci jednotlivých funkcí, až po testování a ladění. Je zde také podrobně rozebrána implementace multitenantního přístupu a jak tento koncept ovlivnil vývoj aplikace.

V dalších kapitolách jsou podrobněji rozebírány jednotlivé aspekty vývoje aplikace. Je zde popsána struktura souborů a adresářů, použité technologie a knihovny, klíčové komponenty aplikace, jako jsou modely, presentery a šablony. Také je popsán postup implementace klíčových funkcí, jako je správa úkolů a multitenantní přístup.

5.1 Základní pojmy

Třídy jsou specifické struktury softwarových aplikací, které integrují data s funkcemi (takzvanými metodami). Mají vlastnosti příznačné pro objektově orientované programování. Různé třídy se mohou navzájem kombinovat a tvořit tak komplexní systémy.

Data jsou digitálním vyjádřením konkrétních vlastností a aspektů reality, které jsou uložené v databázích. Atributy tříd obsahují data, která reprezentují jejich hodnoty. Relační databáze uchovávají data formou záznamů. Na druhou stranu, objektově orientované databáze spojují data a s nimi spojené funkce. Nestrukturovaná data, jako jsou obrázky, zvuky nebo popisné texty, vyžadují specifické typy úložišť.

Objekty jsou konkrétními realizacemi tříd, které zahrnují jak data, tak s nimi spojené chování během spuštění programu. Jelikož objekty vznikají během běhu programu na základě definic tříd, mohou se vyskytovat různé varianty objektů s podobnými charakteristikami. [25]

5.2 Seznámení s projektem

Projekt se zaměřuje na poskytování efektivního a přehledného řešení, které uživatelům umožní snadno spravovat, sledovat a dokončovat úkoly v rámci své organizace.

Aplikace je navržena s ohledem na multitenantní architekturu, což znamená, že každá organizace, která aplikaci používá, má své vlastní, oddělené prostředí. Tento přístup umožňuje oddělení dat mezi různými organizacemi, zatímco všechny sdílejí stejnou aplikační instanci.

Aplikace je postavena na PHP frameworku Nette, který byl zvolen pro jeho robustní architekturu, flexibilitu a silnou podporu komunity. Díky tomu bylo možné rychle implementovat požadované funkce a zároveň zajistit vysokou úroveň bezpečnosti a výkonu aplikace.

6 PŘEHLED APLIKACE

Aplikace je koncipována jako kompletní řešení pro správu úkolů, které organizacím umožňuje efektivně spravovat a sledovat úkoly svých týmů.

6.1 Základní funkce aplikace

6.1.1 Autentizace a autorizace

Uživatelé se mohou přihlásit pomocí svých přihlašovacích údajů, přičemž aplikace implementuje bezpečné postupy pro uchování a zpracování hesel.

6.1.2 Správa úkolů

Základem aplikace je modul pro správu úkolů. Uživatelé mohou vytvářet nové úkoly, přiřazovat je jiným uživatelům, sledovat jejich postup a označovat je jako dokončené. Každý úkol může být spojen s různými detaily, jako je popis, priorita, termín a další.

6.1.3 Multitenantní přístup

Jednou z klíčových vlastností aplikace je její podpora pro multitenantní přístup. Každá organizace, která aplikaci používá, má svůj vlastní prostor pro data, který je oddělen od ostatních. Toto zajišťuje, že data různých organizací jsou bezpečně izolována.

6.1.4 Správa uživatelů

Aplikace poskytuje nástroje pro správu uživatelů. Uživatelé si můžou měnit profilový obrázek, přihlašovací jméno a heslo a email.

6.2 Volba Nette Framework (výhody a nevýhody)

Nette Framework je populární ve vývoji webových aplikací v PHP, který přináší mnoho výhod, ale také některé nevýhody. Je to rodina vyspělých a samostatně použitelných komponent pro PHP 8.

6.2.1 Výhody:

Rozsáhlé nástroje a funkce: Nette Framework obsahuje širokou škálu nástrojů a funkcí, které usnadňují vývoj webových aplikací. To zahrnuje podporu pro MVC architekturu, formulářové komponenty, šablonování, routování a další. To znamená, že většina funkcí, které jsou potřebné pro vývoj moderní webové aplikace, je již zabudována přímo do frameworku.

Bezpečnost: Nette má velký důraz na bezpečnost a obsahuje mnoho vestavěných funkcí, které pomáhají vývojářům vyhnout se běžným bezpečnostním chybám. To zahrnuje ochranu proti útokům XSS, CSRF a SQL Injection.

Silná komunita a dokumentace: Nette má silnou a aktivní komunitu vývojářů, což znamená, že existuje mnoho zdrojů pro učení a podporu. Dokumentace je také dobře napsaná a obsahuje mnoho příkladů a návodů.

6.2.2 Nevýhody:

Menší mezinárodní popularita: Nette je velmi populární v České republice, ale může být méně známý a používaný v mezinárodním měřítku ve srovnání s jinými PHP frameworky, jako je Laravel nebo Symfony. To může omezit dostupnost anglicky psaných zdrojů a komunity.

Omezení a pravidla frameworku: Jako každý framework, i Nette má svá pravidla a strukturu, které musí vývojáři dodržovat. To může být omezující v některých situacích, kdy by vývojáři chtěli jít mimo tyto konvence.

6.3 Koncept Multi-Tenant přístupu

Multitenantní přístup je architektonický model, který se používá v softwaru jako službě (Software as a Service, SaaS), kde jedna instance softwaru slouží mnoha zákazníkům nebo "nájemcům". Tento model je oblíbený pro cloudové aplikace a vývoj webových aplikací, kde může jeden server hostovat mnoho klientů nebo uživatelů.

Efektivita: Jelikož všechny nájemce sdílejí stejnou instanci aplikace, může být serverové zdroje a infrastruktura využity efektivněji.

Snadná údržba a upgrade: Když je potřeba aktualizovat software, stačí to udělat na jednom místě. Všechny nájemce automaticky získají přístup k novým funkcím nebo vylepšením.

Flexibilita: Každý nájemce může mít své vlastní přizpůsobení a konfiguraci, která se aplikuje na úrovni uživatele nebo skupiny uživatelů.

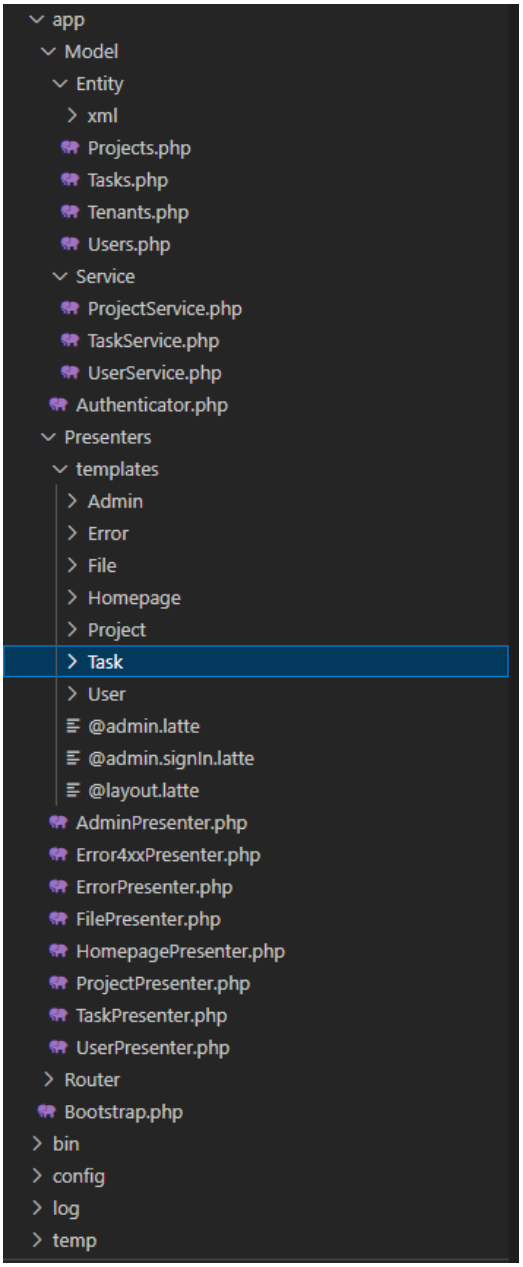
Nicméně multitenantní architektura také přináší určité výzvy, zejména v oblasti bezpečnosti a izolace dat. Jelikož všichni nájemci sdílejí stejnou instanci softwaru, je nezbytné, aby data jednoho nájemce byla bezpečně izolována od dat ostatních nájemců. To vyžaduje pečlivé plánování a provedení bezpečnostních opatření.

Výkon: Pokud jeden nájemce výrazně zvýší své využití zdrojů, může to ovlivnit výkon pro ostatní nájemce. Může to vyžadovat sofistikované řízení zdrojů a plánování kapacity.

V rámci tohoto projektu je multitenantní přístup klíčovým prvkem, který umožňuje každé organizaci spravovat své úkoly a uživatele nezávisle na ostatních, zatímco všechny sdílejí stejnou aplikační instanci.

7 STRUKTURA APLIKACE

7.1 Přehled struktury souborů a adresářů



Obrázek 11: Přehled struktury souborů a adresářů

7.2 Soubor Bootstrap.php

Soubor Bootstrap.php v aplikacích Nette je zodpovědný za vytvoření a nastavení instance třídy Nette\Bootstrap\Configurator. Tato třída je základem pro inicializaci vaší aplikace a nastavuje parametry a služby potřebné pro její provoz.

Bootstrap.php

app > Bootstrap.php > App\Bootstrap > boot

```php
<?php

declare(strict_types=1);

namespace App;

use Nette\Bootstrap\Configurator;


2 references | 0 implementations
class Bootstrap
{
    2 references | 0 overrides
    public static function boot(): Configurator
    {
        $configurator = new Configurator;
        $appDir = dirname(__DIR__);

        if (getenv('NETTE_DEBUG') == 1) {
            $configurator->setDebugMode(true);
        }
        $configurator->enableTracy($appDir . '/log');

        $configurator->setTimeZone('Europe/Prague');
        $configurator->setTempDirectory($appDir . '/temp');

        $configurator->createRobotLoader()
            ->addDirectory(__DIR__)
            ->register();

        $configurator->addConfig($appDir . '/config/common.neon');
        $configurator->addConfig($appDir . '/config/services.neon');
        $configurator->addConfig($appDir . '/config/local.neon');

        return $configurator;
    }
}

```

Obrázek 12: Ukázka kódu souboru Bootstrap.php

7.3 Adresář app/config/

Adresář app/config je zásadní pro správné fungování jakékoliv Nette aplikace. Obsahuje soubory ve formátu NEON, které definují konfiguraci aplikace, včetně definice a konfigurace služeb, parametrů, rozšíření a routování. Všechny tyto soubory jsou načteny a zpracovány třídou Nette\Bootstrap\Configurator v souboru Bootstrap.php. Soubory jsou načteny v určitém pořadí a později načtené soubory mohou přepsat konfigurace definované dříve načtenými soubory. To umožňuje mít konfigurace specifické pro určité prostředí, které mohou přepsat obecné konfigurace.

7.3.1 Soubor common.neon

Tento soubor obsahuje konfigurace, které jsou společné pro všechna prostředí (vývoj, produkce, testování atd.). Může obsahovat obecné nastavení aplikace, definice a konfigurace služeb, které nezávisí na konkrétním prostředí.

7.3.2 Soubor local.neon

Tento soubor obvykle obsahuje konfigurace specifické pro lokální vývojové prostředí, jako jsou například přihlašovací údaje k databázi. Je dobré praxí tento soubor ignorovat v systému pro správu verzí (jako je git), aby se předešlo přenosu citlivých dat.

7.3.3 Soubor services.neon

Tento soubor obsahuje definice a konfigurace služeb používaných v aplikaci. Služby mohou zahrnovat různé třídy a objekty, které aplikace využívá, jako jsou třídy pro práci s databází, třídy pro odesílání e-mailů, třídy pro logování atd.

C: > Users > barbo > Desktop > local.neon

```
parameters:


nettrine.dbal:
    debug:
        panel: %debugMode%
        sourcePaths: [%appDir%]
    connection:
        host: md203.wedos.net
        driver: mysqli
        dbname: d129155_tasks
        user: a129155_tasks
        password: [redacted]
        charset: 'utf8mb4'
        defaultTableOptions:
            charset: 'utf8mb4'
            collate: 'utf8mb4_unicode_ci'

database:
    dsn: 'mysql:host=md203.wedos.net;dbname=d129155_tasks;charset=utf8mb4'
    user: a129155_tasks
    password: [redacted]

```

Obrázek 13: Ukázka kódu souboru local.neon

7.4 Presenters

Adresář app/Presenters obsahuje soubory, které jsou součástí prezenterů v aplikaci Nette. Prezentery v Nette jsou třídy, které slouží jako spojnice mezi logikou aplikace a její vizuální reprezentací (šablonami). Každý presenter odpovídá za zpracování jednoho nebo více požadavků a předání dat šabloně pro zobrazení.

Jednotlivé metody v těchto třídách, často nazývané "akce", odpovídají na konkrétní požadavky. Například metoda renderEdit() v ProjectPresenter by odpovídala na požadavek na zobrazení editační stránky pro projekty. Metoda actionDelete($id) v ProjectPresenter by zase mohla odpovídat na požadavek na smazání pomocí id daného projektu.

7.5 Model

Adresář app/Model je místem, kde jsou umístěny soubory, které se zabývají manipulací s daty v aplikaci. Tento adresář je součástí architektury Model-View-Controller (MVC), kde "Model" představuje datovou vrstvu aplikace.

Modely jsou zodpovědné za získávání dat, jejich úpravu a ukládání. Komunikují s databází a implementují logiku potřebnou pro manipulaci s daty.

7.6 Templates

Adresář app/templates je součástí struktury aplikace Nette a je určen pro ukládání šablon. Tyto šablony jsou soubory, které definují, jak budou data zobrazována uživatelům. Nette používá pro šablony vlastní jazyk Latte, který umožňuje bezpečné a flexibilní zpracování šablon.

Každý presenter má obvykle svůj vlastní podadresář v app/templates, ve kterém jsou uloženy jeho šablony. Názvy těchto podadresářů a souborů odpovídají názvům presenterů a jejich akcí.

7.7 Adresář /www/

Adresář www/ je kořenovým adresářem (také nazývaným "webroot" nebo "document root") pro Nette aplikaci a obsahuje všechny veřejně přístupné soubory. To znamená, že všechny soubory umístěné v tomto adresáři jsou přístupné prostřednictvím webového prohlížeče.

7.7.1 Soubor index.php:

Tento soubor je vstupním bodem aplikace. Když webový prohlížeč pošle požadavek na server, je tento požadavek zpracován index.php souborem, který následně zavolá příslušné části aplikace.

7.7.2 Soubor .htaccess:

Tento soubor je konfigurační soubor pro webové servery, které používají Apache. Obsahuje pravidla pro přesměrování URL, která umožňují "hezké URL" (bez .php na konci) a přesměrování všech požadavků na index.php, takže Nette může správně zpracovat požadavky.

7.7.3 Adresáře css/, js/, images/

Tyto adresáře obsahují statické soubory, které jsou součástí webové aplikace. Adresář css/ obsahuje CSS soubory pro stylování stránek, js/ obsahuje JavaScriptové soubory pro klientské skripty a images/ obsahuje obrázky používané v aplikaci. Všechny styly a skripty v těchto adresářích jsou založeny na Dash UI, což je Bootstrap 5 Admin & Dashboard Theme. Dash UI Kit je poskytován jako sada open source komponent a šablon, plně naprogramovaných pomocí Bootstrap 5. Tímto způsobem jsou využity všechny styly a vizuální efekty v aplikaci.

7.7.4 Favicon

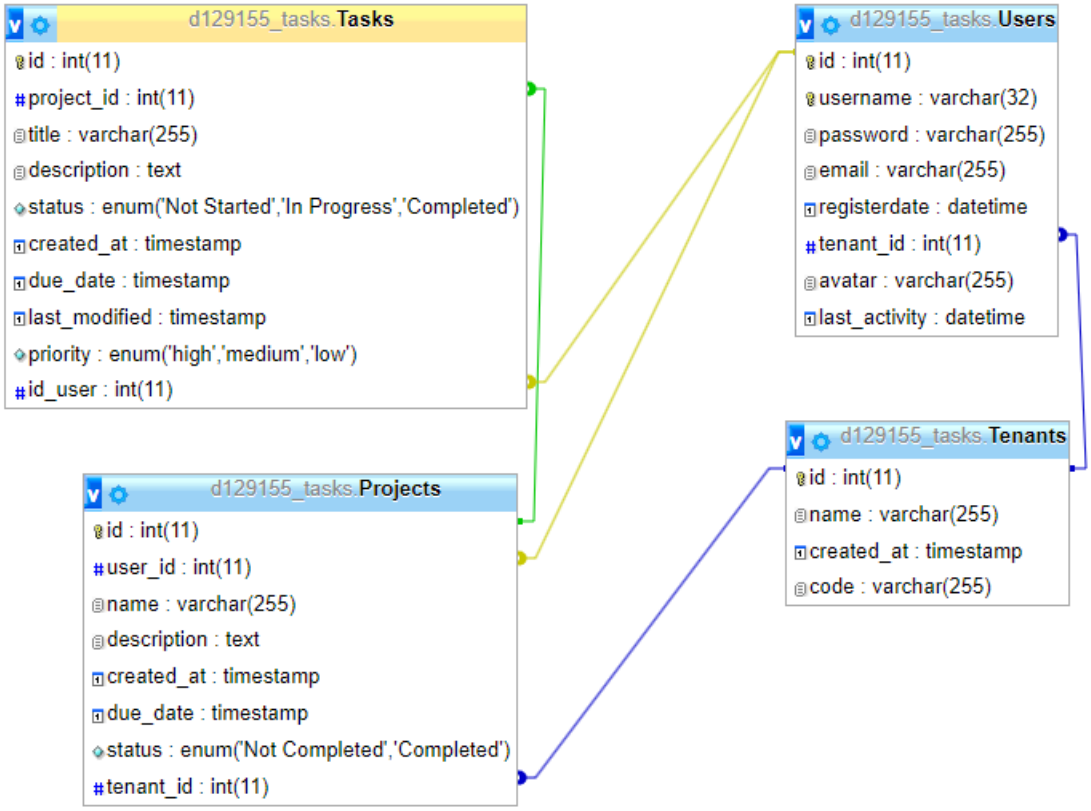
Tento soubor je ikonou webového místa, která se zobrazuje v záložce webového prohlížeče nebo v seznamu oblíbených položek.

7.8 Adresář /temp/

Adresář temp/ v aplikaci Nette je určen pro dočasné soubory, které generuje a využívá sama aplikace během svého běhu. Je důležitý pro správné fungování aplikace a některé jeho části jsou automaticky generovány a spravovány frameworkem Nette. Mezi soubory a podadresáře, které můžete v adresáři temp/ najít, patří: cache a sessions. Cache podadresář obsahuje soubory vyrobené systémem pro cachování Nette. Cachování pomáhá zvýšit výkonnost aplikace tím, že ukládá výsledek náročných operací pro opakované použití. Například výsledek databázových dotazů nebo kompilované šablony mohou být uloženy v cache. Sessions podadresář obsahuje data o relacích. Pokud je v aplikaci aktivováno správa relací, data o jednotlivých uživatelských relacích.

8 MODELY A DATABÁZE

8.1 Popis databázových tabulek a jejich vztahy



Obrázek 14: Schéma databáze

Tabulka **Projects**: Tato tabulka uchovává data týkající se jednotlivých projektů. Každý projekt má svůj jedinečný identifikátor (id), jméno (name), popis (description), datum a čas vytvoření (created_at), termín (due_date), status (status) a identifikátor tenantu (tenant_id). Tabulka také obsahuje cizí klíče pro user_id a tenant_id, které odkazují na tabulky Users a Tenants respektive. Status projektu může být buď 'Not Completed' nebo 'Completed'.

Tabulka **Tasks**: Tabulka Tasks obsahuje informace o jednotlivých úkolech spojených s projekty. Každý úkol má svůj jedinečný identifikátor (id), název (title), popis (description), status (status), datum a čas vytvoření (created_at), termín (due_date), čas poslední úpravy (last_modified), prioritu (priority) a identifikátor uživatele (id_user). Každý úkol je navíc spojen s konkrétním projektem prostřednictvím project_id.

Status úkolu může být 'Not Started', 'In Progress' nebo 'Completed'. Priorita úkolu může být 'high', 'medium' nebo 'low'. Tabulka obsahuje cizí klíče pro project_id a id_user, které odkazují na tabulky Projects a Users.

Tabulka **Tenants**: Tato tabulka obsahuje informace o tenantech. Každý tenant má svůj jedinečný identifikátor (id), název (name), datum a čas vytvoření (created_at) a kód (code).

Tabulka **Users**: Tabulka Users obsahuje informace o jednotlivých uživatelích. Každý uživatel má svůj jedinečný identifikátor (id), uživatelské jméno (username), heslo (password), e-mail (email), datum a čas registrace (registerdate), identifikátor tenantu (tenant_id), avatar (avatar) a datum a čas poslední aktivity (last_activity). Uživatelské jméno je jedinečné. Tabulka obsahuje cizí klíč pro tenant_id, který odkazuje na tabulku Tenants.

Tyto čtyři tabulky tvoří základ databáze. Kromě toho, tabulky Projects, Tasks a Users jsou navzájem propojeny pomocí cizích klíčů, což umožňuje sledovat vztahy mezi projekty, úkoly a uživateli, a to i v rámci konkrétního tenantu.

8.2 Detailní pohled na model

8.2.1 Projects.php

Tento model reprezentuje entitu Projects a je využit v kontextu Doctrine ORM, což je nástroj pro mapování objektů na relační databázi v PHP.

Proměnné: Reprezentují jednotlivé sloupce tabulky Projects v databázi. Například, proměnná $id představuje unikátní identifikátor projektu. Každá z těchto proměnných má specifické anotace (např. @ORM\Column(type="string")), které definují, jak jsou data uložena v databázi.

Konstruktor: Tato metoda se volá při vytvoření nové instance třídy Projects. Parametry konstruktoru odpovídají vlastnostem třídy a jsou přiřazeny během vytváření instance.

Getter a Setter metody: Tyto metody poskytují rozhraní pro získání a nastavení hodnot proměnných. Například metoda getName() vrátí hodnotu proměnné $name a metoda setName(string $name) nastaví novou hodnotu proměnné $name.

Důležitým prvkem je také vztah mezi projekty a uživateli, který je definován pomocí anotace @ORM\ManyToMany. Tato anotace říká, že jeden projekt může být spojen s mnoha uživateli a jeden uživatel může být spojen s mnoha projekty.

V konkrétních termínech, třída Projects je model, který mapuje databázovou tabulku Projects na PHP objekty. Tento model poskytuje metody pro manipulaci s daty a vztahy projektů v databázi.

Tato třída definuje tyto atributy:

id: Jednoznačný identifikátor projektu. Toto pole je generováno automaticky (auto-increment) a je označeno jako primární klíč (@ORM\Id).

user_id: Jednoznačný identifikátor uživatele, který je spojen s tímto projektem.

name: Jméno projektu.

description: Popis projektu.

created_at: Datum a čas vytvoření projektu.

due_date: Datum a čas, kdy by měl být projekt dokončen.

status: Stav projektu.

tenant_id: Jednoznačný identifikátor tenantu, který je spojen s tímto projektem.

users: Kolekce uživatelů spojených s tímto projektem. To je realizováno prostřednictvím mnoho-na-mnoho vztahu (@ORM\ManyToMany) s entitou Users. Join tabulka s názvem "project_user" je definována pro udržování vztahu, kde project_id a user_id jsou cizí klíče, které odkazují na primární klíče v tabulkách Projects a Users.

```php
    /**
     * @ORM\ManyToMany(targetEntity="Users")
     * @ORM\JoinTable(name="project_user",
     *      joinColumns={@ORM\JoinColumn(name="project_id", referencedColumnName="id")},
     *      inverseJoinColumns={@ORM\JoinColumn(name="user_id", referencedColumnName="id")}
     * )
     */
    1 reference
    private Collection $users;
    3 references
    public int $tenant_id;

    1 reference | 0 overrides
    public function __construct(
        int $user_id,
        string $name,
        string $description,
        \DateTimeInterface $created_at,
        \DateTimeInterface $due_date,
        string $status,
        int $tenant_id
    ) {
        $this->users = new ArrayCollection();
        $this->user_id = $user_id;
        $this->name = $name;
        $this->description = $description;
        $this->created_at = $created_at;
        $this->due_date = $due_date;
        $this->status = $status;
        $this->tenant_id = $tenant_id;
    }
```

Obrázek 15:Ukázka modelu Projects.php

8.2.2 Tasks.asp

Třída Tasks je mapována stejně jako Projects.php jako entita v Doctrine, což znamená, že reprezentuje atributy/sloupce v databázové tabulce "Tasks". Každý atribut v této třídě reprezentuje jeden sloupec v dané tabulce.

id: Je to primární klíč tabulky "Tasks". Je označen anotacemi @ORM\Id a @ORM\GeneratedValue, což znamená, že je to jedinečný identifikátor a je automaticky generován.

project_id: Toto pole obsahuje identifikátor projektu, ke kterému úloha patří. Je to obyčejné pole bez specifických vztahů.

Title, description, status a priority obsahují informace o názvu úlohy, jejím popisu, stavu a prioritě. Jsou typu string.

Created_at, due_date, last_modified jsou datové a časové údaje o úloze. created_at je datum a čas vytvoření úlohy, due_date je datum a čas, kdy má být úloha dokončena, a last_modified je datum a čas poslední úpravy úlohy.

id_user: Toto pole je zvláštní, protože je mapováno na cizí klíč pomocí anotace @ORM\ManyToOne. To znamená, že každá úloha je přiřazena k jednomu uživateli. @ORM\JoinColumn určuje, jaký sloupec v databázové tabulce "Users" odpovídá tomuto cizímu klíči.

Všechny tyto atributy jsou nastaveny prostřednictvím konstruktoru třídy a také mohou být získány nebo nastaveny prostřednictvím odpovídajících get/set metod.

Dále existuje statická metoda **create**, která vytváří a vrátí novou instanci třídy Tasks na základě dat z databázové tabulky reprezentovaných instancí ActiveRow. Pokud je ActiveRow null, metoda vrátí null.

8.2.3 Tenants.php

Třída Tenants je také entita v Doctrine a reprezentuje atributy / sloupce v databázové tabulce "Tenants". Každý atribut v této třídě reprezentuje sloupec v této tabulce.

id: Je to primární klíč tabulky "Tenants". Je označen anotacemi @ORM\Id a @ORM\GeneratedValue, což znamená, že je to jedinečný identifikátor a je automaticky generován.

name: Tento atribut reprezentuje název tenantu (nájemce, uživatele, klienta). Je to řetězec a v databázi se pro něj vytvoří sloupec name.

created_at: Tento atribut zaznamenává datum a čas vytvoření tenantu. Je to objekt typu DateTimeInterface.

code: Tento atribut reprezentuje kód tenantu, který je také řetězcem.

Stejně jako u ostatních modelů jsou atributy nastaveny prostřednictvím konstruktoru třídy. Konstruktor přijímá název, datum vytvoření a kód jako vstupní parametry a nastaví odpovídající atributy.

8.2.4 Users.php

Třída Users je entita, která reprezentuje uživatele v aplikaci. Tato třída definuje tyto atributy:

id: Jednoznačný identifikátor uživatele. Toto pole je generováno automaticky (auto-increment) a je označeno jako primární klíč (@ORM\Id).

username: Jméno uživatele.

password: Heslo uživatele.

registerdate: Datum a čas registrace uživatele. Pokud není při vytváření uživatele explicitně zadáno, nastaví se na aktuální datum a čas.

tenant_id: Jednoznačný identifikátor tenantu, ke kterému uživatel patří. Toto je klíč k propojení uživatele s tenantem v rámci multi-tenant architektury.

avatar: Volitelný řetězec reprezentující avatar uživatele. Může obsahovat cestu k obrázku nebo URL adresu.

last_activity: Datum a čas poslední aktivity uživatele. Toto pole je volitelné.

Stejně jako jiné třídy i tato třída obsahuje metody pro získání a nastavení hodnot těchto atributů.

8.3 Adresář Entity/xml/

XML mapovací soubory jsou uloženy v adresáři Entity/xml/. Každý soubor obsahuje mapování pro jednu entitu a obvykle má stejný název jako třída entity, kterou mapuje, ale s příponou .dcm.xml.

XML mapování má několik výhod. Jednou z nich je, že může být snadněji validováno pomocí XML schématu, což usnadňuje odhalování chyb v mapování. Další výhodou je, že XML mapování je oddělené od kódu entity, což může vést k čistšímu a snadněji čitelnému kódu.

Na druhou stranu, udržování XML souborů může být náročnější než práce s anotacemi, protože změny v entitě vyžadují aktualizaci jak kódu entity, tak XML souboru.

8.3.1 Příklad mapovacího souboru Projects

První část určuje, jaká entita v aplikaci se mapuje na jakou tabulku v databázi. V našem případě se třída App\Model\Entity\Projects mapuje na tabulku Projects v databázi.

Následuje definice primárního klíče. Tady se určuje, jaké pole entity odpovídá primárnímu klíči v databázi, jeho typ a jak se generují jeho hodnoty.

Další část souboru definuje jednotlivé atributy entity a jak se mapují na sloupce v databázi. Pro každý atribut se určuje jeho název, typ a případně maximální délka.

Celý tento proces slouží k tomu, aby Doctrine ORM vědělo, jak pracovat s daty v databázi, jak je načítat a ukládat, jak vytvářet nové záznamy nebo měnit existující atd.

App.Model.Entity.Projects.dcm.xml

app > Model > Entity > xml > App.Model.Entity.Projects.dcm.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<doctrine-mapping xmlns="https://doctrine-project.org/schemas/orm/doctrine-mapping"
                  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
                  xsi:schemaLocation="https://doctrine-project.org/schemas/orm/doctrine-mapping
                          https://www.doctrine-project.org/schemas/orm/doctrine-mapping.xsd">

    <entity name="App\Model\Entity\Projects" table="Projects">

        <id name="id" type="integer" column="id">
            <generator strategy="AUTO"/>
        </id>

        <field name="user_id" length="32" type="integer">
        </field>

        <field name="name" length="255" type="string">
        </field>

        <field name="description" length="255" type="string">
        </field>

         <field name="created_at" type="datetime">
        </field>

         <field name="due_date" type="datetime">
        </field>

         <field name="status" length="32" type="string">
        </field>

        <field name="tenant_id" length="32" type="integer">
        </field>

    </entity>

</doctrine-mapping>
```

Obrázek 16: Ukázka xml souboru pro mapování

9 SERVISNÍ TŘÍDY

Servisní vrstva představuje klíčový prvek pro propojení prezentační a datové vrstvy. Prezentační vrstva je odpovědná za uživatelské rozhraní, jako jsou webové stránky nebo aplikace, zatímco datová vrstva se stará o správu a přístup k datům, obvykle uloženým v databázi.

V kontextu modelů, které jsou reprezentací dat v databázi, složka "service" obsahuje třídy nebo funkce, které pracují s těmito modely. To může zahrnovat funkce pro provádění CRUD operací pro čtyři základní funkce potřebné pro práci s databází: Vytvoření (Create), Přečtení (Read), Aktualizace (Update) a Odstranění (Delete).

9.1 Soubor UserService.php

Funkce **findUser**(string $username): ?Users: Tato funkce vyhledává uživatele podle uživatelského jména v databázi. Uživatelské jméno je vstupní parametr. Pokud je uživatel nalezen, funkce vrátí instanci objektu Users. Pokud uživatel neexistuje, vrátí funkce null.

Funkce **validcode**(string $code): ?Tenants: Tato funkce vyhledává nájemce (tenants) podle kódu. Kód je vstupní parametr. Pokud je nájemce nalezen, funkce vrátí instanci objektu Tenants. Pokud nájemce neexistuje, vrátí funkce null.

Funkce **registerUser**(string $username, string $password, string $email, string $code): Users: Tato funkce slouží k registraci nových uživatelů. Jako parametry přijímá uživatelské jméno, heslo, email a kód organizace. Funkce vytvoří novou instanci třídy Users, hashuje heslo, uloží informace o uživateli do databáze a vrátí objekt nově vytvořeného uživatele.

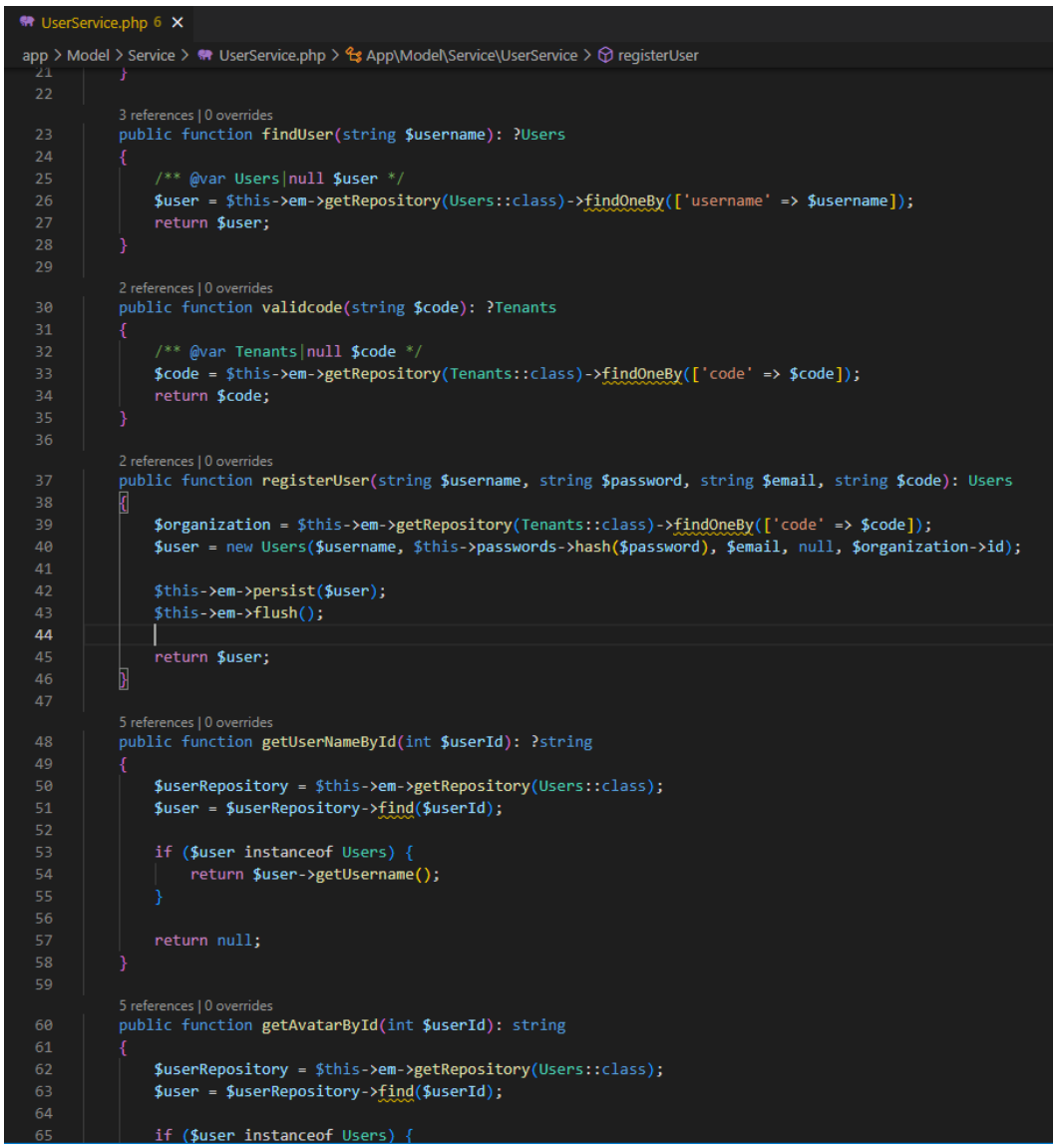
Funkce **getUserNameById**(int $userId): ?string: Tato funkce vyhledává uživatele podle uživatelského ID v databázi. Pokud je uživatel nalezen, funkce vrátí uživatelské jméno. Pokud uživatel neexistuje, funkce vrátí null.

Funkce **getAvatarById(int $userId)**: string je metoda, která se snaží získat cestu k obrázku avatara uživatele z databáze na základě ID uživatele. Nejprve získá repozitář pro entitu Users z entity manageru Doctrine ($this->em->getRepository(Users::class)). Entita Users představuje tabulku Users v databázi. Poté funkce volá metodu find na repozitáři uživatelů, aby získala uživatele s konkrétním ID ($userRepository->find($userId)). Kontroluje se, zda objekt uživatele existuje a je instancí třídy Users. To je důležité pro zabránění chyby, pokud by ID uživatele nebylo v databázi.

Pokud uživatel existuje, funkce získá avatar uživatele pomocí metody getAvatar ($user->getAvatar()). Tato metoda by měla vrátit cestu k souboru obrázku avatara uživatele.

Pokud uživatel má avatar (tj. výsledek metody getAvatar není prázdný), funkce vrátí tuto cestu. Pokud uživatel neexistuje nebo nemá avatar, funkce vrátí výchozí cestu k obrázku avatara ('/images/avatar/avatar-1.jpg').

Funkce **updateLastActivity**(int $userId): void: Tato funkce aktualizuje poslední aktivitu uživatele v databázi na aktuální čas. Uživatelské ID je vstupní parametr. Pokud uživatel existuje, funkce nastaví hodnotu "last_activity" na aktuální čas a uloží změny do databáze. Pokud uživatel neexistuje, funkce neudělá nic.



Obrázek 17: Ukázka kódu UserService.php

9.2 Soubor ProjectService.php

Funkce **getTenantId()**: Tato funkce kontroluje, zda je uživatel přihlášen. Pokud je přihlášen, metoda získá identitu přihlášeného uživatele a vrátí tenant_id, což je identifikátor nájemce, ke kterému uživatel patří.

Funkce **getUserId()**: Podobně jako funkce getTenantId, tato funkce kontroluje, zda je uživatel přihlášen. Pokud je přihlášen, získá se identita uživatele a vrátí se id uživatele.

Funkce **getProjectCountByTenantId()**: Tato funkce slouží k získání celkového počtu projektů pro specifického nájemce. Pro tento účel je použit repozitář 'Projects', z kterého je vytvořena query (dotaz) pro výběr počtu projektů, které mají konkrétní 'tenant_id'. Poté se tento počet vrací jako výsledek funkce.

Funkce **getProjectCompletedCountByTenantId()**: Tato funkce je podobná předchozí, ale navíc se zabývá pouze projekty, jejichž stav je označen jako 'completed'. V tomto případě je tedy výběr projektů filtrovaný nejen podle 'tenant_id', ale také podle stavu projektu. Výsledkem je počet dokončených projektů pro daného nájemce.

Funkce **getAllProjectsByTenantId()**: array: Tato funkce vrací pole všech projektů pro daného nájemce. Opět je použit repozitář 'Projects' pro vytvoření dotazu, který vrací projekty s daným 'tenant_id'. Navíc jsou projekty seřazeny podle jejich data splatnosti (atribut 'due_date') vzestupně. Výsledný seznam projektů je poté vrácen jako pole.

Funkce **getNotCompleteTaskCountByTenantId()**: Tato funkce vytváří dotaz, který vrátí počet nesplněných úkolů pro daného nájemce. Nesplněné úkoly jsou identifikovány stavem, který je jiný než 'completed'.

Funkce **getCompleteTaskCountByTenantId()**: Tato funkce vytváří dotaz, který vrátí počet splněných úkolů pro daného nájemce. Splněné úkoly jsou identifikovány stavem 'completed'.

Funkce **getTotalUserCountByTenantId**(): Tato funkce vytváří dotaz, který vrátí celkový počet uživatelů přidružených k danému nájemci.

Funkce **getUsersByActiveTenantId()**: Tato funkce vytváří dotaz, který vrátí seznam uživatelů přidružených k aktivnímu nájemci.

Funkce **getOrganizationNameByTenantId()**: Tato funkce vytváří dotaz, který vrátí název organizace přidružené k danému nájemci.

Funkce **getAllTasksByProjectId ()**: Tato funkce vytváří dotaz, který vrátí všechny úkoly přidružené k danému projektu, seřazené podle data splnění (due_date).

Funkce **deleteProjectById()**: Tato funkce nejprve najde projekt pomocí daného id, pak najde všechny úkoly přidružené k tomuto projektu a poté odstraní všechny tyto úkoly a samotný projekt z databáze.

Funkce **getProjectsByUserId()**: Tato funkce vytváří dotaz, který vrátí všechny projekty přidružené k danému uživateli.

Funkce **getTasksByUserId():** Tato funkce vytváří dotaz, který vrátí všechny úkoly přidružené k danému uživateli.

Funkce **getProjectCompletedCountByUserId()**: Tato funkce vrací počet projektů, které byly dokončeny (status = "completed") uživatelem, jehož ID je vráceno metodou $this->getUserId().

Funkce **getProjectCountByUserId()**: Tato funkce vrací celkový počet projektů spojených s uživatelem, jehož ID je vráceno metodou $this->getUserId().

Funkce **getProjectSuccessRateByUserId()**: Tato funkce vrací procentuální úspěšnost projektů uživatele. Úspěšnost je počítána jako podíl počtu dokončených projektů vůči celkovému počtu projektů, a to zaokrouhleno na dvě desetinná místa. Pokud uživatel nemá žádné projekty, funkce vrátí 0.

Funkce **getOldestRegistrationDateByTenantId()** je konkrétně určena k získání data nejstarší registrace uživatele v rámci daného nájemce (tenant). To znamená, že hledá uživatele s nejstarším datem registrace (registerdate) z databáze pro daný tenant.

```php
1 reference | 0 overrides
public function getProjectCountByTenantId()
{
    $repository = $this->em->getRepository(Projects::class);
    $count = $repository->createQueryBuilder('p')
        ->select('COUNT(p.id)')
        ->where('p.tenant_id = :tenant_id')
        ->setParameter('tenant_id', $this->getTenantId())
        ->getQuery()
        ->getSingleScalarResult();
    return $count;
}

1 reference | 0 overrides
public function getProjectCompletedCountByTenantId()
{
    $repository = $this->em->getRepository(Projects::class);
    $count = $repository->createQueryBuilder('p')
        ->select('COUNT(p.id)')
        ->where('p.tenant_id = :tenant_id')
        ->andWhere('p.status = :status')
        ->setParameter('tenant_id', $this->getTenantId())
        ->setParameter('status', 'completed')
        ->getQuery()
        ->getSingleScalarResult();
    return $count;
}

1 reference | 0 overrides
public function getAllProjectsByTenantId(): array
{
    $repository = $this->em->getRepository(Projects::class);
    $qb = $repository->createQueryBuilder('p');
    $qb->where('p.tenant_id = :tenantId')
    ->setParameter('tenantId', $this->getTenantId())
    ->orderBy('p.due_date', 'ASC');
    $projects = $qb->getQuery()->getResult();
    return $projects;
}
```

Obrázek 18: Ukázka kódu soubor ProjectService.php

10 PRESENTERY

Presenter slouží jako most mezi datovými modely (Model) a uživatelským rozhraním (View). Aplikace obsahuje několik presenterů, které přímo využívají funkce ze servis vrstvy, která manipuluje s daty z databáze.

10.1 Popis presenteru AdminPresenter.php

Funkce **startup()** v Nette frameworku je metoda volaná při spuštění každé presenteru (controlleru). Slouží k inicializaci presenteru a provádí některé společné úkoly před spuštěním konkrétní akce.

V této funkci se nejprve volá metoda parent::startup(), která zajišťuje spuštění rodičovské metody startup(), pokud je presenter rozšiřován od jiného presenteru.

Následuje blok podmínek, který řídí přístup k různým částem aplikace na základě stavu přihlášení uživatele a aktuální akce.

První podmínka kontroluje, zda uživatel není přihlášen ($this->getUser()->isLoggedIn() === false) a zároveň aktuální akce není 'signIn' ani 'registration'. Pokud tato podmínka platí, zobrazí se flash zpráva s textem "Tato sekce není přístupná bez přihlášení" s typem 'danger'. Následně dochází k přesměrování na stránku pro přihlášení pomocí $this->redirect('signIn').

Druhá podmínka kontroluje, zda uživatel je přihlášen ($this->getUser()->isLoggedIn() === true) a zároveň aktuální akce není 'dashboard' ani 'signOut'. Pokud tato podmínka platí, volá se funkce $this->userService->updateLastActivity($this->getUser()->getId()), která aktualizuje čas poslední aktivity uživatele. Poté dochází k přesměrování na stránku 'dashboard' pomocí $this->redirect('dashboard').

Tato funkce slouží k zajištění, že přístup k určitým částem aplikace je povolen pouze pro přihlášené uživatele a omezuje přístup pro nepřihlášené uživatele na určité akce.

Funkce **createComponentRegistrationForm()**, vytváří formulář s následujícími poli: 'username', 'password', 'email', a 'code'. Každé pole je vyžadováno, jak je indikováno metodou setRequired(). K formuláři je také přidána odesílací tlačítko s názvem 'submit'. Poté, co je formulář úspěšně odeslán, je vyvolána funkce 'registrationFormSuccess'. Tento formulář je následně vrácen jako výstup metody.

Druhá funkce **registrationFormSuccess(Nette\Application\UI\Form $form),** je zpětná volání (callback), která je vyvolána po úspěšném odeslání formuláře. V této metodě jsou získány hodnoty z formuláře a následně je provedena řada kontrol a akcí.

Prvně, funkce kontroluje, zda uživatel s daným uživatelským jménem již existuje. Pokud ano, zobrazí se zpráva o chybě a funkce je ukončena.

Dále se kontroluje, zda je kód validní (tj. zda patří nějaké organizaci). Pokud ne, zobrazí se další chybová zpráva a funkce je znovu ukončena.

Pokud obě tyto kontroly proběhnou úspěšně, je vytvořen nový uživatelský účet pomocí metody registerUser() ze servisu userService. Uživateli se zobrazí zpráva o úspěšné registraci.

Nakonec se pokusí přihlásit nově registrovaného uživatele pomocí metod login() a redirect(). Pokud dojde k chybě při přihlášení (např. pokud jsou uživatelské údaje nesprávné), zobrazí se chybová zpráva a uživatel je přesměrován na přihlašovací stránku.

Funkce **renderDashboard()** se používá k naplnění dat pro šablonu dashboardu. Tato funkce volá různé funkce z projectService, které získávají informace o projektech a úkolech (tasks) pro aktuálního tenant (nájemce).

Funkce **createComponentSignInForm()** slouží k vytvoření a konfiguraci instance formuláře pro přihlášení uživatele. Vrací objekt třídy Nette\Application\UI\Form, který představuje formulář. Ve metodě se vytváří nová instance formuláře pomocí new Nette\Application\UI\Form(). Následně jsou formuláři přidávány různé prvky pomocí metod addText(), addPassword() a addSubmit(). Funkce addText() přidává textové pole s názvem 'username' a popiskem 'Username'. Funkce addPassword() přidává heslové pole s názvem 'password' a popiskem 'Password'. Funkce addSubmit() přidává tlačítko odeslat s názvem 'send' a popiskem 'Sign In'. Dále je přidán callback do události onSuccess[], který se spustí při úspěšném odeslání formuláře. Callback je definován jako metoda $this->signInFormSuccess, která se nachází v tom samém třídním objektu. Tento callback bude zpracovávat údaje z formuláře po jeho odeslání.

Funkce **signInFormSuccess()** je callback, který se volá při úspěšném odeslání formuláře. Přijímá objekt $form třídy Nette\Application\UI\Form, který představuje odeslaný formulář.

V této funkci jsou nejprve získány hodnoty z formuláře pomocí metody getValues(). Poté se používá funkce $this->getUser()->login(), která slouží k přihlášení uživatele s uživatelským jménem a heslem získanými z formuláře. Pokud přihlášení proběhne úspěšně, je volána funkce $this->userService->updateLastActivity(), která aktualizuje čas poslední aktivity uživatele.

V případě, že dojde k chybě při přihlašování (Nette\Security\AuthenticationException), je zobrazena zpráva pomocí $this->flashMessage() s textem "Bylo zadáno špatné heslo" a typem 'danger'. Následně je proveden přesměrování na stránku pro přihlášení pomocí $this->redirect('signIn').

Pokud přihlášení proběhne úspěšně, je provedeno přesměrování na stránku 'dashboard' pomocí $this->redirect('dashboard').

```php
protected function createComponentRegistrationForm(): Nette\Application\UI\Form
{
    $form = new Nette\Application\UI\Form();
    $form->addText('username', 'Jméno:')
        ->setRequired();
    $form->addPassword('password', 'Heslo:')
        ->setRequired();
    $form->addEmail('email', 'Email:')
        ->setRequired();
    $form->addText('code', 'Kód:')
        ->setRequired();
    $form->addSubmit('submit', 'Registrovat');
    $form->onSuccess[] = [$this, 'registrationFormSuccess'];

    return $form;
}

1 reference
public function registrationFormSuccess(Nette\Application\UI\Form $form)
{
    $values = $form->getValues();
    $user = $this->userService->findUser($values->username);
    $validcode = $this->userService->validcode($values->code);

    if ($user != null) {
        $this->flashMessage('Tento uživatel již existuje', 'danger');
        return;
    }elseif($validcode == null){
        $this->flashMessage('Tento kód nepatří žádné organizaci', 'danger');
        return;
    }

    $this->userService->registerUser($values->username, $values->password, $values->email, $values->code);
    $this->flashMessage('Registrace proběhla úspěšně', 'success');

    try {
        $this->getUser()->login($values->username, $values->password);
        $this->redirect('signIn');
    } catch (Nette\Security\AuthenticationException $e) {
        $this->flashMessage($e->getMessage(), 'danger');
        $this->redirect('signIn');
    }
}
```

Obrázek 19: Ukázka kódu souboru AdminPresenter.php

10.2 Popis presenteru FilePresenter.php

Tento kód je součástí aplikace postavené na Nette Frameworku a obsahuje presenter FilePresenter, který spravuje práci se soubory uživatele. Presenter je v Nette odpovědný za přípravu dat pro zobrazení a zpracování uživatelských požadavků.

getUserNameById a getAvatarById: Metody pro získání jména a avataru uživatele podle jeho ID.

createDirectory: Metoda pro vytvoření složky pro daného uživatele, pokud ještě neexistuje.

getFiles: Metoda pro získání všech souborů daného uživatele. Tato metoda také formátuje informace o souborech do požadovaného formátu.

actionDeleteFile: Metoda pro smazání konkrétního souboru daného uživatele.

renderFiles a renderEdit: Metody pro přípravu dat pro zobrazení v šablonách "Files" a "Edit". Vytváří složku pro uživatele a připravuje data jako jméno uživatele, avatar a soubory pro zobrazení.

createComponentFileForm: Metoda pro vytvoření formuláře pro nahrání souborů.

FileFormSucceeded: Metoda, která se spustí po úspěšném odeslání formuláře pro nahrání souboru. Zkontroluje, zda soubor neexistuje a nepřekročil maximální povolenou velikost, a poté soubor nahraje.

Celkově tento presenter umožňuje uživateli nahrávat soubory do svého osobního adresáře, zobrazovat seznam nahraných souborů a mazat je.

```php
    1 reference | 0 overrides
    public function getFiles(int $userId): array
{
    $path = $_SERVER['DOCUMENT_ROOT'] . "/user_files/$userId";

    if (!is_dir($path)) {
        throw new \Exception('Složka neexistuje!');
    }

    $basePath = $this->getHttpRequest()->getUrl()->getBaseUrl();

    $files = [];
    foreach (Finder::findFiles('*')->from($path) as $file) {
        $sizeInBytes = $file->getSize();
        $sizeInMB = round($sizeInBytes / (1024 * 1024), 2);
        $created = date('Y-m-d H:i:s', $file->getMTime());
        $filePath = ltrim(str_replace($_SERVER['DOCUMENT_ROOT'], '', $file->getPathname()), '/');
        $url = $basePath . '/' . $filePath;
        $fileExtension = pathinfo($file->getFilename(), PATHINFO_EXTENSION);
        $files[] = [
            'name' => $file->getFilename(),
            'size' => $sizeInMB,
            'created' => $created,
            'url' => $url,
            'extension' => $fileExtension,
        ];
    }

    usort($files, function ($a, $b) {
        return strtotime($b['created']) - strtotime($a['created']);
    });

    return $files;
}

    0 references | 0 overrides
    public function actionDeleteFile(string $fileName): void
{
    $userId = $this->getUser()->getId();

    $path = $_SERVER['DOCUMENT_ROOT'] . "/user_files/$userId/$fileName";

    if (!file_exists($path)) {
        throw new \Exception('Soubor neexistuje!');
    }

    unlink($path);
    $this->flashMessage('Soubor byl úspěšně smazán.');
```

Obrázek 20: Ukázka kódu FilePresenter.php

11 ŠABLONY

Šablona obdrží data z presenteru (nejčastěji ve formě proměnných, které jsou nastaveny v metodě render nebo action) a na základě těchto dat generuje finální HTML stránku, kterou uvidí uživatel v prohlížeči. Šablony tedy propojují logiku aplikace s jejím vizuálním zobrazením. V kontextu aplikace existuje šablona pro dashboard, která zobrazuje přehledové informace pro uživatele, šablona pro správu úkolů (tasks), která umožňuje uživatelům vytvářet, upravovat a mazat úkoly, šablona pro správu projektů, která zobrazuje informace o projektech a umožňuje jejich správu, a další.

11.1 Nette Latte

Je to šablonovací systém používaný v Nette Frameworku. Představuje moderní a bezpečný způsob, jak psát a zpracovávat HTML šablony. Zde jsou některé ze základních vlastností a funkcionality Nette Latte:

Proměnné: Proměnné jsou do šablony vloženy pomocí složených závorek {}, například {$promenna}. Latte automaticky "escapuje" proměnné pro zabezpečení výstupu a zabránění Cross-site Scripting (XSS) útokům.

Makra: Makra jsou speciální direktivy, které umožňují manipulaci s HTML kódem. Například {link} pro vytvoření odkazu nebo {if} a {else} pro podmíněné bloky. Makra jsou značená pomocí složených závorek {} a mohou být uvnitř HTML tagů (např. n:href).

Podmíněné výrazy a cykly: Latte podporuje podmíněné výrazy a cykly. Například, {if $promenna} a {foreach $pole as $prvek}.

Filtry: Filtry umožňují zpracování proměnných předtím, než jsou vloženy do šablony. Například {$promenna|upper} převede hodnotu proměnné na velká písmena.

Dědění šablon: Může vytvářet "základní" šablony, které definují obecnou strukturu stránky, a pak dědit a přizpůsobovat tyto šablony pro konkrétní stránky.

Zkratky pro HTML tagy: Nabízí zkratky pro některé HTML tagy pro jednodušší a kratší psaní kódu. Například {input text "name", class => "form-control"}.

11.2 Popis a ukázka šablony

```
<nav class="navbar-vertical navbar">
    <div class="nav-scroller">
        <!-- Brand logo -->
        <a class="navbar-brand" n:href="Admin:Dashboard">
            <img src="{$basePath}/images/brand/logo/logo.svg" alt="" />
        </a>
        <!-- Navbar nav -->
        <ul class="navbar-nav flex-column" id="sideNavbar">
            <li class="nav-item">
                <a n:class="'nav-link has-arrow', $presenter->isLinkCurrent('Admin:Dashboard') ? 'active'" n:href="Admin:Dashboard">
                    <i data-feather="home" class="nav-icon icon-xs me-2"></i>Nástěnka
                </a>
            </li>

            <li class="nav-item">
                <a n:class="'nav-link', $presenter->isLinkCurrent('Project:user') ? 'active'" n:href="Project:user">
                    <i data-feather="layers" class="nav-icon icon-xs me-2"></i>Projekty
                </a>
            </li>

            <li class="nav-item">
                <a n:class="'nav-link', $presenter->isLinkCurrent('Task:user') ? 'active'" n:href="Task:user">
                    <i data-feather="list" class="nav-icon icon-xs me-2"></i>Úkoly
                </a>
            </li>

            <li class="nav-item">
                <a n:class="'nav-link', $presenter->isLinkCurrent('User:Edit') ? 'active'" n:href="User:Edit">
                    <i data-feather="settings" class="nav-icon icon-xs me-2"></i>Nastavení
                </a>
            </li>

        </ul>

    </div>
</nav>
```

Obrázek 21: Ukázka souboru admin.latte

Zobrazený kód představuje část šablony v jazyce Latte, která definuje navigační menu. Toto menu je použito univerzálně napříč celou aplikací, což znamená, že je viditelné na všech stránkách.

n:href: Tato direktiva je zkratka pro vytvoření odkazu pomocí metody presenteru link(). V ukázce, n:href="Admin:Dashboard" vytváří odkaz na akci Dashboard v presenteru Admin.

{$basePath}: Latte používá složené závorky {} pro vkládání proměnných do šablony. $basePath je proměnná, která odkazuje na základní cestu aplikace.

n:class: Tato direktiva je použita pro dynamické nastavení třídy HTML elementu. V ukázce, n:class="'nav-link has-arrow', $presenter->isLinkCurrent('Admin:Dashboard') ? 'active'" vytváří třídu active, pokud je aktuální stránka "Admin:Dashboard".

12 BEZPEČNOST APLIKACE

Bezpečnost je jednou z klíčových priorit frameworku Nette. Obsahuje několik vestavěných bezpečnostních funkcí a poskytuje nástroje, které pomáhají předcházet běžným bezpečnostním hrozbám.

12.1 Základní použité bezpečnostní prvky aplikace

Safe templates (Latte): Tento šablonovací systém automaticky escapuje výstup, což chrání aplikaci před útoky typu Cross-Site Scripting (XSS).

CSRF Protection: Nette automaticky implementuje ochranu proti Cross-Site Request Forgery (CSRF) útokům v předvolbě formulářů.

Password Hashing: Obsahuje vestavěnou podporu pro bezpečné hashování hesel pomocí moderních algoritmů.

Input Validation: Obsahuje robustní funkce pro validaci vstupu, které pomáhají chránit aplikaci před neplatnými nebo potenciálně škodlivými daty.

Ochrana proti SQL Injection: Jedním z klíčových bezpečnostních prvků Doctrine je použití parametrizovaných dotazů, které jsou účinné proti SQL injection útokům. Parametrizované dotazy zajišťují, že vstup od uživatele je správně escapován, než je vložen do SQL dotazu, což zabraňuje spuštění škodlivého kódu.

12.2 Přihlášení do aplikace

Kód obsahuje několik bezpečnostních prvků, které jsou poskytnuty prostřednictvím Nette Framework:

Ochrana proti CSRF útokům: Formulář vytvořený pomocí třídy Nette\Application\UI\Form automaticky obsahuje ochranu proti Cross-Site Request Forgery (CSRF) útokům. Tato ochrana zajišťuje, že každý požadavek na odeslání formuláře je jedinečný a nemůže být zneužit třetí stranou.

Ochrana proti SQL Injection: Váš kód využívá funkci login() z třídy Nette\Security\User, která je bezpečná proti SQL injection útokům. Tato metoda využívá parametrizované dotazy, které zabrání vkládání škodlivého kódu do SQL dotazů.

Bezpečné ukládání hesel: metoda login() využívá hashování hesel. Hashování je důležité pro bezpečnost hesel, protože pokud by byla databáze napadena, získal by útočník pouze hash hesel, který je bez původního hesla prakticky k ničemu.

Ošetření chyb při autentizaci: Metoda signInFormSuccess() obsahuje blok try/catch, který zachytává výjimky Nette\Security\AuthenticationException vyhozené při neúspěšné autentizaci. To umožňuje aplikaci bezpečně zpracovat neúspěšné pokusy o přihlášení bez ukončení skriptu nebo zobrazení citlivých informací uživateli.

```php
0 references
protected function createComponentSignInForm(): Nette\Application\UI\Form
{
    $form = new Nette\Application\UI\Form();
    $form->addText('username', 'Username');
    $form->addPassword('password', 'Password');
    $form->addSubmit('send', 'Sign In');
    $form->onSuccess[] = [$this, 'signInFormSuccess'];

    return $form;
}

1 reference
public function signInFormSuccess(Nette\Application\UI\Form $form)
{
    $values = $form->getValues();
    try {
        $this->getUser()->login($values->username, $values->password);
        $this->userService->updateLastActivity($this->getUser()->getId());
    } catch (Nette\Security\AuthenticationException $e) {
        $this->flashMessage("Bylo zadáno špatné heslo", 'danger');
        $this->redirect('signIn');
    }

    $this->redirect('dashboard');
}
```

Obrázek 22: Ukázka kódu (přihlášení do aplikace)

12.3 Rozšiřující třída Authenticator.php

Soubor Authenticator.php je rozšiřující třída, která implementuje rozhraní \Nette\Security\Authenticator z Nette Framework. Toto rozhraní definuje, jak by měla být provedena autentizace uživatele v aplikaci.

Tato třída zajišťuje bezpečnost autentizace tím, že používá bezpečné metody pro ověřování hesel a správně zachází s chybami při autentizaci.

V následujících odstavcích je konkrétní popis kódu v souboru.

Konstruktor: Konstruktor třídy Authenticator přijímá dvě závislosti: službu UserService a instanci třídy \Nette\Security\Passwords. UserService je vlastní služba, kterou jste vytvořili pro správu uživatelů ve vaší aplikaci. Třída \Nette\Security\Passwords je součástí Nette Framework a poskytuje metody pro bezpečné práci s hesly, například pro jejich hashování a verifikaci.

Metoda **authenticate**: Metoda authenticate je hlavní metoda třídy Authenticator, která se zavolá, když se uživatel pokouší přihlásit. Tato metoda přijímá uživatelské jméno a heslo a vrátí identitu uživatele, pokud je přihlášení úspěšné, nebo vyvolá výjimku, pokud je přihlášení neúspěšné.

Metoda nejprve hledá uživatele podle jména pomocí metody findUser() z UserService. Pokud uživatel není nalezen, vyhodí výjimku AuthenticationException s chybovou zprávou.

Poté ověřuje zadané heslo uživatele s uloženým hashem hesla. Metoda verify() třídy \Nette\Security\Passwords porovná zadané heslo s jeho hashem. Pokud hesla neodpovídají, vyhodí se výjimka AuthenticationException.

Nakonec, pokud jsou údaje správné, metoda vrátí novou identitu uživatele prostřednictvím třídy \Nette\Security\SimpleIdentity. Tato identita obsahuje ID uživatele, pole rolí (v tomto případě obsahuje pouze uživatelské jméno) a pole dalších dat (v tomto případě obsahuje ID tenantu).

Authenticator.php

app > Model > Authenticator.php > Authenticator > _construct

```php
<?php declare(strict_types=1);

use App\Model\Service\UserService;
use Nette\Security\IIdentity;

2 references | 0 implementations
class Authenticator implements \Nette\Security\Authenticator
{

    1 reference | 0 overrides
    public function __construct(
        private UserService $userService,
        private \Nette\Security\Passwords $passwords,
    ) {}

    2 references | 0 overrides
    public function authenticate(string $username, string $password): IIdentity
    {
        $user = $this->userService->findUser($username);

        if ($user === null)
            throw new \Nette\Security\AuthenticationException('Uživatel nebyl nalezen.');

        if ($this->passwords->verify($password, $user->password) === false)
            throw new \Nette\Security\AuthenticationException('Špatné heslo.');

        return new \Nette\Security\SimpleIdentity($user->id, ['username' => $user->username], ['tenant_id' => $user->tenant_id]);
    }
}
```

Obrázek 23: Ukázka kódu třídy Authenticator

13 TESTOVÁNÍ A LADĚNÍ

13.1 Tracy

Nette Framework poskytuje pro účely ladění integraci s ladícím nástrojem jménem Tracy. Tracy je robustní ladící nástroj, který je speciálně navržený pro optimalizaci a zjednodušení procesu odhalování a opravování chyb v kódu. Tento nástroj je vysoce konfigurovatelný a nabízí řadu funkcí, které usnadňují práci vývojářů a pomáhají jim při různých fázích testování a ladění aplikace.

Základní dovednosti nástroje Tracy:

Přehledný výpis chyb: Tracy poskytuje velmi podrobné zprávy o chybách, včetně stack trace, proměnných v různých částech kódu a dalších užitečných informací.

Zachycení výjimek: Tracy automaticky zachytává nezachycené výjimky a zobrazuje je v přehledné formě.

Zachycení chyb: Tracy také zachytává PHP chyby a varování, které by jinak mohly být přehlédnuty. To zahrnuje chyby typu E_NOTICE, E_WARNING a další.

Pohodlné logování: Tracy poskytuje funkcionalitu pro snadné logování zpráv a dat, které můžete použít pro sledování toho, co se děje v kódu.

Panel pro vývojáře (Debugger Bar): Tracy zahrnuje panel pro vývojáře, který zobrazuje užitečné informace o aktuální stránce, jako je doba generování stránky, použití paměti, počet SQL dotazů a další.

Zápis chyb do logů: Tracy může automaticky zapisovat chyby a výjimky do logovacích souborů.

Podpora AJAXu a CLI: Tracy správně zobrazuje chyby při Ajaxových požadavcích a při spouštění PHP skriptů z příkazové řádky.

Zabezpečení: Tracy je navržen tak, aby byl bezpečný pro použití i na produkčních serverech. V ladícím módu se detaily chyb zobrazují pouze na základě konfigurace IP adresy nebo cookies.

13.2 Testování PhpUnit

Samotný soubor UserTest.php obsahuje několik testovacích metod pro testování funkcionality třídy UserService.

Testovací metoda **testFindUser** ověřuje správnou funkčnost metody findUser z třídy UserService. Vytváří se instance EntityManagerInterface a EntityRepository pomocí funkce createMock. Poté se nastavují očekávaná chování těchto objektů pomocí metod expects a willReturn. Testovací metoda volá metodu findUser s testovacím jménem uživatele a porovnává očekávaný výsledek s vráceným uživatelem.

```
0 references | 0 overrides
public function testFindUser(): void
{
    $username = 'testuser';
    $expectedUser = $this->createMock(App\Model\Entity\Users::class);

    $entityManager = $this->createMock(EntityManagerInterface::class);
    $repository = $this->createMock(Doctrine\ORM\EntityRepository::class);

    $entityManager->expects($this->once())
        ->method('getRepository')
        ->willReturn($repository);

    $repository->expects($this->once())
        ->method('findOneBy')
        ->with(['username' => $username])
        ->willReturn($expectedUser);

    $this->userService->setEntityManager($entityManager);

    $user = $this->userService->findUser($username);

    $this->assertSame($expectedUser, $user);
}
```

Obrázek 24: Testovací funkce pro FindUser

Testovací metoda **testValidCode** testuje metodu validcode z třídy UserService. Opět se vytváří instance EntityManagerInterface a EntityRepository pomocí createMock. Následně se nastavuje očekávané chování těchto objektů pomocí metod expects a willReturn. Testovací metoda volá metodu validcode s testovacím kódem a porovnává očekávaný výsledek s vráceným tenantem.

Poslední testovací metoda **testRegisterUser** provádí test metody registerUser z třídy UserService. V této metodě se vytvářejí testovací hodnoty pro jméno uživatele, heslo, email a kód. Dále se vytváří instance EntityManagerInterface, EntityRepository, Passwords a očekávaného tenantu pomocí createMock. Poté se nastavují očekávaná chování těchto objektů pomocí metod expects a willReturn. Testovací metoda volá metodu registerUser s testovacími hodnotami a porovnává, zda vrácený uživatel je instancí třídy Users.

Tyto testovací metody slouží k ověření správné funkcionality jednotlivých metod třídy UserService a zajistí, že očekávané výsledky jsou shodné s tím, co bylo předem definováno.

```
# vendor/bin/phpunit --testdox tests/UserTest.php
PHPUnit 9.6.8 by Sebastian Bergmann and contributors.

User
 ✓ Find user
 ✓ Valid code
 ✓ Register user

Time: 00:00.703, Memory: 6.00 MB

OK (3 tests, 10 assertions)
```

Obrázek 25: Ukázka výstupu PhpUnit

14 UKÁZKA APLIKACE

14.1 Dashboard

Navbar: Postranní navigační panel (navbar) s logem aplikace a odkazy na různé části aplikace. Každý odkaz obsahuje ikonu a popis (Nástěnka, Projekty, Úkoly, Soubory, Nastavení). Aktivní stránka má v navigačním panelu zvýrazněný odkaz.

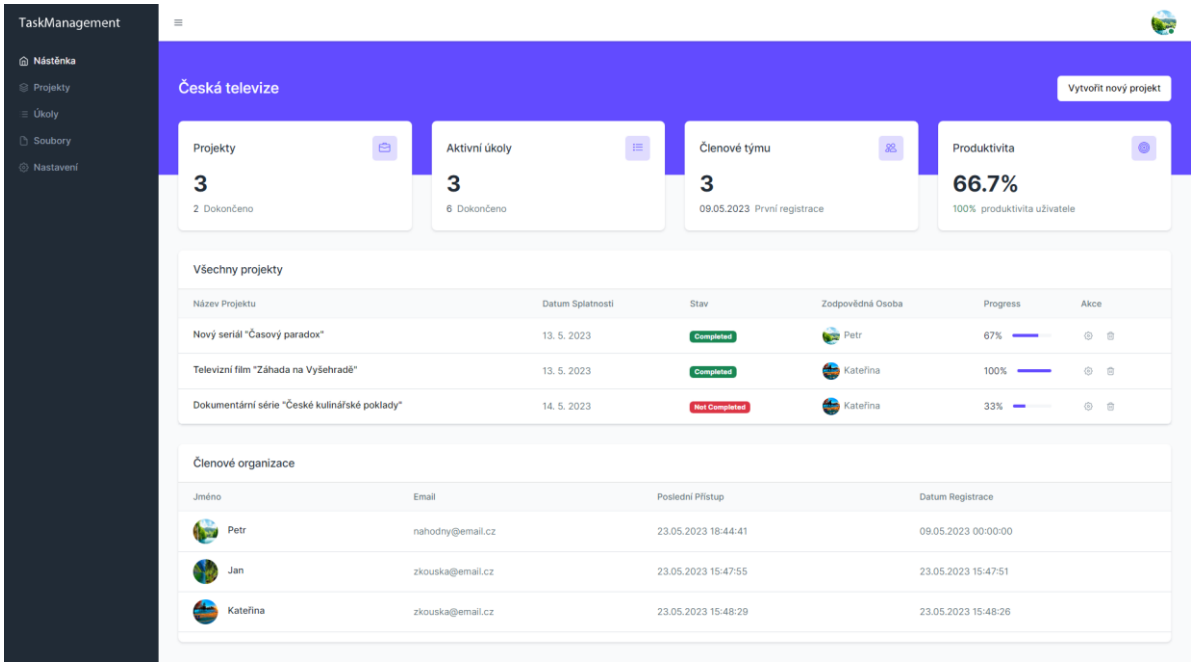
Uživatelský profil: V horním navigačním panelu je avatar uživatele, který, pokud na něj kliknete, otevře rozbalovací menu s odkazy na nastavení profilu a odhlášení.

Obsah stránky: Obsah stránky zahrnuje hlavičku stránky s názvem organizace a tlačítkem pro vytvoření nového proje ktu. Dále zde můžete vidět několik karet zobrazujících informace o Projektech, Aktivních úkolech, Členech týmu a Produktivitě.

Tabulky: První tabulka zobrazuje seznam všech projektů s jejich názvem, datem splatnosti, stavem, zodpovědnou osobou, progressem a akcemi. Druhá tabulka zobrazuje seznam všech členů organizace, jejich jména, emaily, datum posledního přístupu a datum registrace.

Grafické prvky: Na stránce je také k vidění několik grafických prvků, jako jsou progress bary ukazující postup projektu a ikony různých funkcí.

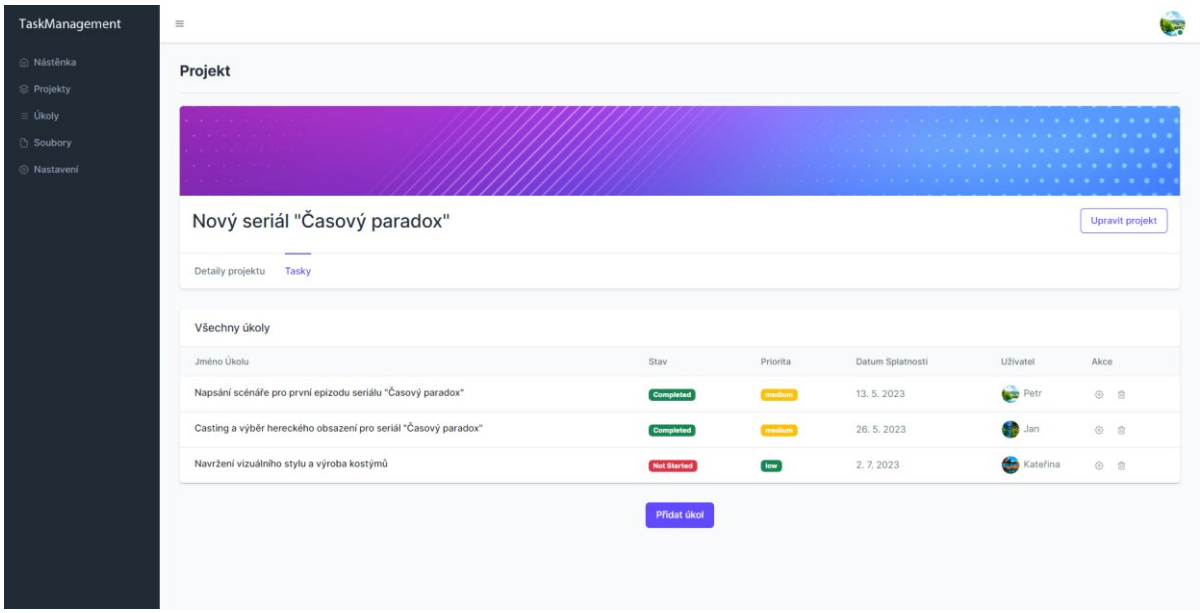
Dynamický obsah: Některý obsah na stránce je dynamický, což znamená, že se může měnit podle toho, jaký data má server k dispozici. Například informace o počtu projektů, úkolů, členů týmu a produktivitě se mohou měnit v závislosti na aktuálních datech.



Obrázek 26: Dashboard Aplikace

14.2 Tasks

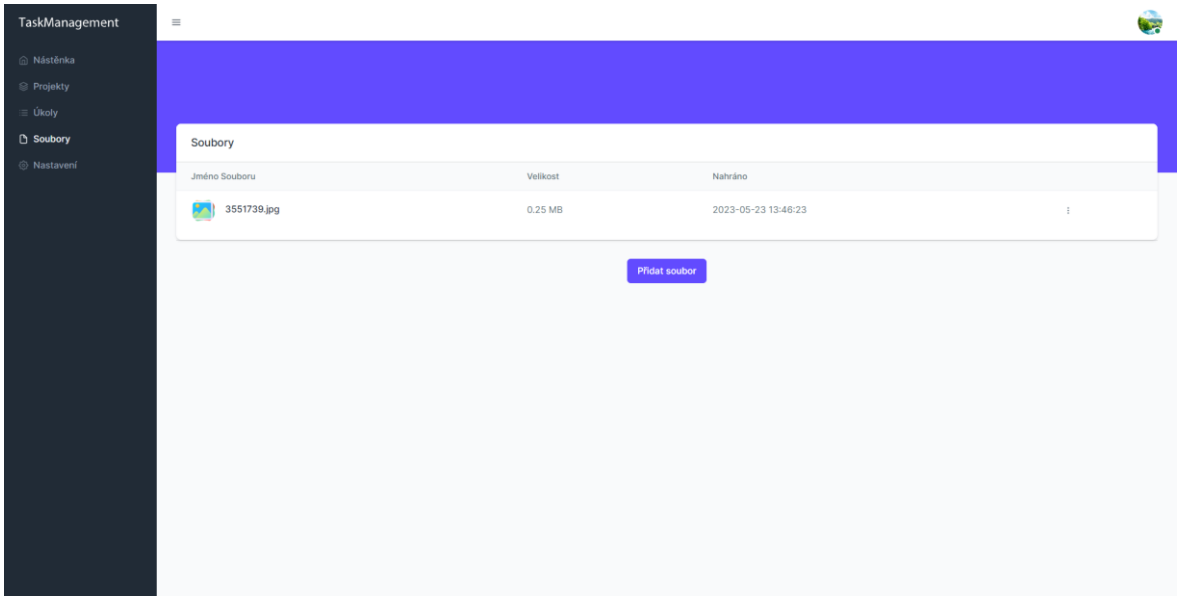
Tato stránka reprezentuje všechny Tasky přiřazené k jednotlivým projektům. U tasků můžeme vidět v panelu všech úkolů stav, prioritu, datum splatnosti, uživatele a akce. Tyto tasky můžeme upravovat nebo mazat. Můžeme také přidávat úkoly pomocí tlačítka pod tabulkou.



Obrázek 27: Ukázka šablony s Tasky

14.3 Soubory

Tato šablona zobrazuje všechny nahrané soubory daného uživatele na serveru. Každý uživatel má svůj prostor na serveru, ke kterému má přístup výhradně pouze on. Soubory se vykreslují v rámci řádků v tabulce, kdy má každý nahraný soubor svou specifickou ikony dle formátu. V tabulce se také zobrazuje velikost nahraného souboru a datum s časem, kdy byl soubor nahrán. V tabulce je u každého souboru k dispozici také nabídka, ve které můžeme soubor zobrazit, zkopírovat do schránky nebo smazat. Pod tabulkou je k dispozici tlačítko, kterým můžeme soubory libovolně přidávat přes zobrazený formulář.



Obrázek 28: Ukázka přehledu souborů

ZÁVĚR

Cílem práce bylo vytvořit efektivní nástroj pro organizaci a sledování pracovních aktivit uživatelů. Aplikace umožňuje vytváření projektů, přidávání a sledování úkolů s důrazem na individuální přístup k datům pro každého nájemce, což zajišťuje bezpečnost a soukromí.

Pro vývoj aplikace byly využity moderní technologie a nástroje, včetně frameworku Nette a objektově-relačního mapování Doctrine. Tyto technologie poskytují robustní a spolehlivé základy pro aplikaci a zároveň umožňují snadnou rozšiřitelnost.

Výsledná aplikace nabízí uživatelům možnost lepší organizace práce, spolupráce s kolegy a dosažení vyšší produktivity ve sdíleném pracovním prostředí. Díky pokročilým funkcím a intuitivnímu uživatelskému rozhraní je práce s aplikací snadná a efektivní.

Bakalářská práce přináší praktickou ukázku vývoje webové aplikace s využitím moderních technologií a přínosy, které taková aplikace může poskytnout v oblasti správy projektů a úkolů ve firemním prostředí.

Kompletní aplikace s návodem pro zprovoznění na lokálním serveru je k dispozici na CD nebo v systému IS/STAG.

SEZNAM POUŽITÉ LITERATURY

[1] PROCHÁZKA, David. CSS a XHTML: tvorba dokonalých WWW stránek krok za krokem. 2., aktuali. vyd. Praha: Grada, 2011. Průvodce (Grada). ISBN 978-80-247-3897-0.

[2] File Transfer Protocol (FTP) Meaning and Definition. In: Fortinet [online]. Sunnyvale, CA 94086 USA: Fortinet, © 2023 [cit. 2023-05-02]. Dostupné z: https://www.fortinet.com/resources/cyberglossary/file-transfer-protocol-ftp-meaning

[3] STRUKTURA HTML DOKUMENTU. In: Rascasone [online]. Praha: rascasone, 2021 [cit. 2023-04-04]. Dostupné z: https://www.rascasone.com/cs/blog/html-pro-zacatecniky-jak-psat-web

[4] CSS, SCSS and Less. In: Visual Studio Code [online]. Seattle: Microsoft, 2023 [cit. 2023-04-04]. Dostupné z: https://code.visualstudio.com/docs/languages/css

[5] JOHNSON, Glenn. Programming in HTML5 with JavaScript and CSS3: training guide. Redmond, Washington: Microsoft, 2013. ISBN 978-0735674387.

[6] JAVASCRIPT PRO ZAČÁTEČNÍKY: CO TO JE A JAK FUNGUJE. Rascasone [online]. Prosecká 527/24, Libeň, 180 00 Praha: Rascasone, 2022 [cit. 2023-04-02]. Dostupné z: https://www.rascasone.com/cs/blog/co-je-javascript-pro-zacatecniky

[7] Using JavaScript Syntax Coloring. In: Eclipse [online]. Eclipse [cit. 2023-04-04]. Dostupné z: https://www.eclipse.org/pdt/help/html/using_javascript_syntax_coloring.htm

[8] JAKOBUS, Benjamin. Mastering Bootstrap 4: Master the latest version of Bootstrap 4 to build highly customized responsive web apps. Packt Publishing Ltd, 2018

[9] Bootstrap. WhatIs.com [online]. Grove Street, Newton, MA 02466, USA: TechTarget, 2022 [cit. 2023-04-02]. Dostupné z: https://www.techtarget.com/whatis/definition/bootstrap

[10] Custom Code. In: Bootstrapstudio [online]. bootstrapstudio [cit. 2023-04-04]. Dostupné z: https://bootstrapstudio.io/docs/custom-code.html#basics

[11] PROCHÁZKA, David. PHP 6: začínáme programovat. Praha: Grada, 2012. Průvodce (Grada). ISBN 978-80-247-3899-4.

[12] 5 Easy Ways to Syntax Highlight PHP Code. In: Perishablepress [online]. perishablepress, 2022 [cit. 2023-04-04]. Dostupné z: https://perishablepress.com/5-easy-ways-to-display-syntax-highlighted-php-code/

[13] What is SQL?. AWS Amazon [online]. Seattle, WA 98108-1226: Amazon Web Services, 2023 [cit. 2023-04-02]. Dostupné z: https://aws.amazon.com/what-is/sql/

[14] BEN-GAN, Itzik; DAVIDSON, Louis; VARGA, Stacia. MCSA SQL Server 2016 Database Development Exam Ref 2-pack: Exam Refs 70-761 and 70-762. Microsoft Press, 2017.

[15] PhpMyAdmin. JavatPoint [online]. Noida, UP, 201301, India: JavatPoint, © 2011-2021 [cit. 2023-05-02]. Dostupné z: https://www.javatpoint.com/phpmyadmin

[16] PhpMyAdmin. In: Synology [online]. New Taipei City: Synology, © 2023 [cit. 2023-05-02]. Dostupné z: https://www.synology.com/cs-cz/dsm/packages/phpMyAdmin

[17] Nette Framework. Wikipedie [online]. San Francisco: Wikipedie, 2022 [cit. 2023-04-02]. Dostupné z: https://cs.wikipedia.org/wiki/Nette_Framework

[18] Přehled ASP.NET Core MVC. Microsoft [online]. USA: Microsoft, 2023 [cit. 2023-04-02]. Dostupné z: https://learn.microsoft.com/cs-cz/aspnet/core/mvc/overview?view=aspnetcore-7.0

[19] GORETTI, Anthony. Introducing ASP. NET Core 6. In: Beginning gRPC with ASP. NET Core 6. Apress, Berkeley, CA, 2022. p. 33-81.

[20] Getting Started with Doctrine. Doctrine Projects [online]. Doctrine Projects, 2023 [cit. 2023-05-13]. Dostupné z: https://www.doctrine-project.org/projects/doctrine-orm/en/2.15/tutorials/getting-started.html

[21] What is multi-tenant?. Ibm [online]. New York: Ibm, 2022 [cit. 2023-04-02]. Dostupné z: https://www.ibm.com/topics/multi-tenant

[22] Hybris Multi-Tenant System Using REST Webservices. In: Dzone [online]. Durham: dzone, 2022 [cit. 2023-04-04]. Dostupné z: https://dzone.com/articles/hybris-multi-tenant-system-using-rest-webservices

[23] Asana. Asana [online]. San Francisco: Asana, 2023 [cit. 2023-05-13]. Dostupné z: https://asana.com/

[24] Trello. Trello [online]. San Francisco: Atlassian, 2023 [cit. 2023-05-13]. Dostupné z: https://trello.com/cs

[25] UNHELKAR, Bhuvan. Software engineering with uml. Broken Sound Parkway NW: CRC Press, 2018. ISBN 978-1-138-29743-2.

SEZNAM OBRÁZKŮ