Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

DIPLOMOVÁ PRÁCE

Software pro podporu výuky algoritmizace a programování

2020

Bc. Veronika Vašinová

Vedoucí práce: Mgr. Tomáš Kühr, Ph.D.

Studijní obor: Aplikovaná informatika, prezenční forma

Bibliografické údaje

Autor:	Bc. Veronika Vašinová
Název práce:	Software pro podporu výuky algoritmizace a programování
Typ práce:	diplomová práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2020
Studijní obor:	Aplikovaná informatika, prezenční forma
Vedoucí práce:	Mgr. Tomáš Kühr, Ph.D.
Počet stran:	45
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliograhic info

Author:	Bc. Veronika Vašinová
Title:	Software for Teaching of Algorithmization and Programming
Thesis type:	master thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2020
Study field:	Applied Computer Science, full-time form
Supervisor:	Mgr. Tomáš Kühr, Ph.D.
Page count:	45
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

Tato diplomová práce se zabývá vytvořením nástroje pro podporu výuky algoritmizace a programování u mladších dětí. Vývojové prostředí, postavené na knihovně Blockly, umožňuje jednoduché skládání programů pomocí přetahování bloků, díky čemuž mohou uživatelé začít programovat prakticky okamžitě, bez nutnosti se zdlouhavě seznamovat se syntaxí některého z běžných programovacích jazyků. Součástí práce je také aplikace pro správu a plnění programovacích úloh.

Synopsis

This master's thesis deals with the creation of tool to support learning algorithms and programming for younger children. Development environment built on Blockly Library, allows easy stacking programs by dragging and dropping blocks, allowing users to start programming almost immediately, without getting to know the syntax of one of the common programming languages. The work also includes applications for managing and implementing the programming task.

Klíčová slova: algoritmizace; programováni; blockly; vuejs

Keywords: algorithmization; programming; blockly; vuejs

Děkuji vedoucímu diplomové práce, panu Mgr. Tomáši Kührovi, Ph.D za odborné vedení, konzultace a za věcné připomínky k práci.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam tabulek

Seznam vět

Seznam zdrojových kódů

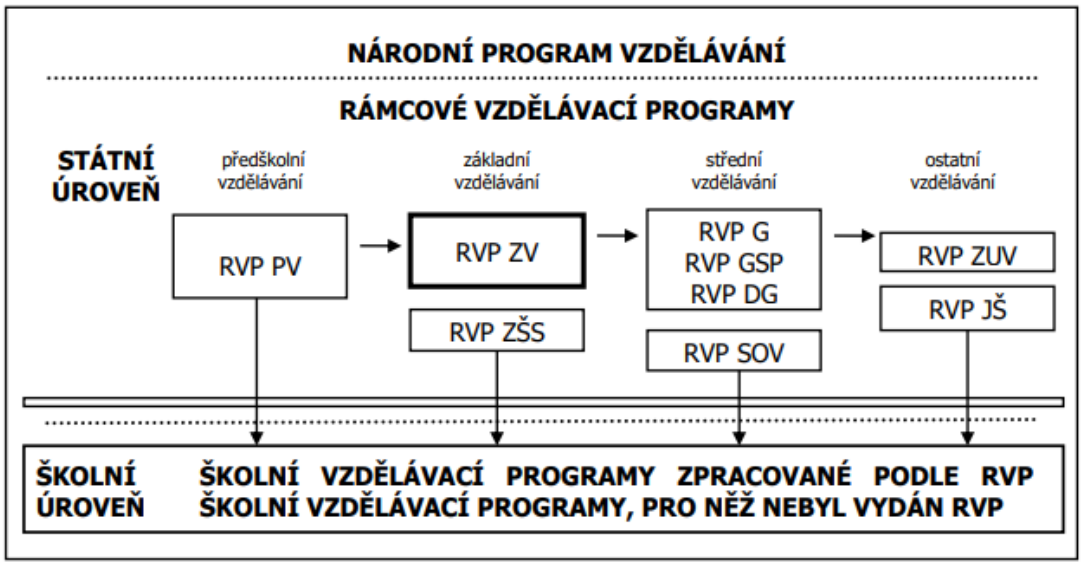
1 Výuka informatiky

Výuka informatiky ve školách by se měla v příštích letech výrazně proměnit. Programování se na českých školách objevuje spíše výjimečně. Učitelé dávají přednost spíše klasickým uživatelským tématům, jako je grafika, práce s kancelářskými programy nebo práce s internetem. Tento stav by měla změnit Strategie digitálního vzdělávání do roku 2020. [1]

1.1 Rámcové a školní vzdělávací programy

Pro každý obor v základním i ve středním vzdělávání je zpracován tzv. rámcový vzdělávací program (dále RVP). Díky němu je na státní úrovni garantováno, čemu se každý student v daném oboru naučí. Na základě RVP si každá škola vypracovává vlastní Školní vzdělávací program (dále ŠVP). [2]

ŠVP tvoří učitelé příslušné školy. Vytvářejí obsah vzdělávání pro dosažení cílů, které jsou obsaženy v RVP. Mohou tak program rozšířit podle svých cílů, uveďme příkladem rozšíření výuky určitých předmětů nebo detailnější zaměření k určitému oboru. ŠVP schvaluje ředitel školy a odpovídá za plnění příslušného RVP. Naplnění programu kontroluje Česká školní inspekce. ŠVP mívá velice často vlastní název. [2]



Obrázek 1: Systém kurikulárních dokumentů. Zdroj: [3]

Osnovy pro výuku informatiky na základní škole jsou zahrnuty v RVP Základní vzdělávání (RVP ZV) pod názvem Informační a komunikační technologie. Pro 2. stupeň obsahuje kurikulum pouze oblasti Vyhledávání informací a komunikace a Zpracování a využití informací, které navazují na učivo z 1. stupně studia. [3]

Z Tabulky 1 lze vyčíst, že současná výuka informatiky na základní škole je zaměřena pouze na uživatelský přístup k technologiím. Situace na většině středních odborných škol je podobná [4], výjimku tvoří gymnázia, kde se objevují základy algoritmizace přímo v RVP [5].

Vyhledávání informací a komunikace - očekávané výstupy	
žák	ověřuje věrohodnost informací a informačních zdrojů, posuzuje jejich závažnost a vzájemnou
učivo	vývojové trendy informačních technologií
	hodnota a relevance informací a informačních zdrojů, metody a nástroje jejich ověřování
	internet
Zpracování a využití informací – očekávané výstupy	
žák	ovládá práci s textovými a grafickými editory i tabulkovými editory a využívá vhodných aplikací
	uplatňuje základní estetická a typografická pravidla pro práci s textem a obrazem
	pracuje s informacemi v souladu se zákony o duševním vlastnictví
	používá informace z různých informačních zdrojů a vyhodnocuje jednoduché vztahy mezi údaji
	zpracuje a prezentuje na uživatelské úrovni informace v textové, grafické a multimediální formě
učivo	počítačová grafika, rastrové a vektorové programy
	tabulkový editor, vytváření tabulek, porovnávání dat, jednoduché vzorce
	prezentace informací (webové stránky, prezentační programy, multimédia)
	ochrana práv k duševnímu vlastnictví, copyright, informační etika

Tabulka 1: RVP ZV pro ICT - 2. stupeň. Zdroj: [3]

Učitelé mají možnost si ve svém ŠVP přizpůsobit obsah učiva informatických předmětů. Naprostá většina ovšem považuje za nejdůležitější vyučovat právě to, v čem se orientují nejlépe. Ve velké míře je to pouze seznámení s hardwarem, tvorba tabulek a prezentací, psaní v textovém editoru nebo komunikace přes internet. To souvisí se skutečností, že tyto předměty velice často vyučují učitelé s jinou aprobací. [6]

V západoevropských zemích je tomu právě naopak. Základy programování jsou zde zahrnuty do výuky a učivo se přizpůsobuje současným trendům. V tomto směru RVP zaostává. [1] Na tuto situaci zareagovalo MŠMT vydáním dokumentu Strategie digitálního vzdělávání do roku 2020.

1.2 Strategie digitálního vzdělávání do roku 2020

Vláda České republiky schválila v roce 2013 koncepci Digitální Česko v. 2.0: Cesta k digitální ekonomice, kde uvádí: *„Informační technologie by měly prostupovat celým procesem výuky na základních školách, nikoli jen v předmětech typu Práce s počítačem. Plné zapojení moderních technologií do výuky všech předmětů vnímá stát jako nezbytné v rámci posunu vzdělávacího systému od prostého memorování faktů k důrazu na čtenářskou gramotnost, komunikační dovednosti a logické myšlení“.* [7]

MŠMT v návaznosti na tuto koncepci reagovalo v roce 2014 vydáním dokumentu Strategie digitálního vzdělávání do roku 2020 (dále SDV). Strategie si klade za cíl [8]:

- otevřít vzdělávání novým metodám a způsobům učení prostřednictvím digitálních technologií
- zlepšit digitální gramotnost
- rozvíjet informatické myšlení žáků

K dosažení cílů strategie má dojít pomocí sady dílčích opatření. Ta kromě úpravy RVP zahrnují tvorbu učebních materiálů, vzdělávání učitelů, podporu sběru informací o dění ve školách, podporu výzkumu v didaktice informatiky a informační kampaň, která má vysvětlit smysl připravovaných změn. Ty mají být dokončeny právě do roku 2020. Tento termín byl daný již v okamžiku schválení. [8]

1.3 Informatické myšlení

Informatické myšlení (computational thinking) je téma, které se poslední roky skloňuje v souvislosti s výukou informatiky. Dle SDV je to schopnost formulovat problém, nacházet a zvažovat různé postupy řešení a ten nejefektivnější zobecnit pro podobné úkoly. Digitální technologie pracují s přesnými instrukcemi. Pro komunikaci s nimi je zapotřebí přesně vyjadřovat myšlenky a postupy ve formálním zápisu. Pracuje se především s obecnými základními pojmy jako je algoritmus, modelování nebo reprezentace informací, které nejsou závislé na současných technologiích. [8]

Konkrétnější a velmi používaná definice vychází ze spolupráce International Society for Technology in Education (ISTE) a Computer Science Teachers Association (CSTA) [9]:

- *formulovat problémy způsobem, který umožňuje jejich strojové řešení*

- *reprezentovat data prostřednictvím abstrakcí, jako jsou modely a simulace*
- *automatizovat řešení pomocí algoritmického myšlení (jako posloupnost kroků)*
- *odhalit, prozkoumat a provést možná řešení s cílem odhalit nejúčinnější kombinaci činností a zdrojů*
- *zobecňovat a přenášet tento postup řešení problémů do nejrůznějších dalších oblastí*

Tyto dovednosti jsou podpořeny předpoklady a postoji, které jsou taktéž nezbytnou součástí CT [9]:

- *sebejistota tváří v tvář složitosti*
- *vytrvalost při řešení obtížného problému*
- *snášení nejednoznačnosti*
- *schopnost vypořádat se s otevřenými problémy*
- *schopnost dorozumět se a spolupracovat s ostatními při dosahování společného cíle*

Pro podporu zařazení informatického myšlení do vyučování byl založen projekt Podpora rozvoje informatického myšlení (PRIM). Na tomto projektu se podílí všechny pedagogické fakulty České republiky a Národní ústav pro vzdělávání. Garantem projektu je Jihočeská univerzita v Českých Budějovicích. Jeho úkolem je například připravit vzdělávací materiály, školení vyučujících nebo popularizace informatiky. Významnou částí jsou i webové stránky `imysleni.cz` obsahující informace pro studenty, vyučující i rodiče o možnostech výuky informatiky. Učitelé ocení volně dostupné výukové materiály. Převládají zde především učebnice pro výuku algoritmizace a programování, což jsou dle projektu hlavní nástroje pro rozvoj informatického myšlení. [10][11]

1.4 Neformální vzdělávání

Další možností, kde se studenti mohou setkat s programováním, je výuka ve formě zájmového kroužku nebo při jiných organizacích, kde může figurovat například jako mimoškolní aktivita v domovech dětí a mládeže. Mezi hlavní výhody této výuky patří nejen zájem žáků o výuku programování, ale i menší počet studentů ve vyučovaných skupinách. Velice často se tyto kroužky zaměřují na programování s roboty jako je Lego Mindstorms nebo Ozobot. Také každoročně probíhá několik akcí pro popularizaci informatiky. Patří sem například Bobřík informatiky, Hour of Code nebo CodeWeek.

1.4.1 Bobřík informatiky

Bobřík informatiky je mezinárodní soutěž, kterou v ČR pořádá Jihočeská univerzita v Českých Budějovicích, která ukazuje žákům, že informatika není jen o ovládání aplikací. Přihlásit se mohou všechny základní a střední školy. Termín konání bývá vždy na podzim. Soutěží se v pěti věkových kategoriích od 4. ročníku základní školy po maturitní ročníky. Soutěž probíhá formou online testu, kde soutěžící vybírá jednu z možných odpovědí nebo přetahuje objekty. Jednotlivé otázky jsou zaměřeny na algoritmizaci, porozumění informacím nebo logiku. [12]

1.4.2 Hour of code

Hour of Code probíhá každý rok v prosinci, kdy se slaví Týden informatiky. Tato akce je pořádána neziskovou organizací Code.org. Jedná se o hodinový úvod do informatiky, který má za úkol uvést co nejvíce lidí do světa programování. Akci může organizovat prakticky kdokoli, jestliže má dost zájemců a prostory ke konání, a to díky velkému množství materiálů a již připravených kurzů. Většina těchto kurzů je zaměřena na vytváření her, protože pro mnoho studentů to představuje nejzábavnější způsob jak se naučit programovat. Do této akce je zapojena i velká skupina partnerů jako je Microsoft, Apple nebo Amazon.[13]

1.4.3 Code week

Školy, rodiče, učitelé, organizace, programátoři a firmy připravují v rámci Evropského týdne programování nejrůznější propagační a vzdělávací akce, kde ukazují, co vše lze programováním vytvořit. Cílem je nadchnout pro informatiku nejen více dětí, ale i jejich rodiče. Code Week tradičně probíhá v první polovině října. [14][15]

2 Nástroje pro podporu výuky algoritmizace a programování

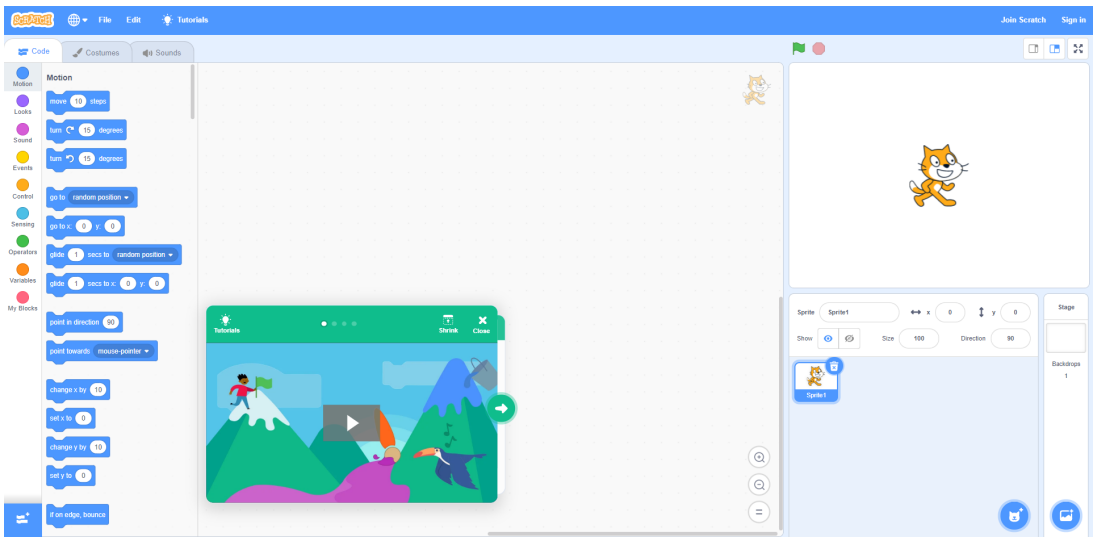
Jednou z metod, kterou lze využít pro výuku základů programování, je tzv. vizuální programování. Programuje se pomocí grafických bloků, které se přetahují a spojují do sebe čímž vzniká program. Díky tomu, že není třeba psát kód, odpadá nutnost znát syntaxi vybraného jazyka a studenti se tak mohou soustředit na samotný princip jejich postupu řešení.

2.1 Scratch

Scratch je vizuální programovací jazyk, jehož autorem je Mitchel Resnick. Je vytvořen pod záštitou skupiny Lifelong Kindergarten, působící v rámci MIT Media Lab. Cílí zejména na uživatele ve věku od 8 do 16 let. [16] Pro mladší uživatele ve věku od pěti do sedmi let existuje varianta ScratchJr. [17]

Aplikace funguje na principu spojování bloků. Nabízí celou škálu zabudovaných bloků od těch, které se zabývají pohybem a polohou objektu, přes logické bloky, jako jsou podmínky a cykly, až po bloky, které zajišťují funkcionalitu událostí a stavu objektu. Lze s ním vytvářet prezentace, hry nebo animace, které pak uživatel může sdílet s ostatními.

Je zcela zdarma a to ve formě webové nebo desktopové aplikace. Verze 2.0 pro své spuštění potřebuje doplněk Adobe Flash. Tento požadavek byl odstraněn v nejnovější verzi 3.0, která byla uvolněna 2. února 2019. Nová verze prošla kompletní proměnou designu i implementace. [18]



Obrázek 2: Náhled aplikace Scratch 3.0

Velkou výhodou Scratche je rozsáhlá komunita a množství dostupných materiálů. Na webových stránkách imysleni.cz jsou k dispozici učebnice pro programování v prostředí Scratch pro první i druhý stupeň základních škol. Z tiš-

těných publikací lze zakoupit Jak se naučit programovat v 10 lekcích od Seana McManuse a Programování pro děti v překladu od Aleny Halouskové. [11]

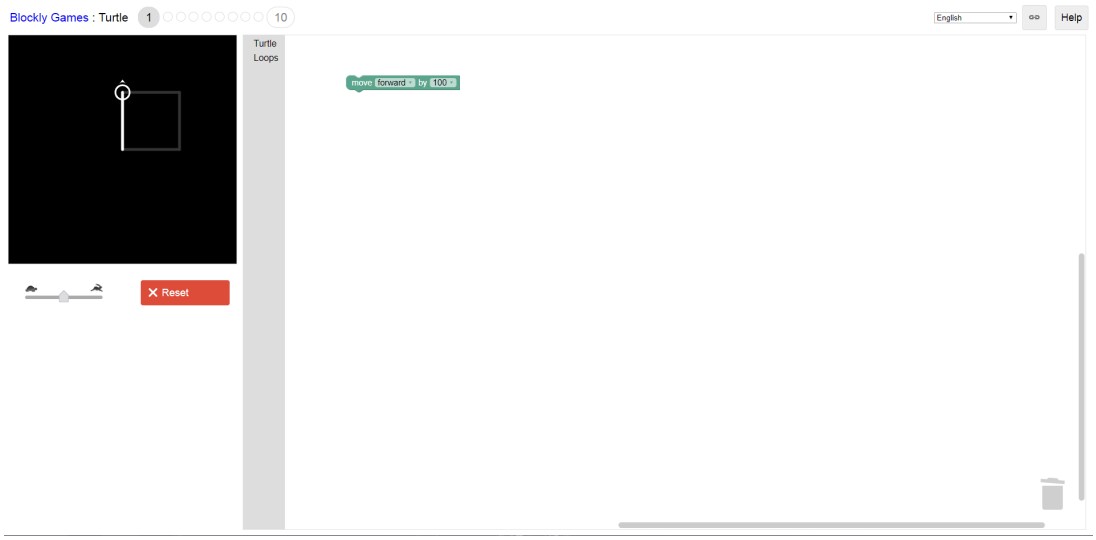
2.2 Blockly

Blockly je open-source vizuální programovací nástroj od společnosti Google, která ho vyvíjí od roku 2012. Jedná se o knihovnu, která dokáže z grafických bloků vygenerovat zdrojový kód. V současně době lze kód generovat do jazyků JavaScript, Python, PHP, Lua a Dart. Případně si lze naprogramovat vlastní generátor. Blockly není určeno jako samostatná aplikace pro výuku, ale je využíván pro velké množství projektů jako základ pro vývojové prostředí. Mezi tyto projekty patří například Blockly Games, Made with Code, App Inventor nebo OzoBlockly. [19]

2.2.1 Blockly Games

Blockly Games se skládá z několika sad úkolů, které postupně rostou na obtížnosti. Například v první úrovni se studenti učí jak skládat bloky jako puzzle. Ve druhé úrovni si zkusí práci se smyčkami a podmínkami, aby vyřešili úkoly v bludišti. Poslední úroveň slouží k přechodu od vizuálního programování k textovému programování v JavaScriptu.

Každá úroveň se skládá z pěti až deseti úkolů. Pevné pořadí umožňuje studentovi využít znalosti získané v předchozím úkolu a postupně jej vede ke komplexnějším programům jako je například kresba pomocí želví grafiky. [20]



Obrázek 3: Náhled aplikace Blockly Games Želva

2.2.2 Ozobot

Ozobot je malý robot ve tvaru koule postavený na optických senzorech a vybavený LED diodami. Dokáže se pohybovat po nakreslené čáře na papíře a rozpozná i její barvu. Podle rozpoznané barvy se pak rozsvítí ve stejné barvě jeho LED diody. Další variantou je použití mobilní aplikace přímo od výrobce, kde se kreslí dráhy a příkazy přímo na displej. [21]

Ozobota lze použít i bez kreslení na papír a mobilní aplikace. Tato varianta je dostupná z prostředí Ozoblockly ve formě webové aplikace, která je postavena na Google Blockly. Poskládá se program a do robota se nahraje pomocí blikání světel. Ozobot využívá ke komunikaci barevný jazyk založený na různých variacích zelené, modré a červené, takzvaný ozokód. Pro seznámení s prostředím lze využít výukové lekce na `http://games.ozoblockly.com`. Momentálně jsou přístupné lekce Shape Tracer a Shape Tracer 2, které stejně jako Blockly-Games, využívají stupňované obtížnosti úkolů. První sada využívá jen sekvence pohybu. Druhá sada provede studenta smyčkami. Obě sady simulují pohyb Ozobota přímo na obrazovce. [21][22]

Obrázek 4: Náhled aplikace Ozobot Shape Tracer

2.3 Baltík

Baltík je programovací jazyk a vývojové prostředí, pocházející od české firmy SGP Systems, vhodný i pro nejmenší studenty (doporučován již od 4 let). Používá se především na základních školách, ale své místo má i v rámci soutěží, jako je Mladý programátor nebo Baltie. Program se tvoří pomocí přetahování a skládání obrázkových ikon místo textových příkazů, ale od verze 4 lze programovat i pomocí kódu, konkrétně v jazyce C#. [23]

Aplikace Baltík 3 obsahuje tři režimy s rostoucí složitostí a možnostmi: skládat scénu, čarovat scénu, programovat. Do scény lze vložit i postavičku čaroděje

Baltíka, který je programovatelný pomocí ikonek. Obsahuje ikony pro proměnné, podmínky, cykly, procedury, základní matematické funkce, animace nebo příkazy pro práci se soubory. Pro ladění programů je k dispozici krokování a zobrazení proměnných.

2.4 Kodu

Kodu je vizuální programovací jazyk vyvinutý společností Microsoft, který byl původně určen pro herní konzole XBOX 360 a systémy MS Windows. Podpora pro konzole však byla v roce 2017 ukončena. Byl primárně vytvořen pro tvorbu 3D her. Můžeme jej tedy nazvat jakýmsi herním enginem, pomocí kterého můžeme vytvořit hru téměř libovolného herního žánru.

Uživatel si nakreslí herní mapu, vytvoří terén a vloží postavu robota Kodu nebo další roboty. Těm následně naprogramuje požadovanou činnost. Programovací logika je založená na jednoduchých podmínkách `WHEN-DO`. Výsledek je možné si hned vyzkoušet. Za nedostatek by se dal označit fakt, že uživatel si do své hry nemůže importovat žádný vlastní obsah. [24][25]

3 Návrh řešení

Po prostudování existujících nástrojů bylo zjištěno, že na žádné z popsaných platforem nelze vytvořit vlastní úkol, který by byl přístupný i pro ostatní uživatele, kteří by ihned získali zpětnou vazbu o správnosti řešení. Scratch tutoriály většinou nabízí jen popis toho, jak něco vytvořit nebo jak používat animace. Blockly Games nebo Ozobot ShapeTracer mají předdefinované úkoly a neumožňují vyučujícímu vytvořit své vlastní zadání. Byla navrhnuta webová aplikace skládající se z vývojového prostředí a aplikace pro tvorbu a procvičování úkolů. Celá práce získala pracovní název *Cubo*. Po konzultaci s vedoucím práce, jsem se rozhodla jako základ vývojového prostředí využít open source knihovnu Blockly a rozšířit ji především o bloky pro kreslení a možnosti ladění.

3.1 Vývojové prostředí

U vývojového prostředí se předpokládá použití i bez nutnosti přihlášení a umožnění jednoduchého skládání z bloků bez nutnosti se seznamovat se syntaxí. Mezi základní funkčnosti by mělo patřit spuštění poskládaného programu s možností podívat se na výsledek a základní ladění - spuštění ladění, krok vpřed, krok zpět, zastavení ladění a výpis hodnot proměnných v každém kroku.

Bloky jako proměnné nebo cykly jsou standardní součástí knihovny Blockly. Nově vytvořené bloky budou sloužit pro kreslení na element `canvas`:

- nastavení vlastností plátna jako jsou pozadí, šířka a výška
- posun po plátně
- kreslení čar a nastavení jejich vlastností
- kreslení tvarů - obdélník, kruh, oblouk nebo polygon
- kreslení textu a nastavení jeho velikosti a typu fontu

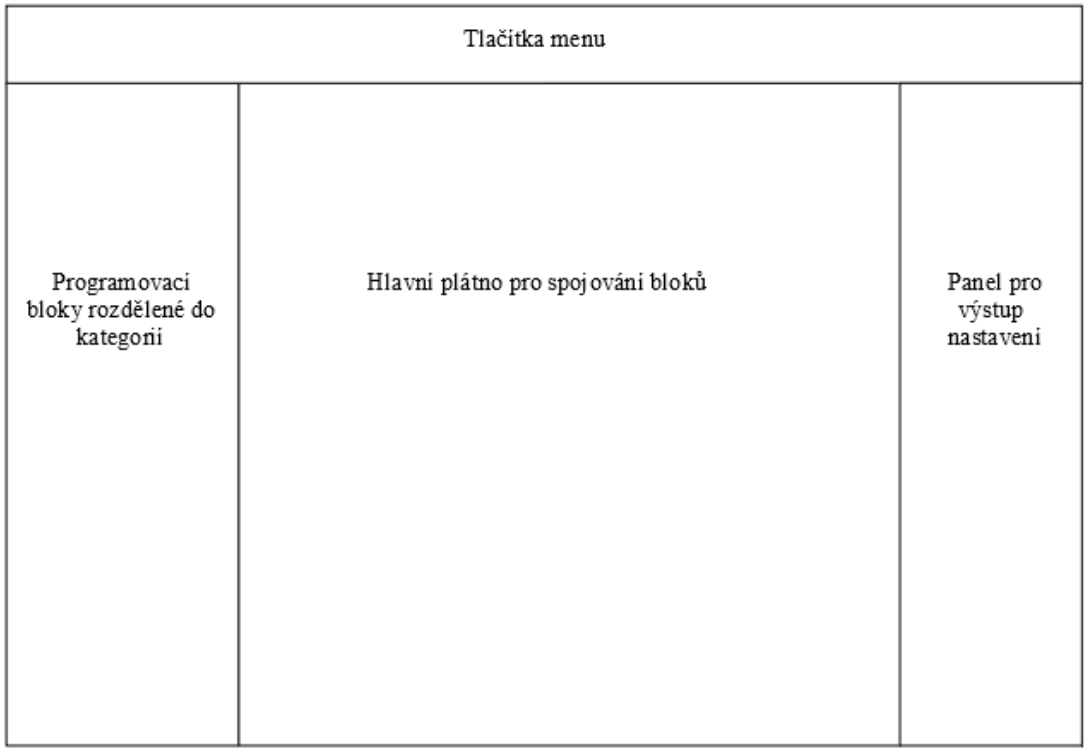
3.2 Aplikace

V této části mi byla inspirací především webová aplikace Codewars [26]. Projekt se zaměřuje na zlepšování programátorských dovedností a předpokládá už alespoň základní znalost některého programovacího jazyka s tím, že nabídka jazyků je opravdu bohatá. Uživatel si vybírá úkol, který chce plnit, tzv. katu neboli výzvu a dostává okamžitě zpětnou vazbu o správnosti svého řešení. Některé výzvy jsou zaměřeny na základy programování, jiné na řešení více komplexních problémů nebo hádanky na testování kreativity při řešení problému.

V aplikaci *Cubo* bude moci přihlášený uživatel tvořit vlastní úkoly nebo procvičovat úkoly vytvořené ostatními uživateli. To může aktivnějším studentům pomoci s prohlubováním jejich znalostí, protože nebudou vázáni pouze na cvičení od svého vyučujícího, ale zpřístupní se jim celá škála různých zadání. Navíc

pokud vytvoří vlastní zadání, tak si jej lehce mohou vyzkoušet spolužáci i vyučující.

Budou implementovány tři typy úkolů. První variantou je *Přesná shoda*, kdy uživatel na pracovní plochu umístí bloky popsáné v zadání, což lze použít pro úkoly na seznámení se s jednotlivými bloky. Dalším variantou je *Vytvoření funkce*, kdy má uživatel za úkol vytvořit funkci, která bude vrátí nějaký výsledek a ten se následně porovná s výstupy ze zadání. Třetím typem je úkol podobný z Blockly Games (Želva), kdy má uživatel předlohu na plátně a musí vytvořit stejný obrázek tak, aby se překrývaly. Po kliknutí na tlačítko kontroly dostane odpověď na správnost svého řešení.



Obrázek 5: Rozložení uživatelského rozhraní vývojového prostředí

4 Implementace

Implementaci aplikace používá JavaScript jak na straně klienta, tak na straně serveru. Na straně klienta se jedná o frontendový framework Vue.js, běhové prostředí Node.js s frameworkem Express.js je pak použito na serveru. Dále byla rozšířena knihovna Blockly o bloky pro kreslení.

4.1 Knihovna Blockly

Knihovna Blockly se stala základem pro vývojové prostředí. Mezi hlavní faktory pro výběr patřila implementace projektu v JavaScriptu, rozšiřitelnost, dobře popsaná dokumentace, pravidelné aktualizace od vývojářů a podpora rozsáhlé komunity. Z některých projektů, jako výše již zmíněného Blockly Games, se dá vycházet při implementaci nových funkcionalit.

Nejdůležitější část knihovny tvoří zdrojové kódy ve složce `core`. Obsahuje především třídy pro vytvoření bloků, toolboxu, proměnných nebo pracovní plochy. Jak jednotlivé bloky vypadají řeší definice bloků ve složce `blocks`.

4.1.1 Vytvoření bloků

Každý blok se skládá z definice bloku a svého generátoru kódu. Definice bloku popisuje, jak blok vypadá a jak se chová, včetně textu, barvy, tvaru, nápovědy a k jakým ostatním blokům se může připojit. Existují dva způsoby definice bloku. Prvním z nich je vytvoření JSON objektu, který je platformově nezávislý a doporučený podle dokumentace. Bohužel neumí přímo definovat pokročilejší vlastnosti jako jsou například `mutators`. Musí být napsány v nativním kódu použité platformy, v tomto případě JavaScriptu, a použity jako rozšíření.

Mezi základní vlastnosti bloku patří identifikátor bloku, nápověda, barva a směr připojení. Svislý směr připojení udává, zda bude možné bloky spojovat do skupiny za sebou prováděných příkazů, což odpovídá odřádkování příkazů v textových programovacích jazycích. Lze zvolit dolní, horní, obě varianty nebo bez připojení. Speciálním případem je svislé vnitřní připojení. Blok obsahuje prostor pro vložení bloků dovnitř, toho využívají podmínky, cykly, funkce nebo mnohoúhelník.

Vodorovné připojení se používá především pro přiřazení nebo dosazení hodnoty. Pokud pro blok zvolíme vodorovné připojení vlevo (výstup), můžeme definovat i typ výstupu. Knihovna nabízí několik typů jako `boolean`, `number`, `string` nebo `array`.

Připojení vpravo je vstupem bloku. Blok může mít několik vstupů, kdy každý vstup může být kromě připojení i hodnotou. V tom případě se v bloku vytváří tzv. pole (`field`). Pole je několika typů například textové, barva, zaškrtávací políčko, datum nebo úhel. Za zvláštní typ se dá označit pole proměnná. Jedná se o rozbalovací seznam, který obsahuje existující proměnné dostupné v pracovní ploše.

```
Blockly.Blocks['fill'] = {
   init: function () {
      this.appendValueInput('fillColour')
         .setCheck('Colour')
         .appendField('výplň')
      this.setPreviousStatement(true, null)
      this.setNextStatement(true, null)
      this.setStyle('lines_blocks')
      this.setTooltip(Blockly.Msg["LINES_FILL"])
      this.setHelpUrl(Blockly.Msg["LINES_HELP_URL"])
   }
}
```

Zdrojový kód 1: Definice bloku výplň

O převod bloků na zdrojový kód se stará generátor. V knihovně jich je k dispozici několik, ve vývojovém prostředí je používán javascriptový generátor dostupný ve složce `generators`. Nejprve se vytvoří kostra kódu s definicí proměnných a následuje samotný překlad jednotlivých bloků v pracovní ploše podle nadefinovaného generátoru bloku. Pro správnou syntaxi poskytuje generátor i výčet priorit operátorů.

Překlad bloku vždy začíná získáním všech argumentů a hodnot z polí (`field`). K tomu slouží především funkce `getFieldValue`, `valueToCode` a `statementToCode`. Jakmile jsou shromážděny všechny argumenty, je možné sestavit konečný kód, což je dost přímé pro všechny bloky viz Zdrojový kód 2 .

```
Blockly.JavaScript['canvas_set_size'] = function (block) {
  var size = block.getFieldValue('SIZE');
  var value = Blockly.JavaScript.valueToCode(block, 'value', Blockly
      .JavaScript.ORDER_ATOMIC);
   return 'setSizeCubo(' + value + ',"' + size + '");\n'
};
```

Zdrojový kód 2: Generátor pro blok nastavení šířky a výšky plátna

Byly vytvořeny následující bloky s generátory:

- **text** - písmeno na pozici, podřetězec
- **proměnné** - lokální proměnná
- **plátno** - nastavit pozadí, šířka/výška plátna, nastavit šířku/výšku plátna, posun na souřadnice, posun o délku a úhel, posun na předdefinovanou pozici, vykreslení aktuální pozice, souřadnice aktuální pozice
- **barvy a čáry** - výplň, barva čáry, šířka čáry, zakončení čáry, kreslení čáry o délce a úhel

- **tvary** - obdélník, kruh, oblouk, oblouk, mnohoúhelník, nastavení fontu a velikosti, kreslení textu

4.1.2 Proměnné a funkce

Proměnné se vytváří pomocí tlačítka *Vytvořit proměnnou*. Po vytvoření alespoň jedné proměnné, se aktivují gettery a settery v toolboxu v kategorii *Proměnné*. Přejmenování a mazání proměnných probíhá přes rozbalovací menu v daném políčku getteru/setteru, které obsahuje všechny proměnné v pracovní ploše. Vytvořené proměnné nemají vazbu na jednotlivé bloky, jsou uloženy v objektu `variableMap` podle typu. Každou proměnnou reprezentuje objekt `variableModel`. Generátor kódu pak vytváří proměnné z této mapy. Všechny vytvořené proměnné jsou globální.

Funkce jsou vytvářeny pomocí příslušného bloku. Po vytvoření funkce se vygeneruje blok pro volání této funkce. Každá funkce generuje do toolboxu své vlastní bloky. Na rozdíl od proměnných jsou funkce uloženy přímo v pracovní ploše. Generátor pak projde všechny bloky na ploše a filtruje je dle typu, aby vybral definované funkce. Tyto bloky mají metodu `getProcedureDefinition`, která určuje název, parametry a návratovou hodnotu. Pokud ale vytvoříme funkci s parametry `x` a `y`, tak zjistíme, že dané proměnné se vytvořily automaticky a uživatel s nimi může pracovat v rámci celé pracovní plochy. Což pro něj může být matoucí, protože je sám přímo nevytvořil. Řešením by bylo zamykání bloku funkce, aby uživatel nemohl přetáhnout danou proměnnou jinam, ale i tento způsob by pro něj mohl být matoucí. Navíc neodpovídá konceptu Blockly, že začátečník nepotřebuje znát problém prostředí [27].

S tím, že jsou všechny proměnné globální, souvisí další problém. Pokud vytvoříme funkci, která obsahuje cyklus s proměnnou `i`, a zároveň v hlavním programu vytvoříme cyklus taktéž s proměnnou `i`, kde funkci zavoláme, nastane situace kdy se tyto dva cykly budou navzájem ovlivňovat. Pro odstranění tohoto nedostatku byl vytvořen nový blok. Ten nastaví vybranou proměnnou jako lokální, zastíní globální. Pomocí události, která sleduje změny tohoto bloku, ho lze vložit pouze do bloku funkce. Jinak bude nedostupný a generátor kódu ho bude ignorovat. Takto nedostupný blok se odlišuje od ostatních pomocí šrafování a výstrahy.

4.1.3 Spuštění a ladění

Po vygenerování kódu se nabízí dvě možnosti jak jej spouštět. První variantou je použít funkci `eval`, která vyhodnotí vygenerovaný javascriptový kód v hlavním vlákně. Druhou variantou je využít sandboxovaný JS Interpreter vytvořený Neilem Fraserem, vedoucím vývoje Blockly. Projekt není součástí knihovny, ale byl vytvořen přímo pro ni. Umožňuje vykonání kódu JavaScriptu řádek po řádku, což vývojové prostředí využívá pro ladění. Spuštění je zcela izolováno z hlavního vlákna JavaScriptu a probíhá v novém vlákně pomocí WebWorkeru. Pomocí interpretu také lze vytvořit API pro volání funkcí z globálního prostředí.

Jelikož spuštění kódu probíhá mimo hlavní javascriptové vlákno, nastává problém jak kreslit na plátno, tedy element `canvas`. Ten potřebuje ke svému fungování kontext `window`. Řešením je použití `OffscreenCanvas` rozhraní, které umožňuje kreslení i mimo hlavní vlákno. Jedná se ovšem zatím pouze o experimentální technologii a nepodporují ji zatím všechny prohlížeče [28]. Pro prohlížeče bez této vlastnosti funguje náhradní řešení, kdy se kód spustí v hlavním vlákně, což ale může způsobit zamrzání vývojového prostředí.

```
initDebugApi(interpreter, scope) {
    interpreter.setProperty(
      scope,
      "highlightBlock",
      interpreter.createNativeFunction((id) => {
        id = id ? id.toString() : "";
        this.highlightBlock(id);
      })
    );
}
```

Zdrojový kód 3: API pro volání funkce z globálního prostředí z interpretu

Knihovna Blockly nenabízí žádný způsob jak získat aktuální hodnotu v proměnné při krokování. Ale umožňuje získat mapu všech použitých proměnných. Z interpretru jsme ale schopni zjistit hodnotu proměnné v aktuálním prostředí, což pak lze použít pro sledování aktuální hodnoty proměnné v daném kroku.

```
this.variableDB.forEach((element) => {
      var v = this.interpreter.getValueFromScope(element.name);
      if (v.data !== undefined) {
        Debug.variables[element.name] = v.data; //primitive value
      } else {
        Debug.variables[element.name] = v.toString(); //object value
      }

      if (
        Debug.variablesLatest[element.name] !== undefined &&
        Debug.variables[element.name] === undefined
      ) {
        Debug.variables[element.name] = Debug.variablesLatest[
            element.name];
      }
    });
```

Zdrojový kód 4: Získání hodnot proměnných z interpretu

Interpret bohužel nepodporuje možnost kroku zpět. Tento nedostatek byl vyřešen počítáním vykonaných bloků/příkazů. Díky tomu, pokud uživatel klikne

na tlačítko *Krok zpět*, proběhne kód na pozadí opět znovu až do počtu kroků sníženého o jedna. Toto řešení je neefektivní, ovšem pro výukové účely postačující a není nutný zásah do kódu interpretu.

4.1.4 Úpravy knihovny

Kromě vytvoření nových bloků byly provedeny změny i v samotné knihovně. Jedná se především o úpravu toolboxu tak, aby do kategorie šla vložit ikona, a generování bloku `lokální proměnná`. Další změny se týkaly především vzhledu jednotlivých komponent v Blockly jako jsou scrollbary nebo tooltipy.

4.2 Node.js

Node.js je asynchronní, událostmi řízené běhové prostředí pro JavaScript, využívající vysoce výkonný Chrome JavaScript Runtime (V8 engine) od Googlu, který je napsaný v C++. Používá jej například i prohlížeč Google Chrome. Node.js se používá na straně serveru. Používá pouze jedno vlákno se smyčkou událostí. Ta umožňuje provádět neblokující vstupně/výstupní operace.Tyto vlastnosti dovolují v Node.js tvořit i škálovatelné systémy. Navíc umožňuje využívat řadu modulů, které se dají jednoduše stáhnout pomocí balíčkovacího systému npm, který se instaluje společně s Node.js. [29]

Webové servery, jako je Apache, nejsou stavěné pro masivní počty požadavků. I z toho důvodu jsem zvolila právě Node.js. Mezi další důvody patří jednoduché vytvoření API, vhodnost pro single page aplikace a také možnosti použít jeden jazyk na serveru i klientovi. Jeden z hlavních balíčků, který aplikace Cubo využívá k provozu serveru a definování REST API, je balíček Express.js.

Express.js je rychlý, flexibilní framework pro tvorbu Node.js aplikací. Umožňuje tvorbu obsluhy žádostí na základně HTTP metod pro různé URL adresy skrze objekt `router`. Implementuje podporu middleware funkcí, které se vkládá do procesu obsluhy požadavků a přidává další funkčnost jako například správu přihlašování. Pomocí Express.js je definováno REST API aplikace. [30]

4.3 MongoDB

Pro perzistenci dat byl vybrán databázový systém MongoDB. MongoDB je multiplatformní NoSQL dokumentová databáze implementovaná v C++. To, že databázový systém nepoužívá jazyk SQL, znamená, že neukládá data do klasických relačních objektů, ale místo toho definuje dokumenty ve formátu BSON (binární JSON). Tyto dokumenty jsou uloženy v kolekcích. MongoDB ukládá data tak, jak je obdrží, takže každý dokument v jedné kolekci může mít jiný formát. [31] MongoDB bylo vybráno jako výborný doplněk k Node.js. Data v MongoDB mají flexibilní schéma, což se hodí pro postupný vývoj a případné rozšíření aplikace. Práce využívá cloudovou databázovou službu mLab [32].

I když má každý dokument flexibilní schéma, stále je potřeba určitá validace dat, která je ale v MongoDB vcelku krkolomná, a proto byla použita knihovna

Mongoose. Mongoose slouží k modelování objektů – definicím jejich podoby, validacím a funkcím pro práci s nimi. Z definovaných schémat se následně pomocí funkce model vytváří tzv. modely reprezentující schémata obohacená o funkce, které odpovídají akcím v databázi, a případně dalšími, které si může vývojář definovat. Jedná se například o:

- `save` – uložení a případné přepsání již existujícího dokumentu v databázi
- `find` – nalezení a vrácení všech dokumentů z množiny daného dokumentu
- `findById` - nalezení a vrácení dokumentu z databáze podle jeho jedinečného identifikátoru

Na některé z těchto základních funkcí knihovny Mongoose lze dále vázat sekundární funkce. Typicky se může jednat o funkci `query` nebo `sort`, které se starají o výběr určité podmnožiny dokumentů z databáze a o seřazení těchto dokumentů podle specifikovaných kritérií. Lze zrychlit i samotné dotazování. Ve výchozím nastavení dotazy vrací instanci třídy `Mongoose Document`. Použitím funkce `lean` je vrácen jednoduchý javascriptový objekt, který není tak náročný na paměť, což je výhodné především při čtení dat. [33]

4.4 REST API

REST (Representational State Transfer) je architektonický standard pro vytváření webových služeb, který ke komunikaci využívá HTTP protokol. Každý HTTP požadavek je typicky následován odpovědí od serveru. HTTP požadavek a odpověď jsou si podobné a oba mají čtyři hlavní části [34]:

- **URL** – cesta, na kterou má požadavek dorazit
- **metoda** – typ akce, která má být provedena na dané URL cestě
- **hlavička** – metadata o dotazu, typicky se zde nachází nějaký ze způsobů autentizace uživatele dané cesty HTTP metod
- **tělo** – obsahuje data dotazu, nejčastěji ve formátu JSON nebo XML

Základní metody požadavku se označují zkratkou CRUD:

- **Create** – vytvoření dat pomocí metody POST
- **Retrieve** – získání dat pomocí metody GET
- **Update** – aktualizace dat pomocí metody PUT
- **Delete** – smazání dat pomocí metody DELETE

Výhodou REST API je jeho nedefinovanost, což znamená, že si jej můžeme vytvořit přesně na míru požadavkům aplikace. Definované endpointy jsou v souboru `server/route.js`. Většina z nich vyžaduje autentizaci uživatele, aby byla zajištěna bezpečnost a integrita poskytovaných dat. Každý endpoint je obsluhován příslušnou funkcí z kontroleru. V aplikaci se vytvoří požadavek na daný endpoint. Ten tento požadavek zpracuje a vrátí požadovaná data ve formátu JSON.

Autentizace uživatelů mezi serverem a klientem je řešena pomocí technologie JSON Web Token (JWT) pomocí knihovny `passport.js` [35]. Pro přístup k API musí být v hlavičkách HTTP požadavku umístěn autentizační token. Pokud požadavek neobsahuje tento token nebo je neplatný, je zamítnut s chybovým stavovým kódem. O vydání tokenu musí klient požádat poskytnutím svých přihlašovacích údajů.

4.5 Vue.js

Vue.js (dále jen Vue) je knihovna, javascriptový framework, který se používá na straně frontendu k vytváření dynamických uživatelských rozhraní. Jedná se o poměrně nový framework vytvořený bývalým vývojářem z Angular týmu. Vue je poměrně rychlá a zároveň malá (myšleno velikostí) knihovna, protože neobsahuje mnoho funkcionality. Za velkou výhodu se dá považovat rychlá přímka učení. Vývojové prostředí i aplikace využívají knihovnu Vue.js 2. [36]

4.5.1 Komponenty

Základním prvkem Vue aplikace je komponenta. Komponenty jsou zapisovány do souboru s koncovkou `.vue`. Soubor lze rozdělit na tři části jako šablona, model a styly. Šablony používají syntaxi založenou na HTML. Lze také využít v-direktivy, což jsou speciální HTML atributy s prefixem „v-“. Pokud se hodnota takové direktivy změní, Vue reaktivně aktualizuje rozhraní. Mezi takové direktivy patří například `v-if` pro podmíněné renderování obsahu nebo `v-for` pro využití cyklu. [36]

```
1 <task v-for="i in items" :key="i._id" :task="i" :user="user" />
```

Zdrojový kód 5: Použití v-for

Část definující model pro šablonu musí být zapsána mezi tagy `script` a její syntaxe odpovídá javascriptovému objektu. Nejdůležitější částí objektu je funkce data, která představuje jeho stav. Data se musí uchovávat v rámci funkce z důvodu uchování jedinečné instance pro každou instanci komponenty. V případě, že by data byla uložena jako jednoduchý objekt, změna dat v jedné komponentě by se promítla i do ostatních instancí. Jednotlivé položky tohoto objektu

je možné tisknout v šabloně. Jakákoliv změna tohoto objektu automaticky promítne změnu i do šablony. Objekt může obsahovat mnoho dalších atributů, mezi které patří například atribut `methods`, `properties` nebo `computed`. [36]

Poslední částí komponenty jsou kaskádové styly zapisované mezi tagy `style`. Tyto styly se vztahují především k dané komponentě.

4.5.2 Vue router

Single page aplikace představuje pro uživatele nestandardní chování, protože se jedná pouze o jednu stránku, jejíž obsah se dynamicky mění, ale URL adresa ne. Uživatel je zvyklý, že se mění URL adresa jednotlivých stránek a že lze využívat tlačítka vpřed a zpět v navigaci. K odstranění této překážky slouží knihovna Vue Router. Umožňuje mapování komponent na různé cesty. Každá stránka je definovaná cestou, jménem a komponentou. [37]

```
{
   path: "/vyvojove-prostredi",
   name: "IDE",
   meta: {
     layout: "ide",
     title: "Vývojové prostředí | Cubo"
   },
   component: () =>
     import("../views/layout/IDE.vue")
 }
```

Zdrojový kód 6: Vytvoření cesty pomocí Vue Router

4.5.3 Vuex

Vuex knihovna slouží pro udržování globálního stavu aplikace a byla vytvořena přímo pro Vue. Využití v projektu najde v případě, pokud více komponent závisí na jednom stavu nebo komponenty z různých částí aplikace potřebují změnit stejná data. Aplikace Cubo toho využívá například pro uložení pracovní plochy. Mezi základní části patří:

- `state` - představuje objekt, který obsahuje data, které mají být k dispozici v rámci celé aplikace
- `getter` - slouží k přístupu k uloženým datům v `state`
- `mutations` - slouží pro provedení změn
- `actions` - slouží pro asynchronní zpracování nebo vyvolání mutace

```
export default {
  namespaced: true,

  state: {
    workspace: null,
  },

  mutations: {
    setWorkspace(state, w) {
      state.workspace = w
    },
  },
  actions: {
    setWorkspace({ commit }, w) {
      commit('setWorkspace', w)
    }
  },
  getters: {
    workspace: state => state.workspace,
  }
}
```

Zdrojový kód 7: Příklad použití Vuex store pro uložení pracovní plochy

Vuex úložiště může narůst do velkých rozměrů a tím se stane velmi nepřehledným. Řešením je rozdělit ho do logických celků. Jednou z možností je rozpad na jeho jednotlivé části jako do samostatných souborů. Druhou možností je vytvoření modulů. Tuto možnost využívá i aplikace Cubo. Každý modul obsahuje své vlastní stavy, mutace, akce a gettery a představuje určitou funkcionalitu aplikace. Jednotlivé moduly mohou existovat jak v globálním jmenném prostoru, tak ve vlastním jmenném prostoru, což odstiňuje problém s případnými kolizemi v názvech proměnných. [38]

4.5.4 Komunikace se serverem

Pro komunikaci se serverem se používá malá knihovna `axios` [39]. Její výhodou je podpora příslibů, práce s objekty odeslaného požadavku a odpovědi nebo také automatická transformace dat ve formátu JSON. Protože většina požadavků musí obsahovat autorizační token, je vytvořeno pomocné rozhraní `services/Api.js`, které obsahuje nastavení tokenu a požadavek je pak volán přes něj.

4.5.5 Webpack

Velmi bohatě konfigurovatelný nástroj, který na základě závislostí z několika vstupních souborů vygeneruje jeden výstupní soubor. Tyto vstupní soubory mohou být jak javascriptové soubory, tak i ostatní části aplikace jako kaskádové

styly a obrázky. Právě díky Webpacku lze použít Vue Single File komponenty s preprocesory a vytvořit přehlednou strukturu práce. [36]

4.5.6 Ikony

Při implementaci uživatelského rozhraní byly použity některé ikony z projektu FontAwesome ve verzi 5. Obsahuje škálovatelné vektorové ikony, které lze přizpůsobit pomocí kaskádových stylů. Obsahuje přes 1500 ikon pro volné použití a stále je rozšiřován. [40] Zbývající ikony byly vytvořeny v programu LibreOffice Draw ve formátu SVG.

4.6 Kontrola úkolů

Kontroly úkolů probíhají v samostatném vlákně na základě typu úkolu. V případě úkolu *Přesná shoda* je kontrolováno, zda vygenerovaný kód uživatele souhlasí s kódem v zadání. Úkol typu *Vytvoření funkce* vezme uživatelem vytvořenou funkci a otestuje zda její výstupy souhlasí s výstupy z funkce v zadání. Pokud se vybere úkol *Kreslení*, uživatel při vykonávání úkolu vidí na plátně požadovaný výsledek. Toho je docíleno použitím dvou elementů `canvas`. Kontrola následně probíhá pomocí překryvu průhlednosti pixelů.

5 Uživatelská příručka

Tato kapitola obsahuje popis použití celé aplikace Cubo pro uživatele, například jak je rozloženo uživatelské rozhraní, funkcionalitu jednotlivých bloků, jak se program spouští, ladí, ukládá nebo načítá.

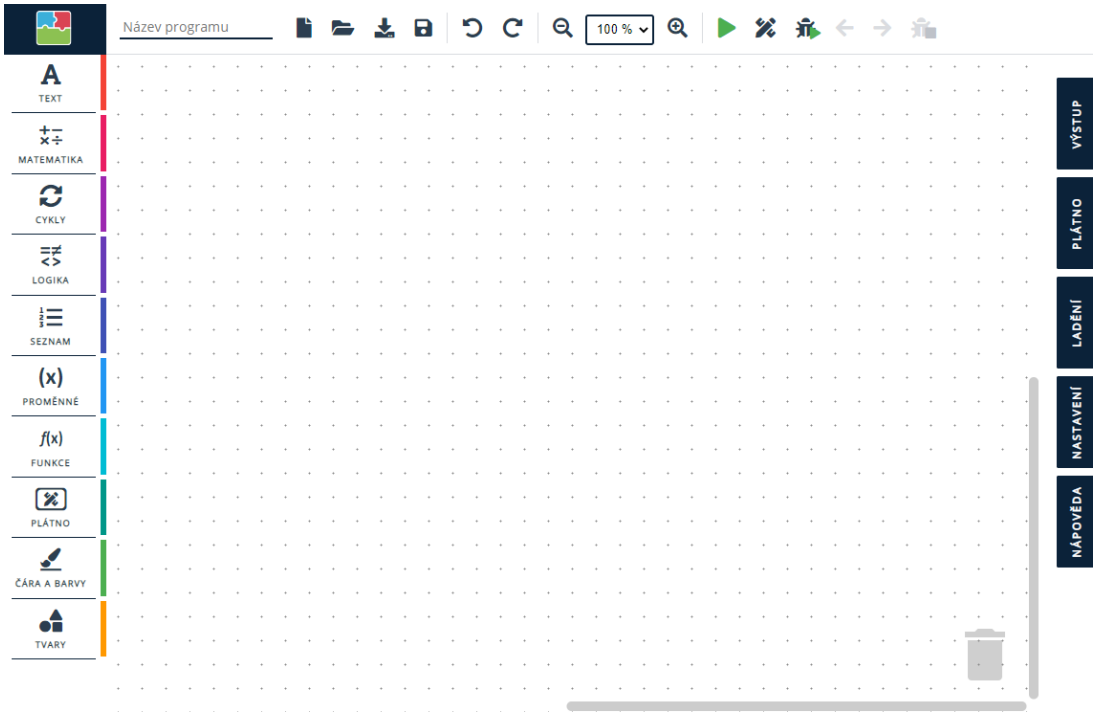
5.1 Vývojové prostředí

Vývojové prostředí má podobu webové aplikace. Aplikace je pouze částečně responzivní, předpokládá se využití především na tabletech a počítačích. Doporučený prohlížeč pro použití je Google Chrome. Je dostupné na https://kmi-cubo.herokuapp.com/vyvojove-prostredi

5.1.1 Hlavní menu

V horní části vývojového prostředí se nachází hlavní ovládací menu. Na Obrázku 6 vlevo nahoře se nachází ikona aplikace, Další zleva jsou tyto akce:

- Název programu
- Nový program
- Načtení programu
- Stažení programu
- Uložit
- Akce zpět
- Akce znovu
- Oddálení
- Lupa
- Přiblížení
- Spustit
- Zobrazení plátna
- Spustit ladění
- Krok zpět
- Krok vpřed
- Zastavit ladění



Obrázek 6: Náhled vývojového prostředí

Tlačítko *Nový program* slouží na vymazání stávajícího projektu bez uložení. Program lze stáhnout do formátu XML tlačítkem *Stažení programu*. Název souboru udává pole *Název programu*. Pokud není vyplněno, název souboru má tvar `bez\_nazvu.xml`. K nahrání tohoto souboru slouží tlačítko *Načíst program*, kdy aplikace vyzve k vybrání cesty k souboru s příponou `.xml`. Pokud je vybrán soubor jiného typu nebo soubor nemá správnou strukturu, objeví se upozornění, že soubor nebyl načten.

Pokud nechceme program stahovat a přesto ho mít nějakým způsobem uložen, je možné využít dočasné lokální uložení do `localStorage` pomocí tlačítka *Uložit*. Při obnovení stránky je pak uložený program automaticky vložen do pracovní plochy, pokud je to povoleno v nastavení vývojového prostředí.

V případě například nechtěného smazání bloku se můžeme vrátit zpět kliknutím na tlačítko *Zpět* nebo opakovat akci tlačítkem *Znovu*. Zobrazení velikosti bloků a pracovní plochy se mění pomocí tlačítek *Oddálení*, *Přiblížení* a *Lupa*. První dvě zmiňované přibližují nebo oddalují o 25%. Tlačítko *Lupa* nastavuje velikost změny zobrazení ručně nebo vybráním z předdefinovaných hodnot.

Tlačítko *Spustit* spustí vytvořený program. Pokud něco kreslíme výsledek zobrazuje plátno, dostupné pomocí tlačítka *Zobrazit plátno*. Po kliknutí na tlačítko *Spustit ladění* se automaticky zpřístupní ostatní tlačítka, která slouží k ovládání ladění viz kapitola 5.1.5.

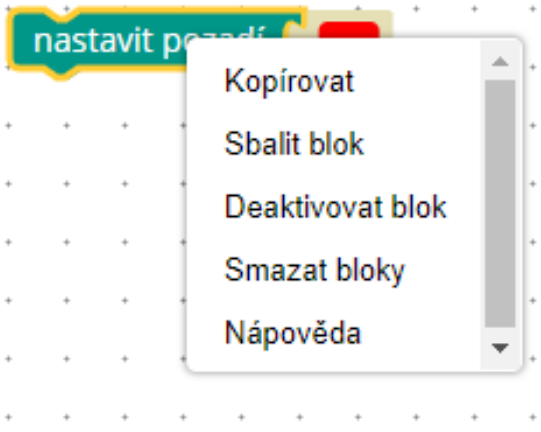
Vývojové prostředí podporuje klávesové zkratky uvedené v Tabulce 2.

5.1.2 Kategorie bloků a pracovní plocha

V levé části prostředí se nachází menu kategorií programovacích bloků. Každá kategorie je definována nějakou ikonou, která vyjadřuje funkcionalitu nabízených bloků. Po kliknutí na kategorii se otevře její nabídka bloků. Přidání bloku do programu se provede jednoduchým přetažením bloku (podržením levého tlačítka myši a táhnutím) na pracovní plochu. Zde se také provádí spojování bloků do logických celků. Kliknutím pravým tlačítkem na pracovní plochu myši se otevře kontextové menu, které umožňuje například sbalit a smazat všechny bloky z pracovní plochy. Pokud je program příliš dlouhý, je možné jej posouvat vodorovně a svisle pomocí posuvníků v pravé a dolní části pracovní plochy. Bloky se automaticky přichytávají k nejbližšímu bodu zobrazené mřížky.

5.1.3 Ovládání bloků

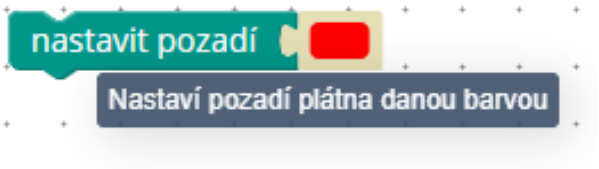
Přidání bloku je realizováno pomocí přetažení požadovaného bloku z kategorie bloků a umístění na pracovní plochu. Některé bloky mají rozevírací seznam, který umožňuje změnit fungování bloku. Jiné bloky mají jako vstup „puzzle“, kam lze umístit další bloky, například konstantu nebo proměnnou, a zároveň tento tvar značí to, že umístěný blok vrací nějakou hodnotu. Bloky pro cykly, rozhodování nebo vytvoření mnohoúhelníku umožňují zanořování ostatních bloků dovnitř sebe samých. Může se stát, že blok po vložení na pracovní plochu bude mít v levé části ikonu s vykřičníkem. Toto značí upozornění, že blok má problém s místem vložení, více informací poskytuje kliknutí na ikonu.



Obrázek 7: Kontextové menu bloku

K přesunutí jednoho nebo více pospojovaných bloků, vybereme ten na vrcholu a přetáhneme na novou pozici. Rozpojování probíhá zespodu. Smazat blok můžeme přetažením na ikonu koše v dolní části pracovní plochy, stisknutím klávesy `Delete` nebo kliknutím pravým tlačítkem na blok a vybráním možnosti *Smazat blok* z kontextového menu. Menu pak dále poskytuje možnost blok duplikovat, sbalit, deaktivovat nebo zobrazit nápovědu. Deaktivace bloku změní barvu bloku

a v tomto stavu se daný blok neprojeví při spuštění programu. Pokud necháme kurzor na bloku, objeví se krátká nápověda k bloku.



Obrázek 8: Nápověda bloku

Klávesová zkratka	Funkcionalita
Ctrl + E	Nový program
Ctrl + O	Načtení programu
Ctrl + D	Stáhnutí programu
Ctrl + S	Dočasné uložení
Ctrl + Z	Zpět
Ctrl + Y	Znovu
Ctrl + C	Kopie bloku
Ctrl + X	Vyjmutí bloku
Ctrl + V	Vložit blok
Ctrl + =	Přiblížení
Ctrl + -	Oddálení
Ctrl + L	Spuštění programu
Ctrl + K	Zobrazení plátna
Delete	Smazání bloku
Esc	Zavření dialogového okna

Tabulka 2: Přehled klávesových zkratek

5.1.4 Panel záložek

V pravé části prostředí jsou tzv. záložky. Jedná se o Výstup, Plátno, Ladění, Nastavení a Nápověda. Po kliknutí na některou z nich se otevře boční panel s danou záložkou. Pro přepnutí se na jinou záložku na ni stačí kliknout, zavře se po kliknutí na aktivní záložku.

Záložka Výstup slouží k vypsání výstupu programu realizovanému blokem `vypiš`. K zobrazení výsledku musíme program spustit. Záložka Plátno nabízí

omezený pohled na plátno, kde se zobrazí výsledek po spuštění programu obsahující kreslící bloky. Celé plátno se zobrazí tlačítkem „Zobrazit plátno“ v horním menu. Záložka Ladění slouží ke sledování stavu aplikace během spuštěného ladění. V záložce nastavení můžeme nastavit, zda se nám program na pracovní ploše automaticky uloží lokálně do tzv. `localStorage` při každém spuštění. Pokud ano, při znovunačtení vývojového prostředí se tento program automaticky načte do pracovní plochy. Dalším nastavením je automatické otevírání bočního panelu při spuštění programu. Záložka Nápověda obsahuje přehled dostupných klávesových zkratek ve vývojovém prostředí a odkaz na plnou nápovědu.

5.1.5 Spuštění a ladění programu

Program lze spustit pomocí tlačítka *Spustit* v horním menu. Pokud se použije blok `vypiš` lze zobrazit výsledek v záložce Výstup. Pokud se použije některý z kreslících bloků, je výsledek zobrazen v záložce Plátno nebo pomocí tlačítka *Zobrazení plátna* v horním menu.

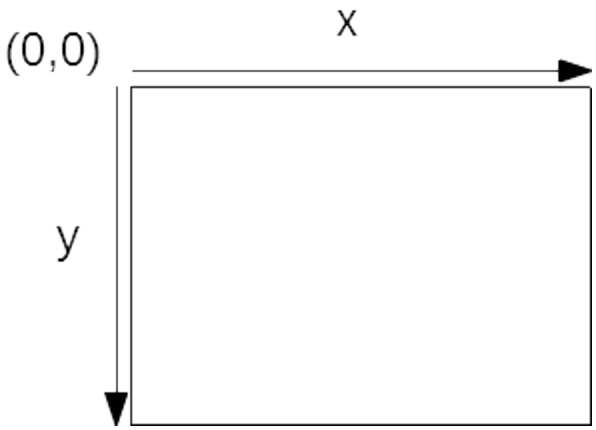
Ladění programu, na rozdíl od spuštění programu, nespustí program najednou. Místo toho se dá program tzv. krokovat. Ladění se spustí kliknutím na tlačítko *Spustit ladění*. Následně můžeme provádět kroky po jednotlivých blocích a to pomocí tlačítek *Krok vpřed* a *Krok zpět*. Při kliknutí na jedno z těchto tlačítek se provede buď další krok programu nebo krok předchozí. Aktuálně prováděný blok je označen světlejší barvou. Jestliže program využívá proměnné, jejich hodnoty se zobrazí pod ladícími tlačítky. Jsou vypisovány hodnoty pouze proměnných použitých v programu. Takto si lze projít celý program krok po kroku a podívat se, jak se stav programu mění. Pokud pracovní plocha neobsahuje žádné bloky, nelze nic ladit. Během ladění také nelze program spustit.

5.1.6 Plátno

Plátno je dvourozměrná mřížka pixelů. Každý tento bod má souřadnice `x` a `y`. Levý horní roh plátna má souřadnice `(0,0)`. Pro zobrazení výsledku na plátno je nutné program spustit.

Plátnu můžeme nastavit barvu pozadí blokem `nastavit pozadí`. Výchozí velikost plátna má šířku a výšku 400 pixelů. Ke změně velikosti slouží blok `nastavit šířku/výšku`. Pokud potřebujeme zjistit aktuální velikost plátna, použijeme blok `šířka/výška plátna`.

Pro posun na plátně máme několik bloků. Prvním způsobem je posun pomocí zadání souřadnic `x` a `y`. Dalším je posun o určitou délku a úhel. Úhel určíme hodnotou nebo výběrem z grafické nápovědy pro úhly. Poslední možnost posunu do obsahuje již předdefinované pozice. Aktuální souřadnice zjistíme pomocí bloku `x/y pozice`. Souřadnice, kam jsme se posunuli, se stávají aktuální pozicí. K ulehčení orientace, kde se nachází kurzor, slouží blok `aktuální pozice`, který na plátně zobrazí červený křížek. Výchozí pozice je střed plátna.

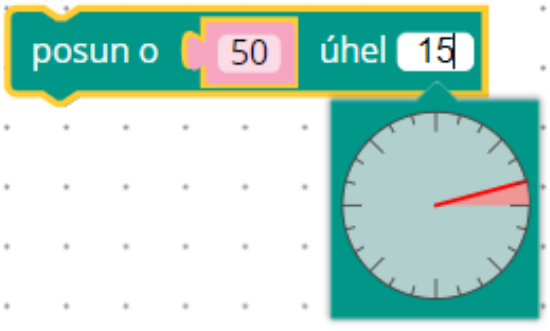


Obrázek 9: Souřadnicový systém plátna

5.1.7 Barvy a čáry

Nejjednodušší způsob jak získat barvu je výběr z palety. Vypadá jako červený obdélník. Pokud se na něj klikne, otevře se paleta barev. Barvu vybereme tak, že na požadovaný čtvereček s barvou klikneme. Dalším způsobem vytvoření barvy je pomocí bloku `náhodná barva`. Blok `vytvoř barvu` umožňuje uživateli zadat požadované procento množství červené, zelené a modré. Všechny hodnoty musí být mezi 0 a 100. Fialovou barvu vytvoříme maximálním množstvím červené a modré.

U čáry se dá nastavit několik vlastností. Jak bude čára široká, jakou bude mít barvu a jak ji zakončíme. Aby se tyto vlastnosti projevily na čáře, musí se všechny tyto vlastnosti nastavit před jejím vykreslením. Styl zakončení čáry je hranatý nebo zakulacený. Výchozí hodnotou je hranatý styl. Samotnou čáru vykreslíme pomocí bloku `čára` se zadanou délkou a úhlem. Úhel určíme buď hodnotou nebo vybereme z grafické nápovědy pro úhly.



Obrázek 10: Grafická nápověda pro zvolení úhlu

Blok `výplň` nastaví barvu, kterou budou vybarveny tvary, které kreslíme na plátno. Lze si to představit jako výběr pastelky předtím než začneme vybarvovat. Výchozí barva je černá.

5.1.8 Tvary

U každého tvaru existuje možnost volby, zda se vybarví nebo se kreslí pouze okraj. Tyto vlastnosti jsou ovlivněny bloky pro nastavení čáry a blok výplň. Pro změnu je nutné umístit tyto bloky před kreslení tvaru.

Obdélník potřebuje ke svému vykreslení šířku a výšku, kdy kreslení začíná od jeho levého horního rohu v místě, kde je aktuálně kurzor. Blok polygon umožňuje vytvořit z čar jeden útvar tak, že může být vybarven. Čar může být libovolný počet. Pokud například nakreslíme pouze 2 čáry, cesta nám automaticky spojí koncový bod druhé čáry s počátečním bodem první čáry a vznikne nám trojúhelník.

Kruh i oblouk se kreslí o zadaném poloměru, s tím, že střed je dán aktuální pozicí. Oblouk navíc potřebuje úhly od a do, které určují rozsah oblouku, a vykresluje se po směru hodinových ručiček.

Ke kreslení textu jsou k dispozici dva bloky. Jeden nastaví vlastnosti písma a druhý zajišťuje samotné vykreslení se zadaným textem. Výchozí hodnota pro velikost písma je deset pixelů a výchozím fontem je Arial.

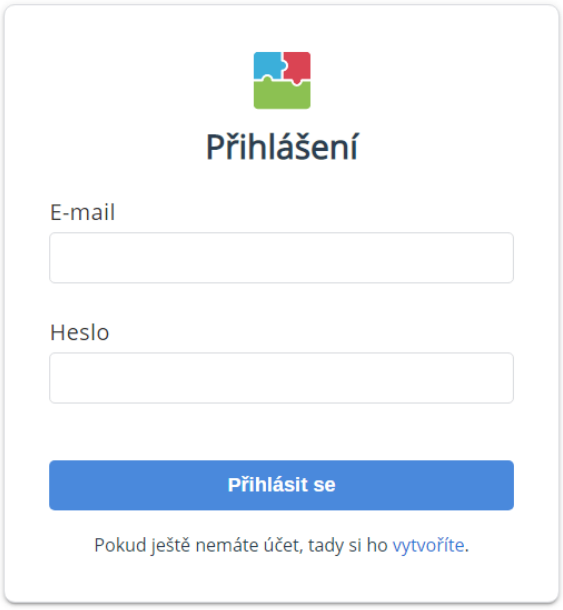
5.2 Aplikace

Aplikace Cubo slouží pro tvorbu a procvičování úkolů zaměřených na algoritmizaci a programování.

5.2.1 Registrace a přihlášení

Pro vytvoření účtu je nutné vyplnit všechna pole registračního formuláře, tedy přezdívku, e-mail, heslo a znovu zadání hesla. Po úspěšné registraci se může uživatel přihlásit.

Pro přihlášení je nutné zadat email vytvořeného účtu a heslo. Pokud jsou zadané údaje špatně, aplikace na to uživatele upozorní. Po úspěšném přihlášení se aplikace přesměruje na stránku *Přehled*.



Obrázek 11: Náhled přihlašovací stránky

5.2.2 Přehled

Obsahuje několik částí. První z nich je přehled počtu získaných bodů, dokončených úkolů a vytvořených kolekcí. Další části jsou seznamy s posledními pěti dokončenými úkoly a vytvořenými úkoly.

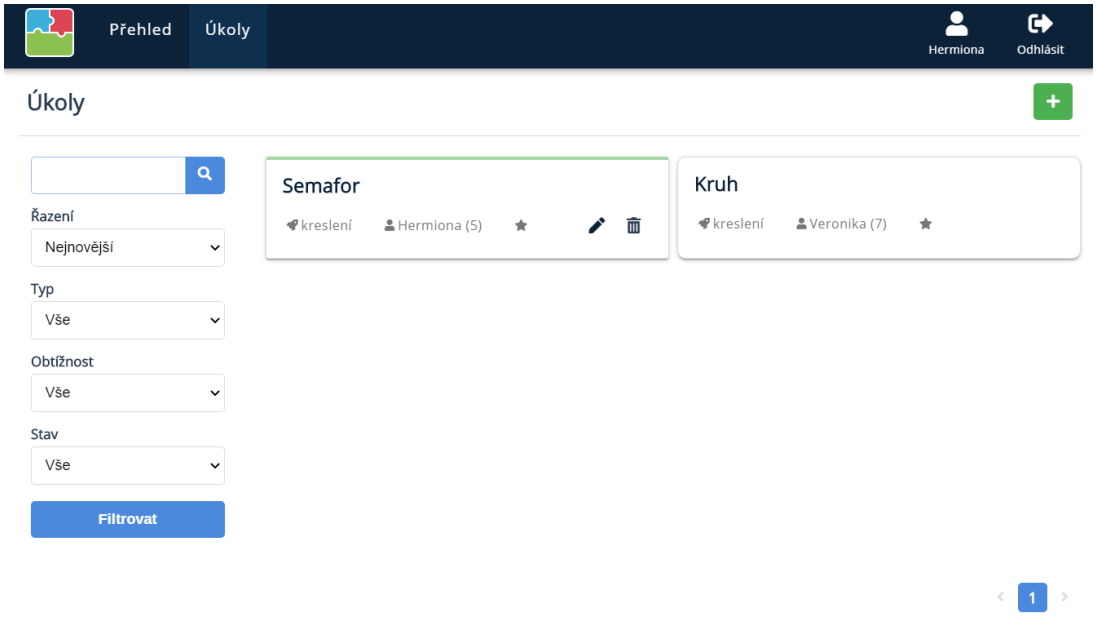
5.2.3 Nastavení uživatelského profilu

Pro změnu hesla je nutné nové heslo zadat dvakrát.V případě úspěšného změnění hesla se objeví dialogové okno se zprávou, že heslo bylo úspěšně změněno.

Po kliknutí na tlačítko *Zrušit účet*, se objeví dialogové okno. Pokud je deaktivace úspěšná, tak je uživatel přesměrován na úvodní stránku.

5.2.4 Úkoly

Každý úkol obsahuje název, typ a obtížnost. Typy úkolů jsou *Shoda*, *Kreslení* a *Funkce*. Podle vybraného typu záleží co se při tvoření nebo editaci předpokládá za akce nebo použité bloky. Zvolená obtížnost určuje počet získaných bodů. Lehká obtížnost přiděluje jeden bod, těžká pak tři. Úkoly lze filtrovat, řadit a vyhledávat podle názvu. Úkoly se nachází na stejnojmenné stránce. Pokud je přihlášený uživatel autorem úkolu, má možnost ho smazat nebo upravit pomocí ikon tužky a koše. K procvičení k úkolu se přistupuje kliknutím na název úkolu.



Obrázek 12: Náhled seznamu úkolů

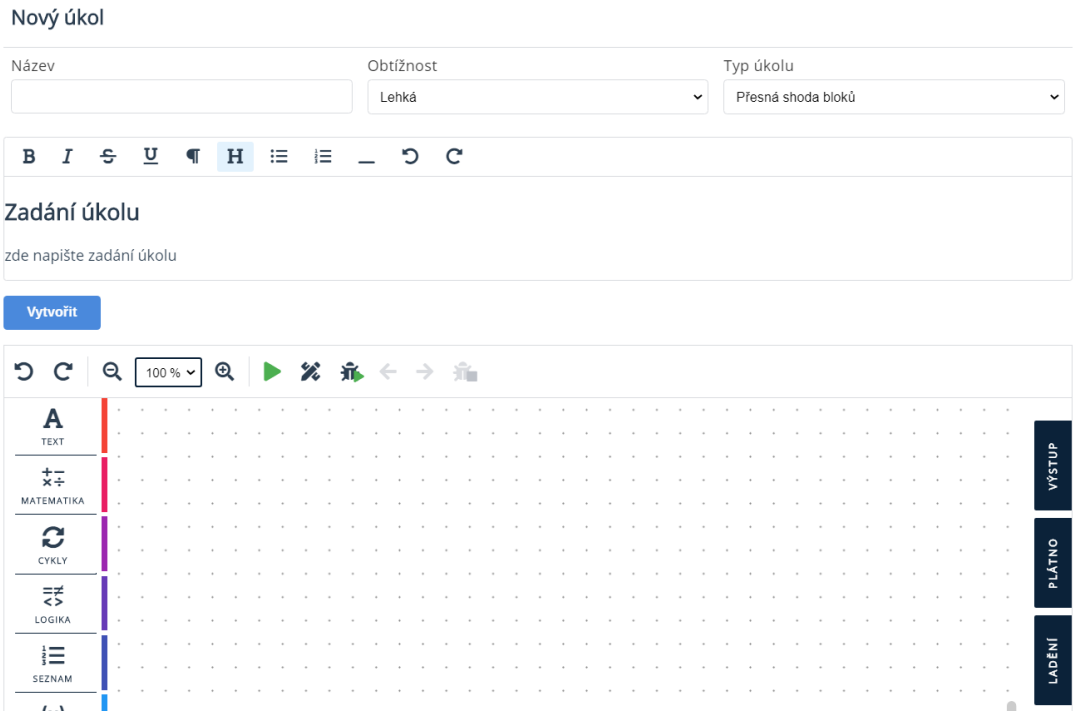
5.2.5 Vytvoření a editace úkolu

Pro vytvoření úkolu je nutné vyplnit název, typ, obtížnost a napsat zadání, co vlastně mají ostatní naprogramovat. Při uložení úkolu typu *Shoda* se jako řešení berou všechny bloky na pracovní ploše. Při *Kreslení* je nutné program alespoň jedenkrát spustit, jako výsledek se pro porovnání ukládá právě ten. Pokud použijeme typ *Funkce*, řešení se neuloží pokud plátno neobsahuje funkci s návratovou hodnotou a alespoň jedno její zavolání.

5.2.6 Procvičení úkolu

Procvičovaný úkol se skládá ze zadání, kontrolního tlačítka a vývojového prostředí. Postup řešení může být následující. Přečteme si zadání, řešíme úkol skládáním bloků na pracovní plochu a můžeme program libovolně spouštět nebo ladit. Až si budeme jistí svým řešením, klikneme na tlačítko Zkontrolovat. Kontrola úkolu se odvíjí jeho typu:

- **shoda** - nutné držet se zadání, pro splnění úkolu musí být shoda na 100%
- **funkce** - na pracovní ploše je po načtení úkolu vložena neúplná definice funkce, kterou je třeba doplnit tak, aby pro daný vstup vrátila výstup shodný s výstupem zadání
- **kreslení** - na záložce Plátno nebo po kliknutí na tlačítko Zobrazit plátno získáme předlohu toho co kreslit, kontroluje se obrázek za zadání s aktuálně vykresleným obrázkem na plátně

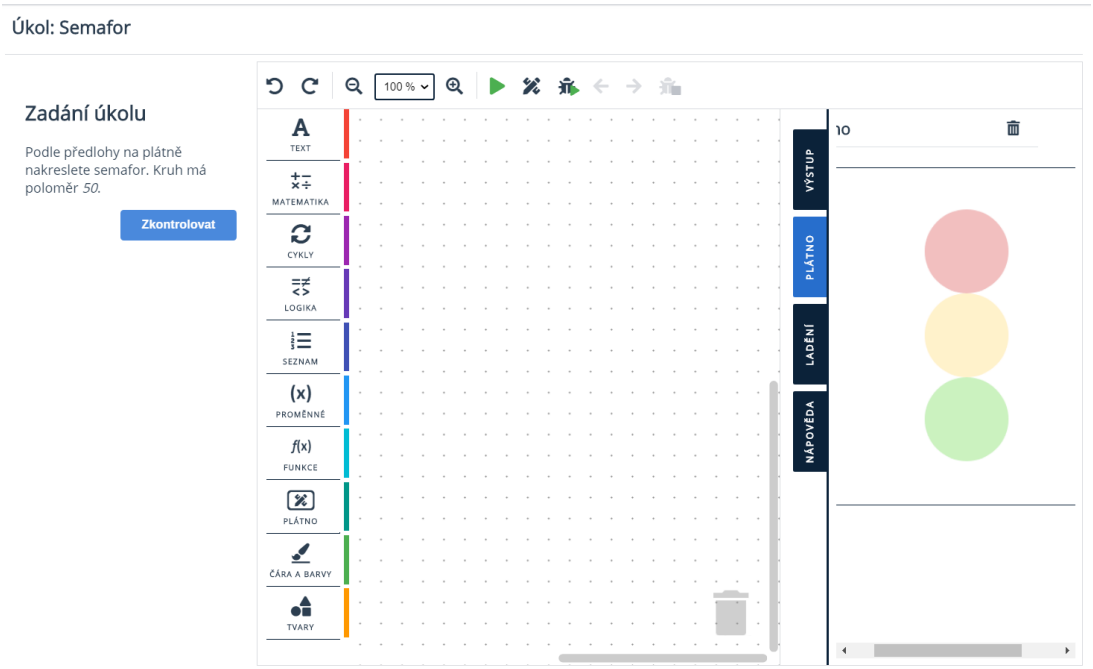


Obrázek 13: Náhled vytvoření/editace úkolu

Aplikace řešení zkontroluje se zadáním a dá zpětnou vazbu o úspěšnosti. V případě úspěchu se započtou body a jsme přesměrováni na seznam s úkoly. Za procvičení vlastního úkolu se body nepřičítají a neprojeví se ani v počtu dokončených úkolů. Několikanásobné procvičení se započítá pouze poprvé.

5.3 Zveřejnění

Aplikace z této diplomové práce byla zveřejněna na cloudové aplikační platformě Heroku [41] a to na adrese `https://kmi-cubo.herokuapp.com`. Jedná se o free plán (zdarma) a to s sebou nese určité omezení. Server je vypnut po třiceti minutách neaktivity, proto může být první spuštění (otevření webové stránky) pomalé.



Obrázek 14: Náhled procvičení úkolu

Závěr

Cílem této práce bylo navrhnout nástroj pro podporu výuky algoritmizace a programování u mladších studentů. Zároveň bylo potřeba navrhnout a vytvořit uživatelsky přívětivé vývojové prostředí s možností základního „ladění“ programů.

Prostudovala jsem si aktuální situaci kolem výuky informatiky a jaké existující řešení se dají používat. Nakonec se mi podařilo vytvořit vývojové prostředí postavené na knihovně Blockly, které je jednoduché na pochopení a používání. Celé prostředí je napsáno pomocí jazyka JavaScript s využitím knihovny Vue.js. Spouštění a ladění je realizováno přímo na klientovi. Nad tímto vývojovým prostředím funguje jednoduchá aplikace pro tvorbu a procvičování úkolů.

Co se týče možných vylepšení, je jich hned několik. Jedno z nich je mazání více bloků najednou nebo vytvoření bloků pro objektově orientované programování. Další vylepšení se týkají aplikace pro procvičování, zlepšení administrace, možnosti komentovat jednotlivá řešení nebo zařazování úkolů do skupin.

Conclusions

Goal of this thesis was to design tool for support teaching of algorithms and programming for younger students. Simultaneously it was necessary to design and create user friendy development environment with basic debugging functions.

I studied materials about current situation in the teaching of computer science and explored solutions that can be used. Finally, I created development environment based on the Blockly library that is easy to understand and use. The whole environment is written in JavaScript language using the Vue.js library. Running and debugging is performed directly on the client. Above this development environment works simple application for creating and practicing tasks.

There are several possible improvements. One is to delete multiple blocks at once or to create blocks for object-oriented programming. Other improvements include the application for practice, improved administration, the ability to comment on individual solutions or grouping tasks.

A Obsah přiloženého CD/DVD

Přiložené CD má následující strukturu:

doc/
: Text práce ve formátu PDF, vytvořený s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce, včetně všech příloh, a všechny soubory potřebné pro bezproblémové vygenerování PDF dokumentu textu (v ZIP archivu).

src/
: Kompletní zdrojové texty webové aplikace Cubo. Dostupné také na: https://bitbucket.org/VeronikaVas/diplomova_prace_cubo

readme.txt
: Instrukce pro nasazení webové aplikace Cubo na webový server, včetně všech požadavků pro její bezproblémový provoz, a webová adresa, na které je aplikace nasazena pro účel testování při tvorbě posudků práce a pro účel obhajoby práce.

U veškerých cizích převzatých materiálů obsažených na CD/DVD jejich zahrnutí dovolují podmínky pro jejich šíření nebo přiložený souhlas držitele copyrightu. Pro všechny použité (a citované) materiály, u kterých toto není splněno a nejsou tak obsaženy na CD/DVD, je uveden jejich zdroj (např. webová adresa) v bibliografii nebo textu práce nebo v souboru `readme.txt`.

Literatura

[1] KLEMENT, Milan. Možnosti rozšíření výuky algoritmizace a programování z pohledu žáků 9. tříd základních škol. *Media4u Magazine* [online]. 2018, roč. 3, č. 15 [cit. 2019-8-10], s. 24–32. Dostupný z: ⟨http://www.media4u.cz/mm032018.pdf⟩. ISSN 1214-9187.

[2] ČAPKOVÁ, Michaela. *Inovace RVP a ŠVP v závislosti na aktuálních trendech výuky ICT na ZŠ a víceletých gymnáziích [online]* [online]. [cit. 2019-8-10]. Dostupný z: ⟨https://theses.cz/id/vcbh06/%3E⟩.

[3] MŠMT. *Rámcový vzdělávací program pro základní vzdělávání* [online]. 2017 [cit. 2019-8-10]. Dostupný z: ⟨https://www.msmt.cz/file/43792/⟩.

[4] MŠMT. *RVP pro střední odborné vzdělávání* [online]. [cit. 2019-8-10]. Dostupný z: ⟨http://www.nuv.cz/t/rvp-os⟩.

[5] PRAZE, Výzkumný ústav pedagogický v. *Rámcový vzdělávací program pro gymnázia* [online]. [cit. 2019-8-10]. Dostupný z: ⟨http://www.nuv.cz/file/159⟩.

[6] IVA STUCHLÍKOVÁ, Tomáš Janík a kol. *Oborové didaktiky: vývoj - stav - perspektivy*. Brno: Masarykova univerzita, 2015. Dostupný také z: ⟨https://www.ped.muni.cz/didacticaviva/data_pdf/knihy/oborove-didaktiky_online.pdf⟩. ISBN 978-80-210-7884-0.

[7] REPUBLIKY, Vláda České. *Strategie Digitální Česko v. 2.0: Cesta k digitální ekonomice* [online]. [cit. 2019-8-10]. Dostupný z: ⟨https://www.vlada.cz/assets/media-centrum/aktualne/Digitalni-Cesko-v--2-0_120320.pdf⟩.

[8] MŠMT. *Strategie digitálního vzdělávání do roku 2020* [online]. 2014 [cit. 2019-7-7]. Dostupný z: ⟨http://www.vzdelavani2020.cz/images_obsah/dokumenty/strategie/digistrategie.pdf⟩.

[9] LESSNER, Daniel. *How do we translate computational thinking into Czech?* [online]. 2014 [cit. 2019-8-10]. Dostupný z: ⟨https://ksvi.mff.cuni.cz/~lessner/w/data/_uploaded/file/papers/2014_02_lessner_didactig.pdf⟩.

[10] MYŠLENÍ, Informatické. *O nás* [online]. [cit. 2019-8-10]. Dostupný z: ⟨https://imysleni.cz/o-projektu/o-nas⟩.

[11] *Informatické myšlení - Vzdělávací materiály.* [online]. [cit. 2019-8-12]. Dostupný z: ⟨https://imysleni.cz/ucebnice/⟩.

[12] *Bobřík informatiky.* [online]. [cit. 2019-8-9]. Dostupný z: ⟨https://www.ibobr.cz/⟩.

[13] CODE.ORG (ed.). *Hodina kódu* [online]. [cit. 2019-1-25]. Dostupný z: ⟨https://hourofcode.com/cz⟩.

[14] *O CodeWeeku.* [online]. [cit. 2019-8-9]. Dostupný z: ⟨https://codeweek.saferinternet.cz/⟩.

[15] *Europe Code Week*. [online]. [cit. 2019-8-9]. Dostupný z: ⟨https://codeweek.eu/⟩.

[16] GROUP, Lifelong Kindergarten (sest.). *Scratch* [online]. [cit. 2019-8-12]. Dostupný z: ⟨https://scratch.mit.edu/⟩.

[17] *ScratchJr*. [online]. [cit. 2019-8-12]. Dostupný z: ⟨https://www.scratchjr.org/⟩.

[18] WIKI, Scratch (sest.). *Scratch 3.0* [online]. [cit. 2019-8-12]. Dostupný z: ⟨https://en.scratch-wiki.info/wiki/Scratch_3.0⟩.

[19] DEVELOPERS, Google (sest.). *Blockly* [online]. [cit. 2019-8-12]. Dostupný z: ⟨https://developers.google.com/blockly/l⟩.

[20] FRASER, Neil (ed.). *Blockly Games* [online]. [cit. 2019-8-12]. Dostupný z: ⟨https://blockly-games.appspot.com⟩.

[21] HÁJKOVÁ, Miluše (ed.). *Ozoboti ve školství aneb programování hrou* [online]. [cit. 2019-8-12]. Dostupný z: ⟨https://spomocnik.rvp.cz/clanek/21588/OZOBOTI-VE-SKOLSTVI-ANEB-PROGRAMOVANI-HROU.html⟩.

[22] *Ozobot*. [online]. [cit. 2019-8-12]. Dostupný z: ⟨https://ozobot.com/educate⟩.

[23] SYSTEMS, SGP. *SGP Baltík* [online]. 2018 [cit. 2018-12-30]. Dostupný z: ⟨https://www.sgpsys.com/cz/⟩.

[24] RESEARCH, Microsoft (ed.). *Kodu Game Lab* [online]. [cit. 2019-8-10]. Dostupný z: ⟨https://www.kodugamelab.com/⟩.

[25] WIKIPEDIA, the free encyclopedia (sest.). *Kodu Game Lab* [online]. [cit. 2019-8-10]. Dostupný z: ⟨https://cs.wikipedia.org/wiki/Kodu_Game_Lab⟩.

[26] QUALIFIED.IO. *Codewars*. Dostupný z: ⟨https://www.codewars.com/⟩.

[27] DEVELOPERS, Google (sest.). *Blockly - Best Practices* [online]. [cit. 2019-8-12]. Dostupný z: ⟨https://developers.google.com/blockly/guides/app-integration/best-practices⟩.

[28] *Can I use... OffscreenCanvas*. [online]. [cit. 2020-5-25]. Dostupný z: ⟨https://caniuse.com/#feat=offscreencanvas⟩.

[29] WIKIPEDIA, the free encyclopedia (sest.). *Node.js* [online]. [cit. 2020-1-10]. Dostupný z: ⟨https://cs.wikipedia.org/wiki/Node.js⟩.

[30] *Express.js*. [online]. [cit. 2020-1-10]. Dostupný z: ⟨https://expressjs.com/en/guide/routing.html⟩.

[31] WIKIPEDIA, the free encyclopedia (sest.). *MongoDB* [online]. [cit. 2019-8-10]. Dostupný z: ⟨https://en.wikipedia.org/wiki/MongoDB⟩.

[32] *mLab*. [online]. [cit. 2019-8-10]. Dostupný z: ⟨https://docs.mlab.com/⟩.

[33] *Mongoose*. [online]. [cit. 2019-8-10]. Dostupný z: ⟨https://mongoosejs.com/docs/guide.html⟩.

[34] WIKIPEDIA, the free encyclopedia (sest.). *Representational state transfer* [online]. [cit. 2019-8-10]. Dostupný z: ⟨https://en.wikipedia.org/wiki/Representational_state_transfer⟩.

[35] *Passport.js*. [online]. [cit. 2020-1-10]. Dostupný z: ⟨http://www.passportjs.org/docs/authenticate/⟩.

[36] *Vue.js*. [online]. [cit. 2019-8-10]. Dostupný z: ⟨https://vuejs.org/v2/guide/⟩.

[37] *Vue Router Guide*. [online]. [cit. 2019-8-10]. Dostupný z: ⟨https://router.vuejs.org/⟩.

[38] *Vuex Guide*. [online]. [cit. 2019-8-10]. Dostupný z: ⟨https://vuex.vuejs.org/guide/⟩.

[39] *axios*. [online]. [cit. 2020-1-10]. Dostupný z: ⟨https://www.npmjs.com/package/axios/⟩.

[40] *Font Awesome Icons*. [online]. [cit. 2019-8-10]. Dostupný z: ⟨https://fontawesome.com/icons?d=gallery/⟩.

[41] *Heroku*. Dostupný z: ⟨https://www.heroku.com/⟩.

[42] WIKIPEDIA, the free encyclopedia (sest.). *Školní vzdělávací program* [online]. 2017 [cit. 2019-8-12]. Dostupný z: ⟨https://cs.wikipedia.org/wiki/%C5%A0koln%C3%AD_vzd%C4%9Bl%C3%A1vac%C3%AD_program⟩.

[43] *Blockly Wiki*. [online]. [cit. 2020-5-25]. Dostupný z: ⟨https://github.com/google/blockly/wiki⟩.