Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Monitoring funkcionality webových aplikací

Plzeň 2021 Valentin Horáček

ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta aplikovaných věd

Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Valentin HORÁČEK, DiS.**
Osobní číslo:	**A20B0105P**
Studijní program:	**B0613A140015 Informatika a výpočetní technika**
Studijní obor:	**Informatika**
Téma práce:	**Monitoring funkcionality webových aplikací**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Seznamte se se stavem monitorování webových aplikací a pozornost věnujte i předpokládaným praktickým potřebám provozovatelů těchto aplikací. Dále se seznamte s vhodnými nástroji pro ovládání webových aplikací, zejména se Selenium Grid.
2. Navrhněte aplikaci ovládanou z webového rozhraní, která umožní opakovaný monitoring zvolených webových aplikací. Navrhněte rozhraní, které umožní jednotně začlenit již existující testy, které pak budou provádět vlastní monitoring.
3. Realizujte navrženou aplikaci s pomocí vhodných technologií tak, aby uživatelsky příjemným způsobem umožnila snadnou volbu rozsahu a frekvence monitoringu a poskytovala přehledně výsledky své činnosti.
4. Vytvořenou aplikaci ověřte na minimálně třech existujících webových aplikacích s využitím již existující sady jejich testů. Dosažené výsledky zdokumentujte a kriticky zhodnoťte.

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce: **Doc. Ing. Pavel Herout, Ph.D.**
Katedra informatiky a výpočetní techniky

Datum zadání bakalářské práce: **5. října 2020**
Termín odevzdání bakalářské práce: **6. května 2021**

L.S.

Doc. Dr. Ing. Vlasta Radová
děkanka

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 26. října 2020

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 24. června 2021

Valentin Horáček

Abstract

The bachelor thesis focuses on the issue of monitoring of web applications with user defined functions. The main goal is to design and implement a web interface for uploading and periodically executing of monitoring scripts based on user requirements. In addition to the monitoring application, a complete technological solution is described, including designed database model and server settings. Application can monitor scripts on different web browsers according to user defined set of the browsers.

Abstrakt

Bakalářská práce se zabývá problematikou monitorování zákazníkem definované funkčnosti webových aplikací. Cílem je navrhnout a implementovat webové rozhraní pro nahrávání a periodické pouštění dle požadavků zákazníka vytvořených monitorovacích skriptů. Kromě monitorovací aplikace, je popsáno kompletní technologické řešení včetně navrženého databázového modelu a nastavení serveru. Aplikace umožňuje monitorování pomocí několika různých webových prohlížečů, jejichž výslednou množinu si volí zákazník.

Obsah

1 Úvod

Cílem této práce je vytvořit webovou aplikaci, která umožní opakovaný monitoring zvolených webových aplikací. Aplikace je zaměřena na firmy, jejichž byznys je založen nebo závisí na rozsahově menších či středních webových aplikacích, které byly vyvinuty třetí stranou.

Aplikace vytvářená v této bakalářské práci by měla sloužit k periodickému monitorování řádného běhu webových aplikací a v případě jejich zjištěné nefunkčnosti k reportování o chybách. Majitel webových stránek by tak měl možnost rychlé reakce na případnou chybu dříve, než ho o tom informují nespokojení zákazníci.

Úkolem aplikace není problémy vyřešit, ale pouze na ně prokazatelně co nejdříve po jejich výskytu upozornit. Nápravu pak musí provést buď správci hostujícího webového serveru nebo třetí strana odpovědná za aplikaci.

2 Monitorování aplikací

S postupným rozšířením internetových technologií přibylo množství webových aplikací a služeb. I když takové aplikace usnadňují přístup běžným uživatelům, stává se, že jsou tyto služby z nějakého důvodu občas nedostupné. To může mít výrazně negativní dopad jak na uživatele tak na majitele webových stránek. Nejvíce je to kritické u služeb v podnikovém prostředí, kdy výpadek může znatelně ohrozit kontinuitu podnikání. Monitorování je pro takové aplikace nepostradatelnou součástí. Několik typů monitorování bylo navrženo a je dále popsáno, ale nejprve si rozdělíme typy webových aplikací.[9]

2.1 Typy webových aplikací

Mluvíme-li o webových aplikacích, myslíme tím software, který se spouští v internetovém prohlížeči a který je implementován s využitím jednoho z programovacích jazyků, jako je například `JavaScript`, `Java` a `C#`, nebo s použitím datových formátů jako je `XML`, `HTML` nebo `CSS`. Technologie využívané pro tvorbu webových aplikací v posledních letech dosáhly významného pokroku, ale uvedené programovací jazyky patří k těm nejzákladnějším využívaným technologiím.

Podle způsobu implementace můžeme webové aplikace rozdělit do následujících tří skupin.

2.1.1 Statické stránky

Do této kategorie spadá většina webových stránek. Ke svému spuštění nepotřebují žádnou dodatečnou instalaci od uživatele. Webové aplikace napsané v `HTML5` můžeme zahrnout do této kategorie, neboť nevyžadují žádné jiné nástroje než webový prohlížeč.

2.1.2 Menší webové aplikace

Aplikace s programovým rozhraním (API) jsou aplikace, které poskytují nějaké služby, data nebo nástroje ostatním uživatelům. Jedná se spíše o sbírku procedur či protokolů, které je možno dále využívat. Aplikace jsou dynamické a interagují s uživateli, což může vést k jejich pádu. Příkladem takovéto aplikace je `University Information System (UIS)`.

Implementovaný systém dále popsaný v této práci se právě soustředí na monitoring aplikací tohoto typu.

2.1.3 Komplexní webové systémy

Posledním typem jsou aplikace, které se od předchozích liší větším rozsahem a komlexitou. Mezi takovéto aplikace můžeme zahrnout `Informační systém studijní agendy (IS/STAG)` využívaný na univerzitě.

2.2 Způsoby monitorování

Následuje výčet nejčastěji využívaných technik pro testování a monitorování běžných webových aplikací.

2.2.1 Tradiční monitorování webových stránek

Jako jedním z prvních řešení monitorování webových aplikací byly navrženy metody pro automatické generování testů za pomoci standardních protokolů. Web Services Description Language (WSDL) nebo Simple Object Access Protocol (SOAP) jsou platformově nezávislé protokoly na zasílání zpráv. Oba protokoly umožňují popis rozhraní webové služby a formátu přeposílaných dat.

Testy na základě WSDL jsou nejprve analyzovány a následně transformovány do DOM stromové struktury. Testovací případy se dělí na generování testů pro data a generování testů pro operace. Samotné testy jsou pak generovány na základě syntaxe XML schéma. [5]

2.2.2 Document object model-based automated monitoring

Jedná se o rozšíření tradičního monitorování webových aplikací. Webové aplikace dokáží vyjádřit svůj stav nebo updatovat obsah stránky pomocí `DOM` struktury. Různé studie se pak věnovaly hledání webových elementů na stránce, právě přes tuto strukturu. [11]

2.2.3 Visual-based automated monitoring

Webové elementy, jako jsou obrázky nebo widgety, se nechají také nalézt pomocí metod rozpoznávání obrazu. Obecně není tento způsob tak robustní

jako hledání elementů pomocí DOM struktury, ale v případech, kdy se monitorovaná aplikace stále vyvíjí, má své využití. V takovýchto případech je vzhled webové stránky více stabilní než struktura elementů. Elementy jsou na stránkách nalézány tím, že se simuluje interakce uživatele s Graphical User Interface (GUI). [10]

2.2.4 Capture/Replay-based monitoring

Nástroje pro funkční testování, které poskytují testerům grafické rozhraní. V tomto rozhraní lze velice jednoduše zaznamenávat uživatelskou interakci s webovou aplikací. Tyto záznamy je pak možno opakovat pro regresní testování. Nejznámějším nástrojem pro tento typ testování je Selenium IDE. Další kategorie tohoto typu monitorování využívá testovací framework, například JUnit, pro analýzu HTTP/HTTPS požadavků. Touto metodou lze odchytávat všechny síťové pakety pro webovou aplikaci a následně tak simulovat Java API. Nevýhoda tohoto přístupu je, že organizace a manipulace s takto připravenými testovacími sady je manuálně náročná.

2.2.5 Quality of Service monitoring

Pro zajištění kvality webových služeb můžeme sledovat faktor QoS. QoS je množina popisů a metrik. Ty mohou sledovat responsivitu a spolehlivost webových služeb, stejně jako celkový výkon dané služby.
Obecně se QoS metrika nechá rozdělit na základě způsobu jejího získání do 3 kategorií. [16]

- Metrika inzerována provozovatelem služby. Většinou ji udává provozovatel ke konkrétní službě. Může se jednat například o cenu inzerovanou provozovatelem.
- Metrika na základě zákaznického hodnocení. Jedná se o typ metriky, který je odvozen od zpětné vazby uživatelů a jejich hodnocení. Mezi tento typ metriky patří například reputace poskytované služby.
- Pozorovatelné metriky se nechají sledovat dle provozních událostí odvozených od činností jak zákazníka tak poskytovatele služby. Tyto metriky jsou většinou specifické pro konkrétní poskytovanou službu.

Pro metriky musíme zajistit, že budou spočítány a updatovány v reálném čase, neboť získané hodnoty mohou být časově kritickou informací.

2.3 Nástroje využitelné pro monitorování

Téměř většina nástrojů pro monitorování se soutředí na monitorování dostupnosti, výkonu či odezvy webových aplikací a kontrolu případného pádu některé ze služeb, například emailového serveru. Takovéto nástroje bohužel neumí spouštět v monitorované aplikaci testy a ověřit tak nejen dostupnost webové aplikace, ale i její obsah.

Na druhou stranu jsou hojně rozšířené nástroje pro psaní a vytváření testovacích skriptů s jednotlivými testy. Tyto testovací frameworky jsou často implementovány přímo ve zdrojovém kódu aplikace a pokud se dají spustit automaticky, vyžaduje to začnou znalost konkrétního systému a programování.

Komplexní nástroj, který obsahuje a kombinuje obě složky pro monitoring, je `Selenium`.

2.3.1 Selenium

Je open-source nástroj pro automatické monitorování a testování webových stránek a webových aplikací. Nejedná se pouze o jediný nástroj, ale o sadu softwarových aplikací pro podporu částí automatizovaného testování. Selenium se skládá ze 3 základních komponent.

- Selenium Grid – popsán níže v podkapitole 2.3.2.
- Selenium IDE – nástroj pro vytváření testovacích sad a jednotlivých testů.
- Selenium Webdriver – umožňuje automatizaci činnosti prohlížečů pomocí jejich `API` pro simulaci uživatelské interakce.

2.3.2 Selenium Grid

Slouží pro spouštění testů na různých webových prohlížečích, různých operačních systémech a zařízeních ve stejný čas. První předností tohoto způsobu monitorování je zajištění dostupnosti webové aplikace a její kompatibilitu na širokém množství kombinací operačních systémů a prohlížečů. Druhým způsobem užití je zrychlení doby běhu testovacích sad, jelikož testy mohou být spuštěny paralelně a lze tak značně urychlit potřebný čas.

Pro ověřování kvality softwaru je Selenium Grid jedním z nejvyužívanějších nástrojů. Zejména z důvodu, že dokáže replikovat interakci uživatele

a jeho chování na webové stránce či webové aplikaci bez potřeby manuálního vstupu.

Grid podporuje spuštění WebDriver skriptů na vzdálených zařízeních, reálných nebo virtuálních, přesměrováním příkazů z klientského zařízení na vzdálenou instanci webového prohlížeče. Tento přístup umožňuje centrální správu prohlížečů a není tak potřeba konfigurovat nastavení prohlížeče pro každý test zvlášť. [1]

Grid 3 Jelikož je tento nástroj aktuálně stále vyvíjen, je verze Grid 3 již zastaralá a nedoporučuje se ji využívat. Uvádím ji z důvodu, že porozumění této verzi je stále relevantní i pro novou a že na této verzi byly prováděny experimenty popsané v kapitole 5.1.

Zobrazení architektury pro Grid 3 můžete vidět na obrázku 2.1. Z tohoto obrázku je patrné, že Grid se skládá ze dvou komponent, které jsou popsány níže.

- `Hub` – Centrální prvek na lokálním zařízení, který dostává všechny požadavky na testy. Distribuuje jednotlivé testy na správný `Node`, který zajištuje spuštění konkrétního testu. Může být pouze jeden v celé síti. `Hub` spustí test, ale prohlížeče jsou spuštěny a automatizovány na jiném zařízení.

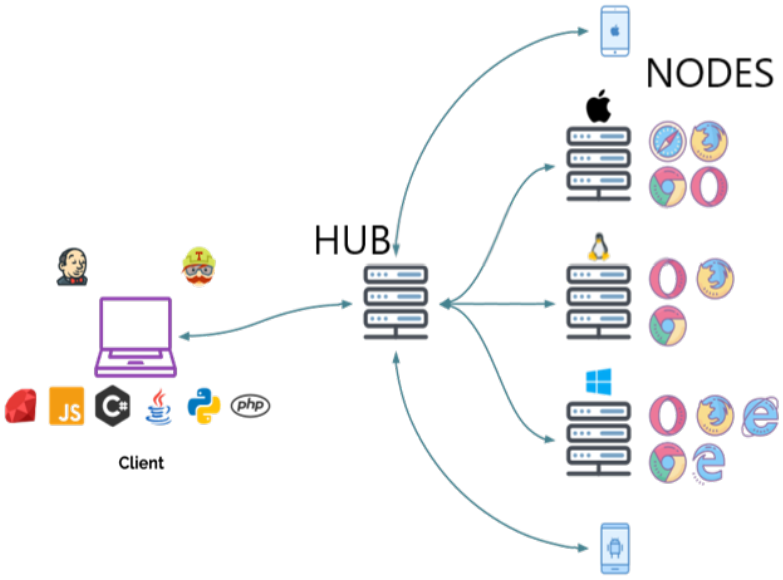
- `Node` – Instance Selenium, které spustí testy nahrané na `Hub`. Mohou být puštěny na více zařízeních s různými platformami a prohlížeči. V síti musí být alespoň jeden `Node`, ten ale nemusí být na stejné platformě jako je `Hub`.

Grid 3 lze nastavit pomocí příkazů přímo z konzole, nebo přes `JSON` konfigurační soubory pro `Hub` a `Node`.

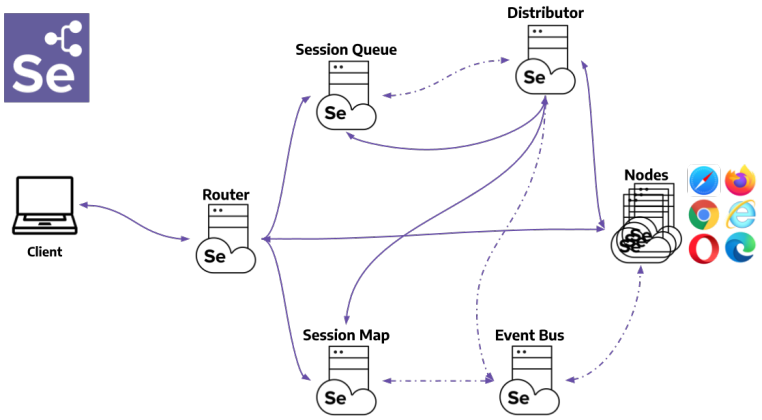
Grid 4 Aktuální verze Gridu nabízí implementaci nových technologií. I když 4. verze nesdílí `codebase` s předešlou verzí, stále podporuje spuštění na lokálním zařízení. Na obrázku 2.2 je znázorněna nová architektura pro Grid.

Tato verze se skládá z následujících komponent.

- `Router` – přijímá všechny externí požadavky a přeposílá je na požadovanou komponentu. Je to vstupní bod sítě a jeho chování je dáno typem požadavku. Úkolem `Routeru` je také vyvažování zátěže sítě a posílat požadavky na komponentu, která daný požadavek zvládne obsloužit nejlépe bez přetížení ostatních.



Obrázek 2.1: Grid 3 - architektura



Obrázek 2.2: Grid 4 - architektura

- `Distributor` – hlavním jeho úkolem je zpracovávat požadavek na vytvoření nové `session`. Ze seznamu vybere nejvhodnější `Node` podle jejich nastavených parametrů. Záznam o relaci mezi `session` a konkrétním `Node` je pak uložena do `Session Map`.
- `Node` – jako jediná komponenta se může v síti vyskytovat vícekrát. V defaultním nastavení `Node` automaticky registruje všechny webové ovladače dostupné na daném zařízení, kde je nastaven. Přes `Event Bus` se `Node` registruje na `Distributor` spolu se svojí konfigurací. Hlavním

úkolem je spravovat sloty pro nastavené prohlížeče a provádět přijmuté požadavky. Neprovádí žádnou kontrolu či hodnocení a může být nastaven na jiné platformě než ostatní prvky.

- `Session Map` – uchovává záznamy o `session id` a `Node`. Slouží jako podpůrný prvek pro `Router` pro přeposílání požadavků na konkrétní `Node`.
- `Session Queue` – má na starosti vytváření nové `session`. Kontroluje požadované vlastnosti pro `Node`. Pokud žádný `Node` neodpovídá nastavení, je požadavek okamžitě zahozen. Pokud existuje vhodný `Node`, vybere se volný slot, který je následně předán na `Distributor`. V případě, že není žádný slot volný, opakuje se požadavek dokud není úspěšný nebo dokud nevyprší čas na jeho provedení.
- `Event Bus` – je implementována pro interní komunikaci mezi komponentami. V plně distribuovaném módu je to první nastartovaný prvek. Komunikace je realizována pomocí zpráv, aby se vyhnulo zbytečně dlouhým `HTTP` požadavkům.

V předešlé verzi bylo možné spustit celý Grid, tedy `Hub` a `Node`, v samostatném módu. Obdobný princip je možný i pro novou verzi, kdy se komponenty `Router`, `Distributor`, `Session Map`, `Session Queue` a `Event Bus` spojí dohromady a fungují jako `Hub`. Samostatný funkční Grid pak získáme, když se přidá jeden `Node`.

3 Návrh aplikace

3.1 Zadání

Vytvoření systému pro pravidelné spouštění monitorovacích skriptů a ukládání výsledků jejich běhů do databáze. Návazným cílem je uživatelsky přívětivá vizualizace monitorovacích výsledků.

3.1.1 Komu může monitorování pomoci

Naprosto typicky se jedná o firmy, jejichž byznys je založen – nebo dokonce závisí – na rozsahově menších či středních webových aplikacích. Jedná se o firmy, jejichž webové aplikace vyvinuly třetí strany, které pak zajišťují jen opravy a aktualizace nikoliv vlastní běh aplikace.

Další potenciální využití je pro instituce státní správy, které potřebují mít uspokojivou případně i dokladovatelnou míru jistoty, že jejich aplikace fungují pro veřejnost.

3.1.2 K čemu je to dobré

Webové aplikace dnes každý zná a každý používá. Jsou pro uživatele velmi jednoduché – pro ovládání jim stačí jen jakýkoliv webový prohlížeč (Chrome, Safari, Edge, Firefox, Opera apod.). Ovšem ve skutečnosti jsou webové aplikace složité systémy sestávající z mnoha spolupracujících částí a tyto části jsou často umístěny i na různých počítačích a vzájemně spolupracují. A samozřejmě platí známé pravidlo, že čím je systém složitější, tím je v něm více možností, jak se může pokazit. K problémům dochází a docházet bude i přes značné úsilí všech správců systémů.

Primárním cílem monitorování webových aplikací je, když k problému dojde, aby se o případné nefunkčnosti dozvěděl její majitel jako první a měl čas zajistit nápravu. To, čemu se snažíme vyhnout je, že nefunkčnost oznamují telefonicky nespokojení zákazníci. A to je ještě ten lepší případ, protože častý jev je, že když zákazník není na aplikaci závislý (typicky, že webová aplikace je e-shop), pak její nefunkčnost neřeší, ale použije konkurenční aplikaci.

Pokud je webová aplikace provozována na nějakém solidně spravovaném webovém serveru, pak na něm určitě budou mít jeho správci nasazeny prostředky, které kontinuálně sledují stav tohoto serveru a všech jeho zásadních součástí, jako např. databázového serveru. A v případě nefunkčnosti sami rychle zasáhnou a zjednají nápravu.

Z tohoto pohledu by se mohlo zdát, že je vše pod kontrolou, ale ve skutečnosti je to jen první krok k nápravě a ten neřeší další problémy.

Jaké jsou to problémy:

- Na webovém serveru je umístěno („hostováno“) často větší množství navzájem nezávislých webových aplikací. Při problémech se serverem typu, že „spadne“, tj. je kompletně či částečně nefunkční, jej jeho správci opět uvedou do chodu. Ale můžete si být jisti, že zkontrolují správný chod všech hostovaných aplikací? Pravděpodobně ne a těžko to od nich můžeme požadovat, protože kontrola správného chodu jedné konkrétní aplikace je principiálně jiná, než kontrola aplikace jiné. Samozřejmě jsou tam společné prvky, které se dají zautomatizovat, např. kontrola možnosti přihlášení nebo kontrola spojení na databázi. Ovšem individuální vlastnosti aplikace nikdo nekontroluje, pokud to nemáte výslovně uvedeno ve smlouvě.

- Předchozí případ, kdy webový server nebo databázový „spadne“ je extrémní případ, který je s vysokou pravděpodobností ošetřen. Ovšem mohou nastat méně závažné poruchy typu extrémního zpomalení nebo částečné nefunkčnosti. Na tyto poruchy se přijde obtížněji a jen v případě, kdy ovlivňují činnost celého webového serveru, tj. všech aplikací. Ovšem, pokud je ovlivněna činnost jen části hostovaných aplikací, např. z důvodu špatné funkčnosti nějakého specializovaného modulu, pak mají správci serveru jen velmi omezené možnosti, jak se o výskytu této poruchy dozvědět.

- Další zdroj problémů, kdy webová aplikace není funkční, nemusí být vůbec na straně hostujícího serveru. Na něm může být vše v naprostém pořádku. Problém je na straně zákazníka, respektive jeho webového prohlížeče. Běžně používaných webových prohlížečů je několik – ty nejznámější jsou Chrome, Safari, Edge, Firefox, Opera. Pokud webovou aplikaci vyvíjela nějaká solidní firma (a ta je často jiná, než firma provozující hostitelský webový server), pak webovou aplikaci vytvářela tak, aby byla co nejméně (omezit na nulu to nelze) závislá na

typu webového prohlížeče. A při vývoji toto důsledně zkontrolovala, tj. otestovala existující webovou aplikaci pomocí všech hlavních prohlížečů. Problém je ale v tom, že webové prohlížeče se často aktualizují a většina zákazníků má zapnutou jejich automatickou aktualizaci. Takže při vývoji webové aplikace byla tato řádně otestována webovým prohlížečem verze X, ale již po roce je tento prohlížeč ve verzi X+3. Samozřejmě, že webová aplikace nemůže úplně nefungovat v novější verzi webového prohlížeče, ale nemusí tam fungovat správně nějaká její část.

- Dalším problémem, který není úplně známý, jsou jazykové mutace webových prohlížečů. I když je webová aplikace v češtině, určitě existují zákazníci, kteří mají svůj webový prohlížeč v angličtině. Tím se předchozí problém rychlé změny verzí webových prohlížečů násobí o jejich případně jazykové mutace.

Na všechny výše zmíněné problémy může monitorování webové aplikace upozornit, protože je připraveno přesně na míru monitorované aplikaci.

Pozor – nemůže problémy vyřešit! Pouze na ně prokazatelně co nejdříve po jejich výskytu upozorní. Nápravu pak musí provést buď správci hostujícího webového serveru nebo firma, která webovou aplikaci vytvářela. Důležité je ale to, že se o problému ví!

Výhoda monitorování je, že je to aktivita nezávislá na provozovateli hostujícího webového serveru, provozovaná na jiném serveru. Takže není pravděpodobné, že by spadly jak webový server hostující vaši webovou aplikaci, tak i současně i jiný server hostující monitorovací aplikaci.

Z tohoto faktu vyplývá i další možné použití monitorování. V případě, že máte s firmou hostující webovou aplikaci smlouvu o nepřetržitém běhu 7/24 (tj. sedm dní v týdnu a 24 hodin denně), pak můžete díky monitorování částečně kontrolovat, zda je tento závazek plněn. Při pádu hostitelského serveru nebo jeho problémech se samozřejmě jeho správci snaží uvést co nejrychleji věci do pořádku, ale pravděpodobně se s tím nebudou chlubit. Takže webová aplikace může být nějakou dobu mimo provoz. Protože je ale výsledek monitorování prokazatelně zaznamenán, máte pak případně důkazy o tom, že závazek 7/24 nebyl zcela dodržen.

3.1.3 Životní cyklus

Celá aplikace je postavena na uživatelském nastavení neboli poptávce. Tu si může zvolit dle implementovaných prostředků. Existující poptávky musí dále doplnit administrátor. Poptávka je poté monitorována dle nastavených hodnot. Podrobnější popis všech požadavků je popsán v podkapitole 3.2.1.

3.2 Požadavky

Požadavky na aplikaci jsem rozdělil do 3 kategorií. Funkční požadavky popisují minimální předpokládané chování aplikace, které je nutné zaručit. Operačními požadavky je myšleno, jak vyřešit nasazení aplikace do reálného prostředí a jak bude vyřešena komunikace mezi jednotlivými celky aplikace. Vlastní preference pro programovací nástroje, jazyk či framework jsou pak zahrnuty ve vývojových požadavcích.

3.2.1 Funkční požadavky

Chování programu se odvíjí od typu uživatele, který k aplikaci přistupuje. Aplikace je koncipována pro běžného uživatele bez jakýchkoli technických znalostí. Webové stránky pouze umožňují přístup do systému, ale pokud chce uživatel využívat monitorovací službu musí komunikovat s administrátorem.

Nepřihlášený uživatel má možnost si prohlédnout přednastavenou demonstrační ukázku poptávky. Poptávkou je myšleno nastavení pravidelného spouštění. Na tomto defaultním nastavení si pak může prohlédnout náhodně vygenerované výsledky. Pokud si uživatel chce vyzkoušet nastavit vlastní poptávku, musí se do systému přihlásit.

Přihlášený uživatel si nejprve nastaví vlastní poptávku. Může si nastavit na jakých prohlížečích chce monitorovat (Edge, Firefox, Chrome), periodu spouštění monitorovacích skriptů a způsob informování o výsledcích. Periody spouštění jsou buď každý den, týden nebo každý měsíc ve specifický čas. Monitorovací skripty obsahují několik testů dohodnutých a naprogramovaných s administrátorem monitorovací aplikace. Informovat uživatele lze ihned po provedení monitoringu s neúspěšným výsledkem, po provedení každého (i úspěšného) monitoringu či reportovat pouze jednou za periodu, která se nemusí shodovat s periodou monitorování. Reporty jsou uživateli zaslány emailem či `SMS` s co nejjednoduším shrnutím a odkazem na aplikaci, kde si lze prohlédnout detailní výsledky. Po vytvoření poptávky musí uživatel

kontaktovat administrátora, který připraví monitorovací skripty (testovací sadu). Teprve až v momentě, kdy administrátor naprogramuje jednotlivé testy a nahraje monitorovací skript do systému, jsou uživatelské poptávky pravidelně monitorovány a jejich výsledky zaznamenávány. Uživatel si může kdykoli prohlédnout výsledky monitorování, které musí být přehledně zobrazeny. Z každého běhu musí být jasně patrná úspěšnost testů.

Administrátor přistupuje k aplikaci obdobně jako běžný uživatel s tím rozdílem, že má přístup k větší části systému. Hlavním úkolem administrátora je nahrávání testů do systému, proto má přístup na přehled všech aktivních poptávek, kde má připravené rozhraní pro nahrávání souborů s testy. V případě potřeby si může prohlédnout výsledky ostatních uživatelů.

3.2.2 Operační požadavky

Původní plán nasazení na lokální počítač na univerzitě na Internet Information Services (IIS) bylo potřeba upravit z důvodu omezeného přístupu na univerzitu. Jelikož se nejednalo pouze o aplikaci, ale i o databázi, přišlo mi jednodušší mít jeden server, který zvládne obě části. Z tohoto důvodu jsem se rozhodl pro řešení na linuxovém serveru, než například využít službu Azure.

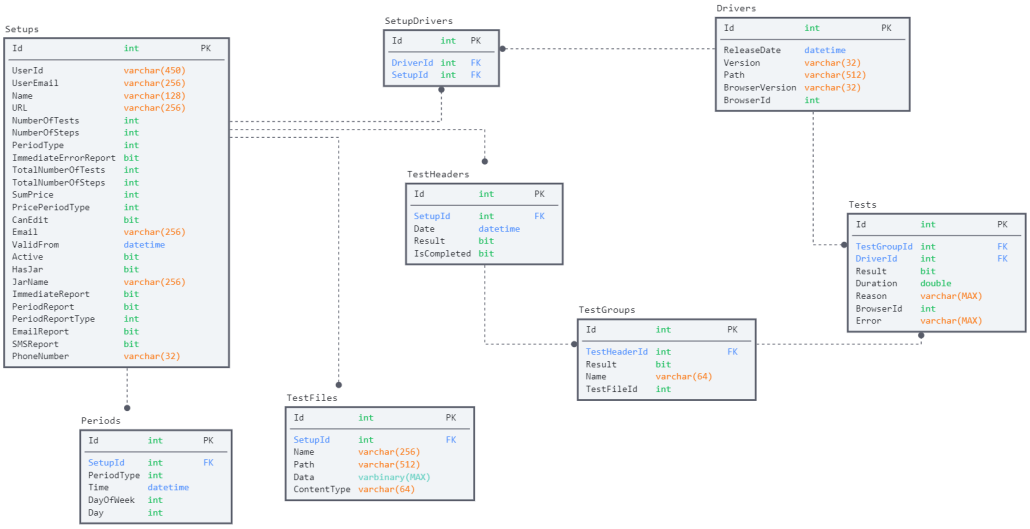
3.2.3 Vývojové požadavky

Monitorovací systém musí umět komunikovat s programem pro spouštění testů. Ten byl poskytnut vedoucím práce a je napsán v jazyce Java. Pokud by byla výsledná aplikace implementována v jazyce Java, asi by byla lepší návaznost na tento již vytvořený projekt, ale má neznalost Java webových aplikací a naopak zkušenost s jazykem C# a frameworkem ASP.NET mi donutily se rozhodnout pro druhou možnost. Toto rozhodnutí podpořilo vydání verze .NET 5 a chuť naučit se s novou verzí pracovat.[13] Tím bylo rozhodnuto, že jako IDE pro implementaci budu využívat Visual Studio 2019, která tento typ projektu podporuje. V návaznosti na volbu IDE jsem se rozhodl pro Microsoft SQL databázi díky její jednoduché integraci s tímto prostředím.

3.3 Návrh databáze

Návrh databáze prošel několika úpravami, ale už od počátku bylo patrné, že základním kamenem bude tabulka nastavených poptávek a testy bude

potřeba rozdělit dle času, testovaného prohlíže a napsaného testu. Výsledný návrh je zobrazen na obr. 3.1. Hlavní částí databáze je tabulka `Setups`, do které se ukládají uživatelská nastavení poptávek. Tabulka `Drivers` obsahuje uložené webové drivery, které jsou vázané na webové prohlížeče, na kterých se provádí monitoring. Do tabulky `Periods` jsou zapsány časy, kdy se mají testy spouštět. `TestHeaders` pak zaznamenávají hlavičky monitorovacích záznamů s konkrétním datem spuštění. `TestGroups` pro každou z hlaviček drží záznam pro každý test a nakonec do tabulky `Tests` jsou uloženy výsledky konkrétního monitorovacího záznamu pro zvolené prohlížeče. Poptávky mají také vazbu na tabulku `TestFiles` s uloženými soubory s testy. `TestFiles` a `TestGroups` nejsou propojeny vazbou, i přesto, že je tam patrné určité logické propojení. Vazbu jsem vynechal, protože mi postačilo zaznamenávat identifikátor souboru do skupiny daného testu a vazba by dle mého názoru zbytečně navýšila režii na vkládání a čtení záznamů.



Obrázek 3.1: ERA model databáze

4 Implementace

Z návrhu vyplynulo, že nejsnažší pro mne bude implementovat program v jazyce C#, přesněji ve frameworku .NET 5. Jako hlavní nástroj pro implementaci bylo využito IDE Visual Studio 2019 a pro práci s databází SQL Server Management Studio (SSMS). Aplikaci s databází bylo také nutné publikovat do reálného prostředí.

4.1 Server

Jako operační systém serveru jsem zvolil `Ubuntu 18.04` LTS pro co největší kompatibilitu s ostatními nástroji. Kompletní návod a postup pro nastavení serveru je uveden v příloze B. Návod také obsahuje postup pro přípravu SQL databáze a kompilaci webové aplikace.[6]

Linode Pro hostování linuxového serveru jsem využil službu `Linode`. Díky tomu, že se jedná o placenou službu, byla větší záruka bezporuchového provozu. Pro využití SQL Serveru musel `Linode` server splňovat podmínku alespoň 2 GB operační paměti. Základní informace o serveru viz obr. 4.1. Při instalaci na vlastní server by bylo vhodné považovat tyto informace jako minimální požadavky na výkon serveru.

Obrázek 4.1: Základní informace o serveru

4.1.1 NGINX

Nastavení také obsahuje konfigurační soubory pro NGINX, který slouží jako reverzní proxy. To bylo potřeba nastavit pro spuštění .NET aplikace pod systémem Linux [14]. NGINX zajišťuje přesměrování URL do publikované webové aplikace. Webová stránka také podporuje HTTPS protokol, ale pro ten bylo potřeba zaregistrovat doménu. Po zakoupení domény bylo možné vytvořit certifikát nástrojem Certbot.

4.1.2 Databáze

Pro instalaci databazové instance byl využit SQL Server. Instrukce k jeho instalaci jsou taktéž popsány v příloze. Dále jsou připraveny dotazy pro vytvoření čisté databázové struktury a importy s defaultními daty. Defaultními daty je myšleno přednastavení povolených webových prohlížečů, driverů a uživatele s administrátorským právem.

4.2 Webová aplikace

4.2.1 .NET 5

Aplikace je postavena na frameworku `.NET 5`, což je nejnovější LTS verze sady .NET. Program dodržuje strukturu architektury MVC. Vstupním bodem aplikace je třída `Program.cs`, která zajišťuje správné sestavení všech potřebných komponent. Jednotlivé moduly jsou připojeny v konfigurační třídě `Startup.cs`. Mezi tyto konfigurace patří připojení do databáze, inicializace servisních tříd a repozitářů, emailová konfigurace a další nastavení webového projektu.

Modul uživatelského rozhraní má vlastní konfiguraci ve třídě `IdentityHostingStartup.cs`, kde se jedná zejména o nastavení složitosti hesla a obdobných atributů.

4.2.2 Struktura aplikace

Program WebMonitoring je rozdělen do tří na sobě závislých projektů, kterým jsou věnovány následující podkapitoly.

- WebMonitoring
- Webmonitoring.Data
- WebMonitoring.Tools

Pro celou aplikaci je využit `StyleCop.Analyzers` pro analýzu kódu a dodržení základních pravidel pro programování v jazyce `C#`. Nastavení analyzátoru je udáno přes soubory `GlobalSupressions.cs` a `stylecop.json`, které jsou uloženy v globální struktuře a nasdíleny do jednotlivých projektů. Jednotlivé části kódu jsou komentovány dle stejného nástroje, ale nakonec bylo potřeba potlačit varování pro automaticky vygenerovaný kód.

Jelikož periodického monitorování na serveru nešlo debugovat z `IDE`, implementoval jsem balík `Serilog`, který umožňuje přehledné logování s formátovatelným výstupem. Logování je umožněno několika způsoby a pro tuto aplikaci jsem zvolil logování do databáze do tabulky `Logs`. Aplikace sama o sobě loguje určité informace o webových požadavcích, takže lze snadno dohledat, jaké akce byly spouštěny. Pro produkční prostředí jsem nastavil úroveň informování na `Information`, protože je dostatečně podrobný, a nižší level `Debug` by zbytečně zahlcoval databázi, ale v průběhu vývoje byl nezbytný.

4.3 WebMonitoring

Hlavní projekt programu obsahuje veškeré konfigurace. Jsou zde také umístěny základní bloky architektury `MVC`, kterou `VS` samo předvygenerovalo pro usnadnění práce.

4.3.1 Model

Modely předepisují s jakými daty se na dané stránce bude pracovat. Standardně se využívá nový model pro každou stránku. Vyjímkou je `InputViewModel`, který je využit na dvou stránkách `Input` a `Edit`, jelikož model by byl totožný. Následuje seznam použitých modelů:

ViewModels

- DetailViewModel
- DriverViewModel
- ErrorViewModel
- InputViewModel
- OutputViewModel
- SetupsViewModel

Uvedeny jsou pouze názvy tříd reprezentující modely. Jejich popis je podrobněji vysvětlen v kapitole 4.3.3 popisující akce kontroleru, které na daných modelech závisí. Název pro modely je dle mých zvyklostí odbozen od toho, že model je zpracováván konkrétním `view`.

Do části modelů se zpravidla zahrnují i entity reprezentující databázová data. Ta jsou popsána v rámci daného projektu v podkapitole 4.4.

4.3.2 View

Jednotlivá View reprezentují individuální stránky monitorovacího systému.

ActiveSetups Přehled vytvořených aktivních poptávek pro administrátora systému. Po rozkliknutí názvu každé poptávky si může prohlédnout její výsledky. Proto aby se poptávka pravidelně spouštěla, musí administrátor nahrát soubory s testy.

AddDriver Další stránka pouze pro administrátora, která umožňuje přidání nového webového driveru do systému.

Detail Detail zobrazené poptávky obsahuje a přehledně zobrazuje uložené nastavení.

Edit Editace poptávky je umožněna prostřednictvím této stránky.

Index Úvodní stránka projektu obsahuje popis aplikace k čemu a jak se používá.

Input Vstupní formulář pro zadání nové poptávky.

Output Kompletní přehled výsledků seskupený pro přehlednost do rozklikávatelných časových úseků. V případě chyby lze test rozkliknout a zobrazit důvod selhání.

Kromě výše uvedených stránek byl použit existující vzor Idenity, který poskytuje defaultní registraci a přihlašování uživatele. Tyto stránky byly pouze přeloženy do češtiny a upraveny pro jednoduchou správu uživatelů.

4.3.3 Controller

Controller je řídící část webové aplikace, veškeré dotazy jsou zpracovány požadovanou akcí. Akce jsou opatřeny atributy podle požadované funkcionality. V programu jsou akce rozděleny na `GET` a `POST` dotazy. Akce s anotací `GET` slouží pro načtení a zobrazení stránek s daty, kdežto akce s anotací `POST` přijímají data odeslaná uživatelem do aplikace. Na úrovni kontroleru se také řeší, zda má uživatel dostatečná oprávnění k provedení daného dotazu. Tato kontrola je zabezpečena většinou atributem `Authorize`, ale pokud k akci přistupuje jak přihlášený tak nepřihlášený uživatel, vyhodnotí se chování akce v průběhu jejího volání.

HomeController Aplikace obsahuje pouze jediný kontroler. Většina akcí slouží k přepínání stránek a zpracování uživateli odeslaných dat. Kromě těchto akcí obsahuje kontroler pomocné metody na inicializaci nových modelů a nastavení defaultních hodnot.

Input Akce `Input` podporuje jak požadavek typu `GET`, tak `POST`. Metoda s `GET` připraví prázdný formulář pro zadání poptávky a předpřipraví defaultní hodnoty pro `InputViewModel`.
Při odeslání formuláře je akce a model odchycen požadavkem na `POST`. Odeslaný model nejprve validuje správnost dat dle nastaveného kontraktu. Pokud jsou data zadána správně, doplní se poptávka o ID uživatele a přes servisní třídu se uloží do databáze. Po vytvoření poptávky také proběhne vytvoření monitorovacích period na období následujícího měsíce, aby se snížila režie při pravidelném spouštění monitoringu.

Edit Akce edit funguje obdobně jako akce `Input` s tím rozdílem, že při `GET` požadavku je model načten z databáze podle přihlášeného uživatele. Akce využívá model `InputViewModel`, protože mi přišlo zbytečné mít dva naprosto totožné modely. Akce `Edit` by se nechala spojit s akcí `Input`, ale dle mého názoru by to snížilo čitelnost a srozumitelnost kódu a komplexitu kombinovaného pohledu. Z těchto důvodů jsem se rozhodl zachovat tyto akce prozatím odděleně.

Detail Tato akce slouží k načtení a prohlížení uživatelské poptávky. Pokud uživatel není přihlášen, načte se demonstrační ukázka ze souborů. Pro přihlášeného uživatele se načtou příslušná data z databáze. Data ať už ze souboru nebo z databáze jsou načtena do `DetailViewModel`, který je poté zobrazen na příslušné stránce.

Output Slouží k zobrazení monitorovacích výsledků. Akce není kontrolována atributem `Authorize`, ale kontrola na uživatele je implementována v těle metody. To je z důvodu, že k motodě má přístup i nepřihlášený uživatel, který si takto prohlíží náhodně vygenerovaná data. Do modelu `OutputViewModel` jsou seskupeny seznamy výsledků monitorování.

ActiveSetups Akce je přístupná pouze administrátorovi aplikace. Touto akcí načte všechny aktivní poptávky v systému do modelu `SetupsViewModel`.

AddDriver Metoda na zadávání nových webových driverů. Data jsou načtena v `DriverViewModel` a v případě odeslání nového driveru je tento model zachycen voláním `POST`.

DeleteFile Tato akce slouží k vymazání souboru s testy pro danou poptávku.

DeleteJarFile Akce k odstranění uloženého JAR souboru.

DeleteTests V případě potřeby lze vymazat historii monitorování pomocí této metody.

DownloadFile Akce `DownloadFile` je zavolána v případě, že si uživatel chce stáhnout uložený soubor s testy.

GenerateResults Podle uložených souborů s defaultní poptávkou je načteno nastavení. Akce následně zavolá příslušnou servisní třídu na generování náhodných monitorovacích výsledků.

Index Defaultní akce pro zobrazení stránky s informacemi o projektu.

4.3.4 Uživatelské rozhraní

Správa uživatelů je implementována pomocí modulu `APS.NET Core Identity`. Tento balíček zajišťuje jednoduchou implementaci uživatelského rozhraní pro správu hesel, profilů, rolí, identit a tokenů [3].

Modul má nastavenu svoji vlastní databázi, která vznikla migrací. Připojení do této databáze je uloženo v samostatném `ConnectionStringu`, což by v případě potřeby mělo zajistit jednoduší nahrazení za jiné uživatelské rozhraní. Aby se importované části balíčku nepletly s ostatními částmi aplikace je modul oddělen do vlastní části `Area/Identity`. Bylo použito co nejjednodušší nastavení identity a přihlašování. Pro registraci se vyžaduje po uživateli pouze email a heslo. Aplikace má pouze jednoho administrátora. Jedná se o uživatelský účet, který má přiřazenou roli `Admin`. Změnu administrátora je nutné provést přímo v databázi.

Defaultní nastavení pro administrátorský účet je následující.

```
User: monitor@test.com
Password: monitor0
```

4.4 WebMonitoring.Data

Obsahuje definice entity repozitáře a soubor `WebMonitoringContext`. Tento soubor definuje databázový kontext a do aplikace se připojí přidání následujícího kódu do metody `ConfigureService` v souboru `Startup.cs`.

```
var connection = Configuration["ConnectionStrings:
   WebMonitoring"];

services.Configure<ForwardedHeadersOptions>(
   options =>
{
    options.KnownProxies.Add(IPAddress.Parse("
       139.162.180.141"));
});

services.AddDbContext<WebMonitoringContext>(
   options => options.UseSqlServer(connection));
```

K datům se přistupuje přes `RepositoryPattern`, a veškeré repozitáře a jejich metody jsou popsány zde.

4.4.1 Entities

Database Databázové entity představují data, které se mapují na data z tabulek databáze. Kromě toho, že třídy obsahují parametry shodné se sloupci v tabulkách, obsahují také parametry pro lepší zobrazení a ulehčení práce.

- `Driver` – představuje data o webových driverech.
- `DriverSetup` – relace pro uložení vazeb mezi tabulkami `Driver` a `Setup`.
- `Period` – zaznamenává periodu spouštění poptávky.
- `Setup` – data uživatelské poptávky.
- `Test` – konkrétní data o výsledcích na zvoleném prohlížeči.
- `TestFile` – obsahuje parametry pro data monitorovacího skriptu.
- `TestGroup` – monitorovací záznam pro daný monitorovací skript.
- `TestHeader` – monitorovací záznam pro datum dle periody.

Input

- `BrowserType` – výčtový typ pro možné webové prohlížeče, na kterých se nechá monitoring provádět.
- `Date` – třída reprezentuje konkrétní datum zvolený pro monitoring.
- `DriverForm` – zadávací formulář pro přidání webového driveru.
- `Form` – vstupní formulář pro poptávku.
- `PeriodType` – výpis možných period.
- `PricePeriodType` – výčtový typ pro účtovací období.
- `Prices` – pomocná třída pro nastavení základních cen (viz dále `prices.json`).

Output

- `TimeResult` – třída reprezentuje zobrazené výsledky, které jsou vnořené a seskupené pro lepší přehlednost.

Utils

- `DropDown` – pomocná třída pro rozbalovací seznam.
- `MailSettings` – třída pro přenesení konfigurace to servisní třídy z globálního nastavení.

4.4.2 Files

Webová aplikace obsahuje 4 statické soubory a každý z nich má specifickou funkci. Soubory jsou ve formátu `JSON` pro jednoduchou serializaci do jazyka `C#` a zpět.

drivers.json Obsahuje webové ovladače pro defaultní poptávku.

prices.json V souboru jsou uloženy ceny za jeden test a jeden krok. Data jsou poté využita k výpočtu ceny poptávky.

setup.json Soubor obsahuje data demonstrační ukázky poptávky.

tests.json Do souboru se ukládají data vygenerovaná pro demonstrační ukázku. Data jsou ukládána do souboru, aby bylo možné využít stejných akcí v kontroleru pro přihlášeného i pro nepřihlášeného uživatele.

4.4.3 Repositories

Repozitáře jsou třídy dle návrhového vzoru `Repository`, který se využívá pro komunikaci s daty v databázi. Třídy existují pro každou databázovou entitu a dědí příslušné rozhraní. V konstruktoru každého repozitáře je nastaven databázový kontext, přes který pak jednotlivé metody přistupují k datům. Databázový kontext dokáže po dotazu vrátit data dle jejich propojení v databázi, což jednoznačně ulehčuje práci s daty, ale v některých případech to naopak může vézt k potížím.
Pro návrhový vzor jsem se rozhodl, protože jsem s ním dobře obeznámen a slouží dobře pro oddělení přístupu do databáze další vrstvou.

4.5 WebMonitoring.Tools

Pomocné služby pro vykonání specifických akcí jsou součástí tohoto projektu. Jedná se zejména o služby pro správu repozitářů, ale existují zde i ostatní služby pro specifické účely.

4.5.1 Services

DriverService Tato servisní třída pomáhá s manipulací webových ovladačů, jejich vazbou na poptávku a obecnému přístupu k datům uložených v databázi.

FileService Třída má za úkol řešit veškerou manipulaci se soubory, ať už se jedná o čtení či zápis. Třída také zajišťuje čtení souborů defaultní poptávky pro demostrační ukázku.

GeneratorService Jak název napovídá, služba generuje náhodné výsledky dle zadané poptávky. Během vývoje byla tato služba využívána pro generování výsledků monitorování na ověření stránky s jejich zobrazením.

JavaService Služba obsluhuje spuštění `JAR` souboru pro dané nastavení a přečtení výsledků testu. Dle nastavení je zavolán uložený `JAR` soubor se zvoleným testem, který je spuštěn s potřebnými parametry. Výsledky spuštění testu jsou předány zpět aplikaci. V případě chyby se do databáze zaznamená důvod selhání testu. Podrobnější popis formátu vstupů je v podkapitole 4.6.

MessageService MessageService je služba pro zaslání emailové či `SMS` zprávy. Zasílání emailu je řešeno přes Google účet. Tento účet musí mít

povolený přístup pro méně zabezpečené aplikace. [7] Během vývoje, jsem narazil na problém, že `Google` automaticky toto nastavení po nějakém čase vypne. Je tedy dobré ho průběžně kontrolovat a do budoucna, by chtělo toto omezení vyřešit. Metody pro zaslání `SMS` zpráv jsou připraveny pouze v kontraktu rozhraní, ale nejsou implementovány.

Přístup k zavedenému účtu je následující.

```
User: webmonitoring.info@gmail.com
Password: WebMonitoring0mail
```

PeriodService Služba pro přístup k datům nastavené periody spouštění.

ReportService Třída obsahuje metody pro vytvoření obsahu zasílaných reportů. Také řídí zasílání periodických reportů v daném časovém okně, obdobně jako se spouští monitorování, které je popsáno níže.

ScheduleService Ze servisních tříd je ScheduleService nejvíce odlišná. Nejedná se o přístup k databázovému kontextu přes repozitáře, ale jedná se o HostedService. [8] Úkolem této třídy je pravidelné spouštění monitorování. Služba je nastavena tak, aby se spustila alespoň každých pět minut. Pokud by proces monitorování v daném časovém úseku trval déle, aplikace si uloží datum posledního spuštění, kterým bude následující perioda spouštění začínat. Proces pak najde všechny poptávky v databázi, které měly v posledních pěti minutách nastavenou periodu spuštění. Poté se pro všechny vybrané periody dohledají testovací soubory, a pro každou takovou testovanou skupinu se zavolá JavaService na zvoleném webovém prohlížeči.

Po doběhnutí monitoringu a vyhodnocení jeho výsledku se dle nastavení odešlou okamžité reporty uživateli. Periodické reporty se spouštějí přes `ReportService`. Reportování výsledků je možno jen přes email, ale formuláře a databáze aplikace jsou připraveny na zasílání i SMS.

SetupService Má za úkol komunikaci s příslušným repozitářem k manipulaci s nastavenou poptávkou.

TestFileService Třída zajišťuje přístup k repozitáři s uloženými testovacími soubory.

TestService Servisní třída pro přístup k monitorovacím datů. Řeší přístup k repozitářům pro `TestHeader`, `TestGroup` a `Test`.

4.6 Formát monitorovacích skriptů

Zdrojový kód pro monitorovací skripty byl poskytnut vedoucím práce. Kód je napsán v jazyce `Java` s využitím `Selenium WebDriver`. Pro potřeby této práce byl zdrojový kód upraven pouze pro lepší komunikaci s webovou aplikací a předání nezbytně nutných parametrů. Implementace probíhala s využitím `Java JDK 11` a vývojového prostředí `IntelliJ IDEA`.

Nejpodstatnější úpravou je rozdělení jednotlivých testů do samostatných souborů. Test se pak volí na základě vstupního parametru. Dále bylo potřeba předat aplikaci informaci o tom, na jakém webovém ovladači se test spouští. To je udáno pomocí dalšího parametru. Hlavní třída `Hlavni.java` monitorovacího skriptu byla tedy potřebně upravena, aby splňovala zmíněné požadavky. Kód vzorového archivu je uveden se zdrojovými soubory a návod pro vytvoření je popsán v příloze A.5.1. Archiv vytváří administrátor Archiv je spouštěn tedy s následujícími vstupy.

```
java -jar [jar_file] [browser] [driver_path] [
   test_name]
```

- `jar_file` – spouštěný `JAR` soubor.
- `browser` – zvolený webový prohlížeč, dle výčtového typu.
- `driver_path` – cesta ke zvolené verzi webového ovladače.
- `text_name` – jméno monitorovacího skriptu.

Vstupní parametry jsou do webové aplikace načteny z databáze a při spuštění monitorovacího skriptu jsou předány metodě `RunJarAsync`, která vytvoří nový proces pro spuštění `JAR` souboru.

Výsledky navrácené z `JAR` souboru musí odpovídat následujícímu formátu pro úspěšný test.

```
BROWSER: name=edge, version=89.0.774.75
Start: 2021-04-11 22:07:09 > Prevodnik_Fail: FAIL,
    reason: '2.54<>10', duration[ms] = 3287
```

Nebo pro neúspěšný test.

```
BROWSER: name=edge, version=89.0.774.75
Start: 2021-04-11 22:07:15 > Prevodnik_HappyDay:
   pass, duration[ms] = 953
```

Formát výsledků byl ponechán beze změny, protože se využívaly již existující testy a formát obsahoval všechny potřebné informace.

5 Dosažené výsledky

Navržené rozhraní pro nahrávání existujících testů bylo dle zadání ověřeno na 3 existujících webových aplikacích. Jedná se o aplikace `Prevodnik`, `OsobniCislo` a `UIS`.

- `Prevodnik` – `http://oks.kiv.zcu.cz/Prevodnik/`
- `OsobniCislo` – `http://oks.kiv.zcu.cz/OsobniCislo/`
- `UIS` – `https://projects.kiv.zcu.cz/tbuis/web/page/homepage`

Aplikace byly vybrány z důvodu, že pro ně již existují napsané sady testů, které lze pro ověření výsledků využít a testy splňují požadovaný formát. Pro získání výsledků bylo tedy potřeba vytvořit nové poptávky. Pro každou z těchto aplikací byl proto vytvořen účet, aby výsledky mohly být zpětně dohledány.

Vytvořené účty mají následující přihlašovací údaje.

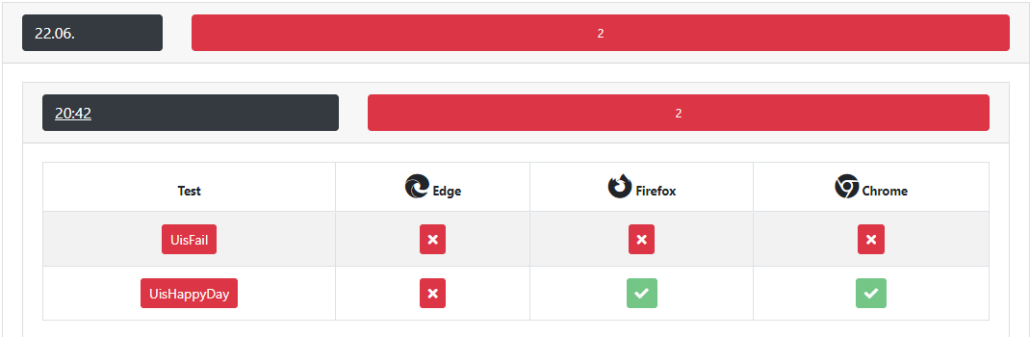
```
User: prevodnik@test.com
Password: prevodnik0

User: osobnicislo@test.com
Password: osobnicislo0

User: uis@test.com
Password: uis00
```

Po vytvoření poptávky bylo potřeba pouze sestavit monitorovací skript a spolu se soubory s jednotlivými testy je nahrát do monitorovací aplikace k příslušné poptávce. Výsledky testů nesou kromě informace o úspěšnosti testu také informaci o době jeho trvání. Tato informace je ukládána do databáze, ale není zatím dáno, jak by se dala využít. Jednou z možností je poskytnout grafy na stránce s výsledky a dobu trvání testu v tom zahrnout. Data by se dala dále využít pro statistiku a uvádět průměrný čas konkrétního testu.

Příklad ověřených výsledků lze vidět na obrázku 5.1. Uveden je pouze příklad pro aplikaci `UIS`, ale výsledky ostatních systému vypadají obdobně. Z obrázku 5.1 je také patrné, že monitorovací skript `UisHappyDay` selhal pro prohlížeč `Microsoft Edge`, což se při monitorování reálných aplikací



Obrázek 5.1: Ukázka výsledků – `UIS`

může stát, například z nedostupnosti aplikace či velké prodlevy u spouštěného skriptu. Monitorovací skripty je proto nutné psát co nejrobustněji, aby k podobným chybám zbytečně nedocházelo.

5.1 Experimenty se Selenium Grid

Po předběžné analýce problému v počátcích vývoje bylo zřejmé, že nástroj `Selenium Grid` bude v aplikaci do určité míry využit. V době pokusů bylo `Selenium` ve verzi 3, která je dnes již zastaralá.

Pokusy zahrnovaly nastavení vlastní sítě a s tím jeden `Hub` a jeden `Node`. Toto nastavení proběhlo v pořádku, ale na problém jsem narazil při implementaci různých webových prohlížečů na stejný `Node`. Také více uzlů se nepodařilo do sítě zahrnout a tak jsem zůstal u výchozího nastavení a využití `Selenium WebDriver` knihoven v existujících testech.

Nová verze `Selenium Grid` přináší komplexnější řešení a věřím, že by se tento nástroj nechal využít pro monitorovací aplikaci. Pokud by se dokázala nastavit síť, tak by to značně urychlilo proces spouštění monitorovacích skriptů, který sekvenčně spouští `JAR` soubor pro každý test zvlášť.

6 Závěr

Bakalářská práce se zabývala návrhem webové aplikace pro monitorování, která slouží jako rozhraní pro nahrání již existujících testů. Aplikace je navržena tak, aby si běžný uživatel bez znalostí programování mohl nastavit požadavky na monitorování vlastní aplikace. Potřebnou technologickou znalost pak dodá administrátor monitorovací aplikace, který po domluvě se zadavatelem dopíše potřebné testy. Výsledky monitorování jsou pak prezentovány v jednoduché a srozumitelné formě.

Na začátku práce jsem se zaměřil na existující nástroje vhodné pro monitorování, zejména na `Selenium Grid`. I když výsledky experimentů s tímto nástrojem nebyly úspěšné, představuje tento software možnost, jak aplikaci zlepšit.

Navržená databáze a webové rozhraní pak byly implementovány pomocí moderních technologií a běžně využívaných programovacích postupů. Kombinace programovacích jazyků `Java` a `C#` možná není úplně obvyklá, ale na základě mých znalostí, zkušeností a s využitím existujících testovacích sad to bylo nejjednoduší použitelné možné řešení.

Ověření monitorovací aplikace probíhalo na webových aplikacích, ke kterým byly dodány testovací sady, a funkčnost aplikace byla od začátku koncipována na tyto aplikace.

Výsledný monitorovací systém je funkční pro požadované účely, ale stále je zde místo pro rozvoj. Nástroj `Selenium Grid` by se dal mnohem víc využít. Monitorovací proces by bylo vhodné spouštět asynchronně a jednotlivé testy provádět paralelně a ne sekvenčně. Aplikace je dále připravena pro další rozšíření jako je reportování výsledků pomocí `SMS` nebo využití naměřené doby monitorování. Databáze, vstupní formuláře a data jsou připravené, ale samotná implementace těchto částí nebyla stěžejní pro tuto práci.

Literatura

[1] *Grid* [online]. Software Freedom Conservancy (SFC), 2021. Dostupné z: https://www.selenium.dev/documentation/en/grid/.

[2] Adegeo. *Install .NET on Ubuntu - .NET* [online]. Microsoft Docs, Jan 2021. Dostupné z: https://docs.microsoft.com/en-us/dotnet/core/install/linux-ubuntu#1804.

[3] Anderson, R. *Introduction to Identity on ASP.NET Core* [online]. Jul 2020. Dostupné z: https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-5.0&tabs=visual-studio.

[4] Angert, E. *Secure HTTP Traffic with Certbot* [online]. Jan 2021. Dostupné z: https://www.linode.com/docs/guides/secure-http-traffic-certbot/#what-is-certbot.

[5] Bai, X. et al. WSDL-Based Automatic Test Case Generation for Web Services Testing. *IEEE International Workshop on Service-Oriented System Engineering (SOSE'05)*. 2005. doi: 10.1109/sose.2005.43.

[6] Dykstra, T. et al. *Web server implementations in ASP.NET Core* [online]. Nov 2019. Dostupné z: https://docs.microsoft.com/en-us/aspnet/core/fundamentals/servers/?view=aspnetcore-3.1&tabs=linux.

[7] Google. *Manage third-party apps and services with access to your account* [online]. Google. Dostupné z: https://support.google.com/accounts/answer/3466521?hl=en.

[8] Huan, J. L. *Background tasks with hosted services in ASP.NET Core* [online]. Oct 2020. Dostupné z: https://docs.microsoft.com/en-us/aspnet/core/fundamentals/host/hosted-services?view=aspnetcore-3.1&tabs=visual-studio#consuming-a-scoped-service-in-a-background-task.

[9] Kim, S. – Suh, I. – Chung, Y. D. Automatic monitoring of service reliability for web applications: a simulation-based approach. *Software Testing, Verification and Reliability*. 2020. doi: 10.1002/stvr.1747.

[10] Leotta, M. et al. Visual vs. DOM-Based Web Locators: An Empirical Study. *Lecture Notes in Computer Science*. 2014, s. 322–340. doi: 10.1007/978-3-319-08245-5˙19.

[11] Marchetto, A. – Tonella, P. – Ricca, F. State-Based Testing of Ajax Web Applications. *2008 International Conference on Software Testing, Verification, and Validation*. 2008. doi: 10.1109/icst.2008.22.

[12] Namecheap.com. *How to Change DNS For a Domain - Domains* [online]. Namecheap, Mar 2021. Dostupné z: `https://www.namecheap.com/support/knowledgebase/article.aspx/767/10/how-to-change-dns-for-a-domain/`.

[13] Pine, D. – Kulikov, P. – Warren, G. *What's new in .NET 5* [online]. Microsoft, Nov 2020. Dostupné z: `https://docs.microsoft.com/en-us/dotnet/core/dotnet-five`.

[14] Shirhatti, S. *Host ASP.NET Core on Linux with Nginx* [online]. Oct 2020. Dostupné z: `https://docs.microsoft.com/en-us/aspnet/core/host-and-deploy/linux-nginx?view=aspnetcore-3.1#data-protection`.

[15] VanMSFT. *Ubuntu: Install SQL Server on Linux - SQL Server* [online]. Microsoft, Apr 2021. Dostupné z: `https://docs.microsoft.com/en-us/sql/linux/quickstart-install-connect-ubuntu?view=sql-server-ver15`.

[16] Zeng, L. – Lei, H. – Chang, H. Monitoring the QoS for Web Services. *Service-Oriented Computing – ICSOC 2007*. 2007, s. 132–144. doi: 10.1007/978-3-540-74974-5˙11.

Seznam zkratek

API Application Programming Interface. 10, 12, 13

CSS Cascading Style Sheet. 10

DNS Domain Name Server. 50

DOM Document Object Model. 11, 12

ERA Entity-Relationship model. 22

GB Gigabyte. 23, 55

GPG GNU Privacy Guard. 55

GUI Graphical User Interface. 12

HTML Hypertext Markup Language. 10

HTTP HypertextTransfer Protocol. 12, 16, 49

HTTPS HypertextTransfer Protocol Secure. 12, 23, 50

ID Identity. 27

IDE Integrated Development Envoronment. 12, 13, 21, 23, 25

IIS Internet Information Services. 21

IP Internet Protocol. 51

IS/STAG Informační systém studijní agendy. 11

JAR Java ARchive. 28, 31, 33, 36, 46, 47, 53

JDK Java Development Kit. 33, 53, 54

JSON JavaScript Object Notation. 14, 30

LTS Long-term support. 23, 24, 53

MVC Model-View-Controller. 24, 25

QoS Quality of Service. 12

SA Server Administrator. 55

SDK Software Development Kit. 48

SMS Short Message Service. 20, 31, 32, 37

SMTP Simple Mail Transfer Protocol. 53

SOAP Simple Object Access Protocol. 11

SQL Structured Query Language. 21, 23, 55, 56

SSL Secure Sockets Layer. 49, 50

SSMS SQL Server Management Studio. 23, 56, 57

UIS University Information System. 10, 35, 36

URL Uniform Resource Locator. 23

VS Visual Studio 2019. 23, 25

WSDL Web Services Description Language. 11

XML Extensible Markup Language. 10, 11

A Uživatelská příručka

Příručka popisuje základní použití aplikace pro monitorování webových aplikací. Systém je momentálně nastaven a je dostupný na adrese `https://web-monitoring.net/`. V následujících kapitolách je popsáno jak s aplikací pracovat.

A.1 Nepřihlášený uživatel

Při prvním načtení stránky se může nepřihlášený uživatel dozvědět, o čem aplikace je, jak se využívá a k čemu ji lze využít. Na záložce `Poptávka` v hlavním menu si uživatel může prohlédnout demonstrační ukázku poptávky. Co jednotlivé části znamenají, je popsáno v podkapitole A.3. V horní části detailu ukázky se pak nachází tlačítko pro vygenerování náhodných výsledků. To vygeneruje náhodné výsledky dle demonstrační ukázky na dobu jednoho měsíce a přesměruje uživatele na jejich přehledné zobrazení.

A.2 Registrace a přihlášení

Pro registraci, přihlášení a obecnou správu uživatelského účtu se v pravé části hlavního menu nachází ovládací prvky pro tyto akce.

Nepřihlášený uživatel má pouze možnost se registrovat pod novým účtem, nebo se přihlásit již pod existujícím účtem.

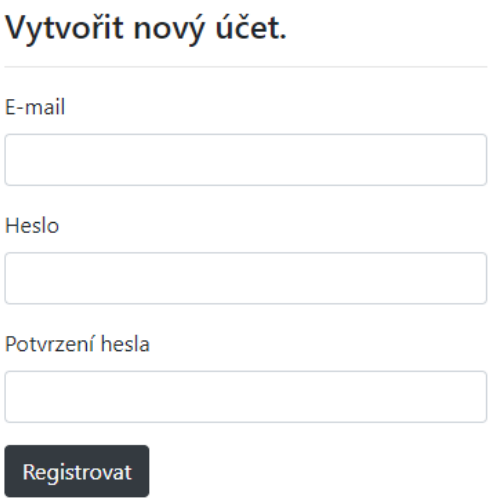
Na obrázku A.1 se nachází formulář potřebný pro registraci. Aplikace vyžaduje od uživatele pouze email a heslo.

Pro přihlášení uživatel musí zadat registrovaný email a heslo a poté má plný přístup (dle svých práv) do webové aplikace. Pokud si uživatel nepamatuje heslo pro přístup, má možnost si ho resetovat a příslušný odkaz mu bude odeslán na zadaný email. Obrázek A.2 ukazuje obrazovku pro přihlášení.

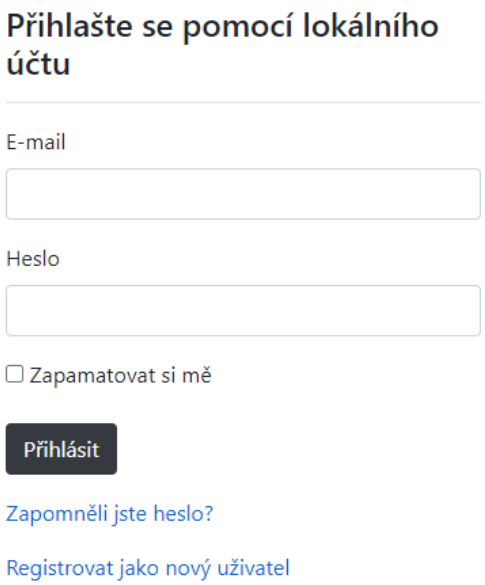
Pokud není užvatel přihlášen, může si prohlédnout formulář poptávky a nechat si tlačítkem `Generovat výsledky` vygenerovat náhodné výsledky.

A.3 Vytvoření poptávky

Na vytvoření poptávky neboli uživatelského nastavení monitorování se uživatel dostane z hlavního menu přes odkaz `Poptávka`. Má-li uživatel již po-



Obrázek A.1: Poptávka - registrace uživatele

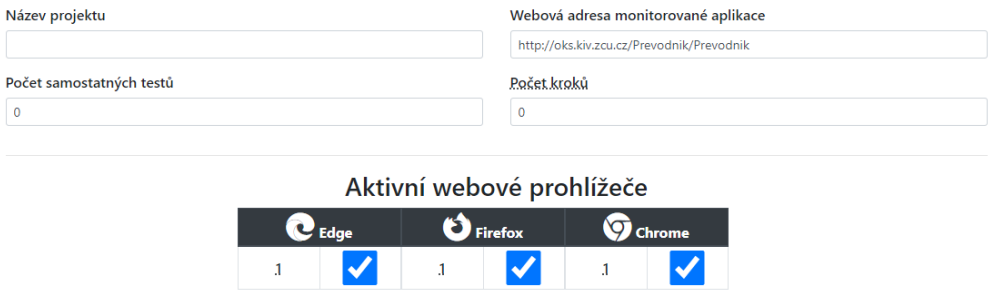


Obrázek A.2: Poptávka - přihlášení uživatele

ptávku vytvořenou, zobrazí se mu detail jeho uložené poptávky. Pokud chce, lze toto nastavení změnit přes tlačítko `Edit`.

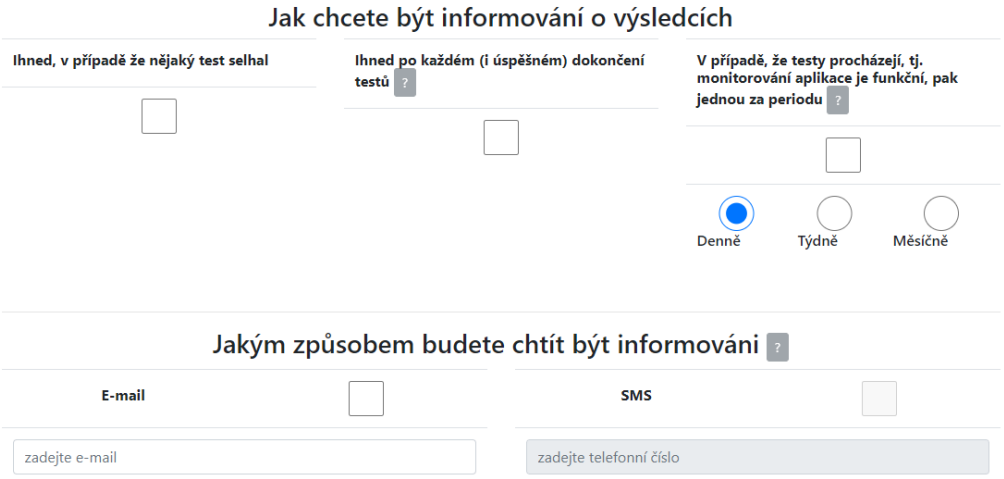
V základní části si uživatel musí zvolit název, pod jakým se poptávka uloží do databáze, adresu webové aplikace, kterou chce nechat testovat a počet testů a kroků. Počtem testů se rozumí počet jednotlivých testů, kdežto počtem se kroků je myšlen souhrný počet všech kroků ve všech testech dohromady (jedná se o kvalifikovaný odhad uživatele). První část zadávacího

formuláře je na obrázku A.3.



Obrázek A.3: Poptávka - základní část

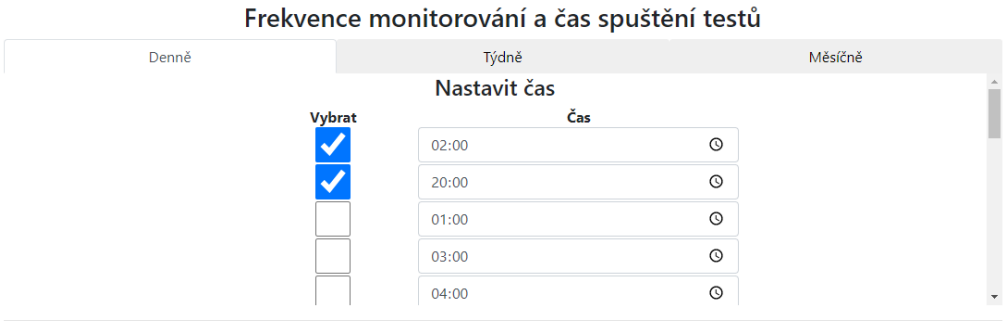
V další části formuláře se volí, jakým způsobem a jak často chce být uživatel informován o výsledcích testů. Tato často formuláře je zobrazena na obrázku A.4.



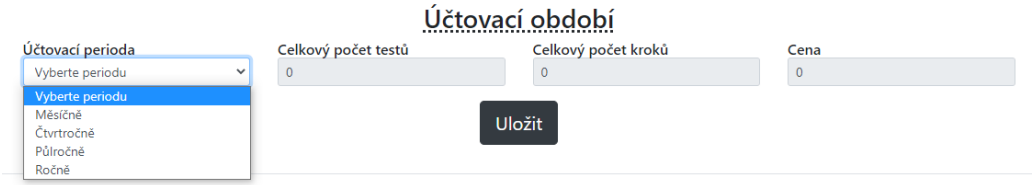
Obrázek A.4: Poptávka - způsob informování

Následující a nejdůležitější část nastavení poptávky je nastavení periody spouštění monitorování. Uživatel má na výběr denní, týdenní či měsíční periodu. Příklad pro nastavení denní periody je na obrázku A.5.

Poslední částí v nastavení poptávky je účtovací období, kde si uživatel může vybrat mezi měsíčním, čtvrtročním, půlročním nebo ročním. Formulář automaticky dopočte, kolik testů za zvolené období proběhne podle aktuálních hodnot a jaká by byla účtovaná cena.
Ukázka účtovacích období je na obrázku A.6.



Obrázek A.5: Poptávka - perioda monitorování



Obrázek A.6: Poptávka - účtovací období

Některé vstupní hodnoty je nutné zadat, nebo mají určitý rozsah povolených hodnot. Pokud zadané hodnoty nebudou splňovat tyto požadavky, formulář nebude uložen a uživatel je vyzván k nápravě těchto hodnot.

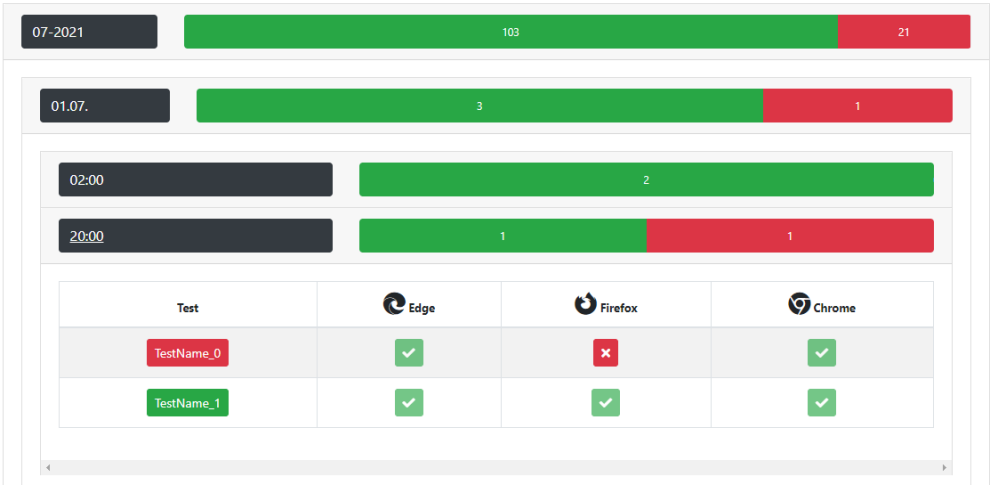
A.4 Prohlížení výsledků

Na výsledky se uživatel dostane z hlavního menu odkazem `Výsledky`. Výsledky jsou tříděny a seskupeny podle časových období. Jednotlivé časové úseky se dají rozklikat, až se uživatel dostane na konkrétní test. Příklad stromové struktury výsledků je zobrazen na obrázku A.7.

A.5 Administrace poptávek

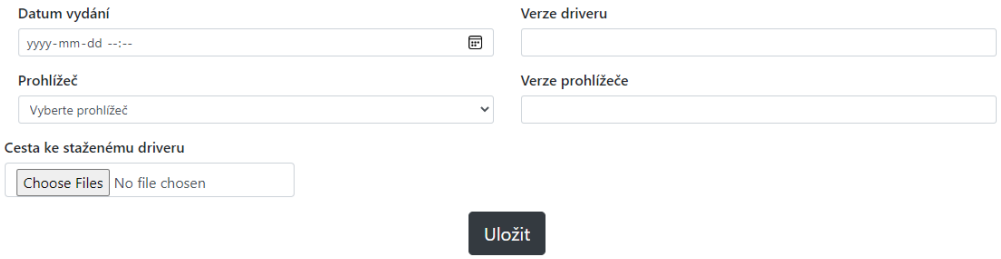
Pokud přistupuje uživatel k aplikaci pod administrátorským účtem, má zachovánu stejnou funkcionalitu jako běžný uživatel. Hlavní menu má ale rozšířené o `Přidat ovladač` a `Přehled poptávek`.

Přidat ovladač Pro přidání ovladače je nutné vyplnit informace ohledně verze a datumu vydání ovladače a typ a verzi prohlíže. Také se musí vybrat a nahrát požadovaný soubor. Webové drivery, které se nahrávají do systému,



Obrázek A.7: Zobrazení výsledků

musí být shodné se systémem na jakém aplikace běží. Aplikace je nasazena na linuxovém serveru, proto i nahrané webové drivery musí být pro linuxový operační systém.

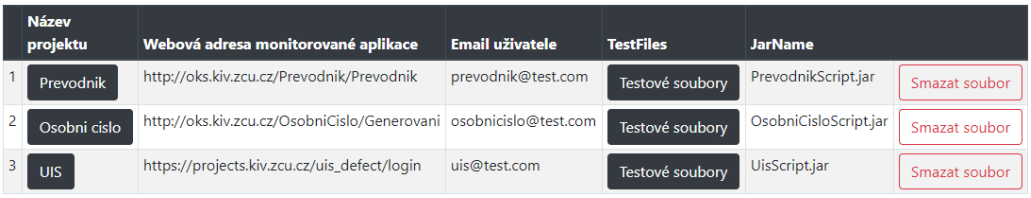


Obrázek A.8: Přidání webového ovladače

Přehled poptávek V tomto přehledu jsou zobrazeny všechny aktivní poptávky z databáze. Administrátor má tak možnost k nim nahrát soubory s monitorovacími skripty a výsledný `JAR` soubor pro spouštění těchto skriptů. Pro kontrolu může administrátor rozkliknout název dané poptávky a podívat se tak na její výsledky. Příklad zobrazených poptávek najdete na obrázku A.9.

A.5.1 Vytvoření monitorovacích skriptů

Do aplikace se nahrávají jak jednotlivé `.class` soubory s testy, tak výsledný `JAR` archiv, který testy spouští. Nahrávání může nějaký čas trvat, proto



	Název projektu	Webová adresa monitorované aplikace	Email uživatele	TestFiles	JarName	
1	Prevodnik	http://oks.kiv.zcu.cz/Prevodnik/Prevodnik	prevodnik@test.com	Testové soubory	PrevodnikScript.jar	Smazat soubor
2	Osobni cislo	http://oks.kiv.zcu.cz/OsobniCislo/Generovani	osobnicislo@test.com	Testové soubory	OsobniCisloScript.jar	Smazat soubor
3	UIS	https://projects.kiv.zcu.cz/uis_defect/login	uis@test.com	Testové soubory	UisScript.jar	Smazat soubor

Obrázek A.9: Přehled aktivních poptávek

prosím mějte strpení, zvlášť při nahrávání `JAR` souboru. Archiv musí splňovat formální požadavky uvedené v podkapitole 4.6.

Vzorový příklad monitorovacích skriptů s jednotlivými využívanými knihovnami je poskytnut se zdrojovým kódem. Pro přidání vlastních testů je možno vytvořit vlastní `JAR` dle následujících příkazů. Jméno hlavní třídy a adresářová struktura je zachována dle ukázky.

```
echo Main-Class: plain.Hlavni> man.txt
jar cmf man.txt [jar_name] plain/*.class
```

- `jar_name` – název výsledného archivu.

Po vytvoření archivu je nezbytné přidat využívané knihovny, což může být provedeno v aplikaci `Total Commander`. Vytvořený `JAR` otevřeme kombinací kláves `Ctrl + PgDn`. V druhém okně vybereme knihovny, které budeme přidávat a otevřeme je stejným způsobem. Obsah zvolených knihoven (adresáře, kromě metadat) pak nakopírujeme do výsledného `JAR` souboru.

Seznam využívaných knihoven.

- `selenium-server-standalone-3.141.59.jar` – obsahuje samostanou část pro podporu `Selenium WebDriver`.
- `msedge-selenium-tools-java-3.141.0.jar` – nutné pro podporu webového prohlížeče `Microsoft Edge`.

B Nastavení serveru

B.1 Server

Tato kapitola popisuje, jak nastavit server ve službě `Linode` pro aplikaci WebMonitoring. Uvedené kroky jsou nutné pro nastavení serveru a všech jeho využívaných služeb. Jednotlivé kroky jsou podrobně rozepsány a uvedené příkazy jsou pro linuxovou konzoli, pokud není uvedeno jinak. V případě, že aplikaci hodláte instalovat na vlastní server, tak se některé kroky mohou lišit. V případě nutosti si můžete ověřit funkčnost aktuálního serveru.

Přístup na server Pro prohlédnutí nastaveného serveru využijte následující účet.

```
User: admin
Password: WebMonitoring0admin
```

B.1.1 .NET 5

Webová aplikace je napsaná v jazyce `C#` s využitím frameworku `.NET`. Pro instalaci potřebných komponent následujte níže uvedené kroky.[2]

Přidání balíku `Microsoft` pro stažení požadovaného `Software Development Kit (SDK)`.

```
wget https://packages.microsoft.com/config/ubuntu
   /18.04/packages-microsoft-prod.deb -O packages-
   microsoft-prod.deb
sudo dpkg -i packages-microsoft-prod.deb
```

Instalace `SDK` pro `.NET 5`. Pokud se instaluje `SDK`, nemusí se instalovat zvlášť příslušné `run-time` prostředí.

```
sudo apt-get update; \
sudo apt-get install -y apt-transport-https && \
sudo apt-get update && \
sudo apt-get install -y dotnet-sdk-5.0
```

Správnost instalace můžete ověřit pomocí následujícího příkazu.

```
dotnet --version
```

B.1.2 NGINX

Instalace webového serveru `NGINX` slouží pro umožnění přístupu do webové aplikace z internetu.

```
sudo apt-get install nginx
```

Po instalaci je nezbytné službu zapnout a povolit standardní síťový port pro `HypertextTransfer Protocol`.

```
sudo service nginx start
sudo ufw allow 80
sudo ufw allow 443
```

Zda služba běží, můžete ověřit pomocí následujícího příkazu.

```
sudo service nginx status
```

Konfigurace Pro službu je nezbytné nastavit příslušné konfigurační soubory. Protože bylo potřeba upravit konfiguraci pro podporu `SSL` je kompletní konfigurace uvedena až v podkapitole B.1.3.

```
cd /etc/nginx/sites-available/default
```

Pro podporu pouze `HTTP` protokolu upravte soubor `nginx_default` dle následujících instrukcí.

```
server {
  listen 139.162.182.170:80;
  root /var/www/build;

  location / {
    proxy_pass http://localhost:5000;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_ugrade;
    proxy_set_header Connection keep-alive;
    proxy_set_header Host $host;
    proxy_cache_bypass $http_upgrade;

    proxy_set_header   X-Forwarded-For
       $proxy_add_x_forwarded_for;
    proxy_set_header   X-Forwarded-Proto $scheme;
  }
}
```

```
server {
  listen 80;
  listen [::]:80;

  return 444;
}
```

Po změně je dobré verifikovat, zda je soubor nastaven správně.

```
  sudo nginx -t
```

Po ověření je nutné službu restartovat.

```
  sudo nginx -s reload
```

B.1.3 HTTPS

Po větší zabezpečení a možnosti zasílat emailové zprávy je nutné povolit `HTTPS` protokol na serveru.

Nejprve je nutné zaregistrovat doménu, to můžete provést například na stránce `https://www.namecheap.com/`, kde byla zaregistrována doména `web-monitoring.net`, přes kterou je aplikace dostupná.

Pro `Linode` server je nutné vytvořit doménu podle zaregistrované adresy. Dále v konfiguraci domény je potřeba nastavit odpovídající `DNS` záznamy. Pro referenci se podívejte na obrázek B.1.[12]

Name Server	Subdomain	TTL
ns1.linode.com	web-monitoring.net	Default
ns2.linode.com	web-monitoring.net	Default
ns3.linode.com	web-monitoring.net	Default
ns4.linode.com	web-monitoring.net	Default
ns5.linode.com	web-monitoring.net	Default

Obrázek B.1: Nastavení `DNS` záznam

SSL certifikát Získat `SSL` certifikát lze několika způsoby a záleží na ostatních využívaných službách.

Na serveru ověřte stav a verzi `openssl`. Využíváte-li `Linode` server, je základní knihovna už nainstalována.

```
  openssl version -a
```

Zkontrolujte, jak je NGINX nainstalován.

```
nginx -V
```

Pokud instalace obsahuje --with-mail_ssl_module mělo by vše pokračovat v pořádku. V opačném případě by bylo vhodné server NGINX přeinstalovat přímo ze zdroje. K odebrání služby slouží následující příkaz.

```
sudo apt remove nginx
```

Když je NGINX správně nainstalován, pokračujte v instalaci certifikační autority CertBot.

```
sudo apt update
sudp apt install certbot python3-certbot-nginx
```

Nainstalujte CertBot pro NGINX server.[4]

```
sudo certbot --nginx
```

Dále je potřeba upravit nastavení serveru v souboru.

```
/etc/nginx/nginx.conf
```

V tomto souboru v části http zakomentujte řádku s nastavením a zajistěte načtení vlastních konfirgurací.

```
include /etc/nginx/conf.d/*.conf;
#include /etc/nginx/sites-enabled/*;
```

Následně v adresáři s konfiguracemi vytvořte vlastní konfigurační soubor.

```
cd /etc/nginx/conf.d
touch web-monitoring.net.conf
```

Obsah souboru určuje nastavenou IP adresu serveru, doménu a adresář s publikovanou aplikací. Další konfigurace slouží k nastavení NGINX jako reverzní proxy server.

```
server {
  listen 139.162.180.141:443 ssl;
  # listen [::]:80 default_server;
  server_name web-monitoring.net;
  root /var/www/build;

  client_max_body_size 100M;
```

```
  location / {
    proxy_pass http://localhost:5000;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection keep-alive;
    proxy_set_header Host $host;

    proxy_cache_bypass $http_upgrade;

    proxy_set_header X-Forwarded-For
       $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;

    proxy_set_header X-SSL-CERT
       $ssl_client_escaped_cert;

    proxy_connect_timeout       240;
    proxy_send_timeout          240;
    proxy_read_timeout          240;
    send_timeout                240;
  }

ssl_certificate /etc/letsencrypt/live/web-monitoring
   .net/fullchain.pem; # managed by Certbot
ssl_certificate_key /etc/letsencrypt/live/web-
   monitoring.net/privkey.pem; # managed by Certbot
}
server {
  client_max_body_size 100M;

  if ($host = web-monitoring.net) {
    return 301 https://$host$request_uri;
  } # managed by Certbot

  listen 139.162.180.141:80;
  server_name web-monitoring.net;
  return 404; # managed by Certbot
}
```

Po uložení konfiguračního souboru si můžete ověřit jeho správnost a ná-

sledně službu restartovat s novým nastavením.

```
sudo nginx -t
sudo nginx -s reload
```

Od tohoto momentu je webová aplikace přístupná přes adresu `https://web-monitoring.net`. Zda certifikát funguje správně, je také možno otestovat například přes `https://www.ssllabs.com/ssltest/analyze.html?d=web-monitoring.net`.

Pro ověření automatické obnovy certifikátu spusťte následující příkaz.

```
sudo certbot renew --dry-run
```

I když byl certifikát takto vytvořen a nastaven, v některých případech se stává, že je uživatel upozorněn před vstupem na tuto stránku.

B.1.4 Email

Webová aplikace může posílat emaily o výsledcích monitorování. Pro ty je potřeba otevřít náležité porty.

```
sudo ufw allow 587
sudo ufw allow 465
```

Na `Linode` serveru je nutné povolit posílání přes `SMTP`. To je v defaultním nastavení zakázáno, a je nutno zažádat uživatelskou podporu pro její zpřístupnění pro konkrétní server.

B.1.5 JAVA

Aplikace spouští existující `JAR` programy, proto je potřeba nainstalovat na server `JDK` verze.

V době psaní práce je aktuální defaultní `JDK` pro `LTS` verze 11.

```
sudo apt install default-jdk
```

Pro nainstalování konkrétní verze využijte následující příkaz.

```
sudo apt install openjdk-11-jdk
```

Po instalaci přidejte adresu nainstalovaného `JDK` do proměnné `JAVA_HOME`.

```
export JAVA_HOME=/var/lib/jvm/java-11-openjdk-
   amd64

echo $JAVA_HOME
```

Nainstalované JDK je také nutné přidat do proměnné PATH.

```
export PATH=$PATH:$JAVA_HOME/bin

echo $PATH
```

B.1.6 Webové prohlížeče

Pro správný chod aplikace je také potřeba nainstalovat požadované webové prohlížeče. Monitorovací skripty jsou spouštěny na jednotlivých prohlížečích a testovací kroky jsou postupně vykonávány. Z tohoto důvodu je potřeba mít k dispozici grafické rozhraní, i když je jeho zobrazení při monitoringu vypnuto.

```
sudo apt-get update && sudo apt-get upgrade
sudo apt-get install xfce4
sudo startxfce4
```

Pro instalaci prohlížečů následujte níže uvedené kroky. Jsou uvedené pouze příkazy pro prohlížeče, se kterými aplikace výsledně pracuje.

Chrome

```
wget https://dl.google.com/linux/direct/google-
   chrome-stable_current_amd64.deb
sudo apt install ./google-chrome-
   stable_current_amd64.deb
google-chrome --version
```

Firefox

```
sudo apt-get update
sudo apt install firefox
firefox --version
```

Edge

```
sudo apt update
sudo apt install software-properties-common apt-
   transport-https wget
wget -q https://packages.microsoft.com/keys/
   microsoft.asc -O- | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://
   packages.microsoft.com/repos/edge stable main"
sudo apt install microsoft-edge-dev
```

```
microsoft-edge --version
```

B.1.7 Webové ovladače

Ovladače se nachází v adresáři `/var/www/Drivers` a po ručním přidání je dobré ověřit, že mají práva na spuštění. V případě problémů zkontrolujte práva a upravte je následujícím příkazem.

```
chmod 777 [driver]
```

B.2 Databáze

Postup pro nastavení a instalaci databáze `SQL Server` na `Ubuntu 18.04`. Minimální požadavky na server jsou `2 GB` operační paměti. Pro instalaci budete potřebovat administrátorská práva.

B.2.1 Instalace

Pro nainstalování balíku `mssql-server` postupujte dle následujících kroků v terminálu na linuxovém serveru.[15]

Nejprve importujte GPG klíč pro veřejné úložiště.

```
wget -qO- https://packages.microsoft.com/keys/
   microsoft.asc | sudo apt-key add -
```

Zaregistrujte `Microsoft SQL Server Ubuntu` úložiště pro `SQL Server 2019`.

```
sudo add-apt-repository "$(wget -qO- https://
   packages.microsoft.com/config/ubuntu/18.04/
   mssql-server-2019.list)"
```

Stáhněte a nainstalujte `SQL Server`.

```
sudo apt-get update
sudo apt-get install -y mssql-server
```

Po instalaci balíku spusťte `mssql-conf setup` pro nastavení SA hesla a používané edice. Zvolte edici `SQL Server - Express`.

```
sudo /opt/mssql/bin/mssql-conf setup
```

Jakmile je konfigurace dokončena, ověřte, že služba běží.

```
systemctl status mssql-server --no-pager
```

Pro povolení vzdáleného přístupu je nutné povolit port `1433`.

```
sudo ufw enable
sudo ufw allow 1433
```

V případě, že chcete ovládat `SQL Server` z konzole, je nutné importovat pomocný balík.

```
curl https://packages.microsoft.com/keys/microsoft
   .asc | sudo apt-key add -
```

Registrovat `Microsoft` repozitář pro `Ubuntu`.

```
curl https://packages.microsoft.com/config/ubuntu
   /18.04/prod.list | sudo tee /etc/apt/sources.
   list.d/msprod.list
```

Uvedeným příkazem pak instalaci spustíte.

```
sudo apt-get update
sudo apt-get install mssql-tools unixodbc-dev
```

Proměnou `PATH` je pak třeba rozšířit o adresář s pomocnými nástroji.

```
echo 'export PATH="$PATH:/opt/mssql-tools/bin"' >>
    ~/.bash_profile
```

Následně už zbývá jen vytvořit účet, pod kterým se bude přistupovat do databáze.

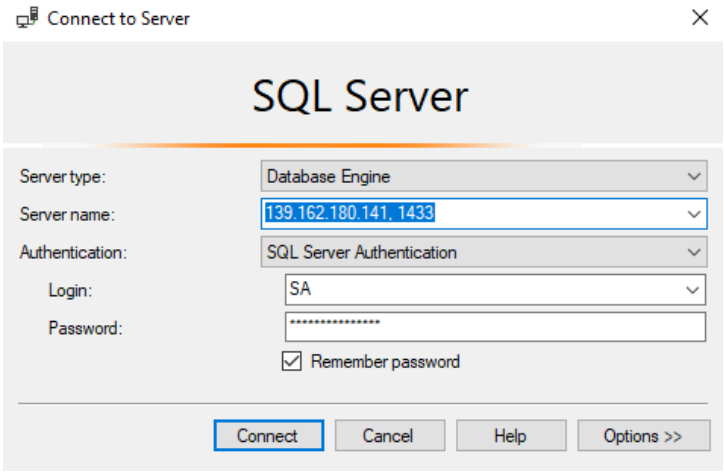
```
sqlcmd -S localhost -U SA -P 'Express0admin'
```

Nezapomeňte také povolit porty pro vzdálený přístup, budete-li k databázi přistupovat z jiného zařízení.

```
sudo ufw allow from any to any port 1433 proto tcp
sudo ufw allow from any to any port 1434 proto tcp
```

Na obrázku B.2 je uveden příklad připojení do databáze přes program `SQL Server Management Studio` verze 18. Pro připojení do aktuální databáze využijte následující účet.

```
User: SA
Password: Express0admin
```



Obrázek B.2: Připojení do databáze

B.2.2 Import dat

Data jsou zazálohována do souboru, ze kterého je možné databázi obnovit, nebo naimportovat do nové. Soubor se zálohou nahrajte do databáze a poté spusťte obnovu přes `SSMS`. Během obnovy je potřeba přepsat některé soubory.

```
scp [source_address] root@139.162.180.141:~/
   WebMonitoring.bak
```

Pokud vytváříte vlastní databázi, je nutné přidat roli `Admin` do tabulky `AspNetRoles`. Následně vybranému uživateli přidejte tuto roli v tabulce `AspNetUserRoles`.

B.2.3 Připojení databáze

Ve webové aplikaci se připojení do databáze řeší pomocí proměnné `ConnectionStrings`, která je uložena v souboru `appsetings.json`. Uvádí se dva parametry. První slouží k připojení databáze s daty a druhý je využit pro uživatelskou databázi.

Příklad připojení do lokální databáze.

```
 "ConnectionStrings": {
  "WebMonitoring": "Server=(localdb)\\mssqllocaldb
     ;Database=WebMonitoring;Trusted_Connection=
     True;MultipleActiveResultSets=true",
```

```
    "AuthContextConnection": "Server=(localdb)\\
      mssqllocaldb;Database=WebMonitoring;
      Trusted_Connection=True;
      MultipleActiveResultSets=true"
  }
```

Příklad připojení do databáze na serveru.

```
  "ConnectionStrings": {
  "WebMonitoring": "Server=139.162.180.141,1433;
    Database=WebMonitoring;User Id=SA;Password=
    Express0admin",
  "AuthContextConnection": "Server=139.162.180.141,1
    433;Database=WebMonitoring;User Id=SA;Password=
    Express0admin"
}
```

B.3 Nasazení aplikace

Webová aplikace předpokládá určitou strukturu adresářů, kam se zdrojový kód publikuje.

B.3.1 Příprava adresářů

Na linuxovém serveru je potřeba vytvořit pomocné adresáře, dle následujícího příkazu.

```
  cd /var/www
  mkdir Drivers
  mkdir TestFiles
  mkdir build
  mkdir WebMonitoring
```

B.3.2 Zdrojový kód

Pro nahrání zdrojového kódu na server je nejjednodušší použít `Git`. Zkontrolujte tedy nejprve zda je `Git` nainstalován.

```
  git --version
```

Do adresáře `WebMonitoring` nahrajte zdrojový kód aplikace poskytnutý s textem této práce. Máte-li připravený zdrojový kód, je zapotřebí aplikaci publikovat. To provedete uvedeným příkazem.

```
cd WebMonitoring
sudo dotnet publish --output "/var/www/build" --
   configuration release
```

Vytvořte novou službu pro zapnutí webové aplikace.

```
cd /etc/systemd/system
sudo touch WebMonitoring.service
```

Obsah souboru `WebMonitoring.service`.

```
[Unit]
Description=WebMonitoring description

[Service]
WorkingDirectory=/var/www/build
ExecStart=/usr/bin/dotnet /var/www/build/
   WebMonitoring.dll
Restart=always
RestartSec=10
KillSignal=SIGINT
SyslogIdentifier=webmonitoring
User=root
Environment=ASPNETCORE_ENVIRONMENT=Development
Environment=DOTNET_PRINT_TELEMETRY_MESSAGE=false

[Install]
WantedBy=multi-user.target
```

Zaregistrujte službu a zapněte ji následujícími příkazy. Také můžete zkontrolovat, v jakém stavu se služba nachází.

```
sudo systemctl enable WebMonitoring.service
sudo systemctl start WebMonitoring.service
sudo systemctl status WebMonitoring.service
```

Pro opakované nasazení aplikace je vhodné postupovat podle následujícího postupu.

```
cd /var/www/build
sudo rm -r *
```

```
cd /var/www/WebMonitoring

sudo dotnet publish --output "/var/www/build" --
   configuration release

sudo systemctl daemon-reload
sudo systemctl restart WebMonitoring.service
```

Nejdůležitější část těchto příkazů po nasazení nové verze kódu je restartování dané služby. Pokud dojde k nějaké poruše, může restartování služby obnovit funkčnost aplikace.