Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

# Výukový program demonštrujúci matematický princíp

Bakalárska práca

2015 Monika Švaralová

**Univerzita Komenského v Bratislave**
**Fakulta matematiky, fyziky a informatiky**

# Výukový program demonštrujúci matematický princíp

## Bakalárska práca

Študijný program: Aplikovaná informatika
Študijný odbor: 2511 Aplikovaná informatika
Školiace pracovisko: FMFI.KAI – Katedra aplikovanej informatiky
Školiteľ: Mgr. Pavel Petrovič, PhD.

**Bratislava 2015** **Monika Švaralová**

56474065

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

# ZADANIE ZÁVEREČNEJ PRÁCE

**Meno a priezvisko študenta:** Monika Švaralová
**Študijný program:** aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
**Študijný odbor:** 9.2.9. aplikovaná informatika
**Typ záverečnej práce:** bakalárska
**Jazyk záverečnej práce:** slovenský
**Sekundárny jazyk:** anglický

**Názov:** Výukový program demonštrujúci matematický princíp
*Educational program demonstrating a mathematical principle*

**Cieľ:** Špecifikovať, navrhnúť a implementovať výukový program pre stredoškolskú matematiku, ktorý názornou interaktívnou simuláciou vysvetľuje matematický princíp. Interaktivita spočíva v nastavení rozličných parametrov, alebo predpokladov simulácie. Inšpiráciou k práci študenta môžu byť publikácie zo série Škola mladých matematikov. Študent analyzuje oblasti vhodné pre uplatnenie interaktívnej simulácie a vyberie konkrétnu oblasť matematiky, ktorú na názorných príkladoch aplikácia bude vysvetlovať.

**Literatúra:**
1. Lukáš Slovák: Výukový program demonštrujúci fyzikálny princíp, bakalárska práca, FMFI UK, Bratislava, 2011.
2. Jozef Belko: Výukový program demonštrujúci fyzikálny princíp, bakalárska práca, FMFI UK, Bratislava, 2013.
Oracle: Java FX API Documentation, online: http://docs.oracle.com/javafx/2/api/index.html

**Kľúčové slová:** výukový program, matematika

**Vedúci:** Mgr. Pavel Petrovič, PhD.
**Katedra:** FMFI.KAI - Katedra aplikovanej informatiky
**Vedúci katedry:** prof. Ing. Igor Farkaš, PhD.

**Dátum zadania:** 05.10.2014

**Dátum schválenia:** 04.11.2014

doc. RNDr. Mária Markošová, PhD.
garant študijného programu

študent

Paul Petrovič

vedúci práce

## Čestné vyhlásenie

Čestne prehlasujem, že som túto bakalársku prácu vypracovala samostatne s použitím uvedených zdrojov.

V Bratislave .....

Monika Švaralová

## Poďakovanie

Chcela by som sa poďakovať školiteľovi Mgr. Pavlovi Petrovičovi, PhD. za jeho cenné rady, odbornú pomoc a čas, ktorý mi venoval pri tvorbe práce.

# Abstrakt

Táto bakalárska práca sa zaoberá návrhom a implementáciou výukovej webovej aplikácie, v ktorej si študenti môžu precvičiť základné princípy matematickej logiky pri konštruovaní matematických dôkazov. V aplikácii sa nachádza séria úloh, v ktorých si študent vyskúša odvodenie viet Boolovej algebry z axióm použitím dvoch pravidiel logiky: pravidla substitúcie a nahradenia ekvivalentných podformúl. Program mu pri tom napomáha a kontroluje správnosť jeho postupu. Vo výbere príkladov a ich riešení vychádzame z knihy Škola mladých matematikov: Booleova algebra [1].

**Kľúčové slová:** *matematický dôkaz, Boolova algebra, logika, výukový program, webová aplikácia*

# Abstract

The main goal of this bachelor thesis is to design and implement a web application that allows students to practice the basic principles of mathematical logic in constructing mathematical proofs. The application contains a series of exercises, in which the user can derive theorems of Boolean algebra from axioms using rules of inference in propositional logic. The program aids the user and checks validity of the proof steps. We use the book School for Young Mathematicians: Boolean Algebra [1] as a primary resource of exercises and their solutions.



**Keywords:** *mathematical proof, Boolean algebra, logic, educational program, web application*

# Obsah

# 1. Úvod

Riešenie matematického príkladu sa dá prirovnať k stromu rozhodnutí. Rozhodujeme sa, či použijeme niektorú z viet, o ktorej vieme, že platí, alebo upravíme výraz v príklade pomocou určitého pravidla, ktoré poznáme. Nie vždy je cesta k cieľu v tomto strome priamočiara. Niekedy je potrebné výraz na chvíľu trochu skomplikovať, aby sa napokon zjednodušil a dosiahli sme želaný výsledok. Hľadanie riešenia je ako hlavolam.

Jedným z tohto typu príkladov sú matematické dôkazy. Máme nejakú matematickú vetu, o ktorej je potrebné dokázať, že platí. Musíme si vybrať, akú metódu použijeme, z akých predpokladov budeme vychádzať a skonštruovať logický argument. Celý tento proces môže byť pre študenta odstrašujúci. Navyše, na rozdiel od hľadania hodnoty premennej $x$ v rovnici, nie je také jednoduché sám si skontrolovať, či je výsledok, teda logický argument, správny.

To bolo inšpiráciou k téme mojej práce. Jej cieľom je navrhnúť a implementovať webovú aplikáciu pre študentov vysokých a stredných škôl, v ktorej si budú môcť krok za krokom "vyskladať" postup matematického dôkazu, ktorého správnosť sa overí automaticky.

Matematické dôkazy však majú často tú komplikáciu, že aj keď ich slovná interpretácia je zrozumiteľná riešiteľovi, ich počítačová reprezentácia môže byť oveľa komplikovanejšia a nevhodná na výukové účely. Preto som zvolila dôkazy, ktoré sa dajú takmer rovnako jednoducho reprezentovať v počítačovej aplikácii ako popísať v knihe: vety z Boolovej algebry. Keďže jednou z požiadavok zadania práce bolo vychádzať zo série kníh Škola mladých matematikov, použila som príklady z jednej knihy v tejto sérii: Booleova algebra [1].

Aby bola aplikácia prístupná pre užívateľov bez nutnosti čokoľvek sťahovať a inštalovať, iba po načítaní webovej adresy vo svojom prehliadači, vytvorím ju ako interaktívnu webovú single-page aplikáciu. Dôležitá vo výukovej aplikácii je tiež jej použiteľnosť. Aj nový používateľ by mal vedieť v krátkom čase pochopiť, ako sa používa. Preto sa budem snažiť o jednoduché a zrozumiteľné ovládanie, čistý dizajn a na naučenie sa jej používania bude slúžiť aj interaktívny tutoriál.

Nasleduje kapitola Východiská, v ktorej opíšem matematické základy, z ktorých aplikácia vychádza, podobné existujúce programy a technológie, ktoré využijem pri tvorbe aplikácie.

Ďalej popíšem návrh aplikácie, teda požiadavky, ktoré by mala spĺňať, analýzu riešenia a vnútornú architektúru všetkých vrstiev aplikácie.

V kapitole Implementácia opíšem vybrané algoritmy a problémy, s ktorými som sa stretla počas implementácie.

Aplikáciu otestujem so študentami Fakulty informatiky a informačných technológií STU, ktorí v tomto semestri preberajú matematickú logiku, a výsledky tohto testovania budem hodnotiť v kapitole Testovanie.

# 2. Východiská

## 2.1 Prehľad teórie

### 2.1.1 Séria kníh Škola mladých matematikov

Škola mladých matematikov je séria 61 kníh vydávaná v rokoch 1961-1988 v Československu, založená na podnet Ústredného výboru matematickej olympiády [2]. Bola vytvorená kolektívom autorov pre stredoškolských študentov, najmä pre tých, čo sa venujú matematickej olympiáde, na prehĺbenie matematických vedomostí.

Každá z kníh sa venuje určitej matematickej téme, niektoré vychádzajú zo stredoškolského učiva, ale väčšinou preberajú aj témy z oblasti vyššej matematiky. Knihy na seba navzájom nenadväzujú. Väčšina z nich obsahuje krátky výklad preberanej témy a podrobne riešené príklady.

Keďže jednou z požiadaviek zadania práce bolo vychádzať z tejto série kníh, pri výbere témy aplikácie som sa zamerala na ňu, a vybrala som konkrétne 31. knihu v sérii: Booleova algebra. V aplikácii budem používať príklady dôkazov a navrhnutý postup riešenia práve z tejto knihy.

### 2.1.2 Matematické dôkazy

Matematický dôkaz je vysvetlenie alebo argument, ktorý dokáže presvedčiť ostatných matematikov, že dané tvrdenie je pravdivé. Dobrý dôkaz im však aj pomôže pochopiť, prečo je tvrdenie pravdivé.

V súčasnosti je niektoré dôkazy možno automatizovať použitím počítačov. Napríklad dôkaz vety, že každú mapu je možné vyfarbiť najviac 4 farbami tak, aby sa oblasti rovnakých farieb nenachádzali pri sebe, pomocou automatického výpočtu, vyvolal v matematickej komunite určitú kontroverziu. Nie preto, že by matematici pochybovali o pravdivosti vety, ale skôr to odráža dôležitý zámer štúdia matematiky: snahu o ľudské pochopenie, prečo je veta pravdivá [3].

Dôkazy využívajú logiku, ale v praxi sa v matematických publikáciách a najmä vo vzdelávacej literatúre používa skôr neformálna logika, často v kombinácii s prirodzeným jazykom, ktorý môže zavádzať do dôkazov nejasnosti.

Formálny dôkaz je postupnosť viet, z ktorých každá je buď axióma, alebo vyplýva z predchádzajúcich viet podľa inferenčných pravidiel matematickej logiky. Takýmito dôkazmi sa zaoberá teória dôkazov. Formálne dôkazy sú často zostrojené za pomoci automatizovaných dokazovacích programov.

Dokázať môžeme matematický výrok. Je to veta, ktorá je buď pravdivá, alebo nepravdivá, ale nie oboje. Na matematické výroky je možné aplikovať logické operácie, ako $\neg$, $\wedge$, $\vee$, alebo $\Rightarrow$. Axiomatickú metódu, ktorá pomocou deduktívnej logiky dokazuje tvrdenia z axióm, teda viet, ktoré boli pokladané za pravdivé, použil Euklides už v roku 300 pred n.l.

### 2.1.3 Metódy dôkazov

#### 2.1.3.1 Priamy dôkaz

Pomocou priameho dôkazu určíme pravdivostnú hodnotu výroku použitím logickej dedukcie z axióm a viet, pre ktoré už existuje dôkaz. Používa sa prvorádová logika, ktorá využíva okrem logických operácií aj kvantifikátory $\forall$, a $\exists$. Aby sme priamo dokázali výrok $p \Rightarrow q$, musíme zvážiť všetky situácie, kedy je $p$ pravdivé.

#### 2.1.3.2 Nepriamy dôkaz

Pri nepriamom dôkaze využívame, že pravdivostná hodnota výroku $p \Rightarrow q$ je rovnaká ako výroku $\neg q \Rightarrow \neg p$. Dokazuje sa tento obrátený výrok. Používa sa teda na dôkaz výrokov v tvare $p \Rightarrow q$.

#### 2.1.3.3 Dôkaz sporom

Pri dôkaze sporom sa snažíme ukázať, že ak by bolo určité tvrdenie pravdivé, nastane logický spor, a teda dané tvrdenie musí byť nepravdivé. Pre dôkaz výroku $p$ ukážeme, že ak by bol výrok $p$ nepravdivý, a teda by bol pravdivý výrok $\neg p$, dostaneme sa pri dôkaze dedukciou k tvrdeniu, o ktorom vieme, že je nepravdivé, a teda je to spor s východiskovým predpokladom.

#### 2.1.3.4 Dôkaz matematickou indukciou

Matematická indukcia je užitočná pri dokazovaní výrokov o prirodzených číslach. Je potrebné dokázať základný prípad, a ďalej pravidlo, že každý prípad implikuje nasledujúci prípad. Pre dokázanie výroku, že vlastnosť $P(n)$ platí pre všetky prirodzené čísla $n$, je potrebné dokázať nasledujúce dve tvrdenia:
(a) $P(1)$.
(b) $(\forall n \in \mathbb{Z}^+)P(n) \Rightarrow P(n+1)$.
Potom je $P(n)$ pravdivé pre všetky prirodzené čísla.

### 2.1.4 Logický dôsledok a pravidlá odvodzovania formúl výrokovej logiky

Logický dôsledok je "*presne špecifikovaný spôsob odvodzovania logických zákonov (tautológií), pričom sa vychádza z niekoľko málo vopred východiskových zákonov – axióm (tautológií), z ktorých pomocou presne špecifikovaného spôsobu dôkazu zostrojíme nové zákony (tautológie)*"[4].

V [4] sú ďalej uvedené tri základné pravidlá pre konštrukciu logického dôsledku:

*(1)* ***Pravidlo modus ponens*** *(pravidlo odlúčenia). Ak formuly $\phi$ a $\phi \Rightarrow \psi$ sú pravdivé, potom je pravdivá aj formula $\psi$. Toto pravidlo sa niekedy zapisuje aj ako schéma*

$$\frac{p \qquad p \Rightarrow q}{q}$$

*(2)* ***Pravidlo substitúcie****. Nech $\phi$ je tautológia, ktorá obsahuje výrokové premenné $(p_1, p_2, \ldots, p_n)$. Nech $\{\psi_1, \psi_2, \ldots, \psi_n\}$ je množina ľubovoľných formúl (ktorých počet je rovnaký ako počet premenných v $\phi$). Nech formula $\psi$ vznikne z $\phi$ tak, že každá premenná $p_1$ je substituovaná formulou $\psi_1$, pre $i = 1, 2, \ldots, n$*

$\psi = \phi(p_1/\psi_1, p_2/\psi_2, \ldots, p_n/\psi_n)$

*Potom takto vytvorená formula $\psi$ je opäť tautológiou.*

*(3)* ***Pravidlo nahradenia ekvivalentných podformúl****. Nech $\phi$ je tautológia a nech $\psi$ vznikne z $\phi$ substitúciou jej ľubovoľnej podformuly $\phi' \subset \phi$ formulou $\psi'$, ktorá je s ňou ekvivalentná, $\phi' \equiv \psi'$*

$\psi = \phi(\phi'/\psi')$

*potom aj $\psi$ je tautológia.*

Pravidlá (2) a (3) budú použité v programe na vytváranie odvodených viet a ekvivalentných úprav výrazov Boolovej algebry.

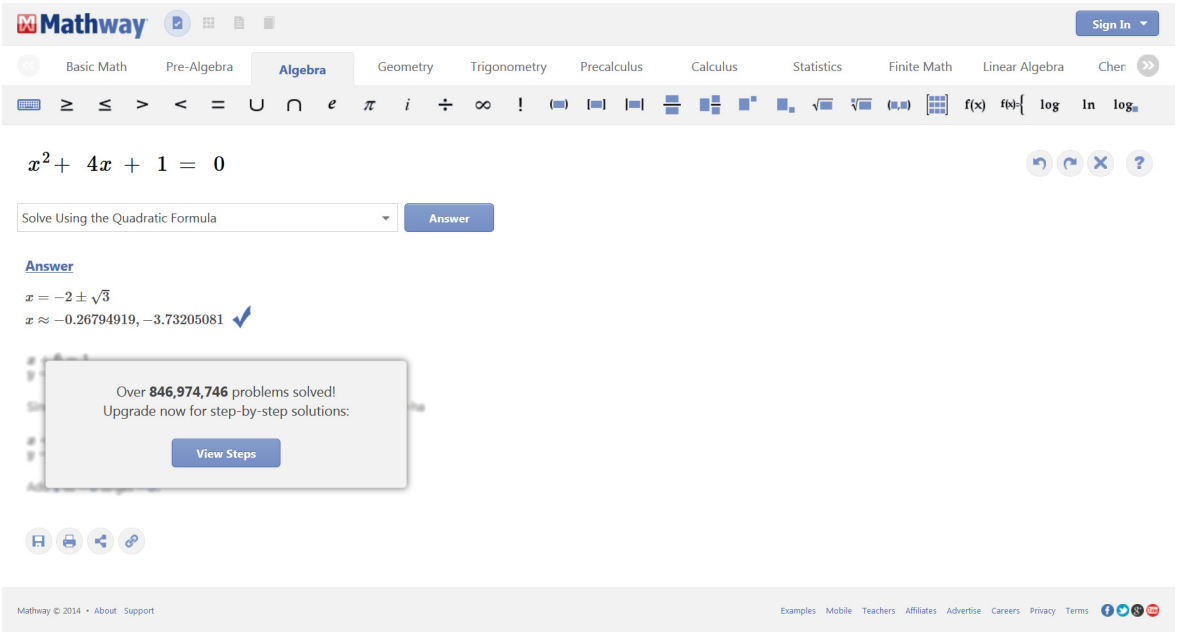
### 2.1.5 Boolova algebra

Podľa knihy Booleova algebra zo série Škola mladých matematikov [1] je definícia Boolovej algebry nasledovná: *Majme danú neprázdnu množinu $B$, v ktorej je definovaná rovnosť, existujú v nej vzájomne rôzne prvky $0, 1$ a sú na nej definované operácie $u = x + y$, $u = xy$ a unárna operácie $u = x'$. Množinu $B$ spolu s príslušnými operáciami budeme nazývať Boolova algebra, práve keď pre všetky prvky $x, y, z$ množiny $B$ platí:*
*(1)* $x + y = y + x$
*(2)* $x \cdot y = y \cdot x$
*(3)* $(x + y) + z = x + (y + z)$
*(4)* $(x \cdot y) \cdot z = x \cdot (y \cdot z)$
*(5)* $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$
*(6)* $x + (y \cdot z) = (x + y) \cdot (x + z)$
*(7)* $x + 0 = x$
*(8)* $x \cdot 1 = x$
*(9)* $x + x' = 1$
*(10)* $x \cdot x' = 0$
*Takto zavedenú Boolovu algebru budeme označovať* $(B, +, \cdot, ')$.

## 2.2 Existujúce podobné programy

V súčasnosti existuje mnoho webových, mobilných a desktopových aplikácií pre žiakov základných a stredných škôl, zameraných na pochopenie matematických princípov. Sú orientované na interaktívnu výuku, často aj s tréningom učebných konceptov pomocou automaticky hodnotených príkladov. Najmä pre menšie deti je niekoľko aplikácií, ktoré používajú formu hry na vysvetlenie matematických princípov, často s veľmi inovatívnym nápadom. Dajú sa však nájsť aj zatiaľ málo pokryté oblastí školskej matematiky, alebo nové kreatívne riešenia pri jej výuke pomocou informačných technológií.

V oblasti matematických dôkazov, ktorej sa budem v tejto práci venovať, som nenašla žiadne prístupné a interaktívne výukové aplikácie pre žiakov. V tejto sekcii uvádzam niekoľko populárnych výukových webových aplikácii, ktoré sú podobné v niektorých elementoch, alebo boli inšpiráciou pri tvorbe rozhrania aplikácie.




**Obr. 2.1:** Mathway [5]

### 2.2.1 Mathway

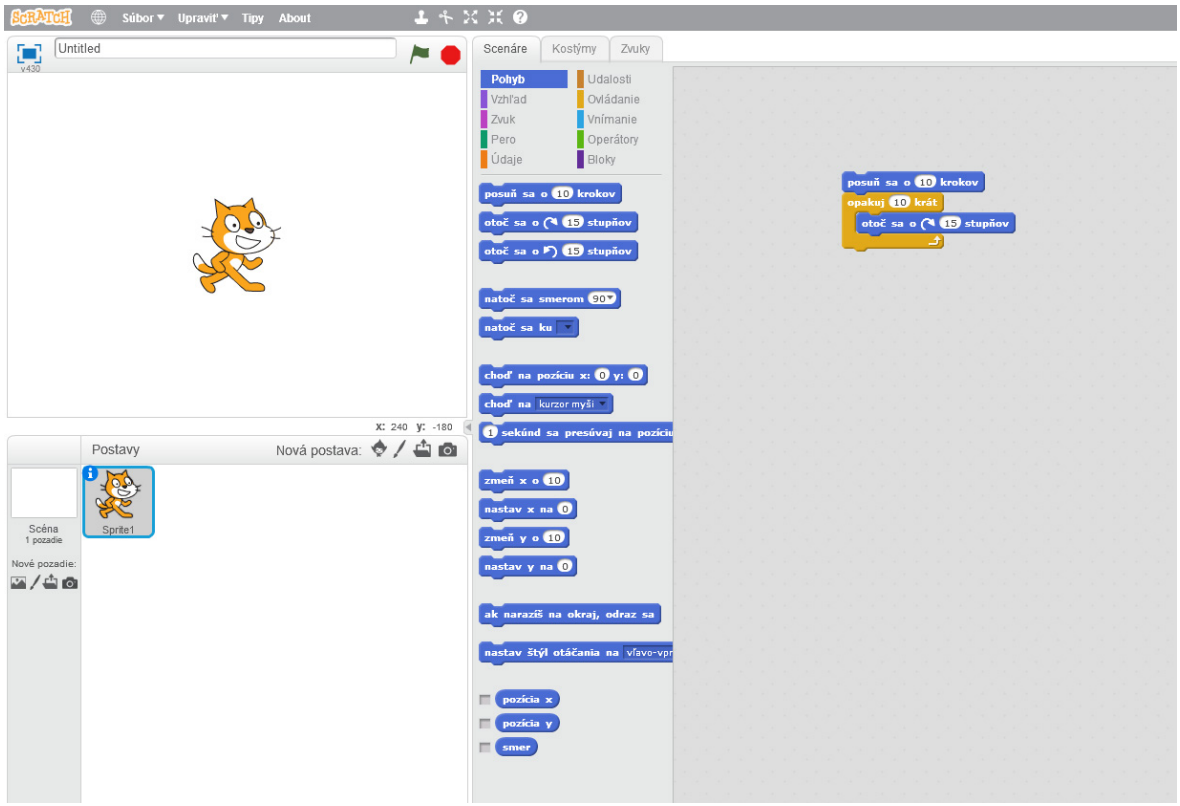
Mathway (Obr. 2.1) je webová aplikácia, ktorá ukáže pre zadaný príklad výsledok, grafy a riešenie krok za krokom. Pokrýva mnoho oblastí matematiky z osnov pre základné a stredné školy, a pri vložení príkladu je možné aj vybrať niektorú z možností jeho interpretovania, ktoré program nájde, že sa dajú naň aplikovať. Obsahuje aj množstvo vzorových príkladov. Neobsahuje však možnosť riešenia príkladov matematickej logiky. Riešenia ľubovoľných príkladov krok za krokom sú dostupné iba pri predplatenej verzii.

Príklady je možné zadať pomocou grafického rozhrania, ktoré je intuitívne na použitie, takže nie je potrebné poznať žiadne textové formátovacie príkazy pre matematickú notáciu.

Riešenie príkladov ponúka aj webová aplikácia Wolfram Alpha [6], tá však nemá také jednoduché grafické rozhranie pre zadávanie príkladov. Je v ňom možné aj hľadať riešenia a pravdivostné tabuľky výrazov z Boolovej algebry. Mathway je však viac zameraný na žiakov základných a stredných škôl a ich študijný plán.

### 2.2.2 Khan Academy

Khan Academy [7] je jeden z najpopulárnejších online zdrojov interaktívnych vzdelávacích materiálov pre školákov. Od svojich začiatkov bol zameraný prevažne na matematiku základných a stredných škôl a okrem výukových videí obsahuje na precvičenie




**Obr. 2.2:** Scratch [8]

aj generované príklady s automatickým vyhodnotením správnosti odpovede a rozličné mini-aplikácie na interaktívne vyskúšanie si vyučovanej látky.

### 2.2.3 Scratch

Scratch (Obr. 2.2) je webová aplikácia na výučbu programovania najmä pre deti, zábavnou formou. Hoci je určený na výučbu programovania, niektoré prvky jeho grafického rozhrania so skladaním algoritmov krok za krokom ma inšpirovali k návrhu rozhrania v aplikácii.

## 2.3 Technológie

Aby bola aplikácia prístupná pre používateľov bez nutnosti čokoľvek sťahovať a inštalovať, bude vytvorená ako interaktívna single-page webová aplikácia primárne v JavaScripte s pomocou knižnice jQuery. Okrem toho bude využívať aj server-side technológie – jazyk PHP s MVC frameworkom CodeIgniter na niektoré pomocné funkcionality, ako napríklad ukladanie dát a prihlasovanie používateľov, a MySQL na prácu s databázou.

### 2.3.1 JavaScript

JavaScript je interpretovaný jazyk implementovaný vo všetkých moderných webových prehliadačoch a teda veľmi rozšírený a jednoducho dostupný.

Skladá sa z troch základných častí [9]:

- ECMAScript, jadro JavaScriptu, je programovací jazyk, nezávislý od prehliadača
- DOM (Document Object Model) slúži na reprezentáciu a manipuláciu webstránky v HTML formáte, ako stromu uzlov
- BOM (Browser Object Model) umožňuje prístup k oknu prehliadača

Medzi moderné prvky JavaScriptu patrí aj AJAX (Asynchronous JavaScript and XML), čo je skupina technológii, ktorá umožňuje asynchrónne posielanie HTTP požiadaviek serveru, a načítavanie dát do už existujúcej stránky.

### 2.3.2 jQuery

jQuery je open-source knižnica postavená na JavaScripte, ktorá zjednodušuje písanie JavaScriptových programov. Uľahčuje navigáciu v DOM strome, výber elementov pomocou CSS selektorov, spracovanie udalostí a animácii, a posielanie AJAXových volaní na server. Okrem toho pomáha eliminovať nekompatibility v implementácii JavaScriptu medzi rôznymi prehliadačmi. Kód v jQuery pripomína skôr popis udalostí, ktoré by mali nastať, než program v JavaScripte.

### 2.3.3 jQueryUI

JQueryUI je postavená na jQuery, a rozširuje ju o niekoľko používateľských interakcii, efektov a tém. V aplikácii z nej použijem funkcionalitu ťahania HTML elementov (draggable) a autocomplete pri input prvku, ktorý je možné prepojiť aj na načítavanie dát z databázy na strane servera.

### 2.3.4 Math.js

Knižnica math.js obsahuje okrem mnohých matematických funkcií aj funkcionality na parsovanie a reprezentáciu výrazových stromov (expression trees) a ich vypísanie v TeX formáte. Túto knižnicu som však nakoniec nepoužila vo finálnej aplikácii.

### 2.3.5 MathJax

MathJax slúži na zobrazenie matematického textu vo formáte TeX. Znaky, ktoré vzniknú po renderovaní, sú vo vektorovom formáte – ako webové fonty alebo SVG, takže vyzerajú pekne aj po zväčšení. Okrem toho podporuje aj iné formáty, napríklad MathML. Po kliknutí na renderovaný výraz sa otvorí menu s niekoľkými možnosťami na zmenu nastavení a exportovanie v niektorom z podporovaných formátov.

### 2.3.6 KaTeX

KaTeX, podobne ako MathJax, slúži na zobrazenie výrazov vo formáte TeX. Renderované výrazy sú tiež vektorové. Podľa jej webovej stránky [10] je rýchlejšia ako MathJax, aj vďaka tomu, že negeneruje množstvo elementov navyše pre možnosť nastavovania a exportovania výrazu. Je tiež jednoduchšia na použitie. Túto knižnicu používam v programe na renderovanie matematických výrazov v TeX.

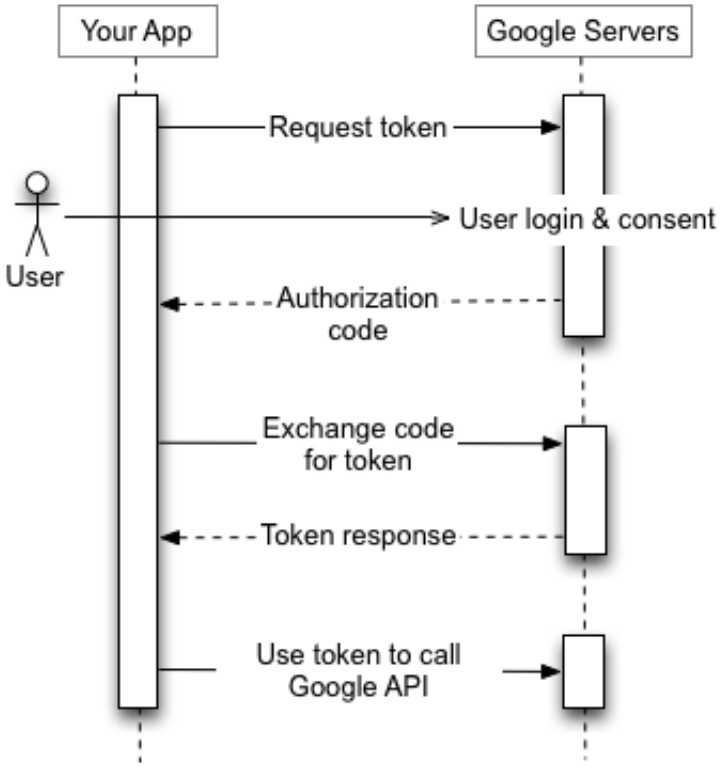
### 2.3.7 CodeIgniter

CodeIgniter je MVC (Model-view-controller) framework v jazyku PHP, ktorý budem používať na štruktúrovanie server-side časti aplikácie. Je jeden z jednoduchších PHP frameworkov na použitie. Okrem MVC štruktúry ponúka aj mnoho užitočných nadstavieb PHP, napríklad knižnicu Active Record na zjednodušenie prístupu k databáze.

### 2.3.8 Google OAuth 2.0

Google OAuth 2.0 je protokol na autentifikáciu a autorizáciu používateľa pre prístup ku Google API [11]. Proces autentifikácie prebieha tak, že klientska aplikácia presmeruje používateľa na Google URL, ktoré obsahuje identifikačné parametre aplikácie. Tie je potrebné pre aplikáciu získať predtým manuálnym zaregistrovaním sa cez Google Developers Console. Google spracuje autentifikáciu a súhlas používateľa so zdieľaním dát aplikácii. Potom odošle autorizačný kód, ktorý aplikácia môže vymeniť za access token a refresh token. Access token používa na komunikáciu s Google API a refresh token môže použiť, keď access token expiruje, aby získala nový. Tento proces je znázornený na obr. 2.3.

Na prácu s týmto API použijem knižnicu Google APIs Client Library for PHP.




**Obr. 2.3:** Priebeh protokolu OAuth 2.0 [11]

### 2.3.9 Bootstrap

Bootstrap je rozšírený front-end framework, ktorý obsahuje základnú CSS šablónu a niekoľko CSS a JavaScriptových komponentov. V aplikácii použijem z CSS komponentov napríklad navigačné panely a skupiny tlačidiel. Z Javascriptových komponentov použijem Modal okná, čiže okná, ktoré sa zobrazujú priamo vo webstránke, Popover, teda bublinu s HTML obsahom, ktorá sa otvorí napríklad po stlačení tlačidla a Dropdown menu.

### 2.3.10 Téma Paper pre Bootstrap

Ako vizuálnu tému som zvolila Paper pre Bootstrap. Má jednoduchý a moderný dizajn, ktorý vychádza z princípov materiálneho dizajnu. Tie propaguje najmä spoločnosť Google. V materiálnom dizajne má byť interakcia s aplikáciou podobná ako s materiálmi v reálnom svete, pričom jednotlivé prvky aplikácie pripomínajú hárky papiera položené na sebe (a vrhajú tiene podľa zdanlivej výšky). Farby v materiálnom dizajne sú často výrazné, ale musia byť starostlivo zvolené, aby vytvárali význam a hierarchiu elementov v aplikácii [12].

# 3. Návrh

## 3.1 Požiadavky na aplikáciu

Aplikácia by mala umožniť stredoškolákom, vysokoškolákom a matematickým nadšencom precvičiť si dokazovanie matematických viet zo sady axióm. Malo by byť možné takto dokázať sériu viet z Boolovej algebry. Program by im pri tom mal napomáhať návrhom možností a zároveň kontrolovať správnosť krokov. Dôraz má byť kladený na interaktivitu a názornú vizualizáciu procesu riešenia úlohy, vďaka čomu používateľ získa kvalitatívne inú skúsenosť ako pri riešení úlohy na papieri. Hoci sa táto práca zameriava na úlohy Boolovej algebry, cieľom je navrhnúť ju tak, aby bola ľahko rozšíriteľná aj na iné problémové domény pri zachovaní rovnakých princípov používateľského rozhrania aj vnútornej modulárnej architektúry.

Aplikácia by mala byť dostupná cez webový prehliadač a spĺňať nasledujúce konkrétne požiadavky na funkcionalitu.

### 3.1.1 Rozhranie na riešenie úloh

V rozhraní na riešenie úloh, ktoré je jadrom aplikácie, by mali byť pre používateľa dostupné nasledovné možnosti:

- Zobrazenie zadania aktuálnej úlohy vo forme matematického vzorca. Obe strany rovnice oddelené a umiestnené v časti obrazovky, kde prebieha úprava výrazov
- Zoznam ďalších úloh z toho istého modulu so zvýraznenou aktuálnou úlohou a vizuálne odlíšenými úlohami, ktoré už sú vyriešené. Po kliknutí na úlohu sa tá stane aktuálnou
- Zobraziť krátky interaktívny tutoriál, ktorý prevedie používateľa riešením vzorovej vety
- Zoznam axióm a odvodených viet (tautológií). Odvodené vety majú byť vizuálne rozdelené na dočasné (vytvorené počas riešenia aktuálnej úlohy) a trvalé (vety dokázené v predchádzajúcich úlohách)

- Použiť tautológie na vykonanie ekvivalentnej úpravy výrazu z aktuálnej úlohy. Po zvolení strany výrazu vygenerovať možnosti jeho ekvivalentnej úpravy a umožniť používateľovi výber. Potom pridať k zvolenému výrazu vybranú ekvivalentnú úpravu tak, aby boli vizuálne prepojené (napr. použitie axiómy (8) $a \wedge 1 = a$ na nahradenie podformuly vo výraze $x \vee (x \wedge 1)$, používateľ môže vybrať ekvivalentný výraz $x \vee x$)
- Odvodiť novú vetu z tautológie nahradením krátkych výrazov za niektoré jej premenné. Krátky výraz by mal pozostávať z novej premennej (alebo jej negácie či konštanty), prípadne aj logickej spojky a ďalšej premennej (alebo negácie). Napr. odvodenie novej vety z axiómy (8) $a \wedge 1 = a$ substitúciou $a/(a \vee a)$. Výsledná veta je $(a \vee a) \wedge 1 = a \vee a$
- Zmazať posledný krok, navrátiť zmazaný krok a začať úlohu od začiatku
- Odstrániť dočasnú odvodenú vetu

### 3.1.2 Prihlasovanie a ďalšie funkcionality

Okrem hlavnej aplikácie by mali byť dostupné aj pomocné funkcionality:

- Zobraziť zoznam modulov a úloh
- Prihlásiť sa prostredníctvom Google účtu
- Uložiť prihlásenému používateľovi stav aplikácie: vyriešené úlohy, postupy riešení, odvodené vety. Po prihlásení zachovať stav riešenia
- Umožniť prihlásenému používateľovi (učiteľovi) vytvoriť používateľskú skupinu a stať sa jej administrátorom. Nastaví jej školu, ročník, triedu a krúžok. Do skupiny sa môžu pridať ďalší prihlásení používatelia (žiaci) a učiteľ si môže pozrieť ich výsledky
- Používateľ môže vyhľadávať medzi skupinami a pridať sa do jednej alebo viacerých z nich. Tiež môže byť administrátorom viacerých skupín
- Zmeniť si meno a priezvisko, ktoré sa zobrazuje ostatným používateľom v skupinách

## 3.2 Analýza návrhu riešenia

Z dôvodu rozsahu práce som sa rozhodla postaviť tento program na čo najjednoduchších priamych, lineárnych dôkazoch. Na nich sa študent môže naučiť základné pravidlá v matematickej logike - substitúciu a nahradenie ekvivalentných podformúl.

Zvolila som sústavu axióm a viet z Boolovej algebry. Postup riešenia príkladu pomocou aplikácie je podrobnejší, ako sa ilustruje v [1], pretože je potrebné vykonať niektoré úpravy výrazu na viac krokov - napríklad nahradenie premennej v axióme výrazom, a potom vykonanie ekvivalentnej úpravy pomocou takto odvodenej vety. Tento postup ukážem na príklade v kapitole 3.2.1.

Riešenie úloh je pomerne voľné, čiže používateľ môže robiť ľubovoľné ekvivalentné úpravy (pokiaľ sú platné v sústave axióm), a môže dospieť k riešeniu vlastnou cestou. Takýchto ciest k riešeniu môže byť aj viac. Napríklad je možné upravovať rovnicu z ľavej alebo pravej strany, či vyskúšať rôzne úpravy. To vedie k tomu, aby používateľ v súlade s princípmi konštruktivizmu skúšal svoje vlastné nápady a plánoval dopredu. Tým si môže precvičiť matematické a logické myslenie, na rozdiel od niektorých iných typov úloh, kde je vopred daný spôsob, akým nájsť správny výsledok.

Pre zvýraznenie rozdielu medzi tautológiami a dokazovanými výrazmi používam v tautológiách premenné $a, b, c, \ldots$ a v dokazovaných výrazoch premenné $x, y, z, \ldots$. Na reprezentáciu operátorov namiesto znakov $+, \cdot, '$ používaných v [1] použijem $\vee, \wedge, \neg$, pretože by mohli byť viac známe žiakom stredných škôl, ktorí poznajú výrokovú logiku. Pôvodný zápis Boolovej algebry používa znaky ako + a ·, ale v inom význame, než sa učí na stredných školách.

### 3.2.1 Príklady použité v programe

#### 3.2.1.1 Logika – vety

Úlohy, ktoré má v tomto module používateľ dokázať, sú niektoré dôležité vety o prvkoch Boolovej algebry. Prvou z nich je voliteľný tutoriál, ktorý novému používateľovi predvedie ovládanie programu. Ten je ilustrovaný na jednoduchšej, pre tento účel vytvorenej vete, a slúži iba na to, aby používateľovi vysvetlil ekvivalentné úpravy a odvodzovanie z axióm. Pre dôkaz viet v tejto sekcii sú použité axiómy a ich číslovanie z kapitoly 2.1.5.

**Tutoriál** Ako tutoriál, teda príklad na predvedenie ovládania programu, som vytvorila vlastný jednoduchý príklad: $x \vee (x \wedge 1) = (x \vee x) \wedge 1$. Je potrebné dokázať, že rovnosť platí. Na ňom sú ilustrované obe základné funkcie programu: použitie axiómy na úpravu výrazu (nahradenie ekvivalentnej podformuly vo výraze) a odvodenie vety s jej následným použitím na úpravu výrazu.

Dôkaz by mohol vyzerať napríklad takto: $x \vee (x \wedge 1) \overset{(8)}{=} x \vee x \overset{(8)}{=} (x \vee x) \wedge 1$. V aplikácii prebieha riešenie nasledovným spôsobom:

1. Použijeme axiómu (8) $a \wedge 1 = a$ na nahradenie podformuly $x \wedge 1$ za $x$ v ľavej strane dokazovanej vety, teda $x \vee (x \wedge 1)$. Z vygenerovaných ekvivalentných výrazov, ktoré sa nám zobrazia, keď použijeme danú axiómu na úpravu ľavej strany, vyberieme $x \vee x$. Na ľavej strane je teraz rovnosť $x \vee (x \wedge 1) = x \vee x$.

2. Odvodíme z tej istej axiómy (8) $a \wedge 1 = a$ novú vetu substitúciou $a/(a \vee a)$. Vznikne nám nová veta (tautológia) $(a \vee a) \wedge 1 = a \vee a$. Použijeme ju na ekvivalentnú úpravu ľavej strany, kde je aktuálne výraz $x \vee x$, ktorý sme získali v predchádzajúcom kroku. Z toho nám vznikne výraz $(x \vee x) \wedge 1$, ktorý sa pridá na ľavú stranu. Ten sa rovná pravej strane dokazovanej rovnice, príklad je teda vyriešený.

Ďalej nasleduje séria deviatich úloh – viet z knihy ŠMM.

**2. úloha** $x \vee x = x$
Dôkaz [1]: $x \vee x \overset{(8)}{=} (x \vee x) \wedge 1 \overset{(9)}{=} (x \vee x) \wedge (x \vee \neg x) \overset{(6)}{=} x \vee (x \wedge \neg x) \overset{(10)}{=} x \vee 0 \overset{(7)}{=} x$.

V aplikácii je potrebné pre riešenie tejto úlohy zľava doprava postupovať napríklad týmto spôsobom:

1. Odvodíme z axiómy (8) $a \wedge 1 = a$ novú vetu substitúciou $a/(a \vee a)$ rovnako ako v tutoriáli. Výslednú vetu (tautológiu) $(a \vee a) \wedge 1 = a \vee a$ použijeme na ekvivalentnú úpravu ľavej strany $x \vee x$. Na ľavú stranu sa pridá ekvivalentný výraz $(x \vee x) \wedge 1$.

2. Použijeme axiómu (9) $a \vee (\neg a) = 1$ na nahradenie 1 vo výraze $(x \vee x) \wedge 1$ za $x \vee (\neg x)$. Na ľavú stranu sa pridá $(x \vee x) \wedge (x \vee (\neg x))$

3. Odvodíme z axiómy (6) $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$ novú vetu substitúciou $c/(\neg c)$. Výslednú vetu $a \vee (b \wedge (\neg c)) = (a \vee b) \wedge (a \vee (\neg c))$ použijeme na ekvivalentnú úpravu výrazu $(x \vee x) \wedge (x \vee (\neg x))$ na $x \vee (x \wedge (\neg x))$.

4. Použijeme axiómu (10) $a \wedge (\neg a) = 0$ na úpravu výrazu $x \vee (x \wedge (\neg x))$ na $x \vee 0$.

5. Použijeme axiómu (7) $a \vee 0 = a$ na úpravu výrazu $x \vee 0$ na $x$. To sa rovná pravej strane dokazovanej rovnice, príklad je teda vyriešený.

**3. úloha** $x \wedge x = x$

Dôkaz je podobný ako pri predchádzajúcej úlohe.

**4. úloha** $x \wedge 0 = 0$

Dôkaz [1]: $x \wedge 0 \overset{(7)}{=} (x \wedge 0) \vee 0 \overset{(10)}{=} (x \wedge 0) \vee (x \wedge \neg x) \overset{(5)}{=} x \wedge (0 \vee \neg x) \overset{(7)}{=} x \wedge \neg x \overset{(10)}{=} 0$.

**5. úloha** $x \vee 1 = 1$

Dôkaz je podobný ako pri predchádzajúcej úlohe.

**6. úloha** $x \vee (x \wedge y) = x$

Dôkaz [1]: $x \vee (x \wedge y) \overset{(8)}{=} (x \wedge 1) \vee (x \wedge y) \overset{(5)}{=} x \wedge (1 \vee y) \overset{5.\text{úloha}}{=} x \wedge 1 \overset{(8)}{=} x$.

**7. úloha** $x \wedge (x \vee y) = x$

Dôkaz je podobný ako pri predchádzajúcej úlohe.

**8. úloha** $x \vee (\neg x \wedge y) = x \vee y$

Dôkaz [1]: $x \vee (\neg x \wedge y) \overset{(6)}{=} (x \vee \neg x) \wedge (x \vee y) \overset{(9)}{=} 1 \wedge (x \vee y) \overset{8)}{=} x \vee y$.

**9. úloha** $x \wedge (\neg x \vee y) = x \wedge y$

Dôkaz je podobný ako pri predchádzajúcej úlohe.

**10. úloha** $\neg(\neg x) = x$

Túto úlohu, hoci je najkratšia, je najnáročnejšie dokázať iba pomocou ekvivalentných úprav. Dôkaz: $\neg(\neg x) \overset{(7)}{=} \neg(\neg x) \vee 0 \overset{(10)}{=} \neg(\neg x) \vee (x \wedge \neg x) \overset{2.\text{úloha}}{=} (\neg(\neg x) \vee x) \wedge (\neg(\neg x) \vee \neg x) \overset{(9)}{=} (\neg(\neg x) \vee x) \wedge 1 \overset{(9)}{=} (\neg(\neg x) \vee x) \wedge (x \vee \neg x) \overset{(6)}{=} x \vee (\neg(\neg x) \wedge \neg x) \overset{(10)}{=} x \vee 0 \overset{(7)}{=} x$

Medzi základné vety Boolovej algebry patria aj De Morganove pravidlá. Pre tie sa nám však nepodarilo nájsť dôkaz iba pomocou ekvivalentných úprav, preto ich nebolo možné použiť ako úlohy v aplikácii.

### 3.2.2 Logika – príklady

Modul Logika – príklady obsahuje ďalšie vety a príklady z logiky. Okrem axióm sú v paneli s tautológiami dostupné aj vety, ktoré sa odvodzujú v sekcii Logika – vety a tiež De Morganove pravidlá. Riešenie týchto príkladov je pomerne priamočiare.

**1. úloha** $(x \vee y) \wedge (z \vee v) = ((x \wedge z) \vee (y \wedge z)) \vee ((x \wedge v) \vee (y \wedge v))$

**2. úloha** $x \wedge ((y \vee z) \vee v) = ((x \wedge y) \vee (x \wedge z)) \vee (x \wedge v)$

**3. úloha** $x \vee ((y \wedge z) \wedge v) = ((x \vee y) \wedge (x \vee z)) \wedge (x \vee v)$

**4. úloha** $\neg((x \vee y) \vee z) = ((\neg x) \wedge (\neg y)) \wedge (\neg z)$

**5. úloha** $\neg((x \wedge y) \wedge z) = ((\neg x) \vee (\neg y)) \vee (\neg z)$

Tieto príklady mali niektoré časti v pôvodnom zadaní bez zátvoriek, napr. $y \wedge z \wedge v$. Bolo však potrebné pridať zátvorky: $(y \vee z) \vee v$, pretože program dokáže pracovať iba s operátormi, ktoré majú maximálne dva operandy.

## 3.3 Výber technológií

Keďže ide o webovú aplikáciu, rozhodla som sa použiť najpoužívanejšie technológie na vývoj takýchto aplikácií - PHP, MySQL, JavaScript, HTML, CSS.

Samotná aplikácia na riešenie úloh je vytvorená prevažne na strane prehliadača, v JavaScripte. Ostatná funkcionalita je naprogramovaná na strane servera s pomocou MVC frameworku CodeIgniter.

### 3.3.1 Client-side technológie

Na zjednodušenie ovládania používateľskej interakcie používam knižnicu jQuery. Pomocou nej naväzujem funkcie na udalosti ako kliknutie myšou na tlačidlo. Tiež uľahčuje výber a prechádzanie elementov v DOM.

Pre zobrazenie matematických rovníc, ktoré sú vo formáte TeX, používam knižnicu KaTeX. Táto knižnica je veľmi jednoduchá na použitie a renderuje výrazy rýchlo. Okrem nej som vyskúšala ešte MathJax, no ten je oveľa objemnejší a generuje okrem renderovaného výrazu aj množstvo ďalších elementov, ktoré nie sú v tejto aplikácii potrebné.

Na implementovanie funkcionality ťahania elementov a vlastný autocomplete pri input prvku používam knižnicu jQueryUI.

Mnoho funkcií spojených s používateľským rozhraním využívam aj z JavaScriptovej knižnice Bootstrapu - napríklad Modal okná, Popovery, čiže informačné bubliny, ktoré sa zobrazia pri elemente napríklad po kliknutí naň, Dropdown menu a ďalšie. Bootstrap je tiež framework, na ktorom je postavená CSS štruktúra stránky. Ako vizuálnu tému som zvolila Paper pre Bootstrap.

### 3.3.2 Server-side technológie

Na strane servera som zvolila MVC framework CodeIgniter, pretože je jednoduchý na použitie a rýchly. Postačuje pre množstvo aplikačnej logiky na strane servera.

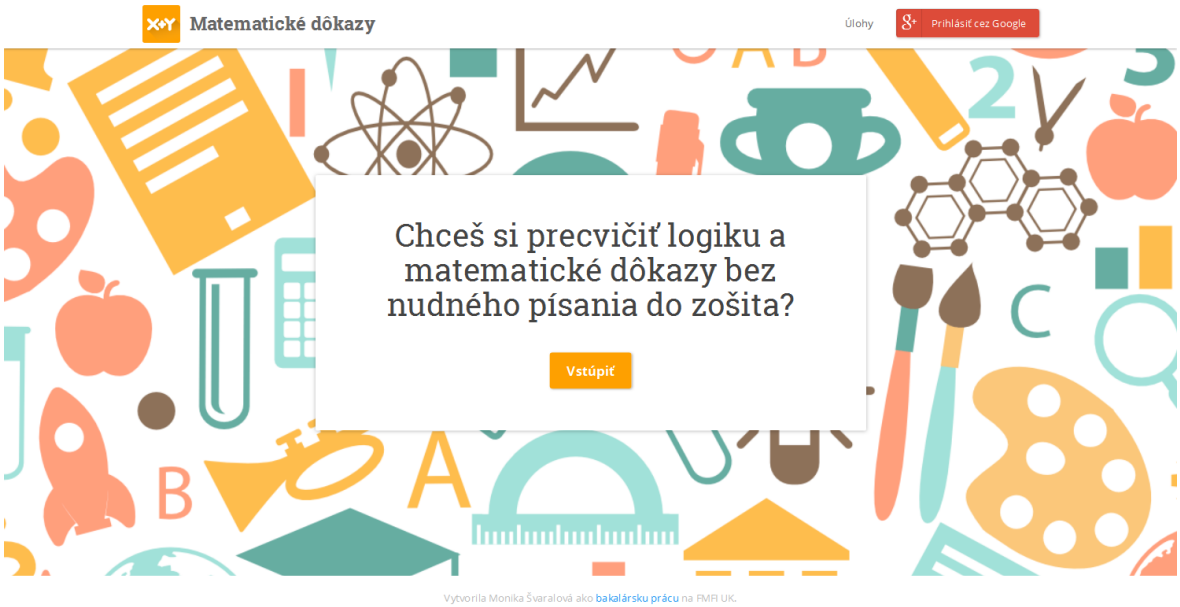
Na prihlasovanie používateľa cez Google používam Google API Client pre PHP. Pre prihlasovanie cez Google namiesto vytvorenia vlastnej registrácie na stránke som sa rozhodla preto, že potreba registrovať sa by mohla odradiť používateľov. Ideálne by bolo pridať ešte možnosť prihlásenia cez Facebook, a tiež možnosť registrácie pre tých, čo nemajú alebo nechcú použiť niektorý z týchto dvoch účtov. Nakoniec však pre jednoduchosť a rozsah aplikácie implementujem iba prihlasovanie cez Google.

## 3.4 Návrh používateľského rozhrania

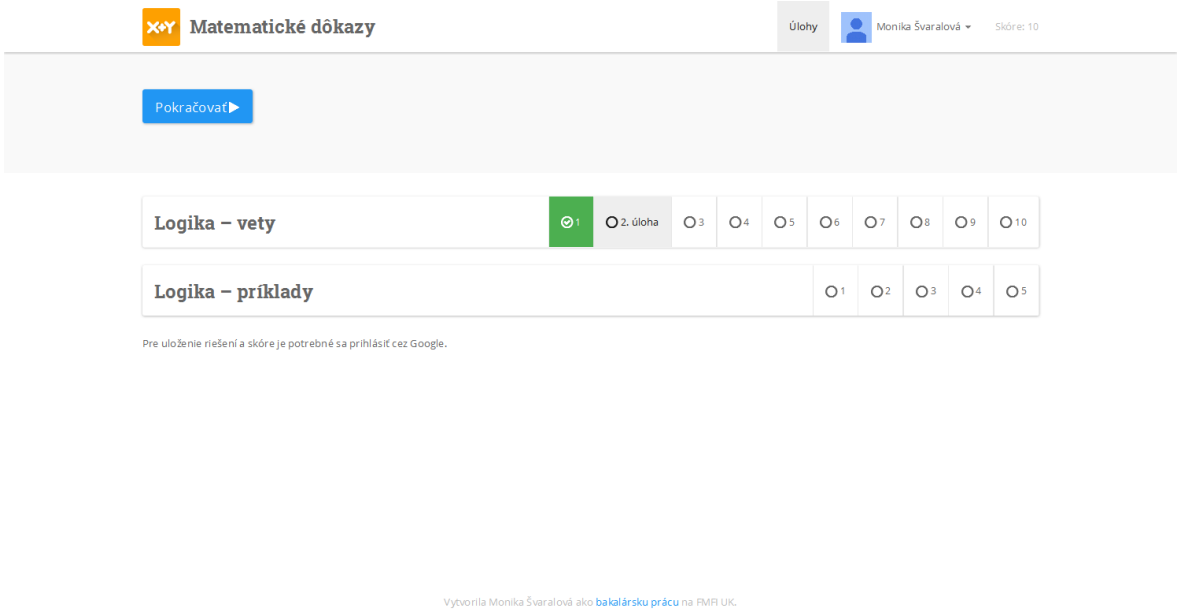
Používateľské rozhranie má byť navrhnuté tak, aby bola aplikácia čo najjednoduchšie použiteľná, aj keď používateľ nie je expertom na matematickú logiku. Malo by byť jasné, aké interakcie môže používateľ robiť v danom stave, k čomu má pomôcť aj krátky tutoriál. Cieľom je tiež navrhnúť minimalistické vizuálne rozhranie s čistým dizajnom, teda zbaviť sa zbytočných prvkov, ktoré by mohli používateľa zmiasť.

Po načítaní základného URL aplikácie sa zobrazí uvítacia stránka s krátkym neformálnym textom o aplikácií (Obr. 3.1). Hore sa na všetkých stránkach nachádza menu, kde je možné sa prihlásiť prostredníctvom Google a vrátiť sa na zoznam úloh. Prihlásený používateľ má na výber možnosť zobraziť si stránku používateľských skupín pomocou otváracieho menu vpravo hore.

Po kliknutí na tlačidlo Vstúpiť sa zobrazí stránka so zoznamom modulov a úloh, ktoré sa v nich nachádzajú. Zvýraznené sú úlohy, ktoré už má používateľ vyriešené, a nachádza sa tam tlačidlo Pokračovať, ktoré ho pošle na ďalšiu nevyriešenú úlohu (Obr. 3.2).




**Obr. 3.1:** Úvodná stránka aplikácie




**Obr. 3.2:** Stránka so zoznamom úloh (prihlásený používateľ)

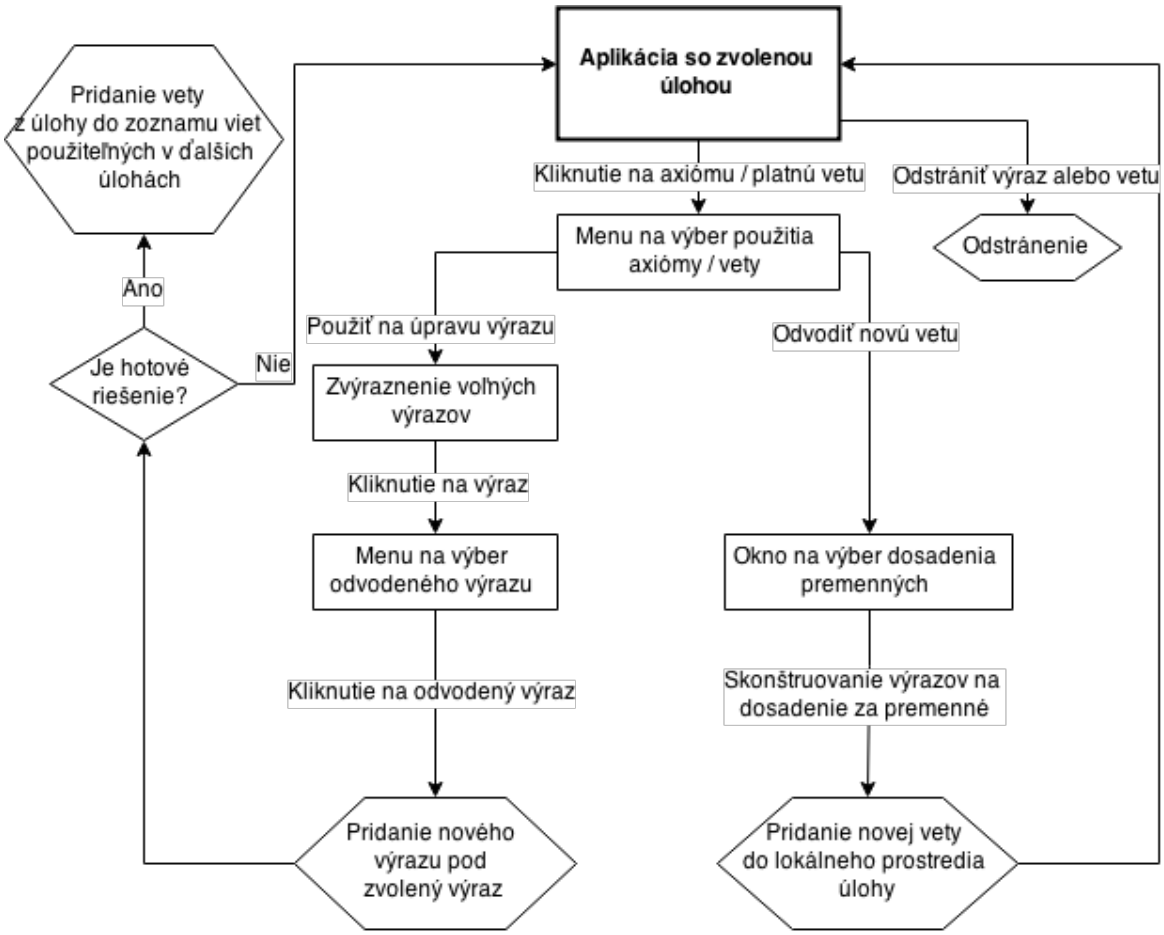

**Obr. 3.3:** Riešenie úlohy v aplikácii. Na ľavej strane sa nachádza dokazovaný výraz. Je rozdelený na ľavú a pravú stranu, každú je možné osobitne upravovať. V paneli napravo sa nachádzajú tautológie – axiómy a vety z nich odvodené. Používateľ klikol na jednu z axióm, má na výber dve možnosti.

Keď používateľ klikne na tlačidlo Pokračovať, otvorí sa mu samotná aplikácia s úlohou. Ak je používateľ prihlásený, úloha má uložený stav od predchádzajúceho razu, kedy ju riešil. Aplikácia počas riešenia úlohy je znázornená na obrázku 3.3.

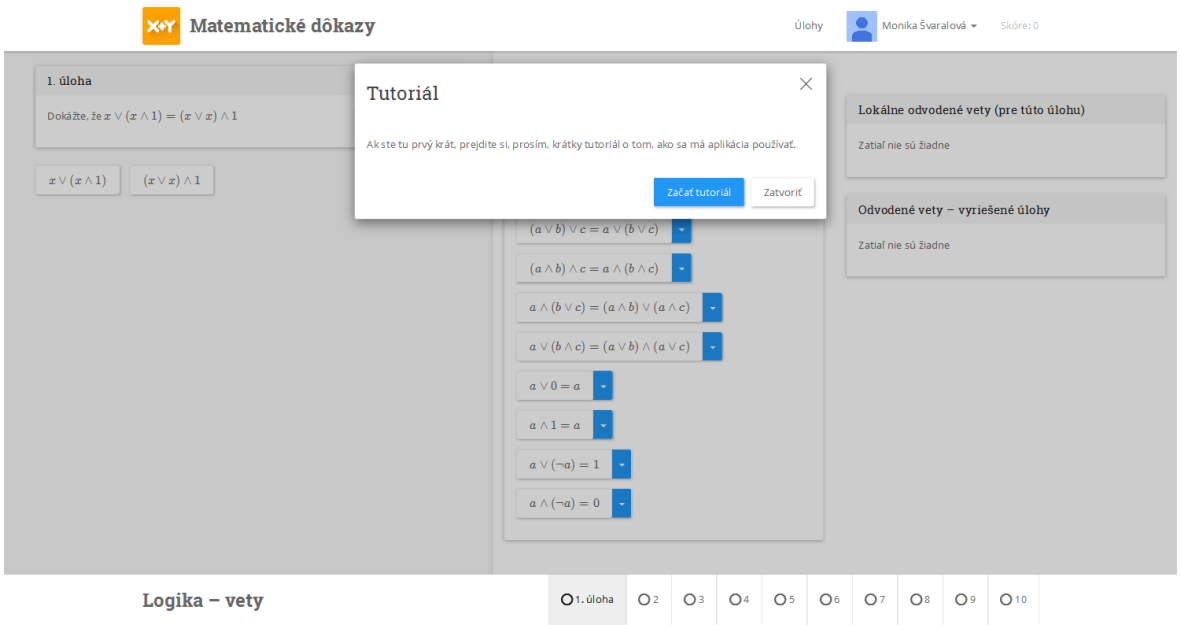
Aktivity, ktoré môže používateľ vykonať v aplikácii na riešenie úloh, je vidieť na obr. 3.4.

Pokiaľ je pre používateľa aplikácia nová, teda nemá vyriešenú 1. úlohu, zobrazí sa mu možnosť prejsť si tutoriál (Obr. 3.5). V tutoriáli sa automaticky prechádza dôkaz 1. úlohy a zobrazujú sa informačné bubliny o tom, čo sa práve deje. Používateľ po prečítaní informácie v bubline klikne na tlačidlo Ďalej a program vykoná ďalší predprogramovaný krok (Obr. 3.6). Počas tutoriálu nemôže používateľ sám klikať na aktívne komponenty, sú teda zosivené.

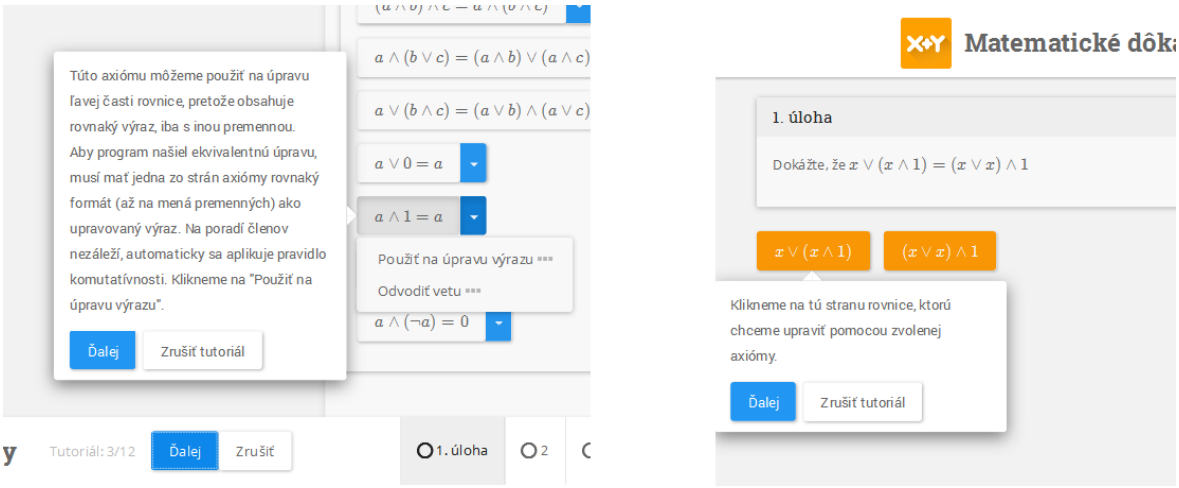
Po kliknutí na Skupiny v menu vpravo hore sa používateľovi zobrazí stránka s používateľskými skupinami (Obr. 3.7). Nachádza sa na nej zoznam skupín, v ktorých je používateľ členom alebo administrátorom. Je možné skupiny filtrovať podľa školy, pridať sa do niektorej z nich, alebo vytvoriť vlastnú.



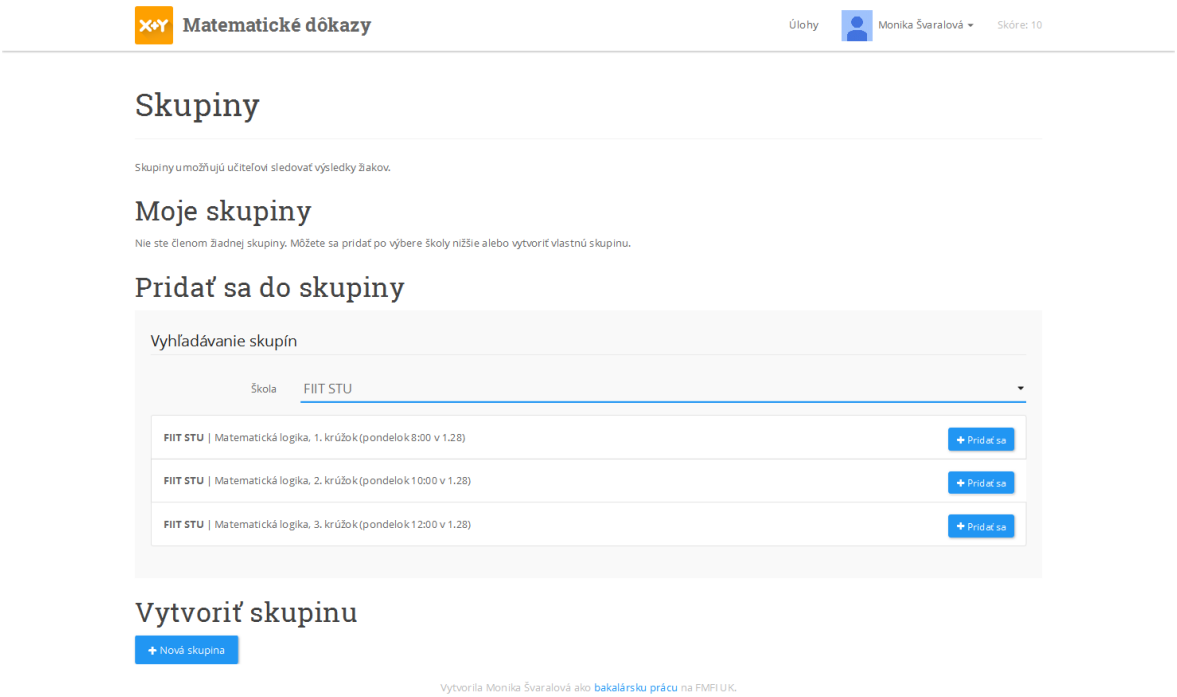

**Obr. 3.4:** User Interface Flow pre aplikáciu na riešenie úlohy



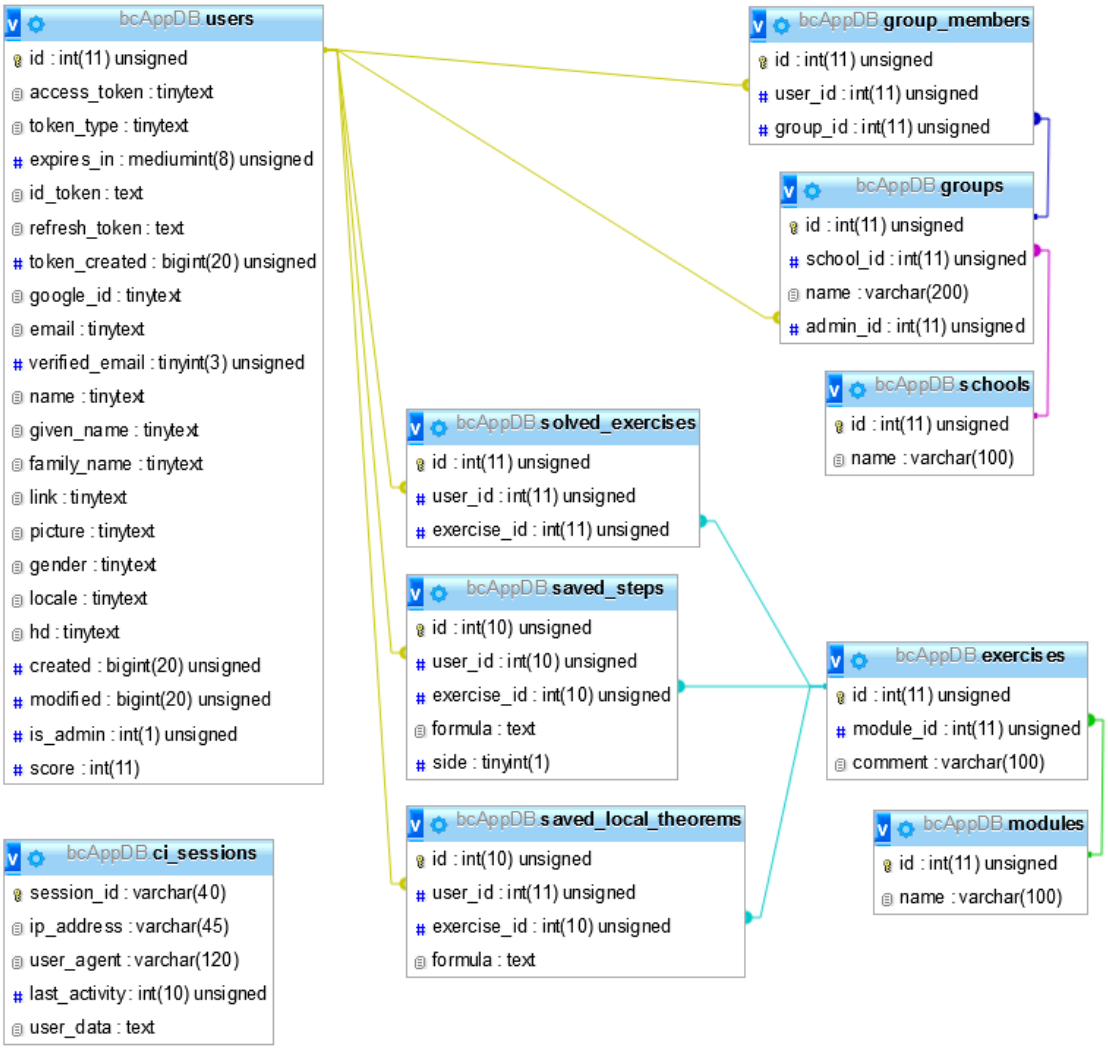

**Obr. 3.5:** Okno (Modal) s výzvou na začatie tutoriálu, ktoré sa zobrazí novému používateľovi




**Obr. 3.6:** Počas tutoriálu sa zobrazujú v aplikácii bubliny s informáciami. Používateľ nemôže klikať na komponenty v úlohe, iba posúvať sa dopredu tlačidlom Ďalej, riešenie prebieha automaticky.




**Obr. 3.7:** Stránka Skupiny




**Obr. 3.8:** Entitno-relačný diagram MySQL databázy

## 3.5 Dátový model

Na obrázku 3.8 je znázornený dátový model MySQL databázy v aplikácii.

### 3.5.1 Tabuľka users

V tejto tabuľke sa nachádza záznam o používateľovi, ktorý sa vloží, keď sa prvýkrát prihlási cez Google, a aktualizuje sa pri každom prihlásení. Obsahuje osobné údaje získané cez Google (celé meno, obrázok, pohlavie, jazykové prostredie) a prístupové tokeny. Dáta sú získavané cez protokol OAuth 2.0. Prihlásený používateľ sa vždy priradí k svojmu záznamu pomocou kľúča uloženého v Session. V tejto tabuľke sa nachádza aj stĺpec score, kde je uložené skóre používateľa v aplikácii.

### 3.5.2 Tabuľka ci_sessions

Túto tabuľku používa CodeIgniter na uloženie svojej Session. Tá sa na rozdiel od $_SESSION na serveri ukladá do cookies. V tejto tabuľke je uložená kópia týchto cookies, aby sa zistila ich prípadná modifikácia používateľom. Priradia sa podľa unikátneho session_id.

### 3.5.3 Tabuľka schools

Jedna z tabuliek potrebných pre modul spravovania používateľských skupín. Každá škola má unikátny záznam, aby bolo možné filtrovať skupiny priradené k tejto škole. Školy môže pridávať každý prihlásený používateľ, takže nie je kontrolovaná ich správnosť.

### 3.5.4 Tabuľka groups

Jedna z tabuliek potrebných pre modul spravovania používateľských skupín. Každá skupina má svoj záznam. Obsahuje cudzie kľúče do tabuliek schools a users. Skupiny môže pridávať každý prihlásený používateľ. Názov skupiny vytvorí vo vlastnom formáte, môže obsahovať napríklad ročník/triedu, predmet, krúžok.

### 3.5.5 Tabuľka group_members

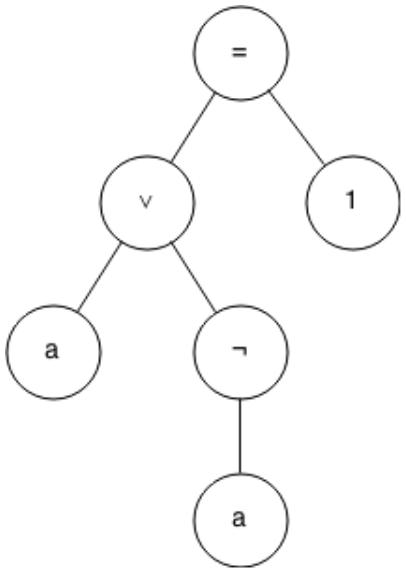
Jedna z tabuliek potrebných pre modul spravovania používateľských skupín. Slúži na zaznamenávanie členov skupín (s výnimkou admina). Jeden používateľ môže byť členom aj viacerých skupín. Vytvára reláciu many-to-many medzi tabuľkami users a groups. Obsahuje cudzie kľúče do týchto dvoch tabuliek.

### 3.5.6 Tabuľka modules

Jedna z tabuliek potrebných pre aplikáciu a zobrazovanie úloh. Slúži na zaznamenávanie modulov, teda sád úloh. Záznamy môže upravovať iba interný administrátor, ktorý má prístup k databáze.

### 3.5.7 Tabuľka exercises

Jedna z tabuliek potrebných pre aplikáciu a zobrazovanie úloh. Slúži na zaznamenávanie modulov, teda sád úloh. Záznamy môže upravovať iba interný administrátor, ktorý má prístup k databáze. Obsahuje cudzí kľúč, do tabuľky modules – id modulu, do ktorého patrí úloha.




**Obr. 3.9:** Reprezentácia výrazu $a \vee \neg a = 1$

### 3.5.8 Tabuľka solved_exercises

Zaznamenáva, ktoré úlohy používateľ vyriešil. Vytvára reláciu many-to-many medzi tabuľkami users a exercises a obsahuje cudzie kľúče do týchto dvoch tabuliek.

### 3.5.9 Tabuľka saved_steps

Zaznamenáva uložené kroky dôkazu pre prihláseného používateľa. Obsahuje cudzie kľúče do tabuliek users a exercises. Okrem toho obsahuje výraz kroku dôkazu vo formáte JSON (JavaScript Object Notation) a stranu rovnice, kde bol pridaný krok.

### 3.5.10 Tabuľka saved_derived_theorems

Zaznamenáva lokálne odvodené vety pre prihláseného používateľa. Obsahuje cudzie kľúče do tabuliek users a exercises. Okrem toho obsahuje rovnicu vo formáte JSON.

## 3.6 Návrh aplikačnej logiky

### 3.6.1 Reprezentácia

Výrazy a rovnice sú v aplikácii reprezentované ako stromy (expression trees). Na obrázku 3.9 je vzorový strom výrazu.

Ako implementáciu tohto stromu som zvolila jednoduchý objekt, aby ho bolo možné serializovať do databázy vo formáte JSON. Typ uzla je uložený v členskej premennej objektu.

Tento objekt je potom členskou premennou `content` triedy `Formula`. Poskytuje metódy na všetky operácie, ktoré sa v aplikácii vykonávajú s výrazmi. Každý výraz a rovnica v aplikácii je inštanciou tejto triedy.

Vyskúšala som aj knižnicu math.js, ktorá obsahuje reprezentáciu podobných typov uzlov – `OperatorNode`, `ConstantNode`, `SymbolNode` a niekoľko funkcií na prácu s nimi, ale nakoniec som sa rozhodla vytvoriť vlastnú reprezentáciu a implementáciu algoritmov, pretože knižnica neponúkala väčšinu z požadovaných procedúr aplikácie a nebola orientovaná na Boolovu algebru.

### 3.6.2 Client-side štruktúra aplikácie

Premenná `app` je jediná inštancia triedy `App`, ktorá ovláda aplikáciu. V nej sú zadefinované reakcie na používateľské interakcie.

V triede `Formula` sú implementované všetky algoritmy pre prácu s výrazom, ktorý je uložený v členskej premennej `content`. S výrazmi sa uskutočňujú hlavne dve komplexné operácie:

- Nájdenie všetkých možných nahradení podformuly s použitím danej axiómy
- Odvodenie novej vety substitúciou výrazov za premenné

### 3.6.3 Algoritmy

Nájdenie možných nahradení podvýrazov vo výraze za použitia axiómy prebieha nasledovným spôsobom (negenerujú sa úplne všetky možné nové výrazy, pretože tých by mohlo byť nekonečne veľa, iba tie s použitím pravidla nahradenia podformuly jeden raz a s obmedzeným počtom premenných):

- Najprv sa získa zoznam všetkých rôznych premenných vo výraze aj v axióme. Pri výraze zahrnieme do zoznamu aj konštanty, pretože chceme vygenerovať všetky substitúcie premenných z axiómy za premenné (resp. konštanty) z výrazu
- Nájdu sa všetky substitúcie z premenných v axióme na premenné z výrazu. Hľadáme všetky $k$-prvkové variácie s opakovaním z $n$ prvkov pričom $k$ je počet premenných v axióme a $n$ je počet premenných vo výraze. Počet takýchto variácií je $n^k$
- Vykoná sa každá z týchto substitúcií v axióme na kópii jej stromu

- Pre každú substituovanú axiómu sa zoberie ľavá strana. Výraz sa prehľadá, či niektorý jeho podstrom nie je rovnaký ako ľavá strana axiómy, vrátane rovnakých premenných. Z každého jeho uzla sa spustí nové hľadanie a porovnávanie. Ak sa nájde identický podstrom, nahradí sa kópiou pravej strany substituovanej axiómy
- To isté sa vykoná pre všetky pravé strany, výsledky sa uložia do jedného poľa. Potom sa odstránia duplikáty a pole sa vráti ako výsledok

Pravdepodobne existuje aj efektívnejší algoritmus, tento však rýchlosťou postačoval na riešenie úloh v aplikácii.

Pre odvodenie novej vety môže používateľ poskladať aj pre každú z premenných v axióme krátky výraz: novú premennú alebo jej negáciu, prípadne aj operátor a druhú premennú (alebo jej negáciu). To sa potom spracuje týmto spôsobom:

- Vytvorí sa kópia stromu axiómy
- Pre každú z premenných v axióme sa vygeneruje strom výrazu na jej substitúciu
- Prehľadá sa strom axiómy do šírky a ak sa nájde výskyt premennej, nahradí sa stromom výrazu na substitúciu. Takáto novovytvorená veta sa potom vráti

Okrem toho obsahuje táto trieda aj funkciu na generovanie uzátvorkovaného zápisu výrazu vo formáte TeX. Strom sa prechádza rekurzívne od koreňa spôsobom podobným ako inorder, pričom sa generuje zápis aj so zátvorkami. Výnimkou je unárny operátor, ktorý musí byť vypísaný pred svojim dcérskym uzlom.

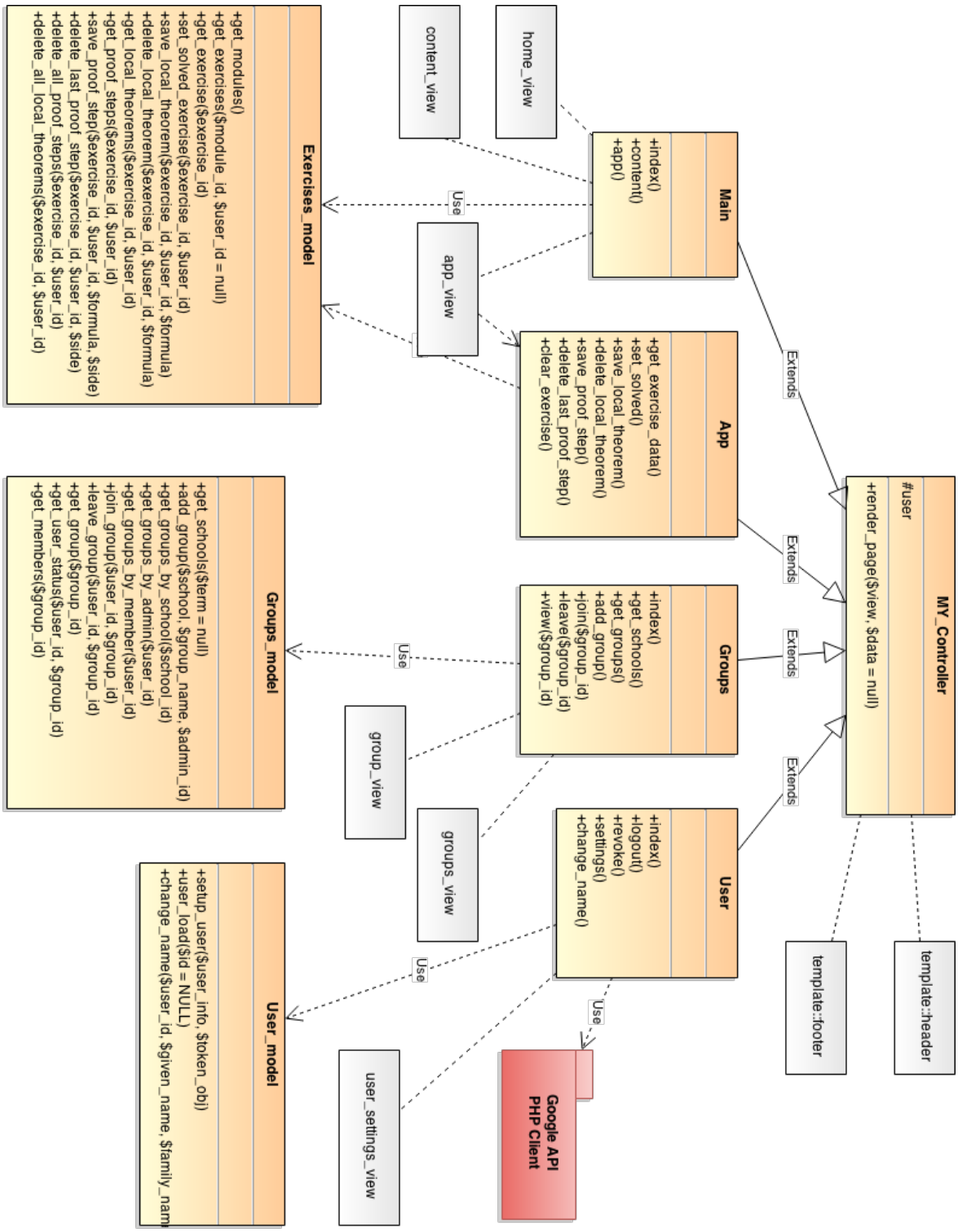
### 3.6.4 Server-side štruktúra aplikácie

Diagram štruktúry na strane servera je znázornený na obrázku 3.10.

#### 3.6.4.1 Controllery

**MY_Controller** Rodičovská trieda všetkých vlastných controllerov. Skontroluje sa v nej, či je používateľ prihlásený podľa záznamu v session a zavolá sa funkcia v modeli, ktorá ho skontroluje v databáze. Načíta sa záznam o používateľovi do členskej premennej, aby bol dostupný pre všetky ostatné controllery.

Okrem toho obsahuje funkciu `render_page()`, ktorá okrem načítania príslušného viewu pridá aj šablóny pre hlavičku a pätičku aplikácie a nastaví potrebné premenné, ktoré sa im majú odoslať.




**Obr. 3.10:** Triedny diagram MVC architektúry na strane servera

**Main** Slúži na zobrazenie úvodnej stránky, stránky so zoznamom úloh a stránky aplikácie. Načíta tiež zoznam modulov a úloh z modelu, ktoré je potrebné zobraziť na týchto stránkach.

**App** Obsahuje metódy, ktoré sa volajú cez AJAX aplikáciou počas riešenia úloh. V konštruktore sa preto kontroluje, či boli zavolané cez AJAX, ináč sa zobrazí chybová stránka:

```
if(!$this->input->is_ajax_request()){
        show_404();
}
```

Všetky z týchto metód okrem metódy `get_exercise_data()` tiež potrebujú, aby bol používateľ prihlásený, pretože ukladajú a načítavajú dáta z databázy, priradené konkrétnemu používateľovi. Pracujú s modelom Exercise_model.

Metóda `get_exercise_data()` slúži na načítanie informácií o úlohe (jej modul, zoznam ostatných úloh v tom istom module). Pre prihláseného používateľa načíta aj uložené dáta priradené tomuto používateľovi – lokálne odvodené vety a kroky riešenia, ktoré urobil.

Ďalej sú tam metódy na uloženie a zmazanie kroku riešenia a uloženie a zmazanie odvodenej vety, ktoré reagujú na tieto činnosti v aplikácii a volajú sa cez AJAX vždy po ich vykonaní. Ďalšia metóda slúži na vyčistenie cvičenia. Zmaže odvodené vety aj kroky riešenia. Posledná metóda nastaví úlohu ako vyriešenú. Funguje pre prihláseného aj neprihláseného používateľa.

**User** User Controller slúži na spravovanie používateľa: jeho prihlásenie, odhlásenie a upravenie nastavení (konkrétne zobrazovaného mena), ktoré má aj vlastný view. Metóda prihlasovania používa knižnicu Google API PHP Client a komunikuje s Google API.

**Groups** Slúži na spravovanie skupín. Hlavná stránka skupín, ktorej prislúcha aj view, obsahuje zobrazenie ich zoznamu, filtrovanie podľa školy (načítavajú sa cez AJAX) a možnosť vytvorenia novej skupiny. Vo formulári sa pomocou autocomplete dopĺňa názov školy použitím AJAX volania.

Ďalej obsahuje metódy na vytvorenie novej skupiny, pridanie a odobratie používateľa zo skupiny. Ďalší view, ktorý používa, je zobrazenie konkrétnej skupiny, kde sa načíta

zoznam členov a ich skóre (viditeľné iba pre administrátora). K nemu majú prístup iba členovia skupiny.

#### 3.6.4.2 Modely

Všetky modely pracujú s knižnicou Active Record, ktorá je vstavaná v CodeIgniteri a zjednodušuje vkladanie, získavanie a aktualizovanie dát v databáze. Poskytuje vlastné metódy a z nich vytvára SQL dotazy. Takýto prístup zvyšuje aj bezpečnosť aplikácie, pretože všetky premenné sú automaticky escape-ované.

**Exercises_model** Je používaný controllermi Main a App. Slúži na prácu s niekoľkými tabuľkami databázy, ktoré sa týkajú cvičení. Obsahuje metódy na získanie modulov, získanie zoznamu úloh, pričom sa zistí aj to, ktoré úlohy sú vyriešené (pre prihláseného používateľa z databázy, pre neprihláseného zo Session).

Ďalšie metódy v tomto modeli zodpovedajú metódam v App controlleri – načítanie uložených dát k úlohe, uloženie a zmazanie odvodenej vety či kroku riešenia, zmazanie uložených dát k úlohe a nastavenie úlohy ako vyriešenej.

**User_model** Obsahuje metódu na uloženie používateľa. Pomocou nej sa vloží nový používateľ do databázy, keď sa prvýkrát prihlási cez Google. Uložia sa jeho údaje získané cez Google rozhranie, spolu s tokenmi. Zavolá sa aj vždy, keď sa existujúci používateľ znovu prihlási. Vtedy sa jeho údaje aktualizujú na základe jeho unikátneho id, ktoré je tiež získané z Google. Takto sa pri každom prihlásení udržujú jeho údaje synchronizované s údajmi, ktoré má v Google účte.

Ďalšie metódy tohto modelu slúžia na načítanie záznamu o používateľovi z databázy na základe jeho id, a zmenu zobrazovaného mena používateľa.

**Groups_model** Slúži na prácu so skupinami. Využíva ho controller Groups. Obsahuje metódy na získanie zoznamu škôl, zoznamu skupín podľa školy, podľa administrátora a podľa člena a získanie záznamu jednej skupiny. Ďalej vytvorenie novej skupiny, pridanie a odobratie člena skupiny, získanie statusu používateľa v skupine (administrátor, člen, nečlen) a zoznamu členov skupiny.

# 4. Implementácia

V tejto kapitole uvediem niekoľko ukážok z kódu aplikácie a problémov, ktoré sa vyskytli.

## 4.1 Generovanie výrazu v TeX formáte

Generovanie zápisu výrazu v TeX formáte (ktorý sa potom renderuje pomocou knižnice KaTeX) je jednou z metód triedy Formula, ktorej inštanciami sú všetky výrazy v aplikácii. Táto metóda, tak ako všetky ostatné, je umiestnená na prototype triedy (podľa návrhového vzoru Prototype), aby sa pre každý objekt nevytvorila vlastná kópia metódy. Metóda `toTeX` má v sebe rekurzívne volanie lokálnej funkcie `toTexNode`, ktoré sa vykoná na každom uzle v poradí inorder. Táto funkcia vráti správne uzátvorkovaný výraz s TeX znakmi pre operátory $\vee, \wedge, \neg$ z uzla, ktorý dostane ako argument. Ďalším argumentom je, či má okolo aktuálneho výrazu pridať zátvorky. Tie sa pridávajú okolo každého výrazu okrem tých, ktoré sú v koreni stromu alebo sú strany rovnice.

Nasleduje okomentovaný zdrojový kód tejto funkcie.

```
Formula.prototype.toTeX = function() {

        var toTexNode = function(node, useParentheses) {
                var isEquation = (node.value != '='); // ak súčasný
                   vrchol obsahuje "=", jeho deti nemusia byť uzá
                   tvorkované

                var result = '';
                if (node.children && useParentheses)
                        result += ' (';
                if (node.children && node.children.length > 1) { // v
                   node je buď jeden z binárnych operátorov "and", "
                   or" alebo unárny operátor "not". V druhom prípade
                   vypíšeme znak operátora pred jeho prvým dieťaťom
                        result += toTexNode(node.children[0],
                           isEquation)
                }
```

```
                if (node.children) {
                        result += this.TeXmappings[node.value]; //
                           trieda Formula má ako členskú premennú
                           pole TeXmappings, ktoré obsahuje pre každý
                            zo zápisov operátorov "and", "or", "not"
                           (takto sú uložené v premennej value) jeho
                           zápis v TeX formáte
                } else {
                        result += node.value;
                }

                if (node.children) {
                        if (node.children.length > 1)
                                result += toTexNode(node.children[1],
                                   isEquation)
                        else
                                result += toTexNode(node.children[0],
                                   isEquation)
                }

                if (node.children && useParentheses)
                result += ') ';

                return result;
        }

        return toTexNode(this.content); // zavolá sa rekurzívna
           funkcia na koreni stromu výrazu, druhý argument
           useParentheses je automaticky false
}
```

## 4.2 Prihlasovanie cez Google

Metóda `index()` controllera User obsahuje nasledujúci kód na použitie knižnice Google API PHP Client a nasledovnú autentifikáciu používateľa cez Google API. Obsahuje sadu kľúčov vygenerovanú po registrácii aplikácie na Google Developers Console. Bolo tam tiež potrebné napísať presné URL, na ktoré sa pošle naspäť autorizačný kód. Toto registrované URL sa musí zhodovať s URL v kóde. Oba kroky protokolu – získanie autorizačného kódu aj jeho výmena za token (znázornené v sekcii 2.3.8) – sa dejú v

tejto jednej funkcii. Rozlišujú sa pomocou GET parametrov získaných z Google API.

Nasleduje skrátený zdrojový kód tejto funkcie.

```
require_once APPPATH . 'libraries/google-api-php-client/src/Google/
    autoload.php';
require_once APPPATH . 'libraries/google-api-php-client/src/Google/
    Client.php';
require_once APPPATH . 'libraries/google-api-php-client/src/Google/
    Service/Oauth2.php';

$client_id = '<<client id>>';
$client_secret = '<<client secret>>';
$simple_api_key = '<<simple api key>>';

$client = new Google_Client();
$client->setClientId($client_id);
$client->setClientSecret($client_secret);
$client->setDeveloperKey($simple_api_key);
$client->setRedirectUri('<<this page uri>>');
$client->setScopes(array('openid', 'email', 'profile'));
$client->setApprovalPrompt('auto');
$oauth2 = new Google_Service_Oauth2($client);

if ($this->input->get('code')) // či bol získaný autorizačný kód
{
        $client->authenticate($this->input->get('code'));
        $this->session->set_userdata(array('token' => $client->
            getAccessToken())); // token sa uloží do Session
        redirect('user'); // presmeruje sa na túto istú funkciu
}
if ($this->session->userdata('token'))
{
        $client->setAccessToken($this->session->userdata('token'));
}
if ($client->getAccessToken())
{
        if ($this->session->userdata('revoke')) // ak je prístup zruš
            ený
        {
                $client->revokeToken();
                $this->session->sess_destroy();
                redirect(base_url());
        }
        else
```

```
        {
                $token_details = json_decode($client->getAccessToken()
                    );
                $user = $oauth2->userinfo->get();
                $this->load->model('user_model');
                $user_id = $this->user_model->setup_user($user,
                    $token_details);

                // tu prebehne uloženie vyriešených úloh (ak nejaké
                    používatel vyriešil pred prihlásením) do databázy
                    a presmerovanie používatela na stránku, kde bol
                    predtým
        }
}
else
{
        $auth_url = $client->createAuthUrl();
        redirect($auth_url);
}
```

## 4.3 Bezpečnosť

Pri tvorbe aplikácie sa vyskytla otázka zabezpečenia aplikácie, hlavne na strane klienta. Jedným z problémov tvorby aplikácie v JavaScripte je, že kód aj stav programu je otvorený. Každý, kto vie používať konzolu v prehliadači, dokáže meniť premenné a volať funkcie podľa vlastného uváženia. Takto sa môže v tejto aplikácii vyskytnúť problém, že znalec takýchto metód pozmení premenné, alebo pošle volanie na server, že vyriešil úlohu, a to sa uloží do databázy, hoci nemusí byť vyriešená korektne alebo vôbec.

V produkčnej aplikácii s väčším bezpečnostným rizikom by bolo potrebné ošetriť posielané volania kompletnou validáciou na strane servera. V tejto aplikácii by to znamenalo posielanie všetkých krokov dôkazu a kontrolovanie ich správnosti na strane servera, na to by však musela byť na serveri implementovaná podobne pokročilá štruktúra na reprezentáciu výrazov ako na strane klienta. Prípadne by bolo možné presunúť celú štruktúru iba na stranu servera, pričom by sa pri každom kroku posielala AJAX požiadavka na server. Odvodzovanie výrazov by prebiehalo tam a vrátil by sa hotový výraz. Toto posielanie požiadaviek na server by mierne spomalilo odozvu aplikácie.

Ďalšou možnosťou je obfuskácia, teda znečitateľnenie zdrojového kódu v JavaScripte. Existuje niekoľko softvérov, ktoré to dokážu. To však nezaručuje úplnú bezpečnosť aplikácie, pretože používateľ môže stále modifikovať premenné, len by bolo pre neho náročnejšie identifikovať, ktoré čo reprezentujú.

# 5. Testovanie

Aplikáciu sme testovali na študentoch 1. ročníka FIIT STU, ktorí v tomto semestri absolvovali predmet Matematická logika. Na ňom sa venovali aj Boolovej algebre a pravidlám logiky použitým pri dôkazoch v tejto aplikácii.

Keďže nebola k dispozícii počítačová učebňa, prebiehalo testovanie iba na jednom počítači s použitím projektoru. Po slovnom úvode a tutoriáli bol vybraný jeden študent, ktorý pracoval s aplikáciou. Podarilo sa mu spoločne s pomocou ostatných vyriešiť štyri úlohy.

Aplikáciu mali študenti možnosť potom vyskúšať aj doma a vyplniť krátky anonymný dotazník s vlastným hodnotením a postrehmi. Do aplikácie sa prihlásilo cez Google 16 študentov, z toho 9 vyplnilo dotazník. Výsledky sú nasledovné:

**Ako sa Vám páčila aplikácia celkovo (od 1 do 10, 10 je najvyššie skóre)?** Priemer: **7,1**

**Zdala sa Vám aplikácia dostatočne prehľadná na použitie? (od 1 do 10, 10 je najvyššie skóre)?** Priemer: **7**

**Ako by ste hodnotili dizajn aplikácie? (od 1 do 10, 10 je najvyššie skóre)?** Priemer: **8,4**

**Bol tutoriál dostatočne zrozumiteľný? (od 1 do 10, 10 je najvyššie skóre)?** Priemer: **6,8**

**Koľko minút Vám trvalo priemerne vyriešenie jednej úlohy?**

- Do 5 minút: 0 odpovedí
- 5-10 minút: 5 odpovedí
- 10-15 minút: 2 odpovede
- 15-20 minút: 1 odpoveď

- Viac ako 20 minút: 1 odpoveď
- Úlohy som neriešil(a): 0 odpovedí

**Koľko úloh ste vyriešili (okrem tutoriálu) od 0 do 9?** Priemer: **7**

**Komentáre (nepovinná položka)**. Čo sa Vám páčilo alebo nepáčilo? Aké máte pripomienky k funkcionalite, použiteľnosti, či dizajnu? Mali ste ťažkosti s riešením úloh?

- *Grafika a celý dizajn toho prostredia je super. Chvíľu mi trvalo pokiaľ som pochopil, čo sa ako robí ale po tých prvých to už bolo v pohode. Ku koncu sa zjavili príklady, kde už bolo celkom treba odvodzovať a celkom mi trvalo pokiaľ som prišiel na to, že odvodené vzorce sa dajú ďalej odvodzovať. A všimol som si, že ak vyberiem použiť na úpravu výrazu a prekliknem sa na inú úlohu tak mám stále zvolené to použiť na úpravu výrazu ale nedá sa mi nič upravovať.*
- *Odvodenie viet kedy sa namiesto A dalo len negované A je podľa mňa zbytočne dlhé, ale je to v poriadku. Inak je to super grafika aj to funguje veľmi dobre, takže žiadne výhrady.*
- *Sem tam svojvoľne odhlási človeka a zmiznú mu vyriešené úlohy, pokým si to všimne. Odvodzovanie viet je zbytočne zložité a treba na to pár trikov.*
- *Veľmi ma otravuje klikať Zrušiť v okne 'Vyberte na ktorý výraz chcete aplikovať axiómu (button Zrušiť)' (je to fajn prvých pár krát ale potom to naozaj dosť vadí). A niekedy mi nehodí všetky možné spôsoby úpravy. Inak by to bolo aj fajn.*
- *Dizajn som hodnotil čo sa týka krasy nie funkčnosti pri funkčnosti by to bolo za 10*
- *Aplikácia sa mi zdá prehľadná, možno k tutoriálu by sa mohli pridať nejaké vysvietenia panelov a výrazov, aby bolo lepšie vidieť čo sa deje.*

Testovanie na študentoch ukázalo, že program dokážu po krátkom úvode a prezretí tutoriálu samostatne použiť. Mnohým z nich sa podarilo dokázať pomocou neho všetky úlohy z Boolovej algebry, čo je v porovnaní s predchádzajúcimi ročníkmi študentov, ktorí dostali rovnakú bodovanú prémiovú domácu úlohu, niekoľkonásobný nárast. Všetky aspekty aplikácie hodnotia prevažne pozitívne.

# 6. Záver

Cieľom práce bolo navrhnúť a implementovať interaktívnu webovú aplikáciu, v ktorej by študenti vysokých a stredných škôl mali možnosť precvičiť si dokazovanie viet z Booleovej algebry. Aplikácia kontroluje správnosť riešenia a vytvára určité obmedzenia, v ktorých sa môže študent pohybovať, avšak stále je to on, kto musí intenzívne rozmýšľať a hľadať riešenie. Takýmto spôsobom si môže trénovať schopnosť matematicky myslieť a orientovať sa v abstraktnej matematike.

V aplikácii je možné dokázať sériu štandardných viet (s výnimkou De Morganových pravidiel) z Booleovej algebry a ďalšie príklady použitím dvoch pravidiel logiky: pravidla substitúcie a nahradenia ekvivalentných podformúl.

Dôraz bol kladený aj na použiteľnosť a čistý dizajn. Je dôležité, aby nový používateľ za krátky čas pochopil, ako sa aplikácia používa. Pre tento účel som do nej pridala aj interaktívny tutoriál, ktorý predvedie jej použitie na vzorovej úlohe.

Testovanie na študentoch FIIT STU ukázalo, že dokážu v aplikácii samostatne vyriešiť aj všetky z úloh. Aplikáciu celkovo, jej použiteľnosť a dizajn ohodnotili pozitívne.

Aplikáciu je možné ďalej rozvíjať tak, aby bola použiteľná aj na iné typy úloh, nie len na matematické dôkazy. Jednou možnosťou, na ktorú by sa dalo zamerať, je hľadanie riešení rovníc. Na tie by bolo možné aplikovať matematické vety a pravidlá podobným spôsobom. Prípadne po rozšírení dokazovacích možností (aby nezahŕňali iba ekvivalentné úpravy) by ho bolo možné použiť aj na riešenie dôkazov z iných oblastí matematiky.

# Zdroje


[1] ODVÁRKO, O. *Booleova algebra.* Praha: ÚV matematické olympiády, Mladá Fronta, 1973. Škola mladých matematiků.

[2] *Škola mladých matematiků*, Dostupné na: http://www.karlin.mff.cuni.cz/katedry/kdm/literatura/skolamm.htm, [cit. 24.5.2015].

[3] THURSTON, W. P. "On proof and progress in mathematics", *Bull. Amer. Math. Soc.*, s. 161–177, Apr. 1994.

[4] KVASNIČKA, V. - POSPÍCHAL, J. *Matematická logika.* Bratislava: STU vydavateľstvo, 2007.

[5] *Mathway | math problem solver.* Dostupné na: https://mathway.com/, [cit. 13.2.2015].

[6] *Wolfram|alpha: computational knowledge engine.* Dostupné na: http://www.wolframalpha.com/, [cit. 13.2.2015].

[7] *Khan academy.* Dostupné na: https://www.khanacademy.org/, [cit. 13.2.2015].

[8] *Scratch - vymysli, programuj, zdieľaj.* Dostupné na: http://scratch.mit.edu/, [cit. 13.2.2015].

[9] ZAKAS, Nikolas C. *Professional JavaScript for Web Developers.* 3. vyd. Wrox, 2012.

[10] EISENBERG, E. - ALPERT, B. *The fastest math typesetting library for the web.* Dostupné na: http://khan.github.io/KaTeX/, [cit. 24.5.2015].

[11] *Using OAuth 2.0 to access Google APIs - Google Identity Platform - Google Developers.* [online]. Dostupné na: https://developers.google.com/identity/protocols/OAuth2, [cit. 24.5.2015].

[12] *Introduction - material design - Google design guidelines.* Dostupné na: http://www.google.com/design/spec/material-design/introduction.html, [cit. 24.5.2015].

# Prílohy

Priložené CD obsahuje:

- Zdrojové kódy aplikácie
- Bakalársku prácu vo formáte pdf

Aplikáciu je možné vyskúšať na adrese http://kempelen.ii.fmph.uniba.sk/logika/.