FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání diplomové práce

Název:	Testovací prostředí carsharingového systému
Student:	Bc. Jan Levý
Vedoucí:	Ing. Filip Ravas
Studijní program:	Informatika
Obor / specializace:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2023/2024

Pokyny pro vypracování

1. V projektu Uniqway se seznamte a analyzujte aktuální způsob testování backendové aplikace před nasazením do produkčního prostředí.
2. Navrněte řešení přidání nového prostředí, které bude sloužit pro účely testování základní funkcionality aplikace v cloudu, před nasazením do stabilních prostředí. Toto prostředí musí splňovat požadavky jako jsou možnost spuštění více testú paralelně, po dokončení běhu testu se potřebná infrastruktura automaticky smaže.
3. Navrhněte implementaci nutných změn v systému a zaintegrujte možnost spuštění testu v automatizovaném procesu sestavování aplikace a jejího postupného nasazení až do produkčního prostředí.
4. Zhodnoťte použitelnost implementovaného řešení z pohledu vývojáře a navrhněte další kroky které povedou k zlepšením jako například zkrácení běhu testu, snížení ceny infrastruktury potřebné pro dokončení testu, nebo zvětšení pokrytí testované plochy systému.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 16. ledna 2023 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Testovací prostředí carsharingového systému

Bc. Jan Levý

Katedra softwarového inženýrství
Vedoucí práce: Ing. Filip Ravas

26. dubna 2023

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací. Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona a to na dobu určitou do skončení trvání ochrany dle Smlouvy. Nakládání s předloženou prací se řídí Smlouvou o spolupráci uzavřenou v návaznosti na spolupráci mezi Českým vysokým učením technickým v Praze a společností ŠKODA AUTO a.s. a Smart City Lab s.r.o na výzkumném projektu "CarSharing pro vysokoškolské studenty", uveřejněné v registru smluv na adrese https://smlouvy.gov.cz/smlouva/5973503. Jsem vázán Smlouvou o zachování mlčenlivosti, že nezpřístupním třetí osobě důvěrné informace, které jsem při své práci na Projektu získal.

V Praze dne 26. dubna 2023 . . . . . .

Odkaz na tuto práci

Levý, Jan. *Testovací prostředí carsharingového systému.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2023.

Abstrakt

V této diplomové práci je popsán návrh a implementace procesu, pomocí kterého je možné dynamicky vytvořit nové testovací prostředí pro backendovou aplikaci carsharingového projektu Uniqway. Výsledné řešení zároveň umožňuje spuštění automatizovaných testů a následné odstranění vytvořeného prostředí v momentě, kdy již není využíváno. Tento proces umožňuje správné otestovaní backendové aplikace projektu Uniqway.

Celé řešení je vytvořeno v prostředí Amazon Web Services a nástrojem JetBrains Teamcity, který slouží ke kontinuální integraci a kontinuálnímu nasazení aplikace.

Klíčová slova Uniqway, testovací prostředí, AWS, Amazon Web Services, JetBrains Teamcity

Abstract

This diploma thesis describes the design and implementation of a process by which it is possible to dynamically create a new test environment for the backend application of the Uniqway carsharing project. The resulting solution also enables the launch of automated tests and the subsequent removal of the

created environment at the moment when it is no longer used. This process allows the backend application of the Uniqway project to be properly tested.

The entire solution is created in the Amazon Web Services environment and the JetBrains Teamcity continuous integration and continuous deployment tool.

Keywords Uniqway, test environment, AWS, Amazon Web Services, JetBrains Teamcity

Obsah

Seznam obrázků

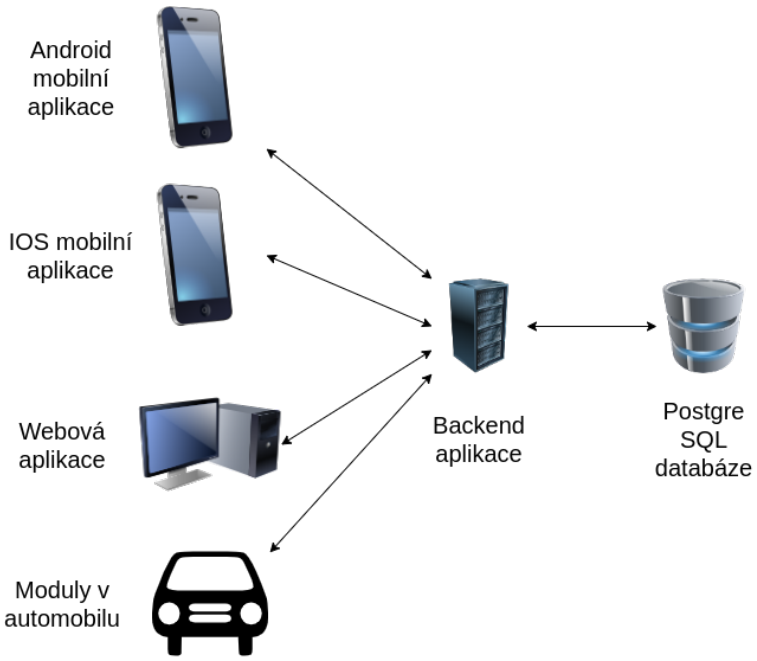
Seznam tabulek

Úvod

Při vývoji jakéhokoliv software nastává potřeba jeho otestování, tedy kontrola, zda daný software funguje správně. Může se například jednat o kontrolu samotné funkcionality vytvořeného díla a nebo o testování, zda daný software zvládne zpracovat požadovanou zátěž. V této diplomové práci se budu zabývat analýzou existujícího procesu testování backendové aplikace studentského projektu Uniqway, která momentálně využívá pouze stagingové a produkční prostředí. Chybí zde tedy prostředí, které by bylo určené pro testování této aplikace. V této práci se zabývám návrhem a implementací procesu, který umožňuje dynamické vytvoření nového testovacího prostředí a jeho následné odstranění v momentě, kdy již není využíváno.

Abychom lépe pochopili celou problematiku této diplomové práce, tak si musíme představit studentský projekt Uniqway. „Uniqway je první český carsharing pro studenty a zaměstnance vysokých škol.“ [1]. Využívat studentský projekt Uniqway tedy mohou všichni studenti a zaměstnanci vysokých škol na území České republiky. Tito uživatelé si mohou zapůjčit automobil pro své osobní potřeby. Celý proces zapůjčení vozidla probíhá v mobilní aplikaci, pomocí které si daný uživatel může vybraný automobil zarezervovat, vypůjčit a následně vrátit.

Projekt Uniqway se skládá z několika komponent. Hlavní komponentou je aplikace backendu, která je propojena s databází a k níž se připojují mobilní Android a IOS aplikace, webová aplikace a moduly pro sběr dat z automobilů. Propojení jednotlivých komponent je zobrazeno na obrázku 0.1. V této diplomové práci se budu zabývat vytvořením testovacího prostředí právě pro aplikaci backendu.



Obrázek 0.1: Zobrazení architektury projektu Uniqway

Kapitola 1

Uniqway backend aplikace

Cílem této diplomové práce je analyzovat, navrhnout a implementovat testovací prostředí pro webovou backend aplikaci Uniqway. Abych mohl správně nastavit proces vytváření testovacího prostředí a následné spuštění automatizovaných testů, tak nejprve musím pořádně porozumět všem použitým technologiím a procesům v této aplikaci. Této problematice se budu věnovat v následujících sekcích.

1.1 Aktuální stav testování a nasazení Uniqway backend aplikace

V této části se budu zabývat tím, jak je Uniqway backend aplikace testována (sekce 1.1.1) a jaká jsou využívána prostředí pro její běh (sekce 1.1.2). Zároveň popíšu proces nasazení nové verze aplikace do produkčního prostředí (sekce 1.1.3).

1.1.1 Unit a API testy

Uniqway backend aplikace momentálně využívá dva druhy testů. Prvním z nich jsou takzvané unit testy. Jak uvádí zdroj [2], tak tyto testy se využívají ke kontrole malých částí kódu. Většinou se využívají k otestování pouze jedné metody, nekontrolují tedy správnost chování celé aplikace, ale pouze její části. K jejich spuštění dochází relativně často a to zpravidla několikrát denně. Zároveň unit testy jsou automaticky spuštěny při nahrání změn do sdíleného repozitáře verzovacího systému GIT a každý vývojář si je může spustit lokálně na svém počítači.

Jako druhý typ testů využívá backend aplikace API testy. „API testing is a type of software testing that analyzes an application program interface (API) to verify that it fulfills its expected functionality, security, performance and reliability.“[3] Tyto testy tedy kontrolují správné chování aplikace z pohledu uživatele a testují ji jako celek. K jejich spuštění nedochází tak často, jako u

unit testů, ale zpravidla pouze při nahrání nové verze aplikace do testovacího prostředí.

1.1.2 Existující prostředí pro běh Uniqway backend aplikace

Momentálně existují dvě prostředí - stagingové a produkční prostředí. „A staging environment or staging site is a copy of your live website and is the last step in the deployment process before changes are deployed to your live website.“[4] Stagingové prostředí je tedy využívané ke spuštění posledních testů předtím, než je daná verze aplikace nasazena do produkčního prostředí.

Jelikož Uniqway backend projekt nemá testovací prostředí, tak se využívá stagingové prostředí zároveň jako testovací. Tento krok s sebou nese mnohá rizika a nevýhody. Například se v tomto prostředí může nacházet verze aplikace, která není určena pro spuštění v produkci, ale vývojář, který do tohoto prostředí danou verzi nasadil, si pouze potřeboval otestovat určitou funkcionalitu aplikace. Správně by se pro toto testování mělo využívat testovací prostředí, které bohužel momentálně neexistuje.

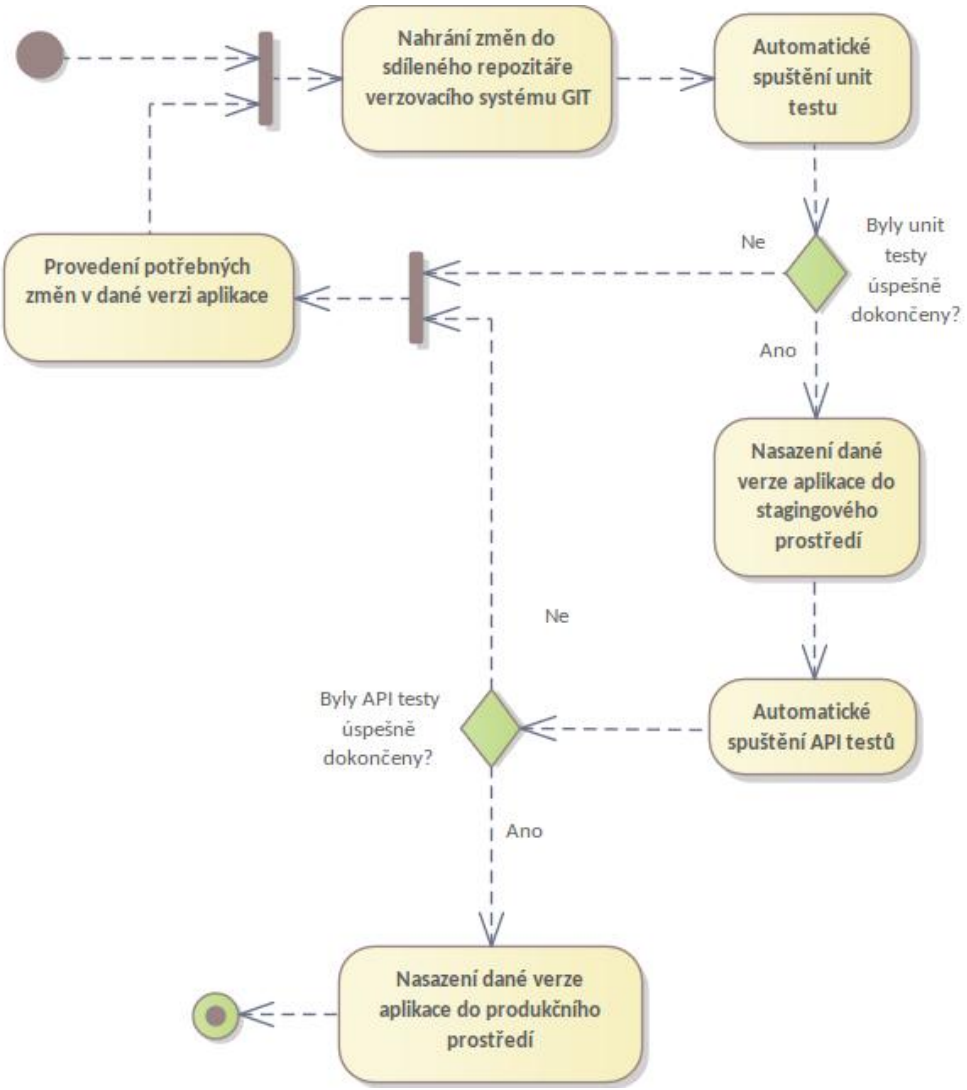
1.1.3 Momentální proces nasazení Uniqway backend aplikace do produkčního prostředí

Aby bylo možné nasadit danou verzi aplikace do produkčního prostředí, musí nejprve úspěšně splnit všechny testy popsané v sekci 1.1.1. Ke spuštění unit testů dochází automaticky při nahrání změn do sdíleného repozitáře verzovacího systému GIT. API testy jsou spuštěny automaticky po nahrání nové verze aplikace do stagingového prostředí. Pokud jsou tedy obě sady těchto testů úspěšně splněny, může být daná verze aplikace nasazena do produkčního prostředí, kde jsou opět automaticky spuštěny API testy. Celý tento proces je graficky znázorněn na obrázku 1.1.

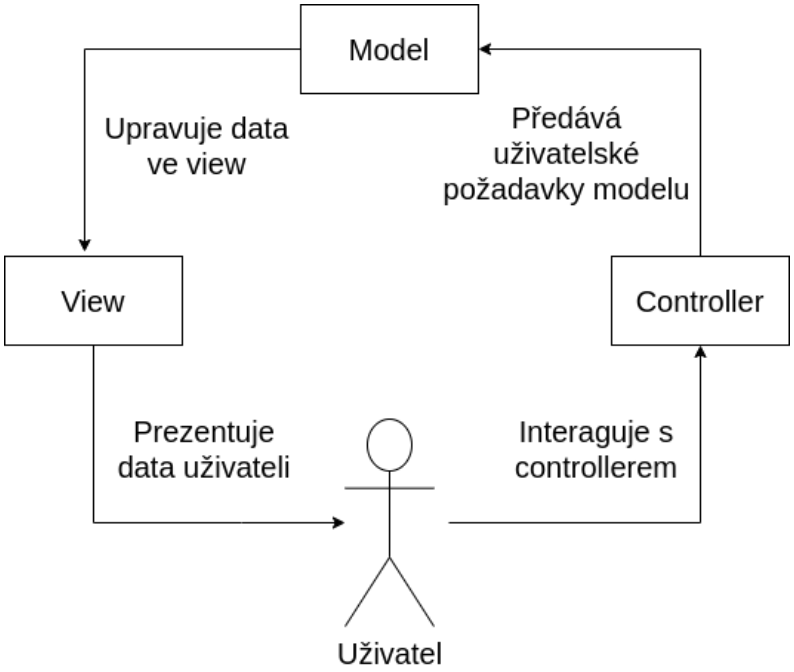
1.2 Java Play

Celá backend aplikace Uniqway je napsána pomocí jazyka Java a frameworku Play. Jak uvádí zdroj [5], Play framework se využívá k vytváření webových aplikací za pomocí architektury model-view-controller. Tento architektonický vzor je tedy rozdělen do tří částí:

- Model - Zde se nachází hlavní logika aplikace.
- View - Upravuje data získaná z modelu do vhodné podoby, aby mohla být prezentována uživateli.
- Controller - Umožňuje uživateli interagovat s aplikací.



Obrázek 1.1: Proces nasazení nové verze aplikace do produkčního prostředí



Obrázek 1.2: Architektura model-view-controller

Grafické znázornění architektury model-view-controller je zobrazeno na obrázku 1.2.

Aplikace napsaná pomocí frameworku Play využívá při svém spuštění konfigurační soubor *application.conf*. Tento soubor se musí nacházet ve složce *conf* v kořenovém adresáři aplikace. V konfiguračním souboru je definováno nastavení aplikace, například nastavené připojení k databázi a mail serverům.

1.3 Postgre SQL

K perzistentnímu uložení dat využívá Uniqway backend aplikace databázi Postgre SQL. „PostgreSQL is a powerful, open source object-relational database system with over 35 years of active development.“[6] Jedná se tedy o objektově relační databázi. Jsou zde uložena všechna potřebná data pro správné fungování projektu Uniqway. Nachází se zde například uživatelská data, záznamy o provedených jízdách a data o automobilech.

1.4 Amazon Web Services (AWS)

Uniqway backend aplikace pro svůj běh využívá zdroje a služby poskytované Amazon Web Services. Právě těmto webovým službám se budu věnovat v

následujících sekcích, jelikož při vytváření testovacího prostředí budu muset vytvořit veškeré zdroje, které jsou potřebné pro správný běh aplikace.

1.4.1 Amazon Resource Name (ARN)

„Amazon Resource Names (ARNs) uniquely identify AWS resources.“[7] Amazon Resource Name tedy představuje unikátní identifikátor zdroje v AWS. Každý zdroj v AWS má tedy svůj vlastní ARN, pomocí kterého je následně možné s daným zdrojem manipulovat.

1.4.2 Amazon Elastic Container Service (ECS)

„Amazon Elastic Container Service (Amazon ECS) is a fully managed container orchestration service that helps you easily deploy, manage, and scale containerized applications.“[8] Amazon ECS tedy umožňuje provozovat kontejnerizované aplikace. Backendová aplikace Uniqway využívá technologii Docker. Pomocí této technologie je z Uniqway aplikace vytvořen balíček, který je následně spuštěn v Amazon ECS.

1.4.3 Amazon Code Deploy

„CodeDeploy is a deployment service that automates application deployments to Amazon EC2 instances, on-premises instances, serverless Lambda functions, or Amazon ECS services.“[9] Jelikož backendová aplikace Uniqway je spuštěna v prostředí Amazon ECS (viz sekce 1.4.2), tak projekt Uniqway využívá Amazon Code Deploy k automatizaci procesu nasazení aplikace do tohoto prostředí.

1.4.4 Amazon Realtional Database Service (RDS)

„Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the AWS Cloud.“[10] Backendový projekt Uniqway využívá k perzistenímu uložení dat relační databázi Postgre SQL, viz sekce 1.3. Tuto databázi provozuje právě pomocí služby Amazon RDS.

1.4.5 Amazon Elastic Load Balancing (ELB)

„Elastic Load Balancing (ELB) is a fully managed load balancing service that automatically distributes incoming application traffic to multiple targets and virtual appliances across AWS and on-premises resources.“[11] Služba Amazon Elastic Loadbalancing umožňuje distribuovat příchozí požadavky mezi více cílů. Backendový projekt Uniqway využívá tuto službu jako vstupní bránu ke své aplikaci, jelikož vytvořený load balancer dokáže přijímat požadavky z veřejného internetu.

1.4.6 Amazon Target Group

„Target groups route requests to one or more registered targets, such as EC2 instances, using the protocol and port number that you specify.“[12] Target group přijímá požadavky z daného load balanceru a následně je přeposílá do požadovaných cílů, kterých může být více. Uniqway projekt využívá target group k přeposílání obdržených požadavků z load balanceru do backendové aplikace.

1.4.7 Amazon Virtual Private Cloud (VPC)

„Amazon Virtual Private Cloud (Amazon VPC) enables you to launch AWS resources into a virtual network that you've defined.“[13] Projekt Uniqway využívá Amazon VPC k vytvoření své privátní sítě, ve které má následně vytvořené ostatní AWS zdroje. K těmto zdrojům mohou následně přistupovat ostatní zdroje a uživatelé, pouze pokud se nacházejí také ve stejném Amazon VPC.

1.4.8 Amazon Security Groups

„In AWS VPCs, AWS Security Groups act as virtual firewalls, controlling the traffic for one or more stacks (an instance or a set of instances).“[14] Amazon Security Groups tedy fungují jako firewall a kontrolují přístup k jednotlivým AWS zdrojům. Projekt Uniqway využívá například security group, která řídí přístup k vytvořenému load balanceru, viz sekce 1.4.5.

1.4.9 Amazon S3

„Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance.“[15] Amazon S3 je tedy služba, která poskytuje bezpečné uložení dat. Projekt Uniqway využívá tuto službu například k uložení naskenovaných dokumentů svých uživatelů, jako jsou občanské, řidičské a studentské průkazy.

1.4.10 Amazon Secrets Manager

„Secrets Manager enables you to replace hardcoded credentials in your code, including passwords, with an API call to Secrets Manager to retrieve the secret programmatically.“[16] Amazon Secrets Manager tedy poskytuje úložiště pro citlivá data, která je následně možné přečíst pomocí API požadavku přímo v kódu aplikace. Projekt Uniqway využívá této služby například k uložení přihlašovacích údajů k databázi.

1.5 Terraform

„HashiCorp Terraform is an infrastructure as code tool that lets you define both cloud and on-prem resources in human-readable configuration files that you can version, reuse, and share.“[17] Terraform je tedy nástroj, pomocí kterého lze definovat a konfigurovat požadované zdroje. Projekt Uniqway využívá Terraform k nastavení stagingového a produkčního prostředí.

1.6 JetBrains Teamcity

„JetBrains TeamCity is a powerful and user-friendly Continuous Integration and Deployment server that works out of the box.“[18] JetBrains Teamcity je nástroj, který umožňuje kontinuální integraci a kontinuální nasazení vybrané verze aplikace. Přesně k tomuto účelu jej využívá i projekt Uniqway.

1.6.1 Kotlin DSL

Při použití nástroje Jetbrains Teamcity je potřeba uchovávat nastavení jednotlivých kroků. Jak uvádí zdroj [19], tak je možné k tomuto účelu využít jazyk Kotlin. Celé nastavení se poté nachází v předem definovaném adresáři a je možné jej verzovat společně se samotným kódem aplikace. Zároveň pokud dojde ke změně nastavení, tak JetBrains Teamcity tuto změnu detekuje a pokusí se ji automaticky provést.

Kapitola 2

Analýza nového testovacího prostředí

V této kapitole se budu věnovat analýze a návrhu vytváření nového testovacího prostředí, pomocí nichž poté budu implementovat tento proces.

2.1 Analýza uživatelských požadavků pro nové testovací prostředí

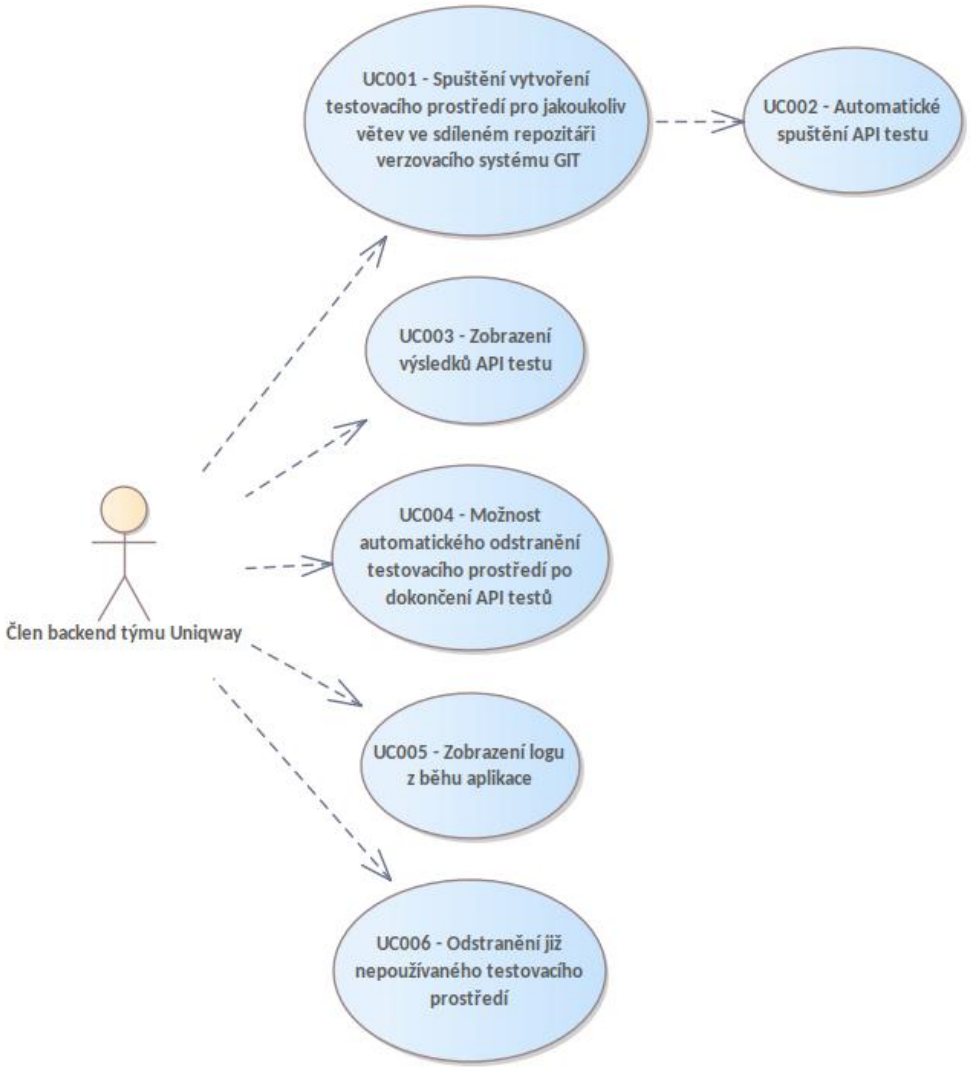
Tato sekce je věnována analýze uživatelských požadavků na nové testovací prostředí. Grafické znázornění těchto požadavků je zobrazené na obrázku 2.1.

2.1.1 Uživatelský požadavek UC001 - Spuštění vytvoření testovacího prostředí pro jakoukoliv větev ve sdíleném repozitáři verzovacího systému GIT

Testovací prostředí by mělo být možné dynamicky vytvořit pro jakoukoliv větev, která existuje ve sdíleném repozitáři verzovacího systému GIT. Při vytváření nového testovacího prostředí bude muset dojít k vytvoření všech zdrojů popsaných v kapitole 1.4. Tyto zdroje vždy musí obsahovat unikátní název, proto bude vhodné připojit ke každému jménu ještě číslo sestavení a název větve, pro které je toto prostředí vytvořeno.

Zároveň je důležité, aby možnost vytvoření nového testovacího prostředí měl každý člen backend vývojářského týmu. Každý člen tedy bude moci vytvořit libovolný počet testovacích prostředí pro libovolný počet větví existujících ve sdíleném repozitáři verzovacího systému GIT.

Backendový projekt Uniqway využívá aplikaci Teamcity (viz sekce 1.6) k zahájení sestavení nové verze backend aplikace a její následné nasazení do stagingového a produkčního prostředí. Bude tedy vhodné využít Teamcity i ke spuštění vytvoření nového testovacího prostředí.



Obrázek 2.1: Use case diagram

V rámci tohoto požadavku tedy dojde k vytvoření nové testovací databáze a její připravení na provoz v testovacím prostředí. Následně dojde k vytvoření všech potřebných zdrojů v Amazon Web Services a nasazení požadované verze aplikace do nově vytvořeného testovacího prostředí. Realizace tohoto use case je popsána v sekcích 3.3.1, 3.3.2, 3.3.3 a 3.3.4.

2.1.2 Uživatelský požadavek UC002 - Automatické spuštění API testů

Po dokončení všech potřebných kroků popsaných v sekci 2.1.1 dojde k automatickému spuštění API testů proti běžící backend aplikaci Uniqway v nově vytvořeném testovacím prostředí. Realizace tohoto use case je popsána v sekci 3.3.5.

2.1.3 Uživatelský požadavek UC003 - Zobrazení výsledků API testů

Každý člen backend týmu Uniqway by měl mít možnost zobrazení výsledků API testů spuštěných proti Uniqway backend aplikaci v daném testovacím prostředí. V případě, že API testy nebyly dokončeny úspěšně, tak by uživatel měl být schopen jednoduše dohledat testovací scénáře, které byly neúspěšné. Realizace tohoto use case je popsána v sekci 3.3.5.

2.1.4 Uživatelský požadavek UC004 - Možnost automatického odstranění testovacího prostředí po dokončení API testů

Při vytváření nového testovacího prostředí by měl mít každý člen backendového týmu Uniqway možnost zvolit, jestli má být dané testovací prostředí odstraněno po dokončení API testů. Realizace tohoto use case je popsána v sekcích 3.3.6, 3.3.7 a 3.3.8.

2.1.5 Uživatelský požadavek UC005 - Zobrazení logů z běhu aplikace

Každý člen backendového týmu Uniqway by měl mít možnost zobrazení logů z běhu aplikace v daném testovacím prostředí. Tento požadavek by měl být splněn i v případě, že již došlo k odstranění daného testovacího prostředí. Realizace tohoto use case je popsána v sekci 3.5.5.

2.1.6 Uživatelský požadavek UC006 - Odstranění již nepoužívaného testovacího prostředí

Každý člen backendového týmu Uniqway by měl mít možnost spustit odstranění již existujícího testovacího prostředí. Tento požadavek bude užitečný,

pokud při vytváření nového testovacího prostředí nebyla zvolena možnost automatického odstranění testovacího prostředí po dokončení API testů, která je popsána v sekci 2.1.4. Realizace tohoto use case je popsána v sekci 3.4.

2.2 Vytvoření databáze pro testovací prostředí - Amazon Aurora Serverless V2

Jelikož je možné, že bude existovat více testovacích prostředí zároveň, tak každé z těchto prostředí bude muset mít svou vlastní databázi. Backendový projekt Uniqway pro svoji databázi využívá Amazon Relational Database Service a to konrétně PostgreSQL server (viz 1.4.4). Tento databázový server má definovanou velikost a výpočetní výkon již při spuštění. Pro potřeby testovacího prostředí bych chtěl využít Amazon Aurora Serverless V2 pro PostgreSQL.

„Aurora Serverless v2 is an on-demand, auto-scaling feature designed to be a cost-effective approach to running intermittent or unpredictable workloads on Amazon Aurora. It automatically starts up, shuts down, and scales capacity up or down, as needed by your applications.“ [20] Jedná se tedy o databázi, která se umí automaticky škálovat podle aktuálního zatížení. Výhodou této databáze je, že se platí pouze za výkon, který opravdu poskytuje. Zároveň by měla být oproti čisté PostgreSQL databázi připravena k použití rychleji.

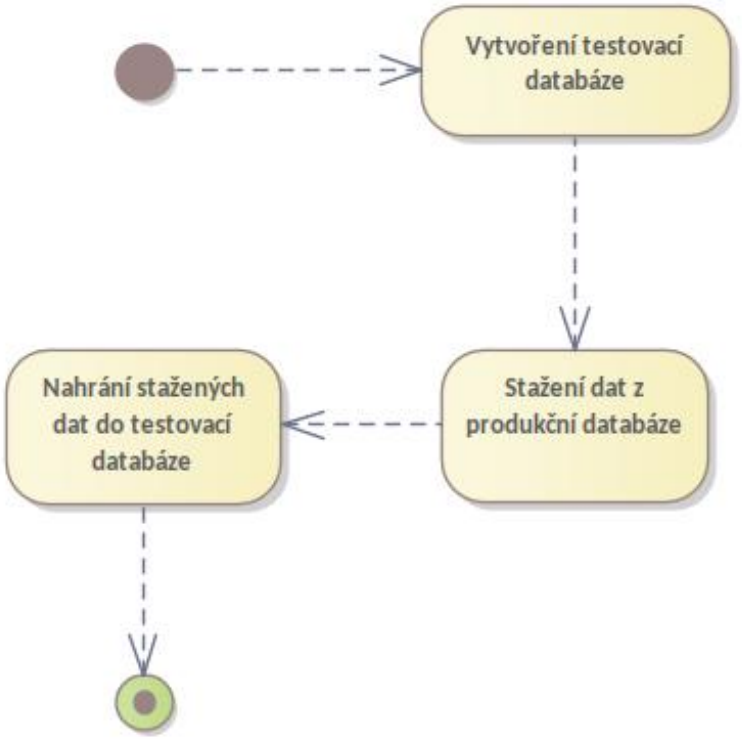
Po vytvoření testovací databáze bude potřeba tuto databázi naplnit daty tak, aby byla připravena k použití v testovacím prostředí. K tomuto účelu využiji data z produkční databáze, která nejprve stáhnu a následně je vložím do vytvořené testovací databáze. Stejný proces je použit při vytváření databáze ve stagingovém prostředí. Celý tento proces je zobrazen na obrázku 2.2.

2.3 Analýza životního cyklu testovacího prostředí

Celý životní cyklus testovacího prostředí bude začínat tím, že člen backendového týmu Uniqway nechá toto prostředí vytvořit pro danou větev ze sdíleného repozitáře verzovacího systému GIT. Zároveň si při této akci zvolí, jestli má být vytvořené testovací prostředí automaticky odstraněno po dokončení API testů.

Nejprve tedy dojde k vytvoření a připravení testovací databáze pro použití v daném testovacím prostředí (viz sekce 2.2). Po úspěšném nastavení testovací databáze budou vytvořeny veškeré zdroje potřebné pro správné spuštění dané verze Uniqway backend aplikace a následně bude tato aplikace nasazena do nově vytvořeného testovacího prostředí.

V tuto chvíli budou automaticky spuštěny API testy. Pokud si na začátku uživatel zvolil, že se má testovací prostředí automaticky odstranit, tak k této akci dojde po dokončení API testů.



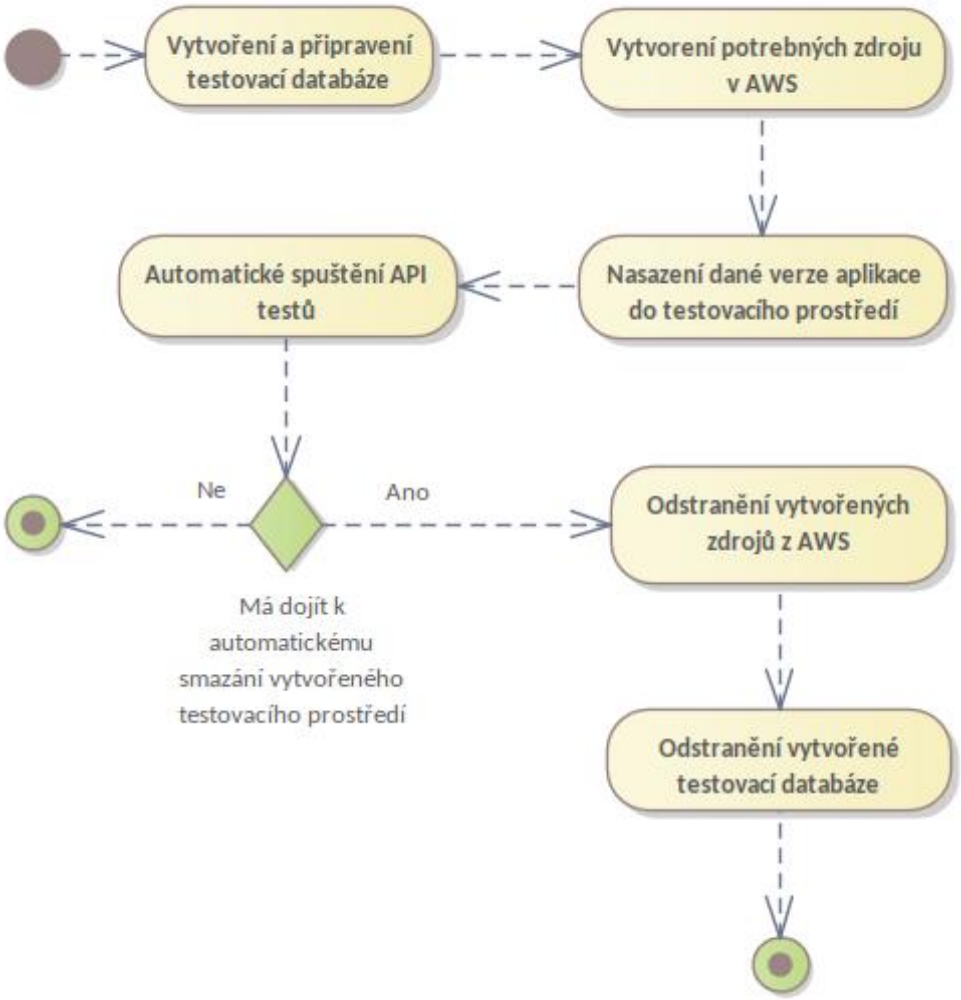
Obrázek 2.2: Analýza vytvoření testovací databáze

Pokud nedojde k automatickému odstranění testovacího prostředí, bude možné danou aplikaci dále testovat. Následně kdokoliv z backendového týmu bude mít možnost spustit odstranění tohoto prostředí. Touto akcí vytvořené testovací prostředí zaniká a s tím končí i jeho životní cyklus. Celý tento proces je graficky znázorněn na obrázku 2.3.

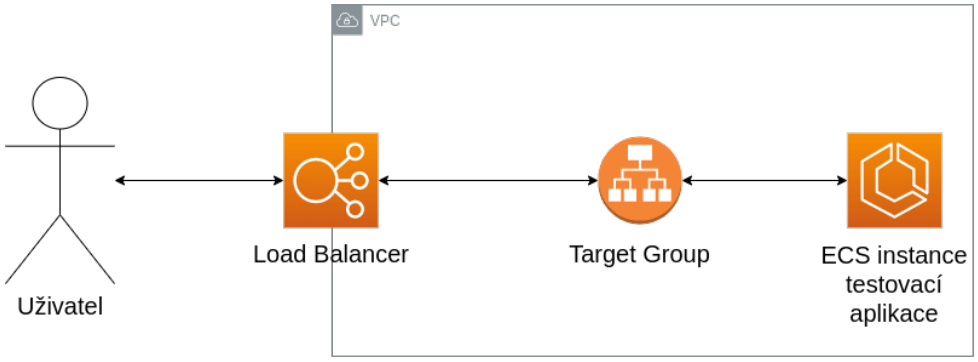
2.4 Zpřístupnění aplikace v testovacím prostředí

Aby bylo možné aplikaci v testovacím prostředí testovat, tak je potřeba, aby byla daná aplikace dostupná pomocí internetu. Backendový projekt Uniqway využívá pro svůj běh Amazon Elastic Container Service (viz sekce 1.4.2). Běžící instance aplikace pomocí této služby bohužel není přístupná z veřejného internetu.

Jeden z možných způsobů, jak tuto aplikaci zpřístupnit z veřejného internetu, je vytvořením load balanceru (viz sekce 1.4.5) a příslušné target groupy (viz sekce 1.4.6). Vytvořený load balancer i target groupa se budou muset nacházet ve stejném VPC (viz sekce 1.4.7) jako daná testovací aplikace. Load balancer bude mít přiřazenou veřejnou IP adresu, pomocí níž k němu bude



Obrázek 2.3: Životní cyklus testovacího prostředí



Obrázek 2.4: Naznačení možné komunikace s testovací aplikací

možné přistoupit. Následně bude load balancer přeposílat přijaté požadavky do dané target groupy a ta je poté odešle do příslušné aplikace v testovacím prostředí. Obrázek 2.4 zobrazuje popsaný průběh komunikace.

Kapitola 3

Implementace dynamického vytvoření testovacího prostředí

V této kapitole se budu zabývat praktickou částí této diplomové práce. Popíšu zde celý proces implementace dynamického vytvoření testovacího prostředí.

3.1 Testovací databáze

Každé nově vytvořené testovací prostředí využívá Amazon Aurora Serverless V2 s použitím PostgreSQL databázového enginu. Výhodami, které poskytuje použití tohoto typu databáze jsem se zabýval v sekci 2.2. Každé testovací prostředí je vytvořeno zároveň s vlastním databázovým clusterem a databázovou instancí, která slouží ke čtení a zapisování požadavků z a do databáze. Vytvoření nového databázového clusteru je popsáno v sekci 3.3.1. Vytvoření databázové instance a následné připravení databáze pro použití v testovacím prostředí je popsáno v sekci 3.3.2.

3.2 Přidání nových procesů do Teamcity pomocí Kotlin DSL

Celý backendový projekt Uniqway využívá aplikaci Teamcity (viz sekce 1.6) ke spouštění akcí sestavení nové verze aplikace a nasazení dané verze aplikace do stagingového a poté i produkčního prostředí. Přidal jsem tedy do Teamcity dva nové procesy. První proces spustí vytvoření nového testovacího prostředí, nasazení požadované verze backendové aplikace Uniqway do tohoto prostředí a otestuje tuto aplikaci pomocí API testů. V případě, že má být dané testovací prostředí odstraněno po dokončení API testů, tak tento proces ještě odstraní nově vytvořené testovací prostředí. Celý tento proces je popsán v sekci 3.3.

Druhý proces přidaný do Teamcity pouze odstraní požadované testovací prostředí. Tento proces je užitečný v případech, kdy nedochází k odstranění

testovacího prostředí ihned po dokončení API testů. Celý popis tohoto procesu se nachází v sekci 3.4.

Veškeré akce, které jsou spouštěné v rámci Teamcity, jsou definované pomocí kódu, který je napsaný v jazyce Kotlin (viz sekce 1.6.1). Vytvořil jsem tedy dva nové soubory ve složce *.teamcity/_Self/uniqplay* s názvy *RunTestsInTestingEnvironment* a *RemoveResourcesFromTestingEnvironment*. Tyto soubory obsahují definice výše popsaných akcí.

3.3 Proces vytvoření nového testovacího prostředí, spuštění API testů a případné odstranění vytvořeného testovacího prostředí

V této sekci se budu zabývat detailním popisem implementace dynamického vytvoření testovacího prostředí, nasazení požadované verze backendové aplikace Uniqway do tohoto prostředí a spuštění API testů proti této aplikaci. Poté popíšu proces odstranění vytvořeného testovacího prostředí, který je spuštěn v případě, že má být dané testovací prostředí odstraněno po dokončení API testů.

Jednotlivé kroky použité, při tomto procesu, jsou stručně popsány v tabulce 3.1 a detailně popsány v následujících sekcích 3.3.1 - 3.3.8. Pro lepší představu je celý tento proces ještě graficky znázorněn na obrázku 3.1.

3.3.1 Vytvoření nového databázového testovacího clusteru

Krok pro vytvoření nového databázového clusteru pro testovací prostředí je zobrazen na obrázku 3.2. Tento krok tedy obsahuje následující tři hodnoty nastavení:

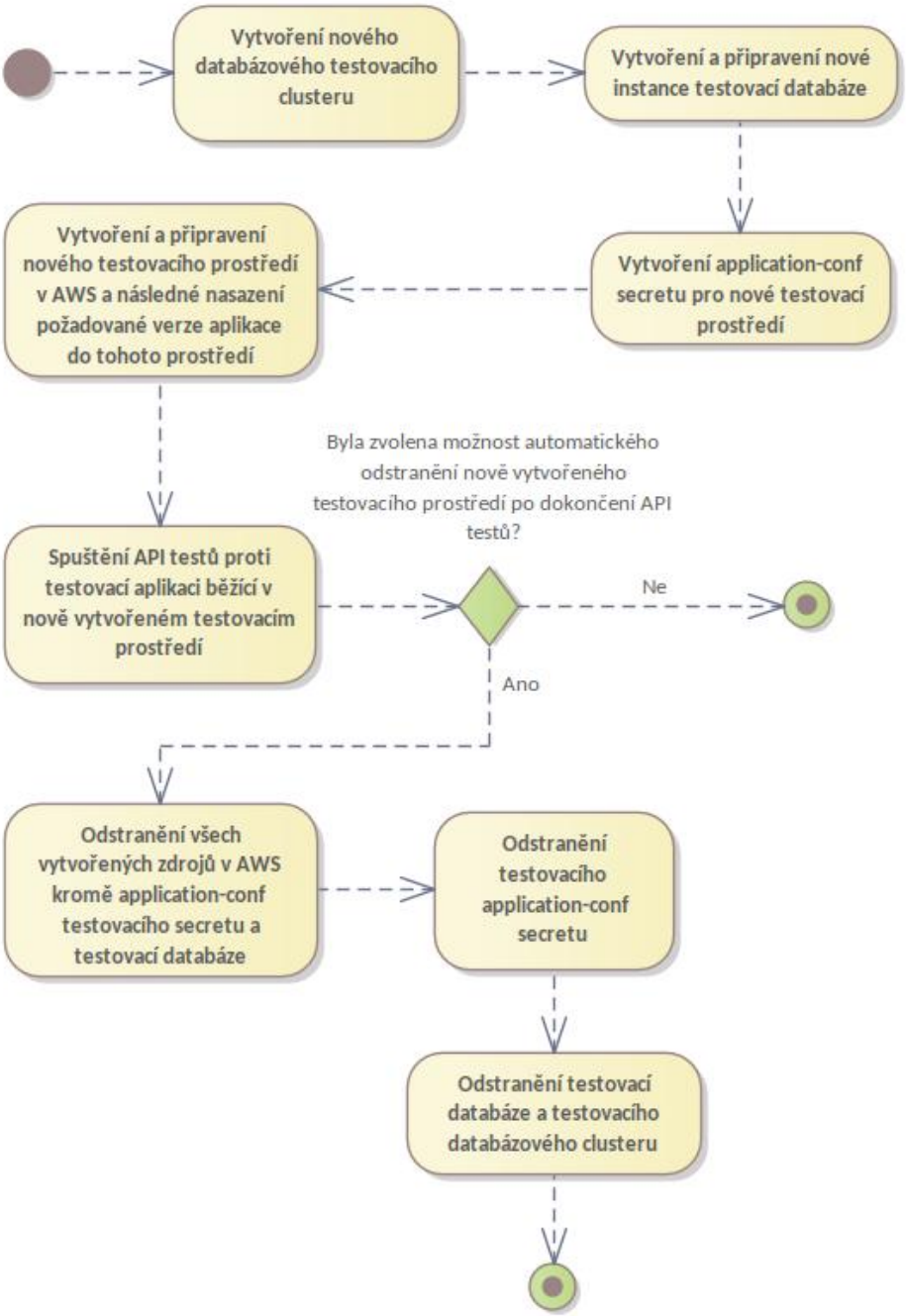
- name - Název daného kroku = trigger DB cluster creation
- workingDir - Adresář, ze kterého má být daný krok spuštěn = infra/aws
- scriptContent - Spouštěný skript = ./create-new-aurora-db-cluster.sh db-cluster-tst %teamcity.build.branch% %build.number%

V tomto kroku dojde ke spuštění bashového skriptu s názvem *create-new-aurora-db-cluster.sh* v adresáři *infra/aws*. Tomuto skriptu jsou předány parametry *db-cluster-tst*, *%teamcity.build.branch%* a *%build.number%*, kde *db-cluster-tst* je použito jako předpona názvu vytvořeného clusteru, *%teamcity.build.branch%* odpovídá názvu větve ze sdíleného repozitáře verzovacího systému GIT, pro kterou má být tento cluster vytvořen a *%build.number%* odpovídá číslu sestavení.

Jednotlivé kroky, které obsahuje skript *create-new-aurora-db-cluster.sh* jsou stručně popsány v tabulce 3.2 a detailně popsány v následujících sekcích. Zároveň jsou jednotlivé kroky graficky zobrazeny na obrázku 3.3.

Tabulka 3.1: Jednotlivé kroky použité při vytváření nového testovacího prostředí

Pořadí	Název	Cíl	Poznámka
1	trigger DB cluster creation	Vytvoření nového databázového testovacího clusteru.	
2	trigger DB instance creation	Vytvoření a připravení nové instance testovací databáze.	
3	trigger application-conf secret creation	Vytvoření application-conf secretu pro nové testovací prostředí.	
4	trigger service creation via terraform	Vytvoření a připravení nového testovacího prostředí v AWS a následné nasazení požadované verze aplikace do tohoto prostředí.	
5	run Rest API tests for testing	Spuštění API testů proti testovací aplikaci běžící v nově vytvořeném testovacím prostředí.	
6	trigger destruction of resources created via terraform	Odstranění všech vytvořených zdrojů v kroku 4.	Tento krok bude spuštěn, pokud bylo zvoleno automatické odstranění vytvořeného testovacího prostředí. K jeho spuštění dojde, i když některý z předchozích kroků nebyl úspěšně dokončen.
7	trigger application-conf secret destruction	Odstranění secretu application-conf vytvořeného v kroku 3.	Tento krok bude spuštěn, pokud bylo zvoleno automatické odstranění vytvořeného testovacího prostředí. K jeho spuštění dojde, i když některý z předchozích kroků nebyl úspěšně dokončen.
8	trigger DB instance and cluster destruction	Odstranění testovací databáze a testovacího databázového clusteru vytvořených v krocích 1 a 2.	Tento krok bude spuštěn, pokud bylo zvoleno automatické odstranění vytvořeného testovacího prostředí. K jeho spuštění dojde, i když některý z předchozích kroků nebyl úspěšně dokončen.



Obrázek 3.1: Jednotlivé kroky použité při vytváření nového testovacího prostředí

```
script {
    name = "trigger DB cluster creation"
    workingDir = "infra/aws"
    scriptContent = "./create-new-aurora-db-cluster.sh
    ↪  db-cluster-tst %teamcity.build.branch% %build.number%"
}
```

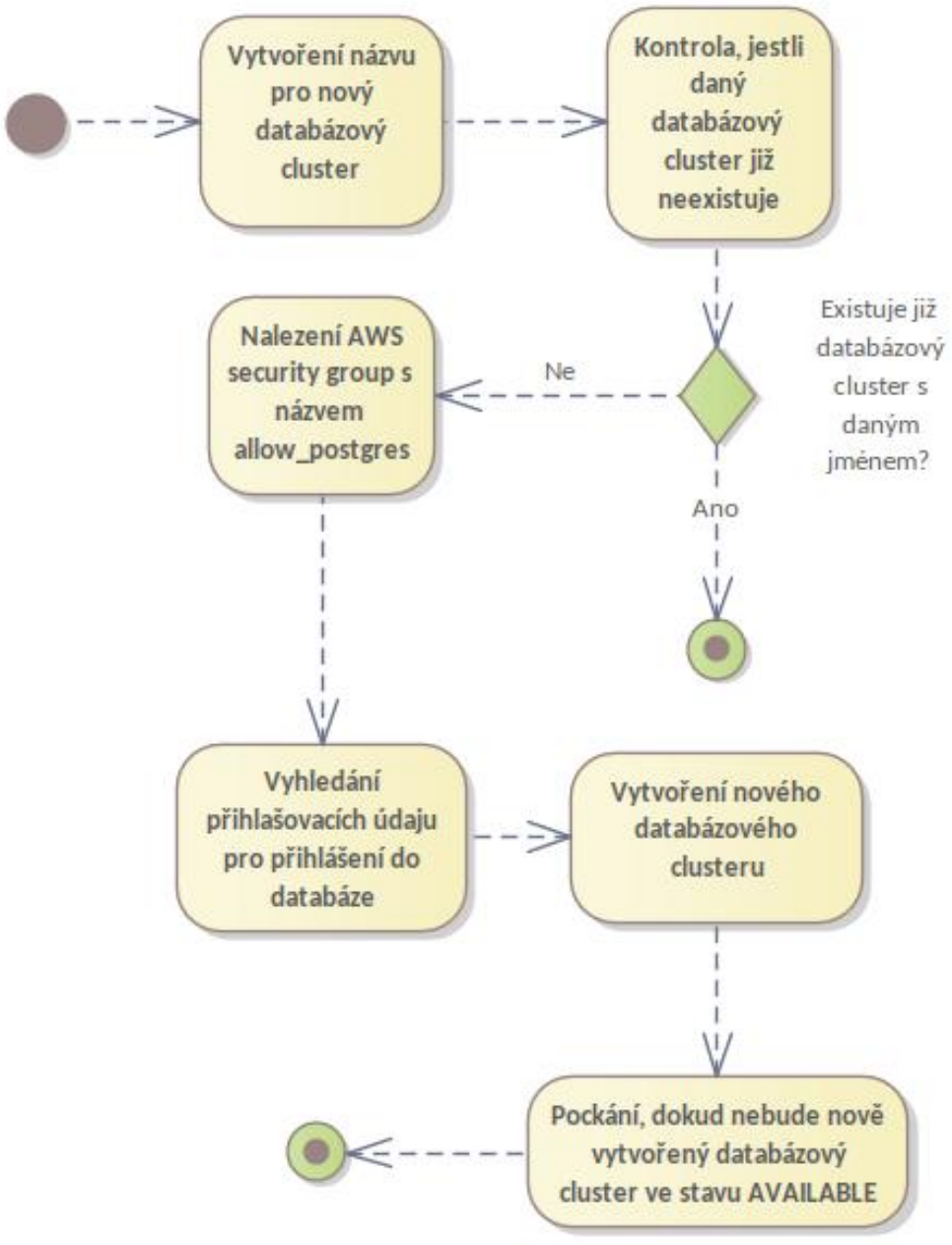
Obrázek 3.2: Krok pro vytvoření nového databázového testovacího cluster

Tabulka 3.2: Jednotlivé kroky použité při vytváření nového databázového testovacího clusteru

Pořadí	Název metody	Cíl	Vstupní parametry
1		Vytvoření názvu pro nový databázový cluster.	
2	check_if_already_exists_cluster	Kontrola, jestli databázový cluster s daným jménem již neexistuje.	Název databázového clusteru.
3	find_allow_postgres_security_group	Nalezení AWS security group s názvem allow_postgres.	
4	create_new_aurora_cluster	Vytvoření nového databázového clusteru.	Název databázového clusteru a identifikátor AWS security group s názvem allow_postgres.
5	wait_for_available_status	Počkání, dokud se nově vytvořený databázový cluster nebude nacházet ve stavu AVAILABLE.	Název databázového clusteru.

3.3.1.1 Vytvoření názvu pro nový databázový cluster

Obrázek 3.4 ukazuje kód, pomocí kterého dochází k vytvoření jména nového testovacího clusteru. Název vytvořeného databázového clusteru bude tedy složen ze tří částí. První část je tvořena předponu, kterou skript obdržel jako první parametr. Druhá část obsahuje číslo sestavení, které skript obdržel jako třetí parametr. Poslední část je tvořena názvem větve, který skript obdržel jako druhý parametr. Z názvu větve jsou odstraněny veškeré nealfanumerické znaky. Všechny tyto tři části jsou spojeny pomocí pomlček a výsledný název je navíc ještě oříznut pouze na prvních šedesát tři znaků. Tyto kroky jsou zapotřebí kvůli podmínkám napsaným v dokumentaci, viz zdroj [21].



Obrázek 3.3: Jednotlivé kroky použité při vytváření nového databázového clusteru

```
CLUSTER_PREFIX=$1;
BRANCH_NAME=$2;
BUILD_NUMBER=$3;

cluster_name=${CLUSTER_PREFIX}-${BUILD_NUMBER}-$(echo
↪  ${BRANCH_NAME} | tr -dc '[:alnum:]\n\r');
cluster_name=${cluster_name:0:63};
```

Obrázek 3.4: Kód pro vytvoření jména nového databázového testovacího cluster

```
aws --profile uniqway --region eu-west-1 rds describe-db-clusters \
        --filters "Name=db-cluster-id,Values=${cluster_name}" \
        --output text \
        --query 'DBClusters[*]'
```

Obrázek 3.5: Použití AWS CLI funkce rds describe-db-clusters

Název clusteru tedy musí splňovat následující čtyři podmínky:

- Nesmí být delší než 63 znaků.
- První znak musí být písmeno.
- Nesmí obsahovat více než jednu pomlčku za sebou.
- Nesmí končit pomlčkou.

Pro ilustraci uvedu následující příklad. Skript pro vytvoření nového databázového clusteru obdrží následující parametry:

- Předponu názvu = db-cluster-tst
- Název větve ze sdíleného repozitáře verzovacího systému GIT = testovaci--vetev-pro-demonstraci-vytvoreni-databazoveho-clusteru
- Číslo sestavení = 123

Výsledný název vytvořeného databázového clusteru bude tedy *db-cluster-tst-123-testovacivetevprodemonstracivytvorenidatabazovehoclusteru* zkrácený na 63 znaků, tedy *db-cluster-tst-123-testovacivetevprodemonstracivytvorenidatabaz*.

3.3.1.2 Kontrola, jestli databázový cluster s daným jménem již neexistuje

Metoda *check_if_already_exists_cluster* je využita ke kontrole, zda databázový cluster s daným názvem není již vytvořený. K tomuto účelu využívá AWS rozhraní příkazové řádky, konkrétně funkci *rds describe-db-clusters*. Použití této funkce je zobrazeno na obrázku 3.5.

Jak můžeme vidět, tak funkce využívá přepínač *--filters*, který přijímá parametry Name a Values. Následně využije přijatých parametrů z tohoto přepínače a pokusí se vyhledat veškeré databázové clustery, které odpovídají danému filteru. Přepínač filters přijímá tedy hodnoty *Name=db-cluster-id* a *Values=*${*cluster_name*}, kde ${cluster_name} bude nahrazeno za název databázového clusteru.

Dojde tedy k vyhledání všech databázových clusterů, které mají název odpovídající hodnotě přijatého parametru. V případě, že žádný takový cluster neexistuje, tak bude vrácen prázdný řetězec jako návratová hodnota. Tato hodnota je uložena do proměnné *found_existing_cluster*.

Pokud proměnná *found_existing_cluster* obsahuje pouze prázdný řetězec, tak víme, že cluster s daným jménem neexistuje. V opačném případě víme, že daný cluster existuje a nebude tedy vytvořen.

3.3.1.3 Nalezení AWS security group s názvem allow_postgres

Aby bylo možné přistupovat k vytvořenému databázovému clusteru, tak je potřeba, aby měl definovanou AWS security group (viz sekce 1.4.8), která tento přístup umožňuje. Metoda *find_allow_postgres_security_group* slouží k vyhledání security group s názvem allow_postgres. Tato security group již existuje v backendovém projektu Uniqway a povoluje přístup k databázi pomocí nástroje psql každému, kdo je ve stejném VPC (viz sekce 1.4.7) jako daná databáze.

K vyhledání potřebné security group je využito AWS rozhraní příkazové řádky, konkrétně funkce *ec2 describe-security-groups*. Tato funkce je zobrazena na obrázku 3.6. Jak můžeme vidět, tak funkce využívá přepínač *--filters*, který přijímá parametry Name a Values. Následně využije přijatých parametrů z tohoto přepínače a pokusí se vyhledat veškeré security groups, které odpovídají danému filteru. Přepínač filters přijímá tedy následující hodnoty *Name=group-name* a *Values=allow_postgres*.

Dojde k vyhledání všech security groups, které mají název allow_postgres. Následně pomocí přepínače *--query*, který v našem případě přijímá hodnotu *SecurityGroups[0].[GroupId]*, dojde k vyhledání identifikátoru nalezené security group.

Nalezená hodnota identifikátoru security group s názvem allow_postgres je následně uložena do globální proměnné allow_postgres_security_group.

```
aws --profile uniqway --region eu-west-1 ec2 describe-security-groups \
        --filters 'Name=group-name,Values=allow_postgres' \
        --output text \
        --query 'SecurityGroups[0].[GroupId]'
```

Obrázek 3.6: Použití AWS CLI funkce ec2 describe-security-groups

```
aws --profile uniqway --region eu-west-1 secretsmanager get-secret-value \
        --secret-id "database-username-testing" \
        --output text \
        --query "SecretString" | base64 --decode

aws --profile uniqway --region eu-west-1 secretsmanager get-secret-value \
        --secret-id "database-password-testing" \
        --output text \
        --query "SecretString" | base64 --decode
```

Obrázek 3.7: Použití AWS CLI funkce secretsmanager get-secret-value pro získání přístupového jména a hesla k testovací databázi

3.3.1.4 Vytvoření nového databázového clusteru

Pro vytvoření nového databázového clusteru je použita metoda s názvem *create_new_aurora_cluster*, která přijímá dva vstupní parametry - název nového databázového clusteru a security group umožňující přístup k tomuto clusteru. Tato metoda nejprve vyhledá jméno a heslo uložené v AWS Secrets Manageru (viz sekce 1.4.10), které budou sloužit pro přihlášení se k tomuto databázovému clusteru. Pro vyhledání těchto hodnot je použito AWS rozhraní příkazové řádky, konkrétně funkce *secretsmanager get-secret-value*. Obrázek 3.7 zobrazuje použití této funkce pro získání přístupového jména a hesla k databázi.

Funcke *secretsmanager get-secret-value* přijímá přepínač *--secret-id*, který obsahuje jméno hledaného secretu v Amazon Secrets Manageru. V našem případě jsem využil tuto metodu dvakrát. Poprvé s hodnotou přepínače *--secret-id* nastavenou na *database-username-testing* pro získání jména použitého pro přihlášení k databázi. Podruhé s hodnotou přepínače *--secret-id* nastavenou na *database-password-testing* pro získání hesla použitého pro přihlášení k databázi. Při použití obou funkcí je ještě použit přepínač *--query*, který přijímá hodnotu *SecretString* a pomocí nějž dojde k vyhledání uloženého textu v daném secretu. Nalezené jméno je uloženo do proměnné *database_username* a nalezené heslo je uloženo do proměnné *database_password*.

Následně je využita funkce AWS rozhraní příkazové řádky *rds create-db-*

```
aws --profile uniqway --region eu-west-1 rds create-db-cluster \
        --db-cluster-identifier ${cluster_name} \
        --engine aurora-postgresql \
        --serverless-v2-scaling-configuration "MinCapacity=1,MaxCapacity=4" \
        --engine-version 14.5 \
        --master-username "${database_username}" \
        --master-user-password "${database_password}" \
        --database-name "uniqplay_db" \
        --db-subnet-group-name "database" \
        --vpc-security-group-ids "${security_group}" \
        --output text \
        --query "DBCluster[0].[DBClusterArn]"
```

Obrázek 3.8: Použití AWS CLI funkce rds create-db-cluster

```
aws --profile uniqway --region eu-west-1 rds wait db-cluster-available \
        --db-cluster-identifier ${cluster_name}
```

Obrázek 3.9: Použití AWS CLI funkce rds wait db-cluster-available

cluster, která vytvoří nový databázový cluster. Kód zobrazený na obrázku 3.8 ukazuje použití této funkce. Tabulka 3.3 obsahuje popis využitých přepínačů.

3.3.1.5 Počkání, dokud se nebude nacházet vytvořený databázový cluster ve stavu AVAILABLE

Ihned po vytvoření nového databázového clusteru není možné tento cluster využívat. Je potřeba ještě počkat na to, než bude daný cluster ve stavu AVAILABLE. K tomuto účelu je využita metoda *wait_for_available_status*, která přijímá název clusteru jako vstupní parametr. Tato metoda využívá funkci AWS rozhraní příkazové řádky *rds wait db-cluster-available*, která využívá přepínač *--db-cluster-identifier*. Tento přepínač přijímá jako hodnotu název clusteru. Funkce *rds wait db-cluster-available* bude čekat, dokud se daný cluster nebude nacházet ve stavu AVAILABLE. Kód zobrazený na obrázku 3.9 ukazuje použití AWS funkce *rds wait db-cluster-available*.

Tabulka 3.3: Popis přepínačů využitých v AWS funkci *rds create-db-cluster*

Název	Hodnota
--db-cluster-identifier	Název vytvořeného databázového clusteru.
--engine	Typ vytvořeného databázového clusteru. V našem případě se jedná o databázi Amazon Aurora využívající PostgreSQL.
--serverless-v2-scaling-configuration	Konfigurace, která bude využita pro škálování databáze. Obsahuje minimální a maximální kapacitu využitou pro škálování.
--engine-version	Verze databázového enginu. V našem případě bude využito PostgreSQL ve verzi 14.5.
--master-username	Uživatelské jméno použité pro přihlášení k databázi.
--master-user-password	Heslo použité pro přihlášení k databázi.
--database-name	Název databáze, která má být v daném clusteru vytvořena. V našem případě dojde k vytvoření databáze s názvem uniqplay_db.
--db-subnet-group-name	Název subnetu, který má být využit pro daný databázový cluster. V našem případě je využit již existující subnet s názvem database.
--vpc-security-group-ids	Identifikátor security group, která má být použita pro daný databázový cluster. V našem případě tato security group umožňuje přístup k databázi pomocí nástroje psql.
--output	V jakém formátu má být návratová hodnota dané funkce. V našem případě má tato hodnota být v textovém formátu.
--query	Jaká hodnota má být z vrácena z dané funkce. V našem případě má být vrácen Amazon Resource Name vytvořeného clusteru (viz sekce 1.4.1).

```
script {
    name = "trigger DB instance creation"
    workingDir = "infra/aws"
    scriptContent = "./create-new-aurora-db-instance.sh
    ↪ db-cluster-tst db-instance-tst %teamcity.build.branch%
    ↪ %build.number%"
}
```

Obrázek 3.10: Krok pro vytvoření nové instance testovací databáze

3.3.2 Vytvoření a připravení nové instance testovací databáze

Krok pro vytvoření nové databázové instance, která umožňuje číst a zapisovat data z a do testovací databáze je zobrazen na obrázku 3.10. Tento krok tedy obsahuje následující tři hodnoty nastavení:

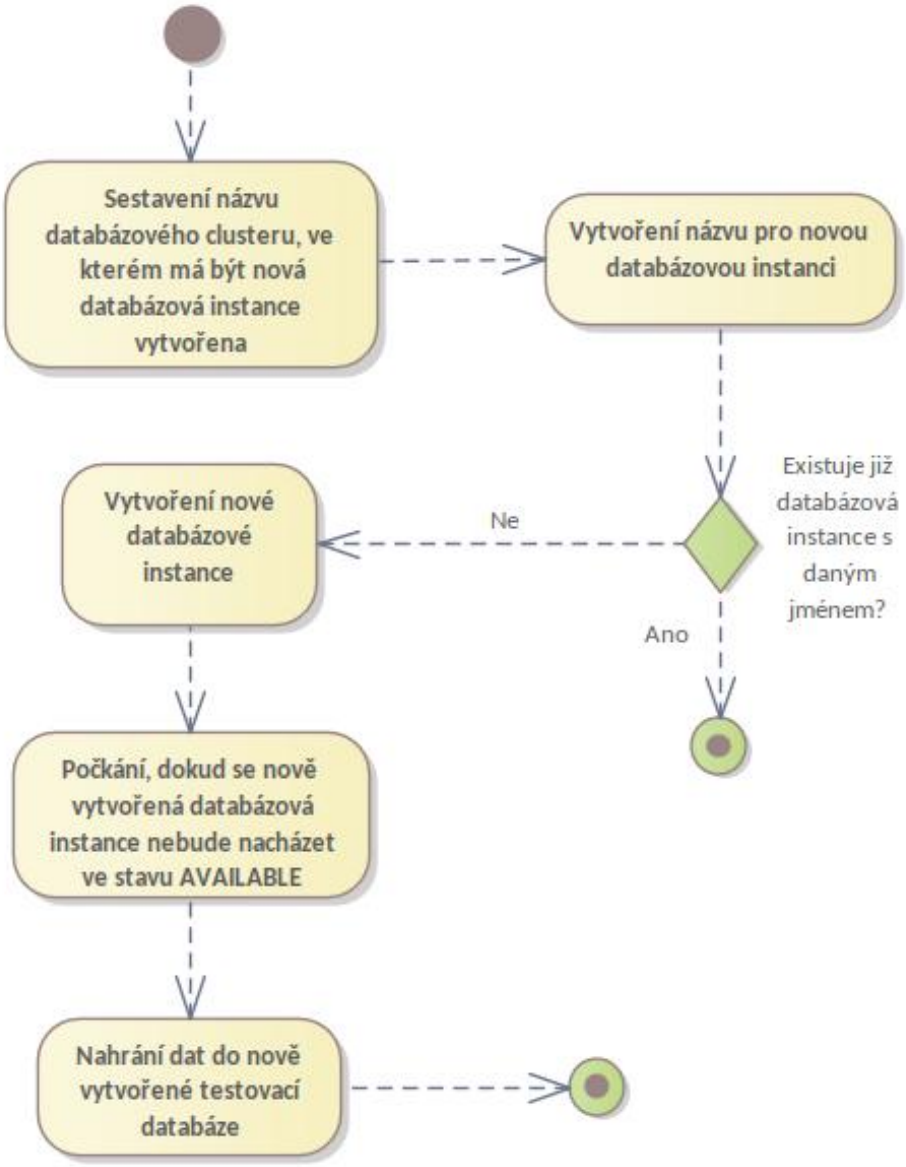
- name - Název daného kroku = trigger DB instance creation
- workingDir - Adresář, ze kterého má být daný krok spuštěn = infra/aws
- scriptContent - Spouštěný skript = ./create-new-aurora-db-instance.sh db-cluster-tst db-instance-tst %teamcity.build.branch% %build.number%

V tomto kroku dojde ke spuštění bashového skriptu s názvem *create-new-aurora-db-instance.sh* v adresáři *infra/aws*. Tomuto skriptu jsou předány parametry *db-cluster-tst*, *db-instance-tst*, *%teamcity.build.branch%* a *%build.number%*, kde *db-cluster-tst* je použito pro sestavení názvu databázového clusteru, ve kterém má být nová databázová instance vytvořena, *db-instance-tst* odpovídá předponě názvu nové databázové instance, *%teamcity.build.branch%* představuje názvu větve ze sdíleného repozitáře verzovacího systému GIT, pro kterou má být tato instance vytvořena a *%build.number%* odpovídá číslu sestavení.

Jednotlivé kroky, které obsahuje skript *create-new-aurora-db-instance.sh* jsou stručně popsány v tabulce 3.4 a detailně popsány v následujících sekcích. Zároveň jsou jednotlivé kroky graficky zobrazeny na obrázku 3.11.

3.3.2.1 Sestavení názvu databázového clusteru, ve kterém má být nová databázová instance vytvořena

Při vytváření nové databázové instance je zapotřebí znát jméno databázového clusteru, ve kterém má být daná instance vytvořena. Tento proces je totožný s procesem sestavení jména databázového clusteru, který má být vytvořen. Proces je tedy popsán v sekci 3.3.1.1.



Obrázek 3.11: Jednotlivé kroky použité při vytváření nové databázové instance

Tabulka 3.4: Jednotlivé kroky použité při vytváření nové databázové testovací instance

Pořadí	Název metody	Cíl	Vstupní parametry
1		Sestavení názvu databázového clusteru, ve kterém má být nová databázová instance vytvořena.	
2		Vytvoření názvu pro novou databázovou instanci.	
3	check_if_already_exists_instance	Kontrola, jestli databázová instance s daným jménem již neexistuje.	Název databázové instance.
4	create_new_aurora_instance	Vytvoření nové databázové instance.	Název databázového clusteru a nové databázové instance.
5	wait_for_available_status	Počkání, dokud se nově vytvořená databázová instance nebude nacházet ve stavu AVAILABLE.	Název databázové instance.
6	deploy_data_to_database	Nahrání dat do nově vytvořené testovací databáze.	Název databázové instance.

```
1 INSTANCE_PREFIX=$2;
2 BRANCH_NAME=$3;
3 BUILD_NUMBER=$4;
4
5 instance_name=${INSTANCE_PREFIX}-${BUILD_NUMBER}-$(echo
  ↪  ${BRANCH_NAME} | tr -dc '[:alnum:]\n\r');
6 instance_name=${instance_name:0:63};
```

Obrázek 3.12: Kód pro vytvoření jména nového databázového testovacího clusteru

```
aws --profile uniqway --region eu-west-1 rds describe-db-instances \
    --db-instance-identifier ${instance_name}
```

Obrázek 3.13: Použití AWS CLI funkce rds describe-db-instances

3.3.2.2 Vytvoření názvu pro novou databázovou instanci

Obrázek 3.12 ukazuje kód, pomocí kterého dochází k vytvoření jména nové databázové testovací instance. Název vytvořené databázové instance bude tedy složen ze tří částí. První část je tvořena předponu, kterou skript obdržel jako druhý parametr. Druhá část obsahuje číslo sestavení, které skript obdržel jako čtvrtý parametr. Poslední část je tvořena názvem větve, který skript obdržel jako třetí parametr. Z názvu větve jsou odstraněny veškeré nealfanumerické znaky. Všechny tyto tři části jsou spojeny pomocí pomlček a výsledný název je navíc ještě oříznut pouze na prvních šedesát tři znaků. Tyto kroky jsou zapotřebí kvůli podmínkám napsaným v dokumentaci, kterou obsahuje zdroj [22]. Název instance tedy musí splňovat následující čtyři podmínky:

- Nesmí být delší než 63 znaků.
- První znak musí být písmeno.
- Nesmí obsahovat více než jednu pomlčku za sebou.
- Nesmí končit pomlčkou.

Pro ilustraci uvedu následující příklad. Skript pro vytvoření nové databázové instance obdrží následující parametry:

- Předponu názvu = db-instance-tst
- Název větve ze sdíleného repozitáře verzovacího systému GIT = testovaci--vetev-pro-demonstraci-vytvoreni-databazove-instance
- Číslo sestavení = 123

Výsledný název vytvořené databázové instance bude tedy *db-instance-tst-123-testovacivetevprodemonstracivytvorenidatabazoveinstance* zkrácený na 63 znaků, tedy *db-instance-tst-123-testovacivetevprodemonstracivytvorenidataba.*

3.3.2.3 Kontrola, jestli databázová instance s daným jménem již neexistuje

Metoda *check_if_already_exists_instance* má za úkol zkontrolovat, zda databázová instance s daným názvem není již vytvořena. K tomuto účelu je

```
aws --profile uniqway --region eu-west-1 rds create-db-instance \
        --db-instance-identifier ${instance_name} \
        --db-cluster-identifier ${cluster_name} \
        --engine aurora-postgresql \
        --db-instance-class db.serverless \
        --output text \
        --query "DBInstance[0].[DBInstanceArn]"
```

Obrázek 3.14: Použití AWS CLI funkce rds create-db-instance

využito AWS rozhraní příkazové řádky, konkrétně funkce *rds describe-db-instances*. Použití této funkce je zobrazeno na obrázku 3.13.

Jak můžeme vidět, tak funkce využívá přepínač *--db-instance-identifier*, který přijímá název databázové instance jako parametr. Funkce se pokusí vyhledat veškeré databázové instance s daným identifikátorem, v našem případě se jedná o název pro novou databázovou instanci. Funkce vrací informace o nalezené databázové instanci. V případě, že žádná taková databázová instance neexistuje, tak je vrácen prázdný řetězec. Vrácená hodnota je uložena do proměnné *found_existing_instance*.

Pokud proměnná *found_existing_instance* obsahuje pouze prázdný řetězec, tak víme, že databázová instance s daným jménem neexistuje. V opačném případě víme, že daná databázová instance existuje, nebude tedy vytvořena a bude použita tato existující instance.

3.3.2.4 Vytvoření nové databázové instance

Pro vytvoření nové databázové instance je použita metoda s názvem *create_new_aurora_instance*, která přijímá dva vstupní parametry - název databázového clusteru, ve kterém má být nová databázová instance vytvořena a název nové databázové instance. Tato metoda využívá AWS rozhraní příkazové řádky, konkrétně funkce *rds create-db-instance*. Obrázek 3.14 zobrazuje použití této funkce pro vytvoření nové databázové instance. Tabulka 3.5 obsahuje popis využitých přepínačů.

3.3.2.5 Počkání, dokud se nově vytvořená databázová instance nebude nacházet ve stavu AVAILABLE

Ihned po návratu ze spuštěného příkazu vytvoření nové databázové instance není možné tuto instanci využívat, jelikož stále ještě probíhá proces jejího vytvoření. Je potřeba ještě počkat na to, než bude daná instance kompletně vytvořena, tzn. bude ve stavu AVAILABLE. K tomuto účelu je využita metoda *wait_for_available_status*, která přijímá název databázové instance jako vstupní parametr. Tato metoda využívá funkci AWS rozhraní příkazové řádky

Tabulka 3.5: Popis přepínačů využitých v AWS funkci *rds create-db-instance*

Název	Hodnota
--db-instance-identifier	Název vytvořené databázové instance.
--db-cluster-identifier	Název databázového clusteru, ve kterém má být nová databázová instance vytvořena.
--engine	Databázový engine vytvořené databázové instance. V našem případě se jedná o databázi Amazon Aurora využívající PostgreSQL.
--db-instance-class	Typ vytvořené databázové instance. V našem případě se jedná o serverless databázi, viz sekce 2.2.
--output	V jakém formátu má být návratová hodnota dané funkce. V našem případě má tato hodnota být v textovém formátu.
--query	Jaká hodnota má být vrácena z dané funkce. V našem případě má být vrácen Amazon Resource Name vytvořené databázové instance (viz sekce 1.4.1).

```
1 aws --profile uniqway --region eu-west-1 rds wait db-instance-available \
2     --db-instance-identifier ${instance_name}
```

Obrázek 3.15: Použití AWS CLI funkce rds wait db-instance-available

rds wait db-instance-available, která využívá přepínač *--db-instance-identifier*. Tento přepínač přijímá jako hodnotu název instance. Funkce *rds wait db-instance-available* bude čekat, dokud se daná instance nebude nacházet ve stavu AVAILABLE. Kód zobrazený na obrázku 3.15 ukazuje použití AWS funkce *rds wait db-instance-available*.

3.3.2.6 Nahrání dat do nově vytvořené testovací databáze

Pro správné fungování testovacího prostředí je potřeba naplnit testovací databázi daty. K tomuto účelu slouží metoda *deploy_data_to_database*, která přijímá název databázové instance jako vstupní parametr. Tato metoda zkopíruje data z produkční databáze a nahraje je do vytvořené testovací databáze. Následně provede potřebné změny v testovací databázi tak, aby bylo možné ji správně použít v testovacím prostředí (důvody provedení změn jsou popsány na konci této kapitoly).

Pro připojení ke konkrétní databázi je potřeba znát její endpoint, jedná se o URL adresu dané databázové instance. Pomocí endpointů je možné se připojit k produkční a k testovací databázi. K tomuto účelu je využito AWS rozhraní příkazové řádky *rds describe-db-instances*, které využívá přepínač *--db-instance-identifier*. Tento přepínač přijímá jako hodnotu název dané databázové instance. Víme, že produkční databáze má identifikátor *„uniqplay-database-production“*. Pro získání informací o testovací databázi využijeme

```
aws --profile uniqway --region eu-west-1 rds describe-db-instances \
        --db-instance-identifier ${instance_name} \
        --output text \
        --query 'DBInstances[0].Endpoint.Address'
```

Obrázek 3.16: Použití AWS CLI funkce rds describe-db-instances

název, jehož sestavení je popsáno v sekci 3.3.2.1. Následně pomocí přepínače *--query* je možné vyfiltrovat přímo hodnotu endpointu. K tomuto účelu je potřeba předat přepínači *--query* hodnotu *„DBInstances[0].Endpoint.Address“*. Získaný endpoint je možné využít k připojení k dané databázové instanci. Kód zobrazený na obrázku 3.16 ukazuje použití této AWS funkce.

Pro uložení souřadnic využívá Uniqway databáze tzv. Geometries. Jedná se o speciální datový typ, který je obsažen v rozšíření PostgreSQL s názvem postgis. Je tedy potřeba toto rozšíření nainstalovat v nově vytvořené testovací databázi. K tomuto účelu je využit nástroj *psql*, který může obsahovat přepínač *-c*. Tento přepínač přijímá jako hodnotu skript v textové podobě, který je následně spuštěn v dané databázi. Stačí tedy předat příkaz *CREATE EXTENSION postgis;*, který nainstaluje rozšíření postgis v dané databázi.

Poté dojde k vytvoření kopie produkční databáze pomocí nástroje *pg_dump*. Tento nástroj se využívá k vytvoření zálohy celé databáze. Pomocí něj tedy získáme kopii celé produkční databáze, která je následně uložena do souboru. Následně jsme schopni pomocí nástroje *pg_restore* obnovit celou databázi, stačí když tomuto příkazu předáme soubor, který byl vytvořen pomocí *pg_restore*.

V posledním kroku je potřeba upravit vytvořenou testovací databázi tak, aby bylo možné ji použít v testovacím prostředí. K tomuto účelu je vytvořen skript s názvem *stg-edits.sql*. Tento skript vytvoří v dané databázi potřebné uživatelské a administrátorské role a naplní ji daty, která jsou potřebná pro úspěšné dokončení API testů.

```
script {
    name = "trigger application-conf secret creation"
    workingDir = "infra/aws"
    scriptContent = "./application-conf-testing.sh create
    ↪ %teamcity.build.branch% %build.number% db-instance-tst"
}
```

Obrázek 3.17: Krok pro vytvoření nového testovacího secretu spplication-conf

3.3.3 Vytvoření application-conf secretu pro nové testovací prostředí

Testovací secret application-conf je potřebný pro správné spuštění Java Play aplikace, jelikož je následně použit jako konfigurační soubor *application.conf*, viz sekce 1.2. V testovacím prostředí je například využit pro připojení Uniqway aplikace k databázi. Krok pro vytvoření nového testovacího secretu application-conf je zobrazen na obrázku 3.17. Tento krok tedy obsahuje následující tři hodnoty nastavení:

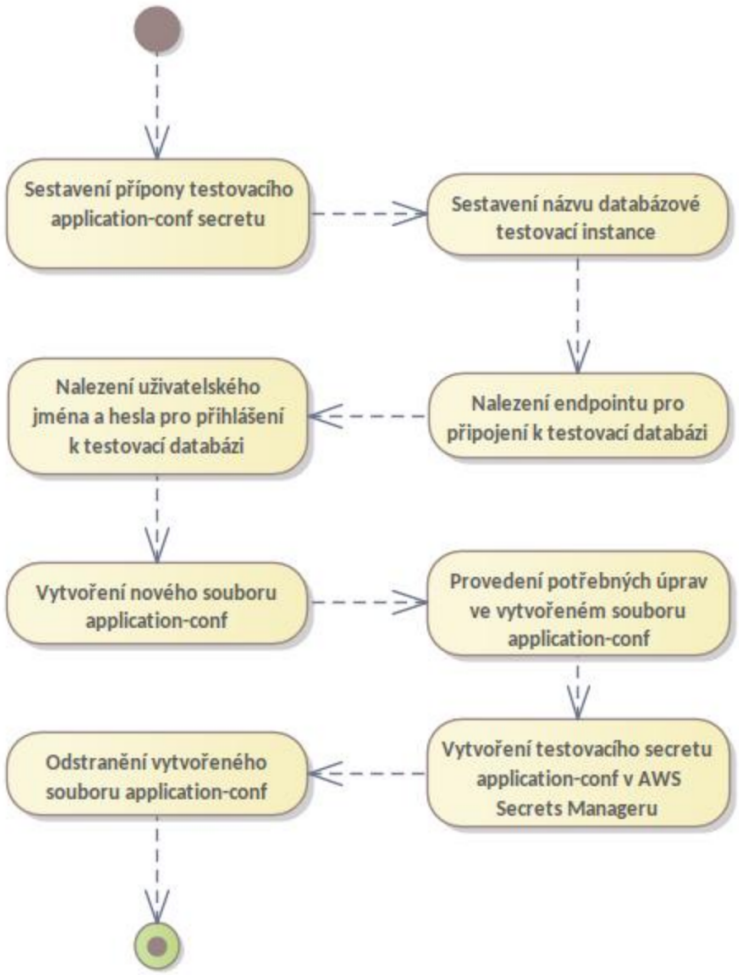
- name - Název daného kroku = trigger application-conf secret creation
- workingDir - Adresář, ze kterého má být daný krok spuštěn = infra/aws
- scriptContent - Spouštěný skript = ./application-conf-testing.sh create %teamcity.build.branch% %build.number% db-instance-tst

V tomto kroku dojde ke spuštění bashového skriptu s názvem *application-conf-testing.sh* v adresáři *infra/aws*. Tomuto skriptu jsou předány parametry *create*, *%teamcity.build.branch%*, *%build.number%* a *db-instance-tst*, kde *db-instance-tst* je použito jako předpona názvu vytvořené databázové testovací instance, *%teamcity.build.branch%* odpovídá názvu větve ze sdíleného repozitáře verzovacího systému GIT, pro kterou má být tento secret vytvořen a *%build.number%* odpovídá číslu sestavení.

Jednotlivé kroky, které obsahuje skript *application-conf-testing.sh* jsou stručně popsány v tabulce 3.6 a detailně popsány v následujících sekcích. Zároveň jsou jednotlivé kroky graficky zobrazeny na obrázku 3.18.

3.3.3.1 Sestavení přípony testovacího application-conf secretu

Každý testovací application-conf secret je použit právě pro jedno testovací prostředí. Je tedy potřeba určit, jaký application-conf secret má být použit pro dané testovací prostředí. K tomuto účelu slouží název daného application-conf secretu a to konkrétně jeho poslední část, k jíž vytvoření slouží tento



Obrázek 3.18: Jednotlivé kroky použité při vytváření nového testovacího secretu application-conf

Tabulka 3.6: Jednotlivé kroky použité při vytváření nového secretu application-conf pro testovací prostředí

Pořadí	Název metody	Cíl	Vstupní parametry
1		Sestavení přípony testovacího application-conf secretu.	
2		Sestavení názvu databázové testovací instance.	
3	set_up_database_information	Nalezení a nastavení informací o databázové testovací instanci.	Název databázové instance.
4	create_application_conf_file	Vytvoření a upravení souboru appliaction-conf.	Přípona nově vytvořeného secretu.
5	create_secret	Vytvoření nového secretu v AWS Secrets Manageru.	Přípona nově vytvořeného secretu.
6	delete_tmp_file	Odstranění vytvořeného souboru z kroku 4.	Přípona nově vytvořeného secretu.

krok. Tato část se skládá z čísla sestavení a názvu větve ze sdíleného repozitáře verzovacího systému GIT, pro kterou je dané testovací prostředí vytvořeno. Obě tyto části jsou spojeny pomocí znaku pomlčky. Přípona tedy bude vypadat následovně „<BUILD_NUMBER>-<BRANCH_NAME>“, kde <BUILD_NUMBER> odpovídá číslu sestavení a <BRANCH_NAME> odpovídá názvu větve ze sdíleného repozitáře verzovacího systému GIT, pro kterou je dané testovací prostředí vytvořeno. Celý název application-conf testovacího secretu vypadá následovně „application-conf-<BUILD_NUMBER>-<BRANCH_NAME>“.

3.3.3.2 Sestavení názvu databázové testovací instance

Aby bylo možné získat informace o databázové instanci, tak je nejprve potřeba znát její název. Tento název je specifický pro databázové testovací instance a jeho vytvoření je stejné, jako při jejím vytváření. Celý tento proces je popsán v sekci 3.3.2.2.

3.3.3.3 Nalezení a nastavení informací o databázové testovací instanci

Tento krok zodpovídá za získání informací potřebných k připojení k testovací databázi. Je potřeba získat tři informace, kterými jsou jméno a heslou používané pro přihlášení k databázi a endpoint, pomocí kterého je možné se k této databázi připojit.

Získání informace o databázovém endpointu, pomocí kterého je možné připojení k testovací databázi je totožný, jako při vytváření nové databázové

```
db {
  default {
    driver = org.postgresql.Driver
    url = "jdbc:postgresql://<database_url>:5432/uniqplay_db"
    username = <database_username>
    password = "<database_password>"
  }
}
```

Obrázek 3.19: Konfigurace připojení k databázi v souboru application-conf-testing-[postfix]

testovací instance popsané v sekci 3.3.2.6. Informace o endpointu je následně uložena do proměnné *database_endpoint*.

Jméno a heslo potřebné pro přihlášení k databázi jsou uloženy jako secrety v AWS Secrets Manageru. Proces jejich získání je totožný jako při vytváření nového databázového clusteru, viz sekce 3.3.1.4. Získané hodnoty jsou následně uloženy do proměnných *database_username* a *database_password*.

3.3.3.4 Vytvoření a upravení souboru appliaction-conf

Jelikož Uniqway backendová aplikace komunikuje s databází Uniqway, tak je potřeba správně nastavit její připojení k této databázi. Připojení k databázi je definované pomocí konfiguračního souboru *application.conf*, viz sekce 1.2. Každá testovací instance má k dispozici právě jednu testovací databázi.

V tomto kroku dochází k vytvoření souboru *application-conf*, který následně může být použit jako konfigurační soubor *application.conf*. K vytvoření nového souboru application-conf je použita již předpřipravená šablona *application-conf.example*. Nejprve dojde k jejímu zkopírování a tím i k vytvoření nového souboru s názvem *application-conf-testing-[postfix]*, kde [postfix] odpovídá vytvořené příponě, která jednoznačně identifikuje testovací prostředí. Vytvoření této přípony je popsané v sekci 3.3.3.1.

Nově vytvořený soubor *application-conf-testing-[postfix]* obsahuje nastavení potřebné pro připojení k testovací databázi. Toto nastavení je zobrazené na obrázku 3.19. Je tedy potřeba v tomto souboru upravit URL, jméno a heslo potřebné pro připojení k databázi. K tomu je využit příkaz *sed*, který slouží k úprávě textového souboru. Tomuto příkazu jsou předány parametry pro potřebné úpravy. Příkaz vždy obdrží hledaný text, nový text a název daného textového souboru. Poté dojde k nahrazení všech výskytů hledaného textu novým textem v daném souboru. Využití příkazu *sed* je zobrazeno na obrázku 3.20.

```
sed -i -e "s/<database_url>/${database_endpoint}/g"
  ↪ ./application-conf-testing-${application_conf_postfix};
sed -i -e "s/<database_username>/${database_username}/g"
  ↪ ./application-conf-testing-${application_conf_postfix};
sed -i -e "s/<database_password>/${database_password}/g"
  ↪ ./application-conf-testing-${application_conf_postfix};
```

Obrázek 3.20: Použití příkazu *sed* k nahrazení potřebných parametrů pro připojení k databázi

```
aws --profile uniqway --region eu-west-1 secretsmanager
  ↪ create-secret \
    --name
      ↪ "application-conf-testing-${application_conf_postfix}" \
    --secret-string "$(cat
      ↪ "./application-conf-testing-${application_conf_postfix}"
      ↪ | base64)"
```

Obrázek 3.21: Použití AWS CLI funkce secretsmanager create-secret

3.3.3.5 Vytvoření nového secretu v AWS Secrets Manageru

V předchozím kroku popsaném v sekci 3.3.3.4 dojde k vytvoření a správnému upravení nového souboru application-conf. Aby bylo možné dále tento soubor používat, tak je uložen jako secret do AWS Secrets Manageru. Do AWS Secrets Manageru je uložen také z toho důvodu, že obsahuje citlivá data, jako například přístupové údaje k databázi. K vytvoření tohoto secretu je využito AWS rozhraní příkazové řádky, konkrétně funkce *secretsmanager create-secret*. Tato funkce využívá přepínače *--name*, který přijímá název nově vytvořeného secretu, a *--secret-string*, který přijímá hodnotu, která má být uložena do nově vytvořeného secretu. Použití této funkce je zobrazeno na obrázku 3.21.

Obsah vytvořeného souboru application-conf-[postfix] je tedy zakódován pomocí base64 a následně uložen do secretu s názvem application-conf-testing-[postfix]. V tomto případě [postfix] odpovídá vytvořené příponě, která jednoznačně identifikuje testovací prostředí. Vytvoření této přípony je popsané v sekci 3.3.3.1.

3.3.3.6 Odstranění vytvořeného application-conf souboru

Během procesu vytváření nového secretu application-conf dojde k vytvoření pomocného souboru, viz sekce 3.3.3.4. Je tedy potřeba tento soubor odstranit. K tomuto účelu slouží metoda *delete_tmp_file*, která jako parametr obdrží příponu vytvořeného application-conf souboru, který následně odstraní.

```
script {
    name = "trigger service creation via terraform"
    workingDir = "infra/"
    scriptContent = "./deploy-testing-app.sh
    ↪ %teamcity.build.branch% %build.number%
    ↪ ./terraform/ecs-cluster/uniqplay/testing"
}
```

Obrázek 3.22: Krok pro vytvoření nového testovacího prostředí a nasazení aplikace

3.3.4 Vytvoření a připravení nového testovacího prostředí v AWS a následné nasazení požadované verze aplikace do tohoto prostředí

Krok pro vytvoření nového testovacího prostředí a následné spuštění aplikace v tomto prostředí je zobrazen na obrázku 3.22. Tento krok tedy obsahuje následující tři hodnoty nastavení:

- name - Název daného kroku = trigger service creation via terraform
- workingDir - Adresář, ze kterého má být daný krok spuštěn = infra/
- scriptContent - Spouštěný skript = ./deploy-testing-app.sh

  %teamcity.build.branch% %build.number%

  ./terraform/ecs-cluster/uniqplay/testing

V tomto kroku dojde ke spuštění bashového skriptu s názvem *deploy-testing-app.sh* v adresáři *infra*. Tomuto skriptu jsou předány parametry *%teamcity.build.branch%*, *%build.number%* a *./terraform/ecs-cluster/uniqplay/testing*, kde *%teamcity.build.branch%* představuje název větve ze sdíleného repozitáře verzovacího systému GIT, pro kterou má být tato instance vytvořena, *%build.number%* odpovídá číslu sestavení a *./terraform/ecs-cluster/uniqplay/testing* odpovídá relativní cestě k terraform skriptu, který má být spuštěn pro vytvoření nového testovacího prostředí.

Jednotlivé kroky, které obsahuje skript *deploy-testing-app.sh* jsou stručně popsány v tabulce 3.7 a detailně popsány v následujících sekcích. Dojde tedy k vytvoření potřebných zdrojů a celého testovacího prostředí v AWS. Následně bude nasazena požadovaná verze backendové aplikace Uniqway do tohoto prostředí. Celý tento proces je realizovaný pomocí terraform skriptů. Technologie terraform je popsána v sekci 1.5.

Tabulka 3.7: Jednotlivé kroky použité při vytváření nového testovacího prostředí a následném spuštění Uniqway aplikace

Pořadí	Název metody	Cíl	Vstupní parametry
1	create_init_tf	Vytvoření souboru init.tf použitého pro vytvoření nového testovacího prostředí.	Název větve ze sdíleného repozitáře verzovacího systému GIT, číslo sestavení a cesta k adresáři, který obsahuje terraform skripty.
2	create_variables_tfvars	Vytvoření souboru s proměnnými, které jsou využívané při spuštění terraform skriptů.	Název větve ze sdíleného repozitáře verzovacího systému GIT, číslo sestavení a cesta k adresáři, který obsahuje terraform skripty.
3	apply_terraform	Spuštění terraform skriptu, který vytvoří nové testovací prostředí a následně v něm spustí Uniqway aplikaci.	Cesta k adresáři, který obsahuje terraform skripty.
4	clean_up_generated_terraform_files	Odstranění vytvořených terraform souborů v krocích 1 a 2.	Cesta k adresáři, který obsahuje terraform skripty.

3.3.4.1 Vytvoření souboru init.tf použitého pro vytvoření nového testovacího prostředí

Jelikož může existovat více testovacích prostředí zároveň, tak je potřeba vytvořit unikátní init.tf soubor, při jehož spuštění dojde k vytvoření nového testovacího prostředí. K tomuto účelu slouží metoda *create_init_tf*, která přijímá jako parametry název větve ze sdíleného repozitáře verzovacího systému GIT, číslo sestavení a cestu k adresáři, ve kterém má být vytvořen nový init.tf soubor.

Metoda *create_init_tf* nejprve zkopíruje již existující soubor s názvem init.tf.example, který obsahuje šablonu skriptu init.tf. Tento terraform skript obsahuje konfiguraci s cestou k souboru, ve kterém je uchovávaná informace o aktuálním stavu zdrojů, které terraform spravuje. Tento soubor je uložen v Amazon S3 úložišti (viz sekce 1.4.9). Je tedy potřeba správně nastavit cestu k tomuto souboru. Šablona pro vytvoření init.tf souboru již obsahuje částečnou konfiguraci pro tento účel. Obrázek 3.23 tuto konfiguraci zobrazuje. Metoda *create_init_tf* tedy po vytvoření nového init.tf souboru upraví tento soubor tak, že pomocí funkce *sed* (použití příkazu *sed* je popsáno v sekci 3.3.3.4) nastaví místo <build_number> číslo sestavení a místo <branch_name> název větve ze sdíleného repozitáře verzovacího systému GIT, pro kterou má být nové testovací prostředí vytvořeno. Obě tyto hodnoty obdrží metoda *create_init_tf* jako

```
terraform {
  backend "s3" {
    key =
↪   "ecs-cluster/uniqplay/testing-<build_number>-<branch_name>/terraform.tfstate"
  }
}
```

Obrázek 3.23: Konfigurace pro načtení souboru s aktuálním stavem zdrojů spravovaných terraformem

```
profile = "uniqway"

branch-name = "<branch_name>"
build-id = "<build_number>"
```

Obrázek 3.24: Obsah souboru variables.tfvars

vstupní parametry.

3.3.4.2 Vytvoření souboru s proměnnými, které jsou využívané při spuštění terraform skriptů

Terraform skripty spouštěné pro vytvoření nového testovacího prostředí využívají proměnné, které jsou definované v souboru variables.tfvars. K jeho vytvoření slouží metoda *create_variables_tfvars*, která přijímá jako parametry název větve ze sdíleného repozitáře verzovacího systému GIT, číslo sestavení a cestu k adresáři, ve kterém má být vytvořen nový variables.tfvars soubor.

Metoda *create_variables_tfvars* nejprve zkopíruje již existující soubor s názvem variables.tfvars.example, který obsahuje šablonu pro vytvoření souboru variables.tfvars obsahujícímu potřebné konfigurační proměnné. Obsah vytvořeného souboru *variables.tfvars* je zobrazen na obrázku 3.24. Je tedy potřeba ještě zaměnit hodnotu <branch_name> za název větve ze sdíleného repozitáře verzovacího systému GIT, pro kterou má být nové testovací prostředí vytvořeno. Také je potřeba zaměnit hodnotu <build_number> za číslo sestavení. Obě tyto akce jsou provedeny pomocí funkce *sed*, která nahradí oba řetězce požadovanými hodnotami.

3.3.4.3 Spuštění terraform skriptu, který vytvoří nové testovací prostředí a následně v něm spustí Uniqway aplikaci

Pro vytvoření nového testovacího prostředí je využita metoda *apply_terraform*, která jako vstupní parametr přijímá cestu k adresáři, ve kterém má být spuštěn terraform skript. Tato metoda spustí již existující skript s názvem *deploy.sh*,

který je používaný pro správné nastavení stagingového a produkčního prostředí. Tento bashový skript jsem upravil tak, aby byl schopen vytvořit i testovací prostředí.

Skript nejprve inicializuje prostředí pro terraform pomocí příkazu *terraform init* spuštěného z adresáře, kde se nachází vytvořený soubor *init.tf*, viz sekce 3.3.4.1. Následně dojde k naimportování již existujících zdrojů, kterými jsou secrety v AWS Secrets Manageru (viz sekce 1.4.10) a které jsou společné pro všechna testovací prostředí. K tomuto účelu využije funkci terraformu *terraform import*.

Nakonec dojde ke spuštění dříve vytvořeného terraform skriptu *init.tf*, který se již postará o vytvoření celého testovacího prostředí a následné spuštění požadované verze backendové aplikace Uniqway. K tomuto účelu je využita funkce terraformu *terraform apply -auto-approve -var-file="variables.tfvars"*, jejíž přepínač *-var-file* obdrží dříve vytvořený soubor obsahující konfigurační proměnné, viz sekce 3.3.4.2.

Spouštěné terraform skripty již existovali a jsou používané pro správné nastavení stagingového a produkčního prostředí. Udělal jsem v nich potřebné změny, aby je bylo možné použít i pro vytvoření testovacího prostředí. Jelikož nejsem jediný autor použitých terraform skriptů, tak je v rámci této práce nebudu detailně popisovat. Popis zdrojů vytvořených pomocí terraform skriptu je v sekci 3.5.

3.3.4.4 Odstranění vytvořených terraform souborů

Při vytvoření nového testovacího prostředí jsou zároveň vytvářeny dva soubory - init.tf (viz sekce 3.3.4.1) a variables.tfvars (viz sekce 3.3.4.2). Jelikož již nejsou potřeba můžeme tyto dva soubory odstranit. K tomuto účelu slouží metoda *clean_up_generated_terraform_files*, která jako vstupní parametr přijímá cestu k adresáři, ve kterém byly tyto dva soubory vytvořeny.

3.3.5 Spuštění API testů proti testovací aplikaci běžící v nově vytvořeném testovacím prostředí

Krok pro spuštění API testů (viz sekce 1.1.1) v testovacím prostředí je zobrazen na obrázku 3.25. Tento krok tedy obsahuje následující tři hodnoty nastavení:

- name - Název daného kroku = run Rest API tests for testing
- workingDir - Adresář, ze kterého má být daný krok spuštěn = quality-assurance
- scriptContent - Spouštěný skript = ./run-tests-in-testing-environment.sh %teamcity.build.branch% %build.number%

```
script {
    name = "run Rest API tests for testing"
    workingDir = "quality-assurance"
    scriptContent = "./run-tests-in-testing-environment.sh
    ↪ %teamcity.build.branch% %build.number%"
}
```

Obrázek 3.25: Krok pro spuštění API testů v testovacím prostředí

Tabulka 3.8: Jednotlivé kroky použité při spuštění API testů v testovacím prostředí

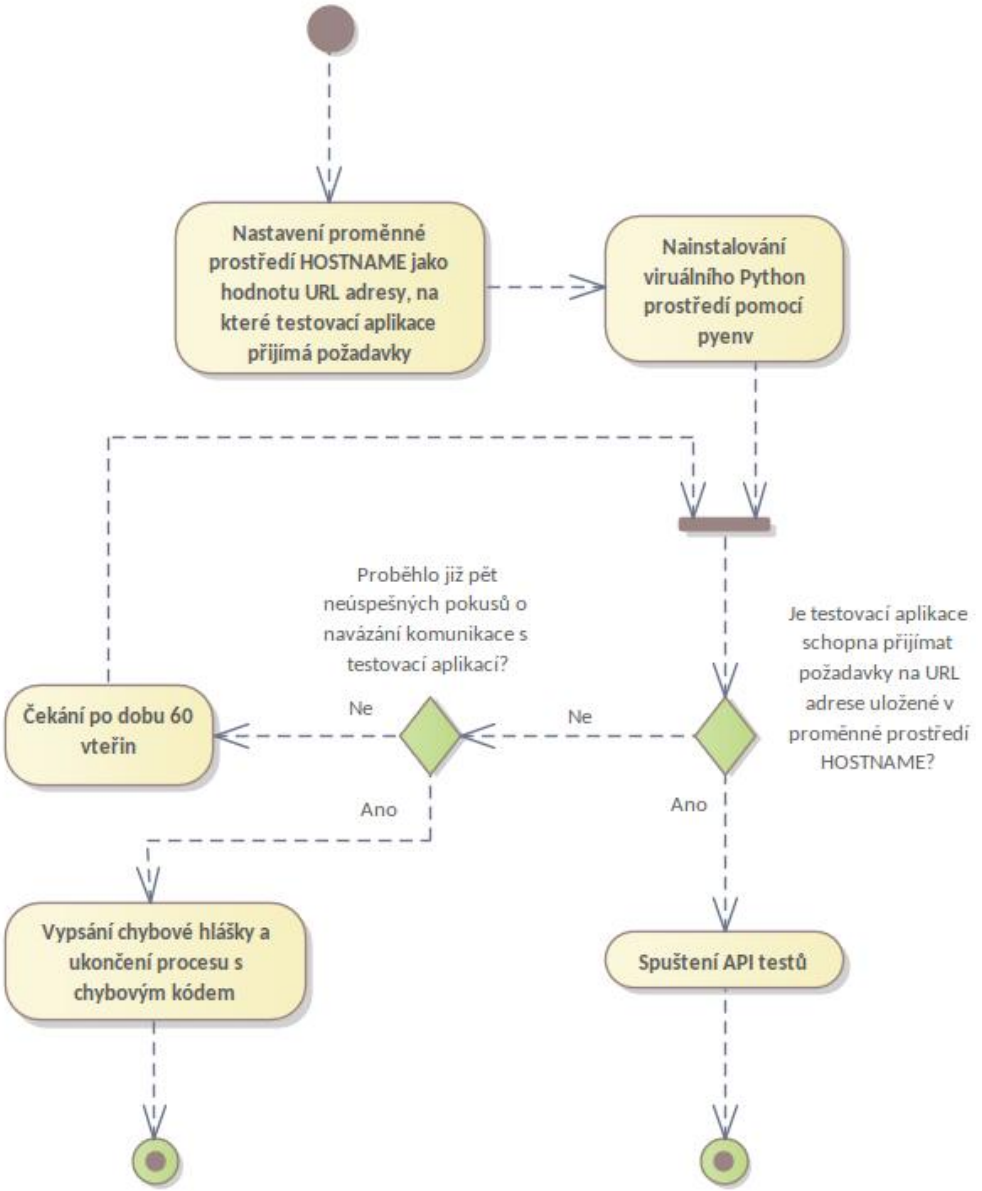
Pořadí	Název metody	Cíl	Vstupní parametry
1	setup_environment	Připravení prostředí pro běh API testů.	Název větve ze sdíleného repozitáře verzovacího systému GIT a číslo sestavení.
2	wait_for_service_available	Čekání, dokud nebude běžící aplikace v testovacím prostředí schopna přijímat požadavky.	
3	run_tests	Spuštění API testů.	

V tomto kroku dojde ke spuštění bashového skriptu s názvem *run-tests-in-testing-environment.sh* v adresáři *quality-assurance*. Tomuto skriptu jsou předány parametry *%teamcity.build.branch%* a *%build.number%*, kde *%teamcity.build.branch%* odpovídá názvu větve ze sdíleného repozitáře verzovacího systému GIT, pro kterou mají být testy spuštěny a *%build.number%* odpovídá číslu sestavení.

Jednotlivé kroky, které obsahuje skript *run-tests-in-testing-environment.sh* jsou stručně popsány v tabulce 3.8 a detailně popsány v následujících sekcích. Zároveň jsou jednotlivé kroky graficky zobrazeny na obrázku 3.26. Výsledky z běhu API testů je možné zobrazit přímo v Teamcity při zobrazení logů tohoto kroku.

3.3.5.1 Připravení prostředí pro běh API testů

Pro správný běh využívají API testy proměnné prostředí, které jsou popsané v tabulce 3.9. Většina proměnných má již nastavené hodnoty pomocí parametrů využitých v samotném Kotlin DSL skriptu (viz sekce 1.6.1). Pouze proměnná HOSTNAME není nastavená, jelikož její hodnota je vždy unikátní v závislosti na vytvořeném testovacím prostředí. Nastavení proměnné prostředí HOSTNAME zajišťuje metoda *setup_environment*, která přijímá jako parametry název větve ze sdíleného repozitáře verzovacího systému GIT a číslo sestavení, pro které mají být API testy spuštěny.



Obrázek 3.26: Jednotlivé kroky použité při spuštění API testů

Tabulka 3.9: Proměnné prostředí využité při spuštění API testů

Název proměnné	Nastavená hodnota	Popis
ENVIRONMENT	AWS	Prostředí, proti kterému se mají API testy spustit.
ADMIN_EMAIL	stgadmin@uniqway.cz	E-mail uživatele s administrátorskými právy.
CLIENT_EMAIL	stgclient@uniqway.cz	E-mail zákazníka využívaného v testech.
CLIENT_CUSTOMER_ID	1000001	ID zákazníka využívaného v testech.
PYTHONPATH	%teamcity.build.workingDir%	Cesta k adresáři, kde jsou umístěné Python knihovny.
LOCAL_ENVIRONMENT	false	Hodnota true/false označující, jestli se jedná o spuštění testů v lokálním prostředí.
HOSTNAME		URL na které testovací aplikace přijímá požadavky.

```
1 aws --profile uniqway --region eu-west-1 elbv2
  ↪ describe-load-balancers \
2         --names ${load_balancer_name} \
3         --output text \
4         --query 'LoadBalancers[0].DNSName'
```

Obrázek 3.27: Využití AWS funkce elbv2 describe-load-balancers pro získání DNS jména

Metoda nejprve vytvoří jméno load balanceru, který přijímá požadavky pro danou verzi aplikace v testovacím prostředí. Toto jméno je sestaveno stejně jako při vytváření samotného load balanceru, viz sekce 3.5.2. Následně je využita funkce AWS rozhraní příkazové řádky *elbv2 describe-load-balancers*, která využívá přepínač *--names*. Tento přepínač přijímá jako hodnotu název load balanceru, o kterém chceme zjistit informace. Následně pomocí přepínače *--query* vyfiltrujeme požadovanou hodnotu, kterou je DNS jméno, na kterém daný load balancer přijímá požadavky. Získaná hodnota je následně nastavena jako proměnná prostředí HOSTNAME. Použití funkce *elbv2 describe-load-balancers* je zobrazeno na obrázku 3.27.

Posledním úkolem metody *setup_environment* je instalace virtuálního prostředí pro Python. K tomuto kroku je využit nástroj *pipenv*, kterému je pouze předán příkaz *install*.

3.3.5.2 Čekání, dokud nebude běžící aplikace v testovacím prostředí schopna přijímat požadavky

Aplikace v nově vytvořeném prostředí nemusí být ihned po svém spuštění dostupná. K tomu abychom zajistili, že API testy budou spuštěny proti dostupné aplikaci slouží metoda *wait_for_service_available*, která kontroluje, zda je daná aplikace schopna přijímat požadavky. Tato metoda nejprve spustí Python test, který odešle požadavek na testovanou aplikaci a následně zkontroluje kód odpovědi. Pokud obdržený kód nemá hodnotu 200, tak víme, že aplikace není schopna přijímat požadavky a spuštěný test vrátí chybovou hodnotu. Metoda *wait_for_service_available* tuto hodnotu zkontroluje a vyhodnotí, jestli je aplikace schopna přijímat požadavky. Pokud ne, tak dojde k čekání po dobu 60 vteřin a k zopakování celé kontroly. V případě, že aplikace není dostupná i po páté iteraci této kontroly, dojde k vyhodnocení, že tato aplikace není správně spuštěna a skript se ukončí s chybovou hláškou a chybovým kódem. Aplikace je běžně dostupná již při první kontrole. Časový limit 60 vteřin s pěti opakováními by měl být při správném spuštění vždy více než dostačující.

3.3.5.3 Spuštění API testů

Spuštění API testů zajišťuje metoda *run_tests*. Tato metoda pouze spustí již existující Python testy pomocí příkazu *„pipenv run pytest tests;“*.

3.3.6 Odstranění všech vytvořených zdrojů pomocí terraformu

Krok pro odstranění zdrojů testovacího prostředí v AWS vytvořených pomocí terraformu je zobrazen na obrázku 3.28. Tento krok tedy obsahuje následující čtyři hodnoty nastavení:

- name - Název daného kroku = trigger destruction of resources created via terraform
- executionMode - Nastavení, že má vždy dojít ke spuštění tohoto kroku, i pokud kterýkoliv z předchozích kroků nebyl úspěšně dokončen = BuildStep.ExecutionMode.ALWAYS
- workingDir - Adresář, ze kterého má být daný krok spuštěn = infra/aws
- scriptContent - Spouštěný skript

Můžeme si všimnout, že tento krok využívá nastavení *executionMode* s hodnotou *BuildStep.ExecutionMode.ALWAYS*, jelikož může nastat situace, kdy některý z předchozích kroků nebyl dokončen správně. Například při spuštění API testů došlo k chybě. S tímto nastavením dojde vždy ke spuštění kroku, i pokud některý z předchozích kroků nebyl úspěšně dokončen.

```
script {
    name = "trigger destruction of resources created via
    ↪ terraform"
    executionMode = BuildStep.ExecutionMode.ALWAYS
    workingDir = "infra/aws"
    scriptContent = """
        if [[ %destroy_after_tests% == true ]]
        then
          ./remove-resources-in-testing-environment.sh
↪ %teamcity.build.branch% %build.number%;
        else
          echo "Destroy after tests is '%destroy_after_tests%'.
↪ Not removing created resources";
        fi;
    """.trimIndent()
}
```

Obrázek 3.28: Krok pro odstranění zdrojů testovacího prostředí v AWS vytvořených pomocí terraformu

Nejprve dojde k vyhodnocení, zda má být vytvořené testovací prostředí odstraněno po dokončení API testů. K tomuto účelu slouží podmínka kontrolující hodnotu proměnné *destroy_after_tests*, kterou je možné nastavit při spuštění vytvoření nového testovacího prostředí. Přednastavená hodnota této proměnné je vždy *true*. K odstranění zdrojů v AWS vytvořených pomocí terraformu dojde pouze pokud je hodnota proměnné *destroy_after_tests* rovna hodnotě *true*. V opačném případě dojde k přeskočení tohoto kroku.

Pokud mají být odstraněny vytvořené zdroje pomocí terraformu v AWS, tak dojde ke spuštění bashového skriptu s názvem *remove-resources-in-testing-environment.sh* v adresáři *infra/aws*. Tomuto skriptu jsou předány parametry *%teamcity.build.branch%* a *%build.number%*, kde *%teamcity.build.branch%* představuje název větve ze sdíleného repozitáře verzovacího systému GIT, pro kterou bylo vytvořeno testovací prostředí a *%build.number%* odpovídá číslu sestavení daného testovacího prostředí.

Jednotlivé kroky, které obsahuje skript *remove-resources-in-testing-environment.sh* jsou stručně popsány v tabulce 3.10 a detailně popsány v následujících sekcích. Dojde tedy k postupnému odstranění všech zdrojů, které byly pro testovací prostředí vytvořeny pomocí terraformu.

Tabulka 3.10: Jednotlivé kroky použité při odstranění zdrojů z AWS vytvořených pomocí terraformu

Pořadí	Název metody	Cíl	Vstupní parametry
1		Vytvoření přípony s jednoznačným identifikátorem testovacího prostředí.	
2	delete_service	Odstranění aplikace.	Přípona s jednoznačným identifikátorem testovacího prostředí.
3	deregister_task_definition	Odhlášení definice ECS tasku.	Přípona s jednoznačným identifikátorem testovacího prostředí.
4	delete_code_deploy_application	Odstranění code deploy aplikace.	Přípona s jednoznačným identifikátorem testovacího prostředí.
5	delete_load_balancer	Odstranění load balanceru.	Přípona s jednoznačným identifikátorem testovacího prostředí.
6	delete_target_group	Odstranění target group.	Předpona dané target group a přípona s jednoznačným identifikátorem testovacího prostředí.
7	delete_security_group	Odstranění security group.	Předpona dané security group a přípona s jednoznačným identifikátorem testovacího prostředí.
8	delete_s3_bucket_object	Odstranění souboru z S3 Bucketu s aktuálním stavem zdrojů, které spravuje terraform.	Název S3 Bucketu a cesta k danému souboru.
9	delete_s3_bucket	Odstranění user verification objektu z S3 Bucketu.	Předpona daného S3 Bucketu a přípona s jednoznačným identifikátorem testovacího prostředí.
10	detach_iam_policy_from_role	Odpojení vytvořených IAM Policies od IAM Roles.	Předpona ARN pro danou IAM Policy, předpona názvu dané IAM Policy, předpona názvu dané IAM Role a přípona s jednoznačným identifikátorem testovacího prostředí.
11	detach_aws_managed_iam_policy_from_role	Odpojení IAM Policies spravovaných AWS od IAM Roles.	ARN dané IAM Policy, předpona názvu dané IAM Role a přípona s jednoznačným identifikátorem testovacího prostředí.
12	delete_iam_role	Odstranění IAM Roles.	Předpona názvu dané IAM Role a přípona s jednoznačným identifikátorem testovacího prostředí.
13	delete_iam_policy	Odstranění IAM Policies.	Předpona ARN pro danou IAM Policy, předpona názvu dané IAM Policy a přípona s jednoznačným identifikátorem testovacího prostředí.

```
aws --profile uniqway --region eu-west-1 ecs delete-service \
    --cluster uniqway-cluster \
    --service ${service_name} \
    --force \
    --output text \
    --query "service.serviceArn"
```

Obrázek 3.29: Použití AWS CLI funkce ecs delete-service

3.3.6.1 Vytvoření přípony s jednoznačným identifikátorem testovacího prostředí

Název každého vytvořeného zdroje v AWS pomocí terraformu má stejnou koncovku. Proces jejího sestavení při odstranění testovacího prostředí je stejný, jako při vytváření tohoto prostředí. Detailní popis tohoto procesu se nachází v sekci 3.5.1.

3.3.6.2 Odstranění aplikace

O odstranění samotné aplikace se stará metoda *delete_service*, která jako vstupní parametr přijímá příponu s jednoznačným identifikátorem daného testovacího prostředí. Testovací Uniqway backend aplikace je spuštěna jako Docker container běžící jako Amazon ECS služba. K jejímu odstranění je využito AWS rozhraní příkazové řádky, konkrétně funkce *ecs delete-service*. Tato funkce využívá přepínače *--cluster*, který jako hodnotu přijíma název ECS clusteru se spuštěnou aplikací a *--service*, který jako hodnotu přijímá název spuštěné aplikace.

Testovací Uniqway backend aplikace je vždy spuštěna v clusteru s názvem „uniqway-cluster“. Název spuštěné aplikace je složen z první společné části „uniqplay-service-“ a následné unikátní části, která odpovídá jednoznačnému identifikátoru daného testovacího prostředí. Tento identifikátor obdrží metoda *delete_service* jako vstupní parametr. Použití funkce *ecs delete-service* je zobrazeno na obrázku 3.29.

3.3.6.3 Odhlášení definice ECS tasku

Každá Amazon ECS musí mít svou definici. K odstranění této definice je využita metoda *deregister_task_definition*, která jako vstupní parametr přijímá příponu s jednoznačným identifikátorem daného testovacího prostředí. ECS definice je následně odstraněna pomocí AWS rozhraní příkazové řádky, konkrétně funkce *ecs deregister-task-definition*. Tato funkce využívá přepínač *--task-definition*, který jako hodnotu příjíma název a verzi dané definice. Název definice je nejprve složen ze společné části „uniqplay-task-“ a následné unikátní části, která

```
local revision_number=1;
local status=0;
while [ $status == 0 ]
do
  aws --profile uniqway --region eu-west-1 ecs
  ↪  deregister-task-definition \
      --task-definition
      ↪  "${task_definition_name}:${revision_number}" \
      --output text \
      --query "taskDefinitions[0].[taskDefinitionArn]";

  local status=$?;
  revision_number=$((${revision_number} + 1));
done;
```

Obrázek 3.30: Proces odhlášení všech verzí definice ECS tasku

odpovídá jednoznačnému identifikátoru daného testovacího prostředí. Verze dané definice je poté připojena za složený název pomocí znaku dvojtečky.

Aby bylo zajištěno, že jsou odhlášeny všechny verze dané definice, tak se začnou postupně odstraňovat od verze 1. Hodnota verze se pro každý pokus o odstranění inkrementuje o hodnotu 1. Pokud dojde k vrácení chybové hodnoty, tak víme, že daná verze definice již neexistuje a můžeme tedy celý tento proces ukončit. Proces odhlášení všech verzí dané definice je zobrazen na obrázku 3.30.

3.3.6.4 Odstranění code deploy aplikace

Při vytváření nového testovacího prostředí vždy dojde také k vytvoření služby code deploy. Tato služba má za úkol automatické nasazení nové verze aplikace do tohoto prostředí, viz sekce 1.4.3. K odstranění této služby je využita metoda *delete_code_deploy_application*, která jako vstupní parametr přijímá příponu s jednoznačným identifikátorem daného testovacího prostředí. Code deploy služba je následně odstraněna pomocí AWS rozhraní příkazové řádky, konkrétně funkce *deploy delete-application*. Tato funkce využívá přepínač *--application-name*, který jako hodnotu přijímá název dané code deploy služby. Název této služby je vždy složen z první společné části „uniqplay-“ a následné unikátní části, která odpovídá jednoznačnému identifikátoru daného testovacího prostředí. Použití funkce *deploy delete-application* je zobrazeno na obrázku 3.31.

```
aws --profile uniqway --region eu-west-1 deploy
↪  delete-application \
    --application-name ${code_deploy_application_name};
```

Obrázek 3.31: Použití AWS CLI funkce deploy delete-application

```
local load_balancer_arn=$(aws --profile uniqway --region
↪  eu-west-1 elbv2 describe-load-balancers \
    --names ${load_balancer_name} \
    --output text \
    --query "LoadBalancers[0].[LoadBalancerArn]");

aws --profile uniqway --region eu-west-1 elbv2
↪  delete-load-balancer \
    --load-balancer-arn ${load_balancer_arn};
```

Obrázek 3.32: Použití AWS CLI funkcí elbv2 describe-load-balancers a elbv2 delete-load-balancer

3.3.6.5 Odstranění load balanceru

Přístup k testovací aplikaci Uniqway je řízen pomocí load balanceru, viz sekce 3.5.2. K odstranění tohoto load balanceru je využita metoda *delete_load_balancer*, která jako vstupní parametr přijímá příponu s jednoznačným identifikátorem daného testovacího prostředí. Pro odstranění load balanceru je nutné znát jeho ARN. Tento ARN získáme pomocí AWS rozhraní příkazové řádky, konkrétně funkce *elbv2 describe-load-balancers*, která přijímá přepínač *--names* s hodnotou odpovídající názvu daného load balanceru. Název testovacího load balanceru se skládá z první společné části „lb-tst-“ a následné unikátní části, která odpovídá jednoznačnému identifikátoru daného testovacího prostředí.

Po získání ARN daného load balanceru jej můžeme odstranit. K tomuto účelu je využito AWS rozhraní příkazové řádky, konkrétně funkce *elbv2 delete-load-balancer*, která přijímá přepínač *--load-balancer-arn* s hodnotou odpovídající ARN hodnotě daného load balanceru. Využití obou použitých funkcí je zobrazeno na obrázku 3.32.

3.3.6.6 Odstranění target group

Target group přeposílá požadavky obdržené z load balanceru do aplikace, viz sekce 1.4.6. Při vytváření nového testovacího prostředí jsou vždy vytvořeny dvě target group, viz sekce 3.5.3. K jejich odstranění je využita metoda *delete_target_group*, která jako vstupní parametry přijímá předponu názvu dané target groupy a příponu s jednoznačným identifikátorem daného testovacího

```
1 local target_group_arn=$(aws --profile uniqway --region
  ↪ eu-west-1 elbv2 describe-target-groups \
2     --names ${target_group_name} \
3     --output text \
4     --query "TargetGroups[0].[TargetGroupArn]");
5
6 aws --profile uniqway --region eu-west-1 elbv2
  ↪ delete-target-group \
7     --target-group-arn ${target_group_arn};
```

Obrázek 3.33: Použití AWS CLI funkcí elbv2 describe-target-groups a elbv2 delete-target-group

prostředí. Tato metoda je tedy využita dvakrát, jednou pro každou target group.

V metodě *delete_target_group* je nejprve sestaven název target groupy, která má být odstraněna. Tento název je sestaven stejným způsobem jako při vytváření dané target groupy, viz sekce 3.5.3. Pro odstranění target groupy je nutné znát její ARN. Tento ARN získáme pomocí AWS rozhraní příkazové řádky, konkrétně funkce *elbv2 describe-target-groups*, která přijímá přepínač *--names* s hodnotou odpovídající názvu dané target groupy.

Po získání ARN dané target groupy ji můžeme odstranit. K tomuto účelu je využito AWS rozhraní příkazové řádky, konkrétně funkce *elbv2 delete-target-group*, která přijímá přepínač *--target-group-arn* s hodnotou odpovídající ARN hodnotě dané target groupy. Využití obou použitých funkcí je zobrazeno na obrázku 3.33.

3.3.6.7 Odstranění security group

Security group kontroluje přístup z insternetu, viz sekce 1.4.8. Při vytváření nového testovacího prostředí jsou vždy vytvořeny dvě security group, viz sekce 3.5.9. K jejich odstranění je využita metoda *delete_security_group*, která jako vstupní parametry přijímá předponu názvu dané security groupy a příponu s jednoznačným identifikátorem daného testovacího prostředí. Tato metoda je tedy využita dvakrát, jednou pro každou security group.

V metodě *delete_security_group* je nejprve sestaven název security groupy, která má být odstraněna. Tento název je sestaven stejným způsobem jako při vytváření dané target groupy, viz sekce 3.5.9. Pro odstranění security groupy je nutné znát její identifikátor. Tento indentifikátor získáme pomocí AWS rozhraní příkazové řádky, konkrétně funkce *ec2 describe-security-groups*, která přijímá přepínač *--filter*. Tento přepínač přijímá řetězec se dvěma hodnotami, „Name“ a „Values“. V hodnotě „Name“ předáme parametr podle kterého chceme vyfiltrovat danou security group. V našem případě chceme vyfiltrovat

```
1 local security_group_id=$(aws --profile uniqway --region
  ↪ eu-west-1 ec2 describe-security-groups \
2     --filter "Name=group-name,Values=${security_group_name}"\
3     --output text \
4     --query "SecurityGroups[0].[GroupId]");
5
6 aws --profile uniqway --region eu-west-1 ec2
  ↪ delete-security-group \
7     --group-id ${security_group_id};
```

Obrázek 3.34: Použití AWS CLI funkcí ec2 describe-security-groups a ec2 delete-security-group

security group podle názvu, tedy „group-name“. V hodnotě „Values“ předáme možné hodnoty filtru. V našem případě se jedná o celý název dané security group.

Po získání identifikátoru dané security group ji můžeme odstranit. K tomuto účelu je využito AWS rozhraní příkazové řádky, konkrétně funkce *ec2 delete-security-group*, která přijímá přepínač *--group-id* s hodnotou odpovídající hodnotě identifikátoru dané security group. Využití obou použitých funkcí je zobrazeno na obrázku 3.34.

3.3.6.8 Odstranění souboru z S3 Bucketu s aktuálním stavem zdrojů, které spravuje terraform

Terraform si ukládá aktuální stav všech zdrojů, které spravuje, viz 1.5. V případě Uniqway je tento stav uložen v souboru, který se nachází v Amazon S3 Bucketu. Odstranění tohoto souboru zajišťuje metoda *delete_s3_bucket_object*, která jako vstupní parametry přijímá název Amazon S3 Bucketu a cestu k danému souboru. Uniqway k uložení terraform stavového souboru využívá Amazon S3 bucket s názvem „uniqway-terraform-states“. Stavový soubor s názvem „terraform.tfstate“ je uložen v adresáři „ecs-cluster/uniqplay/testing-<BUILD_NUMBER>-<BRANCH_NAME>“, kde <BUILD_NUMBER> odpovídá číslu sestavení daného testovacího prostředí a <BRANCH_NAME> odpovídá názvu větve ze sdíleného repozitáře verzovacího systému GIT, pro kterou bylo testovací prostředí vytvořeno. K odstranění souboru je využito AWS rozhraní příkazové řádky, konkrétně funkce *s3api delete-object*, která přijímá přepínače *--bucket* s hodnotou odpovídající názvu Amazon S3 Bucketu a *--key* s hodnotou odpovídající cestě, na které je daný soubor uložen. Použití funkce *s3api delete-object* je zobrazeno na obrázku 3.35.

```
aws --profile uniqway --region eu-west-1 s3api delete-object \
    --bucket ${bucket_name} \
    --key ${object_key};
```

Obrázek 3.35: Použití AWS CLI funkce s3api delete-object

```
aws --profile uniqway --region eu-west-1 s3api delete-bucket \
    --bucket ${bucket_name};
```

Obrázek 3.36: Použití AWS CLI funkce s3api delete-bucket

3.3.6.9 Odstranění user verification objektu z S3 Bucketu

Uniqway backend aplikace k ověření uživatele využívá objekty uložené v Amazon S3 Bucketu. Každé testovací prostředí má svůj vlastní bucket, můžeme jej tedy celý odstranit. K jeho odstranění je využita metoda *delete_s3_bucket*, která jako vstupní parametry přijímá předponu názvu daného Amazon S3 Bucketu a příponu s jednoznačným identifikátorem daného testovacího prostředí. Nejprve dojde k sestavení názvu Amazon S3 Bucketu. Tento název je sestaven stejným způsobem, jako při vytváření daného Amazon S3 Bucketu, viz sekce 3.5.6. Následně dojde k jeho odstranění pomocí AWS rozhraní příkazové řádky, konkrétně funkce *s3api delete-bucket*, která přijímá přepínač *--bucket*, s hodnotou odpovídající názvu daného Amazon S3 Bucketu. Využití této funkce je zobrazeno na obrázku 3.36.

3.3.6.10 Odpojení vytvořených IAM Policies od IAM Roles

Aby bylo možné odstranit vytvořené IAM Policies a IAM Roles, tak je neprve potřeba je od sebe odpojit. K tomuto účelu je využita metoda *detach_iam_policy_from_role*, která jako vstupní parametry přijímá ARN předponu IAM Policy, předponu názvu IAM Policy, předponu názvu IAM Role a příponu s jednoznačným identifikátorem daného testovacího prostředí.

Nejprve dojde k sestavení názvu dané IAM Policy. Tento název je sestaven stejným způsobem, jako při vytváření dané IAM Policy, viz sekce 3.5.7. Následně je pomocí tohoto názvu sestaven ARN dané IAM Policy jako „<ARN_PREFIX>/<NAME>“, kde <ARN_PREFIX> odpovídá předponě ARNu dané IAM Policy a <NAME> odpovídá názvu dané IAM Policy. Poté dojde k sestavení názvu dané IAM Role. Tento název je sestaven stejným způsobem, jako při vytváření dané IAM Role, viz sekce 3.5.8.

K odpojení IAM Policy od IAM Role je využito AWS rozhraní příkazové řádky, konkrétně funkce *iam detach-role-policy*, která přijímá přepínače *--role-name* s hodnotou odpovídající názvu dané IAM Role a *--policy-arn* s hodno-

```
aws --profile uniqway --region eu-west-1 iam detach-role-policy \
    --role-name ${role_name} \
    --policy-arn ${policy_arn};
```

Obrázek 3.37: Použití AWS CLI funkce iam detach-role-policy

```
aws --profile uniqway --region eu-west-1 iam delete-role \
    --role-name ${role_name};
```

Obrázek 3.38: Použití AWS CLI funkce iam delete-role

tou odpovídající ARN dané IAM Policy. Využití této funkce je zobrazeno na obrázku 3.37.

3.3.6.11 Odpojení IAM Policies spravovaných AWS od IAM Roles

Backendový projekt Uniqway využívá tři IAM Policies spravované přímo Amazonem. Tyto policies jsou „AWSCodeDeployRole“, „AmazonEC2ContainerServiceAutoscaleRole“ a „AmazonECSTaskExecutionRolePolicy“. Všechny tyto policies musejí být odpojeny od IAM Rolí, které je využívají. K tomuto účelu slouží metoda *detach_aws_managed_iam_policy_from_role*, která jako vstupní parametry přijímá ARN IAM Policy, předponu názvu IAM Role a příponu s jednoznačným identifikátorem daného testovacího prostředí. Nejprve dojde k sestavení názvu dané IAM Role. Tento název je sestaven stejným způsobem, jako při vytváření dané IAM Role, viz sekce 3.5.8.

K odpojení IAM Policy od IAM Role je využito AWS rozhraní příkazové řádky, konkrétně funkce *iam detach-role-policy*, která přijímá přepínače *--role-name* s hodnotou odpovídající názvu dané IAM Role a *--policy-arn* s hodnotou odpovídající ARN dané IAM Policy. Využití této funkce je zobrazeno na obrázku 3.37.

3.3.6.12 Odstranění IAM Roles

K odstranění vytvořených IAM Roles je využita metoda *delete_iam_role*, která jako vstupní parametry přijímá předponu názvu dané IAM Role a příponu s jednoznačným identifikátorem daného testovacího prostředí. Nejprve dojde k sestavení názvu dané IAM Role. Tento název je sestaven stejným způsobem, jako při vytváření dané IAM Role, viz sekce 3.5.8. Následně dojde k jejímu odstranění pomocí AWS rozhraní příkazové řádky, konkrétně funkce *iam delete-role*, která přijímá přepínač *--role-name* s hodnotou odpovídající názvu dané IAM Role. Využití této funkce je zobrazeno na obrázku 3.38.

```
aws --profile uniqway --region eu-west-1 iam delete-policy \
    --policy-arn ${policy_arn};
```

Obrázek 3.39: Použití AWS CLI funkce iam delete-policy

```
script {
    name = "trigger application-conf secret destruction"
    executionMode = BuildStep.ExecutionMode.ALWAYS
    workingDir = "infra/aws"
    scriptContent = """
        if [[ %destroy_after_tests% == true ]]
        then
          ./application-conf-testing.sh delete
↪   %teamcity.build.branch% %build.number%;
        else
          echo "Destroy after tests is '%destroy_after_tests%'.
↪   Not removing created secret";
        fi;
    """.trimIndent()
}
```

Obrázek 3.40: Krok pro odstranění application-conf testovacího secretu

3.3.6.13 Odstranění IAM Policies

K odstranění vytvořených IAM Policies je využita metoda *delete_iam_policy*, která jako vstupní parametry přijímá předponu ARN dané IAM Policy, předponu názvu dané IAM Policy a příponu s jednoznačným identifikátorem daného testovacího prostředí. Nejprve dojde k sestavení názvu dané IAM Policy. Tento název je sestaven stejným způsobem, jako při vytváření dané IAM Policy, viz sekce 3.5.7. Následně je pomocí tohoto názvu stestaven ARN dané IAM Policy jako „<ARN_PREFIX>/<NAME>“, kde <ARN_PREFIX> odpovídá předponě ARNu dané IAM Policy a <NAME> odpovídá názvu dané IAM Policy. Poté dojde k jejímu odstranění pomocí AWS rozhraní příkazové řádky, konkrétně funkce *iam delete-policy*, která přijímá přepínač *--policy-arn* s hodnotou odpovídající ARN hodnotě dané IAM Policy. Využití této funkce je zobrazeno na obrázku 3.39.

3.3.7 Odstranění testovacího application-conf secretu

Krok pro odstranění testovacího application-conf secretu je zobrazen na obrázku 3.40. Tento krok tedy obsahuje následující čtyři hodnoty nastavení:

Tabulka 3.11: Jednotlivé kroky použité při odstranění vytvořeného secretu application-conf pro testovací prostředí

Pořadí	Název metody	Cíl	Vstupní parametry
1		Sestavení přípony testovacího application-conf secretu.	
2	delete_secret	Odstranění vytvořeného application-conf secretu.	Přípona testovacího application-conf secretu.

- name - Název daného kroku = trigger application-conf secret destruction
- executionMode - Nastavení, že má vždy dojít ke spuštění tohoto kroku, i pokud kterýkoliv z předchozích kroků nebyl úspěšně dokončen = BuildStep.ExecutionMode.ALWAYS
- workingDir - Adresář, ze kterého má být daný krok spuštěn = infra/aws
- scriptContent - Spouštěný skript

Můžeme si všimnout, že tento krok využívá nastavení *executionMode* s hodnotou *BuildStep.ExecutionMode.ALWAYS*, aby vždy došlo k jeho spuštění. Detailní popis tohoto nastavení se nachází v sekci 3.3.6.

Nejprve dojde k vyhodnocení, zda má být vytvořený testovací application-conf secret odstraněn po dokončení API testů. K tomuto účelu slouží podmínka kontrolující hodnotu proměnné *destroy_after_tests*. Detailní popis této kontroly se nachází v sekci 3.3.6.

Pokud má být odstraněn vytvořený testovací application-conf secret, tak dojde ke spuštění bashového skriptu s názvem *application-conf-testing.sh* v adresáři *infra/aws*. Tomuto skriptu jsou předány parametry *delete*, *%teamcity.build.branch%* a *%build.number%*, kde *delete* označuje akci, že má být testovací application-conf secret odstraněn, *%teamcity.build.branch%* představuje název větve ze sdíleného repozitáře verzovacího systému GIT, pro kterou bylo vytvořeno testovací prostředí a *%build.number%* odpovídá číslu sestavení daného testovacího prostředí.

Jednotlivé kroky, které obsahuje skript *application-conf-testing.sh* jsou stručně popsány v tabulce 3.11 a detailně popsány v následujících sekcích.

3.3.7.1 Sestavení přípony testovacího application-conf secretu

Sestavení přípony testovacího application-conf secretu při jeho odstranění je totožné jako při jeho vytvoření. Celý postup je tedy popsán v sekci 3.3.3.1.

```
aws --profile uniqway --region eu-west-1 secretsmanager delete-secret \
    --secret-id "application-conf-testing-${application_conf_postfix}" \
    --force-delete-without-recovery
```

Obrázek 3.41: Použití AWS CLI funkce secretsmanager delete-secret

```
script {
    name = "trigger DB instance and cluster destruction"
    executionMode = BuildStep.ExecutionMode.ALWAYS
    workingDir = "infra/aws"
    scriptContent = """
        if [[ %destroy_after_tests% == true ]]
        then
          ./delete-aurora-db-instance.sh db-instance-tst
↪   %teamcity.build.branch% %build.number%
        else
          echo "Destroy after tests is '%destroy_after_tests%'.
↪   Not removing created database";
        fi;
    """.trimIndent()
}
```

Obrázek 3.42: Krok pro odstranění testovacího databázového clusteru a instance

3.3.7.2 Odstranění vytvořeného application-conf secretu

K odstranění testovacího application-conf secretu je využita metoda *delete_secret*, která jako vstupní parametr přijímá příponu daného testovacího application-conf secretu. Tato metoda využívá AWS rozhraní příkazové řádky, konkrétně funkci *secretsmanager delete-secret*. Tato funkce využívá přepínač *--secret-id*, který jako hodnotu přijíma název secretu, který má být odstraněn. Použití funkce *secretsmanager delete-secret* je zobrazeno na obrázku 3.41.

3.3.8 Odstranění testovací databáze a testovacího databázového clusteru

Krok pro odstranění testovacího databázového clusteru a instance je zobrazen na obrázku 3.42. Tento krok tedy obsahuje následující čtyři hodnoty nastavení:

- name - Název daného kroku = trigger DB instance and cluster destruction
- executionMode - Nastavení, že má vždy dojít ke spuštění tohoto kroku, i pokud kterýkoliv z předchozích kroků nebyl úspěšně dokončen = BuildStep.ExecutionMode.ALWAYS
- workingDir - Adresář, ze kterého má být daný krok spuštěn = infra/aws
- scriptContent - Spouštěný skript

Můžeme si všimnout, že tento krok využívá nastavení *executionMode* s hodnotou *BuildStep.ExecutionMode.ALWAYS*, aby vždy došlo k jeho spuštění. Detailní popis tohoto nastavení se nachází v sekci 3.3.6.

Nejprve dojde k vyhodnocení, zda mají být testovací databázový cluster a instance odstraněny po dokončení API testů. K tomuto účelu slouží podmínka kontrolující hodnotu proměnné *destroy_after_tests*. Detailní popis této kontroly se nachází v sekci 3.3.6.

Pokud mají být odstraněny testovací databázový cluster a instance, tak dojde ke spuštění bashového skriptu s názvem *delete-aurora-db-instance.sh* v adresáři *infra/aws*. Tomuto skriptu jsou předány parametry *db-instance-tst*, *%teamcity.build.branch%* a *%build.number%*, kde *db-instance-tst* odpovídá předponě testovací databázové instance, *%teamcity.build.branch%* představuje název větve ze sdíleného repozitáře verzovacího systému GIT, pro kterou bylo vytvořeno testovací prostředí a *%build.number%* odpovídá číslu sestavení daného testovacího prostředí.

Jednotlivé kroky, které obsahuje skript *delete-aurora-db-instance.sh* jsou stručně popsány v tabulce 3.12 a detailně popsány v následujících sekcích.

3.3.8.1 Sestavení názvu testovací databázové instance

Sestavení názvu testovací databázové instance při jejím odstranění je totožné jako při jejím vytvoření. Celý postup je tedy popsán v sekci 3.3.2.1.

3.3.8.2 Kontrola, zda se jedná opravdu o testovací databázovou instanci

Před samotným odstraněním testovací databázové instance a clusteru dochází nejprve ke kontrole, zda se opravdu jedná o testovací databázovou instanci

Tabulka 3.12: Jednotlivé kroky použité při odstranění testovacího databázového clusteru a instance

Pořadí	Název metody	Cíl	Vstupní parametry
1		Sestavení názvu testovací databázové instance.	
2	check_if_testing_instance	Kontrola, zda se jedná opravdu o testovací databázovou instanci.	Název testovací databázové instance.
3	delete_instance	Odstranění testovací databázové instance.	Název testovací databázové instance.
4	wait_for_deleted_instance_status	Čekání, dokud nedojde k úplnému odstranění testovací databázové instance.	Název testovací databázové instance.
4	delete_cluster_if_zero_instances	Odstranění testovacího databázového clusteru, pokud neobsahuje žádné databázové instance.	

a cluster. K této kontrole dochází kvůli tomu, aby omylem nedošlo k odstranění databáze z jiného, než testovacího prostředí. Ke kontrole je využita metoda *check_if_testing_instance*, která jako vstupní parametr přijímá název databázové instance, která má být odstraněna. Tato metoda zkontroluje, zda obdržený název začíná řetězcem „db-instance-tst“, kterým musí začínat každá testovací databázová instance. Pokud tato podmínka není splněna, tak dojde k vypsání chybové hlášky a ukončení celého procesu.

3.3.8.3 Odstranění testovací databázové instance

K odstranění testovací databázové instance je využita metoda *delete_instance*, která jako vstupní parametr přijímá název dané testovací databázové instance. Tato metoda využívá AWS rozhraní příkazové řádky, konkrétně funkci *rds delete-db-instance*. Tato funkce využívá přepínače *--db-instance-identifier*, který jako hodnotu příjíma název databázové instance, která má být odstraněna a *--query*, který přijímá hodnotu „DBInstance.[DBClusterIdentifier]“, pomocí nějž dojde k vyhledání identifikátoru databázového clusteru, ve kterém se daná databázová instance nacházela. Hodnota identifikátoru databázového clusteru je následně uložena do proměnné „cluster_identifier“, která je následně využita pro jeho odstranění, viz sekce 3.3.8.5. Použití funkce *rds delete-db-instance* je zobrazeno na obrázku 3.43.

```
cluster_identifier=$(aws --profile uniqway --region eu-west-1
↪  rds delete-db-instance \
    --db-instance-identifier ${instance_name} \
    --output text \
    --query "DBInstance.[DBClusterIdentifier]");
```

Obrázek 3.43: Použití AWS CLI funkce rds delete-db-instance

```
aws --profile uniqway --region eu-west-1 rds wait db-instance-deleted \
    --db-instance-identifier ${instance_name}
```

Obrázek 3.44: Použití AWS CLI funkce rds wait db-instance-deleted

3.3.8.4 Čekání, dokud nedojde k úplnému odstranění testovací databázové instance

Při odstranění databázové instance nedojde k této akci okamžitě, proto je potřeba vyčkat, dokud databázová instance není opravdu kompletně odstraněna. K tomuto kroku je využita metoda *wait_for_deleted_instance_status*, která jako vstupní parametr přijímá název dané testovací databázové instance. Tato metoda využívá AWS rozhraní příkazové řádky, konkrétně funkci *rds wait db-instance-deleted*. Tato funkce využívá přepínač *--db-instance-identifier*, který jako hodnotu příjíma název databázové instance. Tato funkce zajistí, že dojde k čekání, dokud nebude daná databázová instance kompletně odstraněna. Použití funkce *rds wait db-instance-deleted* je zobrazeno na obrázku 3.44.

3.3.8.5 Odstranění testovacího databázového clusteru, pokud neobsahuje žádné databázové instance

Po odstranění testovací databázové instance stále ještě existuje testovací databázový cluster, ve kterém se daná instance nacházela. K jeho odstranění je využita metoda *delete_cluster_if_zero_instances*. Tato metoda využívá AWS rozhraní příkazové řádky, konkrétně funkci *rds delete-db-cluster*. Tato funkce využívá přepínače *--db-cluster-identifier*, který jako hodnotu příjíma identifikátor databázového clusteru (tento identifikátor získáme při odstranění testovací databázové instance, viz sekce 3.3.8.3) a *--skip-final-snapshot*, který zabrání vytvoření snapshotu daného databázového clusteru. Použití funkce *rds delete-db-cluster* je zobrazeno na obrázku 3.45.

```
aws --profile uniqway --region eu-west-1 rds delete-db-cluster \
    --db-cluster-identifier "${cluster_identifier}" \
    --skip-final-snapshot \
    --output text \
    --query "DBCluster[0].[DBClusterArn]"
```

Obrázek 3.45: Použití AWS CLI funkce rds delete-db-cluster

Tabulka 3.13: Jednotlivé kroky použité při odstranění existujícícho testovacího prostředí

Pořadí	Název	Cíl
1	trigger destruction of resources created via terraform	Odstranění všech vytvořených zdrojů pro dané testovací prostředí v AWS kromě application-conf secretu a testovací databáze.
2	trigger application-conf secret destruction	Odstranění application-conf secretu vytvořeného pro dané testovací prostředí.
3	trigger DB instance and cluster destruction	Odstranění vytvořené testovací databázové instance a databázového clusteru pro dané testovací prostředí.

3.4 Proces odstranění existujícího testovacího prostředí

Při vytváření nového testovacího prostředí si uživatel může zvolit možnost, že dané testovací prostředí nemá být odstraněno po dokončení API testů. Kvůli této možnosti musel být vytvořen proces, který umožní odstranit již existující testovací prostředí. Kroky pro odstranění existujícího testovacího prostředí jsou stručně popsány v tabulce 3.13 a detailně popsány v následujících sekcích.

Tento proces využívá dvě systémové proměnné „branch_name“ a „testing_build_number“, které musí uživatel při jeho spuštění vyplnit. Systémová proměnná „branch_name“ odpovídá názvu větve ze sdíleného repozitáře verzovacího systému GIT, pro kterou bylo dané testovací prostředí vytvořeno a „testing_build_number“ odpovídá číslu sestavení daného testovacího prostředí.

```
script {
    name = "trigger destruction of resources created via
    ↪ terraform"
    executionMode = BuildStep.ExecutionMode.ALWAYS
    workingDir = "infra/aws"
    scriptContent =
    ↪ "./remove-resources-in-testing-environment.sh
    ↪ %branch_name% %testing_build_number%"
}
```

Obrázek 3.46: Krok pro odstranění zdrojů testovacího prostředí v AWS kromě application-conf testovacího secretu a testovací databáze

3.4.1 Odstranění všech vytvořených zdrojů pro dané testovací prostředí v AWS kromě application-conf secretu a testovací databáze

Krok pro odstranění zdrojů testovacího prostředí v AWS kromě application-conf secretu a testovací databáze je zobrazen na obrázku 3.46. Tento krok tedy obsahuje následující čtyři hodnoty nastavení:

- name - Název daného kroku = trigger destruction of resources created via terraform
- executionMode - Nastavení, že má vždy dojít ke spuštění tohoto kroku, i pokud kterýkoliv z předchozích kroků nebyl úspěšně dokončen = BuildStep.ExecutionMode.ALWAYS
- workingDir - Adresář, ze kterého má být daný krok spuštěn = infra/aws
- scriptContent - Spouštěný skript = ./remove-resources-in-testing-environment.sh %branch_name% %testing_build_number%

Můžeme si všimnout, že tento krok využívá nastavení *executionMode* s hodnotou *BuildStep.ExecutionMode.ALWAYS*, jelikož může nastat situace, kdy některý z předchozích kroků nebyl úspěšně dokončen. Dojde tedy ke spuštění bashového skriptu s názvem *remove-resources-in-testing-environment.sh* v adresáři *infra/aws*. Tomuto skriptu jsou předány parametry *%branch_name%* a *%testing_build_number%*, kde *%branch_name%* představuje název větve ze sdíleného repozitáře verzovacího systému GIT, pro kterou bylo vytvořeno testovací prostředí a *%testing_build_number%* odpovídá číslu sestavení daného testovacího prostředí. Jedná se o ten samý skript, který je spouštěný při automatickém odstranění AWS zdrojů testovacího prostředí po dokončení API testů. Jednotlivé kroky v tomto skriptu jsou již popsány v sekci 3.3.6.

```
script {
    name = "trigger application-conf secret destruction"
    executionMode = BuildStep.ExecutionMode.ALWAYS
    workingDir = "infra/aws"
    scriptContent = "./application-conf-testing.sh delete
    ↪  %branch_name% %testing_build_number%"
}
```

Obrázek 3.47: Krok pro odstranění testovacího application-conf secretu

3.4.2 Odstranění application-conf secretu vytvořeného pro dané testovací prostředí

Krok pro odstranění testovacího application-conf secretu je zobrazen na obrázku 3.47. Tento krok tedy obsahuje následující čtyři hodnoty nastavení:

- name - Název daného kroku = trigger application-conf secret destruction
- executionMode - Nastavení, že má vždy dojít ke spuštění tohoto kroku, i pokud kterýkoliv z předchozích kroků nebyl úspěšně dokončen = BuildStep.ExecutionMode.ALWAYS
- workingDir - Adresář, ze kterého má být daný krok spuštěn = infra/aws
- scriptContent - Spouštěný skript = ./application-conf-testing.sh delete %branch_name% %testing_build_number%

Můžeme si všimnout, že tento krok využívá nastavení *executionMode* s hodnotou *BuildStep.ExecutionMode.ALWAYS*, jelikož může nastat situace, kdy některý z předchozích kroků nebyl úspěšně dokončen. Dojde tedy ke spuštění bashového skriptu s názvem *application-conf-testing.sh* v adresáři *infra/aws*. Tomuto skriptu jsou předány parametry *delete*, *%testing_build_number%* a *%branch_name%*, kde *delete* představuje akci, že má dojít k odstranění daného secretu, *%branch_name%* představuje název větve ze sdíleného repozitáře verzovacího systému GIT, pro kterou bylo dané testovací prostředí vytvořeno a *%testing_build_number%* odpovídá číslu sestavení daného testovacího prostředí. Jedná se o ten samý skript, který je spouštěný při automatickém odstranění testovacího application-conf secretu po dokončení API testů. Jednotlivé kroky v tomto skriptu jsou již popsány v sekci 3.3.7.

```
script {
    name = "trigger DB instance and cluster destruction"
    executionMode = BuildStep.ExecutionMode.ALWAYS
    workingDir = "infra/aws"
    scriptContent = "./delete-aurora-db-instance.sh
    ↪  db-instance-tst %branch_name% %testing_build_number%"
}
```

Obrázek 3.48: Krok pro odstranění testovací databázové instance a testovacího databázového clusteru

3.4.3 Odstranění vytvořené testovací databázové instance a databázového clusteru pro dané testovací prostředí

Krok pro odstranění testovací databázové instance a testovacího databázového clusteru je zobrazen na obrázku 3.48. Tento krok tedy obsahuje následující čtyři hodnoty nastavení:

- name - Název daného kroku = trigger DB instance and cluster destruction
- executionMode - Nastavení, že má vždy dojít ke spuštění tohoto kroku, i pokud kterýkoliv z předchozích kroků nebyl úspěšně dokončen = BuildStep.ExecutionMode.ALWAYS
- workingDir - Adresář, ze kterého má být daný krok spuštěn = infra/aws
- scriptContent - Spouštěný skript = ./delete-aurora-db-instance.sh db-instance-tst %branch_name% %testing_build_number%

Můžeme si všimnout, že tento krok využívá nastavení *executionMode* s hodnotou *BuildStep.ExecutionMode.ALWAYS*, jelikož může nastat situace, kdy některý z předchozích kroků nebyl úspěšně dokončen. Dojde tedy ke spuštění bashového skriptu s názvem *delete-aurora-db-instance.sh* v adresáři *infra/aws*. Tomuto skriptu jsou předány parametry *db-instance-tst*, *%branch_name%* a *%testing_build_number%*, kde *db-instance-tst* představuje předponu testovací databázové instance, *%branch_name%* představuje název větve ze sdíleného repozitáře verzovacího systému GIT, pro kterou bylo vytvořeno testovací prostředí a *%testing_build_number%* odpovídá číslu sestavení daného testovacího prostředí. Jedná se o ten samý skript, který je spouštěný při automatickém odstranění testovací databázové instance a testovacího databázového clusteru po dokončení API testů. Jednotlivé kroky v tomto skriptu jsou již popsány v sekci 3.3.8.

```
locals {
  branch-name-without-dashes = replace(var.branch-name, "-", "")
  environment-postfix = var.environment == "testing" ?
  ↪  "${var.environment}-${var.build-id}-${local.branch-name-without-dashes}"
  ↪  : var.environment
}
```

Obrázek 3.49: Kód, pomocí kterého je vytvářena přípona názvu AWS zdrojů v testovacím prostředí

3.5 Zdroje vytvořené pomocí terraformu

V této kapitole jsou popsány zdroje v AWS vytvořené pomocí terraform skriptů, viz sekce 3.3.4. Terrafrom skripty již existovaly pro vytváření zdrojů ve stagingovém a produkčním prostředí. Tyto skripty jsem tedy upravil, aby bylo možné je využít i v testovacím prostředí.

3.5.1 Vytvoření přípony s jednoznačným identifikátorem testovacího prostředí

Jelikož může existovat více testovacích prostředí najednou, tak je potřeba rozlišit které AWS zdroje patří k jakému testovacímu prostředí. K tomuto účelu slouží přípona v názvu každého AWS zdroje, která jednoznačně identifikuje dané testovací prostředí. Kód, pomocí kterého je tato přípona vytvořena, je zobrazen na obrázku 3.49.

Nejprve tedy dojde k odstranění všech znaků pomlčky z názvu větve ze sdíleného repozitáře verzovacího systému GIT, pro kterou je dané testovací prostředí vytvořeno. Přípona názvu je následně složena ze tří částí, které jsou spojené pomocí znaku pomlčky. První část označuje prostředí. V našem případě se tedy jedná o testovací prostředí, které je označováno jako „testing“. Druhá část odpovídá číslu sestavení daného testovacího prostředí. Třetí část představuje názvev větve ze sdíleného repozitáře verzovacího systému GIT, pro kterou je dané testovací prostředí vytvořeno a z kterého jsou odstraněny všechny znaky pomlčky.

3.5.2 Application Load Balancer

Pro přístup k backendové testovací aplikaci Uniqway je využit Load Balancer. Přístup k load balanceru je řízen pomocí security group, která umožňuje přístup z internetu pomocí HTTP protokolu přes port 80, viz sekce 3.5.9.1. Přijaté požadavky jsou následně přeposlány do target group, která je dále přeposílá přímo do backendové testovací aplikace Uniqway, viz sekce 3.5.3.

```
locals {
  is-production = var.environment == "production"
  is-original-uniqplay-service = var.service-name == "uniqplay"
  alb-name-first-part = local.is-production ? "uniqway" :
  ↪  (var.environment == "testing" ? "lb" : var.service-name)
  alb-name-middle-part = var.environment == "testing" ? "tst" :
  ↪  (local.is-original-uniqplay-service ? "load-balancer" :
  ↪  "lb")
  alb-name-last-part = var.environment-postfix
}

resource "aws_alb" "service" {
  name = substr("${local.alb-name-first-part}-
  ↪  ${local.alb-name-middle-part}-${local.alb-name-last-part}",
  ↪  0, 32)
}
```

Obrázek 3.50: Kód, pomocí kterého je vytvářen název testovacího load balanceru

```
name = var.environment == "testing" ?
↪  substr("green-tst-${var.environment-postfix}", 0, 32) :
↪  "${var.service-name}-green-${var.environment}"
```

Obrázek 3.51: Kód, pomocí kterého je vytvářen název testovací target group

Kód pro sestavení názvu testovacího load balanceru je zobrazen na obrázku 3.50. Název je sestaven ze tří částí, které jsou spojené pomocí znaku pomlčky. První část označuje, že se jedná o load balancer a obsahuje hodnotu „lb“. Druhá část označuje, že se jedná o testovací prostředí a obsahuje hodnotu „tst“. Třetí část obsahuje jednoznačný identifikátor daného testovacího prostředí, jehož vytvoření je popsané v sekci 3.5.1. Celý tento název je ještě navíc oříznut pouze na prvních 32 znaků kvůli omezení AWS.

3.5.3 Target group

Úkolem vytvořené target groupy je přijímat obdržené požadavky z load balanceru. Následně dojde k přeposlání obdržených požadavků do backendové testovací aplikace Uniqway na portu 9000.

Kód pro sestavení názvu testovací target group je zobrazen na obrázku 3.51. Skládá se ze dvou částí, které jsou spojené pomocí znaku pomlčky. První část je statický text dané testovací target group, v našem případě se jedná o

```
locals {
  family-name =
  ↪  "${var.service-name}-task-${var.environment-postfix}"
}
```

Obrázek 3.52: Kód, pomocí kterého je vytvářen název testovací ECS Task definice

```
locals {
  service-name =
  ↪  "${var.service-name}-service-${var.environment-postfix}"
}

resource "aws_ecs_service" "service" {
  name = local.service-name
}
```

Obrázek 3.53: Kód, pomocí kterého je vytvářen název testovací ECS instance aplikace

„green-tst“. Druhá část obsahuje jednoznačný identifikátor daného testovacího prostředí, jehož vytvoření je popsané v sekci 3.5.1. Celý tento název je ještě navíc oříznut pouze na prvních 32 znaků kvůli omezení AWS.

3.5.4 Definice ECS tasku

Backendová aplikace Uniqway je spouštěna jako ECS Task, proto je nejprve potřeba její definice, která říká, jak má být daná služba spuštěna, viz sekce 1.4.2. Kód pro sestavení názvu testovací ECS Task definice je zobrazen na obrázku 3.52. Skládá se ze tří částí, které jsou spojené pomocí znaku pomlčky. První část je název dané služby, v našem případě se jedná o „uniqway“. Druhá část obsahuje statický text „task“. Třetí část obsahuje jednoznačný identifikátor daného testovacího prostředí, jehož vytvoření je popsané v sekci 3.5.1.

3.5.5 Aplikace spuštěna jako služba v ECS

Během vytváření zdrojů v AWS pomocí terraformu dojde zároveň i k vytvoření ECS (viz sekce 1.4.2) testovací backednové aplikace Uniqway. Tato aplikace je spuštěna jako Docker kontejner, který přijímá požadavky na portu 9000. Přístup k testovací aplikaci je řízen pomocí security group, která umožňuje přístup pomocí portu 9000, viz sekce 3.5.9.2. Logy z běhu testovací aplikace je možné zobrazit přímo v AWS ECS sekci po rozkliknutí dané ECS služby s požadovanou testovací aplikací.

```
1 resource "aws_s3_bucket" "user-verification" {
2   bucket =
  ↪  substr("uniqway-user-verification-${local.environment-postfix}",
  ↪  0, 63)
3 }
```

Obrázek 3.54: Kód, pomocí kterého je vytvářen název User Verification Bucketu

Kód pro sestavení názvu ECS instance aplikace je zobrazen na obrázku 3.53. Skládá se ze tří částí, které jsou spojené pomocí znaku pomlčky. První část je název dané služby, v našem případě se jedná o „uniqway“. Druhá část obsahuje statický text „service“. Třetí část obsahuje jednoznačný identifikátor daného testovacího prostředí, jehož vytvoření je popsané v sekci 3.5.1.

3.5.6 User Verification Bucket

User Verification Bucket se nachází v AWS S3 bucketu (viz sekce 1.4.9) a slouží k uložení naskenovaných dokladů uživatelů, kteří využívají projekt Uniqway. Jedná se například o občanský nebo řidičský průkaz nebo studentskou kartu. Kód pro sestavení názvu User Verification Bucketu je zobrazen na obrázku 3.54. Skládá se ze dvou částí, které jsou spojené pomocí znaku pomlčky. První část obsahuje statický text „uniqway-user-verification“. Druhá část obsahuje jednoznačný identifikátor daného testovacího prostředí, jehož vytvoření je popsané v sekci 3.5.1. Celý tento název je ještě navíc oříznut pouze na prvních 63 znaků kvůli omezení AWS.

3.5.7 IAM Policy

Během vytváření nového testovacího prostředí jsou vytvořeny i IAM Policies, které poskytují práva pro manipulaci se zdroji v AWS. Celkově jsou vytvořeny tři IAM Policies, které jsou popsané v následujících sekcích.

```
resource "aws_iam_policy" "service_deployment" {
  name =
  ↪  substr("${var.service-name}-deployment-${var.environment-postfix}",
  ↪  0, 128)
}
```

Obrázek 3.55: Kód, pomocí kterého je vytvářen název Deployment Policy

```
resource "aws_iam_policy" "service_deployment_iam_pass_role" {
  name = substr(
  ↪  "${var.service-name}-deployment-iam-pass-${var.environment-postfix}",
  ↪  0, 128)
}
```

Obrázek 3.56: Kód, pomocí kterého je vytvářen název Service IAM Pass Role Policy

3.5.7.1 Deployment Policy

Tato IAM policy obsahuje práva potřebná k úspěšnému nasazení a spuštění aplikace pomocí Amazon ECS. Kód pro sestavení názvu Deployment Policy je zobrazen na obrázku 3.55. Skládá se ze tří částí, které jsou spojené pomocí znaku pomlčky. První část je název dané služby, v našem případě se jedná o „uniqway“. Druhá část obsahuje statický text „deployment“. Třetí část obsahuje jednoznačný identifikátor daného testovacího prostředí, jehož vytvoření je popsané v sekci 3.5.1. Celý tento název je ještě navíc oříznut pouze na prvních 128 znaků kvůli omezení AWS.

3.5.7.2 Service IAM Pass Role Policy

Tato IAM policy obsahuje práva k tomu, aby při vytváření ECS instance dostala tato instance přístup ke spouštěným kontejnerům. Kód pro sestavení názvu Service IAM Pass Role Policy je zobrazen na obrázku 3.56. Skládá se ze tří částí, které jsou spojené pomocí znaku pomlčky. První část je název dané služby, v našem případě se jedná o „uniqway“. Druhá část obsahuje statický text „deployment-iam-pass“. Třetí část obsahuje jednoznačný identifikátor daného testovacího prostředí, jehož vytvoření je popsané v sekci 3.5.1. Celý tento název je ještě navíc oříznut pouze na prvních 128 znaků kvůli omezení AWS.

```
resource "aws_iam_policy" "service_task_container" {
  name = substr("${var.service-name}-task-container-policy-
  ↪  ${var.environment-postfix}", 0, 128)
}
```

Obrázek 3.57: Kód, pomocí kterého je vytvářen název Task Container Policy

```
resource "aws_iam_role" "service_task_container" {
  name = substr("${var.service-name}-task-container-role-
  ↪  ${var.environment-postfix}", 0, 64)
}
```

Obrázek 3.58: Kód, pomocí kterého je vytvářen název Task Container Role

3.5.7.3 Task Container Policy

Tato IAM policy obsahuje potřebná práva pro správný běh samotného tasku v rámci celé ECS instance. Například umožňuje přístup k uloženým secretům (viz sekce 1.4.10) nebo User Verification Bucketu (viz sekce 3.5.6). Kód pro sestavení názvu Task Container Policy je zobrazen na obrázku 3.57. Skládá se ze tří částí, které jsou spojené pomocí znaku pomlčky. První část je název dané služby, v našem případě se jedná o „uniqway“. Druhá část obsahuje statický text „task-container-policy“. Třetí část obsahuje jednoznačný identifikátor daného testovacího prostředí, jehož vytvoření je popsané v sekci 3.5.1. Celý tento název je ještě navíc oříznut pouze na prvních 128 znaků kvůli omezení AWS.

3.5.8 IAM Role

Během vytváření nového testovacího prostředí jsou vytvořeny i IAM Roles, které vyžívají vytvořené IAM Policies (viz sekce 3.5.7), které poskytují těmto rolím práva pro manipulaci se zdroji v AWS. Celkově jsou vytvořeny čtyři IAM Roles, které jsou popsané v následujících sekcích.

3.5.8.1 Task Container Role

Tato IAM Role využívá Task Container Policy (viz sekce 3.5.7.3) a poskytuje potřebná práva pro správný běh samotného tasku v rámci celé ECS instance. Například umožňuje přístup k uloženým secretům (viz sekce 1.4.10) nebo User Verification Bucketu (viz sekce 3.5.6). Kód pro sestavení názvu Task Container Role je zobrazen na obrázku 3.58. Skládá se ze tří částí, které jsou spojené pomocí znaku pomlčky. První část je název dané služby, v našem případě se jedná o „uniqway“. Druhá část obsahuje statický text

```
resource "aws_iam_role" "service_deployment" {
  name = substr("${var.service-name}-deployment-role-
  ↪ ${var.environment-postfix}", 0, 64)
}
```

Obrázek 3.59: Kód, pomocí kterého je vytvářen název Deplyment Role

```
locals {
  service-autoscaler-name =
  ↪ substr("${var.service-name}-ecs-scale-${var.environment-postfix}",
  ↪ 0, 64)
}

resource "aws_iam_role" "service_autoscaler" {
  name = local.service-autoscaler-name
}
```

Obrázek 3.60: Kód, pomocí kterého je vytvářen název ECS Scale Role

„task-container-role“. Třetí část obsahuje jednoznačný identifikátor daného testovacího prostředí, jehož vytvoření je popsané v sekci 3.5.1. Celý tento název je ještě navíc oříznut pouze na prvních 64 znaků kvůli omezení AWS.

3.5.8.2 Deployment Role

Tato IAM Role využívá Deployment Policy (viz sekce 3.5.7.1) a poskytuje potřebná práva k úspěšnému nasazení a spuštění aplikace pomocí Amazon ECS. Kód pro sestavení názvu Deployment Role je zobrazen na obrázku 3.59. Skládá se ze tří částí, které jsou spojené pomocí znaku pomlčky. První část je název dané služby, v našem případě se jedná o „uniqway“. Druhá část obsahuje statický text „deployment-role“. Třetí část obsahuje jednoznačný identifikátor daného testovacího prostředí, jehož vytvoření je popsané v sekci 3.5.1. Celý tento název je ještě navíc oříznut pouze na prvních 64 znaků kvůli omezení AWS.

3.5.8.3 ECS Scale Role

Tato IAM Role poskytuje práva pro automatické škálovaní ECS instance. Kód pro sestavení názvu ECS Scale Role je zobrazen na obrázku 3.60. Skládá se ze tří částí, které jsou spojené pomocí znaku pomlčky. První část je název dané služby, v našem případě se jedná o „uniqway“. Druhá část obsahuje statický text „ecs-scale“. Třetí část obsahuje jednoznačný identifikátor daného

```
resource "aws_iam_role" "service" {
  name = substr("${var.service-name}-task-execution-role-
  ↪ ${var.environment-postfix}", 0, 64)
}
```

Obrázek 3.61: Kód, pomocí kterého je vytvářen název Task Execution Role

```
resource "aws_security_group" "loadbalancer" {
  name = var.service-name == "uniqplay" ?
  ↪ substr("${var.service-name}-load-balancer-${var.environment-postfix}",
  ↪ 0, 255) :
  ↪ substr("${var.service-name}-lb-${var.environment-postfix}",
  ↪ 0, 255)
}
```

Obrázek 3.62: Kód, pomocí kterého je vytvářen název Load Balancer Security Group

testovacího prostředí, jehož vytvoření je popsané v sekci 3.5.1. Celý tento název je ještě navíc oříznut pouze na prvních 64 znaků kvůli omezení AWS.

3.5.8.4 Task Execution Role

Tato IAM Role poskytuje práva ECS kontejneru, aby mohl využívat AWS API k tomu, aby spravoval běžící ECS tasky. Kód pro sestavení názvu Task Execution Role je zobrazen na obrázku 3.61. Skládá se ze tří částí, které jsou spojené pomocí znaku pomlčky. První část je název dané služby, v našem případě se jedná o „uniqway“. Druhá část obsahuje statický text „task-execution-role“. Třetí část obsahuje jednoznačný identifikátor daného testovacího prostředí, jehož vytvoření je popsané v sekci 3.5.1. Celý tento název je ještě navíc oříznut pouze na prvních 64 znaků kvůli omezení AWS.

3.5.9 Security groups

Během vytváření nového testovacího prostředí jsou vytvořeny i Security Groups (viz sekce 1.4.8), které umožňují přístup k daným zdrojům pomocí TCP protokolu. Celkově jsou vytvořeny dvě security groups, které jsou popsané v následujících sekcích.

3.5.9.1 Load Balancer Security Group

Tato security group umožňuje přístup k vytvořenému load balanceru (viz sekce 3.5.2) z veřejného internetu pomocí HTTP protokolu přes port 80. Kód

```
1 resource "aws_security_group" "ecs_tasks" {
2   name = var.service-name == "uniqplay" ?
  ↪  "${var.service-name}-tasks-security-group-${var.environment-postfix}"
  ↪  : "${var.service-name}-task-${var.environment-postfix}"
3 }
```

Obrázek 3.63: Kód, pomocí kterého je vytvářen název ECS Tasks Security Group

pro sestavení názvu Load Balancer Security Group je zobrazen na obrázku 3.62. Skládá se ze tří částí, které jsou spojené pomocí znaku pomlčky. První část je název dané služby, v našem případě se jedná o „uniqway“. Druhá část obsahuje statický „lb“. Třetí část obsahuje jednoznačný identifikátor daného testovacího prostředí, jehož vytvoření je popsané v sekci 3.5.1. Celý tento název je ještě navíc oříznut pouze na prvních 255 znaků kvůli omezení AWS.

3.5.9.2 ECS Tasks Security Group

Tato security group umožňuje přístup k testovací backend aplikaci Uniqway z vytvořené target group (viz sekce 3.5.3) pomocí protokolu TCP a portu 9000. Kód pro sestavení názvu ECS Tasks Security Group je zobrazen na obrázku 3.63. Skládá se ze tří částí, které jsou spojené pomocí znaku pomlčky. První část je název dané služby, v našem případě se jedná o „uniqway“. Druhá část obsahuje statický text „task“. Třetí část obsahuje jednoznačný identifikátor daného testovacího prostředí, jehož vytvoření je popsané v sekci 3.5.1. Celý tento název je ještě navíc oříznut pouze na prvních 255 znaků kvůli omezení AWS.

Závěr

Výsledek této diplomové práce splňuje všechny cíle, které byly v zadání. Zanalyzoval jsem způsob testování backendové aplikace Uniqway, navrhl řešení přidání nového testovacího prostředí a následně toto řešení zaintegroval do projektu.

Řešení zadaného problému umožňuje dynamické vytvoření libovolného počtu testovacích prostředí. V každém testovacím prostředí může být spuštěna libovolná verze Uniqway backendové aplikace. Každé prostředí automaticky otestuje spuštěnou aplikace a zobrazí výsledky spuštěných testů. Vytvořené testovací prostředí může být automaticky odstraněno po dokončení automatických testů a nebo může být dále ponecháno pro další testování. V případě, že nebylo vytvořené prostředí automaticky odstraněno, tak je umožněno manuální spuštění procesu odstranění tohoto prostředí.

Celý tento proces umožňuje správné otestování backendové aplikace projektu Uniqway a ulehčuje její další vývoj a testování.

Návrhy pro další rozvoj procesu dynamického vytvoření nového testovacího prostředí

Při vytváření nového testovacího prostředí je potřeba vždy vytvořit novou databázi, kterou bude toto prostředí využívat. Proces vytvoření nové testovací databáze je momentálně časově náročný, jelikož vždy dochází k vytvoření nového databázového clusteru a nové databázové instance v tomto clusteru. Tyto dva kroky mohou dohromady trvat přes deset minut.

Jedno z možných řešení tohoto problému je vytvoření pouze jednoho testovacího databázového clusteru a instance. Tato instane by následně byla sdílena všemi testovacími prostředími. Pro každé testovací prostředí by vždy byla vytvořena nová databáze v testovací databázové instanci. Touto úpravou by došlo k odstranění časově náročných kroků.

Literatura

[1] Kdo je Uniqway? *Uniqway [online]*, Březen 2023, [cit. 2023-03-03]. Dostupné z: `https://uniqway.cz/about`

[2] Kripner, M.: Unit testy v Javě a JUnit. *itnetwork.cz [online]*, Červenec 2015, [cit. 2023-03-08]. Dostupné z: `https://www.itnetwork.cz/java/testovani/java-unit-testy-v-junit`

[3] Gillis, A. S.: API testing. *TechTarget [online]*, Březen 2023, [cit. 2023-03-08]. Dostupné z: `https://www.techtarget.com/searchapparchitecture/definition/API-testing`

[4] What is a Staging Environment? *Umbraco [online]*, Květen 2019, [cit. 2023-03-08]. Dostupné z: `https://umbraco.com/knowledge-base/staging-environment/`

[5] Williams, A.: What Is Play Framework? *Built In [online]*, Prosinec 2022, [cit. 2023-03-05]. Dostupné z: `https://builtin.com/software-engineering-perspectives/play-framework`

[6] What is PostgreSQL? *PostgreSQL [online]*, Duben 2018, [cit. 2023-03-08]. Dostupné z: `https://www.postgresql.org/about/`

[7] Amazon Resource Names (ARNs). *Amazon Web Services [online]*, Duben 2023, [cit. 2023-04-09]. Dostupné z: `https://docs.aws.amazon.com/IAM/latest/UserGuide/reference-arns.html`

[8] What is Amazon Elastic Container Service? *Amazon Web Services [online]*, Duben 2023, [cit. 2023-04-09]. Dostupné z: `https://docs.aws.amazon.com/AmazonECS/latest/developerguide/Welcome.html`

[9] What is CodeDeploy? *Amazon Web Services [online]*, Duben 2023, [cit. 2023-04-09]. Dostupné z: `https://docs.aws.amazon.com/codedeploy/latest/userguide/welcome.html`

[10] What is Amazon Relational Database Service (Amazon RDS)? *Amazon Web Services [online]*, Duben 2023, [cit. 2023-04-09]. Dostupné z: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html

[11] What Is Load Balancing? *Amazon Web Services [online]*, Duben 2023, [cit. 2023-04-09]. Dostupné z: https://aws.amazon.com/what-is/load-balancing/

[12] Target groups for your Application Load Balancers. *Amazon Web Services [online]*, Duben 2023, [cit. 2023-04-09]. Dostupné z: https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-target-groups.html

[13] What is Amazon VPC? *Amazon Web Services [online]*, Duben 2023, [cit. 2023-04-09]. Dostupné z: https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html

[14] Security groups. *Amazon Web Services [online]*, Duben 2023, [cit. 2023-04-10]. Dostupné z: https://docs.aws.amazon.com/managedservices/latest/userguide/about-security-groups.html

[15] What is Amazon S3? *Amazon Web Services [online]*, Duben 2023, [cit. 2023-04-10]. Dostupné z: https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html

[16] What is AWS Secrets Manager? *Amazon Web Services [online]*, Duben 2023, [cit. 2023-04-10]. Dostupné z: https://docs.aws.amazon.com/secretsmanager/latest/userguide/intro.html

[17] What is Terraform? *HashiCorp [online]*, Září 2022, [cit. 2023-04-10]. Dostupné z: https://developer.hashicorp.com/terraform/intro

[18] Getting Started with TeamCity. *JetBrains [online]*, Listopad 2021, [cit. 2023-04-10]. Dostupné z: https://www.jetbrains.com/help/teamcity/getting-started-with-teamcity.html

[19] Kotlin DSL. *JetBrains [online]*, Únor 2023, [cit. 2023-04-10]. Dostupné z: https://www.jetbrains.com/help/teamcity/kotlin-dsl.html

[20] Aurora Serverless v2. *Amazon Web Services [online]*, Březen 2023, [cit. 2023-03-10]. Dostupné z: https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/Concepts.Aurora_Fea_Regions_DB-eng.Feature.ServerlessV2.html

[21] create-db-cluster. *Amazon Web Services [online]*, Prosinec 2022, [cit. 2023-03-11]. Dostupné z: https://awscli.amazonaws.com/v2/documentation/api/2.9.6/reference/rds/create-db-cluster.html

[22] create-db-instance. *Amazon Web Services [online]*, Prosinec 2022, [cit. 2023-03-12]. Dostupné z: `https://awscli.amazonaws.com/v2/documentation/api/2.9.6/reference/rds/create-db-instance.html`

Příloha A

Seznam použitých zkratek

API Application Programming Interface

ARN Amazon Resource Name

AWS Amazon Web Services

ECS Elastic Container Service

ELB Elastic Load Balancing

RDS Relational Database Service

SQL Structured Query Language

URL Uniform Resource Locator

VPC Virtual Private Cloud

Příloha B

Obsah přiloženého CD

- `README.md` ..... Stručný popis obsahu CD
- `uniqway-server` ..... Zdrojové kódy implementace
- `DP_Levy_Jan_2023.pdf` ..... Text práce ve formátu PDF