Výuková aplikace pro automatické dokazování ve výrokové logice

Bakalářská práce

Vedoucí práce:

Ing. Pavel Haluza, Ph.D. Dominik Farlík

Poděkování

Zde je možné psát text poděkování. Pokud chcete poděkování vynechat, smažte tento text a u dvoustranného dokumentu i následující stránku.

Čestné prohlášení

Prohlašuji, že jsem práci Výuková aplikace pro automatické dokazování ve výrokové logice vypracoval/a samostatně a veškeré použité prameny a informace uvádím v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů a v sou­ladu s platnou Směrnicí o zveřejňování závěrečných prací.

Jsem si vědom/a, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně dne 12. května 2025

Abstract

Farlík, D. Educational Application for Automated Theorem Proving in Propositional Logic.  
Bachelor Thesis. Brno: Mendel University, 2025.

This bachelor's thesis focuses on the design and implementation of an educational application aimed at using the resolution method for proving the truth of propositional formulas. The application will help users better understand the process of formula decomposition through the visualization of individual steps.   
The theoretical part explains the fundamental concepts related to propositional logic and the use of the resolution method. It also includes an analysis of the application's requirements and currently available alternatives.   
The practical part is dedicated to the application's design and the selection of technologies used for its development. Additionally, it covers the actual implementation and user testing.

Keywords

Propositional logic, resolution method, web application, Flask, React, Bootstrap, MySQL

Abstrakt

Farlík, D. Výuková aplikace pro automatické dokazování ve výrokové logice.

Bakalářská práce. Brno: Mendelova univerzita, 2025.

Bakalářská práce je zaměřena na návrh a implementaci výukové aplikace zaměřené na používání rezoluční metody pro dokazování pravdivosti výrokových formulí. Aplikace bude sloužit pro lepší pochopení procesu rozkladu formulí díky vizualizaci jednotlivých kroků postupu.

V teoretické části jsou vysvětleny základní pojmy problematiky okolo výrokové logiky a použití rezoluční metody. Součástí je i analýza požadavků aplikace a aktuálně dostupné alternativy.

Praktická část se věnuje návrhu aplikace a výběru použitých technologií pro vytvoření. Dále se v ní také vyskytuje samotná implementace a uživatelské testování.

Klíčová slova

Výroková logika, rezoluční metoda, webová aplikace, Flask, React, Bootstrap, MySQL

Obsah

Obsah 8

Seznam obrázků 10

Seznam tabulek 11

1 Úvod a cíl práce 13

1.1 Úvod 13

1.2 Cíl práce 13

2 Analýza 14

2.1 Aktuální stav 14

2.2 Požadavky aplikace 14

2.2.1 Funkční požadavky 14

2.2.2 Nefunkční požadavky 14

2.3 Podobné aplikace 15

3 Automatické dokazování 16

3.1 Pravdivostní tabulka 16

3.2 Rezoluční metoda 16

3.2.1 Důkaz sporem 17

3.2.2 Postup použití rezoluční metody 18

4 Návrh aplikace a použité technologie 20

4.1 Typ aplikace 20

4.2 Backend 20

4.2.1 Python 20

4.2.2 Flask 20

4.2.3 API 21

4.3 Frontend 21

4.3.1 Javascript 21

4.3.2 React 22

4.3.3 Bootstrap 22

4.4 Databáze 22

4.4.1 MySQL 22

4.5 Bezpečnost 22

4.5.1 API Komunikace 23

4.5.2 Komunikace uživatele se serverem 24

5 Implementace 25

6 Diskuse a zhodnocení 26

6.1 Přínos aplikace 26

6.2 Omezení a možná vylepšení 26

6.3 Možnosti dalšího rozvoje 26

7 Závěr 27

7.1 Shrnutí výsledků 27

7.2 Možnost praktického využití 27

8 Literatura 28

8.1 Seznam bibliografických citací 28

Seznam obrázků

No table of contents entries found.

Seznam tabulek

No table of contents entries found.

1. Úvod a cíl práce
   1. Úvod
   2. Cíl práce
2. Analýza
   1. Aktuální stav

Hlavním důvodem pro vznik této aplikace byla skutečnost, že dosud neexistuje žádný nástroj, který by splňoval požadované parametry a zároveň nabízel potřebnou funkcionalitu. Při analýze dostupných řešení bylo zjištěno, že žádná ze současných aplikací nenabízí ucelený a srozumitelný přístup k řešení výrokových formulí pomocí rezoluční metody. Chybí nástroj, který by nejen dokázal určit, zda je daná množina splnitelná, ale zároveň by uživateli krok za krokem ukázal, jakým způsobem k výsledku došel. Právě tato kombinace řešení a edukativní složky je klíčovým prvkem této aplikace, která má za cíl usnadnit pochopení rezoluční metody.

* 1. Požadavky aplikace

Tato sekce se zaměřuje na klíčové požadavky, které musí aplikace splňovat, aby byla plně funkční a přínosná pro koncové uživatele. Požadavky jsou rozděleny do dvou hlavních kategorií – funkční a nefunkční.

Funkční požadavky popisují, jaké konkrétní operace bude možné v aplikaci provádět a jakým způsobem bude uživatel s aplikací interagovat. Nefunkční požadavky se pak týkají technických a systémových vlastností, jako je například výkon, spolehlivost, přehlednost rozhraní nebo možnosti dalšího rozšiřování aplikace.

* + 1. Funkční požadavky

Aplikace nabídne přehledné a intuitivní webové rozhraní, které umožní uživatelům zadávat logické formule ve snadno čitelné podobě. Bude obsahovat integrovaný editor s podporou infixové notace, možností automatického převodu do konjunktivní normální formy (KNF).

Uživatelé budou moci vybírat z předdefinovaných příkladů a sledovat průběh řešení krok za krokem. Implementovaná rezoluční metoda zajistí rozhodování o splnitelnosti množiny formulí, přičemž každý rezoluční krok bude zaznamenán a přehledně vizualizován. Výstup bude prezentován formou stromového diagramu.

Bude navržena s důrazem na kompatibilitu s běžně používanými webovými prohlížeči a připravena na budoucí rozšiřování funkcionality.

* + 1. Nefunkční požadavky

Aplikace by měla být optimalizována pro efektivní zpracování i většího množství vstupních dat, aby mohla rychle a plynule zobrazovat jednotlivé kroky řešení. Zvláštní důraz bude kladen na přehledné a uživatelsky přívětivé rozhraní, které usnadní práci jak začátečníkům, tak pokročilým uživatelům.

Z architektonického hlediska bude aplikace navržena modulárně, což umožní její snadné rozšiřování, například o další metody dokazování či nové formáty vstupních a výstupních dat.

Celkový návrh bude klást důraz na udržitelnost, rozšiřitelnost a dlouhodobou použitelnost systému.

* 1. Podobné aplikace

Existují různé aplikace, které dokážou řešit splnitelnost logických formulí, dokonce i s možností výběru metody, která bude použita pro řešení, ale zatím neexistuje žádná, která by nabízela i vizualizaci postupu řešení.

<https://www.erpelstolz.at/gateway/formular-uk-zentral.html>

<https://www.emathhelp.net/calculators/discrete-mathematics/truth-table-calculator/>

<https://atozmath.com/MathLogic.aspx>

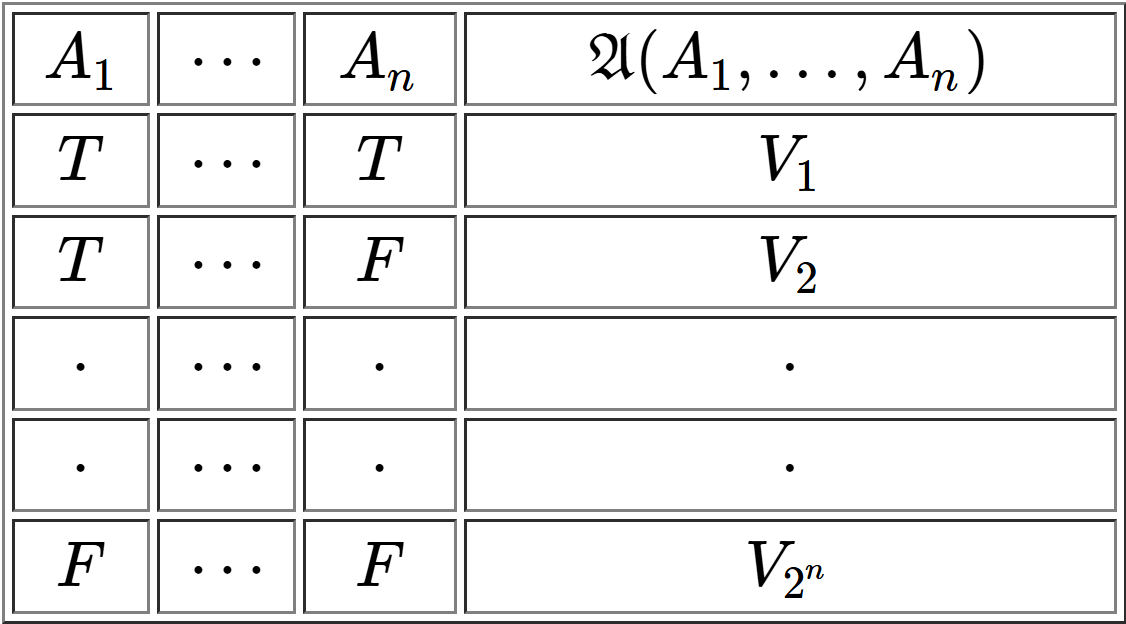
1. Automatické dokazování

Automatizované dokazování vět je zásadní pro různé aspekty informatiky. Obvykle se implementuje pomocí jednoduchých propozičních důkazových systémů. Objevování obtížných tautologií vrhá světlo na možnosti a užitečnost různých technik automatizovaného dokazování (Ben-Sasson and Wigderson, 2001).

Klíčovým prvkem většiny systémů pro dokazování vět je unifikační algoritmus, který pro danou dvojici vstupních výrazů vrací množinu unifikačních substitucí – přiřazení proměnným těchto výrazů, která je činí identickými nebo ekvivalentními. Typickým příkladem je unifikační algoritmus Robinsona pro unifikaci atomických formulí v rámci dokazování vět metodou rezoluce v predikátové logice prvního řádu (Stickel, 1981).

* 1. Pravdivostní tabulka

Pravdivostní tabulka je tabulka vyjadřující pravdivostní hodnoty složené výrokové formule na základě pravdivostních hodnot jednoduchých výroků, které ji tvoří (viz Pravdivostní hodnota). Pravdivostní tabulka má podobu tabulky níže, kde T označuje „pravda“ a F označuje „nepravda“.  
V tabulce Obr. 1 jsou A1, …, An​ výrokové proměnné, A(A1, …, An) je výroková formule a její pravdivostní hodnota je určena pravdivostními hodnotami A1, …, An. Každý řádek tabulky odpovídá jedné z 2n možných kombinací pravdivostních hodnot výrokových proměnných. Dále, *vi* je pravdivostní hodnota A(A1, …, An), pokud mají   
A1, …, An ​ pravdivostní hodnoty uvedené v *i*-tém řádku (*Truth table - Encyclopedia of Mathematics*, no date).

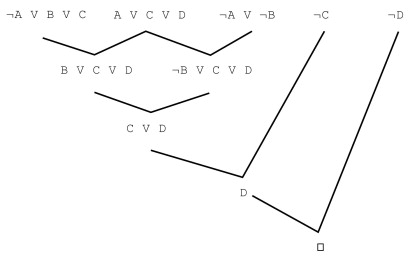


1. Pravdivostní tabulka

(Truth table – Encyclopedia of Mathematics, no date)

* 1. Rezoluční metoda

Rezoluční metoda, na kterou se tento článek zaměřuje, je možná tím nejjednodušším netriviálním systémem. Všechny tvrzení v tomto důkazovém systému jsou klauzule (tedy disjunkce literálů). Tautologie je reprezentována svou negací – jako množina protikladných klauzulí. (To je vždy možné díky NP-úplnosti problému SAT.) Důkaz (nebo vyvrácení) využívá jednoduché dedukční pravidlo k odvozování dalších klauzulí z těchto „axiomových klauzulí“, dokud není dosaženo kontradikce ve formě prázdné (nepravdivé) klauzule. Rezoluční metoda tvoří základ mnoha automatizovaných procedur dokazování vět, které se v praxi používají (Ben-Sasson and Wigderson, 2001).



1. Diagram postupu rezoluční metody
   * 1. Důkaz sporem

Chcete-li tvrzení dokázat rozporem, začněte tím, že předpokládáte opak toho, co byste chtěli dokázat. Pak ukažte, že důsledky tohoto předpokladu jsou nemožné. To znamená, že vaše původní tvrzení musí být pravdivé (*Proof by Contradiction | Brilliant Math & Science Wiki*, no date).

* + 1. Postup použití rezoluční metody

1. Tabulka 1 Rezoluce – zadané klauzule

(‘Propositional theorem proving • Unification’, no date)

|  |  |
| --- | --- |
| Dokaž R | |
| 1 | P v Q |
| 2 | P → R |
| 3 | Q → R |

1. Tabulka 2 Rezoluce – krok 1

(‘Propositional theorem proving • Unification’, no date)

|  |  |  |
| --- | --- | --- |
| **Krok** | **Formule** | **Odvození** |
| 1 | P v Q | Ze zadání |
| 2 | ¬ P v R | Ze zadání |
| 3 | ¬ Q v R | Ze zadání |
| 4 | ¬ R | Negace předpokladu |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Tabulka 1 obsahuje zadané klauzule a literál (R), který z nich má být odvozen. Do Tabulka 2 se postupně zapisuje průběh odvozování klauzulí. Aktuálně jsou v Tabulka 2 zapsány klauzule ze zadání, které jsou rovnou převedeny do KNF. V posledním zapsaném řádku se nachází negace předpokladu, jelikož pracujeme s algoritmem, kde hledáme důkaz sporem.

1. Tabulka 3 Rezoluce-krok 2

(‘Propositional theorem proving • Unification’, no date)

|  |  |  |
| --- | --- | --- |
| **Krok** | **Formule** | **Odvození** |
| 1 | P v Q | Ze zadání |
| 2 | ¬ P v R | Ze zadání |
| 3 | ¬ Q v R | Ze zadání |
| 4 | ¬ R | Negace předpokladu |
| 5 | Q v R | 1, 2 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

V Tabulka 3 je poprvé použita rezoluční metoda na řádku 5, kde mezi klauzulemi na řádcích 1 a 2 vzniká rezolventa Q v R po „zkrácení“ literálu P.

1. Tabulka 4 Rezoluce-krok 3

(‘Propositional theorem proving • Unification’, no date)

|  |  |  |
| --- | --- | --- |
| **Krok** | **Formule** | **Odvození** |
| 1 | P v Q | Ze zadání |
| 2 | ¬ P v R | Ze zadání |
| 3 | ¬ Q v R | Ze zadání |
| 4 | ¬ R | Negace předpokladu |
| 5 | Q v R | 1, 2 |
| 6 | ¬ P | 2, 4 |
| 7 | ¬ Q | 3, 4 |
| 8 | R | 5, 7 |
| 9 | {} | 4, 8 |

Stejným způsobem, jako u předchozího kroku se klauzule odvozují mezi sebou, dokud nezůstanou pouze takové, které dále odvozovat nejdou, anebo dokud nezbude pouze prázdná klauzule. V Tabulka 4 lze vidět, že se došlo k požadovanému sporu(kontradikci), což znamená, že množina klauzulí je splnitelná. V opačném případě by to znamenalo její nesplnitelnost(‘Propositional theorem proving • Unification’, no date).

1. Použité technologie

V současné době existuje velké množství způsobů, jak navrhnout a realizovat softwarovou aplikaci. Při vývoji aplikace je nutné provést několik rozhodnutí, například o typu aplikace, vývojovém prostředí, programovacím jazyce nebo konkrétních technologiích, které bude projekt využívat. Tato kapitola se zaměřuje na podrobný výběr jednotlivých komponent a nástrojů, které byly zvoleny pro vývoj této aplikace, včetně odůvodnění těchto rozhodnutí.

* 1. Typ aplikace

Při výběru typu aplikace je klíčové zohlednit, jaké funkce má aplikace plnit a jakým způsobem bude uživateli zpřístupněna. V rámci tohoto projektu existovalo několik variant provedení – například aplikace pro mobilní zařízení, desktopové prostředí nebo jako webovou aplikaci.

Nakonec byla zvolena aplikace pro web, jelikož webové rozhraní lépe vyhovuje uživatelům pracujícím na počítači, a to díky většímu prostoru pro zobrazování informací, kterých může být v této aplikaci značné množství. Zároveň je však přístupné i z mobilních zařízení, což zajišťuje širší dostupnost.

Další výhodou je možnost využít výpočetní výkon serveru, což je užitečné při náročnějších operacích, jako je převod formulí a provádění rezolučního algoritmu. Webová aplikace tak nabízí flexibilní a efektivní řešení pro potřeby tohoto projektu.

* 1. Backend

V odvětví vývoje softwaru termín „backend“ označuje část aplikace, kterou běžný uživatel nevidí. Slouží pro práci s daty a řeší logiku dané aplikace, jako je autentizace uživatele nebo práce s databází. Tato podkapitola se věnuje výběru konkrétních technologií použitých pro tvorbu aplikace.

* + 1. Python

Pro backend aplikace byl použit programovací jazyk python. Je to interpretovaný, objektově orientovaný jazyk. Zahrnuje moduly, výjimky, dynamické datové typy a třídy (*Python 3.13.2 documentation*, no date). I přes to, že python není nejrychlejší programovací jazyk, tak pro potřeby této aplikace je plně dostačující. Zde zvolen, jelikož nabízí různé frameworky, které pokrývají všechny veškeré požadavky aplikace. Výhodou je také, jednoduchá práce databázemi, API, ale i samotným vývojem algoritmu pro řešení logických formulí.

* + 1. Flask

Flask je lehký WSGI framework pro webové aplikace. Je navržen tak, aby umožňoval rychlý a snadný začátek, s možností rozšíření na složité aplikace (*Flask Documentation (3.1.x)*, no date). Tento framework je zvolen z důvodu jeho jednoduchosti, flexibilitě a nabídce různých knihoven, například pro práci s databázemi nebo řešením autentizace.

* + 1. API

API znamená rozhraní pro programování aplikací (Application Programming Interface). V kontextu API slovo „aplikace“ označuje jakýkoli software s konkrétní funkcí. „Rozhraní“ lze chápat jako smlouvu o službě mezi dvěma aplikacemi. Tato smlouva definuje, jak spolu komunikují pomocí požadavků a odpovědí. Dokumentace API obsahuje informace o tom, jak mají vývojáři tyto požadavky a odpovědi strukturovat.

Architektura API je obvykle vysvětlována z hlediska klienta a serveru. Aplikace odesílající požadavek se nazývá klient a aplikace odesílající odpověď se nazývá server. Například v případě počasí je databáze meteorologického úřadu serverem a mobilní aplikace je klientem(*Application Programming Interface Explained - AWS*, no date).

* 1. Frontend

Frontend je to, co uživatelé vidí, a zahrnuje vizuální prvky, jako jsou tlačítka, zaškrtávací políčka, grafika a textové zprávy. Umožňuje uživatelům interakci s aplikací. Dá se také označit jako grafické uživatelské rozhraní (GUI), se kterým mohou uživatelé přímo interagovat, například navigační menu, designové prvky, tlačítka, obrázky a grafy. Z technického hlediska se stránka nebo obrazovka, kterou uživatel vidí a která obsahuje různé prvky uživatelského rozhraní, nazývá model objektů dokumentu (DOM).

Tři hlavní programovací jazyky ovlivňují způsob, jakým uživatelé interagují s frontendem:

* HTML definuje strukturu frontendu a jednotlivé prvky DOM.
* Cascading Style Sheets (CSS) určují styl webové aplikace, včetně rozvržení, fontů, barev a vizuálního vzhledu.
* JavaScript přidává vrstvu dynamické funkcionality manipulací s DOM.

JavaScript může vyvolávat změny na stránce a zobrazovat nové informace. To znamená, že frontend dokáže zpracovat základní uživatelské interakce (nebo požadavky), jako je zobrazení kalendáře nebo ověření, zda uživatel zadal platnou e-mailovou adresu. Složitější požadavky frontend předává backendu(*Front End vs Back End - Difference Between Application Development - AWS*, no date).

* + 1. Javascript

JavaScript je skriptovací nebo programovací jazyk, který vám umožňuje implementovat složité funkce na webových stránkách – pokaždé, když webová stránka dělá víc, než jen sedí a zobrazuje statické informace, na které se můžete podívat – zobrazuje včasné aktualizace obsahu, interaktivní mapy, animovanou 2D/3D grafiku, rolovací video, jukeboxy atd.(*What is JavaScript? - Learn web development | MDN*, no date).

* + 1. React

React (také známý jako React.js nebo ReactJS) je bezplatná a open-source front-end knihovna JavaScriptu, jejímž cílem je učinit vytváření uživatelských rozhraní založených na komponentách „pohodlnějším“(*React*, no date).

Framework React byl vybrán do aplikace, protože nabízí efektivní způsob práce s komponentami a stavem aplikace. Díky jeho deklarativnímu přístupu je snadné vytvářet přehledné a interaktivní uživatelské rozhraní, které se dynamicky aktualizuje podle změn dat.

* + 1. Bootstrap

Bootstrap je HTML, CSS a JS knihovna, která se zaměřuje na zjednodušení vývoje informativních webových stránek (na rozdíl od webových aplikací). Primárním účelem přidání do webového projektu je aplikovat na tento projekt volby barvy, velikosti, písma a rozvržení Bootstrapu. Primárním faktorem proto je, zda odpovědní vývojáři najdou tyto možnosti podle svých představ. Po přidání do projektu poskytuje Bootstrap základní definice stylů pro všechny prvky HTML. Výsledkem je jednotný vzhled prózy, tabulek a formulářových prvků napříč webovými prohlížeči. Kromě toho mohou vývojáři využít třídy CSS definované v Bootstrapu k dalšímu přizpůsobení vzhledu jejich obsahu(*Bootstrap (front-end framework) - Wikipedia*, no date).

Bootstrap v aplikaci pomáhá rychleji a jednodušeji vytvářet responzivní a vizuálně jednotné rozhraní. Díky předpřipraveným komponentám, jako jsou tlačítka, formuláře nebo mřížky, se u vývoje dá více soustředit na logiku aplikace a méně času trávit nad návrhem stylů. Bootstrap také zajišťuje konzistentní vzhled napříč různými prohlížeči a zařízeními, což je důležité pro dobrou uživatelskou zkušenost.

* 1. Databáze

Databáze je organizovaný soubor strukturovaných informací nebo dat, obvykle uložených elektronicky v počítačovém systému. Databáze je obvykle řízena systémem správy databází (DBMS). Společně se data a DBMS spolu s aplikacemi, které jsou s nimi spojeny, označují jako databázový systém, často zkrácený pouze na databázi.

Data v nejběžnějších typech dnes provozovaných databází jsou typicky modelována v řádcích a sloupcích v řadě tabulek, aby bylo zpracování a dotazování na data efektivní. K datům pak lze snadno přistupovat, spravovat je, upravovat, aktualizovat, kontrolovat a organizovat. Většina databází používá pro zápis a dotazování na data strukturovaný dotazovací jazyk (SQL)(*What Is a Database? | Oracle*, no date).

* + 1. MySQL

MySQL je open-source systém pro správu relačních databází (RDBMS)(*MySQL 8.0 Reference Manual 1.2.1*, no date).

MySQL jsem si vybral jako databázový systém pro svou aplikaci, protože je spolehlivý, široce používaný a dobře podporovaný. Umožňuje efektivně ukládat a spravovat strukturovaná data, jako jsou přihlašovací údaje nebo historie řešení výrokových formulí. Díky jednoduché integraci s backendovými technologiemi a dostupnosti nástrojů pro správu databáze bylo snadné nastavit a používat MySQL během vývoje aplikace. Jeho výhodou je také dobrý výkon a možnost rozšíření.

* 1. Bezpečnost

Zabezpečení webových stránek je praxe, metody a protokol zaměřené na ochranu a zabezpečení webové stránky a jejích dat. Metody jsou navrženy tak, aby zabránily širokému spektru kybernetických bezpečnostních hrozeb, jako jsou DDoS útoky, SQL injection útoky nebo malware. Hrozby kybernetické bezpečnosti mohou mít na podniky široký a zničující dopad, jako jsou ekonomické náklady, jako je ztráta příjmů a výdajů na nápravu, poškozená pověst značky v důsledku špatného mediálního pokrytí a ztráty důvěry a regulační náklady, jako jsou značné pokuty za neochrání údajů PII v souladu s GDPR nebo HIPAA(*How to secure a website | Cloudflare*, no date).

* + 1. API Komunikace

Zabezpečení API je základní součástí zabezpečení webových aplikací. Většina moderních webových aplikací spoléhá na to, že fungují rozhraní API, a rozhraní API představují další riziko pro aplikaci tím, že umožňují přístup vnějším stranám. Jedním ze srovnání je podnik, který otevírá svou kancelář veřejnosti: více lidí v prostorách, z nichž někteří mohou být zaměstnanci podniku neznámí, představuje větší riziko. Podobně rozhraní API umožňuje používat program nepovolaným osobám, což představuje větší riziko pro infrastrukturu služby API(*Web API security | Cloudflare*, no date).

**Bezpečnostní rizika:**

* **Zneužití zranitelnosti:**

Zneužití zranitelnosti je, když útočník pošle speciálně vytvořená data na cíl, data, která využívají chybu v konstrukci cíle. Tyto chyby, známé jako „zranitelnosti“, mohou útočníkovi poskytnout různé formy nezamýšleného přístupu k rozhraní API nebo jeho odpovídající aplikaci. Open Web Application Security Project (OWASP) udržuje seznam 10 hlavních zranitelností API, jako je SQL injection, chybná konfigurace zabezpečení a další. Pokud se exploit zaměřuje na zranitelnost, která byla dříve neznámá, nazývá se to zero-day hrozba – takové hrozby je extrémně obtížné zastavit(*Web API security | Cloudflare*, no date).

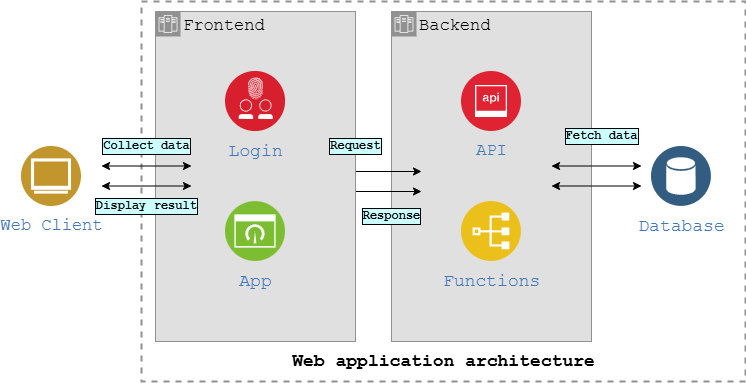
* **Útoky založené na autentizaci:**

Klienti se musí před tím, než mohou zadávat požadavky API, ověřit, aby server API nepřijímal požadavky z neznámých nebo nelegitimních zdrojů. Existuje několik způsobů, jak to udělat, ale každý způsob podléhá kompromisu. Útočník by například mohl získat přihlašovací údaje legitimního klienta, ukrást klíč API nebo zachytit a použít ověřovací token(*Web API security | Cloudflare*, no date).

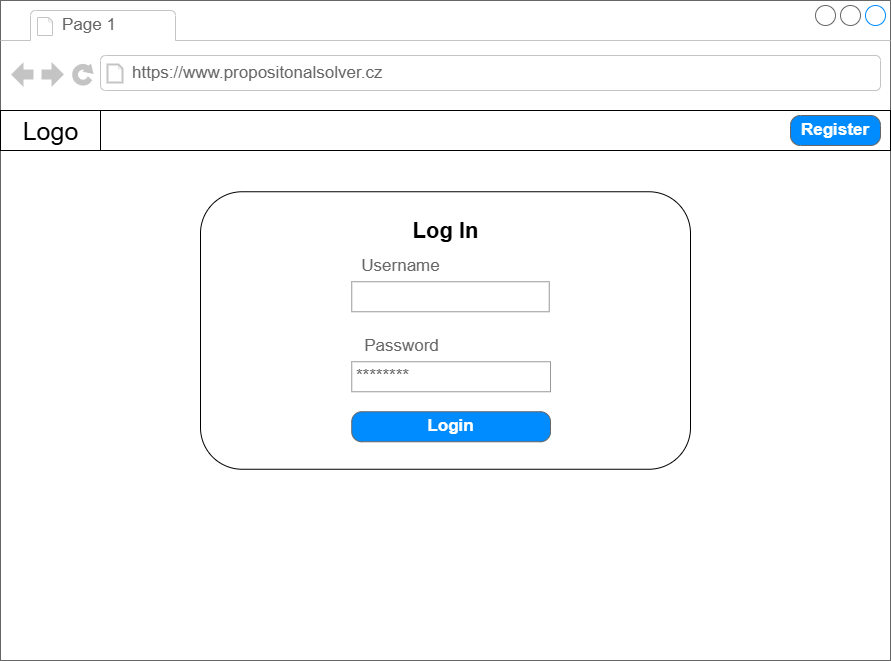
* **Chyby autorizace:** Autorizace určuje úroveň přístupu každého uživatele. Pokud není autorizace pečlivě spravována, může mít klient API přístup k datům, která by mu neměla být dostupná, čímž se zvyšuje pravděpodobnost narušení dat(*Web API security | Cloudflare*, no date).
  + 1. Komunikace uživatele se serverem

Zajištění komplexního zabezpečení webu vyžaduje vyvážené zaměření na ochranu na straně klienta a serveru. Zatímco zabezpečení na straně klienta se zaměřuje na ochranu kódu, který běží v prohlížečích uživatelů, zabezpečení na straně serveru chrání backendové procesy a data. Efektivní strategie zabezpečení webu musí integrovat obě strany, aby zmírnily širokou škálu hrozeb. Například úprava kódu na straně klienta tak, aby byl obtížně čitelný, zabraňuje zpětnému inženýrství, zatímco ověřování a šifrování na straně serveru chrání citlivá data. Jednotný přístup zajišťuje, že zranitelnosti na obou koncích neohrožují celou aplikaci a vytváří bezproblémové a bezpečné uživatelské prostředí. Kombinací těchto vrstev zabezpečení se mohou organizace lépe chránit před moderními útoky, které cílí na obě strany aplikace(‘Client-Side vs. Server-Side Security | Blog | Digital.ai’, no date).

1. Návrh
   1. Architektura



1. Diagram – Architektura aplikace
   1. Drátěný model
      1. Login



1. Drátěný model – login
   * 1. Registrace

A screenshot of a computer

AI-generated content may be incorrect.

* + 1. Solve

A screenshot of a computer

AI-generated content may be incorrect.

* 1. API struktura

|  |  |
| --- | --- |
| Cesta | Popis |
| POST /api/register/{username, password} | Registrace uživatele |
| POST /api/login/{username, password} | Uživatelské přihlášení |
| POST /api/solve | Řešení formule |

1. Implementace

Architektura = obrazek = client side, server side

Struktura api callu - /api/solve, /login, …(tabulka)

Stejne jako navrh, ale s praktickyma ukazkama kodu, diagramu, templatama stranek, …

Api planning

Db = diagram, kus kodu s pozitim databaze

Struktura client side callu = -app/solve, app/login, …

1. Diskuse a zhodnocení
   1. Přínos aplikace
   2. Omezení a možná vylepšení
   3. Možnosti dalšího rozvoje
2. Závěr
   1. Shrnutí výsledků
   2. Možnost praktického využití
3. Literatura

*Application Programming Interface Explained - AWS* (no date). Available at: https://aws.amazon.com/what-is/api/ (Accessed: 2 April 2025).

Ben-Sasson, E. and Wigderson, A. (2001) ‘Short proofs are narrow—resolution made simple’, *Journal of the ACM*, 48(2), pp. 149–169. Available at: https://doi.org/10.1145/375827.375835.

*Bootstrap (front-end framework) - Wikipedia* (no date). Available at: https://en.wikipedia.org/wiki/Bootstrap\_(front-end\_framework) (Accessed: 2 April 2025).

‘Client-Side vs. Server-Side Security | Blog | Digital.ai’ (no date) *https://digital.ai/* [Preprint]. Available at: https://digital.ai/catalyst-blog/client-side-vs-server-side-security/ (Accessed: 2 April 2025).

*Flask Documentation (3.1.x)* (no date). Available at: https://flask.palletsprojects.com/en/stable/ (Accessed: 31 March 2025).

*Front End vs Back End - Difference Between Application Development - AWS* (no date). Available at: https://aws.amazon.com/compare/the-difference-between-frontend-and-backend/ (Accessed: 2 April 2025).

*How to secure a website | Cloudflare* (no date). Available at: https://www.cloudflare.com/learning/security/how-to-secure-a-website/ (Accessed: 2 April 2025).

*MySQL 8.0 Reference Manual 1.2.1* (no date). Available at: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html (Accessed: 2 April 2025).

*Proof by Contradiction | Brilliant Math & Science Wiki* (no date). Available at: https://brilliant.org/wiki/contradiction/ (Accessed: 3 April 2025).

‘Propositional theorem proving • Unification’ (no date).

*Python 3.13.2 documentation* (no date). Available at: https://docs.python.org/3/tutorial/index.html (Accessed: 31 March 2025).

*React* (no date). Available at: https://react.dev/ (Accessed: 2 April 2025).

Stickel, M.E. (1981) ‘A Unification Algorithm for Associative-Commutative Functions’, *Journal of the ACM*, 28(3), pp. 423–434. Available at: https://doi.org/10.1145/322261.322262.

*Truth table - Encyclopedia of Mathematics* (no date). Available at: https://encyclopediaofmath.org/index.php?title=Truth\_table (Accessed: 30 March 2025).

*Web API security | Cloudflare* (no date). Available at: https://www.cloudflare.com/learning/security/api/what-is-api-security/ (Accessed: 2 April 2025).

*What Is a Database? | Oracle* (no date). Available at: https://www.oracle.com/database/what-is-database/ (Accessed: 2 April 2025).

*What is JavaScript? - Learn web development | MDN* (no date). Available at: https://developer.mozilla.org/en-US/docs/Learn\_web\_development/Core/Scripting/What\_is\_JavaScript (Accessed: 2 April 2025).

* 1. Seznam bibliografických citací