Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Bakalářská práce

PHP frameworky

David Pocar

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

David Pocar

Informatika

Název práce

PHP frameworky

Název anglicky

PHP frameworks

Cíle práce

Bakalářská práce je tematicky zaměřena na problematiku vývoje webových aplikací za použití PHP frameworků.
Hlavním cílem práce je analyzovat a zhodnotit vybrané PHP frameworky vhodné pro vývoj webových stránek.
Dílčí cíle práce jsou:
- charakterizovat problematiku vývoje webových aplikací,
- analyzovat možnosti využití PHP frameworků při tvorbě webových aplikací,
- zhodnocení vybraných PHP frameworků.

Metodika

V teoretická část bakalářské práce se bude zakládat na analýze a rešerši odborných zdrojů.

V praktické části práce budou na základě poznatků zjištěných v analytické části zhodnoceny vybrané PHP frameworky a formulovány možnosti využití jednotlivých frameworků pro vybrané typy webových aplikací.

Na základě syntézy teoretických a praktických poznatků budou zpracovány závěry bakalářské práce.

Doporučený rozsah práce

35

Klíčová slova

php, mysql, mvc, framework, webové aplikace

Doporučené zdroje informací

Altaf Hussain. Learning PHP 7 High Performance. Packt Publishing, 2016. ISBN 9781785881633.

Armando Padilla, DUPTim Hawkins. Pro PHP Application Performance: Tuning PHP Web Projects for Maximum Performance. Apress, 2011. ISBN 9781430228998.

BÖHMER, M. *Návrhové vzory v PHP : [23 vzorových postupů pro rychlejší vývoj].* Brno: Computer Press, 2012. ISBN 978-80-251-3338-5.

CASTAGNETTO, J. *PHP : programujeme profesionálně.* Praha: Computer Press, 2001. ISBN 80-7226-310-2.

Chris Pitt. Pro PHP MVC. Apress, 2012. ISBN 1430241640.

Keyvan Akbary, Christian Soronellas, Carlos Buenosvinos. Domain-Driven Design in PHP. Packt Publishing Ltd, 2017. ISBN 9781787284944.

Předběžný termín obhajoby

2018/19 LS – PEF

Vedoucí práce

Ing. Michal Stočes, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 15. 10. 2018

Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 19. 10. 2018

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 21. 02. 2019

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci „PHP frameworky“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 15.3.2018 _____

Poděkování

Rád bych touto cestou poděkoval své rodině, přátelům a především Ing. Michalovi Stočesovi, Ph.D. za cenné rady, věcné připomínky a vstřícnost při konzultacích při vypracovávání bakalářské práce.

PHP frameworky

Abstrakt

Bakalářská práce pojednává o problematice vývoje webových aplikací použitím PHP frameworků. V první řadě byla charakterizována problematika vývoje webových aplikací. Poté na základě určených hledisek byly zvoleny konkrétní PHP frameworky. Těmito frameworky byly vyvinuty zkušební webové aplikace, které napomáhaly v hodnocení dle určených kritérií. U každého PHP frameworku se hodnotila následující kritéria: rychlost, bezpečnost, dokumentace, práce s databází, šablonovací systém, vývojové konzole a ladící nástroje. Dále následovalo samotné hodnocení PHP frameworků. Díky získaným informacím byly formulovány možnosti využití každého frameworku.

Klíčová slova: Codeigniter, Laravel, Nette, Symfony, Yii, framework, webová aplikace, PHP, MySQL, MariaDB, Apache, MVC, MVP

PHP frameworks

Abstract

The bachelor thesis deals with problematics of web applications and its development using PHP frameworks. First of all, the issue involving web application has been characterized. After that, specific PHP frameworks have been chosen based on the specific aspects. With these frameworks, demo web applications have been developed to help with the evaluation of the criteria. For each PHP framework, the following criteria were evaluated: speed, security, documentation, work with database, templating engine, development console, and debugging tools. The very next thing being evaluated were PHP frameworks themselves. With the information obtained, the possibilities of using each framework were formulated.

Keywords: Codeigniter, Laravel, Nette, Symfony, Yii, framework, web application, PHP, MySQL, MariaDB, Apache, MVC, MVP

Obsah

Seznam obrázků

Seznam tabulek

Seznam grafů

1 Úvod

Dnešní moderní společnost a její okolí je stále více ovlivňována různými technologiemi. Těchto technologií je nespočet a postupem času se neustále vyvíjí. Jejich množství neustále vzrůstá. Vhodným příkladem je právě internet. V současnosti je kolem 40 %[1] populace připojeno k internetové síti. To značně zvyšuje využívání různých sociálních sítí a online služeb. Neustále se objevují novější a novější weby. Příkladem může být podnikový web, e-shop, nebo nějaká nová obdoba Facebooku či Twitteru. V minulosti vytvářet takový obsah bylo velice složité a jen určitá skupina lidí takovou schopností disponovala. Dnes je tomu však jinak. Existuje nespočet služeb, které nabízí automatické vytvoření webu, a je k tomu zapotřebí pouze pár kliknutí myší. Ale i takové služby musel někdo vyvinout.

Způsobů, jakými se dá webový obsah vytvářet, je nespočet a jedná se tudíž o velice probíranou problematiku. Tvůrci webových stránek nebo aplikací mají široké spektrum možností, jak takové realizace dosáhnout. To však vede k jedné otázce. Jaký způsob je ten nejlepší? Dá se říci, že takový způsob neexistuje. Vždy bude existovat alternativa, která má určité výhody i nevýhody. Proto je lepší si najít způsob, který dotyčnému nejvíce vyhovuje.

K realizaci webových aplikací je ve většině případů potřeba mnoho úsilí, času a případně kooperaci dalších lidí. Lidé jsou navíc rozdílní a každý má své postupy, jak řešit určité problémy. Při vytváření každé nové aplikace je potřeba napsat velmi složitý kód, což nakonec vedlo ke psaní vlastních knihoven, pomocných funkcí a objektů, které zjednodušovaly vývoj. Z takových knihoven a pomocných kódů nakonec vzešly frameworky. Frameworky mají za úkol právě usnadnit vývoj, sjednotit postupy k řešení konkrétních problémů a zpřehlednit i samotný kód.

V oblasti informačních technologií je tato problematika ohledně vývoje webových aplikací a frameworků běžně probíraným tématem. Snahou této práce je aplikovat zjištěné poznatky do praktického života a přinést pozitivní přínos pro případné budoucí účely.

Hlavním zdrojem informací této práce byla doporučená odborná literatura a další odborné zdroje.

[1] Zdroj: http://www.internetlivestats.com/internet-users

2 Cíl práce a metodika

2.1 Cíle práce

Bakalářská práce je tematicky zaměřena na problematiku vývoje webových aplikací za použití PHP frameworků.

Hlavním cílem práce je analyzovat a zhodnotit vybrané PHP frameworky vhodné pro vývoj webových stránek.

Dílčí cíle práce jsou:

- charakterizovat problematiku vývoje webových aplikací
- analyzovat možnosti využití PHP frameworků při tvorbě webových aplikací
- zhodnocení vybraných PHP frameworků

2.2 Metodika

Teoretická část bakalářské práce se bude zakládat na analýze a rešerši odborných zdrojů.

V praktické části práce budou na základě poznatků zjištěných v analytické části zhodnoceny vybrané PHP frameworky a formulovány možnosti využití jednotlivých frameworků pro vybrané typy webových aplikací.

Na základě syntézy teoretických a praktických poznatků budou zpracovány závěry bakalářské práce.

3 Teoretická východiska

3.1 Vývoj webových aplikací

Kapitola se zabývá náležitostmi spjatými s vývojem webových aplikací a odbornými termíny s nimi spojenými.

3.1.1 Internet

Společnost si pod pojmem internet dokáže představit pouze konkrétní webovou stránku, například Twitter. Ve skutečnosti je to ale systém počítačových sítí celosvětového měřítka, které jsou navzájem propojeny. K samotné komunikaci mezi ostatními uzly (stanice, počítač, tiskárna, server) využívá řadu protokolů. Protokoly jsou standardem určitých pravidel postupů pro datovou komunikaci a jejich dodržení je nezbytně nutné k úspěšnému přenosu dat (Internet Society, 2003).

TCP/IP (anglicky *Transmission Control Protocol/Internet Protocol*) je architekturou, konceptuálním modelem nebo soustavou protokolů internetu, který se skládá ze čtyř vrstev komunikace (Alani, 2014). Těmito vrstvami jsou:

- Aplikační vrstva
- Transportní vrstva
- Síťová vrstva (IP)
- Vrstva síťového rozhraní

S ohledem na téma webových aplikací je aplikační vrstva ta nejdůležitější, jelikož se stará o přenos samotných dat, jak je společnost zná. Jde například o posílání a přijímání emailů. Hlavním důvodem je však HTTP, což je protokol určený ke komunikaci mezi klientem a serverem (The Apache Software Foundation, 2018).

Klient se dá považovat za webový prohlížeč a server jako samotná webová aplikace. Principem tohoto protokolu je předávání zpráv mezi zmíněnými konci. Těmto zprávám se říká požadavky (žádosti) a odpovědi. Klient tedy posílá žádost, kde se zpráva skládá především z použité metody. Metodou může být posílání formuláře nebo zobrazení stránky. Požadavek se pošle na server a ten na tuto zprávu zareaguje odpovědí. Odpověď se skládá ze stavové odpovědi (jak to proběhlo), hlaviček (doplňující informace) a obsah odpovědi (HTML dokument).

Při vývoji webových aplikací může být velice užitečné vědět, jak vypadají konkrétní požadavky a odpovědi. Takovým důvodem může být například ladění aplikace.

3.1.2 **Webová aplikace**

Webové aplikace se v současnosti používají nepřetržitě, jelikož se jedná o software, který není nutné instalovat na zřízení uživatele. Tudíž není omezen platformou konkrétního zařízení a je přístupný z jakéhokoliv místa. Další výhodou může být i možnost aktualizovat aplikaci bez nutnosti uživatelského zásahu (stahováním a instalací) a tím zpříjemnit jeho užití.

V praxi existují statické webové stránky a dynamické. Statické jsou napsané čistě v HTML, případně ještě pomocí CSS a JS, a jejich obsah se nijak nemění. Dynamické stránky jsou vygenerovány právě nějakou webovou aplikací.

Struktura webové aplikace se nejběžněji obecně skládá ze tří vrstev. Jmenovitě jde o prezentační vrstvu, logickou a datovou. Do prezentační vrstvy spadá například prohlížeč, tedy vykreslení a zobrazení dokumentu (nejčastěji html). Logická vrstva je na druhou stranu ta část, která disponuje nástroji pro generování dynamických stránek. Může se jednat především o skriptovací jazyky, programy. A nakonec datová vrstva, do které se zahrnují konkrétní data k zobrazení, a jde například o databázi (Salas-Zárate, a další, 2012).

Vývoj webových aplikací není však omezen pouze jedním jazykem. Způsoby, jakými se dají vytvářet, by se daly zařadit do dvou hlavních kategorií. První kategorií jsou skriptovací jazyky, které běží na straně serveru pomocí Apache. Mezi tyto mimo jiné patří PHP, Python a Perl. Další skupinou nebo kategorií jsou takové jazyky, které jsou prvotně kompilovány do jednoho nebo více *DLL* (sdílené knihovny) souborů. Není tudíž potřeba opakovatelně *parsovat* (překládat) stránky při každém přístupu jako u skriptovacích alternativ. ASP.NET je webovým frameworkem vyvinutý Microsoftem, který obsahuje veliké množství vlastních jazyků, jako je C# a Visual Basic.

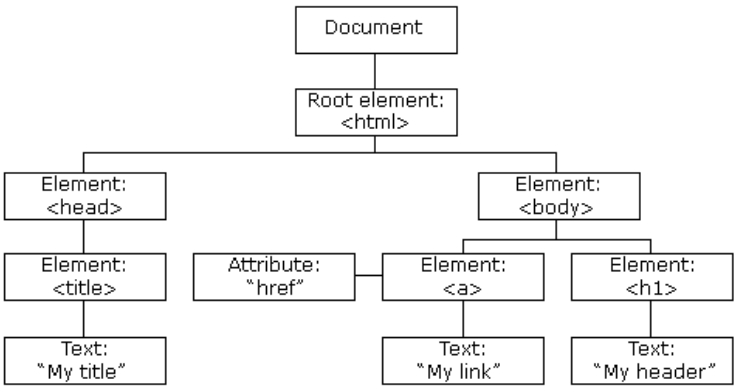
3.1.3 **Technologie webových aplikací**

Jak již bylo zmíněno výše, způsobů a technologií na vytvoření webové aplikace je spousta. Níže jsou tedy popsány technologie určené k vývoji a chodu většiny webových aplikací napsané jazykem PHP. Část těchto technologií je stěžejních, zbytek je spíše doporučením, avšak jsou velice užitečnou součástí vývoje. Mezi taková doporučení patří především *Git* a *Composer*.

HTML

HyperText Markup Language (česky Hypertextový značkovací jazyk) je značkovací jazyk, který se využívá k strukturování a tvorbě webových stránek a online dokumentů. Aktuální verzí tohoto jazyka je HTML5 (Hickson, a další, 2017).

Při načtení každého dokumentu si klient (prohlížeč) vytvoří DOM (Document Object Model, česky Objektový model dokumentu), neboli strom objektů celého dokumentu. Objektový model se využívá zejména s technologiemi jako je JavaScript.



Obrázek 1 – HTML DOM (W3Schools, 2018)

CSS

Cascading Style Sheets (česky Kaskádové styly) je jazyk, kterým se definuje grafický vzhled a chování strukturovaného dokumentu (HTML).

Zápis těchto definicí se dá realizovat v rámci samotného HTML dokumentu nebo externího souboru CSS. U prvního způsobu se kód vloží do hlavičky dokumentu nebo do každého elementu zvlášť jako atribut (vlastnost). Při použití externího souboru se musí vložit odkaz souboru do hlavičky HTML dokumentu (World Wide Web Consortium, 2018).

JavaScript

JavaScript (zkráceně JS) je událostmi orientovaný skriptovací jazyk, který se využívá k manipulaci výše zmíněného DOM.

Skripty se jako u CSS dají psát do samostatného souboru nebo přímo do dokumentu HTML. Při použití externího souboru je potřeba odkázat na soubor opět v hlavičce. Možné je odkazovat i na konci vlastního HTML kódu (konec *body* elementu), což je lepší variantou, jelikož je nutné nejprve načíst DOM a až pak spouštět JS.

Psaní samotných skriptů bývá pracné a ne vždy přehledné. Z toho důvodu byla vytvořena knihovna zvaná JQuery, který zastřešuje funkcionalitu JS do uživatelsky přívětivějšího API.

AJAX

Asynchronous JavaScript and XML (česky *asynchronní JavaScript a XML*) není jazyk, ale kombinace JavaScriptu, DOM a *XMLHttpRequest*. XmlHttpRequest je v prohlížeči vestavěný objekt, který posílá požadavky ohledně dat na server (W3Schools, 2018).

AJAX se tedy využívá k vykonávání požadavků na server bez znovunačtení celé webové stránky. JQuery knihovna manipulaci značně zjednodušuje.

Apache

HTML, CSS, JavaScript jsou technologie, které dokáže zpracovat samotný klient – webový prohlížeč. Na rozdíl od toho jsou PHP skripty zatíženy závislostmi navíc. Nedají se spustit bez určitého nástroje, kterým je právě Apache (Mockus, a další, 2000).

Apache neboli *webový http server*, napsaný v programovacím jazyce C, je software nacházející se na straně serveru a je to hlavní důvod, proč PHP jako takové nefunguje nativně na straně klienta.

Apache podporuje velké množství funkcí, a to například funkce podpory programovacích jazyků na straně serveru nebo různá autentizační schémata. Tudíž se stará o spouštění všech skriptů naspaných v PHP i v dalších jazycích, jako je Python a Perl. Výhodou tohoto webového serveru je především otevřený kód a zaměření na všechny platformy, mezi které patří GNU/Linux, macOS, Microsoft Windows a další (Mockus, a další, 2000).

Mezi další vlastnosti, kterými Apache disponuje, je podpora *htaccess* souborů, umožňující konfiguraci webového serveru: Umožňuje i přepisování URL, manipulaci HTTP hlaviček, podporu XML a nahrávání souborů pomocí FTP protokolu.

Obsahuje navíc i externí modul komprese dat webových stránek, které se posílají protokolem HTTP, open source modul na ochranu webových aplikací před napadením.

PHP

PHP (anglicky *Hypertext Preprocessor*), je zkratkou pro Hypertextový preprocesor a jedná se o skriptovací programovací jazyk, určený k vývoji jednoduchých webů i složitějších webových aplikací. Je navíc nezávislý na platformě a rozdíly v různých operačních systémech se vymezují na několik systémově závislých funkcí a skripty tak lze většinou mezi systémy přenášet bez jakýchkoliv úprav. Nejnovější verzí je 7.2.3, naposledy aktualizována 6. 2. 2019 (PHP, 2018).

Jde o nejvíce používaný jazyk, ačkoliv bývá hodně kritizován ohledně bezpečnosti a konceptů, které velice podporují špagetový kód. To se ale postupně mění s nástupem nové verze PHP 7.

Skladba, syntaxe jazyka, je inspirována několika nejpoužívanějšími programovacími jazyky. Především se jedná o jazyk C, Perl, C, Pascal a Java.

Soubory, napsané tímto jazykem, jsou skripty, které se nacházejí na straně serveru (konkrétně například Apache). Klientovi je pouze http odpovědí zaslán vyhotovený dokument (PHP, 2018).

PHP podporuje mnoho knihoven pro různé účely – například zpracování textu grafiky, práci se soubory, přístup k většině databázových systémů (MariaDB, MySQL, Oracle, MSSQL, ODBC), podporu celé řady internetových protokolů (HTTP, FTP, IMAP, POP3, LDAP) (PHP, 2018).

Na tomto jazyku je založeno mnoho známých webů. Takovými jsou například vícejazyčná online encyklopedie Wikipedia a sociální síť Facebook.

MySQL a MariaDB

Obě jsou relačně databázovými systémy. Databáze je všeobecně definována jako organizovaná sbírka dat, uložená a dostupná v elektronické formě. Pro práci s ní se využívají tzv. DBMS (z anglického *Database Managment System*) neboli *systémy řízení báze dat*, které integrují s databází samotnou a dalšími aplikacemi. Mezi nejčastější příkazy pro práci s databází patří *select, insert, update* a *delete* – získání, uložení, úprava a smazání dat. Na vykonání těchto příkazů je nutné použít SQL jazyk (z anglického *Structured Query Language*) a v něm napsat příkaz. Jedním z nejužívanějších databází je právě MySQL nebo MariaDB.

MySQL je open-source relační databází. Relační databáze znamená, že její obsah je strukturovaný a uložený v tabulkách. Každá tabulka obsahuje určitý konečný počet sloupců – v nich je také konečný počet řádků. Každý záznam je jeden řádek tabulky, který je identifikovaný svým primárním klíčem (MySQL, 2018).

Vývojáři původní verze MySQL se obávali o fungování tohoto systému po odkoupení společností Sun Microsystems (dnes Oracle), která není ani známá poskytováním produktů pod volnou licencí. Právě z těchto obav vznikla větev (*fork*), kopie, MySQL pod názvem MariaDB.

MariaDB se v jistém čase vyčlenila od MySQL. Toto je důvod, proč se všechny struktury, které byly poskládané v MySQL, dají používat i v MariaDB. Díky této unikátní vlastnosti se zajišťuje kompatibilita databází mezi MySQL a MariaDB. Je tudíž možné přesouvat celé databáze bez toho, aby se muselo cokoliv změnit. Z toho vyplývá, že data i definice tabulek jsou plně kompatibilní, stejně tak i klientské protokoly, struktury či API.

Není možné jednoznačně určit, která databáze je z těchto dvou lepší. Je důležité se podívat, pro který typ databáze byla aplikace napsaná. Pokud se stane, že aplikace nevyžaduje pro svůj chod konkrétní typ databáze, tak se otázka opět vrací. Obě mají své výhody a při rozhodování je třeba zvážit, jestli jsou vyžadovány funkce obsažené v MySQL. Pokud ne, zřejmě bude lepší variantou MariaDB, hlavně i kvůli skutečnosti, že bude s jistotou podporovaná ještě dlouhou dobu (Wood, 2019).

MySQL v současné době používají různé organizace a korporáty. Příkladem může být například GitHub, NASA, Tesla, Netflix, Facebook, YouTube a Spotify. MariaDB je používána například Googlem nebo Wikipedií.

Composer

V dnešním světě je běžným zvykem využívání již existujících řešení určitých problémů. Je většinou zbytečné se zabývat problematikou, kterou už někdo vyřešil. Samotné frameworky mají závislosti, které řešil někdo jiný než sám autor frameworku. Composer je tedy nástroj určený především pro PHP knihovny a závislosti. Je s ním možné instalovat, aktualizovat a obecně spravovat již existující projekty a jejich závislosti (Composer, 2016).

Každý takový projekt má někde v sobě soubor s konfigurací, kde jsou informace o různých balíčcích, které jsou nezbytně nutné k běhu aplikace. Můžou to být i závislosti potřebné pouze k vývoji a s během samotné aplikace nemají co dělat.

Git

Na projektu nedělá vždy pouze jeden člověk, ale třeba i více vývojářů. Nastává poté problém například ohledně správy projektu. Člověk udělá nějakou změnu v kódu, opravu nebo přidá novou funkčnost, nahraje na společné úložiště, ke kterému má přístup více lidí. Zbytek nemá ale přehled o tom, co všechno bylo změněno, jaké soubory byly upraveny a které řádky byly smazány či přidány. Bylo by to možné, kdyby to dotyčný řádně sepsal a zdokumentoval. Aby se předešlo podobným problémům, byl vyvinut verzovací systém zvaný git (Git, 2017).

V praxi nástroj umožňuje verzovat aplikaci takovým způsobem, že si pamatuje veškeré změny v kódu. Vývojář má možnost uložit jeho změny s případným komentářem, a tyto změny jsou zvýrazněné pro ostatní.

Důležitou náležitostí je repozitář, nabízený konkrétní službou. Příkladem je Github, Bitbucket nebo Gitlab. Je to taková vývojářská obdoba facebooku. Dá se tam komentovat kód, navrhovat změny a opravy.

3.2 PHP frameworky

Kapitola se zabývá problematikou PHP frameworků a vzorů na ně navazující. Na těchto vzorech – architektonických a návrhových – se zakládají samotné frameworky.

3.2.1 Framework

Framework je softwarová struktura, jejímž účelem je zjednodušení a standardizace vývoje. Standardizace se dá následně chápat jako ucelení konkrétních postupů typických problémů dané oblasti, což nakonec usnadňuje práci jak návrhářům, tak i vývojářům. Často se o frameworku přemýšlí jako o abstrakci poskytující obecnou funkcionalitu, která se dá dále konkretizovat vlastním kódem, což zapříčiní vznik konečného softwaru. Tyto frameworky jsou běžně rozděleny do několika na sobě nezávislých vrstev, které mezi sebou dokáží komunikovat a na každé části se dá pracovat zvlášť.

Využití takových struktur přináší mnoho výhod, ale i nevýhod. Jako nevýhodu se dá považovat konečná velikost dokončeného softwaru, jelikož framework může obsahovat nespočet pomocného kódu, který se ve výsledku ani nevyužije. Důvodem je hlavně jeho obecnost, protože každý software může vyžadovat úplně něco jiného. Typickou nevýhodou je kupříkladu nutnost se naučit s daným frameworkem zacházet a dokonce přijmout koncepty fungování autora frameworku (Edwin, 2014).

Frameworky se jinak nevztahují pouze na webové aplikace, ale i na jiný software. Jedním z nejvíce používaných frameworků je *.NET* („dotnet“), ačkoliv samo o sobě toto označení nevypovídá o konkrétním jazyku. Důvodem je, že tento framework zastřešuje vícero jazyků, jako je C#, Visual Basic .NET, F#, J#. Součástí je pak i samotný ASP.NET, soustředící se na vývoj webových aplikací.

3.2.2 **Návrhové vzory**

Při vývoji webové aplikace, nebo jakéhokoliv softwaru, je velice užitečné (někdy i žádoucí) využívat návrhových vzorů. Takové vzory představují značné usnadnění už při návrhu aplikace. Nejedná se však o zdrojový kód nebo knihovnu, ale pouze o popis řešení určité problematiky. Samotné frameworky, zejména ty pro webové aplikace, se zakládají na návrhových vzorech (Böhmer, 2015).

Existuje několik typů návrhových vzorů, mimo jiné zejména:

- Vytvářející (*creational*) návrhové vzory
- Strukturální (*structural*) návrhové vzory
- Návrhové vzory chování (*behavioral*)

Mezi *vytvářející* patří například *dependency injection* (česky volně *vstřik závislostí*), který se zakládá na předávání závislostí mezi objekty. Principem je tedy to, aby se odebrala třídám zodpovědnost za získávání objektů nutných k jejich činnosti. Je využíván u mnoha PHP frameworků. Vytvářející vzory tudíž řeší problematiku s tvořením objektů.

Strukturálním návrhovým vzorem může být kupříkladu *fasáda*. Ta má za úkol, zjednodušit používání určitého podsystému jednotným rozhraním. Jedná se o vzory, jejichž snahou je zpřehlednit systém a jeho využití.

Vzory *chování* se zajímají o chování systému. Příkladem je *null object* (česky *prázdný objekt*), který eliminuje nutnost využití prázdné hodnoty *null* pro výchozí stav objektu. Toto může značně zpřehlednit a zjednodušit chování daného objektu (Böhmer, 2015).

Existuje mnoho vzorů, které se přestávají používat kvůli existenci jiných a lepší vzorů. Vhodným příkladem je právě zmíněný *dependency injection*. Vlivem tohoto vzoru se přestal používat vzor *singleton*, který měl za úkol vytvářet pouze jedinou instanci napříč celou aplikací, což nebylo úplně ideální.

3.2.3 **Architektonické vzory**

Velice důležitými vzory jsou vzory architektonické. Řeší problematiku na vyšší úrovni než ty návrhové. Hrají důležitou roli v rámci frameworků.

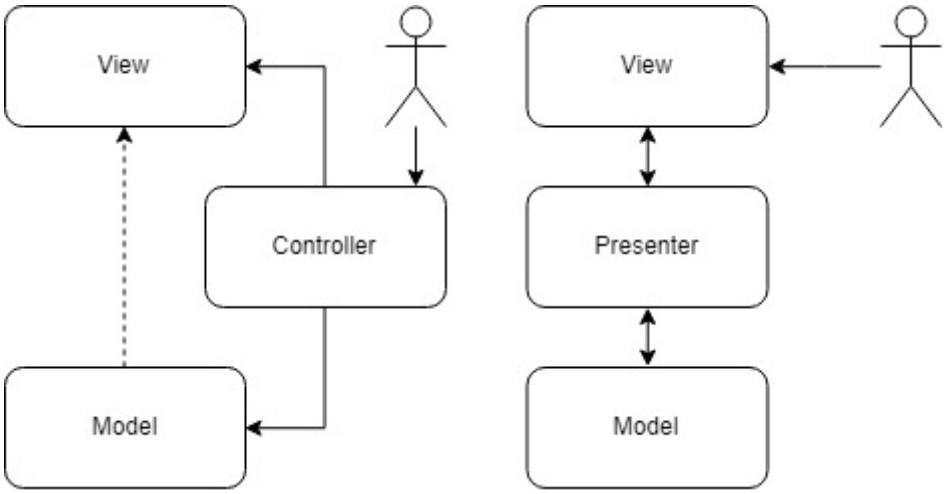
Především se jedná o rozdělení softwaru nebo aplikace na několik navzájem nezávislých částí. Tyto části nebo podsystémy dokáží mezi sebou komunikovat a při vývoji je poté značná výhoda v tom, že na každé části může pracovat jiná skupina vývojářů.

Mezi tyto vzory v rámci PHP patří především:

- MVC architektura
- MVP architektura

MVC architektura se skládá ze tří částí (vrstev) – *Model, View, Controller*. *Modelová* část systému je taková vrstva, která pracuje s daty (bere si je například z databáze) a dokáže je nějakým způsobem transformovat. Obsahuje veškerou aplikační logiku. *View* (česky *pohled*) se stará o vzhled systému (aplikace) – uživatelské vstupy a výstupy. Část *Controller* (česky volně *řadič*) je *prostředník* zprostředkující komunikaci mezi dvěma zbylými vrstvami. Kdykoliv bude vrstva *View* například chtít zobrazit uživateli nějaká data, upozorní na to *Controller*, který následně upozorní *Model*. *Model* a *View* si poté dokáží předat potřebná data (Klíma, 2017).

MVP je novější obdobou architektury MVC s tím rozdílem, že obdobou řadiče je tzv. *Presenter*. Ten, na rozdíl od *Controlleru,* zprostředkovává komunikaci mezi pohledovou a modelovou vrstvou tak, že data jdou přímo přes něj. Pohledová a modelová vrstva neví o existenci toho druhého (Nette, 2018).



Obrázek 2 – Grafické znázornění MVC vs MVP (zdroj: autor)

3.2.4 **Výhody PHP frameworků**

Webové aplikace je možné psát úplně od začátku bez použití jakéhokoliv frameworku. Člověk je tímto však nucen vyvinout více úsilí při psaní kódu. Musí se starat i o další věci, o které by se normálně postaral framework, například bezpečnost. Určité skupiny lidí nebo jedinců si vytvářejí vlastní menší knihovny, které obsahují znovupoužitelný kód, který můžou využít na různých projektech. Z těchto knihoven se může stát právě i framework. Z toho vyplývá, že ačkoliv člověk nechce využívat cizí framework, často si napíše nějakou jeho vlastní obdobu.

Nejčastějším důvodem, proč lidé nepřecházejí na cizí frameworky, je ten, že se nechtějí učit novým věcem. Naučit se nějaký framework zabere určitý čas i úsilí, avšak tato investice přináší hodně výhod.

Jako úplně první důvod použití PHP frameworku je časové hledisko. Čím nižší čas je věnován samotnému vývoji, tím je více času na ladění, testování nebo práce na ostatních zadaných projektech. Tudíž se maximalizuje vývoj. I pro zaměstnavatele toto může být výhodnější.

Frameworky se snaží standardizovat postupy a přinutit vývojáře k lepším návykům psaní aplikací. Toto může maximalizovat spolupráci mezi lidmi v týmu či celé firmě. Aplikace bývá totiž rozdělena do více částí, takže na jednom projektu může pracovat více týmů nezávisle na sobě.

Další výhodou je potenciální dodržení DRY a SRP principů. První princip – *„Don't repeat yourself“* – (česky *„Neopakuj se“*) se zaměřuje právě na snížení redundance neboli opakovatelnosti kódu. Druhý – *„Single responsibility principle“* – (česky *„Princip jedné odpovědnosti“*) nabádá programátora vytvářet objekty s jedinou odpovědností (úkol, schopnost) a *nejedná se o Járu Cimrmana*.

Většina frameworků hodně napomáhá k SEO, což je ve zkratce metodika optimalizace pro vyhledávače (Google, Seznam), která mimo jiné zlepšuje indexaci stránek na daných vyhledávačích. To znamená, že se daná webová stránka objeví ve výsledcích vyhledávání.

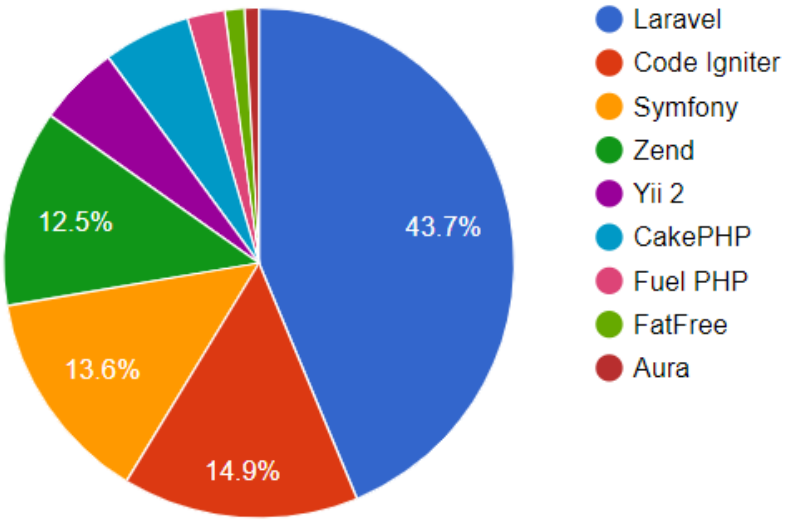
4 Vlastní práce

Oddíl práce pojednává o výběru jednotlivých PHP frameworků, návrhu a tvorbě zkušební webové aplikaci. Nakonec se pomocí předem určených kritérií a vytvořené aplikaci hodnotí samotné vybrané frameworky.

4.1 Výběr PHP frameworků

PHP frameworků je na světě veliké množství a jejich počet neustále roste. Pro tuto práci bylo vybráno pouze pět frameworků, které jsou známější a jsou stále aktivní.

Před výběrem PHP frameworků bylo důležité určit hlavní východiska samotného výběru. Patří mezi nimi především podpora nové verze PHP 7, stejná architektura, aktivita komunity a určitá popularita. K tomu bylo důležité vybrat takové frameworky, které mají dokumentaci psanou alespoň v anglickém jazyce. Velice důležité byla i skutečnost, že samotný kód je psán pouze a jenom v anglickém jazyce.

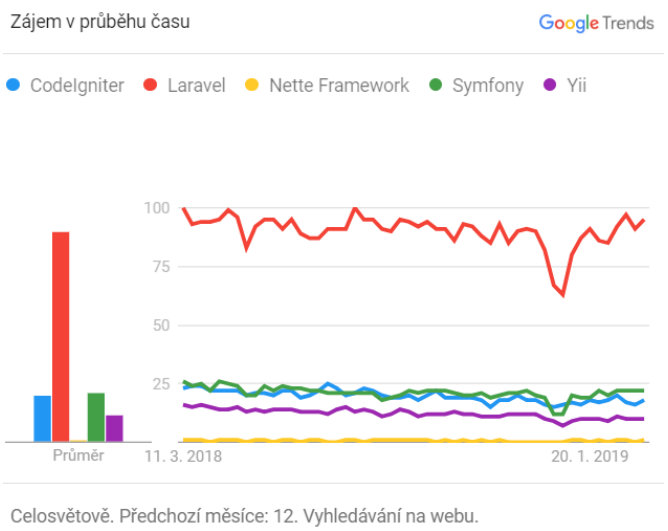


Obrázek 3 – PHP frameworky a jejich využití celosvětově (Reigns, 2019)

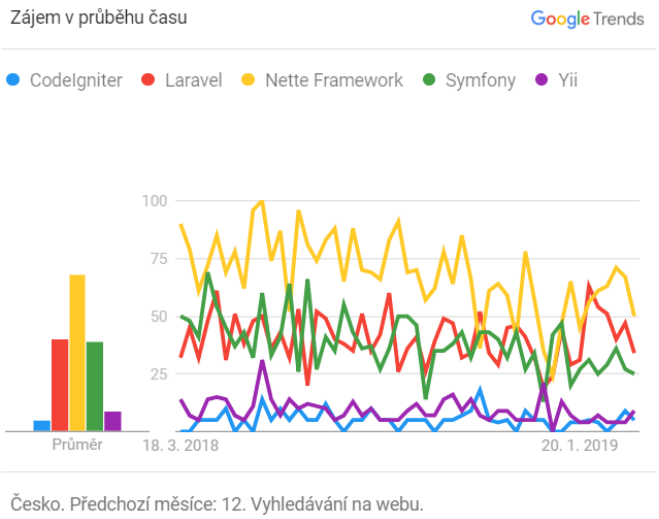
Frameworků je na světě veliké množství a v rámci této práce není možné zpracovat veškeré známé frameworky. Z toho důvodu se na základě několika článků, dostupných na internetu, a odborných zdrojů zvolilo devět celosvětově nejužívanějších PHP frameworků (viz *Obrázek 3*).

V první řadě byly vybrány první tři nejužívanější frameworky (viz *Obrázek 3*), kterými jsou Laravel, Codeigniter a Symfony. Čtvrtým frameworkem byl vybrán Yii, jehož popularita se zakládá na rozšířené konfiguraci a vestavěné podpoře AJAX. Posledním frameworkem byl zvolen Nette framework, který je nejužívanějším českým frameworkem.

Z následujícího grafu (viz Graf 1) je patrné, že Laravel je jedním z mála frameworků, o které je veliký zájem celosvětově. Na druhém grafu (viz Graf 2**Chyba! Nenalezen zdroj odkazů.**) lze vidět, že v České republice je oblíbený spíše framework Nette, což je důvodem jeho výběru.



Graf 1 – Zájem vybraných frameworků celosvětově (zdroj: Google Trends)



Graf 2 – Zájem vybraných frameworků v ČR (zdroj: Google Trends)

Framework	Verze	Web	Licence	PHP
Codeigniter	3.1.10	codeigniter.com	MIT	≥ 5.3.7
Laravel	5.7.27	laravel.com	MIT	≥ 7.1.3
Nette	2.5	nette.org	BSD	≥ 5.6.0
Symfony	4.2.4	symfony.com	MIT	≥ 7.1.3
Yii	2.0.16.1	yiiframework.com	BSD	≥ 5.4.0

Tabulka 1 – Přehled vybraných PHP frameworků

Codeigniter

Jedná se o nejstarší PHP framework, zakládající si na jednoduchosti a rychlosti. Vyžaduje minimální konfiguraci ze strany uživatele a funguje dobře na většině hostujících platformách (Codeigniter, 2018).

Byl vydán v roce 2006. Nejnovější verzí je verze 3.1.10 (16. leden 2019).

Laravel

Nejpopulárnějším PHP frameworkem současnosti je Laravel. Je to framework, který z části vzešel ze Symfony, a proto se některé části hodně podobají (Laravel, 2019).

První vydání bylo roku 2011 a jeho aktuální verzí je 5.7.27 (19. únor 2019).

Nette

Nejužívanější český PHP framework. Disponuje řadou povedených komponent, mezi které patří formuláře a *dependency injection* kontejner (Nette, 2018).

První vydání roku 2008. Aktuální verzí je 2.4 (31. července 2018).

Symfony

Framework, jehož oblíbenost spočívá v opakovatelně použitelných komponentách, které se dají vyžít i mimo framework (Symfony, 2018).

První vydání roku 2005. Aktuální verze 4.2.2 (6. ledna 2019).

Yii

Velice starý framework, který získal na oblíbenosti až od jeho nové verze. Jedna z výhod framework je přímá integrace technologie AJAX (Yii, 2018).

Poprvé vydán roku 2008 a aktuální verzí je 2.0.16 (31. ledna 2019).

4.2 Tvorba aplikace

V každém zvoleném PHP frameworku byla zvlášť vyvinuta zkušební webová aplikace. Oddíl této práce pojednává o různých náležitostech a postupech tvorby vyvinutých aplikací.

4.2.1 Prostředí aplikace

Při vytváření aplikace je velice důležité mít správné prostředí pro vývoj. Přesněji řečeno, je důležité mít k dispozici správné technologie, které webovou aplikaci přivedou k životu. K tomu je možné využít i jiných pomocných nástrojů a technologií.

Základním kamenem každé webové aplikace je Apache (webový server), který rozmlouvá s webovým prohlížečem a stará se o komunikaci s ním. Ještě je důležité samotné PHP, které již dokáže přivést vývojářem napsaný kód k životu. Nakonec je možné využít i nějaké databáze, která obsahuje všechna data, se kterými může webová aplikace pracovat.

V praxi je velice užitečné využívat předpřipravených balíčků, které obsahují výše zmíněné náležitosti každé aplikace. Dále je doporučeno využití *composeru* a *gitu*.

Běhové prostředí

Kombinace PHP s operačním systémem Linux, databázovým systémem a webovým serverem, se uchytila zkratka LAMP – spojení Linux, Apache, MySQL a PHP, Perl nebo Python. Jelikož se jedná především o Linux, byly vytvořeny i další kombinace závislé na ostatních operačních systémech (Dočekal, 2010).

Jedná se tedy o balíčky technologií, které jsou používané jako platforma k vývoji webových aplikací a jejich případný provoz na serveru.

Součástí LAMP je Linux – operační systém, Apache – webový server, MariaDB nebo MySQL – databázový systém a PHP, Perl nebo Python – skriptovací programovací jazyk. MAMP a WAMP jsou obdoby LAMPu, s tím rozdílem, že MAMP je jak pro Windows i macOS, a WAMP pouze pro Windows.

Podobně je na tom XAMPP, ten je však cross-platform, což znamená, že se tento balíček dá využít na kterémkoliv operačním systému (jmenovitě Windows, macOS, Linux). Obsahuje jinak Apache, MariaDB a skriptovací jazyky PHP i Perl.

Zmíněné varianty však nejsou jedinou možností k běhu webové aplikace. Jako alternativou se dá využít *Internet Information Services (ISS)* od Microsfotu.

4.2.2 **Návrh databáze**

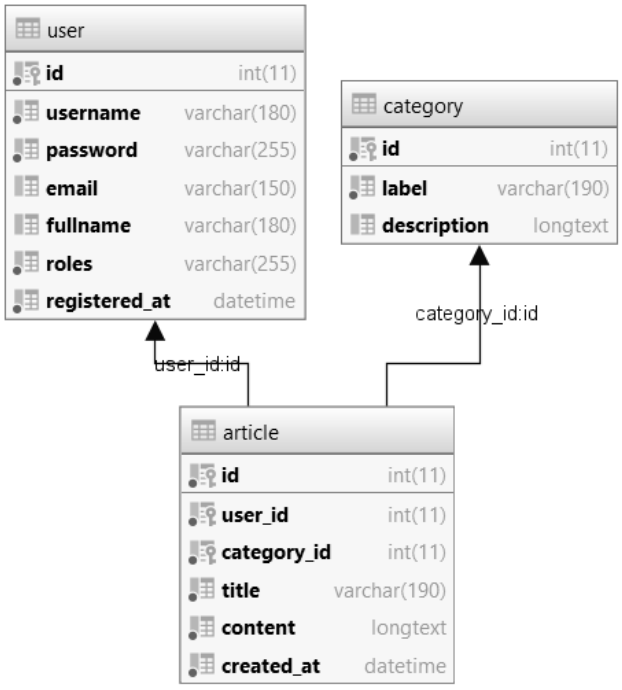
Návrh relační databáze se skládá ze tří tabulek, mezi kterými existuje vzájemný vztah. Vytvoření tabulek je znázorněn ER diagramem (viz Obrázek 4) v SQL jazyce. Zmíněnými tabulkami jsou:

- Článek (*article*)
- Uživatel (*user*)
- Kategorie (*category*)

Tabulka **článků** obsahuje všechny články, které uživatelé vkládají přes dostupné formuláře na daném webu. Na zmíněné články jsou navázány ostatní tabulky pomocí *cizích klíčů* (*user_id, category_id*). Tyto klíče jsou *primárními klíči* (resp. identifikátory) navazujících tabulek uživatelů a kategorií. To znamená, že každý článek má svého uživatele a patří do nějaké kategorie. Dále obsahuje titulek (*title*), obsah (*content*) a datum vytvoření (*created_at*).

Tabulka **uživatelů** obsahuje více informací. Informace se skládají z celého jména (*fullname*), uživatelského jména (*username*), emailu (*email*), hesla (*password*), role (*roles*) a datum registrace (*registered_at*).

Tabulka **kategorií** se skládá pouze z názvu kategorie (*label*) a případného popisu (*description*).



Obrázek 4 – ERD (entitní diagram vztahů) vytvořené databáze (zdroj: autor)

4.2.3 **Návrh a struktura aplikace**

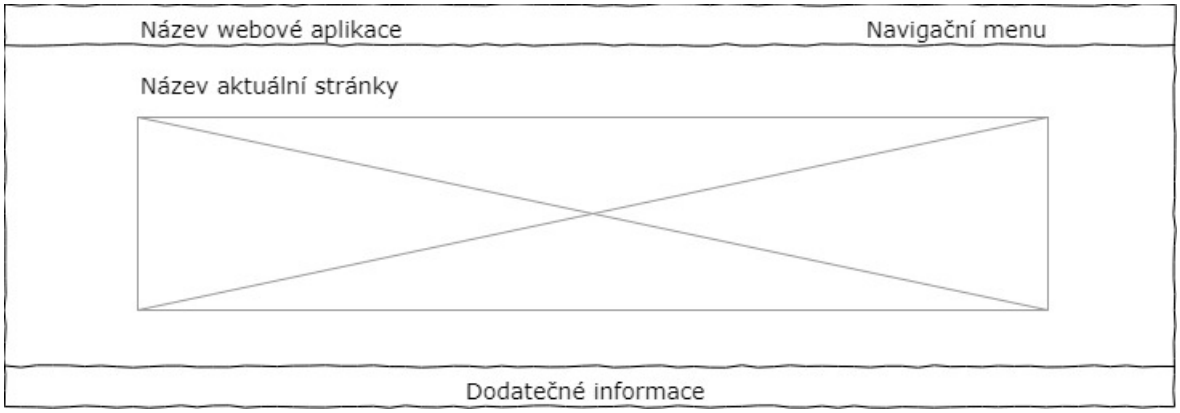
Jako prvním krokem bylo potřeba vymyslet jednotnou šablonu, kterou budou všechny aplikace využívat. Byl tudíž napsán HTML dokument, který byl nastylován pouze za pomocí CSS frameworku Bootstrap4. Pro jednoduchost byla použita pouze CDN verze tohoto frameworku, což znamená, že se vkládá do HTML dokumentu pouze odkazem a není třeba žádného externího souboru.

V levé horní části, takzvané *hlavičce*, se nachází název webové aplikace. V pravé se vyskytuje hlavní navigace, která mimo jiné poskytuje navigaci pro celou aplikaci. Bylo zařízeno, aby se odkazy navigace měnily dle toho, jestli je uživatel přihlášen, či nikoliv.

Samotným obsahem každé stránky je poté například výpis článků, kategorií, autorů a různých formulářů na úpravu informací v databázi. Obsah je doprovázen i názvem aktuální stránky vyskytující se nad obsahem stránky.

Dolní část stránky neboli *patička*, je doplněna o další informace ohledně aktuálního roku a autorovi projektu.

Samotný vývoj aplikace se hodně lišil dle užívaného frameworku. Důvodem je hlavně užití konceptů, kterými se autor a spoluautoři konkrétního frameworku řídí. Struktura každého projektu se dala do jisté míry měnit, ale je byla spíše snaha se o toto nepokoušet.



Obrázek 5 – Drátěný model (wireframe) zkušební webové aplikace (zdroj: autor)

Princip aplikace

Primárním cílem aplikace byla možnost *vytvářet*, *vypisovat*, *měnit* a *mazat* obsah webu. Tomuto principu se ve zkratce říká CRUD aplikace, kde jednotlivá písmena reprezentují již zmíněné akce (v angličtině) – *create*, *read*, *update* a *delete*. Tyto akce budou důležitým aspektem testování rychlosti a bezpečnosti každého vybraného frameworku. Mezi data, která se dají v rámci aplikace upravovat, jsou články, kategorie a uživatelé.

Výpis dat

První stránka, kterou návštěvník uvidí po vstupu na webovou stránku, je domovská stránka. Na této stránce se zobrazují všechny články a data k nim náležející (uživatel, kategorie). Je záměrem, aby byly vypsány všechny články najednou, kvůli testování rychlosti. V praxi by bylo vhodné omezit takový výpis pouze na pár vybraných položek, například pomocí stránkování, a případně ještě oříznout výpis dlouhých textů.

Každý článek vypsaný na domovské stránce obsahuje odkaz, který vede na výpis konkrétního článku. Na této stránce jsou zobrazeny stejné informace, které lze vidět i na domovské stránce. V ideálním případě by články na domovské stránce neobsahovaly všechny možné informace.

Manipulace dat

Jakákoliv data se dají vytvářet a upravovat pomocí dostupných formulářů. Vytvořit se dá článek, kategorie i uživatel, kdežto změna informací je přístupná pouze u článků a kategorií. V ideálním případě by se dal upravovat i uživatel, ale pro jednoduchost aplikace bylo toto zanedbáno.

Formuláře mají při editaci všechny prvky předvyplněné existujícími daty, které se dají upravit a následně uložit zpět do databáze. Pro tuto akci byl vytvořen jednodušší výpis článků s odkazy na úpravu jednotlivých článků a jejich mazání.

ACL

Aby bylo možné otestovat vícero možností bezpečnostních útoků, bylo potřeba implementovat ACL. Access Control List (česky *seznam pro řízení přístupu*) je způsob povolování přístupu oprávněných uživatelů k určitému obsahu. Součástí ACL je *registrace*, *autentizace* a *autorizace* jednotlivých uživatelů (Microsoft, 2018).

Registrace má na starosti vytvoření nového uživatele, který bude mít možnost pouze vytvářet a upravovat (včetně smazání) vlastní články. Registrační formulář vyžaduje pouze heslo a uživatelské jméno návštěvníka (nepřihlášený uživatel). Při ukládání do databáze je nezbytně nutné, a to kvůli bezpečnosti, aby se heslo neukládalo jako čistý text, ale jako *otisk* (anglicky *hash*) tohoto hesla. Účelem takového otisku je zajistit, aby heslo znal pouze uživatel. V databázi je tedy vidět pouze náhodná posloupnost znaků, která se vytvoří pomocí určitého algoritmu (v tomto případě *bcrypt*).

Když se bude chtít uživatel přihlásit do systému, musí do přihlašovacího formuláře zadat uživatelské jméno a heslo. Aplikace se dle zadaných údajů podívá do databáze, zdali takový uživatel existuje. Uživatel nemusí znát otisk svého hesla, aplikace totiž porovnává zadané heslo s otiskem v databázi pomocí funkce na porovnávání za použití stejného algoritmu. Pakliže existuje shoda, aplikace uživatele přihlásí. Tomuto procesu se říká autentizace.

Ve většině případů je žádoucí, aby každý uživatel směl vytvářet, měnit a mazat jen ta data, která mu náleží, a v určitých případech i zobrazovat. Ověření, zdali určitý uživatel smí vykonat určitou akci, se říká autorizace. Určité části aplikace byly tudíž zabezpečeny a znepřístupněny návštěvníkům a neautorizovaným uživatelům.

Obyčejným uživatelům bylo zpřístupněno pouze vytváření, editace a smazání článků. Jediný administrátor je schopen manipulovat s jakýmikoliv daty na této webové aplikaci. Tudíž smí vytvářet i upravovat kategorie.

4.3 Zhodnocení

Hodnocení frameworků je založeno na následujících kritériích a získaných dat jednotlivých aplikací každého frameworku. Ne všechna zvolená kritéria jsou stěžejní pro vývoj a chod aplikace. Dodávají však na uživatelské přívětivosti z vývojářského hlediska. Část kritérií je velice důležitá pro optimální chod webové aplikace na serveru, druhá část je zaměřena spíše na vývojovou problematiku.

Aplikace byly ve prospěch testování rychlosti a bezpečnosti nahrány na server poskytnutý komerční službou. Na serveru běží operační systém Linux a data jsou uložena na *SSD* discích, zajišťujících vysokou rychlost.

Kritéria pro hodnocení byla vybrána na základě poznatků při tvorbě zkušebních aplikací, které ovlivňují vývoj a zajišťují optimální chod aplikace.

4.3.1 Kritéria hodnocení

Kritérií bylo vybráno celkem sedm:

- Rychlost
- Bezpečnost
- Dokumentace
- Databáze
- Šablonovací systém
- Vývojová konzole
- Ladící nástroje

Každé kritérium se skládá z několika *částí*, která mají určitou váhu důležitosti. Na určení důležitosti se podílely skutečnosti přejaté při tvorbě aplikací a na základě rozhovorů s odborníky v praxi.

Rychlost

Rychlost je jednou ze stěžejních vlastností aplikace. Trvá-li provedení jakéhokoliv požadavku (načtení stránky, přihlašování) dlouho, uživatel může do dokončení požadavku stránku opustit už jen ze samotné frustrace. Takové skutečnosti je třeba zabránit.

Měřit se bude rychlost vykonání požadavků v každé aplikaci zvlášť a k samotnému měření bude použit software JMeter (open source) od firmy Apache, napsaný v programovacím jazyce Java. Umožňuje simulovat požadavky na server.

Každý článek se skládá ze všech tří tabulek databáze (články, uživatelé a kategorie).

Bezpečnost

Bezpečnost je nejdůležitějším kritériem webové aplikace. Ačkoliv je bezpečnostních rizik mnoho, tato práce se bude zabývat pouze třemi a to zejména:

- SQL injection
- XSS – *cross-site scripting*
- CSRF – *cross-site request forgery*

SQL injection je založený na manipulaci jakéhokoliv SQL příkazu k získání citlivých údajů nebo jejich změně. Kupříkladu je možné pomocí formuláře poslat hodnotu *„; DROP TABLE article; --“*. Při vkládání do databáze je možné přerušit probíhající SQL dotaz a smazat celou tabulku s články. Na ochranu je možné použít *escapování* (ošetření proměnných) při vytváření SQL dotazu. Účinnější je však nevkládat proměnné přímo do SQL dotazu (při jeho vytváření), ale použít parametrizované (předpřipravené) dotazy. Tímto se vytvoří zvlášť SQL dotaz se zástupnými znaky a pole hodnot. Databáze si následně sama poradí s jejich správným vložením.

XSS je typ útoku, kterým útočník dokáže svým škodlivým skriptem změnit vzhled stránky, obcházet bezpečnostní prvky nebo získat citlivá data. Hojně se využívá při *phishing* útocích, což uživatele přiměje otevřít věrohodnou stránku, kde útočníkův skript může získat uživatelem zadané údaje. Značnou ochranou je *escapování* hodnot. Toto se musí v PHP dělat manuálně pomocí funkce *htmlspecialchars*, kde se nesmí zapomenout i na důležitý parametr *ENT_QUOTES*, který funkci zpřísní o další znaky.

Pomocí *CSRF* je možné například smazat článek, aniž by byl útočník přihlášen. Protiopatřením je použití autorizačního *tokenu*, kterým se kontroluje, zdali jde požadavek z ověřeného zdroje.

Dokumentace

Dokumentace je nedílnou součástí frameworků, která popisuje jejich použití, koncepty a další aspekty s nimi spjaté. V rámci každé dokumentace je k dispozici i API (*Application Programming Interface*), což je popis rozhraní zdrojového kódu jako takového.

Není to však jediný způsob učení. Je zvykem se nejdříve dívat na různá videa a psané návody na internetu. Tudíž existuje možnost, že se do dokumentace vývojář ani nepodívá. Na internetu se vyskytuje nespočet tutoriálů a screencastů vybraného frameworku.

Databáze

Každá aplikace s uživatelskými vstupy a výstupy využívá datové uložiště. Takovým úložištěm jsou například relační nebo souborové databázové systémy. Vybrané PHP frameworky práci s databází podporují. Z toho důvodu se hodnotí způsoby práce s databází

Frameworky často umožňují psaní vlastních SQL dotazů, ačkoliv to není ideálním případem. Z větší části se využívá vrstvy na vyšší úrovni – abstrakce. Abstrakcí se myslí psaní SQL dotazů pomocí různých funkcí.

Určité frameworky využívají tzv. ORM (*object-relational mapping*) – objektově relační mapování. Mapování kompletně odděluje vývojáře od databáze a pracuje s ní pomocí objektů – entit. Entita je v tomto smyslu jakýkoliv objekt, který reprezentuje určitou tabulku databáze.

Šablonovací systém

Šablonovací systém (anglicky *template engine*) je způsob zápisu pohledové vrstvy za použití speciálně pojmenovaných souborů, ve kterých se využívá speciální syntaxe společně s obyčejným HTML zápisem. Přehlednost kódu a oddělení pohledové vrstvy od aplikační logiky se výrazně zvyšuje. Použitím šablonovacího systému je možné se zbavit zvýšené pozornosti na bezpečnostní rizika, která se můžou v aplikaci vyskytnout. V praxi je běžné, že tyto systémy nejsou závislé pouze na jednom frameworku a dají se implementovat i do ostatních projektů (i bez frameworku). PHP je ve své podstatě také šablonovacím systémem, jelikož je možné psát PHP skripty do samotného HTML dokumentu.

Vývojová konzole

Vývojová konzole je PHP soubor, přes který se spouští různé příkazy ovlivňující aplikaci a její vývoj. Může se jednat například o příkazy generování kódu, manipulaci databáze, cache, rout a mnoho dalšího. Frameworky často umožňují psaní vlastních příkazů.

Ladící nástroje

Během vývoje je důležité ladění (anglicky *debugging*) aplikace a případné logování. Ladící nástroje ve vývojářském režimu napomáhají ve vyhledávání chyb – ladění aplikace. Nástroj poskytuje i logování chyb. Je totiž možné, že se v aplikaci objeví neošetřená chyba, na kterou uživatel narazil v produkčním režimu. Logování pak spočívá v zápisu takové chyby do souboru, který je k dispozici pouze vývojáři.

Některé logovací nástroje poskytují zobrazení informací aplikace pomocí plovoucího panelu nebo profileru (samostatná stránka s informacemi). Jedná se například o informace o požadavcích na server, rychlosti, přihlášeném uživateli a provedenými SQL dotazy.

4.3.2 **Hodnocení**

Vhledem ke skutečnosti, že část kritérií je kvalitativního charakteru a druhá kvantitativního, byly zvoleny dva způsoby hodnocení.

Kritéria, která jsou v rámci této práce spíše *kvantitativního* charakteru, byla ohodnocena tak, že získané četnosti byly normalizovány podílem s maximální četností. Čitatelem je tedy číslo (četnost), které se má normalizovat. Ve jmenovateli se pak nachází maximální číslo (četnost) z konkrétní části kritéria. Příkladem může být množina pěti čísel: 5, 25, 12, 1, 50. Z těchto čísel je patrné, že maximem je 50, tudíž bude toto číslo jmenovatelem pro každý výpočet. Výjimkou je rychlost, kde jsou hodnoty normalizovány podílem minimální rychlosti a získané hodnoty.

Kvalitativní kritéria byla hodnocena rovnou čísly mezi 0 až 1. Nejvyšší možné skóre (1) vyjadřuje maximální spokojenost s určitou částí kritéria každého frameworku. Na druhé straně je minimální hodnota, která vyjadřuje naprostou nespokojenost nebo neexistenci dané části. Další možné hodnoty byly popsány u každého kritéria zvlášť.

Testování všech zkušebních webových aplikací proběhlo v rámci stejného prostředí. Výsledky každého kritéria byla ještě jednou normalizována podílem maximálního skóre, aby byla udržena konzistence hodnot v rámci grafu (viz Graf 3).

Rychlost

Každá vytvořená webová aplikace byla naměřena nástrojem JMeter. Byly měřeny následující požadavky na server:

- Přihlášení uživatele
- Načtení všech článků (500+ záznamů)
- Vytvoření článku
- Načtení jednoho článku
- Úprava článku
- Smazání článku
- Odhlášení uživatele

V nástroji JMeter byl vytvořen testovací plán pro každou aplikaci zvlášť. V rámci tohoto plánu bylo potřeba nastavit *Thread group* (vláknová skupina), která definuje počet testovaných uživatelů, dobu vytvoření všech uživatelů (najednou) a celkový počet opakování pro každého uživatele. Příklad v příloze (viz Obrázek 20).

Byly nastaveny následující hodnoty:

- Number of Threads (Users) – 10
- Ramp-up Period (in seconds) – 10
- Loop Count – 10

Počet vláken (*Number of Threads*) znamená, že každý plán provede 10 uživatelů najednou. Čas *Ramp-up* zajistí postupné vytváření uživatelů každou 1 vteřinu (10 uživatelů po dobu 10 vteřin). Počet opakování (*Loop Count*) je nastaven na 10, což znamená, že každá část se bude opakovat 100krát. Pro kontrolu byly testovací plány u každého frameworku provedeny třikrát.

JMeter má možnost výsledky testování analyzovat a ze získaných hodnot vypočítat mediány. Mediány byly použity k výpočtu skalárního součinu každého testu. U jednotlivých frameworků byl vybrán test, kde výsledný skalární součin byl středem zbylých dvou (viz Tabulka 3). Z vybraných testů byly spočítány výsledky (viz Tabulka 4).

Bezpečnost

Bezpečnost byla hodnocena v rámci tří vybraných částí:

- SQL injection
- XSS
- CSRF

SQL injection byl vybrán jako nejdůležitější, jelikož dokáže například mazat tabulky databáze. CSRF bylo hodnoceno jako druhé nejdůležitější, protože umožňuje kupříkladu posílání formulářů bez oprávnění. XSS získalo nakonec nejnižší váhu.

V rámci této práce se jedná především o kvalitativní kritérium, kde se hodnotilo dle určené stupnice:

- 1 – automaticky chrání proti útoku
- 0,5 – potřeba aktivovat ochranu manuálně
- 0 – nedisponuje ochranou proti útoku

Ochranou proti SQL injection má na 0,5 pouze Codeigniter a Yii. Zbytek tuto ochranu provádí automaticky. Stejně tomu tak je i u XSS, kde pouze dva již zmíněné frameworky touto

ochranou disponují, ale je potřeba nezapomenout ji manuálně dopsat. U CSRF ochrany to bylo trochu jiné. Automatickou ochranou nedisponují Laravel a Nette. Bylo potřeba ji manuálně zapnout.

Výsledky jsou v příloze (viz Tabulka 5).

Dokumentace

Dokumentace byla rozdělena pouze na dvě části a to:

- Oficiální dokumentace
- Ostatní materiály (psané návody, videa)

Jako jediné z kritérií byla dokumentace rozdělena na části, které jsou odlišného charakteru. Z toho důvodu budou hodnoceny odlišně. První část byla ohodnocena o něco vyšší váhou, protože oficiální dokumentace zahrnuje i API, které je velice užitečné.

Oficiální dokumentace byla hodnocena následovně:

- 1 – spokojenost
- 0,5 – spokojenost až na výjimky
- 0 – nespokojenost

Část *oficiální dokumentace* byla pouze u dvou hodnocena nejvyšším hodnocením. Nette a Symfony disponují velice přehlednou dokumentaci, u které je možné jít i do hloubky. Na druhé straně byl zbytek ohodnocen číslem 0,5. Codeigniter má dobrou dokumentaci, až na skutečnost, že se opírá pouze o základy, nejde vůbec do hloubky. Laravel disponuje povedenou dokumentací. Neobsahuje pouze základy a v určitých ohledech jde více do hloubky, ale struktura a navigace (dokumentací) není nepřívětivější. Nakonec Yii má povedenou dokumentaci, až na výjimku ohledně ilustračních kódů. Ne vždy uvádějí, kde se daný kód nachází.

Ostatní materiály byly hodnoceny pomocí četností výsledků vyhledávání na Googlu a YouTube (viz Tabulka 7).

Výsledky jsou k dispozici v příloze (viz Tabulka 6).

Databáze

Práce s databází byla rozdělena na tyto části:

- Databázové migrace
- Jednoduché abstrakce
- ORM

Všechny části byly hodnoceny tímto způsobem:

- 1 – existence
- 0 – neexistence

Databázové migrace jsou užitečné, ale jsou nejméně potřebné, proto dostávají nejmenší váhu. Naopak *jednoduchá abstrakce* je nejdůležitější součástí práce s databází. Abstrakce vytváří vyšší vrstvu skládání SQL dotazů, tudíž není třeba psát vlastní SQL dotazy a starat se o bezpečnostní rizika s nimi spjatá. Nakonec velice užitečnou součástí vývoje je *ORM*, které je o něco důležitější než samotné migrace.

Migracemi nedisponují pouze Nette a Codeigniter, avšak je celkem snadné je implementovat pomocí třetích stran. Jednoduchou abstrakci zvládají všechny, tudíž dostávají za 1. Je ale nutno podotknout, že ne všechny jsou velice intuitivní. Každopádně se jedná o velice subjektivní názor a z toho důvodu tato skutečnost nebyla zohledněna. Nakonec se hodnotila existence ORM. Pouze dva frameworky, Laravel a Symfony, toto implementují.

Výsledky jsou k dispozici v příloze (viz Tabulka 8).

Šablonovací systém

Šablonovací systémy byly hodnoceny pouze u tří frameworků. Testované verze Codeigniteru a Yii vlastním šablonovacím systémem nedisponují v základní verzi. U Codeigniteru je potřeba tento systém zapínat zvlášť a jedná se spíše o velice jednoduchý kompilátor. U všech frameworků se dá využít šablonovacích systémů třetích stran.

Hodnotily se tři části kritéria a každé byla přiřazena váha důležitosti. Nejdůležitějšími jsou pomocné funkce, jelikož nejvíce ovlivňují vývoj. Funkce pro vytváření formulářů jsou užitečné, ale vždy existuje alternativa psaní obyčejného HTML kódu, proto nejsou důležitější než pomocné funkce. Filtry byly ohodnoceny nejnižší vahou a to z toho důvodu, že se jedná o nejméně důležitou součást šablonovacích systémů. Umožňují pouze jednoduchou transformaci, která se dá řešit i v aplikační logice.

Části po zvážení:

- Pomocné funkce
- Filtry
- Formuláře

U každého frameworku byly sečteny podporované pomocné funkce, filtry a funkce pro vytvoření a manipulaci formulářů. Mezi tyto funkce patří různé funkce, jako je například

foreach (iterace prvku v poli nebo výčtu), makra (spíše u Nette) a direktiva (v rámci Laravel). Filtry slouží k transformaci samotných proměnných (zkrácení textu na deset znaků atd.). Nakonec se sčítaly funkce, kterými je možné vytvářet formulář a jeho prvky.

Výsledky jsou k dispozici jako tabulky v příloze (viz Tabulka 9 a Tabulka 10).

Vývojová konzole

Kritérium vývojové konzole bylo hodnoceno obdobně, jako šablonovací systémy. Hodnoceny byly pouze tři frameworky, jelikož zbylé dva (Nette a Codeigniter) takovým nástrojem nedisponují, ačkoliv je možné použít konzoli ze Symfony.

Byly vybrány části, kterým byla přiřazena určitá váha:

- Množství základních příkazů
- Generování kódu
- Databáze a migrace
- Cache
- Routy
- Vlastní příkazy

Nejdůležitější částí bylo generování kódu, protože nejvíce ovlivňuje a zjednodušuje vlastní vývoj aplikace. Databáze a migrace byly zvoleny jako druhé nejdůležitější a to z toho důvodu, že dokáží spravovat a udržovat databázi bez potřeby manuálního zacházení. Velice dobrou vlastností konzolí vybraných frameworků je možnost psaní vlastních příkazů. Proto jsou třetí nejdůležitější částí kritéria. Jako méně důležité bylo vybráno množství základních příkazů, jelikož je možné psát vlastní příkazy a větší množství nemusí ihned znamenat lepší konzoli. Nakonec nejméně důležité byly zvoleny části cache a routy. Manipulace s cache je užitečná, ačkoliv není až tolik využívaná. Frameworky se o cache starají často samy (mazání a vytváření). Routy nebo routování, což je obousměrný překlad mezi řadičem a URL, nemají moc využití v rámci konzolí. Jedná se především pouze o příkazy pro výpis existujících rout, proto byla tato část ohodnocena nejmenší vahou.

Výsledky jsou k dispozici v příloze (viz Tabulka 11 a Tabulka 12).

Ladící nástroje

Ladící nástroje byly rozděleny na tři částí:

- Logování
- Panel a profiler
- Vizualizace chyb a výjimek

Vizualizace chyb a výjimek je důležitou součástí vývoje, jelikož se snaží vývojáři přiblížit, kde nastala chyba. Panel a profiler jsou jinou obdobou vizualizace a to především všech informací ohledně webové aplikace. Panel je spíše plovoucí element na stránce, který obsahuje nejdůležitější informace. Profiler je na druhou stranu obohacen o mnoho dalších informací. Obě části byly ohodnoceny stejně, jelikož jsou stejně důležité. Logování na druhou stranu je důležité spíše až na produkčním serveru, což znamená, že byla aplikace dokončena a je nyní k dispozici online.

Jelikož se jedná o kritérium kvantitativního charakteru, byly zvoleny stupně hodnocení:

- 1 – naprosto vyhovuje
- 0,5 – vyhovuje až na pár výjimek
- 0 – nevyhovuje, potřeba zásahu třetí strany

Logování bylo hodnoceno u všech PHP frameworků stejně – 1, jelikož všechny logovacím systémem disponují již od začátku projektu.

U části ohledně plovoucího *panelu a profileru* byly jako jediný Codeigniter a Laravel ohodnoceny číslem 0, jelikož žádným takovým nástrojem nedisponují. Existují však nástroje třetích stran, které se do daných frameworků dají implementovat. Nette a Yii byly hodnoceny 0,50. Nette kvůli neexistenci profileru. Yii kvůli skutečnosti, kde bylo zapotřebí tyto nástroje stáhnout. Nebylo ohodnoceno 0, jelikož tento nástroj je přímo od tvůrců frameworku, kteří pouze tento nástroj oddělili, aby byl základní projekt co nejmenší.

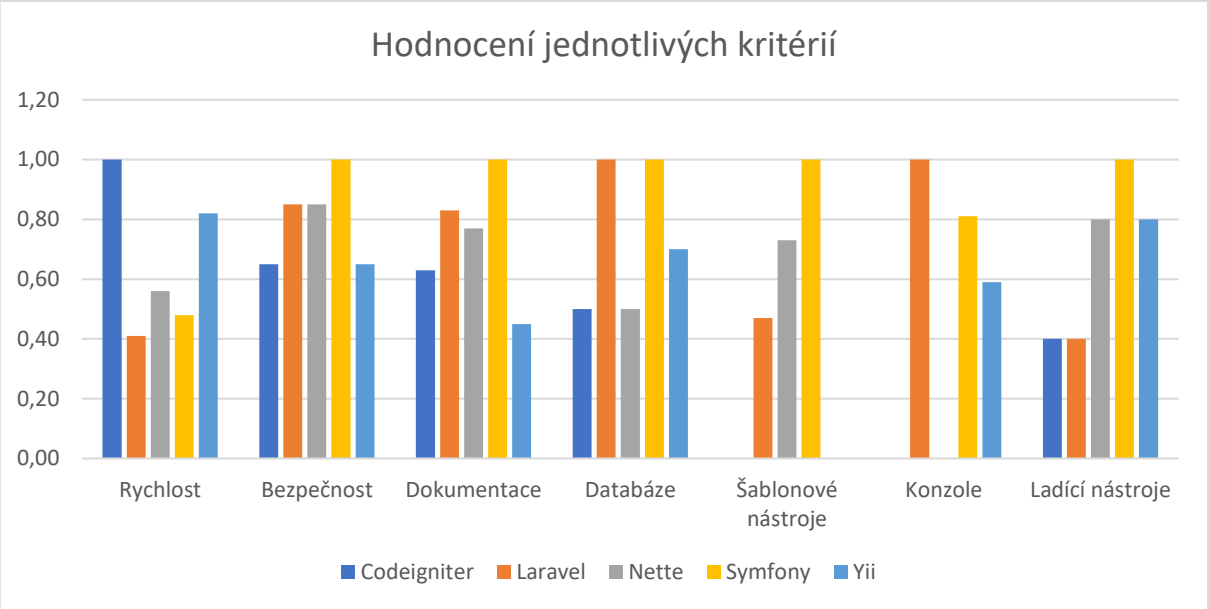
Pro část *vizualizace chyb a výjimek* byly ve vytvořených aplikací nasimulovány navzájem podobné chyby, které měl za úkol framework zachytit. V příloze jsou k dispozici screenshoty (viz Obrázek 12, Obrázek 13, Obrázek 14, Obrázek 15, Obrázek 16). Všechny toto zvládají dobře, tudíž se hodnotila především přehlednost. Nette, Symfony a Yii byly ohodnoceny číslem 1, jelikož jejich přehlednost byla nejlepší. Zbytek byl ohodnocen číslem 0,5. Codeigniter je v tomto ohledu velice omezený (viz Obrázek 12) a Laravel může být chvílemi velice nepřehledný (viz Obrázek 13).

Výsledky jsou k dispozici v příloze (viz Tabulka 13).

5 Výsledky a diskuse

Kapitola se zabývá popisem a možnostmi využití jednotlivých PHP frameworků. Čerpáno bylo z výsledků hodnocení a odborných zdrojů.

Následující graf znázorňuje normované výsledky PHP frameworků z každého kritéria (viz Tabulka 2).



Graf 3 – Graf hodnocení jednotlivých kritérií (zdroj: autor)

Rychlostně si nejlépe vedl Codeigniter. U bezpečnosti se nejlépe projevil Symfony, hlavně díky automatické ochraně, o kterou se uživatel nemusí tolik starat. Oficiální dokumentace byla u všech dobře stravitelná, až na výjimku Codeigniteru. U tohoto frameworku se zdá být kladen důraz pouze pro začátečníky, jelikož nepopisuje problematiku daného frameworku do hloubky. Co se ostatních materiálů týče, nejlépe na tom byl Laravel se Symfony, především kvůli rozsáhlé komunitě a velikého zájmu v rámci trendů. Laravel obzvláště kvůli síti Laracast. Práce s databází byla nejlepší u Symfony a Laravelu, především díky použití ORM a migrací. Mezi šablonovacími systémy si nejlépe vedl Symfony z důvodu velkého množství funkcí, filtrů a pomocných funkcí pro vytváření formulářů. Vývojové konzole, kterými disponují tři vybrané frameworky, má značně širší využití Laravel, jelikož je zaměřen jak na generování kódu, tak i na práci s databází a dalšími náležitostmi. Symfony se zaměřil spíše na práci s databází a Yii spíše na generování kódu. Nejužitečnější ladící nástroje má Symfony, po kterém těsně následují nástroje pro Nette a Yii.

Frameworky se lišily především databázovou vrstvou a ladícími nástroji.

5.1 Možnosti využití PHP frameworků

Účelem hodnocení nebylo určit ultimátum, ale pouze zjistit, pro které účely by se mohl určitý PHP framework hodit více než ty ostatní. Není možné, aby jeden framework stál nad všemi ostatními.

V této části se popisují vybrané PHP frameworky a zjištěné informace (v rámci hodnocení). Na základě těchto informací byly frameworky rozděleny do dvou skupin využití. První skupinou jsou webové projekty malého rozsahu (např. blogy, malé e-shopy). Druhou skupinou jsou weby většího rozsahu, kde je kladen důraz na bezpečnost a velké množství dat – velká data (např. podnikové a korporační weby, REST aplikace).

5.1.1 Codeigniter

Codeigniter je ideálním frameworkem pro rapidní vývoj a to zejména díky jednoduché instalaci a téměř žádnému nastavování. Skutečnost, že se jedná o velice malý framework, co se velikosti týče, dále napomáhá v jeho rozšíření o další funkčnosti.

Speciální na tomto frameworku je, že nevyžaduje po vývojáři využívání MVC architektury. Je tedy možné upravit projekt tak, aby tuto architekturu nevyužíval. Na druhou stranu umožňuje využít této architektury na vyšší úrovni. Tím je architektura HMVC (hierarchický MVC), umožňující udržovat modulární seskupení všech tří vrstev v pod-složkovém formátu.

Nehodí se na podnikové weby a aplikace s velkými daty, kde záleží především na bezpečnosti. Je i nutno podotknout, že samotný systém Codeigniteru není úplně objektově orientovaný a pořád se opírá o starou verzi PHP 4. Ačkoliv nová verze frameworku podporuje nejnovější PHP 7, pořád kvůli tomu nepřebírá všechny koncepty PHP 7. To může vést k problémům v budoucnosti, hlavně co se týče bezpečnosti. Nevyužívá navíc jmenné prostory.

Codeigniter je vhodný spíše na menší aplikace, jako je blog, či jiné publikační systémy malého rozsahu. Je každopádně možné rozšířit framework o šablonovací systém a další funkcionality, které bezpečnost podpoří ve větším měřítku.

5.1.2 Laravel

Laravel je v současné době nejužívanějším PHP frameworkem. O tuto skutečnost se opírá především vysoká úroveň abstrakce systému, která umožňuje psát mnohem přehlednější kód. Zjednodušení a zpříjemnění práce obzvláště omezujícím způsobem je výhradou frameworku,

která rozděluje vývojáře do dvou skupin, protože to není pro všechny ideální. Omezení určité volnosti vede k problému při vytváření velice komplexních webových aplikací.

Eloquent (ORM) je jistým příkladem abstrakce, kde je důležité pouze definování entit a veškerá funkčnost, co se vyhledávání a ukládání dat týče, je k dispozici bez většího úsilí. To však znamená, že aplikace provádí hodně procesů navíc, čímž se velice zpomaluje její rychlost. Například u zmíněné databázové vrstvy je hlavní nevýhodou vytváření mnoha dotazů, které můžou být až zbytečně složité.

Vývojová konzole *artisan* je převážně zaměřena na generování jednotlivých částí aplikace, jako jsou modely, entity a řadiče (*controller*).

Laravel disponuje i velice rychlým a velikostně lehkým šablonovacím systémem Blade (viz Obrázek 8), která vývojáři dodává úplnou kontrolu nad pohledovou vrstvou. Má navíc rozšíření na většinu věcí, která řeší určitou problematiku.

Učebních materiálů má tento framework spoustu. Mezi tyto materiály patří hlavně screencasty pod názvem Laracast, kde vysvětlují náležitosti frameworku velice dobře, zřetelně a přehledně ve formě krátkých videí. Komunita je velice aktivní, hlavně z toho důvodu, že se jedná o nejvíce užívaný framework.

Framework na rozdíl od ostatních podporuje další požadavkové metody, jako jsou například *PUT PATCH* a *DELETE*, které se starají o posílání dat. Vzhledem k této skutečnosti je možné doporučit na tvorbu REST (Representational State Transfer) aplikací. REST je architektura podobná CRUD, zakládající se na jednoduchém způsobu přístupu k datům pouze pomocí HTTP metod a odpovědí ve formátu JSON nebo XML.

5.1.3 **Nette**

Nette Framework je každou aktualizací lepší. Nedávno byla vydána úplně nová verze 3, která přináší kompletní podporu a využití nejnovější PHP verze a jejími koncepty.

Framework má jako jeden z mála velice dobře řešený *dependency injection*. Šablonovací systém Latte a ladící nástroj Tracy jsou obzvláště povedené, hlavně i díky tomu, že nejsou závislé na samotném Nette. Dají se tedy využít u jiných frameworků.

Díky dobrému systému komponent frameworku je využití především pro aplikace zaměřené na manipulaci dat. Světově je tento framework méně známý, kvůli uzavřené komunitě. Na rozdíl od ostatních frameworků postrádá pár kritických komponent, kde je nutno využít například Symfony. Takovou komponentou může být například Doctrine (ORM),

migrace nebo systém na události v rámci kódu. Základní databázová vrstvou je však jednoduchá abstrakce, která je pro začátečníky velice přívětivá a jednoduchá na naučení.

Framework je vhodné využívat na e-shopy a větší podnikatelské weby. Především díky dobrému šablonovacímu systému a vysoké bezpečnosti.

5.1.4 **Symfony**

Symfony je nejstabilnějším PHP frameworkem současnosti. Založený je především na užitečné sadě komponent, které se dají využívat zvlášť, i v rámci jiných frameworků. Například Laravel je z určité části závislý na Symfony.

Framework využívá databázovou vrstvu zvanou Doctrine. Jedná se ORM, které není úzce spjato se Symfony, ale dá se využít u jiných frameworků, jako je Codeigniter. Tato vrstva se podobá ORM u Laravelu, jen není až tak magická.

Vývojová konzole Symfony se zabývá spíše manipulací s Doctrine a migracemi databáze. Disponuje především vytvářením databázového schématu z existujících entit nebo vytvořením testovacích či základních dat.

Framework je založen na vysoké modularitě, tudíž je v praxi velice běžné kombinovat jeho komponenty nebo i samotný framework s jinými aplikacemi.

Díky vysoké bezpečnosti je dobrým kandidátem na podnikatelské weby.

5.1.5 **Yii**

Yii je framework založený na spoustě konfigurací, kterými se dá aplikace různě vylepšovat a zrychlovat její výkon. Konfigurace je velice rozsáhlá, ale i to je velikou nevýhodou, jelikož se vše řídí prostřednictvím vícerozměrných polí. To znamená, že je kód a samotné nastavení aplikace velice nepřehledné.

Vývojová Konzole Yii se zaměřuje na generování kódu a práci s databází. Generovat dokáže všechny potřebné prvky architektury MVC, a je schopen jediným příkazem vytvářet celé CRUD struktury. Existuje i webová verze této konzole, zvaná Gii.

Výhodou tohoto frameworku je vestavěná podpora AJAX, která je především užitečná při vytváření e-shopů a obdobných, kde UX (uživatelský zážitek) hraje důležitou roli při jejich využití.

6 Závěr

V první části práce byl charakterizován vývoj webových aplikací a problematika s nimi spojená. Zejména otázky principu webových aplikací a jejich technologií a problematika vývoje aplikací s a bez použití frameworku.

Následující oddíl se zabýval vlastní prací, kde byly na základě stanovených hledisek vybírány konkrétní PHP frameworky. K hodnocení byly vybrány následující frameworky: *Codeigniter*, *Laravel*, *Nette*, *Symfony* a *Yii*. S využitím těchto frameworků byly vytvořeny zkušební webové aplikace. Získané informace z průběhu vývoje aplikací byly využity k hodnocení frameworků. K hodnocení bylo využito následujících kritérií: *rychlost, bezpečnost, dokumentace, databáze, šablonovací systémy, vývojová konzole, ladící nástroje*. Výsledky zhodnocení poukazují na skutečnost, že každý framework má své koncepty a náležitosti, které se velice liší od ostatních. Příkladem je přístup k řešení databázové vrstvy nebo přítomnost vývojové konzole.

Závěrečná část práce se zabývala možností využití vybraných PHP frameworků. Studiem odborných zdrojů a získaných informací z hodnocení jednotlivých kritérií byly v rámci práce stanoveny dvě skupiny využití. Do první skupiny patří *Codeigniter* a *Yii*, které se hodí především pro webové projekty malého rozsahu. Hlavními výhodami těchto frameworků je rychlost vývoje aplikace a malé nároky na hardware (i software) běhového prostředí díky optimální architektuře frameworku. Do druhé skupiny byly zařazeny *Laravel*, *Nette* a *Symfony*, které lze doporučit na aplikace se zaměřením na citlivá a objemově velká data. A to především z důvodu jejich dobrého zabezpečení a robustní databázové vrstvy.

7 Seznam použitých zdrojů

Alani, Mohammed M. 2014. TCP/IP Model. [Online] 2014. [Citace: 5. 1 2019.] https://link.springer.com/chapter/10.1007/978-3-319-05152-9_3.

Bean, Martin. *Laravel 5 Essentials.* místo neznámé : Packt.

Böhmer, Marian. 2015. *Návrhové vzory v PHP.* Brno : Computer Press, 2015. ISBN 978-80-251-3338-5.

Buenosvinos, Carlos, Akbary, Keyvan a Soronellas, Christian. 2017. *Domain-Driven Design in PHP*. místo neznámé : Packt Publishing, 2017. ISBN 9781787284944.

Castagnetto, Jesus. 2001. *Programujeme PHP profesionálně.* Praha : Computer Press, 2001. ISBN 80-7226-310-2.

Codeigniter. 2018. CodeIgniter User Guide. *CodeIgniter*. [Online] 2018. [Citace: 29. 11 2018.] https://codeigniter.com/user_guide/.

Composer. 2016. Introduction. *getcomposer.org.* [Online] 2016. [Citace: 13. 12 2018.] https://getcomposer.org/doc/00-intro.md.

Dočekal, Michal. 2010. Správa linuxového serveru: Konfigurace LAMP: PHP. *Linuxexpress.cz.* [Online] 2010. [Citace: 11. 12 2018.] https://www.linuxexpres.cz/praxe/sprava-linuxoveho-serveru-konfigurace-lamp-php.

Edwin, Njeru Mwendi. 2014. Software Frameworks, Architectural and Design Patterns. *Journal of Software Engineering and Applications.* 2014.

Git. 2017. About. *Git-scm.com.* [Online] 2017. [Citace: 12. 12 2018.] https://git-scm.com/about.

Grudl, David. 2006. PHP: černá magie optimalizace. *phpFashion.com.* [Online] 30. 9 2006. [Citace: 1. 5 2018.] https://phpfashion.com/php-cerna-magie-optimalizace.

Hickson, Ian a Hyatt, David. 2017. HTML5 specification - w3.org. *World Wide Web Consortium.* [Online] 2017. [Citace: 18. 7 2018.] http://www.w3.org/TR/html5/.

Hussain, Altaf. 2016. *Learning PHP 7 High Performance.* Birmingham : Packt Publishing, 2016. ISBN 9781785881633.

Internet Society. 2003. Brief History of the Internet. *Internetsociety.org.* [Online] 2003. [Citace: 12. 3 2019.] http://www.internetsociety.org/internet/what-internet/history-internet/brief-history-internet.

Jazayeri, Mehdi. 2007. Some Trends in Web Application Development. [Online] 2007. [Citace: 22. 11 2018.] http://ieeexplore.ieee.org/document/4221621.

Khaliluzzaman, Md. a Chowdhury, Iftekher Islam. 2016. Pre and post controller based MVC architecture for web application. [Online] 2016. [Citace: 2. 3 2019.] http://ieeexplore.ieee.org/document/7760064.

Klíma, Tom. 2017. Architektura MVC. *Jakpsatphp.cz.* [Online] 2017. [Citace: 15. 12 2018.] http://jakpsatphp.cz/MVC/.

Laravel. 2019. Release Notes. *Laravel.com.* [Online] 2019. [Citace: 2. 1 2019.] https://laravel.com/docs/5.8/releases.

Mcheick, Hamid a Qi, Yan. 2011. Dependency of components in MVC distributed architecture. [Online] 2011. [Citace: 1. 9 2018.] http://yadda.icm.edu.pl/yadda/element/bwmeta1.element.ieee-000006030542.

Microsoft. 2018. Access Control Lists. *Microsoft.com.* [Online] 2018. [Citace: 10. 12 2018.] https://docs.microsoft.com/cs-cz/windows/desktop/SecAuthZ/access-control-lists.

Mockus, Audris, Fielding, Roy a Herbsleb, James D. 2000. A case study of open source software development: the Apache server. [Online] 2000. [Citace: 13. 3 2019.] http://web.eecs.utk.edu/~audris/papers/apache.pdf.

MySQL. 2018. MySQL Internals Manual. *Dev.mysql.com.* [Online] 2018. [Citace: 10. 8 2018.] http://dev.mysql.com/doc/internals/en/index.html.

—. 2018. Supported Platforms: MySQL Database. *MySQL.* [Online] 2018. [Citace: 1. 12 2018.] http://www.mysql.com/support/supportedplatforms/database.html.

Nette. 2018. Documentation. *Nette.org.* [Online] 2018. [Citace: 1. 10 2018.] https://doc.nette.org/cs/2.4/.

—. 2018. MVC aplikace & presentery. *Nette.org.* [Online] 2018. [Citace: 15. 12 2018.] https://doc.nette.org/cs/2.4/presenters.

Padilla, Armando a Hawkins, DUPTim. 2011. *Pro PHP Application Performance: Tuning PHP Web Projects for Maximum Performance.* New York : Apress, 2011. ISBN 9781430228998.

PHP. 2018. Database Security. *Php.net.* [Online] 2018. http://php.net/manual/en/security.database.php.

—. 2018. History of PHP and related projects. *Php.net.* [Online] 2018. [Citace: 13. 10 2018.] http://www.php.net/history.

—. 2018. PHP Manual. *Php.net.* [Online] 2018. [Citace: 28. 4 2018.] http://php.net/manual/en/.

—. 2018. SQL injection. *Php.net.* [Online] 2018. [Citace: 12. 12 2018.] http://php.net/manual/en/security.database.sql-injection.php.

—. 2018. What can PHP do? *Php.net.* [Online] 2018. [Citace: 12. 12 2018.] http://php.net/manual/en/intro-whatcando.php.

—. 2018. What is PHP? *Php.net.* [Online] 2018. [Citace: 13. 12 2018.] http://php.net/manual/en/intro-whatis.php.

Pitt, Chris. 2012. *Pro PHP MVC.* New York : Apress, 2012. ISBN 1430241640.

Porebski, Bartosz, Przystalski, Karol a Nowak, Leszek. 2011. Building PHP Applications with Symfony, CakePHP, and Zend Framework. [Online] 2011. [Citace: 2. 3 2019.] https://amazon.com/building-applications-symfony-cakephp-framework/dp/0470887346.

Powell, Thomas A. 2008. Ajax: The Complete Reference. [Online] 2008. [Citace: 11. 11 2018.] https://amazon.com/ajax-complete-reference-thomas-powell/dp/007149216x.

Reigns, Stephanie. 2019. 11 Best PHP Frameworks for Modern Web Developers in 2019. *Coderseye.com.* [Online] 2019. [Citace: 18. 1 2019.] https://coderseye.com/best-php-frameworks-for-web-developers.

Salas-Zárate, Marí del Pilar, Alor-Hernández, Giner a Rodríguez-González, Alejandro. 2012. *Developing Lift-based Web Applications Using Best Practices.* 2012. stránky 214-223. Sv. 3.

Samra, Jone. 2015. Comparing Performance of Plain PHP and Four of Its Popular Frameworks. [Online] 2015. [Citace: 22. 9 2018.] http://diva-portal.org/smash/record.jsf?pid=diva2:846121.

Shklar, Leon a Rosen, Rich. 2009. *Web Application Architecture: Principles, Protocols and Practices.* Hoboken : Wiley, 2009. ISBN 9780470518601.

Symfony. 2018. Symfony Documentation. *Symfony.com.* [Online] 2018. [Citace: 16. 12 2018.] https://symfony.com/doc/current/index.html#gsc.tab=0.

The Apache Software Foundation. 2018. Apache HTTP Server. *Apache.org.* [Online] 2018. [Citace: 15. 10 2018.] https://projects.apache.org/project.html?httpd-http_server.

W3Schools . 2018. JavaScript HTML DOM. *W3Schools.com.* [Online] 2018. [Citace: 12. 11 2018.] https://www.w3schools.com/js/js_htmldom.asp.

W3Schools. 2018. AJAX Introduction. *W3Schools.com.* [Online] 2018. [Citace: 7. 9 2018.] https://www.w3schools.com/xml/ajax_intro.asp.

—. 2018. JavaScript HTML DOM. *W3Schools.com.* [Online] 2018. [Citace: 9. 7 2018.] https://www.w3schools.com/js/js_htmldom.asp].

Wood, William. 2019. *Migrating to MariaDB: Toward an Open Source Database Solution.* New York : Apress, 2019.

World Wide Web Consortium. 2018. CSS - Cascading Style Sheets home page. *W3.org.* [Online] 2018. [Citace: 10. 1 2019.] http://www.w3.org/Style/CSS/#browsers.

—. 2018. HTML 4.01 Specification. *W3.org.* [Online] 2018. [Citace: 2. 2 2019.] http://www.w3.org/TR/html401/struct/links.html.

—. 2017. HTML 5.2 W3C Recommendation. *w3.org.* [Online] 2017. [Citace: 12. 12 2018.] https://www.w3.org/TR/html/introduction.html#introduction-history.

—. 2014. The web standards model - HTML CSS and JavaScript. *W3.org.* [Online] 2014. [Citace: 11. 11 2018.] https://www.w3.org/wiki/The_web_standards_model_-_HTML_CSS_and_JavaScript.

Yii. 2018. Intruduction: About Yii. *yiiframework.com.* [Online] 2018. [Citace: 2. 1 2019.] https://www.yiiframework.com/doc/guide/2.0/en/intro-yii.

8 Přílohy

	Codeigniter	Laravel	Nette	Symfony	Yii2
Rychlost	1,00	0,41	0,56	0,48	0,82
Bezpečnost	0,65	0,85	0,85	1,00	0,65
Dokumentace	0,63	0,83	0,77	1,00	0,45
Databáze	0,50	1,00	0,50	1,00	0,70
Šablonovací systém	0,00	0,47	0,73	1,00	0,00
Vývojová konzole	0,00	1,00	0,00	0,81	0,59
Ladící nástroje	0,40	0,40	0,80	1,00	0,80

Tabulka 2 – Souhrn normovaných výsledků všech kritérií

	Přihlášení uživatele	**Načtení všech**	**Vytvoření článku**	**Načtení článku**	**Úprava článku**	**Smazání článku**	**Odhlášení uživatele**	***skalární součin***
váha	*0,125*	*0,15*	*0,15*	*0,15*	*0,15*	*0,15*	*0,125*	
Codeigniter	627	257	478	106	394	486	487	*397,40*
	492	**249**	**458**	**97**	**479**	**423**	**598**	***392,15***
	482	289	608	134	333	350	382	*365,10*
Laravel	482	1273	2214	459	2236	219	2040	*1275,40*
	1492	**1247**	**2203**	**298**	**2145**	**165**	**1231**	***1249,08***
	1296	1547	1509	453	1925	185	1500	*1192,35*
Nette	884	987	573	785	987	597	847	*805,73*
	735	**754**	**625**	**415**	**689**	**597**	**783**	***651,75***
	611	801	578	623	697	647	612	*654,78*
Symfony	1281	564	856	415	798	749	1281	*827,55*
	970	**527**	**793**	**297**	**856**	**697**	**1311**	***760,63***
	831	570	799	393	759	803	1005	*728,10*
Yii	998	629	880	744	310	176	788	*634,10*
	697	**494**	**667**	**472**	**279**	**124**	**887**	***503,40***
	612	479	632	575	267	114	867	*494,93*

Tabulka 3 – Výsledky z JMeter testu

	Přihlášení uživatele	Načtení všech	Vytvoření článku	Načtení článku	Úprava článku	Smazání článku	Odhlášení uživatele	*výsledek*	*norm. výsledek*
váha	*0,125*	*0,15*	*0,15*	*0,15*	*0,15*	*0,15*	*0,125*		
Codeigniter	1,00	1,00	1,00	1,00	0,58	0,29	1,00	*0,83*	*1,00*
Laravel	0,33	0,20	0,21	0,33	0,13	0,75	0,49	*0,34*	*0,41*
Nette	0,67	0,33	0,73	0,23	0,40	0,21	0,76	*0,47*	*0,56*
Symfony	0,51	0,47	0,58	0,33	0,33	0,18	0,46	*0,40*	*0,48*
Yii	0,71	0,50	0,69	0,21	1,00	1,00	0,67	*0,68*	*0,82*

Tabulka 4 – Výsledky rychlosti

	SQL Injection	XSS	CSRF	*výsledek*	*norm. výsledek*
váha	*0,50*	*0,50*	*1,00*	*0,65*	*0,65*
Codeigniter	1,00	1,00	0,50	*0,85*	*0,85*
Laravel	1,00	1,00	0,50	*0,85*	*0,85*
Nette	1,00	1,00	1,00	*1,00*	*1,00*
Symfony	0,50	0,50	1,00	*0,65*	*0,65*
Yii	0,50	0,50	1,00	*0,65*	*0,65*

Tabulka 5 – Výsledky bezpečnosti

	Oficiální dokumentace	Ostatní materiály	*výsledek*	*norm. výsledek*
váha	*0,50*	*0,40*		
Codeigniter	0,50	0,30	*0,42*	*0,63*
Laravel	0,50	0,55	*0,55*	*0,83*
Nette	0,50	1,00	*0,51*	*0,77*
Symfony	1,00	0,04	*0,66*	*1,00*
Yii	1,00	0,55	*0,30*	*0,45*

Tabulka 6 – Výsledky dokumentace

	Ostatní materiály
Codeigniter	15500000
Laravel	28000000
Nette	1200000
Symfony	15300000
Yii	4660000

Tabulka 7 – Výsledky dokumentace (pouze ostatní materiály)

	Databázové migrace	Jednoduchá Abstrakce	ORM	*výsledek*	*norm. výsledek*
váha	*0,20*	*0,50*	*0,30*		
Codeigniter	0,00	1,00	0,00	*0,50*	*0,50*
Laravel	1,00	1,00	1,00	*1,00*	*1,00*
Nette	0,00	1,00	0,00	*0,50*	*0,50*
Symfony	1,00	1,00	1,00	*1,00*	*1,00*
Yii	1,00	1,00	0,00	*0,70*	*0,70*

Tabulka 8 – Výsledky databáze

	Pomocné funkce	Filtry	Formuláře	*výsledek*	*norm. výsledek*
váha	*0,50*	*0,10*	*0,40*		
Codeigniter	0,00	0,00	0,00	*0,00*	*0,00*
Laravel	0,78	0,00	0,20	*0,47*	*0,47*
Nette	0,87	0,58	0,60	*0,73*	*0,73*
Symfony	1,00	1,00	1,00	*1,00*	*1,00*
Yii	0,00	0,00	0,00	*0,00*	*0,00*

Tabulka 9 – Výsledky šablonovacího systému (po normalizaci)

	Pomocné funkce	Filtry	Formuláře
Codeigniter	0	0	0
Laravel	52	0	2
Nette	58	28	6
Symfony	67	48	10
Yii	0	0	0

Tabulka 10 – Výsledky šablonovacího systému (četnost)

	Základní příkazy	Generování kódu	Databáze a migrace	Cache	Routy	Vlastní příkazy	*výsledek*	*norm. výsledek*
váha	*0,10*	*0,3*	*0,25*	*0,10*	*0,05*	*0,20*		
Codeigniter	0,00	0,00	0,00	0,00	0,00	0,00	*0,00*	*0,00*
Laravel	1,00	1,00	0,32	1,00	1,00	1,00	*0,83*	*1,00*
Nette	0,00	0,00	0,00	0,00	0,00	0,00	*0,00*	*0,00*
Symfony	0,84	0,07	1,00	1,00	0,33	1,00	*0,67*	*0,81*
Yii	0,45	0,39	0,35	0,40	0,00	1,00	*0,49*	*0,59*

Tabulka 11 – Výsledky vývojové konzole (po normalizaci)

	Základní příkazy	Generování kódu	Databáze a migrace	Cache	Routy	Vlastní příkazy
Codeigniter	0	0	0	0	0	0
Laravel	73	28	11	10	3	1
Nette	0	0	0	0	0	0
Symfony	61	2	34	10	1	1
Yii	33	11	12	4	0	1

Tabulka 12 – Výsledky vývojové konzole (četnosti)

	Logování	Vizualizace chyb a výjimek	Panel a profiler	*výsledek*	*norm. výsledek*
váha	*0,20*	*0,40*	*0,40*		
Codeigniter	1,00	0,00	0,50	*0,40*	*0,40*
Laravel	1,00	0,00	0,50	*0,40*	*0,40*
Nette	1,00	0,50	1,00	*0,80*	*0,80*
Symfony	1,00	1,00	1,00	*1,00*	*1,00*
Yii	1,00	0,50	1,00	*0,80*	*0,80*

Tabulka 13 – Výsledky ladících nástrojů

```
CREATE TABLE `article` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `user_id` int(11) NOT NULL,
  `category_id` int(11) NOT NULL,
  `title` varchar(190) COLLATE utf8mb4_unicode_ci NOT NULL,
  `content` longtext COLLATE utf8mb4_unicode_ci NOT NULL,
  `created_at` datetime NOT NULL,
  PRIMARY KEY (`id`),
  KEY `IDX_23A0E66A76ED395` (`user_id`),
  KEY `IDX_23A0E6612469DE2` (`category_id`),
  CONSTRAINT `FK_23A0E6612469DE2` FOREIGN KEY (`category_id`) REFERENCES
`category` (`id`),
  CONSTRAINT `FK_23A0E66A76ED395` FOREIGN KEY (`user_id`) REFERENCES `user`
(`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;


CREATE TABLE `category` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `label` varchar(190) COLLATE utf8mb4_unicode_ci NOT NULL,
  `description` longtext COLLATE utf8mb4_unicode_ci,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;


CREATE TABLE `user` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `username` varchar(180) COLLATE utf8mb4_unicode_ci NOT NULL,
  `password` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `email` varchar(150) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `fullname` varchar(180) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `roles` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `registered_at` datetime NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `UNIQ_8D93D649F85E0677` (`username`),
  UNIQUE KEY `UNIQ_8D93D649E7927C74` (`email`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

Obrázek 6 – SQL zápis pro vytvoření tabulek

```
<div class="container">
    <h1 class="py-4">Create article</h1>
    <?= form_open('articles/create', ['method' => 'POST', 'class' => 'mt-4 mb-4']) ?>
    <div class="form-group">
        <label for="exampleInputTitle">Title</label>
        <input name="title" type="text" class="form-control" id="exampleInputTitle" placehol-
der="Some title">
    </div>
    <div class="form-group">
        <label for="exampleSelect">Category</label>
        <select name="category_id" class="form-control" id="exampleSelect">
            <?php foreach ($categories as $category): ?>
                <option value="<?= html_escape($category['id']) ?>"><?= html_escape($cate-
gory['label']) ?></option>
            <?php endforeach; ?>
        </select>
    </div>
    <div class="form-group">
        <label for="exampleInputContent">Content</label>
        <textarea name="content" class="form-control" id="exampleInputContent" placehol-
der="Some text"
                  rows="3"></textarea>
    </div>
    <button type="submit" class="btn btn-primary btn-sm">Submit</button>
    <?= form_close() ?>
</div>
```

Obrázek 7 – Formulář Codeigniter (pohledová vrstva)

```
@extends('layouts.app')
@section('title', 'Create article |')
@section('content')
    <div class="container">
        <h1 class="py-4">Create article</h1>
        <form class="mt-4 mb-4" method="POST" action="{{ route('app_article_store') }}">
            @csrf
            <div class="form-group">
                <label for="exampleInputTitle">Title</label>
                <input name="title" type="text" class="form-control" id="exampleInputTitle"
placeholder="Some title">
            </div>
            <div class="form-group">
                <label for="exampleSelect">Category</label>

                <select name="category_id" class="form-control" id="exampleSelect">
                    @foreach($categories as $category)
                        <option value="{{ $category->id }}">{{ $category->label}}</option>
                    @endforeach
                </select>
            </div>
            <div class="form-group">
                <label for="exampleInputContent">Content</label>
                <textarea name="content" class="form-control" id="exampleInputContent" place-
holder="Some text" rows="3"></textarea>
            </div>
            <button type="submit" class="btn btn-primary btn-sm">Submit</button>
        </form>
    </div>
@endsection
```

Obrázek 8 – Formulář Laravel pomocí systému Blade (pohledová vrstva)

```
{block content}

<h1 n:block="title" class="py-4">Create article</h1>
{form addForm class=>'mt-4 mb-4'}
    <div class="form-group">
        {label title}
        {input title, class=>'form-control', placeholder=>'Some title'}
    </div>
    <div class="form-group">
        {label category_id}
        {input category_id, class=>'form-control'}
    </div>
    <div class="form-group">
        {label content}
        {input content, class=>'form-control', placeholder=>'Some text', rows=>'3'}
    </div>
    {input submit, class=>'btn btn-primary btn-sm'}
{/form}
```

Obrázek 9 – Formulář Nette pomocí systému Latte (pohledová vrstva)

```
{% extends 'base.html.twig' %}

{% block body %}
    <h1 class="py-4">{% block title %}Create article{% endblock %}</h1>
    {{ form_start(form, {attr: {class: 'mt-4 mb-4'} }) }}
    {{ form_errors(form) }}
    <div class="form-group">
        {{ form_label(form.title) }}
        {{ form_widget(form.title, {attr: {
            class: 'form-control',
            placeholder: 'Some title'
        } }) }}
        {{ form_errors(form.title) }}
    </div>
    <div class="form-group">
        {{ form_label(form.category) }}
        {{ form_widget(form.category, {attr: {
            class: 'form-control'
        } }) }}
        {{ form_errors(form.category) }}
    </div>
    <div class="form-group">
        {{ form_label(form.content) }}
        {{ form_widget(form.content, {attr: {
            class: 'form-control',
            placeholder: 'Some content',
            rows: '3'
        } }) }}
        {{ form_errors(form.content) }}
    </div>
    {{ form_widget(form.submit, {attr: {
        class: 'btn btn-primary btn-sm'
    } }) }}
    {{ form_end(form) }}
{% endblock %}
```

Obrázek 10 – Formulář Symfony pomocí systému Twig (pohledová vrstva)

```php
<?php

use yii\helpers\Html;
use yii\widgets\ActiveForm;

/* @var $this yii\web\View */
/* @var $article app\models\Article */
/* @var $categories array */
/* @var $form yii\widgets\ActiveForm */

?>


<?php $form = ActiveForm::begin([
    'options' => ['class' => 'mt-4 mb-4']
]); ?>


<?= $form->field($article, 'title')->textInput(['maxlength' => true, 'placeholder' => 'Some
title']) ?>
<?= $form->field($article, 'category_id')->dropDownList($categories) ?>
<?= $form->field($article, 'content')->textarea(['rows' => 3, 'placeholder' => 'Some
text']) ?>
<?= Html::submitButton('Submit', ['class' => 'btn btn-primary btn-sm']) ?>

<?= $form->field($article, 'created_at')->hiddenInput(['value' => (new DateTimeImmuta-
ble('now'))->format('Y-m-d H:i:s')]) ?>
<?= $form->field($article, 'user_id')->hiddenInput(['value' => Yii::$app->user->id]) ?>

<?php ActiveForm::end(); ?>
```

Obrázek 11 – Formulář Yii (pohledová vrstva)

An uncaught Exception was encountered

Type: TypeError

Message: Argument 1 passed to Article::show() must be an instance of Article, string given, called in C:\GitHub\bakalarka\codeigniter\system\core\CodeIgniter.php on line 532

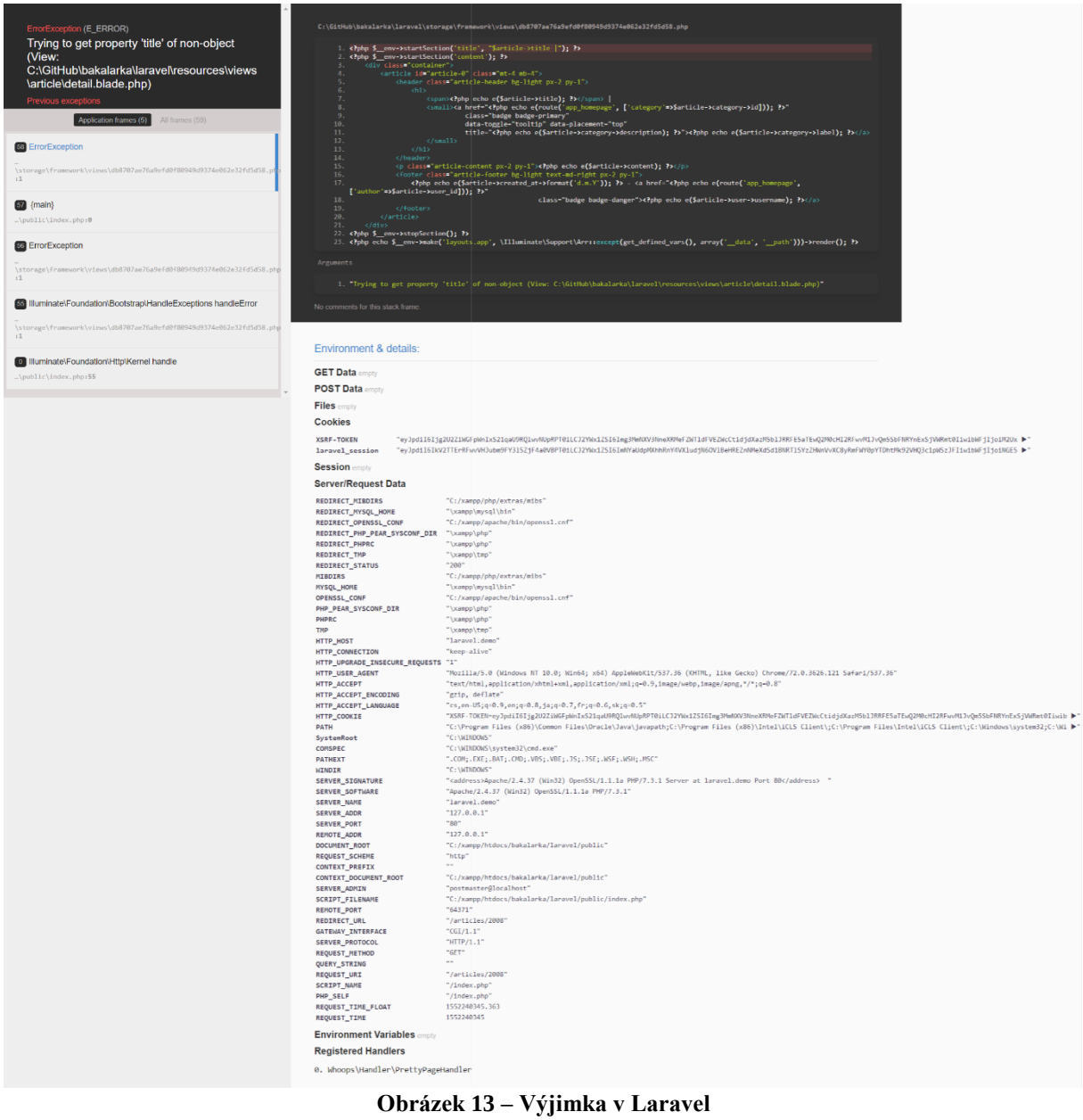
Filename: C:\GitHub\bakalarka\codeigniter\application\controllers\Article.php

Line Number: 34

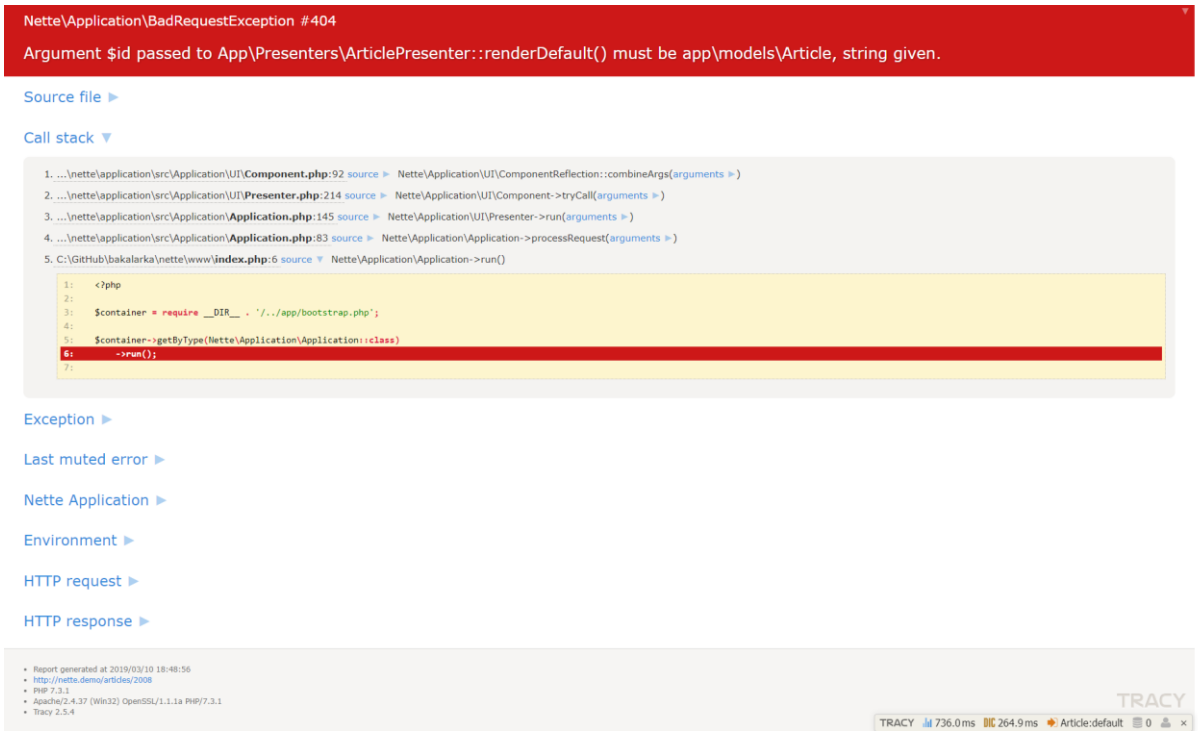
Backtrace:

File: C:\GitHub\bakalarka\codeigniter\index.php
Line: 315
Function: require_once

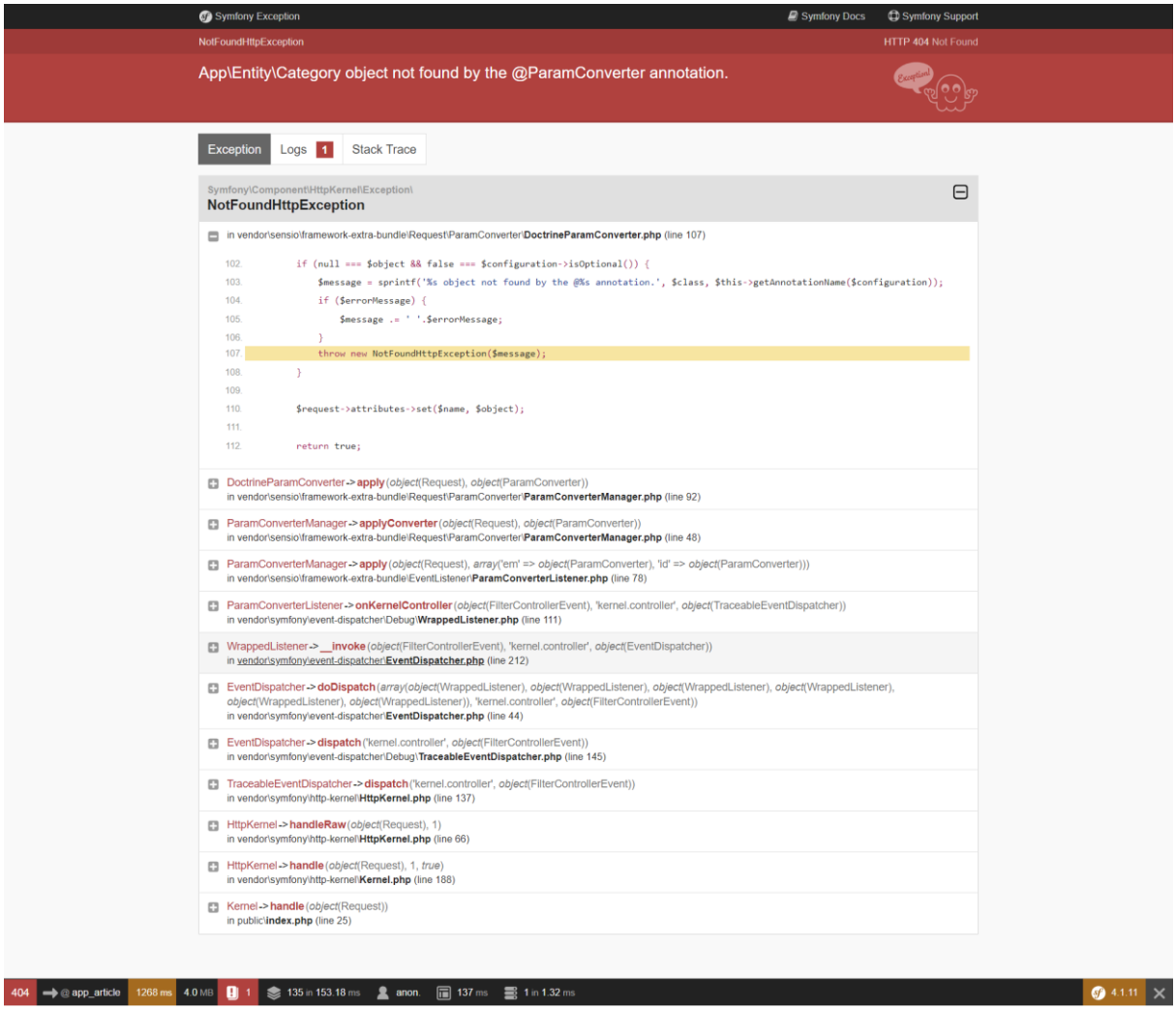
Obrázek 12 – Výjimka v Codeigniter



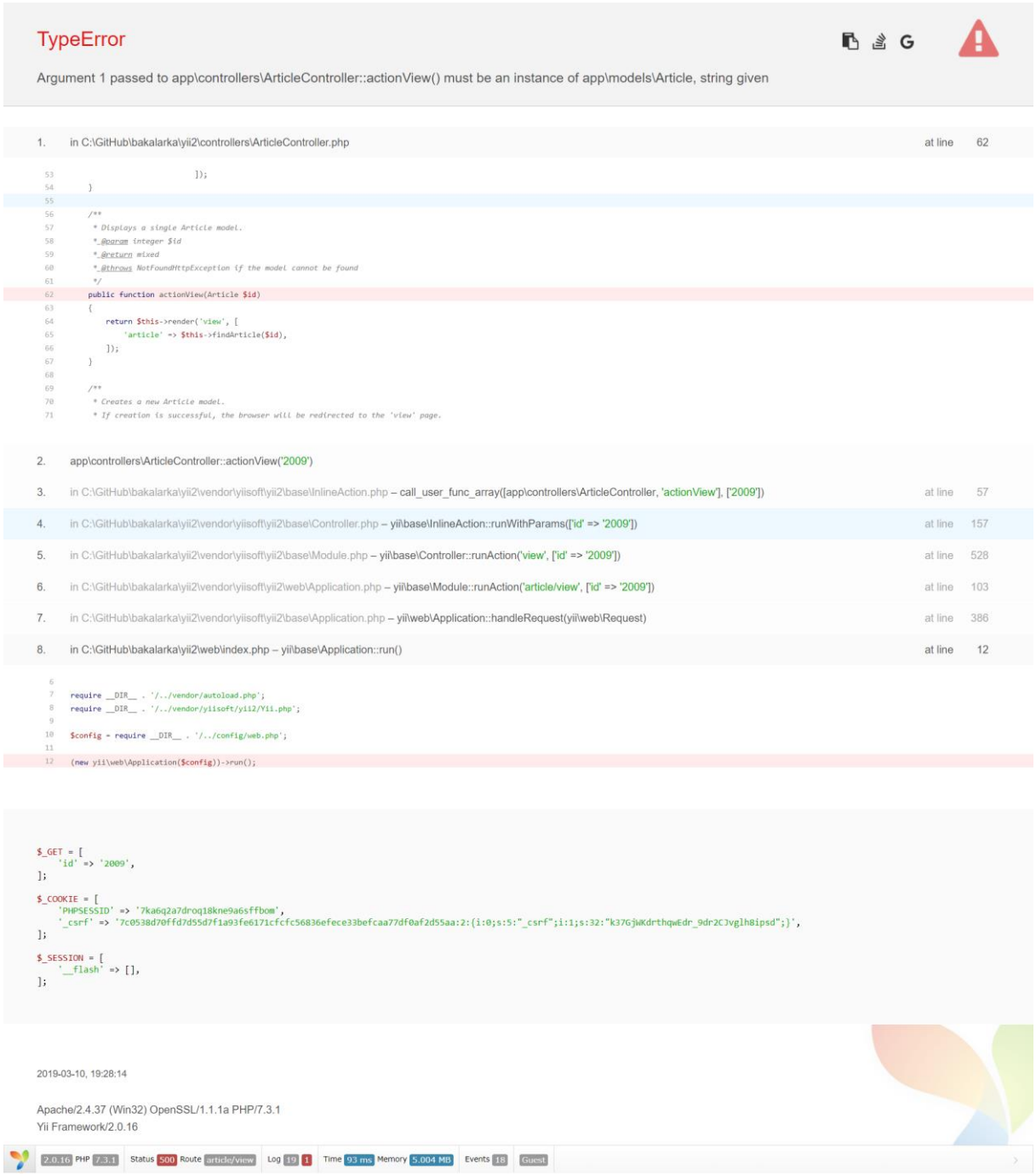
Obrázek 13 – Výjimka v Laravel



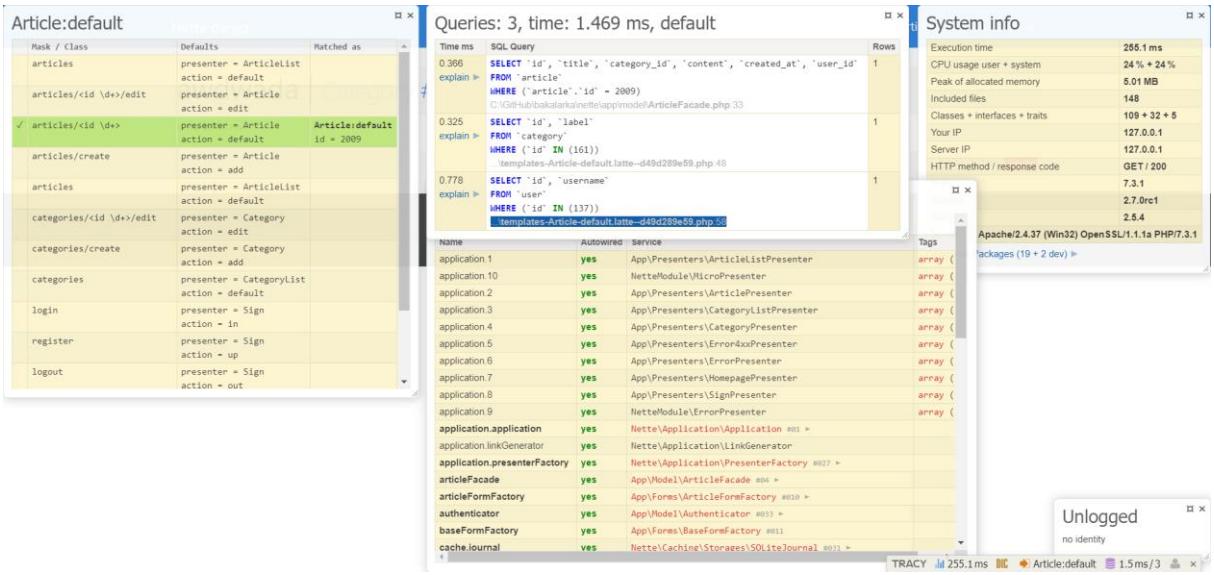
Obrázek 14 – Výjimka v Nette



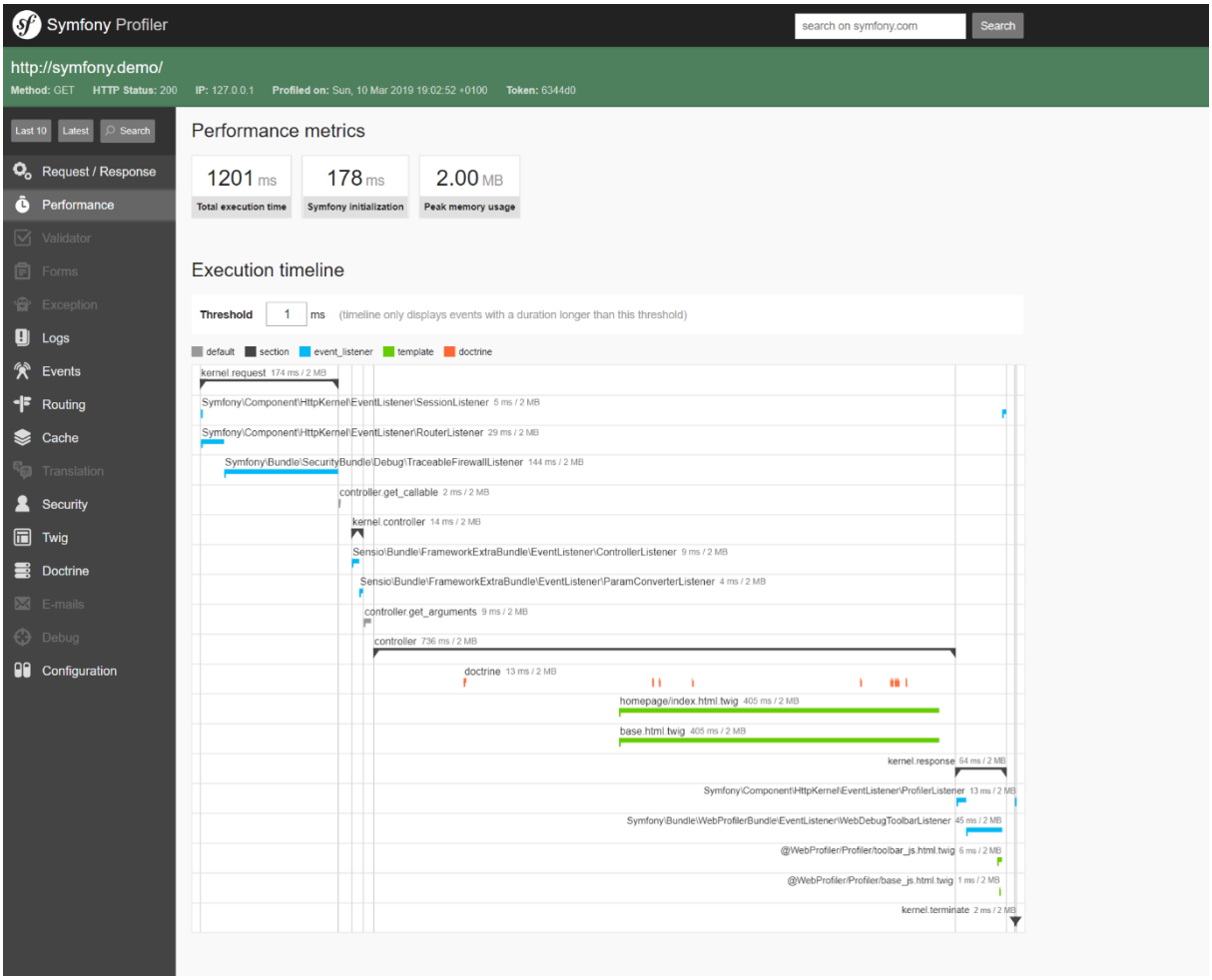
Obrázek 15 – Výjimka v Symfony



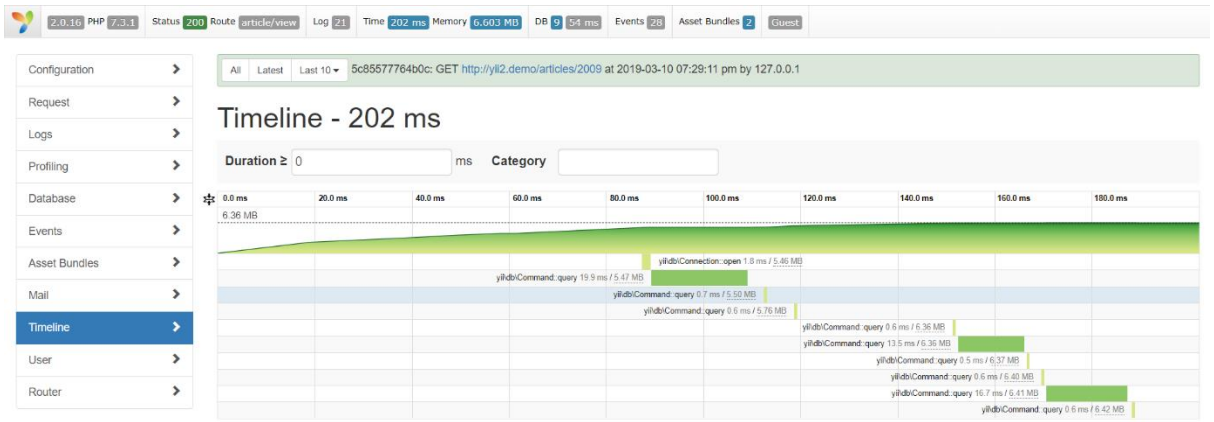
Obrázek 16 – Výjimka v Yii



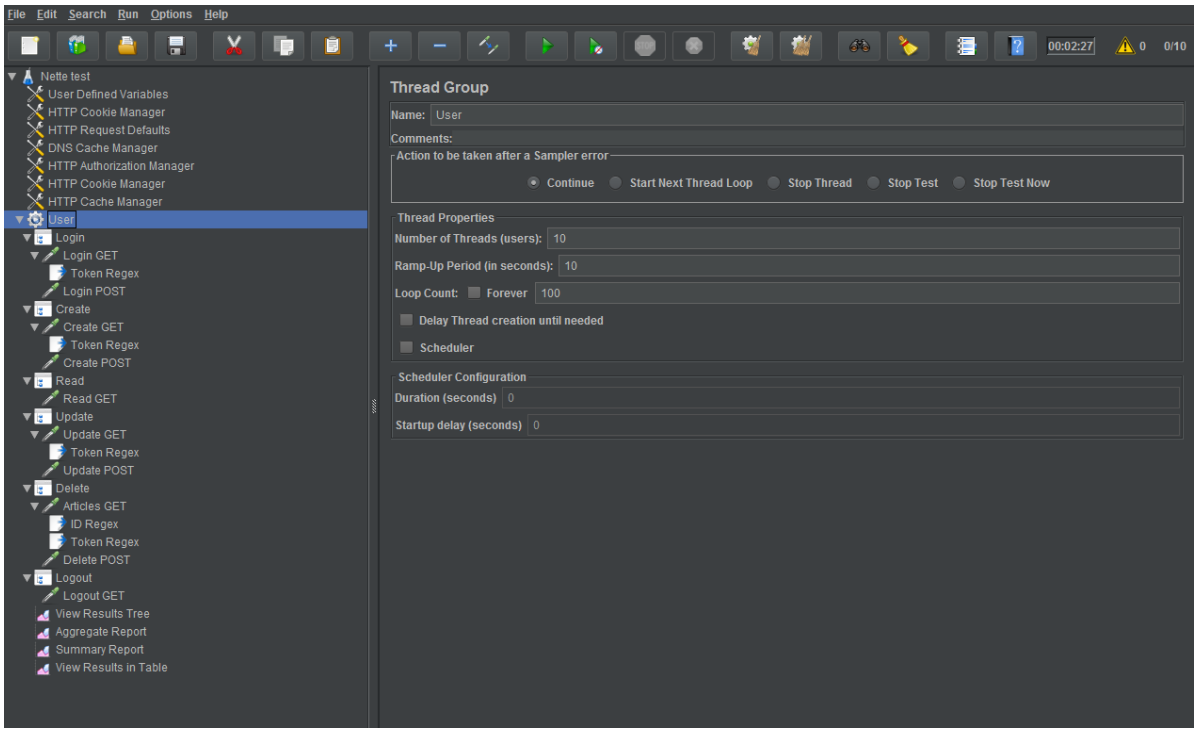
Obrázek 17 – Plovoucí ladící panel Nette



Obrázek 18 – Ladící profiler v Symfony



Obrázek 19 – Ladící profiler v Yii



Obrázek 20 – Testovací plán JMeter Thread Group (Nette)