F3

Fakulta elektrotechnická
Katedra počítačů

Diplomová práce

Cloud-native aplikace pro přípravu na přijímací zkoušky z matematiky

Bc. Petr Jeřábek
Otevřená informatika, Softwarové inženýrství

2024
Vedoucí práce: Ing. Martin Komárek

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Jeřábek** Jméno: **Petr** Osobní číslo: **483630**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Specializace: **Softwarové inženýrství**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Cloud-native aplikace pro přípravu na přijímací zkoušky z matematiky

Název diplomové práce anglicky:

Cloud-native Application for Unified Entrance Examination Preparation

Pokyny pro vypracování:

Vytvořte webovou aplikaci usnadňující přípravu na jednotnou přijímací zkoušku z matematiky na střední školy a gymnázia. Aplikace bude umožňovat procvičování různých typových úloh objevujících se pravidelně v testech. Aplikace bude uživateli zaznamenávat strávený čas při řešení příkladů a úspěšnost.
Analyzujte a parametrizujte úlohy vyskytující se v testech jednotné přijímací zkoušky.
Nastudujte technologie z oblasti cloud native a microservices, které využijete při návrhu a implementaci, nasazení a provozu aplikace.
Při vývoji postupujte iterativně. Vše průběžně testujte, nasazujte na Value Stream Delivery Platformu CodeNOW a dokumentujte.
Aplikaci otestujte jednotkovými testy, automatickými testy UI a zátěžovými testy. Nasaďte aplikaci do produkce a sledujte uživateské chování pomocí nástojů typu Smartlook/Hotjar.

Seznam doporučené literatury:

Richardson, Chris . Microservices Patterns with examples in Java .
Debezium [online]. Dostupné na https://debezium.io/.
KrakenD [online]. Dostupné na https://www.krakend.io/.
Zitadel [online]. Dostupné na https://zitadel.com/.
Jaeger tracing [online]. Dostupné na https://www.jaegertracing.io/.
CodeNOW [online]. Dostupné na https://www.codenow.com/.
CZVV, Testová zadání v PDF [online]. [vid. 2023-05-24]. Dostupné na https://
prijimacky.cermat.cz/menu/testova-zadani-k-procvicovani/testovazadani-v-pdf.
Webová aplikace CZVV pro procvičování úloh [online]. [vid. 2023-05-24]. Dostupné na https://procvicprijimacky.cermat.cz/.
Webová aplikace Umíme To [online]. [vid. 2023-05-24]. Dostupné na https://www. umimeto.org/.
Specifikace požadavků pro jednotnou přijímací zkoušku v přijímacím řízení na střední školy v oborech vzdělání s maturitní zkouškou [online]. [vid. 2023-05-24]. Dostupné na https://prijimacky.cermat.cz/files/files/dokumenty/spec
ifikace-pozadavku/Specifikace_2022-2023/MASPECIFIKACEPOZADAVKU2022. pdf.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Martin Komárek kabinet výuky informatiky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **31.08.2023** Termín odevzdání diplomové práce: _____

Platnost zadání diplomové práce: **16.02.2025**

Ing. Martin Komárek podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

Datum převzetí zadání	Podpis studenta

Poděkování / Prohlášení

Tímto bych rád poděkoval vedoucímu mé diplomové práce Ing. Martinu Komárkovi za podnětné rady, metodickou a odbornou pomoc při zpracování mé práce. Poděkování také patří mé rodině a všem blízkým, kteří mne podporovali nejen během této závěrečné práce, ale i během celého studia.

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.

V Praze dne 9.1.2024

.....

Abstrakt / Abstract

Tato práce se zabývá vytvořením interaktivní aplikace pro přípravu na jednotné přijímací zkoušky z matematiky na střední školy a víceletá gymnázia. Aplikace by měla nabízet dynamicky generované úlohy obdobné těm v testech od Centra pro zjišťování výsledků vzdělávání (Cermat) a nabízet přihlášeným uživatelům možnost sledovat úspěšnost a čas strávený při řešení úloh. Cílem je navrhnout, implementovat, otestovat a nasadit aplikaci na Value Stream Delivery Platformu CodeNOW.

Klíčová slova: jednotné přijímací zkoušky, střední škola, gymnázium, příprava, matematika, cloud, CodeNOW

This thesis deals with the creation of an interactive application for preparation for the Unified Entrance Examination in Mathematics for secondary schools and multi-year grammar schools. The application should offer dynamically generated tasks similar to those in the tests from the Centre for the Determination of Educational Results (Cermat) and offer logged-in users the possibility to monitor the success rate and time spent in solving the tasks. The goal is to design, implement, test and deploy the application on the CodeNOW Value Stream Delivery Platform.

Keywords: unified entrance exam, high school, grammar school, preparation, mathematics, cloud, CodeNOW

Title translation: Cloud-native Application for Unified Entrance Examination Preparation

Obsah /

Kapitola 1
Úvod

1.1 Jednotná přijímací zkouška

Jednotná přijímací zkouška na střední školy byla v České republice zavedena v roce 2017. Tato forma přijímacích zkoušek byla zavedena pro všechny obory střeních škol, jež jsou zakončeny maturitní zkouškou. Jednotné testy ve dvou kategoriích, český jazyk a matematika, zajišťuje příspěvková organizace Cermat neboli Centrum pro zjišťování výsledků vzdělávání, řízená Ministerstvem školství mládeže a tělovýchovy. V roce 2023 se na čtyřleté obory hlásilo celkem 89 729 žáků, na šestiletá gymnázia 7 347 žáků a na osmiletá gymnázia 18 859 žáků. Tedy celkem téměř 116 000 uchazečů vykonávalo v roce 2023 jednotnou přijímací zkoušku [1].

1.2 Znalost, dovednost nebo kompetence

Znalost dovednost a kompetence jsou blízké a občas zaměňované pojmy. Jaký je mezi nimi rozdíl a jak se to odráží v přijímací zkoušce? Znalostmi se rozumí soubor faktických nebo procedurálních informací, které lze použít, například znalost cizích jazyků nebo programovacích jazyků. Znalosti jsou informace nebo expertiza, obvykle získaná prostřednictvím vzdělání nebo technického výcviku.

Dovednosti jsou potřebné k přesnému plnění úkolů, například psychomotorické činnosti, jako je rychlost psaní na klávesnici nebo schopnost řídit auto. Dovednosti lze také popsat jako odborné znalosti nebo talent potřebný k výkonu určité práce nebo úkolu. Dovednosti lze dále dělit do dvou různých kategorií, a to na měkké a tvrdé.

Schopnosti jsou stabilnější charakteristiky, které mohou zahrnovat kognitivní, smyslové a fyzické schopnosti, jako je například empatie. Schopnosti mají méně společného s tréninkem a rozvojem, protože jsou pro jedince většinou přirozené. [2]

Jednotná přijímací zkouška má ověřovat jak znalosti, jako například dobře známé početní algoritmy, tak dovednosti, ať už porozumění textu, či matematizace reálných problémů.

1.3 Motivace

Jednotná přijímací zkouška by měla ověřovat znalosti a dovednosti, ovšem ne vždy jsou ověřované znalosti a dovednosti stejné, jako ty, co se žáci učí na základních školách. Některé úlohy vyžadují dovednosti, které si žáci ve standardní výuce neosvojí, a tak pokud chtějí u testů uspět, musejí se speciálně připravovat. Žáci si musí osvojit učivo vyšších ročníků nebo speciální postupy či algoritmy, kterými lze dané typové úlohy řešit. V důsledku toho si mnoho žáků, respektive jejich rodiče, platí přípravné kurzy, kde se probírají postupy řešení úloh specifických pro jednotnou přijímací zkoušku. Tyto kurzy jsou zajišťovány třetími stranami a jejich cena často přesahuje i 10 000 Kč. Děti z rodin s nižšími příjmy jsou tak znevýhodněny, protože si takto nákladné kurzy jednoduše

nemohou dovolit a snižují se tak jejich šance dostat se na vybranou, někdy i alespoň nějakou, střední školu.

1.4 Cíl práce

Cílem této práce je vytvořit aplikaci, která by pomáhala připravovat uchazeče o studium na středních školách na jednotnou přijímací zkoušku z matematiky. Aplikace by měla nabízet interaktivní procvičování typových úloh objevujících se v testech. Úlohy by měly být co nejvíce parametrizované a dynamicky generované, tak aby jich bylo co největší množství. Zároveň by aplikace měla sledovat uživatelovu úspěšnost a čas při řešení úloh. Mimo rozsah této práce by pak v budoucím vývoji mohla aplikace nabízet uživateli úlohy, při jejichž řešení nebyl úspěšný, podobně jako to dělá známá aplikace na výuku cizích jazyků Duolingo [1], která je v tomto ohledu velkou inspirací.

[1] `https://www.duolingo.com`

Kapitola 2
Spolupráce s Centrem pro zjišťování výsledků vzdělávání (Cermat)

2.1 Souhlas s použitím materiálů

Jelikož primárním cílem vyvíjené aplikace je v současnosti příprava na jednotnou přijímací zkoušku, kterou zajišťuje právě Cermat, je potřeba pracovat přímo s testy připravovanými pro zkoušky. Testová zadání z každého roku, počínaje rokem 2015, jsou dostupná veřejně na stránkách Centra, ale jsou chráněna autorským právem (příloha A). Kontaktoval jsem proto ředitele Centra, Ing. Miroslava Krejčího, zda bych pro potřeby své diplomové práce mohl materiály využít. Tato žádost byla více než úspěšná, nejen, že jsem dostal písemný souhlas k použití materiálů, ale dokonce jsme byli s vedoucím mé práce pozvání na schůzku s ředitelem Centra, který o toto téma projevil zájem.

2.2 Schůzka s ředitelem Cermatu

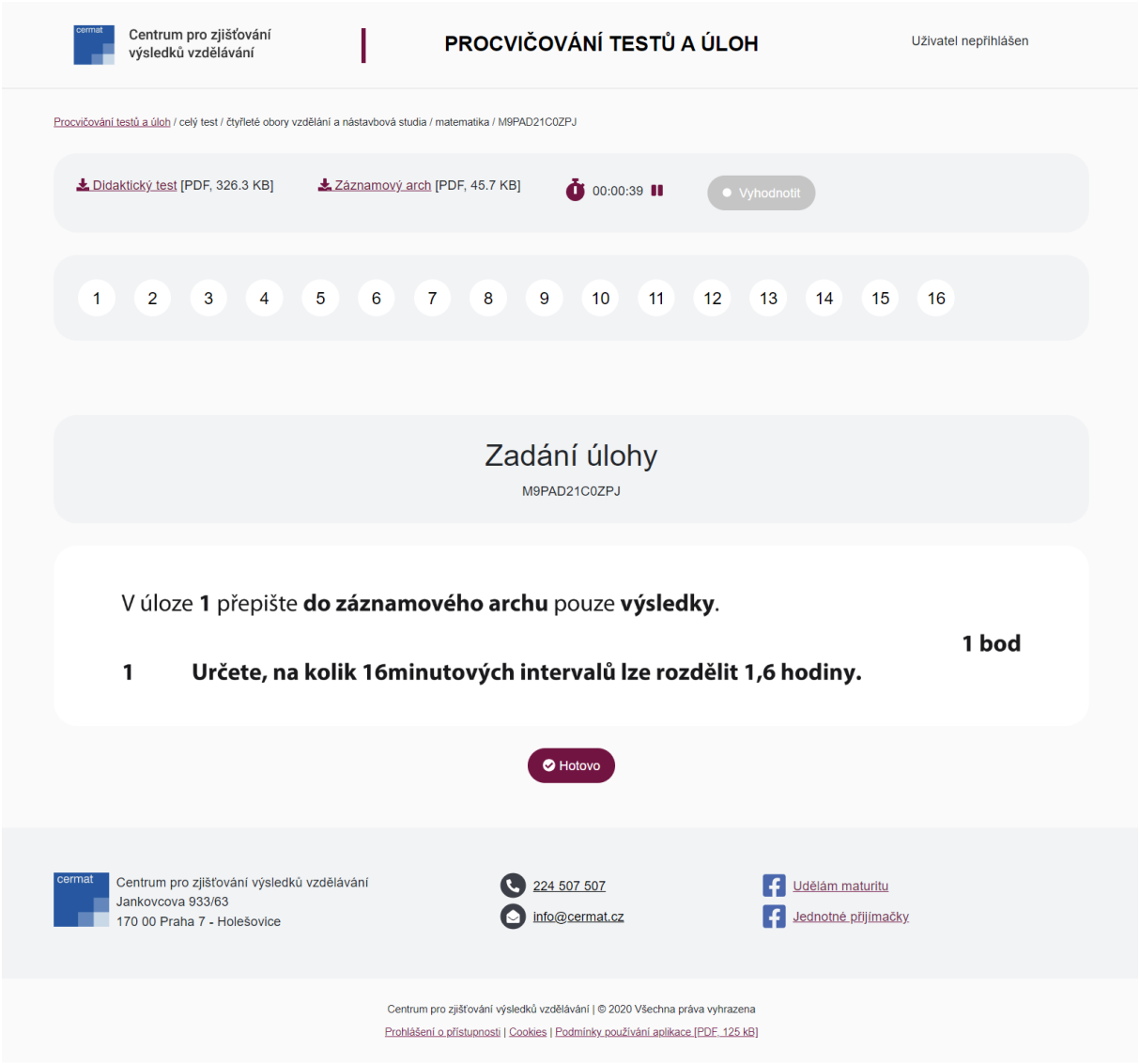
Naše schůzka s ředitelem proběhla 12.5.2023 a převážně šlo o vysvětlení cíle projektu, vizi nasazení a udržování projektu a diskuzi nad možnou další spoluprací ČVUT s Cermatem. Pan Krejčí projevil zájem o další spolupráci, takže je pravděpodobné, že se s dalším vývojem projektu rozroste i tato kapitola.

Kapitola 3
Analýza stávajících řešení

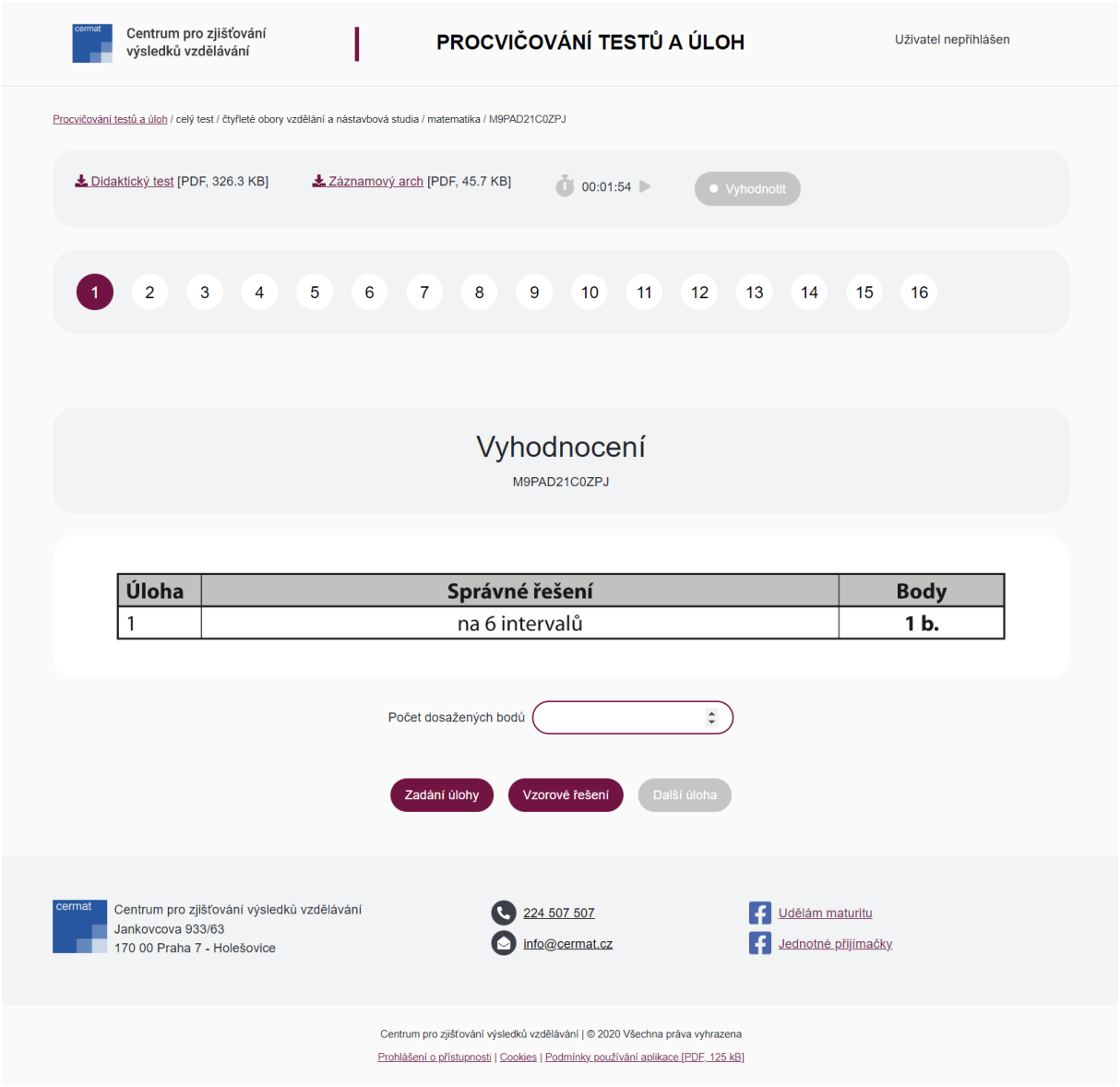
Nejčastějším způsobem přípravy jsou placené kurzy zajišťované různými společnostmi, které probíhají typicky na týdenní bázi a probírají úlohy, vyskytující se u zkoušek, a jejich řešení. Největší výhoda placených kurzů je živý lektor, který by měl být schopný vysvětlit postupy řešení daných úloh.

3.1 Webová aplikace CZVV

Mezi zdarma dostupné přípravné materiály patří testová zadání z uplynulých let přímo na stránkách CZVV [3] (Centra pro zajišťování výsledků vzdělávání). Tyto materiály lze stáhnout v podobě PDF nebo prohlížet ve webové aplikaci, která umožňuje procházet celé testy nebo vybraný typ úloh jednu po druhé a pak ukazuje správné výsledky. Z těchto materiálů také vychází moje analýza typových příkladů a jejich parametrizace a dynamické generování. Aplikace na webu CZVV [4] je dle mého názoru značně nedostačující, v podstatě pouze zobrazuje statické testy (obrázek 3.1). Oproti pouhým zadáním nabízí navíc vzorové řešení, které bývá ale poměrně stručné (obrázek 3.2).



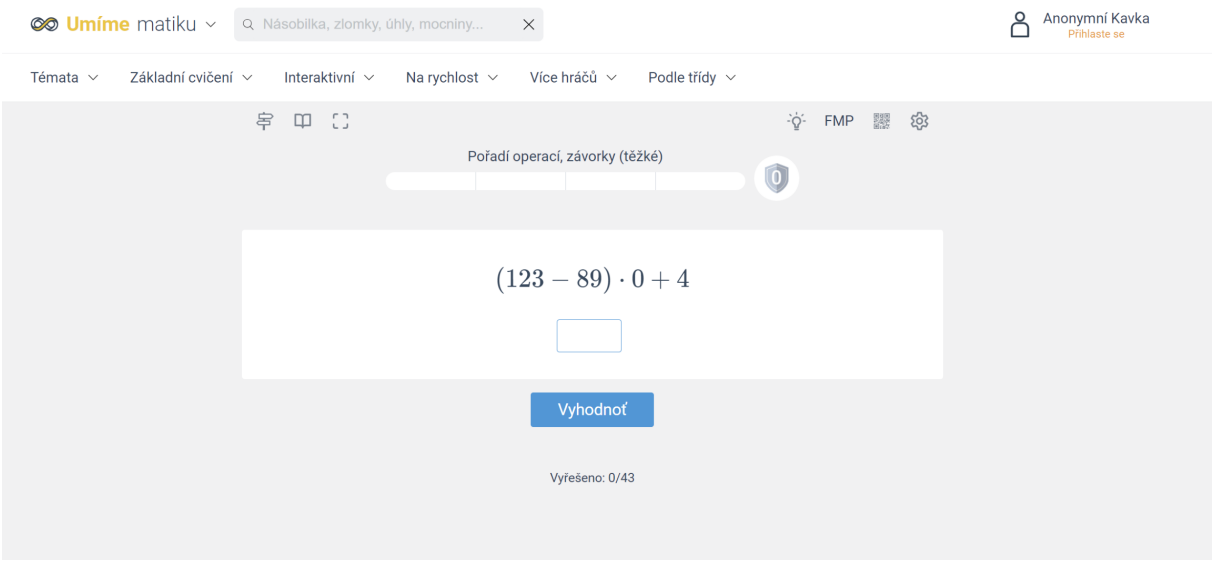
Obrázek 3.1. Webová aplikace CZVV pro procvičování testů - Úloha



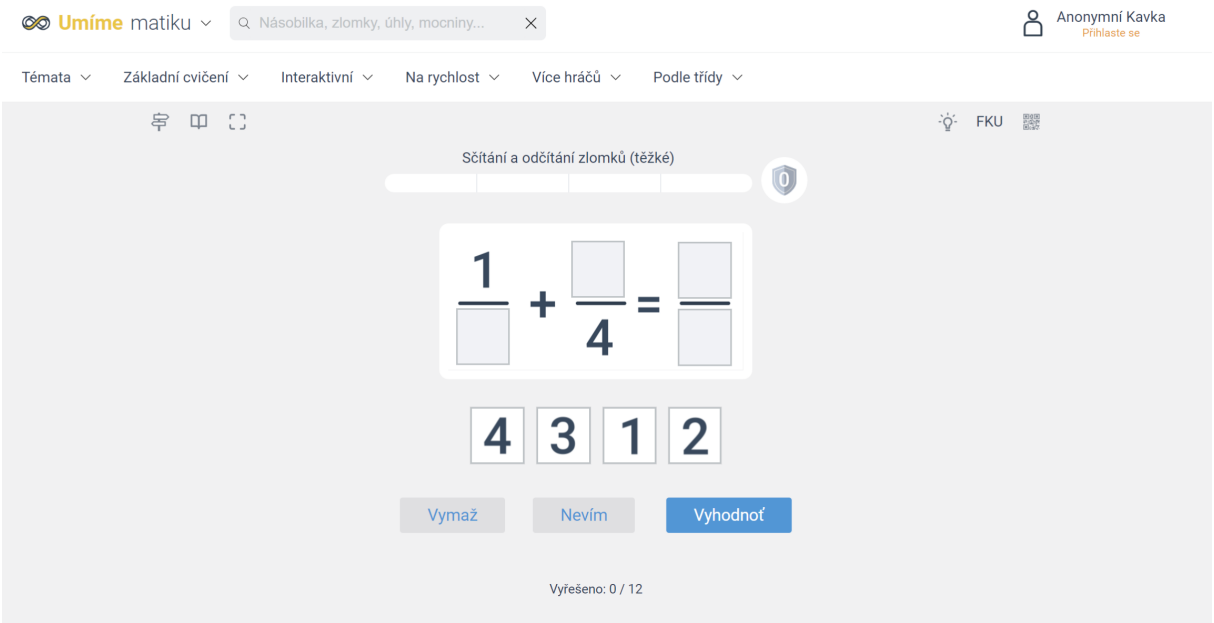
Obrázek 3.2. Webová aplikace CZVV pro procvičování testů - Řešení

3.2 Umíme to

Umíme to [5] je webová aplikace zabývající se procvičováním učiva a znalostí z mnoha školních předmětů od 1. třídy až po maturitní ročníky. Aplikaci lze využívat zdarma s omezeným počtem odpovědí za den, nebo si platit předplatné, které se liší podle délky, počtu osob a předmětů, a využívat neomezeně. (V případě jedné osoby s předměty matematika a český jazyk na 1 rok vychází předplatné na zhruba 850 Kč za celý rok) Aplikace působí uživatelsky velmi přívětivě a nabízí poměrně širokou škálu příkladů a úloh s interaktivními odpovědmi, ale pro potřeby přípravy na jednotné přijímací zkoušky nenabízí vhodné sady úloh (obrázek 3.3 a 3.4).



Obrázek 3.3. Webová aplikace Umíme to

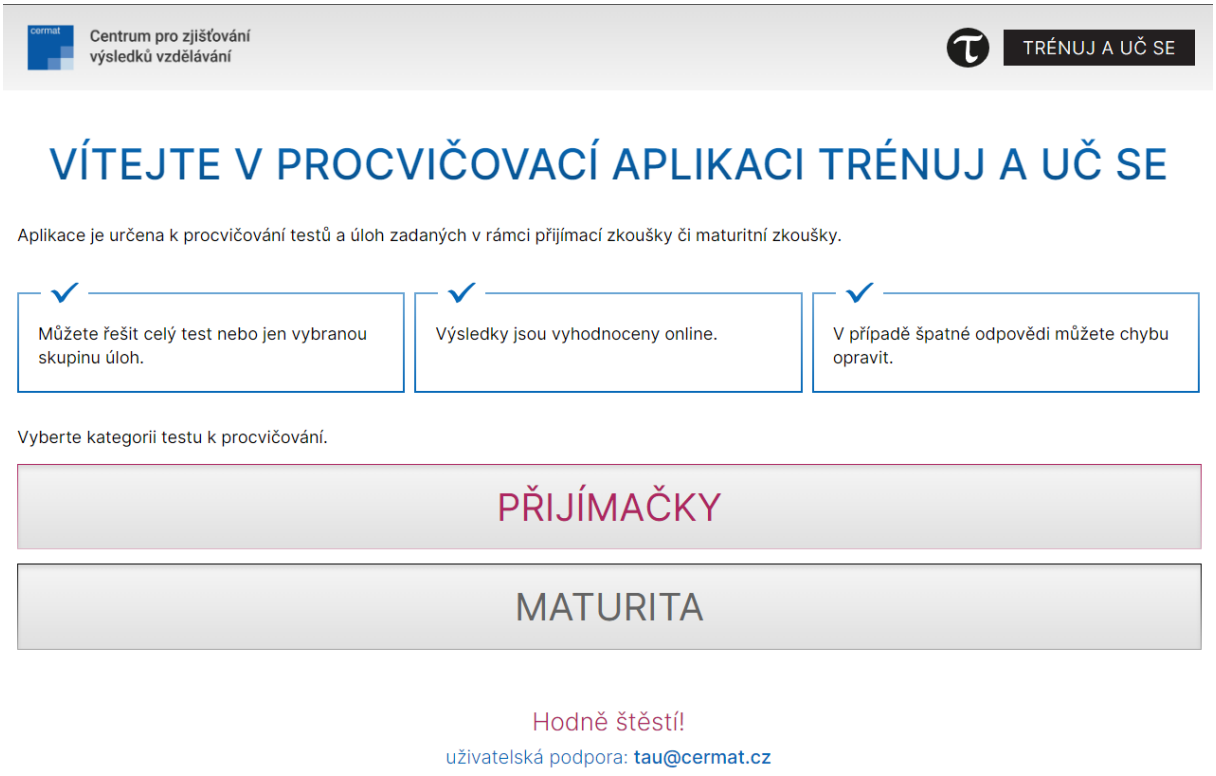


Obrázek 3.4. Webová aplikace Umíme to

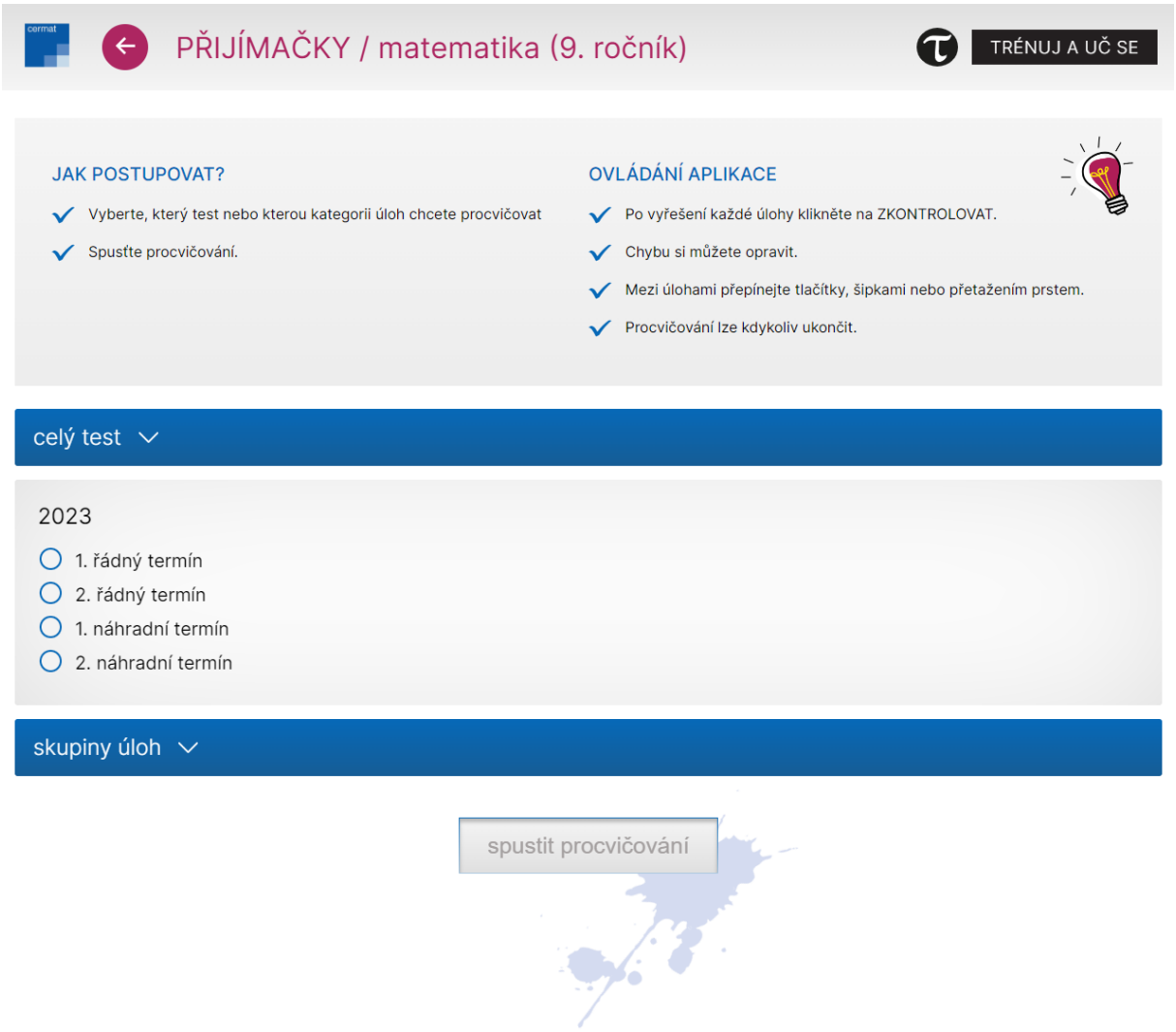
3.3 Trénuj a uč se (TAU)

Trénuj a uč se, zkráceně Tau, je nově spuštěný web Cermatu na podzim 2023, který umožňuje procházet testy z minulých let více interaktivním způsobem (obrázek 3.5 a 3.6). Oproti starší aplikace obsahuje kontrolu výsledků, které uživatel zadává přímo při řešení úlohy, měří čas a počítá procentuální úspěšnost (obrázek 3.7). Aplikace bohužel nepřináší žádné nové úlohy, pouze nový pohled na již vytvořená testová zadání z minulých let. Co se týče rýsovacích úloh, také nepřináší žádnou inovaci, uživatel si pouze zkontroluje svůj výsledek s referenčním obrázkem a dle popisu hodnocení si zvolí, kolik by měl získat bodů za úlohu. V současné chvíli nabízí aplikace pouze testy z roku 2023, případně skupiny úloh rozdělených dle znalostních okruhů, také pouze z tohoto

roku. Pravděpodobně bude další obsah postupně přidán, na to poukazuje například i připravené tlačítko pro maturitní úlohy na úvodní obrazovce, které zatím není funkční. Aplikace nenabízí přihlašování uživatelů a možnost sbírání dlouhodobějších statistik. [6]



Obrázek 3.5. Webová aplikace Trénuj a uč se - Úvodní stránka



Obrázek 3.6. Webová aplikace Trénuj a uč se - Výběr testu

Obrázek 3.7. Webová aplikace Trénuj a uč se - Testová úloha

Kapitola 4
Specifikace požadavků na vědomosti a dovednosti z matematiky

Úlohy z matematiky lze rozdělit do několika okruhů, kdy v každém testu jsou zastoupeny příklady z každého okruhu. Objevuje se několik typů úloh z hlediska odpovědí:

- Otevřené úlohy s postupem
- Otevřené úlohy - pouze výsledek
- Uzavřené úlohy - výběr z možností
- Konstrukční úlohy

Dále můžeme pro každý typ oboru rozdělit úlohy do několika kategorií[7]:

4.1 Čtyřleté obory

4.1.1 Číslo a proměnná

- úlohy na umocňování a odmocňování a geometrický význam druhé mocniny v praxi.
- aplikační úlohy na procenta, úlohy z finanční matematiky s použitím základních pojmů jako jistina, úroková míra, úrok, úrokovací období, daň nebo inflace.
- úlohy s proměnnými, výrazy s proměnnou a jejich rovnost, zápis textu jako rovnice s proměnnými a výpočet hodnoty výrazu pro dané hodnoty proměnných, početní operace s mnohočleny, rozklad mnohočlenu na součin, vytýkání, umocnění a rozložení dvojčlenu na součin dle vzorců
- řešení lineární rovnice pomocí ekvivalentních úprav se zkouškou správnosti řešení, rozhodnutí zda má rovnice jedno, nekonečno nebo žádné řešení, sestavení rovnice ze slovní úlohy
- řešení soustavy dvou rovnic o dvou neznámých dosazovací nebo sčítací metodou, praktické úlohy řešené známým algoritmem nebo úvahou
- matematizace reálné situace užitím rovnic, vyřešení a posouzení reálnosti výsledku

4.1.2 Závislosti, vztahy a práce s daty

- použití a porozumění základním statistickým pojmům, určení četnosti, aritmetického průměru, použití výsledků jednoduchého statistického šetření, použití vhodné tabulky nebo diagramu k jejich znázornění
- posouzení typu závislosti dvou veličin (přímá a nepřímá úměrnost), vyjádření závislosti tabulkou, rovnicí nebo grafem, odhalení funkčního vztahu v textu úlohy

4.1.3 Geometrie v rovině a prostoru

V této sekci se mimo jiné objevují konstrukční úlohy, které se špatně řeší na počítači, telefonu, či jiném elektronickém zařízení. Z toho důvodu tento typ úloh nebudu zahrnovat v řešení.

- rozbor situace pomocí náčrtku, použití správné symboliky
- použití a porozumění pojmům odvěsna a přepona v pravoúhlém trojúhelníku, využití Pythagorovy věty v trojúhelníku i v jiných tělesech a praktických úlohách
- účelné použití přibližné hodnoty π, výpočet obvodu a obsahu kruhu (kružnice) pomocí vzorců
- rozlišení shodných a podobných trojúhelníků, využití vět sss, uu, sus
- rozpoznání jehlanu ve volném rovnoběžném promítání, jeho sítě, využití vlastností jehlanu při řešení úloh
- rozpoznání rotačního válce ve volném rovnoběžném promítání, náčrt sítě, odhad a výpočet objemu a povrchu
- řešení aplikačních slovních úloh s využitím znalostí o válci a kouli
- využití měřítka mapy ve slovních úlohách k určení skutečných rozměrů a naopak

4.1.4 Nestandardní aplikační úlohy a problémy

- řešení úloh úvahou, zápis a zdůvodnění řešení
- použití známých algoritmů při řešení praktických problémů
- řešení jednoduchých strategických a kombinatorických úloh bez použití kombinatorických vzorců
- řešení netradičních geometrických úloh pomocí prostorové představivosti a náčrtků
- použití komplexních poznatků a dovedností z různých oblastí

4.2 Osmileté obory

4.2.1 Číslo a početní operace

- práce s přirozenými čísly od jedné do milionu a s nulou, porozumění pojmům jednotky, desítky, stovky, tisíce, desetitisíce, statisíce, miliony, cifra, jednociferné, dvojciferné až sedmiciferné číslo, rozvinutý zápis čísla
- použití číselných os k porovnání čísel, určení nerovnosti a použití znaků, porozumění pojmům o kolik, kolikrát, kolikrát více, kolikrát méně
- zaokrouhlování, početní operace a jejich vlastnosti, užití závorek při výpočtu
- pojmy související se základními početními operacemi, rozlišení sudých a lichých čísel
- písemné algoritmy početních operací, násobení až čtyřciferným činitelem, dělení jedno nebo dvojciferným dělitelem
- určení části celku, pojmy polovina, třetina atd., čtení zápisu zlomku, porovnání, sčítání a odčítání zlomků se stejným jmenovatelem (kladným), grafické zobrazení celku a části
- čtení zápisu desetinného čísla, znázornění na číselné ose, porozumění zápisu záporného celého čísla a znázornění na číselné ose
- řešení a tvoření slovních úloh se základními početními operacemi, matematizace reálné situace, odhad a reálnost výsledku, formulace odpovědi

4.2.2 Závislosti, vztahy a práce s daty

- struktura času, použití jednotek času, hmotnosti a dalších jednotek z geometrie a jejich převody
- vybírání dat z textu, tabulky nebo diagramu, třídění, doplnění chybějících údajů, organizace dat do tabulky, řešení slovních úloh pomocí schémat, tabulek a grafů

4.2.3 Geometrie v rovině a prostoru

- rozeznání a pojmenování základních rovinných útvarů, pojmy vrchol a strana
- rozeznání a modelování osově souměrných útvarů ve čtvercové síti a praktických situacích
- určení obsahu útvaru pomocí čtvercové sítě, použití základních jednotek obsahu, porovnání útvarů stejného typu dle velikosti
- rozeznání a pojmenování základních prostorových útvarů, porovnání stejného typu dle velikosti

4.2.4 Nestandardní aplikační úlohy a problémy

- využití úvahy při řešení slovních úloh
- matematizace reálných situací
- řešení pomocí grafické interpretace, formulace odpovědi

4.3 Shrnutí

V rámci této práce bude hlavním cílem vytvořit dynamicky generované úlohy s otevřenou odpovědí. Vyhodnocování postupu nebo konstrukčních úloh je velmi komplexní problém, který je nad rámec této práce. Uzavřené úlohy s výběrem z možností vedou k častému tipování výsledků a generování špatných odpovědí je citlivá záležitost, protože pokud je odpověď na první pohled špatně řešitel úlohu pak ani nemusí počítat, proto tedy pro potřeby procvičování je lepší řešit hlavně otevřené úlohy. Přijímací zkoušky se dají dělat i na šestileté obory, ovšem uchazečů o tyto obory je nejméně, tedy primárním cílem je vytvoření úloh pro čtyřleté a osmileté obory.

Kapitola 5
Analýza a parametrizace testových úloh

5.1 Čtyřleté obory

5.1.1 Číslo a proměnná

Zlomky

Početní příklady se zlomky se řídí několika pravidly. Jmenovatel i čitatel jsou celá čísla (kromě 0), příklad obsahuje zhruba 4 až 5 operandů s maximálně dvouciferným čitatelem a jmenovatelem a výsledek obsahuje také maximálně dvouciferný čitatel a jmenovatel. V příkladu se mezi zlomky mohou objevovat závorky a operace sčítání, odčítání a násobení.

Dále se v testech objevují slovní úlohy pracující se zlomky. Jednou z úloh je úloha na postup závodníka na trase závodu, kdy je zadáno za jakou dobu závodník překonal celou trasu, dále jak dlouhé části překonal v jednotlivých časových úsecích a jak jsou některé z částí dlouhé. Cílem je pak najít délku některého z úseků trasy. Příklad lze generovat třemi způsoby. První zadává celkový čas na zdolání trasy, jak velkou část trasy zdolal závodník během první hodiny a jak velkou část zdolal během poslední hodiny a jak tato část byla dlouhá. Otázka je pak, kolik kilometrů zdolal mezi prvním a posledním úsekem. Druhou možností je pak stejný způsob zadání, ale otázka je na počet zdolaných kilometrů v rámci první hodiny. Třetí možností je otočení zadání, kdy opět zadáváme dobu na zdolání celé trasy, jako druhou část zadáváme, jak velký úsek zdolal závodník během poslední hodiny a následně jak velkou část zdolal během hodiny první a jak tato část byla dlouhá. Ptát se můžeme opět na počet kilometrů zdolaných v poslední hodině a nebo mezi první a poslední hodinou.

Podmínky na čísla jsou následující:

$a, b, s \in \mathbb{N}$

$a, b \in \langle 2; 10)$

$s \in \langle 3; 17 - b)$

$bs \ mod \ a = 0 \wedge a \neq b$

Další úlohou je úloha na rozdíl velikosti částí jednoho celku. Úlohu lze měnit jak na úrovni čísel, tak na úrovni jednotek. Můžeme počítat s různými jednotkami objemu, délky nebo hmotnosti. Zadání je pak následující:

Vypočtěte, o kolik *jednotka* se liší $\frac{1}{b}$ z a *jednotka* a $\frac{1}{d}$ z c *jednotka*. Zlomky jsou zadané slovně, například `o kolik se liší polovina z 10 litrů...`.

Podmínky pro čísla:

$b, d \in \langle 2; 10)$

$a = p \cdot b, \ p \in \langle 2; 10)$

$c = q \cdot d, \ q \in \langle 2; 10)$

$a > b \wedge c > d \wedge b \neq d \wedge a \neq c$

$a \ mod \ b = 0 \wedge c \ mod \ d = 0$

$a, b, c, d, p, q \in \mathbb{N}$

Řešení: $|\frac{a}{b} - \frac{c}{d}|$

Slovní úlohy vedoucí na rovnice

Jednou ze slovních úloh vedoucí na rovnici je úloha počtu lahví v přepravce.
Zadání je následující:

V každé *přepravce* je n lahví (*s nějakou tekutinou*). Každá lahev obsahuje v litru *tekutiny*. Ve všech *přepravkách* je celkem t litrů *tekutiny*.
Určete počet *přepravek* s *tekutinou*.
Podmínky:

$n \in \langle 5; 12)$

$v \in \{0.25, 0.3, 0.5, 0.75, 1\}$

$f = min\{\langle 2; 100\rangle\}, \ (v \cdot n \cdot f) \in \mathbb{N}$

$p \in \langle 2f; 1000)$

$t = p \cdot n \cdot v$

$f, n, p, t \in \mathbb{N}$

Řešení: $\frac{t}{n \cdot v} \Rightarrow p$

Do této skupiny úloh by se dala zařadit i stejná úloha jako v příkladech pro osmileté obory, úloha na dobu trvání cesty.

Další úlohou je účast dětí v zájmových kroužcích.
Zadání je následující:

Pro děti klubu SEN se letos otevřel pouze A, B a C kroužek.
Každé dítě klubu SEN navštěvuje alespoň jeden z těchto tří kroužků - i dětí navštěvuje všechny tři kroužky, j dětí navštěvuje právě dva kroužky a ostatní děti jediný kroužek.
A kroužek navštěvuje a dětí, B b dětí a C c dětí.
Kolik dětí klubu SEN navštěvuje pouze jeden kroužek?
Kolik dětí je v klubu SEN?
Podmínky:

A,B,C jsou názvy kroužků (např. sportovní, divadelní a robotický)

$a \in \langle 11; 30)$

$b \in \langle 10; a)$

$c \in \langle 3; 10)$

$i \in \langle 1; 6)$

$j \in \langle 1; 11 - i)$

$a + b + c > 3i + 2j$

$a, b, c, i, j \in \mathbb{N}$

Řešení: $x_1 = a + b + c - (3i + 2j), \ x_2 = i + j + x_1$

Vyjádření vztahu proměnnou, soustava rovnic

Úlohy na vyjádření počtu prvků na základě daného vztahu a proměnné. V zadání lze dynamicky měnit nejen čísla vyjadřující vztah, ale také popis situace, jaké prvky

jakého celku počítáme. Jedním z příkladů je příklad na vztah ceny dvou věcí.
Za a *věcí 1* zaplatíme v *obchodě* celkem x korun, stejně jako za b *věcí 2*.

1) Vyjádřete výrazem s proměnnou x, kolik korun zaplatíme za 1 *věc 1*.

2) Vyjádřete výrazem s proměnnou x, kolik korun zaplatíme za c *věcí 2*.

3) V *obchodě* jsme za d *věcí 1* a e *věcí 2* zaplatili celkem k korun.

Vypočtěte kolik korun jsme zaplatili za jednu *věc 1* nebo *věc 2*.
Podmínky:

$a, b \in \langle 2; 10), \ a \neq b$

$c \in \langle 2; 10)$

$c \neq a$, pokud se ptáme na cenu 1 *věc 1*

$c \neq b$, pokud se ptáme na cenu 1 *věc 2*

$d, e \in \langle 2; 10) \wedge db|a \wedge ea|b$

$a, b, c, d, e, k \in \mathbb{N}$

Řešení:

1) $\frac{x}{a}$

2) $\frac{c}{b}x$

3) Řešení soustavy 2 rovnic o 2 neznámých v_1, v_2 reprezentujících cenu věcí:

$dv_1 + ev_2 = k$

$av_1 = bv_2$

5.1.2 Geometrie v rovině a prostoru

Využití měřítka mapy

Slovní úlohy využívající měřítko mapy k určení skutečných rozměrů a naopak.

Každých a cm na turistické mapě rovinaté oblasti je ve skutečnosti b m.
Délka vycházkové trasy je přesně c km, což je $k-$*násobek* délky přímé trasy.
(Uvažované trasy nemají žádné převýšení.)
Otázky:

1) Kolik měří ve skutečnosti trasa, která na mapě měří d mm?

2) O kolik cm je vycházková trasa na mapě delší než trasa přímá?

3) Jaké je měřítko turistické mapy?

Podmínky: $a = p \cdot q, \ p, q \in \langle 2; 10)$

$d \in \langle 10; 100)$

$k \in \langle 2; 10)$

$g \in \langle 2; 12 - k)$

$b = a \cdot g \cdot 10$

$c = k \cdot g$

$a, b, c, d, p, q, k, g \in \mathbb{N}$

Řešení:

1) $\frac{d \cdot b}{a}$

2) $100 \cdot (c - \frac{c}{k}) \cdot \frac{a}{b}$

3) $1 : \frac{1000b}{a}$

5.2 Osmileté obory

5.2.1 Číslo a početní operace

Určení části celku a porovnání

Výpočet, o kolik se liší části 2 celků. Obecně můžeme zapsat zadání jako:

O kolik <unit> se liší $\frac{1}{b}$ *z a <unit> a* $\frac{1}{d}$ *z c <unit>.*

<unit> - jednotka, například litry nebo kilogramy

$a > b, c > d$

$\frac{a}{b} > \frac{c}{d}$

$a|b,\ c|d$

Řešení: $\frac{a}{b} - \frac{c}{d}$

Početní úlohy

Úlohy na klasický početní příklad, tedy vypočtení výsledku, případně nalezení některého z členů. V tomto případě bylo využito několik příkladů z testů tak, že byla zachována struktura příkladu, závorky a početní operace, ale čísla jsou generována náhodně s použitím pravidel na dělitelnost a velikost čísel, tak aby vycházely přiměřeně velké a celočíselné výsledky. Kostry příkladů jsou následující a jsou vybírány náhodně.

1) $(a - b - c) : (d - e \cdot f) + g = ?$

$a \in \langle 30; 99)$

$b, c \in \langle 1; 9)$

$c \cdot b > a$

$d \in \langle 2; 9)$

$e \in \langle 4; 64)$

$e|d$

$a, b, c, d, e, f, g \in \mathbb{Z}$

2) $a : b\ +- c = d$

$b \in \langle 10; 90)$

$c \in \langle 1; 10)$

$d \in \langle 10; \frac{1000}{b})$

$a = b \cdot (d - c)$

$a, b, c, d \in \mathbb{Z}$

a nebo b je vynecháno a řešitel musí najít správné číslo, aby rovnost platila.

3) $a : b = c,\ zbytek\ d$

$b \in \langle 10; 90)$

$c \in \langle 10; \frac{1000}{b})$

$d \in \langle 1; 10)$

$a = b \cdot c + d$

$a, b, c, d \in \mathbb{Z}$

a nebo b je vynecháno a řešitel musí najít správné číslo, aby rovnost platila.

4) $a - b \cdot c = d + e : f$

$b \in \langle 2; 49)$

$c \in \langle 2; \frac{100}{b})$

$a = b \cdot c + p,\ p \in \langle 1; 100 - b \cdot c)$

$f \in \langle 2; 10)$

$e \in f \cdot q,\ q \in \langle 2; 10)$

$d = a - b \cdot c - \frac{e}{f}$

$a, b, c, d, e, f, p, q \in \mathbb{Z}$

jedno z čísel a, b, c, d, e, f je vynecháno a řešitel musí najít správné číslo, aby rovnost platila.

5) $(a : b - c) \cdot d = e \cdot f$

$b \in \langle 2; 10)$

$a \in \langle 2b; 100)$

$c \in \langle 1; \frac{a}{b} - 1)$

$d \in \langle 2; 10)$

$((\frac{a}{b} - c) \cdot d) | e$

$a, b, c, d, e, f \in \mathbb{Z}$

jedno z čísel a, b, c, d, e, f je vynecháno a řešitel musí najít správné číslo, aby rovnost platila.

6) $(a - b - c) : (d - e \cdot f) + g = ?$

$g \in \langle 11; 100)$

$e, f \in \langle 2; 10)$

$1 \leqq d - e \cdot f \leqq 10$

$c \in \langle 10; 100)$

$b \in \langle 100; 1000)$

$a = b + c + p \cdot (d - e \cdot f),\ p \in \langle 10; 989)$

$a, b, c, d, e, f, g \in \mathbb{Z}$

7) $a \cdot b - (c + d \cdot e) : f = ?$

$a, b \in \langle 10; 100),\ a|10,\ b|10$

$f \in \mathbb{N},$

$a, b, c, d, e, f \in \mathbb{Z}$

8) $a \cdot b - (c + d \cdot e) : (f - g + h) = ?$

$a, b \in \langle 10; 100),\ a|10,\ b|10$

$p \in \langle 10; 100)$

$f \in \langle 2; \lfloor \sqrt{p} \rfloor \rangle$

$d \in (2; \lfloor \sqrt{p} \rfloor \rangle$

$f \neq d$

$q \in \langle 1; \frac{p}{d} - 1)$

$c = q \cdot d$

$e = \frac{p}{d} - q$

$a, b, c, d, e, f, g, h, p, q \in \mathbb{Z}$

9) $a \cdot (b - c) - (d \cdot e - f \cdot g) = ?$

$a \in \langle 2; 10)$

$f \in \langle 2; 9)$
$e \in \langle 110; 1000),\ e|10$
$p \in \langle 2; 5)$
$c = p \cdot e$
$d \in \langle f + 1; 10)$
$b \in \langle ep; 10e),\ b|e$
$g = e$
$a, b, c, d, e, f, g \in \mathbb{Z}$

10) $a - b \cdot c + d :\ e = ?$
$b, c \in \langle 2; 10)$
$e \in \langle 2; 10)$
$d \in \langle 2e; 10e),\ d|e$
$a \in \langle bc; 100 - bc)$
$a, b, c, d, e \in \mathbb{Z}$

5.2.2 Závislosti, vztahy a práce s daty

Použití jednotek a převody

Tento typ úloh pracuje s jednotkami a jejich převody. Typicky je zadáván jako slovní úloha.

- Rozdělení délky
  První parametrizovanou úlohou je úloha na rozdělení délky.
  Zadání zní následovně:

  Z <*objektu*> dlouhého t <*jednotka1*> jsme uřízli l_1 <*jednotka2*> dlouhých kusů a zbytek jsme rozdělili na l_2 stejně dlouhých dílů.

  Určete kolik <*jednotka1*> měří jeden díl.

  Podmínky:

  (<*jednotka1*>, <*jednotka2*>) $\in$
  $\{(centimetr, půlmetr), (centimetr, metr), (centimetr, decimetr)\}$
  $l_1 \in \langle 2; 10)$
  $l_2 \in \langle 3; 10)$
  r je poměr <*jednotka1*> : <*jednotka2*>
  $t = r \cdot l_1 + l_2 \cdot p,\ p \in \langle 10; 100) \wedge p|10$
  $l_1, l_2, t, p \in \mathbb{N}$

  Řešení: $\frac{t - l_1 \cdot r}{l_2}$
- Doba trvání cesty
  Další úlohou je úloha na změnu trvání cesty na základě změny dopravního prostředku. Máme vždy zadaný jeden dopravní prostředek a jak dlouho trvá cesta při jeho použití. Dále máme druhý prostředek s jehož použitím se cesta zkrátí nebo se bez jeho použití doba cesty prodlouží.
  Tím nám vznikají dvě možnosti zadání, které zní následovně:

  Cesta z A do B s využitím P_1 trvá t. Bez využití P_1 se doba cestování prodlouží o t_Δ.
  Vypočtěte, kolik minut trvá cesta z A do B bez využití P_1.

Cesta (*dopravním prostředkem*) P_1 z A do B trvá t. S využitím P_2 se cestování zkrátí o t_Δ.
Vypočtěte, kolik minut trvá cesta z A do B s využitím P_2.

Podmínky:

$t = \check{c}as\ v\ hodin\acute{a}ch\ a\ minut\acute{a}ch,\ 1h < t < 10h$

$t_\Delta \in \{\check{c}tvrt\ hodiny, p\mathring{u}l\ hodiny, t\check{r}i\ \check{c}tvrt\check{e}\ hodiny\}$

Řešení: $t \pm t_\Delta$

- Lomená čára

Lomená čára se skládá ze dvou úseček. Délka celé lomené čáry je a m b cm c mm a první/druhá úsečka je dlouhá d cm e mm.
Vypočtěte v mm délku druhé/první úsečky lomené čáry.
Alternativně: Vypočtěte o kolik mm se liší délky úseček lomené čáry.
Podmínky:

$a, b, c \in \langle 1; 10)$

$d \in \langle 10; 100)$

$e \in \langle d + 1; 100)$

$a, b, c, d, e \in \mathbb{N}$

Řešení: $1000a + 10b + c - 10d - e$, alternativně: $|1000a + 10b + c - 20d - 2e|$

- Představení s přestávkou

Představení trvalo i s přestávkou a hodin a b minut. Přestávka tvořila $\frac{1}{c}$ této doby.
Vypočtěte, kolik minut trvala přestávka.
Podmínky:

$c \in \langle 4; 10)$

$t \in \langle 10, 40 - c)$

$a = \lfloor \frac{ct}{60} \rfloor$

$b = ct\ mod\ 60$

$60a + b | c$

$a, b, c, t \in \mathbb{N}$

5.2.3 Nestandardní aplikační úlohy a problémy

- **Slovní úloha - Počet listů stromu**

Tato úloha obsahuje několik podúloh. Obecně jde o počet listů, vnitřních uzlů nebo všech uzlů stromu, který má pro každý uzel jedné úrovně pevně daný počet potomků. Zadání je slovní, vždy orientované na nějakou reálnou hierarchii, kterou lze reprezentovat stromem. V rámci parametrizace je tedy možné vytvořit několik modelových hierarchií se slovním popisem a dále dynamicky generovat počty potomků v každé hladině stromu. Strom má 3 úrovně, kde uzly z první a druhé úrovně obsahují 2 až 5 potomků a uzly třetí úrovně 5 až 15 potomků.

- **Vytvoření rovnice a využití vlastností násobení**

Obecně máme 2 čísla x,y a víme že jejich součin je roven nějakému zadanému C. Dále máme zadaná čísla A a B, jednociferná. Číslo x pak máme zvětšit A-krát a číslo y zmenšit B-krát a určit, jaký bude výsledek těchto upravených čísel.
Tedy dostáváme rovnice:

$x \cdot y = C$

$Ax \cdot \frac{y}{B} =?$

Pro vyřešení je nutno využít vlastností násobení, převést rovnici do tvaru:

$\frac{A}{B} \cdot x \cdot y =?$

a následně dosadit C:

$\frac{A}{B} \cdot C =?$

Čísla A,B,C generujeme tak, aby vyšel celočíselný výsledek a aby čísla A a B byla jednociferná.

Podmínky na čísla:

$A, B \in \langle 2; 10), \ A \neq B$

$C = k \cdot B, \ k \in \langle 2; 100)$

$A, B, C, k \in \mathbb{N}$

Základní kombinatorika

V testech se objevují i příklady na základní kombinatoriku řešitelné bez použití kombinatorických vzorců.

Parametrizovaný příklad je následující:

n <*osob*> si v mobilní aplikaci posílalo různé vzkazy. Vytvořili si mezi sebou jednak všechny možné k_1-členné skupiny, jednak všechny možné k_2-členné skupiny. Určete kolik vytvořili vytvořili k_1-členných a kolik k_2-členných skupin.

Podmínky:

$n \in \langle 5; 9)$

$k_1, k_2 \in \langle 2; n), \ k_1 \neq k_2$

$n, k_1, k_2 \in \mathbb{N}$

Řešení: $\frac{n!}{(n-k_1)!k_1!}, \ \frac{n!}{(n-k_2)!k_2!}$

5.3 Shrnutí

Parametrizace úloh vyžaduje poměrně velké úsilí a z toho důvodu nebyly pokryty všechny úlohy vyskytující se v testech. Zejména úlohy s grafickými prvky v zadání nebo úlohy konstrukční, které jsou mimo rozsah této práce. Je nutné vzít v potaz, že nejde pouze o generování úloh, ale současně i jejich řešení, aby bylo možné kontrolovat výsledky řešitele. U většiny úloh je také velmi důležité, aby čísla objevující se v úloze, výpočtu i výsledku byla přiměřeně velká a v mnoha případech navzájem dělitelná. To vede k mnoha omezujícím podmínkám, které je potřeba zohlednit při generování a parametrizaci jednotlivých úloh. Některé z neparametrizovaných úloh byly přidány do aplikace staticky, aby se zvětšila nabídka nabízených typů úloh.

Dalším zásadním aspektem je jazyk. Velká část úloh obsahuje slovní zadání a bohužel v tomto případě je nutné správně skloňovat a používat správné koncovky u slov. Typickým problémem je počet nějakých objektů, kdy se mění koncovka slova podle počtu. Například jeden stůl, dva stoly, pět a více stolů. Pro podobné případy byla vždy použita překladová funkce, která pro číslo na vstupu vrací slovo v daném tvaru.

V rámci slovních zadání máme možnost zadání parametrizovat nejen číselně, ale i tematicky. Můžeme tak obměňovat zadání, že jednou například cestujeme autobusem a jindy vlakem. Nejen, že díky tomuto vzniká dojem více druhů úloh, ale také si řešitel pravděpodobně uvědomí, že nezáleží na použitých slovech, ale právě na matematických vztazích, které popisují.

Kapitola 6
Analýza a návrh řešení

6.1 Specifikace požadavků

6.1.1 Funkční požadavky

Funkční požadavky definují, co by měl systém dělat, neboli jakou nabízí funkcionalitu.

- **FR01 - Generování úloh**
  Jako uživatel potřebuji, aby aplikace generovala úlohy, které se typicky objevují v testech přijímací zkoušky, abych si je mohl procvičovat.
- **FR02 - Kontrola výsledků**
  Jako uživatel potřebuji, aby aplikace kontrolovala správnost mých výsledků a ukazovala po kontrole výsledek správný, abych věděl, zda jsem byl při řešení úspěšný.
- **FR03 - Registrace a přihlašování uživatele**
  Jako uživatel se potřebuji registrovat a následně přihlašovat do aplikace, abych mohl sledovat svoje výsledky dlouhodobě při opakovaných návštěvách aplikace.
- **FR04 - Zaznamenávání výsledků**
  Jako uživatel potřebuji, aby aplikace zaznamenávala úspěšnost při řešení různých typů úloh, abych se mohl soustředit na oblasti, kde je moje úspěšnost horší.
- **FR05 - Zaznamenávání stráveného času při řešení úloh**
  Jako uživatel potřebuji, aby aplikace zaznamenával čas strávený při řešení úloh, abych věděl, které úlohy jsou pro mne časově náročnější a musím se zlepšit v jejich řešení a zároveň abych věděl, kolik jsem v jaký den strávil času procvičováním.

6.1.2 Nefunkční požadavky

Nefunkční požadavky nesouvisí s funkčním aspektem, ale definují vlastnosti systému. Typicky mezi tyto vlastnosti patří výkon, škálovatelnost, bezpečnost, spravovatelnost a podobně.

- **NFR01 - Dostupnost na internetu**
  Aplikace bude dostupná na internetu všem uživatelům bez omezení.
- **NFR02 - Nasazení na CodeNOW platformě**
  Aplikace bude nasazena na Value Stream Delivery Platformě CodeNOW.
- **NFR03 - API**
  Komponenty aplikace budou vystavovat REST API, tak aby byla aplikace snadno rozšiřitelná o další komponenty, které toto API mohou využívat.
- **NFR04 - Dostupnost na internetu**
  Aplikace bude navržena s mikroservisní architekturou, tak aby bylo snadné přidávat nové komponenty a škálovat potřebné části systému.
- **NFR05 - Autentizace a autorizace**
  Aplikace bude ověřovat uživatelovu identitu a oprávnění při operacích spojených s uživatelským účtem.

6.2 Případy užití

Případy užití vyjadřují, jakým způsobem bude uživatel systém používat.

- **UC01 - Přihlásit se**
  Uživatel se zadáním svého emailu a hesla přihlásí.
  Scénář:
  1. Uživatel klikne na přihlásit se.
  2. Systém zobrazí přihlašovací formulář.
  3. Uživatel vyplní své údaje a klikne na přihlásit.
  4. IF Uživatelem zadané údaje jsou validní.
     4.1. THEN Systém uživatele přihlásí, přesměruje na stránku uživatelského profilu a uloží token pro další komunikaci.
     4.2. ELSE Systém upozorní na nesprávnost údajů a nabídne možnost obnovit zapomenuté heslo. GOTO 2

- **UC02 - Registrovat se**
  Uživatel se registruje do aplikace.
  Scénář:

2.1. Systém zobrazí registrační formulář
2.2. Uživatel vyplní své údaje a klikne na registrovat.
2.3. Systém zvaliduje zadané údaje a pošle na uživatelem zadaný email potvrzovací email.
2.4. Uživatel na základě instrukcí v emailu aktivuje svůj účet.
2.5. Systém přesměruje uživatele na přihlašovací formulář.

- **UC03 - Odhlásit se**
  Uživatel se odhlásí ze systému.
  Scénář:

3.1. Uživatel klikne na Odhlásit se.
3.2. Systém uživatele odhlásí a zneplatní uživatelskou relaci.

- **UC04 - Procvičovat úlohy**
  Uživatel si procvičuje testové úlohy.
  Scénář:

4.1. Uživatel vybere typ přijímacích zkoušek, na které se připravuje.
4.2. Systém zobrazí seznam dostupných typů úloh.
4.3. Uživatel vybere typ úloh, který chce procvičovat.
4.4. Systém zobrazí náhodnou úlohu vybraného typu.
4.5. Uživatel vyřeší úlohu, zadá výsledek a klikne na zkontrolovat.
4.6. Systém porovná zadanou hodnotu s hodnotou řešení.
4.7. IF Řešení bylo správné
4.7.1. THEN Systém zeleně označí pole se správnou odpovědí a zobrazí hlášku o správnosti řešení.
4.7.2. ELSE Systém označí pole s odpovědí červeně a zobrazí hlášku o nesprávnosti odpovědi a správnou odpověď.
4.8. IF Uživatel je přihlášený THEN Systém zaznamená typ, správnost a čas strávený při řešení úlohy.

- **UC05 - Zobrazit záznamy o aktivitě**
  Přihlášený uživatel si zobrazí záznamy o aktivitě řešení úloh.

5.1. Uživatel otevře stránku uživatelského profilu.
5.2. Systém zobrazí informace o uživateli a tabulku s veškerou aktivitou.
5.3. Uživatel vybere filtry a klikne na Filtrovat.
5.4. Systém aktualizuje tabulku daty odpovídajícími filtrům.

6.3 Architektura

Pro aplikaci byla zvolena mikroservisní architektura hned z několika důvodů. Prvním důvodem je možnost škálování a rozšiřitelnosti. V současném stavu má aplikace na backendu jednu službu generující příklady, jednu pro zprávu statistik při řešení příkladů a jednu pro autentikaci uživatelů. Všechny backendové služby vystavují REST API, tedy není důležité, v jakém jazyce jsou implementovány a při dalším rozšiřování aplikace tak může vývojář zvolit libovolnou technologii a novou mikroslužbu napojit na existující APIs. Pokud by byla aplikace nasazena do samostatného prostředí, může pak i automaticky škálovat pod zátěží jednotlivé služby (více v sekci 9.3).
Komponenty aplikace:

- **Webová aplikace**
  Webová aplikace běžící v prohlížeči uživatele. Aplikace umožňuje procvičování příkladů. Pro registraci a přihlašování přesměrovává do webové aplikace služby Zitadel, od které pak zpětně dostává session token. S ostatními backendovými službami komunikuje zasíláním HTTP požadavků na REST APIs daných mikroslužeb.

- **Zitadel**
  Služba třetí strany poskytující autentikaci uživatelů. Vystavuje API, které je využíváno pro ověřování uživatelova tokenu, ale pro přihlašování má vlastní webovou aplikaci, na kterou je uživatel přesměrován z hlavní webové aplikace. Díky tomu není potřeba v implementaci řešit ukládání hesel a uživatelských citlivých údajů. O to vše se stará tato služba a vyvíjená aplikace pouze využívá poskytované rozhraní.

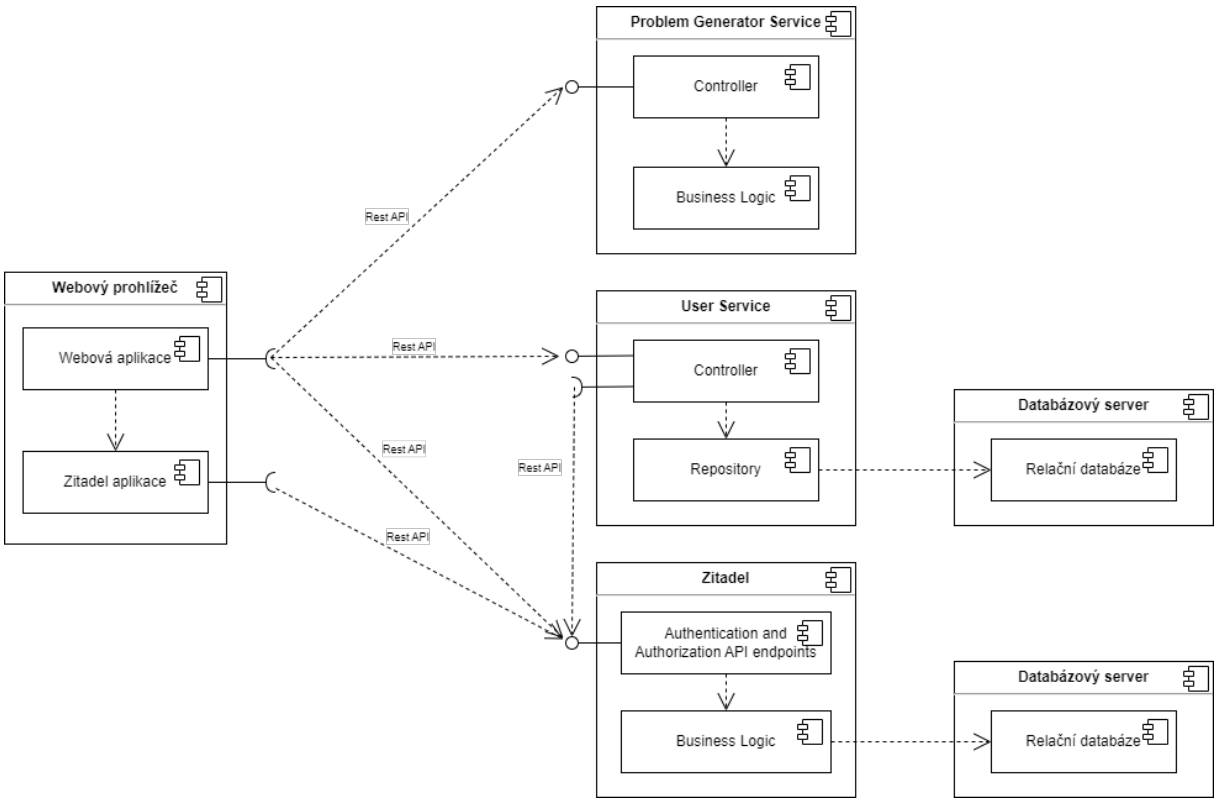
- **Problem Generator Service**
  Pravděpodobně nejdůležitější část aplikace, která je zodpovědná za generování příkladů k procvičování. Nabízí veřejné REST API, které poskytuje generované příklady všem, i nepřihlášeným, uživatelům.

- **User Service**
  Mikroslužba odpovídající za shromažďování statistik o délce a úspěšnosti řešení příkladů. Vyžaduje přihlášení uživatele, které je kontrolováno pomocí předaného tokenu v hlavičce dotazu. Token je pak kontrolován vůči autentikační službě Zitadel.

- **Databázový server**
  Databázový server slouží k perzistenci všech potřebných dat. Jednak je využíván User Service a jednak autentikační službou Zitadel. Každá ze služeb může používat jinou databázi na jiném serveru. Naopak v mikroservisní architektuře by bylo chybou, pokud by dvě služby využívali stejnou databázi.

Komunikace mezi jednotlivými službami probíhá synchronně, jelikož aplikace neobsahuje složitější vícekrokové operace, které by byly zpracovávány více službami. Pokud v budoucnu tato situace nastane, bude vhodné použít managed service Apache

Kafka, integrovanou do CodeNOW, pro asynchronní komunikaci mezi službami. Jediná komunikace mezi backendovými službami probíhá mezi User Service, která zaznamenává uživatelovu aktivitu při řešení úloh a Zitadel, službou pro autentikaci. Zitadel poskytuje REST API a je implementována třetí stranou, tedy není možné využít jiný způsob komunikace.



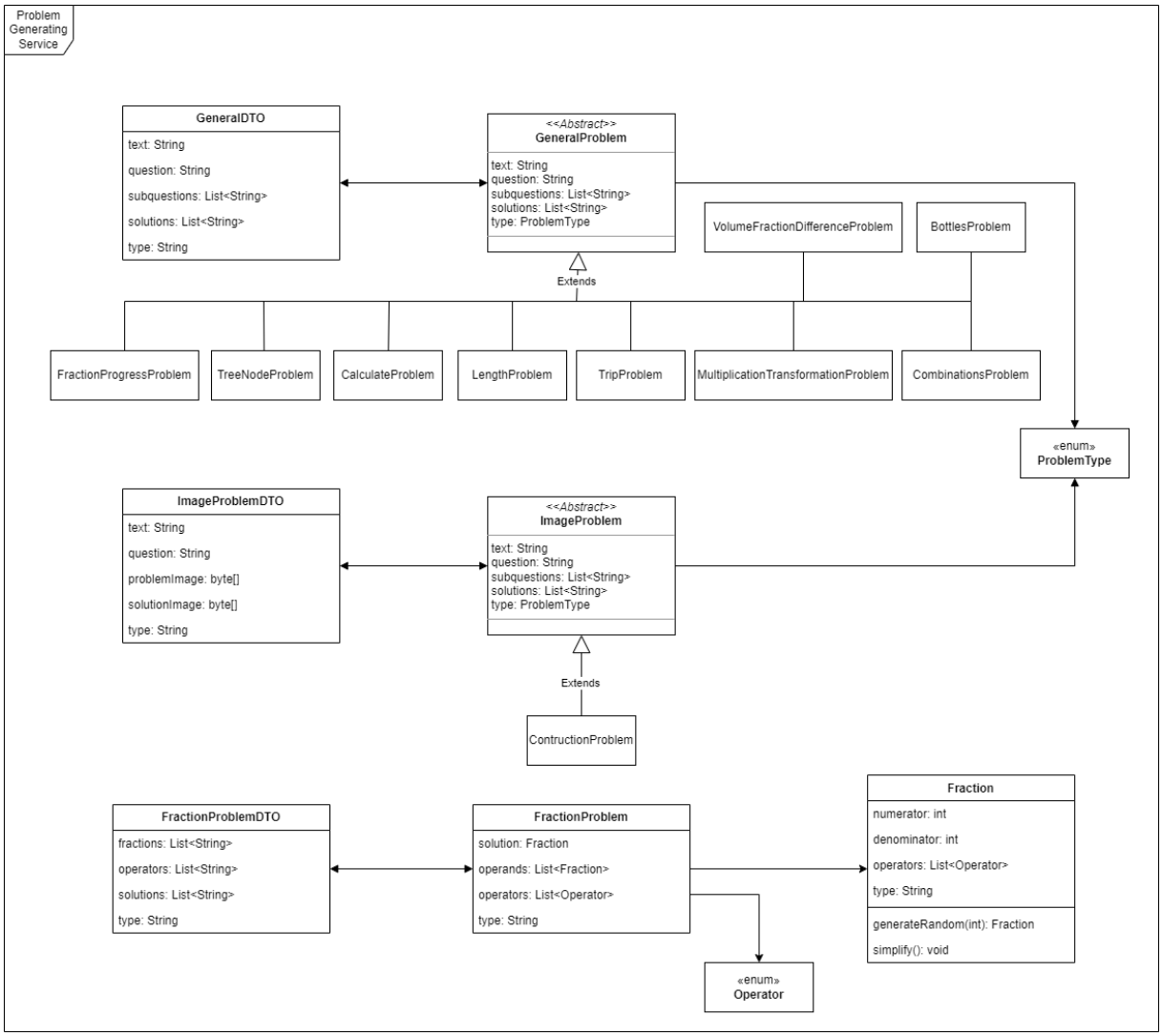
Obrázek 6.1. Architektura

6.4 Datový model

Datový model je poměrně jednoduchý, jelikož hlavní funkcionalitou aplikace je generovat příklady k procvičování a ty nejsou nikde perzistovány.

6.4.1 Problem Generator Service

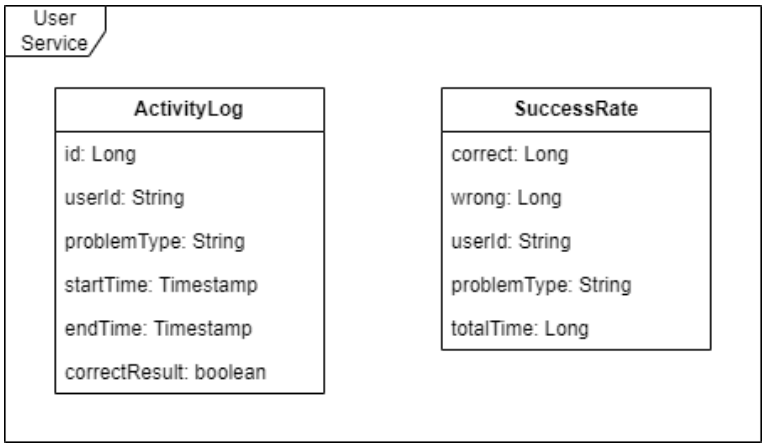
Hlavními entitami jsou tedy entity příkladů, většina z nich je odvozena od nadřazené abstraktní třídy GeneralProblem a pouze některé specifické příklady mají trochu odlišnou strukturu, jmenovitě FractionProblem obsahující příklady na zlomky nebo příklady s obrázkem v zadání. Nadřazené abstraktní třídy mají pak mapování do Data Transfer Objektů, které jsou serializovatelné a jako JSON objekty jsou předávány v tělech HTTP odpovědí.



Obrázek 6.2. Problem Generator Service - Datový model

6.4.2 User service

Služba starající se o uživatelské statistiky využívá pouze jedinou entitu nazvanou ActivityLog, která reprezentuje jeden řešený příklad. Záznam obsahuje typ příkladu, zda byl příklad vyřešen správně, začátek a konec řešení a který uživatel příklad řešil.



Obrázek 6.3. User Service - Datový model

Kapitola 7
Implementace

Implementace aplikace pro přípravu na přijímací zkoušky byla zaměřena na vytvoření komponent, které spolu budou správně komunikovat a interagovat tak, aby si uživatel mohl procvičovat typické úlohy objevující se v přijímacích zkouškách, sledovat svoji úspěšnost a čas strávený při řešení úloh. Zároveň by implementace měla umožňovat poměrně snadné přidávání dalších příkladů a dalších komponent například pro procvičování úloh z geometrie nebo českého jazyka. Pro implementaci jednotlivých částí aplikace byly zvoleny následující technologie, vybrané především na základě jejich popularity a rozšířenosti, aby v projektu mohl případně někdo co nejsnadněji navázat.

7.1 Použité technologie

7.1.1 Frontend

Frontend, tedy webová aplikace byla implementována v React.js [8]. React je v současnosti nejpopulárnější [9] a zároveň nejžádanější javascriptový framework na trhu práce [10]. Díky velké popularitě existuje mnoho návodů, dokumentací, tutoriálů a knihoven.

Pro styling webové stránky byl použit frontendový toolkit Bootstrap [11], který slouží k rychlému vytvoření responzivních aplikací. Hlavní výhodou je responzivní chování při použití Bootstrap komponent, díky kterému je aplikace použitelná na téměř všech zařízeních, bez nutnosti složitého manuálního vytváření kaskádových stylů.

7.1.2 Backend Services

Služby fungující jako backend aplikace byly implementovány v jazyce Java [12] s použitím Spring Boot [13] frameworku. Všechny služby vystavují REST API a jeho Swagger OpenAPI definici [14].

7.1.3 Databáze

Jako databáze byla zvolena open source relační databáze PostgreSQL [15]. Pro ukládání používaných dat se relační databáze hodí nejvíce a s PostgreSQL jsem měl už předchozí dobré zkušenosti a zároveň je integrovaná do CodeNOW jako managed service. Je tedy velmi jednoduché vytvořit novou instanci nebo se připojit k nějaké stávající.

7.1.4 Autentikace uživatelů

Pro autentikaci uživatelů byla zvolena aplikace Zitadel, kterou je možné provozovat self-hosted, jelikož je vydávána i pod open source licencí, a nebo využívat možnosti SaaS. Zitadel používá koncepty event sourcingu a CQRS k dosažení úplného audit

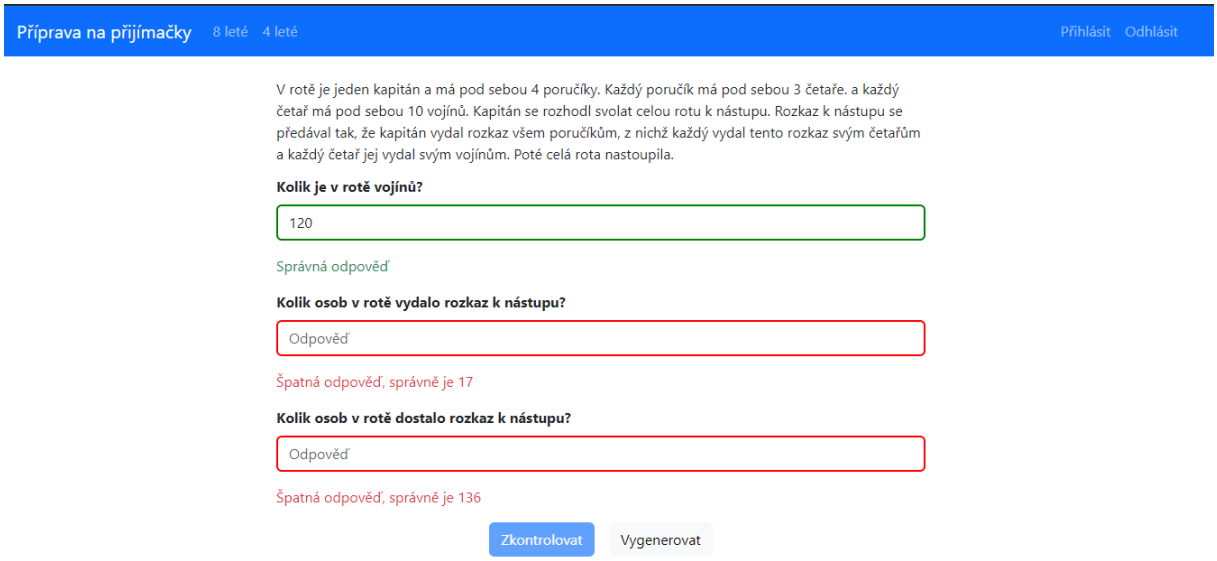
trailu a je celkově moderním a elegantním řešením. V řešení byla nakonec využita možnost SaaS kvůli problémům s nasazením instance Zitadel do prostředí CodeNOW. Výhodou použití SaaS je automatické škálování, jelikož služba běží v cloudu, ale nevýhodou, že v neplaceném režimu je instance omezená na 25 000 požadavků měsíčně. Velkou výhodou použití této aplikace je mimo usnadnění práce s implementací také to, že funguje v souladu s GDPR a OpenID. Zitadel také umožňuje vynutit dvoufaktorové ověření nebo povolit jiné identity providery. [16]

7.2 Uživatelské rozhraní

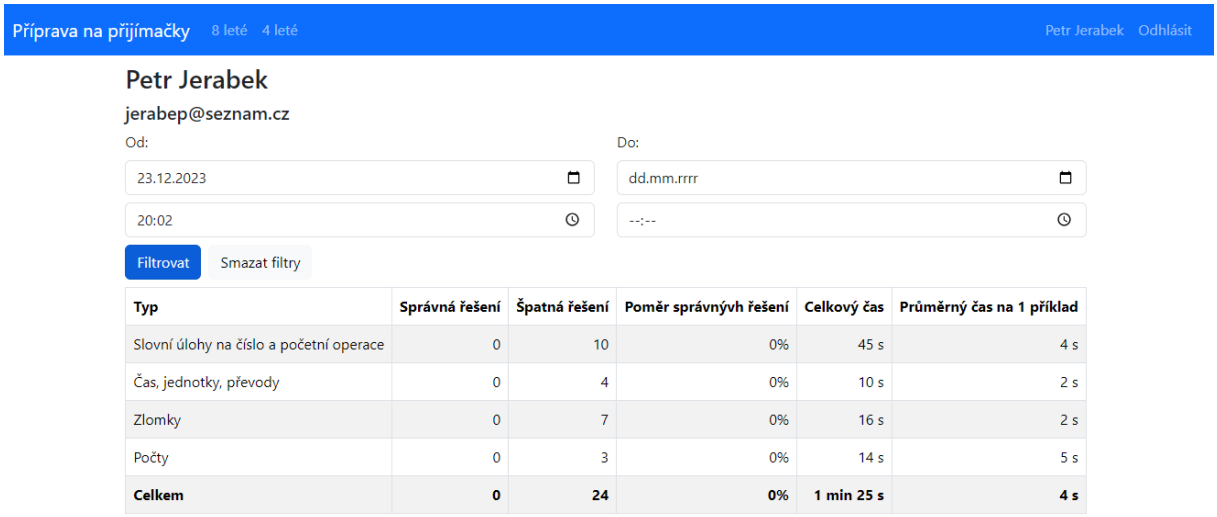
7.2.1 Webová aplikace

Webová aplikace s poměrně jednoduchým designem slouží uživatelům k procvičování příkladů a sledování výsledků. Pro přihlašování a registraci jsou uživatelé přesměrováni do webové aplikace Zitadel.

Obrázek 7.1. Uživatelské rozhraní - Příklad na zlomky



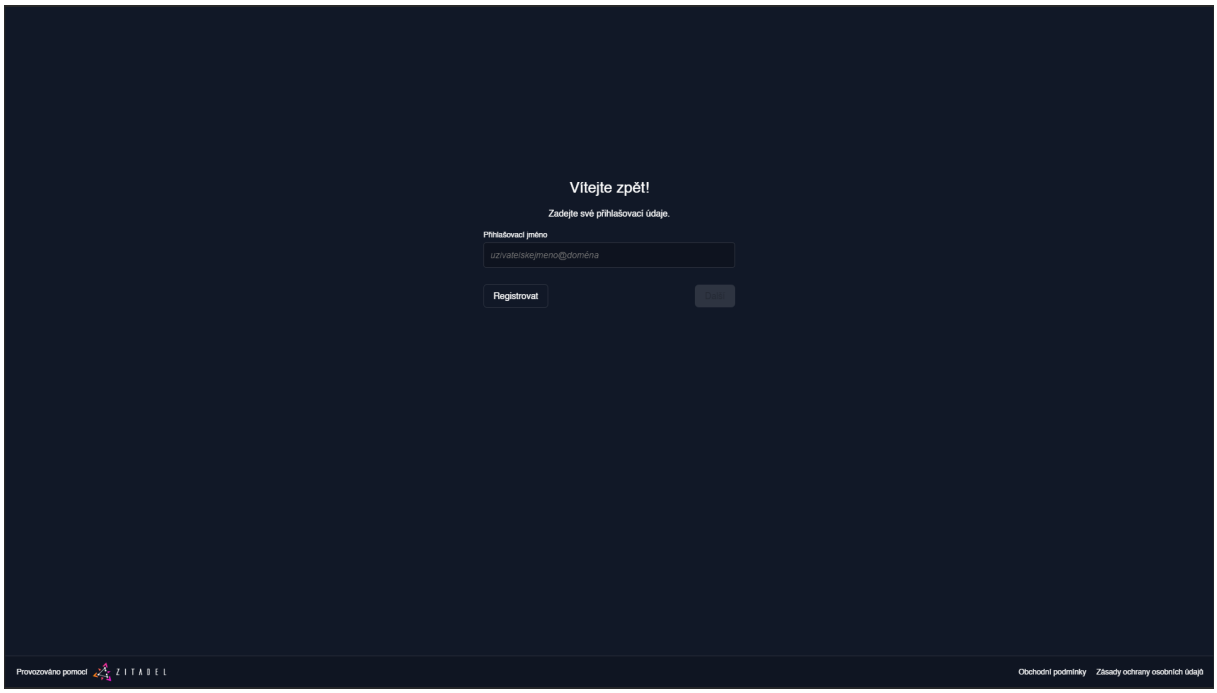
Obrázek 7.2. Uživatelské rozhraní - Příklad s více odpověďmi



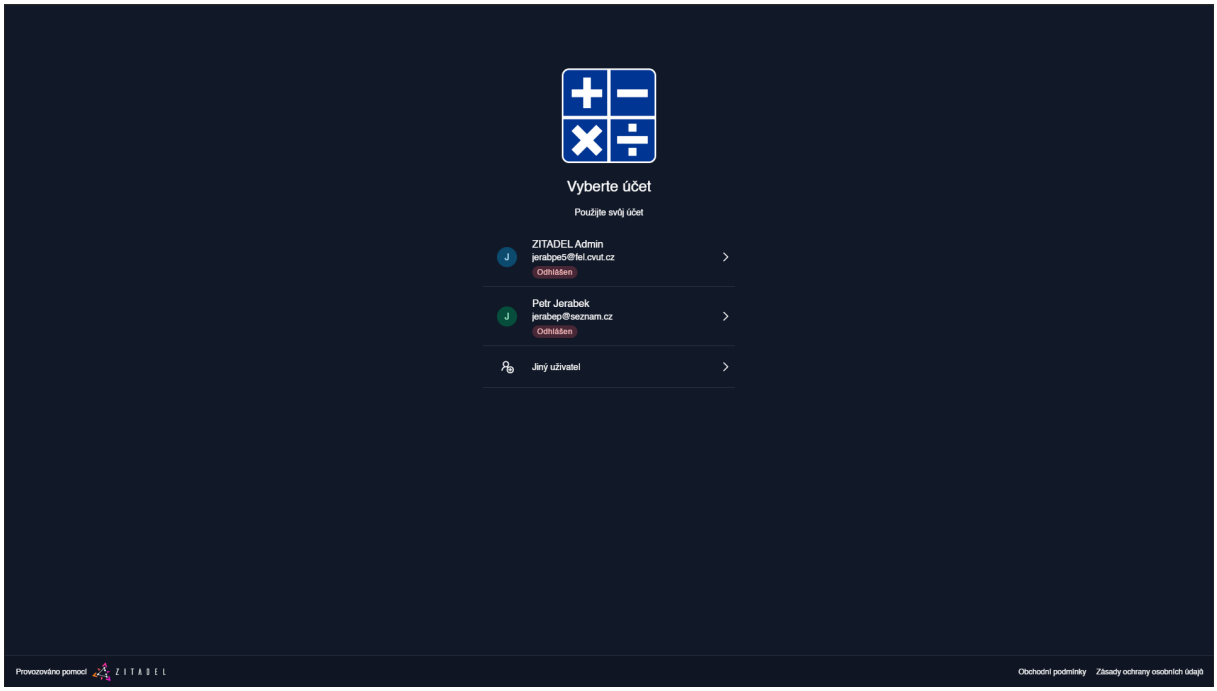
Obrázek 7.3. Uživatelské rozhraní - Přehled uživatelovy aktivity

7.2.2 Zitadel

Uživatelské rozhraní Zitadel je konfigurovatelné, co se týče barev, fontů i zobrazovaných textů skrz administrátorskou webovou aplikaci. Uživatel vyvíjené aplikace se setká pouze s obrazovkami na přihlašování a registraci. Pokud je uživatel na stránce poprvé, uvidí prázdný přihlašovací formulář (obrázek 7.4). Pokud se uživatel vrací a již byl ve svém prohlížeči někdy přihlášen, vidí již dříve přihlášené profily (obrázek 7.5).



Obrázek 7.4. Uživatelské rozhraní - Zitadel přihlašovací formulář



Obrázek 7.5. Uživatelské rozhraní - Zitadel přihlašovací formulář

Kapitola 8
Deployment

Součástí této práce bylo průběžné nasazování vyvíjené aplikace na CodeNOW Value Stream Delivery platformu.

8.1 Cloud Native

Technologická řešení postavená na principech cloud native umožňují vytvářet a provozovat škálovatelné aplikace v moderním prostředí cloudu, ať už veřejném, privátním nebo hybridním. Jako příklady této metodiky můžeme zmínit kontejnery, microservices, service mesh nebo deklarativní API. Díky těmto a dalším technikám je možné vytvářet systémy, které jsou propojené, odolné, spravovatelné a pozorovatelné. Velmi časté je použití robustní automatizace, která umožňuje dělat časté změny za cenu minimálního úsilí. [17]

8.2 Cloud Native Computing Foundation

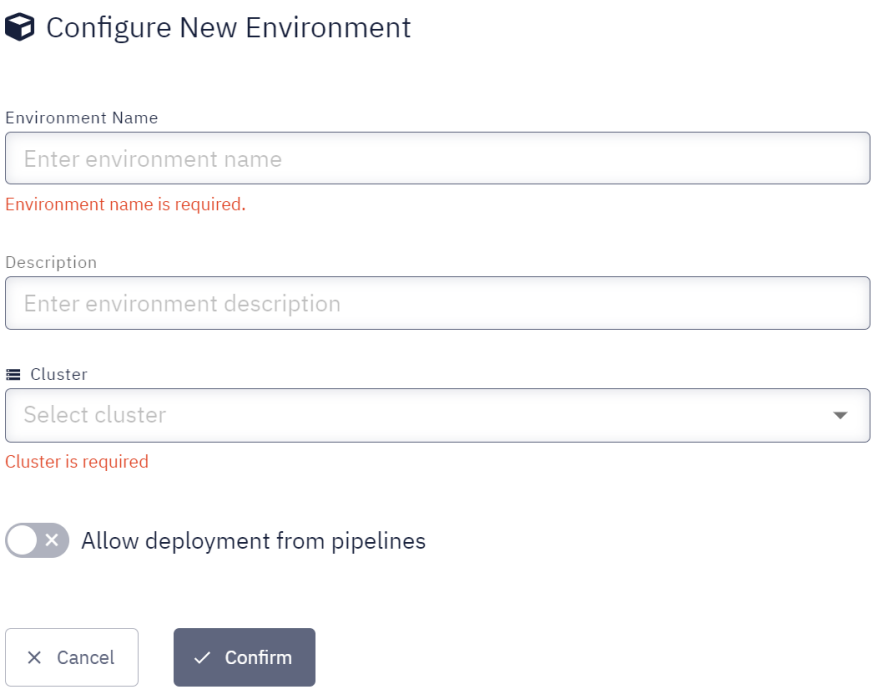
Cloud Native Computing Foundation je součást organizace Linux Foundation zaměřená na podporu, dohled a vedení rychle se rozvíjejících cloudových projektů a cloud-native technologií. Mezi nejvýznamnější z těchto technologií patří například Kubernetes. Cílem této organizace je podporovat a udržovat ekosystém open-source, vendor-neutral projektů a zpřístupnit všechny inovace každému. [17]

8.3 Platforma CodeNOW

CodeNOW je Value Stream Delivery Platform, neboli platforma pro doručování hodnotového produktu, vytvářená v rámci ekosystému Cloud Native Computing Foundation (CNCF). Cílem a výhodou takové platformy je snížení potřeby odborníků a znalostí z oblasti DevOps, potřebných pro doručování cloud-native aplikací. Platforma by také měla zlepšovat výkonnost a user experience vývojářů cloudových aplikací. [18]

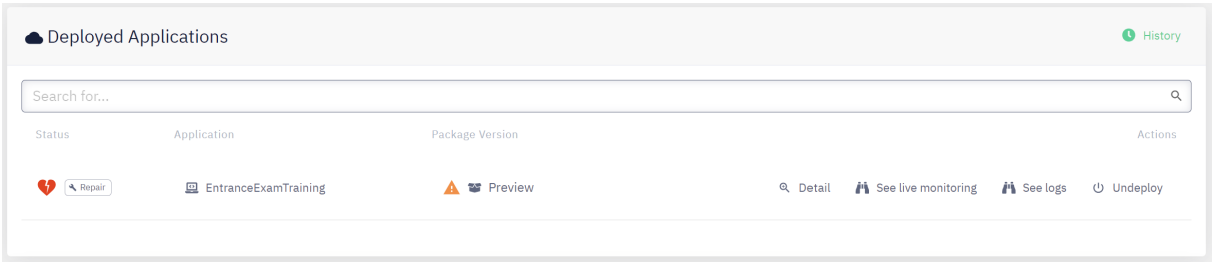
8.3.1 Vytvoření prostředí (Enviroment)

Prvním krokem při nasazování aplikace je vytvoření prostředí, ve kterém naše aplikace poběží. V CodeNOW stačí pouze vyplnit název a vybrat cluster a o zbytek se postará automatizovaný proces.



Obrázek 8.1. CodeNOW - Vytvoření nového prostředí

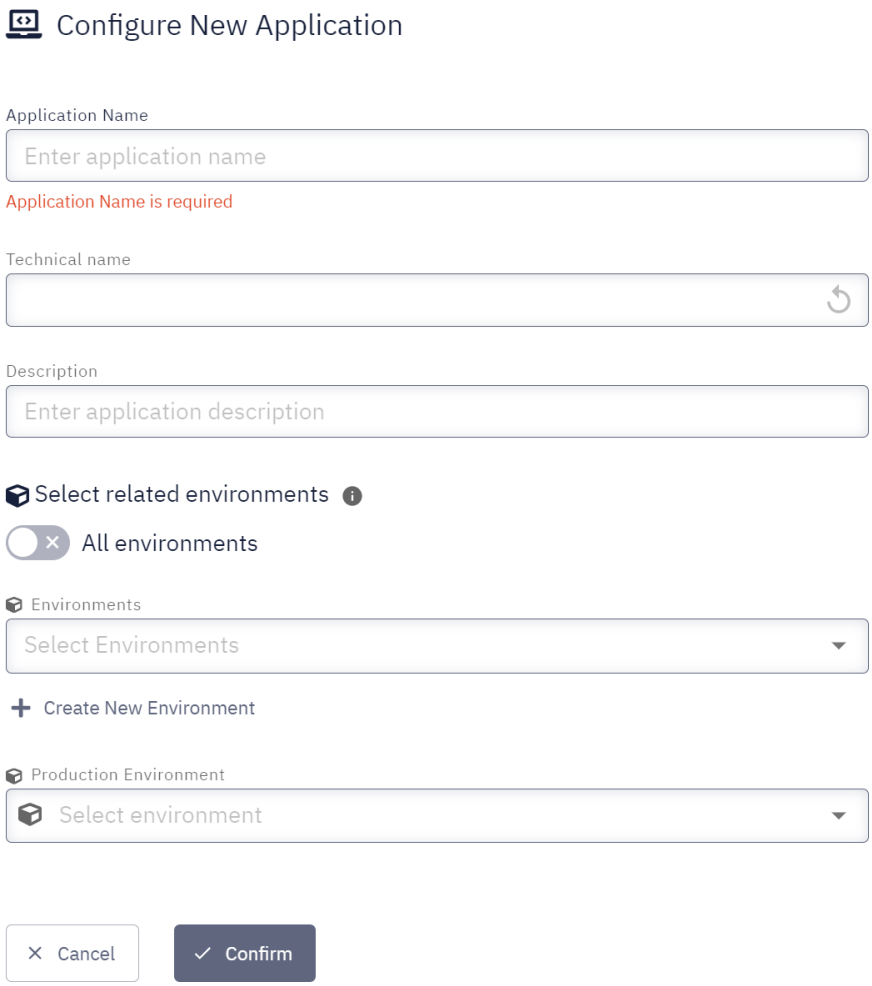
Ve vytvořeném prostředí pak můžeme nasazovat a monitorovat naše aplikace, spravovat konfigurace a připojené služby.



Obrázek 8.2. CodeNOW - Přehled nasazených aplikací ve vybraném prostředí

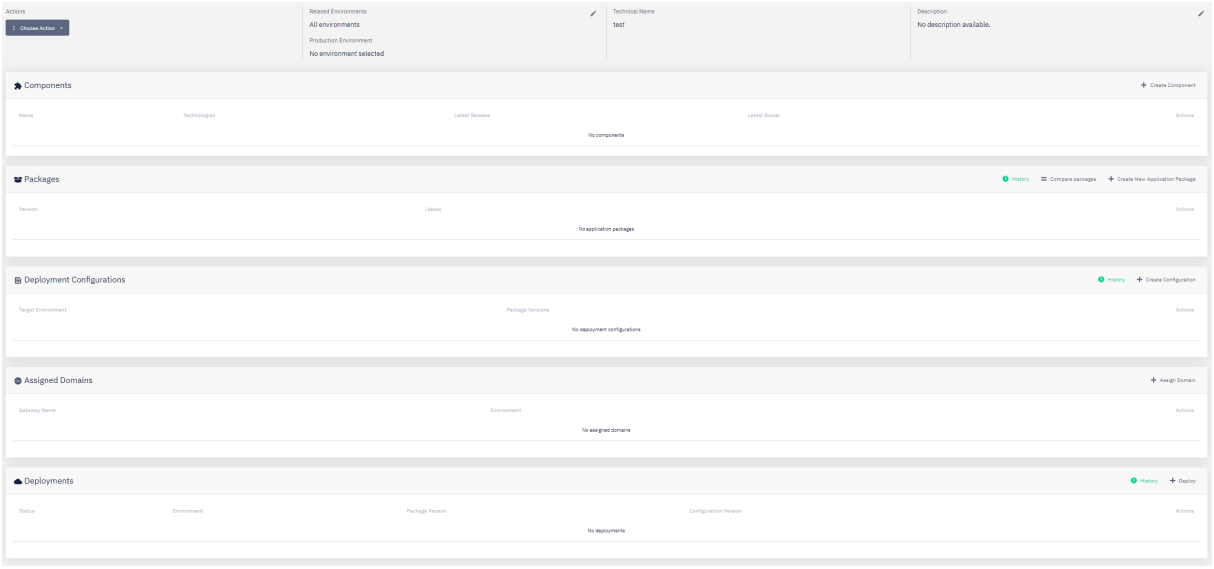
8.3.2 Vytvoření aplikace

V dalším kroku si vytvoříme aplikaci opět skrz jednoduchý formulář, kde vyplníme jméno a nastavíme související prostředí, na která bude možné aplikaci nasazovat, a jedno produkční prostředí.



Obrázek 8.3. CodeNOW - Vytvoření nové aplikace

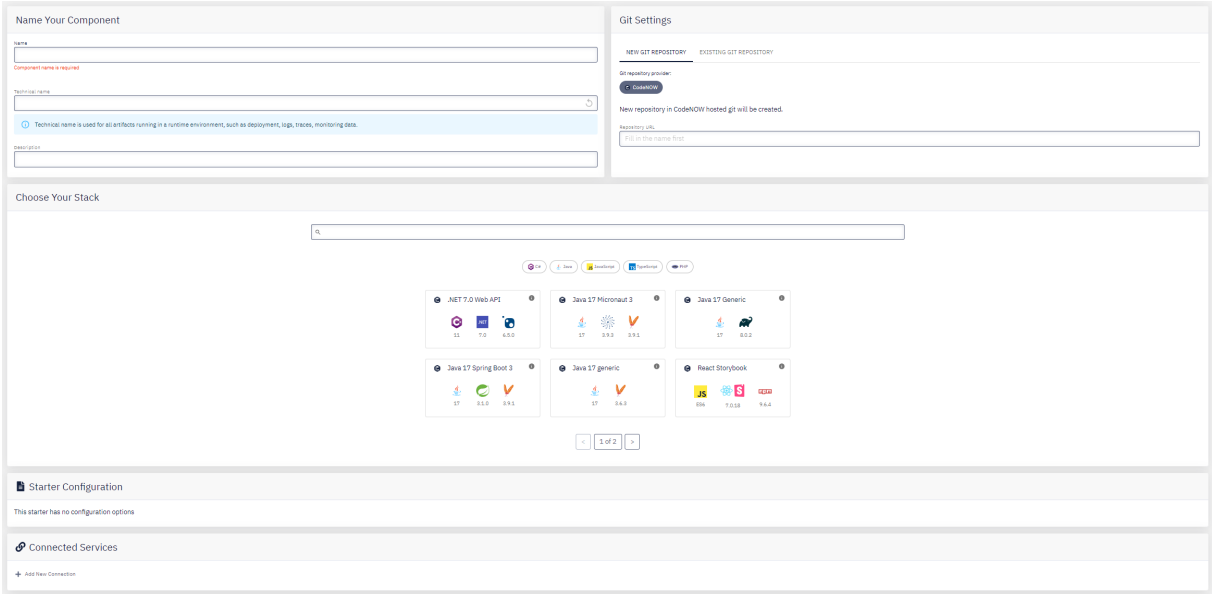
V detailu vytvořené aplikace pak můžeme spravovat jednotlivé komponenty, vytvářet balíčky a konfigurace, nastavovat domény a nasazovat aplikaci do relevantních prostředí.



Obrázek 8.4. CodeNOW - Detail aplikace

8.3.3 Vytvoření komponent

Pokud máme vytvořenou aplikaci, můžeme začít vytvářet jednotlivé komponenty aplikace. Při vytvoření komponenty si vybíráme technologický stack, který bude komponenta používat a dále volíme, zda už máme existující git repozitář nebo chceme aby byl vytvořen. Pokud vytváříme komponentu na čisto, současně s ní se nám vytvoří git repozitář s prvotní konfigurací a minimálním projektem v dané technologii, abychom mohli komponentu rovnou spustit. Repozitář si pak stačí už jen naklonovat a lze začít vyvíjet.



Obrázek 8.5. CodeNOW - Vytvoření komponenty

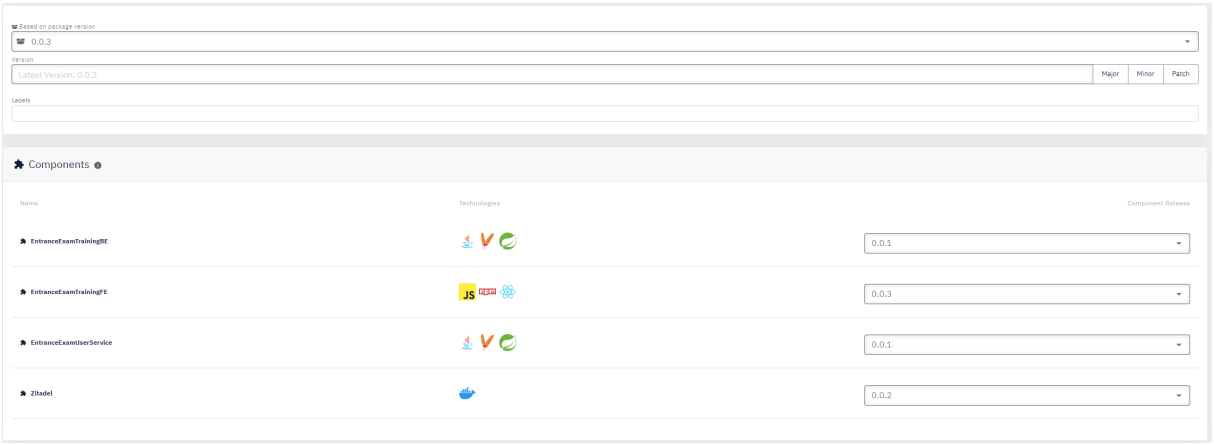
8.3.4 Managed services

Managed services jsou speciální druh komponent, které implementují funkcionality společné pro mnoho různých aplikací. Typicky potřebné managed services jsou úložiště dat, user identity provider nebo prostředek pro komunikaci mezi jednotlivými komponentami. Vytvoření managed service je také automatizované a my pouze vybereme typ, zadáme jméno, vybereme prostředí, ve kterém služba poběží a případně upravíme konfiguraci. Po vytvoření se lze ke službě připojit a lze ji nastavit jako Connected Service pro jednotlivé komponenty, kdy jsou potom předávány connection properties jako proměnné běhového prostředí komponenty.

8.3.5 Build, Package & Deploy

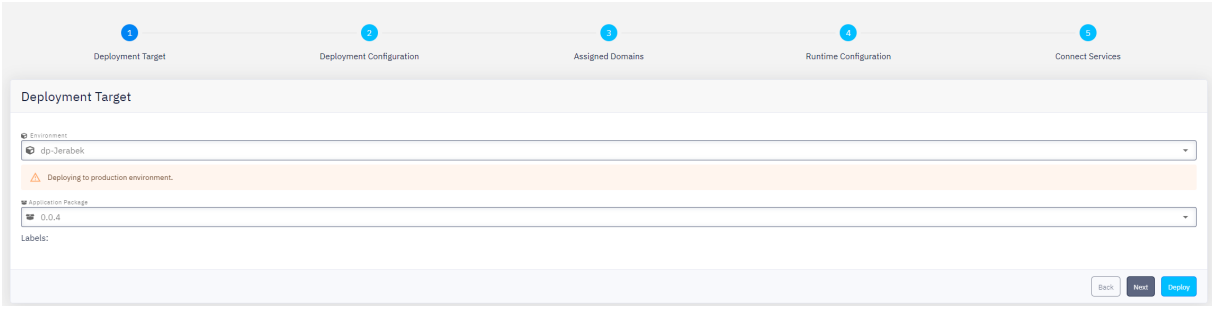
Pokud máme připravené všechny komponenty a manged services, můžeme nasazovat.

1. Nejprve vytvoříme Package z vybraných komponent a jejich verzí a samotný balík označíme číslem verze.



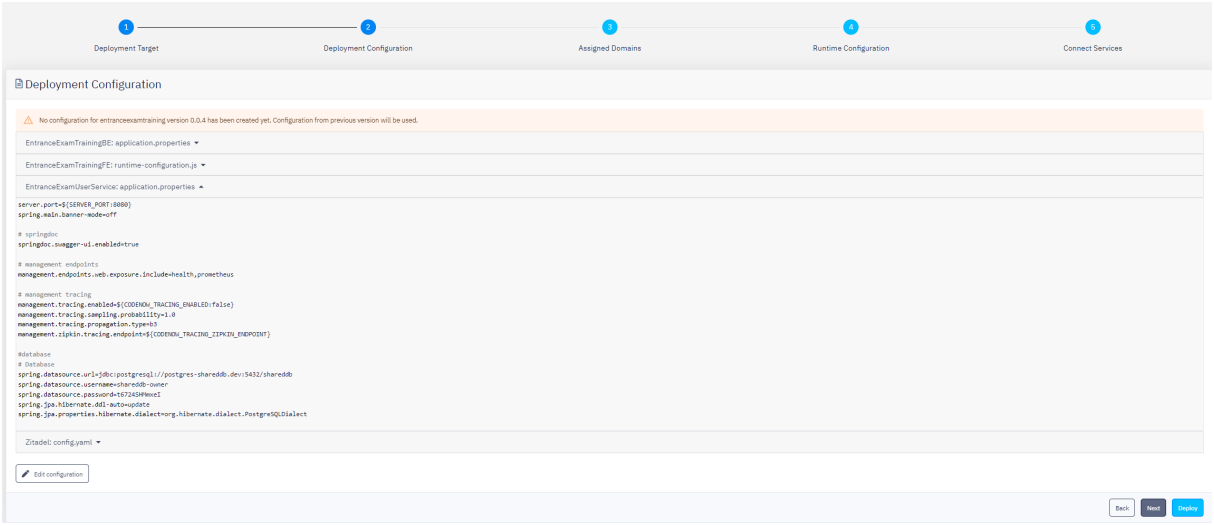
Obrázek 8.6. CodeNOW - Vytvoření Package

2. Vybereme Package a Enviroment, kam chceme nasadit



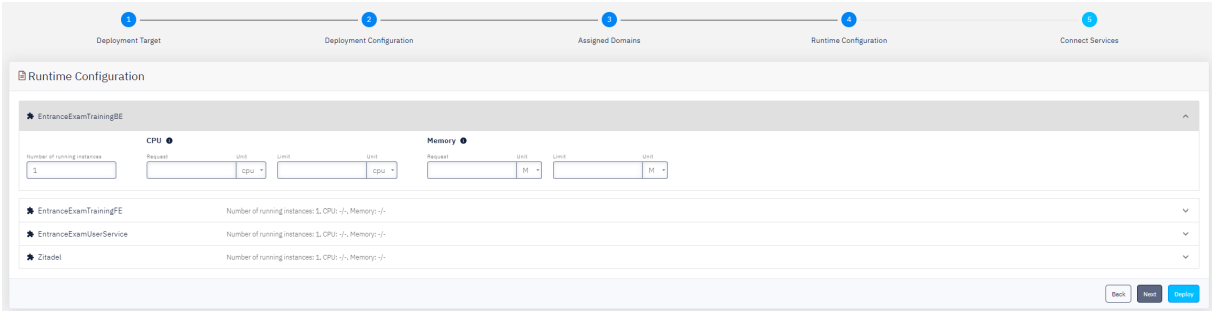
Obrázek 8.7. CodeNOW - Deployment Target

3. Zkontrolujeme a případně upravíme konfigurace jednotlivých komponent



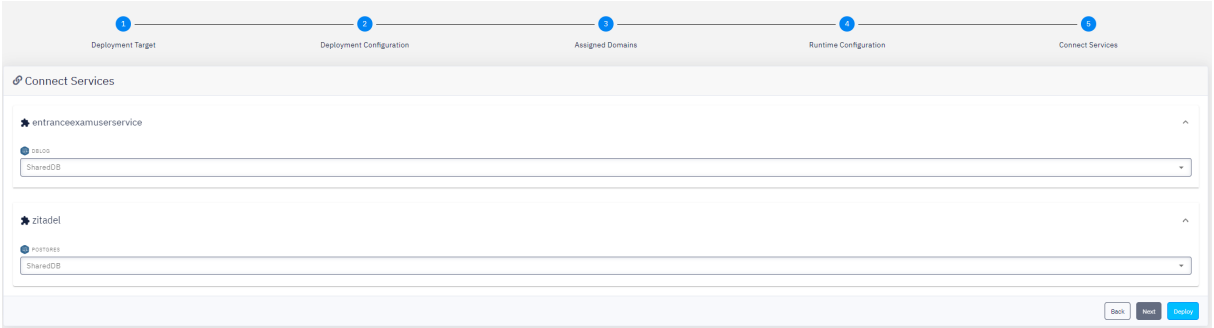
Obrázek 8.8. CodeNOW - Deployment Configuration

4. Assigned Domains - Nastavení domén, pokud máme nějaké přidružené k aplikaci a prostředí.
5. Zkontrolujeme a případně upravíme Runtime konfiguraci pro jednotlivé komponenty - počet instancí, počet CPUs a velikost RAM.



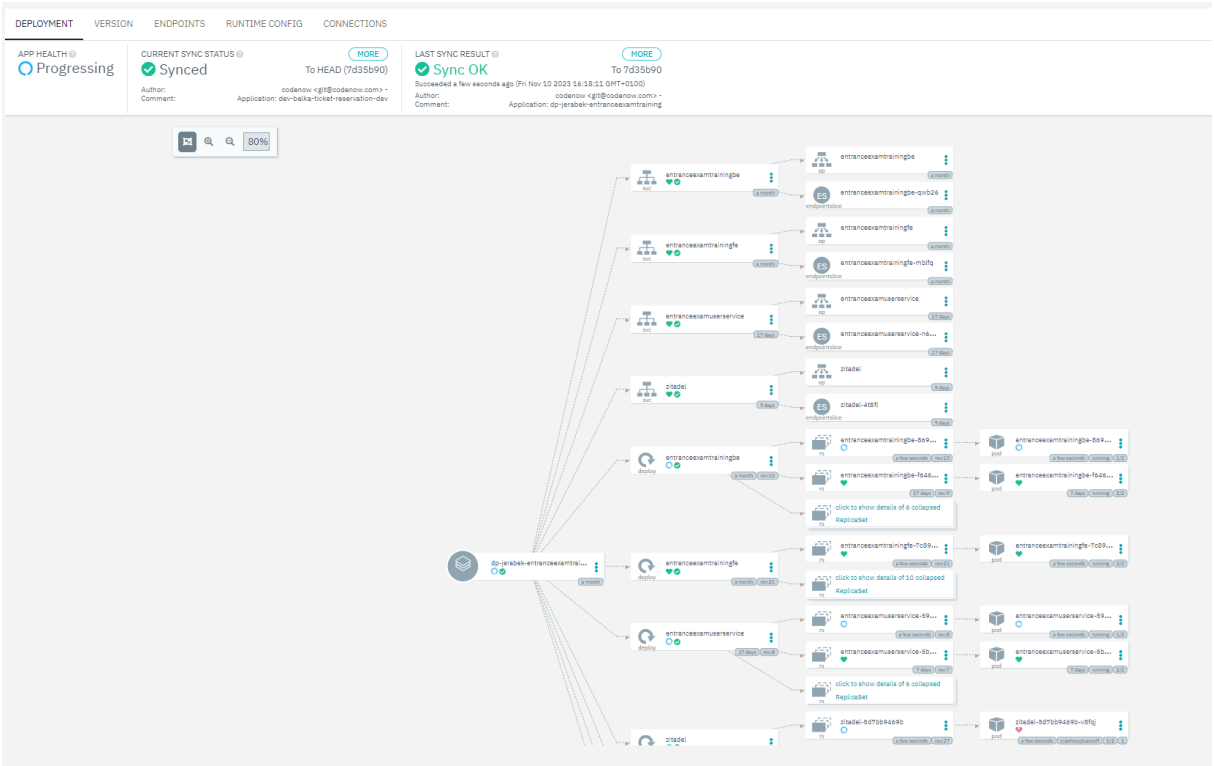
Obrázek 8.9. CodeNOW - Runtime konfigurace

6. Zvolíme správné instance managed services pro každou kompnentu



Obrázek 8.10. CodeNOW - Deploy - Connected Services

7. Spustíme nasazení a můžeme sledovat průběh a stav jednotlivých částí aplikace. V dalších záložkách pak zobrazit a případně změnit vystavené endpointy, konfigurace a připojené služby.



Obrázek 8.11. CodeNOW - Deployment

8.4 Prostředí

CodeNOW prostředí využívá technologii Kubernetes pro automatické nasazování, škálování a správu kontejnerizovaných aplikací [19]. Jednotlivé komponenty aplikace jsou pak vytvářeny v rámci Kubernetes Pods. Každý Pod pak kromě dané komponenty obsahuje Istio Envoy Proxy, což je běžný princip sidecar pattern v kontejnerizovaných aplikacích. Tato služba kontroluje pomocí definovaného healthchecku, zda je komponenta zdravá a připravená k použití a hlavně vytváří service mesh. Service mesh pak umožňuje přidávat sledování, traffic management a zabezpečení bez zásahu do kódu komponenty.

8.5 Konfigurace

8.5.1 Zitadel

Při nasazení na CodeNOW jako Docker komponenta je potřeba upravit soubor `deployment.yaml`, tak aby byl správně prováděn healthcheck:

```
livenessProbe:
    httpGet:
        path: /debug/healthz
        port: http
readinessProbe:
    httpGet:
        path: /debug/ready
        port: http
```

Ve webovém prostředí Zitadel je pak potřeba nastavit alespoň následující:

1.1. Vytvořit nový projekt
1.2. V projektu vytvořit novou aplikaci
 1.2.1. Zvolit správný typ (User Agent)
 1.2.2. Zvolit metodu autentikace (PKCE - recomennded)
 1.2.3. Nastavit Redirect URI, kam se má přesměrovávat po úspěšném přihlášení a odhlášení ({FEURL}/callback)
1.3. Nastavit SMTP server v nastavení instance (případně i SMS provider)
 1.3.1. Emailová adresa
 1.3.2. Název odesilatele
 1.3.3. Host a Port
 1.3.4. Username a heslo

(Pozor při lokálním běhu byla blokována komunikace s mailovým serverem antivirovým programem)

8.5.2 Problem Generator Service

Tato služba není závislá na ostatních.
V její konfiguraci v souboru `application.properties` jsou tedy přbírány pouze proměnné prostředí udávající číslo portu (*SERVER_PORT*), zda je zapnutý tracing (*CODENOW_TRACING_ENABLED*) a endpoint tracingové služby (*CODENOW_TRACING_ZIPKIN_ENDPOINT*).

```
server.port=${SERVER_PORT:8080}
spring.main.banner-mode=off

# springdoc
springdoc.swagger-ui.enabled=true

# management endpoints
management.endpoints.web.exposure.include=health,prometheus

# management tracing
management.tracing.enabled=${CODENOW_TRACING_ENABLED:false}
management.tracing.sampling.probability=1.0
management.tracing.propagation.type=b3
management.zipkin.tracing.endpoint=${CODENOW_TRACING_ZIPKIN_ENDPOINT}
```

8.5.3 User Service

Tato služba využívá stejných proměnných ve stejném konfiguračním souboru `application.properties`, jako služba na generování příkladů, ale navíc využívá databázi a autentikační službu. Z prostředí jsou tedy předávány parametry pro připojení k databázi (*DBLOG_JDBC_URL, DBLOG_USERNAME, DBLOG_PASSWORD*) a endpoint autentikační služby (*ZITADEL_userinfoEndpoint*). V CodeNOW se tyto proměnné prostředí nastavují automaticky po správné konfiguraci v sekci Connected Services.

```
server.port=${SERVER_PORT:8080}
spring.main.banner-mode=off

# springdoc
```

```
springdoc.swagger-ui.enabled=true

# management endpoints
management.endpoints.web.exposure.include=health,prometheus

# management tracing
management.tracing.enabled=${CODENOW_TRACING_ENABLED:false}
management.tracing.sampling.probability=1.0
management.tracing.propagation.type=b3
management.zipkin.tracing.endpoint=${CODENOW_TRACING_ZIPKIN_ENDPOINT}

#database
# Database
spring.datasource.url=${DBLOG_JDBC_URL}
spring.datasource.username=${DBLOG_USERNAME}
spring.datasource.password=${DBLOG_PASSWORD}
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect
    =org.hibernate.dialect.PostgreSQLDialect
#spring.jpa.properties.hibernate.default_schema=public

userinfo.endpoint=${ZITADEL_userinfoEndpoint}
```

8.5.4 FE

Frontendová aplikace využívá všechny ostatní služby.
Potřebuje tedy v `runtime-configuration.js` nastavit URL služby generující příklady (*BE*) a URL služby zaznamenávající uživatelskou aktivitu (*UserBE*). Pro správné fungování autentikace je potřeba také nastavit URL instance Zitadel (*authority*), client ID dostupné v konfiguraci Zitadel/Application/Configuration (*client_id*), URL, kam se má přesměrovat po úspěšném přihlášení (*redirect_uri*) a odhlášení (*post_logout_redirect_uri*) a endpoint pro získání informací o uživateli (*userinfo_endpoint*). Posledním bodem konfigurace je povolení nebo zakázání běhu sledovacího nástroje Smartlook (*enableSmartlook*).

```
window.env = {
    BE: '<ProblemGeneratorServiceURL>',
    UserBE: '<UserServiceURL>',
    authority: '<ZitadelURL>',
    client_id: '<ZitadelClientId>',
    redirect_uri: '<FEURL>/callback',
    post_logout_redirect_uri: '<FEURL>/callback',
    userinfo_endpoint: '<ZitadelURL>/oidc/v1/userinfo',
    enableSmartlook: false
};
```

8.5.5 Healthcheck

CodeNOW, jak bylo již zmíněno, využíva Istio side car pro vytvoření service mesh/reverse proxy. Každá side car provádí periodicky healthcheck komponenty, ke které je napojená. Komponenty vytvořené dle některé předdefinované konfigurace od CodeNOW, například React nebo Java/Spring aplikace, není potřeba upravovat. Takto

vytvořené komponenty mají vygenerovanou kostru projektu a správnou konfiguraci, aby fungoval logging tracing a healthcheck. Pokud ale vytváříme komponentu pomocí Dockeru, je potřeba upravit konfiguraci v souboru `deployment.yaml`, tak aby prováděla healthcheck správně.

- **PostgreSQL**
  CodeNOW používá PSQL verze 12, ale autentikační služba Zitadel vyžaduje verzi 14 nebo vyšší. Proto tedy při snaze o nasazení Zitadel do CodeNOW bylo potřeba nasadit i vyšší verzi PSQL. Konfigurace healthcheck je pak následující:

```
livenessProbe:
    exec:
    command:
        - /bin/sh
        - -c
        - exec pg_isready
readinessProbe:
    exec:
    command:
        - /bin/sh
        - -c
        - exec pg_isready
```

- **Zitadel**
  Zitadel vystavuje dva endpointy, pomocí kterých lze provádět healthcheck. Konfigurace `deployment.yaml` tedy musí obsahovat následující:

```
livenessProbe:
    httpGet:
    path: /debug/healthz

readinessProbe:
    httpGet:
    path: /debug/ready
```

8.5.6 Nasazení Zitadel na CodeNOW

Zitadel potřebuje pro své fungování databázi. Preferovaná je CockroachDB, což je cloud-native distribuovaná SQL databáze. Druhou možností je využití PostgreSQL. Bohužel u obou databází nastaly při nasazování problémy.

CodeNOW nabízí CockroachDB jako managed service, tedy na první pohled vypadá řešení jednoduše. Bohužel toto řešení nebylo použitelné. CockroachDB managed service se vytvoří jako secured, tedy pro připojení k ní je potřeba mít klientský certifikát. Bohužel tento certifikát je potřeba vygenerovat a hlavně přidat mezi certifikáty běžící instance databáze. Bohužel do vytváření managed service a její konfigurace se v tomto ohledu nedá zasáhnout. Tedy takto vytvořená CockroachDB je nepoužitelná.

Další možností je tedy použití PostgreSQL. CodeNOW používá pro managed service PostgreSQL verzi 12, ale Zitadel potřebuje alespoň 14, tedy managed service nelze použít. Nabízí se tedy možnost vytvořit si instanci PostgreSQL vyšší verze pomocí Dockeru. Bohužel v prostředí CodeNOW, kde je využívána Istio proxy, nastával problém s komunikací mezi Zitadel a databází, který se nepodařilo vyřešit.
Error na úrovni proxy:

```
error envoy lua   script log:
    [string "function envoy_on_request(handle)..."]:17:
    attempt to index a nil value
```

a následný error při inicializaci Zitadel:

```
level=fatal msg="unable to initialize the database"
caller="/home/runner/work/zitadel/zitadel/cmd/initialise/init.go:64"
error="ID=DATAB-OpIWD Message=Errors.Database.Connection.Failed
Parent=(failed to connect to `host=postgresql user=zitadel
    database=postgres`: failed to receive message (unexpected EOF))"
```

Poslední možností pro nasazení byl oficiální Helm Chart Zitadel, který obsahuje jak Zitadel, tak kompatibilní verzi CockroachDB. Tato možnost je na první pohled velmi jednoduchá, avšak byla nakonec také neúspěšná. Zmiňovaný Helm Chart obsahuje mnoho konfigurací, ke kterým ale bohužel ani nedošlo. CodeNOW nabízí možnost vytvoření komponenty pomocí Helm Chartu, ale při pokusu o nasazení Zitadel Chartu se hned první Cron job neprovedl správně. Undeploy chartu a smazání rozbitého prostředí následně nešlo provést a prostředí je od té doby ve stavu `Deleting`.

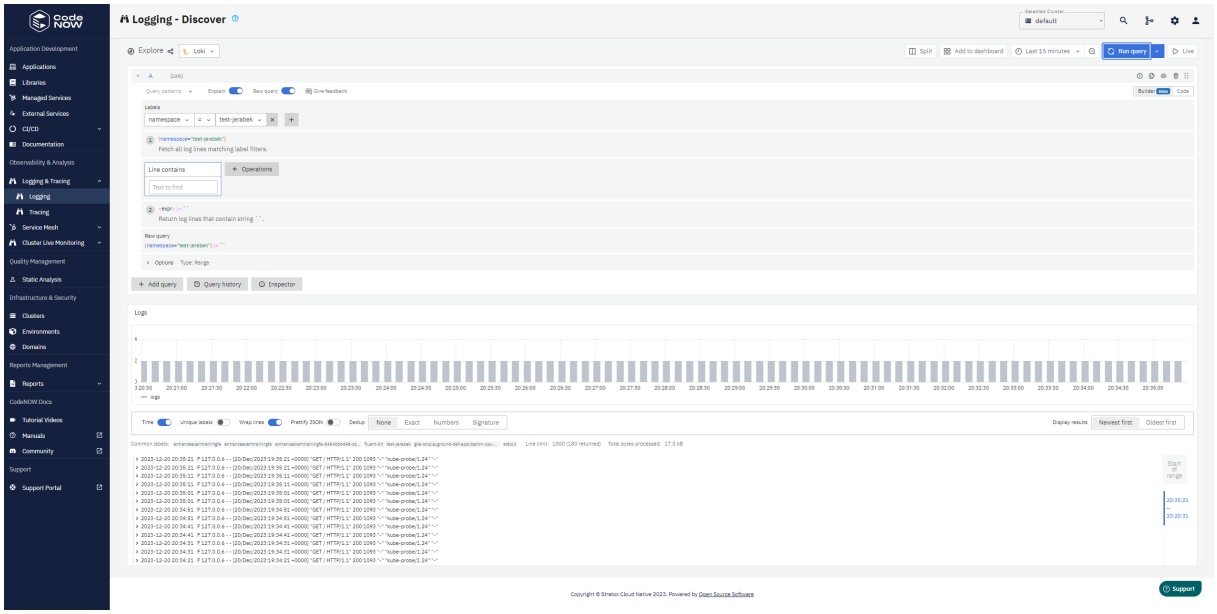
Po těchto neúspěších byla zvolena možnost využití Zitadel SaaS, která je plně funkční a automaticky škálovaná. Jediné její omezení je limitovaný počet požadavků (25 000 za měsíc) v neplacené verzi. Výše zmíněné problémy s platformou CodeNOW byly předány k prozkoumání.

8.6 Logování a Tracing

CodeNOW nabízí možnosti logování a tracingu pomocí nástrojů Grafana Loki [20] a Jaeger Tracing [21].

8.6.1 Logování

Logování slouží k zaznamenávání důležitých událostí, stavů a chyb, které se vyskytují během provozu aplikace. V distribuovaných aplikacích je shromažďování logů složitější než v klasických monolitických, protože máme více běžících aplikací, které můžou používat různý technologický stack. Proto je typické, že jednotlivé mikroslužby logují na svůj standardní výstup a v clusteru běží služba, která se stará o sbírání těchto logů ze standardních výstupů a jejich shromažďování. Pro sběr, agregaci a dotazování nad logy CodeNOW používá nástroj Loki od Grafana Labs díky kterému lze pak ve webovém prostředí logy filtrovat a prohlížet.



Obrázek 8.12. CodeNOW - Logování

8.6.2 Tracing

Tracing neboli trasování se využívá ke sledování execution flow v distribuovaných systémech nebo v komplexních aplikacích. Zachycuje informace o jednotlivých požadavcích a transakcích, jak se propagují skrz různé komponenty a služby. [22] CodeNOW využívá nástroj Jaeger, který slouží přesně k těmto účelům a v rámci svého webového prostředí nabízí možnosti prohlížení zaznamenaného tracingu. Aplikace neobsahuje komplexní distribuované operace, tedy v současném stavu není tracing příliš zajímavý z pohledu aplikace není využit. Avšak i tak trasování poskytuje alespoň informaci o propagování požadavků v rámci běhového prostředí.

8.7 Monitoring využití hardware

CodeNOW využívá nástroj Grafana k monitorování využití hardwarových prostředků běžícími aplikacemi. Ve webovém prostředí nástroje lze sledovat celý cluster a nebo různě filtrovat na základě komponent, běhových prostředí nebo aplikací. Sledované je využití CPU, paměti a sítě. Monitoring byl využit zejména při zátěžových testech v kapitole 9.3

Kapitola 9
Testování

9.1 Jednotkové testy

Jednotkové testy se zaměřují na jednotlivé součásti softwarového systému. Cílem těchto testů je ověřit, že každá dílčí komponenta, neboli jednotka, funguje dle očekávání a splňuje požadavky. Jednotkové testy typicky vytváří vývojáři a jsou prováděny již v brzkých fázích vývoje dříve, než integrační či end-to-end testy. Jednotkové testy by měly být automatizované a spouštěné při každé změně kódu, aby bylo zajištěno, že nový kód neporušuje stávající funkčnost. Testy se provádí v izolovaném prostředí, tak aby ověřovaly co nejmenší část kódu. Díky tomu lze rychle identifikovat a opravit případné chyby. [23]

9.1.1 JUnit

JUnit je framework pro psaní jednotkových testů vytvořený v programovacím jazyce Java. [24] Díky tomu se hodí pro testování programů psaných v Javě a je snadné ho zaintegrovat do procesu sestavení programu a CI/CD pipeline.

Pomocí JUnit testů byla v mikroslužbě User Service testována vrstva ActivityLogService, která obsahuje logiku na agregaci záznamů nalezených dle předaných filtrů. Pro tyto testy byl využit framework Mockito, který umožňuje vytvářet `falešné` objekty, tak aby například při zavolání metody vracely předem daný výsledek. Díky tomu může být třída testována izolovaně, nezávisle na fungování ostatních komponent. Tato mikroslužba je poměrně malá a neobsahuje žádnou další implementovanou logiku, kterou by mělo smysl testovat.

Ve službě generující příklady bylo vytvořeno několik jednotkových testů, avšak protože jsou příklady generované náhodně a je současně generované zadání i řešení, není triviální způsob jak jednotkově testovat. Většina příkladů obsahuje slovní zadání, což je v rámci unit testu těžko zpracovatelné. Je ale možné vybírat z textu zadaná čísla a dosadit je do vztahu nebo rovnice, který předpokládáme, že je řešením. Na základě toho můžeme porovnat spočtený výsledek s výsledkem vygenerovaným. Příklady početní bez slovního zadání se testují snadněji, stačí pouze vyřešit daný příklad a porovnat řešení s referenčním.

9.2 Automatizované testy uživatelského rozhraní

Automatizované testy uživatelského rozhraní se vytváří za pomocí různých nástrojů a jejich hlavním úkolem je zjišťovat, zda se aplikace chová dle očekávání a splňuje zadané požadavky. Jde o uživatelské testy, tedy simulace uživatelových interakcí s aplikací, jako navigace ve stránce, klikání na tlačítka a zadávání vstupů. Výhodou automatizovaných uživatelských testů je jejich rychlost a snadná opakovatelnost ve srovnání s živým uživatelem. Další výhodou je zamezení vstupu lidské chyby do procesu testování, avšak to může být občas i nevýhoda, jelikož vývojáři a testeři

často nevymyslí tak podivné scénáře, jako koncoví uživatelé. Stejně jako Unit testy můžeme automatizované testy přidat do CI/CD pipeliny, čímž zajistíme, že i po každé změně kódu se uživatelské rozhraní aplikace bude chovat správně. Všechny UI testy probíhaly s využitím prohlížeče Google Chrome (Verze 120.0.6099.130) a testováno bylo uživatelské rozhraní nasazené na CodeNOW.

9.2.1 Selenium

Selenium je nástroj pro automatizované testování webových aplikací, který podporuje mnoho programovacích jazyků. Umožňuje vytvářet testy, které simulují interakci uživatele s webovým prohlížečem.

9.2.2 Testované scénáře

- **Přihlášení, Uživatelský profil, Odhlášení**
  1. Otevřít webovou stránku aplikace
  2. Kliknout na `Přihlásit se`
  3. Kontrola přesměrování na očekávanou URL
  4. Vyplnit email a kliknout `Další`
  5. Vyplnit heslo a kliknout `Další`
  6. Kontrola přesměrování zpět do aplikace
  7. Kontrola zobrazení username místo tlačítka `Přihlásit`
  8. Kontrola zobrazení uživatelského profilu a tabulky s daty
  9. Odhlásit uživatele kliknutím na `Odhlásit`
  10. Kontrola zobrazení tlačítka `Přihlásit` místo username
  11. Kontrola přesměrování na homepage
- **Přístup na stránku profilu bez přihlášení**
  1. Otevřít webovou stránku aplikace
  2. Přidat k URL `/profile`
  3. Kontrola přesměrování do Zitadel Login
- **Řešení příkladů**
  1. Otevřít webovou stránku aplikace
  2. Kliknout na `4 leté`
  3. Kliknout na `Číselné výrazy, zlomky, desetinná čísla, mocniny`
  4. Kliknout na `Zkontrolovat`
  5. Kontrola označení inputfield červeně
  6. Kontrola zprávy o správnosti řešení
  7. Kontrola disablování tlačítka `Zkontrolovat`
  8. Kliknout na `Vygenerovat`
  9. Kontrola skrytí červeného ohraničení inputfield
  10. Kontrola enablování tlačítka `Zkontrolovat`
- **Navigace a historie**
  1. Otevřít webovou stránku aplikace
  2. Kliknout na `4 leté`
  3. Kliknout na `Číselné výrazy, zlomky, desetinná čísla, mocniny`
  4. Kliknout na `8 leté`
  5. Kliknout na `Převody jednotek`
  6. Kliknout na zpět
  7. Kontrola přesunu na `/8`
  8. Kliknout na zpět

9. Kontrola přesunu na `/4/fractions`
10. Kliknout na zpět 2x
11. Kontrola přesunu na `/`
12. Kliknout na vpřed
13. Kontrola přesunu na `/4`

9.3 Zátěžové testy

Stress testy obecně slouží k ověření toho, jak se systém chová pod zátěží. Cílem testů může být ověření výkonnosti, identifikace limitů a slabých míst, ověření stability a plánování kapacity. V případě aplikace vyvíjené v rámci této práce byla ověřováno zejména výkonnost a kapacita. Aplikace sice běží na platformě CodeNOW, ale z důvodu běhu na sdíleném prostředí s omezenými prostředky není automaticky škálována při zvětšené zátěži. Testy tedy měly za úkol odhalit, kolik je aplikace schopná obsloužit uživatelů v jednu chvíli a jak se zachová při nadměrné zátěži.

9.3.1 K6

Pro zátěžové testy jsem zvolil nástroj K6, což je open-source nástroj na provádění zátěžových testů od Grafana Labs. Díky psaní testů v Javascriptu je snadné vytvářet jednoduché testy API nebo komplexnější end-to-end testy [25]

9.3.2 API zátěžové testy

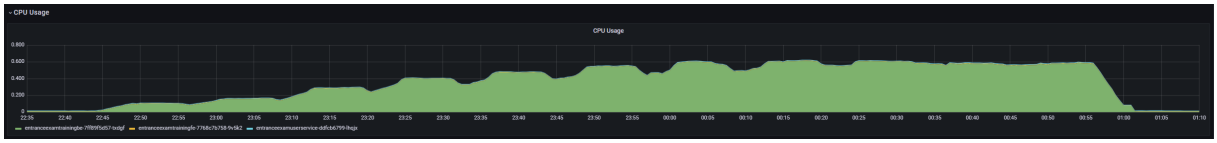
Testy probíhaly s postupně se zvyšující zátěží od 30 uživatelů až po 3000. Každý virtuální uživatel odesílal každou sekundu jeden http požadavek na náhodný endpoint API generujícího příklady. Testy byly nastavené tak, aby se počet uživatelů zvyšoval od jednoho až po dané maximum během první minuty, pak deset minut udržoval maximální zátěž a na konci zase během jedné minuty zátěž postupně snižoval. Toto chování je zřejmé i z obrázku 9.1, který znázorňuje využití procesoru služby poskytující testované API. V tabulce 9.1 jsou zaznamenány měřené hodnoty trvání dotazu. Minimální hodnota není uváděna, protože každý test začínal od jednoho uživatele, tedy tato hodnota byla vždy víceméně stejná okolo 20 ms.

počet VUs	avg	max	p(95)	p(99.9)	failed request
30	32.11 ms	535.54 ms	52.57 ms	429.7 ms	0 %
50	58.16 ms	1.29 s	149.38 ms	666.55 ms	0 %
100	44.33 ms	728.53 ms	105.8 ms	455.05	0 %
150	95.26 ms	4.87 s	309.92 ms	4.06 s	0 %
200	132.28 ms	2.36 s	486.59 ms	1.69 s	0 %
250	226.84 ms	5.6 s	706.69 ms	4.54 s	0 %
350	572.07 ms	5.88 s	1.4 s	5.16 s	0 %
500	1.05 s	5.2 s	2.05 s	3.52 s	0 %
1000	3.06 s	19.99 s	4.68 s	16.05 s	0 %
2000	5.98 s	13.21 s	9.2 s	12.62 s	6.43 %
3000	5.01 s	14.54 s	8.75 s	13.85 s	23.46 %

Tabulka 9.1. Výsledky zátěžových testů

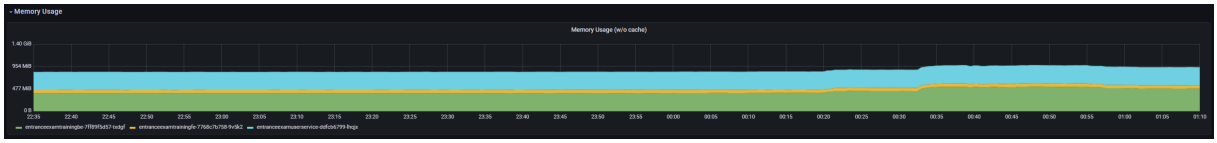
Platforma CodeNOW umožňuje sledovat využití hardwarových prostředků díky integrovanému monitoringu a tyto hodnoty lze sledovat v aplikaci Grafana. Sledované byly hodnoty vytížení procesoru, paměti a provoz na síti.

Na obrázku níže můžeme vidět, že během provádění zátěžových testů se vytížení procesorových jednotek dostalo až na 60%.



Obrázek 9.1. API zátěžový test - využití CPU

Využití paměti se během testu v zásadě neměnilo, to je dáno i tím, že aplikace nepracuje s žádnými velkými objekty, vždy pouze vygeneruje daný příklad a pošle ho v těle odpovědi.



Obrázek 9.2. API load-test - využití RAM

V následujícím snímku můžeme pozorovat, jak se zvýšil provoz na síti posílanými dotazy. Přijímané dotazy jsou typu GET a neobsahují tedy žádná data uvnitř těla dotazu, proto je množství přijímaných dat menší než odesílaných. Posílané objekty příkladů v těle odpovědi jsou serializovány jako JSON objekt a nejsou příliš velké, ale i přesto můžeme pozorovat větší objem v odesílaných datech.

Obrázek 9.3. API load-test - šířka pásma sítě

9.3.3 Shrnutí zátěžových testů

Přestože aplikace je nasazena v cloudovém prostředí, neškáluje automaticky pod zátěží. Prostředí, ve kterém je nasazena, je sdílené více aplikacemi a má omezené hardwarové prostředky. Množství instancí (Kubernetes Pods) je možné zvětšovat pouze manuálně v konfiguraci jednotlivých služeb. Kvůli těmto omezením má aplikace limit zhruba u 1000 uživatelů v jeden okamžik. Při takové zátěži aplikace zvládá obsloužit všechny dotazy, ale trvání většiny dotazů překračuje uživatelsky přijatelnou dobu. Přijatelná doba odezvy se pohybuje mezi 200 ms a 1 s [26]. Při počtu 250 uživatelů se do tohoto intervalu vejdeme s 95 % pravděpodobností.

9.4 Uživatelské testy

Uživatelské testy slouží k hodnocení, jak uživatelé interagují s aplikací, a k získání zpětné vazby od uživatelů. Protože je aplikace dostupná na internetu, není potřeba uživatele dostávat ke speciálnímu zařízením s běžící aplikací, ale můžou si ji otevřít na svých zařízeních. Pro zajištění monitorování uživatelských testů byla zvolen nástroj Smartlook, který lze snadno zaintegrovat do webové aplikace.

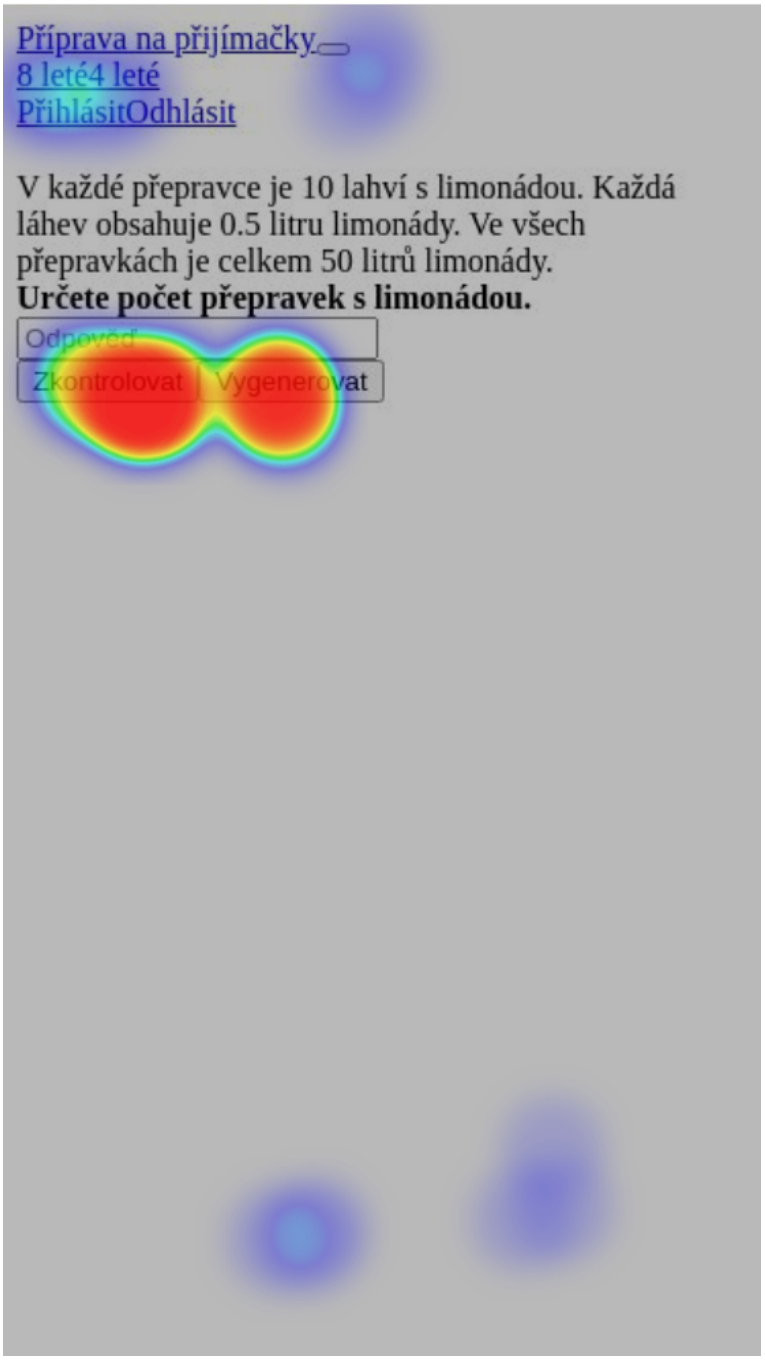
9.4.1 Smartlook

Smartlook je nástroj od společnosti Cisco pro sledování uživatelského chování v aplikaci. Smartlook zaznamenává uživatelovu aktivitu na stránce, tedy pohyb kurzoru, klikání nebo vyplňování formulářových polí a jednotlivé události. K tomu pořizuje záznam obrazovky a k němu záznam jednotlivých aktivit a změny v URL. Vyplňované hodnoty ve formulářích jsou automaticky skryty a nahrazeny symbolem $*$. Dále jsou zaznamenávány údaje o typu zařízení, rozlišení obrazovky, operačním systému, prohlížeči a přibližné lokalitě na úrovni státu, dle IP adresy. Smartlook částečně vyhodnocuje chování uživatele sám, například detekuje tzv. rage click, což je přehnané klikání do jednoho místa typicky způsobené frustrací uživatele z důvodu nefunkčnosti nebo zaseknutí aplikace, nebo vytváří diagramy popisující průběh chování (behavior flow), neboli posloupnost průchodu aplikací. Dále je možné definovat vlastní události, které se mají zaznamenávat. Na základě těchto událostí lze pak vytvořit tzv. funnels, neboli trychtýře. Ty mají specifickou sekvenci kroků, která vede k nějakému specifickému cíli a my sledujeme, kolik uživatelů dokončilo celou sekvenci a kolik skončilo v nějakém mezikroku. Díky tomu získáváme konverzní poměr, kolik z uživatelů dokončí danou sekvenci.

V placené verzi nástroje je pak mnoho dalších možných konfigurací a možností jak sledovat uživatele. Neplacená verze je kromě omezených možností limitovaná počtem uživatelských relací na 3000 za měsíc, 10 sledovaných eventů, 2 funnels a 10 heatmap. [27]

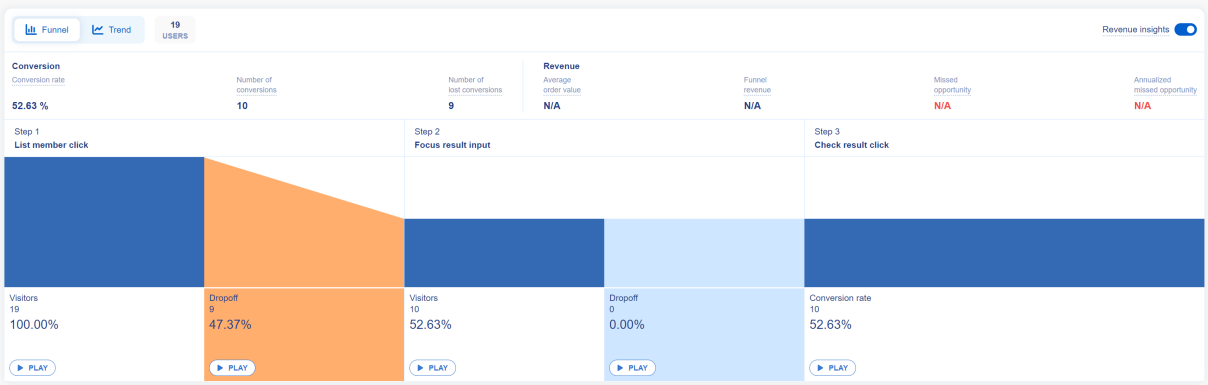
Nástroj Smartlook bohužel není možné použít pro sledování chování uživatele v přihlašovacím formuláři nebo při registraci. Přihlašovací a registrační formulář je součástí služby Zitadel, což je aplikace třetí strany a tedy není možné zasahovat do kódu. Zároveň při registraci musí uživatel aktivovat svůj účet kliknutím na odkaz nebo opsáním kódu z emailu, kde chování také nelze pozorovat. Protože tyto kroky nejsou prováděny v implementované aplikaci, nepřineslo by jejich sledování příliš užitku, protože je není možné měnit.

Po vypuštění aplikace mezi první uživatele bylo možné pozorovat konkrétní uživatelské relace a případné chyby nastávající v nich. Dále bylo sesbíráno dostatečné množství dat pro vytvoření heatmapy (obrázek 9.4), která nepřekvapivě ukazuje, že uživatelé nejčastěji u příkladů klikají na tlačítka Zkontrolovat a Vygenerovat. Bohužel se v heatmapě ztrácí design webové aplikace.



Obrázek 9.4. Smartlook - Heatmap na obrazovce příkladu

Vytvořený funnel zase ukazuje poměr uživatelů, kteří po otevření příkladu zkusili příklad vyřešit, tedy zapsat něco do vstupního pole a zkontrolovat výsledek. Na obrázku 9.5 můžeme vidět, že téměř polovina uživatelů po otevření příkladu stránku opustila. Naopak ti, co zůstali a pokusili se něco zapsat, všichni provedli kontrolu výsledku.



Obrázek 9.5. Smartlook - Funnel vyřešení otevřeného příkladu

9.4.2 Sledované scénáře

Jelikož nebyly prováděny uživatelské testy přímo, nebyly vytvořeny konkrétní specifické scénáře. Částečně byly prováděny uživatelské testy během vývoje, tedy zkoušení funkcionalit uživatelského rozhraní. Sledování pak sloužilo k odhalení neočekávaných problémů, drobných designových nedostatků a zároveň k celkovému hodnocení uživatelské přívětivosti.

Kapitola 10
Závěr

Aplikace vytvořená v rámci této práce představuje koncept přípravy na jednotnou přijímací zkoušku pomocí webové aplikace dostupné bezplatně. Oproti ostatním aplikacím nabízí dynamicky generované typové příklady přijímacích zkoušek, umožňující procvičovat typové úlohy s různými číselnými i slovními zadáními.

Pro vytvoření byla využita moderní architektura mikroslužeb, která umožňuje snadnější rozšiřitelnost o nové mikroslužby, práci více nezávislých lidí nebo týmů, možnost využití různých komponent a technologií a je úzce spojená s konceptem cloud-native. Aplikace byla nasazena do provozu na Value Stream Delivery Platformě CodeNOW, kde je dostupná přes internet.

V rámci této práce byla prozkoumána existující volně dostupná řešení pro přípravu na jednotné přijímací zkoušky a procvičování matematiky a porovnány jejich výhody a nevýhody. Přes úvodní zájem o téma ze strany pana Krejčího, ředitele Centra pro zjišťování výsledků vzdělávání, které přijímací zkoušky tvoří a zaštiťuje, nebyla spolupráce nijak prohloubena přes námi poskytnutou nabídku. Na naší schůze v květnu jsme se shodli na tom, že podobná aplikace pro procvičování chybí, ale nakonec o naši aplikaci zájem ze strany Centra nebyl. Místo toho Cermat představil na podzim roku 2023 novou aplikaci Tau, která přináší již dříve dostupný obsah testů z minulých let v novém kabátu a s možností si ihned kontrolovat odpovědi. Kolem této aplikace se také v médiích objevila kauza ohledně vytvoření aplikace synem ředitele Cermatu, pana Krejčího, v rámci studentského projektu a finanční odměny za tento produkt [28]. Přesto aplikace vytvářená v rámci této práce má smysl a přináší inovaci svým generováním parametrizovaných úloh, díky kterému se neopakují zadání a student se nemůže naučit výsledky nazpaměť, ale musí si osvojit dovednosti k vyřešení úloh.

V rámci analýzy byly prostudovány požadavky na dovednosti a znalosti uchazečů o studim, skládajících přijímací zkoušky, a následně parametrizovány typové úlohy s potřebnými číselnými omezeními, pro následnou implementaci. Řešení bylo rozděleno na jednotlivé komponenty, na základě mikroservisní architektury, tak aby každá mikroslužba odpovídala za uzavřenou část funkcionality a pro každou z nich byla zvolena vhodná struktura a technologie. Objekty, se kterými aplikace pracuje, byly navrženy tak, aby byly co nejvíce obecné a znovupoužitelné, což usnadňuje přidávání dalších typových úloh.

Pro implementaci byly zvoleny dobře známé a rozšířené technologie, vhodné pro vývoj webové aplikace. Díky tomu by mělo být snadné na vývoj navázat a aplikaci dále rozšiřovat. Vyvíjená aplikace byla průběžně nasazována na platformu CodeNOW.

Vyvinutá aplikace byla průběžně testována jednotkovými testy, automatizovnými testy uživatelského rozhraní, později pak proběhly zátěžové testy na instanci běžící na platformě CodeNOW a v produkci pak byl použit nástroj Smartlook pro sledování uživatelského chování.

10.1 Další vývoj

V dalším vývoji by aplikace mohla být rozšířena o další typy příkladů, zejména konstrukční úlohy, které vyžadují naprosto odlišný přístup a mohla by pro ně vzniknout samostatná mikroslužba.

Dále by se dal vylepšit způsob jednotkového testování pro náhodně generované příklady se slovním zadáním, pravděpodobně za využití technologií zpracování textu, nebo pro konstrukční úlohy.

Příklady by se také daly obohatit o nápovědu nebo vysvětlení řešení a to za použití například WolframAlpha API [1] nebo některého AI chatbota poskytujícího API, například ChatGPT [2] od OpenAI. Zde je však potřeba mít na paměti, že poskytnuté výsledky nemusí být vždy správné.

Přestože je aplikace responzivní a použitelná i na mobilních zařízeních, mohla by být vytvořena nativní mobilní aplikace, která by vylepšila uživatelskou zkušenost na mobilních zařízení a umožňovala alespoň částečně procvičovat příklady i bez připojení k internetu díky stažení příkladů do paměti telefonu. Aplikace by mohla uživatele upozorňovat na neaktivitu, motivovat ho k procvičování formou sbírání odměn a doporučovat úlohy, které uživatel vyřešil špatně, tak aby se v jejich řešení zlepšoval.

Další vývoj by mohl být velkým přínosem pro to, aby se aplikace skutečně začala využívat na maximum a postupně by mohla být rozšířena i o sekce pro přípravu na přijímací zkoušku z českého jazyka nebo přípravu na maturitní zkoušky a mohla by být i překládána do více jazyků a sloužit i jako podpora při studiu a procvičování matematiky pro žáky, studenty i širokou veřejnost.

[1] `https://products.wolframalpha.com/api`
[2] `https://platform.openai.com/docs/introduction`

Literatura

[1] *Tabulka s počty podaných přihlášek k JPZ, setříděné podle škol i oborů, a to za období 2017–2023* [online]. [vid. 2023-05-24]. Dostupné na `https://prijimacky.cermat.cz/files/files/JPZ2017-2023_PRIHLASENI_osoby_prihlasky.xlsx`.

[2] *KSAOS - Step by step guide to understanding KSAOs* [online]. [vid. 2023-06--01]. Dostupné na `https://www.thehumancapitalhub.com/articles/ksaos-step-by-step-guide-to-understanding-ksaos`.

[3] *CZVV, Testová zadání v PDF* [online]. [vid. 2023-05-24]. Dostupné na `https://prijimacky.cermat.cz/menu/testova-zadani-k-procvicovani/testova-zadani-v-pdf`.

[4] *Webová aplikace CZVV pro procvičování úloh* [online]. [vid. 2023-05-24]. Dostupné na `https://procvicprijimacky.cermat.cz/`.

[5] *Webová aplikace Umíme To* [online]. [vid. 2023-05-24]. Dostupné na `https://www.umimeto.org/`.

[6] *Trénuj a uč se* [online]. [vid. 2023-12-09]. Dostupné na `https://tau.cermat.cz/`.

[7] *Specifikace požadavků pro jednotnou přijímací zkoušku v přijímacím řízení na střední školy v oborech vzdělání s maturitní zkouškou* [online]. [vid. 2023-05-24]. Dostupné na `https://prijimacky.cermat.cz/files/files/dokumenty/specifikace-pozadavku/Specifikace_2022-2023/MASPECIFIKACEPOZADAVKU2022.pdf`.

[8] *React* [online]. [vid. 2023-05-24]. Dostupné na `https://react.dev/`.

[9] *Top 10 JavaScript Frameworks in 2023* [online]. [vid. 2023-11-15]. Dostupné na `https://hackernoon.com/the-10-best-javascript-frameworks-to-use-in-2023`.

[10] *Top JavaScript Frameworks and Technology 2023* [online]. [vid. 2023-12-09]. Dostupné na `https://medium.com/javascript-scene/top-javascript-frameworks-and-technology-2023-4e4a06d6be93`.

[11] *Bootstrap for React* [online]. [vid. 2023-05-24]. Dostupné na `https://react-bootstrap.github.io/`.

[12] *Java* [online]. [vid. 2023-05-24]. Dostupné na `https://www.java.com/en/`.

[13] *Spring, Spring Boot* [online]. [vid. 2023-05-24]. Dostupné na `https://spring.io/`.

[14] *Swagger* [online]. [vid. 2023-12-30]. Dostupné na `https://swagger.io/`.

[15] *PostgreSQL* [online]. [vid. 2023-05-24]. Dostupné na `https://www.postgresql.org/`.

[16] *Zitadel* [online]. [vid. 2023-05-24]. Dostupné na `https://zitadel.com/`.

[17] *Cloud Native Computing Foundation* [online]. [vid. 2023-11-15]. Dostupné na https://www.cncf.io/.

[18] *CodeNOW* [online]. [vid. 2023-05-24]. Dostupné na https://www.codenow.com/.

[19] *Kubernetes* [online]. [vid. 2024-01-02]. Dostupné na https://kubernetes.io/.

[20] *Grafana Loki* [online]. [vid. 2023-12-20]. Dostupné na https://grafana.com/oss/loki/.

[21] *Jaeger tracing* [online]. [vid. 2023-05-24]. Dostupné na https://www.jaegertracing.io/.

[22] *Difference between logging and tracing* [online]. [vid. 2024-01-02]. Dostupné na https://www.linkedin.com/pulse/difference-between-logging-tracing-antonio-martin/.

[23] *Geeks for Geeks - Unit Testing – Software Testing* [online]. [vid. 2023-12-29]. Dostupné na https://www.geeksforgeeks.org/unit-testing-software-testing/.

[24] *JUnit 5* [online]. [vid. 2023-12-29]. Dostupné na https://junit.org/junit5/.

[25] *K6* [online]. [vid. 2023-12-09]. Dostupné na https://k6.io/.

[26] *Sematext - Glossary - Response Time* [online]. [vid. 2023-12-28]. Dostupné na https://sematext.com/glossary/response-time/.

[27] *Smartlook* [online]. [vid. 2023-12-21]. Dostupné na https://www.smartlook.com/.

[28] *Deník N - Aplikaci Cermatu na procvičování přijímaček vytvořil ředitelův syn, má to jako studentský projekt* [online]. [vid. 2024-01-06]. Dostupné na https://denikn.cz/minuta/1295437/?ref=pop&rtm_source=web&rtm_medium=article&rtm_campaign=share_button&rtm_variant=native&rtm_content=05d663bf-c08f-4059-bbbe-ddf1c59ffb63.

[29] Chris, Richardson. *Microservices Patterns with examples in Java*. Manning, 2019. ISBN 978-1-61729-454-9.

[30] *Debezium* [online]. [vid. 2023-05-24]. Dostupné na https://debezium.io/.

[31] *KrakenD* [online]. [vid. 2023-05-24]. Dostupné na https://www.krakend.io/.

Příloha A
Žádost o využití zadání přijímacích zkoušek v rámci diplomové práce

cermat
Centrum pro zjišťování výsledků vzdělávání
Jankovcova 933/63, 170 00 Praha 7
info@cermat.cz, tel.: +420 224 507 507
www.cermat.cz

Bc. Petr Jeřábek
e-mail: jerabpe5@fel.cvut.cz

Praha 21. 4. 2023
Č. j.: C2154/B/2023/Ř/2

Žádost o využití zadání přijímacích zkoušek v rámci diplomové práce

Vážený pane Jeřábku,

dne 12. 4. 2023 byla Centru pro zjišťování výsledků vzdělávání (dále „Centrum") doručena Vaše žádost o využití testových zadání přijímacích zkoušek z minulých let, dostupných na webovém portálu Centra, v rámci diplomové práce při studiu Českého vysokého učení technického v Praze, oboru Softwarového inženýrství – vytvoření aplikace pro přípravu žáků na přijímací zkoušky, pro 4 leté i 8 leté obory.

Centrum žádosti vyhovuje a sděluje:

Žádáme Vás o dodržení podmínky, kdy pro stažení a použití dat platí, že obsah těchto materiálů je chráněn autorskými právy a jakékoli další užití testových zadání jednotné přijímací zkoušky, než je výše zmiňovaný účel, je možné pouze s dalším písemným souhlasem ředitele Centra.

S pozdravem

Ing. Miroslav Krejčí
ředitel Centra