VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

WEBOVÁ APLIKACE PRO INTERAKTIVNÍ VÝUKU PROGRAMOVÁNÍ

WEB APPLICATION FOR INTERACTIVE PROGRAMING LECTURES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **ROMAN LÁTAL**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. ALENA OMACHTOVÁ**
SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

162698

Ústav:	Ústav počítačové grafiky a multimédií (UPGM)
Student:	**Látal Roman**
Program:	Informační technologie
Název:	**Webová aplikace pro interaktivní výuku programování**
Kategorie:	Uživatelská rozhraní
Akademický rok:	2024/25

Zadání:

1. Seznamte se ze základními principy tvorby webových aplikací.
2. Analyzujte požadavky na systém výuky programování jak ze strany studentů, tak ze strany učitelů. Ve výzkumu se soustřeďte i na existující řešení. Komentujte jejich nevýhody.
3. Navrhněte kurz algoritmizace včetně interaktivity a využití umělé inteligence, který bude sloužit studentům základních a středních škol. Řešení navrhněte i z pohledu učitele, který potřebuje mít přehled o svých studentech a jejich pokroku.
4. Implementuje navržené řešení.
5. Testuje řešení na cílové skupině uživatelů.
6. Vytvořte plakát nebo video reprezentující vytvořenou aplikaci.

Literatura:

- Steve Krug: Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability, ISBN: 978-0321965516, 2014
- Steve Krug: Rocket Surgery Made Easy: The Do-It-Yourself Guide to Finding and Fixing Usability, ISBN: 978-0321657299, 2009
- Joel Marsh: UX for Beginners: A Crash Course in 100 Short Lessons, O'Reilly 2016

Při obhajobě semestrální části projektu je požadováno:
1 až 3 a částečně rozpracovaný bod 4

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce:	**Omachtová Alena, Ing.**
Vedoucí ústavu:	Černocký Jan, prof. Dr. Ing.
Datum zadání:	1.11.2024
Termín pro odevzdání:	14.5.2025
Datum schválení:	12.11.2024

Abstrakt

Tato bakalářská práce řeší nedostatek interaktivních nástrojů pro výuku programování přizpůsobených potřebám českých středních škol. Popisuje návrh a implementaci webové aplikace StartCoder s využitím moderních technologií React, Next.js a Blockly pro vizuální programování. Aplikace nabízí interaktivní cvičení s okamžitou zpětnou vazbou, 3D vizualizaci výstupů, správu uživatelů a přehled pro učitele. Návrh vychází z analýzy existujících řešení a požadavků školství. Výsledkem je funkční aplikace, která poskytuje moderní a poutavou podporu pro výuku základů programování v českém školním prostředí.

Abstract

This bachelor's thesis addresses the lack of interactive tools for teaching programming tailored to the needs of Czech secondary schools. It describes the design and implementation of the web application StartCoder, using technologies such as React, Next.js, and Blockly for visual programming. The application offers interactive exercises with instant feedback, 3D visualization of outputs, user management, and an overview panel for teachers. The design is based on an analysis of existing solutions and the requirements of the educational system. The result is a functional application that provides modern and engaging support for teaching the basics of programming in the Czech school environment.

Klíčová slova

Interaktivní výuka programování, webová aplikace, vizuální programování, Blockly, React, Next.js, střední školy, e-learning, 3D vizualizace.

Keywords

Interactive programming education, web application, visual programming, Blockly, React, Next.js, high schools, e-learning, 3D visualization.

Citace

LÁTAL, Roman. *Webová aplikace pro interaktivní výuku programování.* Brno, 2024. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Alena Omachtová

Webová aplikace pro interaktivní výuku programování

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením **Ing. Aleny Omachtové**. Další informace mi poskytli **Ing. Petr Novosád, Ing. Josef Nevařil** a **Mgr. Kryštof Bogar**. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Roman Látal
8. května 2025

Poděkování

Rád bych na tomto místě vyjádřil své upřímné poděkování vedoucí mé bakalářské práce, **Ing. Aleně Omachtové**, za její cenné rady, odborné vedení, trpělivost a podporu během celého procesu tvorby práce. Její připomínky a konzultace byly pro úspěšné dokončení projektu klíčové.

Dále bych chtěl poděkovat vyučujícím ze Střední průmyslové školy a Obchodní akademie Uherský Brod, jmenovitě **Ing. Petru Novosádovi**, **Ing. Josefu Nevařilovi** a bývalému učiteli **Mgr. Kryštofu Bogarovi**, za jejich ochotu ke konzultacím, poskytnutí cenných pohledů z praxe a zpětné vazby k požadavkům na aplikaci z hlediska školního prostředí. Jejich odborná pomoc významně přispěla k relevanci a praktické využitelnosti výsledné aplikace.

Obsah

1 Úvod **4**

2 Vývoj webových aplikací **5**
2.1 Principy a komponenty 5
2.2 Porovnání technologií 6

3 Analýza existujících řešení **9**
3.1 Metody interaktivní výuky programování 9
3.2 Přehled konkurenčních nástrojů 10

4 Požadavky škol a osnov **16**
4.1 Nadcházející revize rámcových vzdělávacích programů 16
4.2 Požadavky Střední průmyslové školy a Obchodní akademie Uherský Brod (SPŠOA) 17
4.3 Persony 18
4.4 User Stories 21
4.5 User Flows 22

5 Návrh uživatelského rozhraní **25**
5.1 Úvodní obrazovka 25
5.2 Sekce kurzy 25
5.3 Cvičení kurzu 27
5.4 Přehled pro učitele 30
5.5 Databázový návrh 31
5.6 Rozhodování mezi 2D a 3D výstupem 32
5.7 Návrh implementace umělé inteligence 34

6 Implementace **35**
6.1 Výběr technologií 35
6.2 Architektura aplikace 37
6.3 Registrace a přihlášení 39
6.4 Cvičení 40
6.5 Administrační rozhraní pro učitele 49

7 Testování a nasazení **52**
7.1 Testování 52
7.2 Nasazení (Deployment) 53

8 Závěr **55**

Literatura **57**

Seznam obrázků

3.1 Uživatelské rozhraní při výuce na itnetwork. Obsahuje pouze text a obrázky bez jakýchkoliv interaktivních prvků. 11
3.2 Na pravé straně je v nástroji Google Lighthouse výčet prvků, které nesplňují doporučení kontrastu. 12
3.3 Uživatelské rozhraní při výuce na Skillmea. Obsahuje pouze videa bez interaktivních prvků. 13
3.4 Uživatelské rozhraní při výuce na FreeCodeCamp. Cvičení je interaktivní ale chybí čeština. 14

4.1 User flow registrace a přihlášení. Rozlišuje dvě strategie: email a SSO. . . . 23
4.2 User flow cvičení. Uživatel se snaží správně vykonat pokyny a pokud si neví rady, zeptá se chat bota nebo učitele. 24

5.1 Návrh úvodní stránky. 26
5.2 Návrh stránky s nabídkou kurzů. Aktuálně se zde nachází kurzy HTML, CSS, JavaScript, Python a algoritmizace. 27
5.3 Návrh cvičení kurzu. 28
5.4 V přehledu mají učitelé přehled o všech svých třídách společně s informacemi o pokroku studentů v jednotlivých kurzech 30
5.5 ER diagram databázového návrhu pro aplikaci StartCoder 31
5.6 Výsledek dotazníku o preferenci 2D nebo 3D 32
5.7 Důvody preference 2D nebo 3D 33

6.1 Implementovaná obrazovka registrace 39
6.2 Implementovaná obrazovka cvičení. 41
6.3 Implementovaná obrazovka administračního rozhraní pro učitele. 50

Kapitola 1

Úvod

V dnešní digitální době je schopnost programovat jednou z nejžádanějších dovedností napříč mnoha obory. Výuka programování se tak stává klíčovou součástí moderního vzdělávání, a to nejen na vysokých školách, ale stále častěji i na úrovni základních a středních škol. S rostoucí poptávkou po efektivních a srozumitelných výukových nástrojích roste i zájem o vývoj interaktivních řešení, která studentům umožní pochopit základní principy programování hravou a přístupnou formou.

Tato bakalářská práce se zabývá návrhem a implementací webové aplikace, která je určena pro interaktivní výuku programování. Cílem je vytvořit nástroj, jenž uživatelům poskytne intuitivní prostředí pro pochopení základních programátorských konceptů, a to formou praktických úloh a vizuální zpětné vazby. Webová aplikace má sloužit nejen jako výukový nástroj, ale také jako motivace pro studenty, kteří se s programováním teprve seznamují.

V úvodu práce je nejprve představena motivace a důvody pro vývoj této aplikace. Následuje kapitola o vývoji webových aplikací, dále analýza existujících řešení a jejich porovnání z hlediska použitelnosti, funkcionality a přístupnosti. Další kapitoly se věnují návrhu architektury aplikace, výběru technologií, samotné implementaci, testování a nasazení. Závěr práce hodnotí dosažené výsledky a naznačuje možné směry dalšího vývoje.

Kapitola 2

Vývoj webových aplikací

Moderní svět je neodmyslitelně spjat s webovými technologiemi. Webové aplikace, na rozdíl od statických webových stránek, poskytují dynamický obsah a umožňují komplexní interakci s uživatelem. Nacházejí uplatnění v široké škále oblastí, od elektronického obchodování a sociálních sítí až po specializované nástroje pro vzdělávání, jakým je i aplikace vyvíjená v rámci této práce. Tato kapitola stručně představí základní principy vývoje webových aplikací a porovná některé klíčové technologie používané při jejich tvorbě.

2.1 Principy a komponenty

Vývoj webové aplikace je komplexní proces zahrnující návrh, implementaci a údržbu softwaru dostupného prostřednictvím webového prohlížeče. Základními stavebními kameny jsou: [3]

Frontend (Klientská část) je část aplikace, se kterou přímo interaguje uživatel v prohlížeči, a která je zodpovědná za vizuální prezentaci (uživatelské rozhraní – UI) a interaktivitu (uživatelský zážitek – UX). Typicky se vytváří pomocí technologií HTML (struktura), CSS (vzhled) a JavaScript (dynamické chování).

Backend (Serverová část) je část aplikace, která tvoří logické jádro běžící na vzdáleném serveru. Zpracovává požadavky od klienta, komunikuje s databází, implementuje obchodní logiku a poskytuje data frontendu, často prostřednictvím API (Application Programming Interface).

API (Application Programming Interface) Rozhraní pro programování aplikací. Definuje soubor pravidel, protokolů a nástrojů, které umožňují různým softwarovým komponentám (typicky frontendu a backendu, ale i externím službám) vzájemně komunikovat. Frontend prostřednictvím API žádá backend o data nebo provedení akcí, backend odpovídá. Běžně se používají architektury jako REST (Representational State Transfer) nebo GraphQL a datové formáty jako JSON (JavaScript Object Notation).

Databáze Systém pro ukládání, správu a získávání dat aplikace. Může se jednat o relační (SQL) nebo nerelační (NoSQL) databáze.

Životní cyklus vývoje webové aplikace obvykle zahrnuje fáze jako analýza požadavků, návrh architektury a uživatelského rozhraní, implementace (kódování), testování, nasazení (deployment) na server a následná údržba a rozvoj. [7]

2.2 Porovnání technologií

Volba vhodných technologií je klíčovým faktorem ovlivňujícím efektivitu vývoje, výkon, škálovatelnost a udržovatelnost aplikace. Vzhledem k zaměření této práce se soustředíme na technologie relevantní pro moderní frontendový vývoj s možným přesahem do backendu a databází.

2.2.1 Frontend Frameworky a Knihovny

Pro zvládnutí komplexity moderních frontendových aplikací se běžně používají JavaScriptové frameworky nebo knihovny, které poskytují strukturu, znovupoužitelné komponenty a nástroje pro efektivní vývoj. Nejpopulárnější z nich jsou vypsané v seznamu níže. [20]

- **React:** Knihovna vyvíjená společností Meta (Facebook). Je založena na komponentovém přístupu a používá virtuální DOM pro optimalizaci výkonu. Disponuje obrovským ekosystémem a komunitou. Je základem pro framework *Next.js*, který byl zvolen pro tuto práci. [4]
- **Vue.js:** Progresivní framework známý svou jednoduchostí a flexibilitou. Umožňuje snadnou integraci do existujících projektů. Jeho popularita roste, nabízí dobrou dokumentaci a výkon. [8]
- **Svelte:** Kompilátor, který převádí komponenty na efektivní imperativní JavaScript již během sestavení (build time). Odstraňuje potřebu Virtual DOMu a režii frameworku v prohlížeči, což vede k menším balíčkům a potenciálně vysokému výkonu. Má vlastní integrovaný systém reaktivity. Rychle rostoucí alternativa s menším, ale aktivním ekosystémem. [10]

Volba *Reactu* (a následně Next.js) pro tuto práci byla motivována jeho flexibilitou, rozsáhlým ekosystémem, komponentovou architekturou usnadňující tvorbu interaktivních UI a silnou komunitní podporou.

2.2.2 Meta-Frameworky (SSR/SSG)

Pro řešení některých omezení čistě klientských aplikací (SPA – Single Page Applications), jako je SEO (optimalizace pro vyhledávače) a počáteční rychlost načítání, vznikly tzv. meta-frameworky postavené nad frontendovými knihovnami/frameworky. Zvažované frameworky jsou vypsané v seznamu níže.

- **Next.js (pro React):** Framework umožňující Server-Side Rendering (SSR) [19], Static Site Generation (SSG) [21], Incremental Static Regeneration (ISR) a další pokročilé funkce jako file-based routing a API routes. Poskytuje vynikající vývojářský zážitek a optimalizace výkonu. [13]
- **Nuxt.js (pro Vue):** Obdoba Next.js pro ekosystém Vue.js, nabízející podobné renderovací strategie a funkce. [17]
- **SvelteKit (pro Svelte):** Meta-framework pro Svelte, který kompiluje kód do efektivního imperativního JavaScriptu již během sestavení (build time). [6]

Next.js byl vybrán pro svou úzkou integraci s Reactem, robustní sadu funkcí pro SSR a SSG, které zlepšují výkon a SEO, a schopnost snadno vytvářet API endpointy pro backendovou logiku přímo v rámci projektu.

2.2.3 Styling (CSS)

Vzhled aplikace je definován pomocí CSS. Existuje několik přístupů. [5]

- **Čisté CSS/Sass/Less:** Tradiční přístup psaní stylů v samostatných souborech, případně s využitím preprocesorů jako Sass nebo Less pro proměnné, mixiny a další usnadnění.
- **CSS-in-JS:** Knihovny (např. Styled Components, Emotion) umožňující psát CSS přímo v JavaScriptových/TypeScriptových souborech, často navázané na komponenty.
- **Utility-First CSS (např. Tailwind CSS):** Framework poskytující velké množství předdefinovaných utilitních tříd, které se aplikují přímo v HTML/JSX kódu. Urychluje vývoj a zajišťuje konzistenci, ale může vést k zaplevelení HTML kódu třídami. *Tailwind CSS* byl zvolen pro tuto práci kvůli rychlosti vývoje, snadné customizaci a kolokaci stylů se strukturou webové stránky. [22]

2.2.4 Backendové technologie

Pokud aplikace vyžaduje komplexnější serverovou logiku nad rámec toho, co poskytují např. API routes v Next.js, je třeba zvolit backendovou technologii. Zvažované backendové technologie jsou vypsané v seznamu níže. [9]

- **Node.js (s frameworky jako Express, NestJS):** Umožňuje psát backend v JavaScriptu/TypeScriptu, což je výhodné pro týmy používající tyto jazyky i na frontendu. Je populární pro tvorbu API.
- **Python (s frameworky jako Django, Flask):** Oblíbený pro svou jednoduchost, čitelnost a rozsáhlé knihovny, často používaný pro datovou vědu, AI, ale i webové aplikace.
- **Java (Spring), C# (.NET), PHP (Laravel), Ruby (Rails)** a další jsou zavedené platformy s vlastními výhodami a ekosystémy.

Pro backendové potřeby této práce postačují *API routes* a *Server actions* v Next.js.

2.2.5 Databáze

Volba databáze závisí na struktuře dat a požadavcích aplikace. Základní dělení je na relační a nerelační databáze. [16]

- **Relační (SQL):** Např. PostgreSQL, MySQL, SQLite. Ukládají data do tabulek s pevně definovaným schématem a využívají jazyk SQL pro dotazování. Jsou vhodné pro strukturovaná data a zajištění konzistence (ACID transakce).
- **Nerelační (NoSQL):** Široká kategorie zahrnující databáze, které jsou vypsané v seznamu níže.
  - **Dokumentové (např. MongoDB):** Tyto databáze ukládají data ve formátu podobném JSON (dokumenty), schéma je flexibilní. Dobře se integrují s JavaScriptovými aplikacemi. **MongoDB** byla zvolena pro tuto práci pro svou flexibilitu a snadnou integraci s ekosystémem Node.js/Next.js. [14]

- **Key-Value (např. Redis):** Velmi rychlé pro jednoduché operace čtení/zápisu na základě klíče.
- **Grafové (např. Neo4j):** Optimalizované pro ukládání a dotazování na data s komplexními vztahy.

Volba konkrétní sady technologií (tzv. *technology stack*) pro webovou aplikaci je strategické rozhodnutí. Pro tuto práci byl zvolen moderní stack založený na Reactu, Next.js, Tailwind CSS, MongoDB, který umožňuje vytvořit interaktivní, výkonnou a vizuálně atraktivní vzdělávací aplikaci.

Kapitola 3

Analýza existujících řešení

Na trhu existuje řada nástrojů, které poskytují výuku programování různými způsoby. Každý z těchto nástrojů používá odlišné přístupy, jako jsou videa, live coding, interaktivní doplňování kódů, příklady implementace, slovní vysvětlení nebo dokumentace. Tato kapitola se zaměří na některé z těchto nástrojů, jejich výhody, nevýhody a na to, jakým způsobem by mohly inspirovat navrhovanou aplikaci.

3.1 Metody interaktivní výuky programování

Výukové platformy používají různé přístupy k tomu, aby studenti efektivněji pochopili programování.

- **Videa:** Využívání videí je efektivní přístup, protože studentům ukazuje, jak přemýšlí experti při psaní kódu. Pozorování živého kódování s komentářem zlepšuje pochopení procesu a umožňuje reflexi o rozhodnutích při programování. Toto využívají společnosti jako Coursera nebo Udemy. [1]
- **Live coding:** Umožňuje studentům vidět, jak se kód píše v reálném čase, což může přispět k lepšímu pochopení postupů. Například video streamy na Twitch.tv, kde uživatel vidí programování v realném čase a může se ptát vysílajícího na otázky.
- **Doplňování kódů:** Interaktivní úlohy, jako je doplňování chybějících částí kódu, posilují praktické dovednosti studentů. Strukturované přístupy, jako je PRIMM (Predict, Run, Investigate, Modify, Make), umožňují krokovou práci s kódem od predikcí po tvorbu vlastních projektů [1].
- **Příklady implementace:** Poskytují konkrétní příklady toho, jak se teorie přenáší do praxe, a umožňují studentům přímou aplikaci získaných znalostí.
- **Slovní vysvětlení:** Podrobná vysvětlení jednotlivých konceptů nebo bloků kódu, která studentům pomáhají pochopit logiku za jednotlivými kroky.
- **Dokumentace:** Nástroje, které studentům poskytují přístup k oficiálním dokumentacím programovacích jazyků, což jim umožňuje samostatně se učit a prohlubovat znalosti.

Na trhu je již vícero řešení, které učí programování. Problémem je, že se žádné existující řešení nezaměřuje na školství, což je hlavním cílem této práce. Příklady problémů konkurence jsou vypsané v seznamu níže.

- **Vysoká cena** – zejména v případě offline kurzů ale také při online webinářích.
- **Absence českého jazyka** – spousta kvalitních kurzů nedisponuje češtinou, což učiní software nepoužitelný pro spoustu Čechů a tím i škol.
- **Nezaměření na školy** – konkurence se ve většině případů zaměřuje na veřejnost, občasně i společnosti ale už ne na školy.
- **Nepřívětivé uživatelské rozhraní** – neintuitivní uživatelské rozhraní je bohužel stále velký problém zejména na českém trhu.
- **Nutnost fyzické přítomnosti** – zbytečná bariéra pro spoustu lidí mimo metropolitní oblasti. Nutnost fyzické přítomnosti je také často úzce spjaté s vysokou cenou.

Tato kapitola bude tedy věnována konkrétním konkurentům, kde budou nejprve představeni a následně rozebrány jejich výhody a nevýhody.

3.2 Přehled konkurenčních nástrojů

V této části práce analyzujeme existující platformy pro výuku programování, které lze považovat za přímou konkurenci navrhované aplikace. Cílem této analýzy je identifikovat jejich silné i slabé stránky, zejména z pohledu interaktivity, dostupnosti v českém jazyce, přizpůsobení vzdělávacím potřebám škol a celkové uživatelské zkušenosti. Na základě této analýzy bude možné lépe definovat klíčové vlastnosti a hodnoty nové platformy, aby odpovídala specifickým požadavkům českého školství a poskytla efektivnější cestu k osvojování programátorských dovedností.

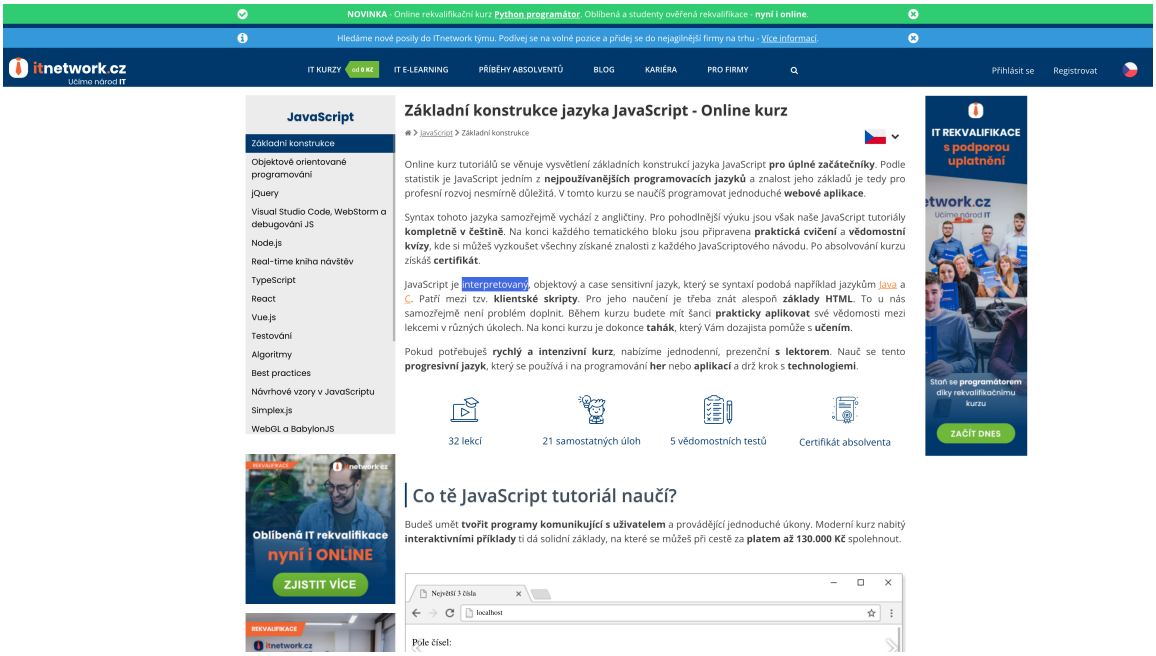
3.2.1 ITnetwork

Prvním a asi největším kompetitorem je právě itnetwork[1]. Itnetwork poskytuje jak offline tak i online kurzy programování. Je to největší česká firma poskytující výuku programování a je na trhu již dlouhou dobu.

Výhody

- **Výhoda prvního tahu** – protože jsou na trhu již dlouhou dobu a jsou jedním z předních výsledků ve vyhledávačích, tak ovládají většinu českého trhu v oblasti výuky programování.
- Mají **velké množství kurzů**. Toto je spojené s dobou na trhu, ale stojí za zmínku, že disponují spoustou kurzů. Tyto kurzy zahrnují jak konkrétní jazyky (HTML, JavaScript, SQL), tak i kurzy, kterých cílem není člověka naučit konkrétní jazyk, ale konkrétní pozici (Front-end developer, Vývojář mobilních aplikací).
- Jsou **propojení s firmami**. Poskytují absolventům kurzů nabídky prací vycházející z jejich spolupráce s konkrétními společnostmi.

[1] Dostupné na: https://www.itnetwork.cz/



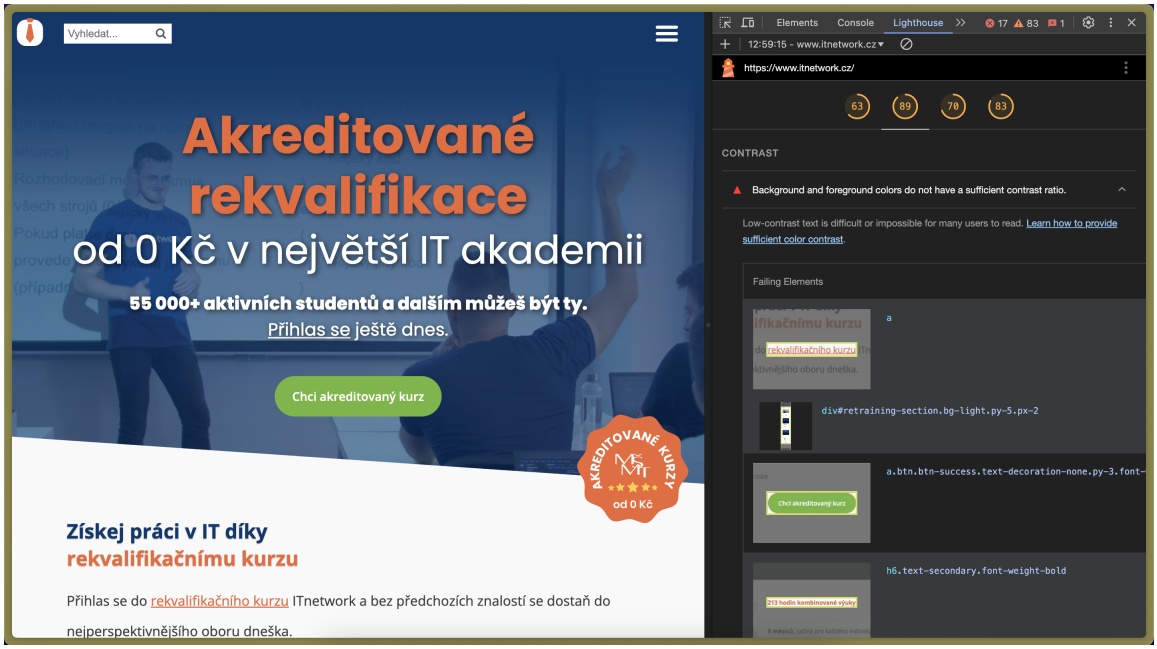
Obrázek 3.1: Uživatelské rozhraní při výuce na itnetwork. Obsahuje pouze text a obrázky bez jakýchkoliv interaktivních prvků.

Nevýhody

- **UI působí zastarale** a není konzistentní, jak jde vidět na obrázku 3.1. Konzistence je důležitá, jelikož usnadňuje uživatelům navigaci a používání systému díky eliminaci nutnosti neustálého učení se novým vzorcům chování. To podporuje plynulejší uživatelské zážitky a efektivnější práci s aplikací [23]. Příklady špatných designových prvků jsou vypsané v seznamu níže.
  - **Příliš mnoho barev**. Hned na úvodní stránce jsou barvy: modrá, zelená, oranžová, bílá, šedá, žlutá. V dnešní době, kdy převládá minimalizmus, většinou chceme mít v návrhu 2-4 barvy, kde dvě z těchto barev jsou bílá a černá, abychom zachovali jednoduchost a konzistenci brandingu.
  - **Mix zaoblení prvků**. Tlačítka ve stránce a ikony jsou zaoblené přičemž obrázky, sekce, vstupy a tlačítka na navigačním baru nejsou.
  - **Nekonzistentní odsazení prvků**. Některé prvky jsou bezdůvodně odsazeny více zleva než jiné.
  - **Špatný kontrast**. Dle následujících směrnic *WCAG 2.1 (AA), WCAG 2.0 (AA), WCAG 2.2 (AA), Trusted Tester, EN 301 549* [2] je doporučován na malém textu kontrast alespoň 4.5:1 a na velkém textu alespoň 3:1, protože text s malým kontrastem je obtížné nebo nemožné pro mnoho uživatelů přečíst. Podle nástroje Google Lighthouse toto jen na hlavní stránce nesplňuje více než 50 prvků (obrázek 3.2).
- **Online výuka z velké části není interaktivní**. Většina výuky probíhá jako čtení knihy a až na konci lekce je kvíz. To může vést k tomu, že studenti nerozvíjejí praktické dovednosti, které by odpovídaly požadovaným znalostem. Pro efektivní vzdělávání je

zásadní zapojit studenty do interaktivních aktivit, jako jsou projekty a programovací úkoly, které mohou zlepšit rozvoj praktických dovedností až o 30 % oproti pouhému pasivnímu učení [18].

- **Absence navedení uživatele ke kurzu**. Software nepomůže novému uživateli zvolit správný kurz pro něj.
- **Absence kurzu algoritmizace pro mladší** generaci, pro které by mohlo být obtížné hned pochopit řádkové programování.
- **Nutnost nastavit si vlastní vývojové prostředí**. Pro kurz Pythonu je hned v první lekci požadováno stáhnout a nainstalovat PyCharm a Python, což může být bariéra pro uživatele, kteří si pouze chtějí vyzkoušet, zda by je programování bavilo.
- **Software není navržen s ohledem na potřeby vzdělávacího sektoru**, které jsou podrobně popsány v kapitole 4.



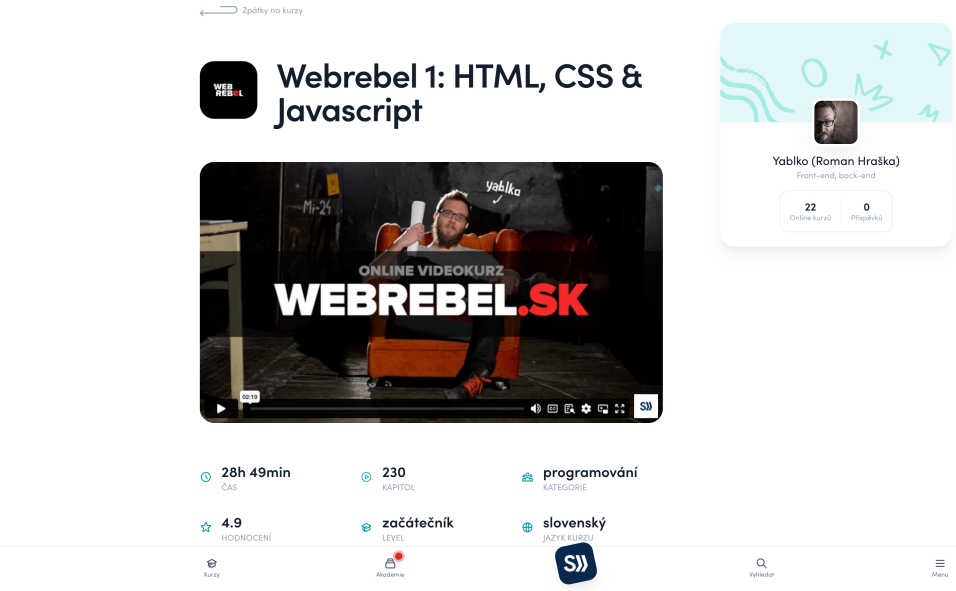
Obrázek 3.2: Na pravé straně je v nástroji Google Lighthouse výčet prvků, které nesplňují doporučení kontrastu.

3.2.2 Skillmea

Další velká konkurence je Skillmea[2]. Skillmea je platforma, která poskytuje lidem možnost nahrát své vlastní kurzy ve formě videí. Na jednu stranu je to výhodné, protože se společnost nemusí zabývat vytvářením kurzů a může se plně soustředit na vylepšování platformy. Na druhou stranu ale musí kontrolovat obsah kurzů, musí si dělit příjem s tvůrci a nikdo nezaručí, že na sebe budou kurzy navazovat. To znamená, že se často můžou informace

[2]Odkaz na Skillmea: https://www.freecodecamp.org/

opakovat nebo také, že můžou nějaké informace chybět k tomu aby se z člověka stal programátor. Dříve se společnost jmenovala Learn2Code, ale expandovala a už nenabízí jen kurzy programování ale i kurzy marketingu, designu, jazyků a jiné.



Obrázek 3.3: Uživatelské rozhraní při výuce na Skillmea. Obsahuje pouze videa bez interaktivních prvků.

Výhody

- **Velké množství kurzů** – Podobně jako ITnetwork disponují spoustou kurzů, ale narozdíl od kurzů ITnetwork jsou tyto kurzy více izolované. To znamená, že zde nejsou kurzy, které z nového programátora udělají front-end programátora nebo developera mobilních aplikací. Uživatel si musí zvolik konkrétní jazyky nebo koncepty, co se chce naučit.
- **Moderně působící UI** – UI člověka zaujme a je intuitivní na používání. UI je vyobrazeno na obrázku 3.3.

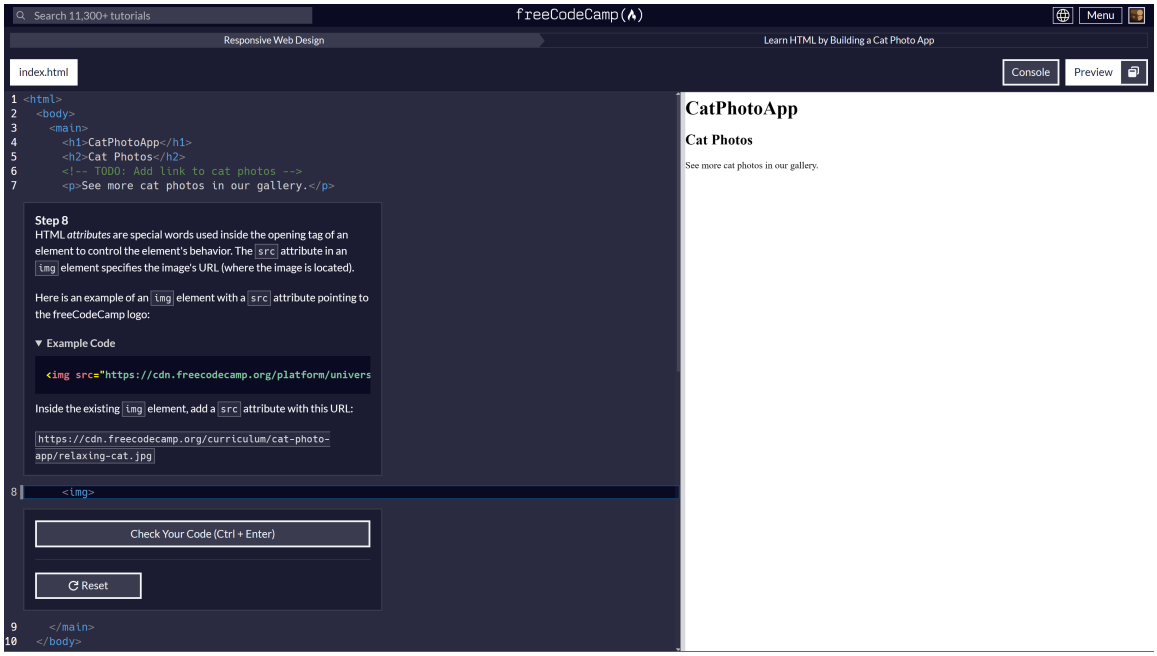
Nevýhody

- **Kurzy jen ve formě videí** – video formát nemusí všem vyhovovat. Zejména při programování, kdy se často hodí něco zkopírovat.
- **Výuka pomocí videí není interaktivní** – Uživatel nikdy nezjistí, zda je jeho kód správný, kromě kontrolování jednotlivých znaků z videa. Toto potlačuje kreativitu, jelikož při psaní odlišného kódu od videa nebude uživatel mít žádnou zpětnou vazbu, zda je jeho kód správný.
- **Žádné navedení uživatele ke kurzu** – Podobně jako u ITnetwork Skillmea nepomůže novému uživateli zvolit správný kurz pro něj.

- Opět **neobsahuje žádný kurz algoritmizace pro mladší** pro které by mohlo být obtížné hned pochopit řádkové programování.
- I u platformy Skillmea **je nutné nastavit si vlastní vývojové prostředí**.
- **Poměrně drahé předplatné** – bez předplatného si uživatel nemůže ani vyzkoušet výuku a předplatné má relativně vysokou cenu: 3499 Kč kvartálně nebo 6999 Kč ročně.
- **Software není navržen s ohledem na potřeby vzdělávacího sektoru**. Potřeby vdělávacího sektoru jsou detailně popsány v kapitole 4.

3.2.3 FreeCodeCamp

Velkým konkurentem zejména v zahraničí je právě FreeCodeCamp[3]. FreeCodeCamp je nezisková organizace, která poskytuje kurz programování zdarma, kde je uživatel proveden kompletní cestou od web designu přes databáze až k machine learningu v pythonu.



Obrázek 3.4: Uživatelské rozhraní při výuce na FreeCodeCamp. Cvičení je interaktivní ale chybí čeština.

Výhody

- **Návaznost kurzů** – Začíná se s HTML, CSS, ve druhém kurzu se přidá JavaScript, ve třetím kurzu se přejde na React, dále na databáze, backend development atd. FreeCodeCamp má velmi holistický a koherentní přístup k výuce programování.
- **Velmi interaktivní** – v každé části se uživatel něco naučí a je požadováno aby uživatel splnil určitý krok, který přispěje k finálnímu výsledku.

[3]Odkaz na FreeCodeCamp: https://www.freecodecamp.org/

- **Projekty** – vše je vyučováno takovým způsobem, že uživatel pracuje na projektu, který po skončení kurzu naprogramoval celý sám.
- **Cena** – celý FreeCodeCamp je zdarma.
- **Uživatel je naveden, kde začít.**
- **Dobře vyřešené uživatelské rozhraní** – Rozhraní působí relativně moderně a je pro člověka intuitivní (viz obrázek 3.4).

Nevýhody

- **Neobsahuje kurz algoritmizace**.
- **Není v češtině**.
- **Software není navržen s ohledem na potřeby vzdělávacího sektoru.**

Na základě analýzy výše uvedených řešení lze konstatovat, že ačkoli existuje mnoho platforem pro výuku programování, žádná z nich neodpovídá plně požadavkům školství, které je zaměřeno na systematické a interaktivní vzdělávání studentů. Největší problémy konkurentů zahrnují nedostatečné zaměření na školní prostředí, vysoké ceny, absenci češtiny a často neinteraktivní výukové metody. To poukazuje na nutnost vývoje nové webové aplikace, která by se těmto výzvám přizpůsobila, přinášela přívětivé uživatelské prostředí a efektivní výukové postupy, a zároveň se zaměřila na specifické potřeby škol a studentů v České republice.

Kapitola 4

Požadavky škol a osnov

Vzdělávací instituce a jejich osnovy hrají klíčovou roli v utváření znalostí a dovedností studentů v oblasti programování. Každá škola má specifické požadavky, které odrážejí jak celonárodní standardy, tak potřeby dané školy a jejích studentů. V kontextu výuky programování je nezbytné zajistit, aby výukové materiály a nástroje odpovídaly nejen teoretickým požadavkům, ale také praktickým dovednostem, které studenti v dnešní době potřebují. Tato kapitola se zaměří spíše na nadcházející revizi rámcových vzdělávacích programů, která má začít platit od roku 2025. Rámcové vzdělávací programy a požadavky škol ovlivňují návrh a vývoj webové aplikace pro interaktivní výuku programování.

4.1 Nadcházející revize rámcových vzdělávacích programů

Nadcházející revize rámcových vzdělávacích programů jsou zaměřeny zejména na digitalizaci. Jedním z hlavních cílů je do výuky zakomponovat práci s moderními digitálními technologiemi a zvyšování digitálních kompetencí u žáků i samotných učitelů. [15]

Revize rámcových vzdělávacích programů (RVP) je zásadní krok koordinovaný Ministerstvem školství, mládeže a tělovýchovy (MŠMT). Tyto revize reflektují aktuální potřeby společnosti a trhu práce, včetně požadavků na digitální gramotnost a schopnost pracovat s algoritmickým myšlením. V kontextu výuky informatiky mají za cíl zajistit, aby studenti získali dovednosti potřebné pro moderní digitální svět.

Souvislost s MŠMT spočívá v tom, že právě toto ministerstvo stanovuje rámcové vzdělávací programy jako základní kurikulární dokumenty pro základní a střední školy. Revize prováděné MŠMT často zahrnují zapojení odborníků na vzdělávání, pedagogů a dalších zainteresovaných stran. Cílem je zajistit, aby byly vzdělávací cíle aktuální a reflektovaly dynamické změny, například ve vztahu k programování, algoritmizaci a schopnosti analyzovat a řešit problémy, což přesně odpovídá uvedeným požadavkům na žáky.

Na základě revize rámcových vzdělávacích programů by měl být žák schopen:

- na základě analýzy problému **specifikuje zadání** pro tvorbu programu, skriptu nebo webové aplikace,
- **rozdělí zadání nebo problém na menší části**, rozhodne, které je vhodné řešit algoritmicky, své rozhodnutí zdůvodní,
- **navrhne algoritmy a datové struktury** podle specifikace zadání a zapíše je vhodnou formou,

- ve vztahu k charakteru a velikosti vstupu **hodnotí algoritmy a datové struktury** podle různých hledisek, **porovná** a **vybere** pro řešený problém ty nejvhodnější; vylepší algoritmus podle daného hlediska,
- **vytvoří jednoduchý spustitelný program**, skript, nebo webovou aplikaci,
- **testuje spustitelný program**, skript nebo webovou aplikaci; najde, specifikuje a opraví případnou chybu,
- **spolupracuje při tvorbě programu s další osobou**, popisuje strukturu programu další osobě.

Z výše uvedených požadavků je patrné, že výuka programování na středních školách bude v nadcházejících letech stále více zaměřena na rozvoj algoritmického myšlení, schopnost vytvářet efektivní informatická řešení a na využití digitálních technologií v praktických úlohách. Revize rámcových vzdělávacích programů reflektuje rostoucí důležitost digitálních kompetencí, což ovlivní nejen obsahovou náplň výuky, ale také potřebu moderních výukových nástrojů. Webová aplikace pro interaktivní výuku programování musí tyto nové požadavky zohlednit, aby efektivně podporovala učitele při výuce a poskytovala studentům prostor pro rozvoj těchto klíčových dovedností.

4.2 Požadavky Střední průmyslové školy a Obchodní akademie Uherský Brod (SPŠOA)

SPŠOA je významnou institucí, která se dlouhodobě zaměřuje na technické vzdělávání. Vzhledem k mé osobní zkušenosti s touto školou, kde jsem sám studoval, a probíhající komunikaci od počátku projektu, se její požadavky staly důležitým vodítkem při vývoji webové aplikace pro interaktivní výuku programování. V současné době každý učitel vyučuje jiným způsobem, kde některý učitel využívá online nástroje na kompilaci a interpretaci kódu jako *Programiz.com*, přičemž někteří využívají standardní prostředí na uživatelském zařízení, jako jsou například integrovaná vývojová prostředí (IDE) **Visual Studio** nebo textové editory typu **Visual Studio Code** v kombinaci s kompilátory či interprety spouštěnými z příkazové řádky. Pro výuku konceptů momentálně slouží prezentace a výklad učitele. Toto není vhodné, jelikož po delším výkladu si už většina studentů nepamatuje syntaxi a význam všech klíčových slov, které se v prezentaci objevily. Tyto informace jsou zejména od učitelů Ing. Petra Novosáda, Ing. Josefa Nevařila a bývalého učitele Mgr. Kryštofa Bogara.

Škola klade důraz na několik klíčových aspektů, které jsou uvedeny v seznamu níže.

- **Kurz algoritmizace** – Jedním z hlavních požadavků je zařazení kurzu zaměřeného na algoritmizaci. Algoritmické myšlení je považováno za základní dovednost, kterou by měl každý student technického směru ovládat. Kurz algoritmizace by měl pomoci studentům pochopit základy struktury programů a řešení problémů pomocí logických postupů.
- **Český jazyk** – Pro studenty i učitele je zásadní, aby byla aplikace dostupná v českém jazyce. Tento požadavek vychází nejen z jazykových potřeb studentů, ale i z požadavků školního kurikula. Absence češtiny by mohla způsobit problémy při výuce, zejména pro studenty, kteří nemají vysokou úroveň znalosti angličtiny.

- **Dostupnost na webových stránkách** – Aplikace musí být dostupná online, což umožní studentům i učitelům přístup odkudkoli, bez nutnosti instalace softwaru. Tento požadavek zajišťuje flexibilitu ve výuce a přístupnost pro všechny uživatele, včetně těch, kteří pracují z domova nebo z jiných lokalit.

Škola rovněž plánuje další rozvoj výukových materiálů a metod. Tyto informace vycházejí zejména z rozhovorů s panem učitelem Josefem Nevařilem. Tyto požadavky jsou uvedeny níže.

- Kurz C++ – C++ je jedním z klíčových programovacích jazyků v technickém vzdělávání. Škola si klade za cíl rozšířit výuku tohoto jazyka a integrovat jej do výukové aplikace. Kurz by měl obsahovat nejen základní syntax a strukturu jazyka, ale také pokročilejší techniky, které studentům umožní vyvinout komplexní programy.
- Uživatelské prostředí pro učitele – Jedním z plánovaných vylepšení je vytvoření administračního rozhraní pro učitele. Toto rozhraní by mělo umožnit sledovat pokrok studentů v reálném čase, zobrazovat jejich výsledky v jednotlivých kurzech, a umožnit jim zasahovat do procesu výuky. Učitelé by tak mohli efektivněji hodnotit pokroky studentů a cíleněji upravovat výukové postupy.

Požadavky SPŠOA jsou velmi důležité v určování směru, kterým by se měla aplikace pro interaktivní výuku programování ubírat. Tyto požadavky, zaměřené na algoritmizaci, výuku v českém jazyce a dostupnost online, vytvářejí solidní základ pro efektivní využití aplikace v praxi. Plánované rozšíření o kurz C++ a administrační rozhraní pro učitele představují další kroky k tomu, aby se aplikace stala ještě účinnějším nástrojem ve vzdělávání studentů na technických školách.

4.3 Persony

Při návrhu jakéhokoliv produktu je užitečné definovat různé typy uživatelů tzv. persony, které reprezentují typického uživatele s různými přístupy k učení a odlišnými motivacemi. Následující persony byly vytvořeny pro potřeby této aplikace: premiant, průměrný student a pomalejší student. Tyto persony reprezentují typy studentů technicky zaměřených středních škol.

4.3.1 Premiant

- **Jméno:** Jan Novák.
- **Věk:** 18 let.
- **Role:** student technické střední školy.
- **Cíle:**
  - Jan chce získat co nejvíce znalostí o programování a algoritmizaci, protože chce mít dobrou kariéru jako programátor,
  - Jan chce předstihnout výukový plán a tvořit samostatné projekty, protože ho baví programování,

  - Jan se tímto způsobem chce připravit se na vysokou školu nebo prestižní technickou pozici.

- **Chování a přístup k výuce:**
  - aktivně využívá všechny možnosti aplikace,
  - pravidelně testuje své znalosti a hledá další zdroje učení,
  - využívá fórum a konzultace při řešení problémů.

- **Potřeby:**
  - Jan chce zdolat pokročilé výzvy a úlohy, aby věděl, zda látku opravdu ovládá,
  - Jan potřebuje rychlou zpětnou vazbu a interaktivní testy, aby ihned věděl, zda má vše správně.

- **Motivace:** být nejlepší ve třídě a překonávat sám sebe.

4.3.2 Průměrný student

- **Jméno:** Eva Novotná.
- **Věk:** 17 let.
- **Role:** studentka technické školy (2. ročník).
- **Cíle:**
  - Eva chce splnit požadované učivo, aby byla připravena na maturitu,
  - Eva si chce udržet dobrý prospěch, aby ji přijali na vysokou školu.

- **Chování a přístup k výuce:**
  - pravidelně se zapojuje, ale málokdy jde nad rámec osnov,
  - při složitějších tématech vyžaduje více času a podporu učitele,
  - při nejasnostech využívá dokumentaci a nápovědu v aplikaci.

- **Potřeby:**
  - Eva potřebuje srozumitelné materiály s jasným postupem, aby se v látce neztratila,
  - Eva potřebuje interaktivní cvičení s okamžitou zpětnou vazbou,
  - Eva potřebuje možnost vracet se k obtížnějším tématům v případě, že by se na ně chtěla zaměřit později.

- **Motivace:** získat jistotu ve svých schopnostech a úspěšně složit testy.

4.3.3 Pomalejší student

- **Jméno:** Martin Dvořák.
- **Věk:** 19 let.
- **Role:** student technické školy (maturant).
- **Cíle:**
  - Martin chce úspěšně splnit povinné úkoly a cvičení z programování, aby prošel předmětem a mohl dokončit školu,
  - Martin chce porozumět alespoň základním konceptům natolik, aby dokázal vyřešit požadované úlohy, i když mu programování celkově připadá složité.
- **Chování a přístup k výuce:**
  - potřebuje více času na pochopení nových konceptů a zadání úkolů,
  - může působit nepozorně, protože má problém sledovat rychlejší výklad nebo složitější vysvětlení,
  - snadno se zasekne na dílčím problému a může být frustrovaný, pokud nenajde rychlou pomoc nebo jasné vysvětlení,
  - preferuje práci podle jasných návodů a jednoduchých příkladů,
  - často potřebuje opakování a návrat k již probrané látce.
- **Potřeby:**
  - velmi srozumitelné, stručné a jasně strukturované teoretické úvody a zadání úkolů, rozdělené na malé, zvládnutelné kroky,
  - jednoduché příklady, na kterých si může ověřit základní pochopení, než přejde ke složitějším úlohám,
  - okamžitou a jednoznačnou zpětnou vazbu, která potvrdí, zda dílčí krok nebo jednoduchý úkol splnil správně, aby získal jistotu,
  - možnost snadno se vracet k předchozím lekcím nebo vysvětlením.
- **Motivace:** Zvládnout požadavky předmětu programování a úspěšně dokončit školu navzdory obtížím s porozuměním této látky.

4.3.4 Učitel

- **Jméno:** Pavel Jančařík.
- **Věk:** 56 let.
- **Role:** učitel programování na střední průmyslové škole.
- **Cíle:**
  - Pavel chce co nejlépe naučit studenty programovat, aby je připravil na vysokou školu nebo práci v IT,

  - Pavel chce motivovat studenty k samostatnému myšlení a tvořivému přístupu při řešení úloh, jelikož je to při programování důležité,
  - Pavel si chce udržet přehled o nejnovějších trendech v oblasti programování a vzdělávání, aby podával studentům aktuální informace.

- **Chování a přístup k výuce:**
  - preferuje strukturovanou výuku, která kombinuje teoretické základy s praktickými cvičeními,
  - ocení interaktivní nástroje, které umožňují studentům lépe pochopit problematiku a získat praktické zkušenosti,
  - otevřený k využívání moderních technologií a nástrojů, pokud zlepšují efektivitu výuky.

- **Potřeby:**
  - jednoduchý a efektivní způsob, jak studentům předat látku a sledovat jejich pokrok,
  - nástroj pro procvičení programování a zadání úkolů s možností automatického vyhodnocování výsledků,
  - přístup ke statistikám a výsledkům studentů pro lepší zpětnou vazbu a přizpůsobení výuky.

- **Motivace:**
  - dosáhnout co nejlepších výsledků u svých studentů a připravit je na jejich budoucí studium nebo praxi,
  - osobní uspokojení z kvalitně odvedené práce a pozitivní zpětné vazby od studentů,
  - zjednodušení přípravy na hodiny pomocí efektivních nástrojů a metod.

4.4 User Stories

Na základě definovaných person byla vytvořena sada user stories, která odráží různorodé potřeby a motivace typických uživatelů aplikace pro výuku programování. Tyto user stories pomáhají lépe pochopit, jak různí uživatelé s aplikací interagují, a slouží jako základ pro návrh uživatelských funkcí a prioritizaci vývoje.

4.4.1 User Stories pro premianty

- **Jako premiant chci mít přístup k pokročilým materiálům**, které přesahují rámec běžných osnov, abych mohl/a rozšířit své znalosti.
- **Jako premiant chci mít možnost testovat své znalosti pomocí náročných úkolů**, abych si mohl/a ověřit svou úroveň a identifikovat oblasti pro další zlepšení.
- **Jako premiant chci mít k dispozici rychlou zpětnou vazbu na své řešení**, abych mohl/a rychle opravovat případné chyby a zlepšovat své dovednosti.
- **Jako premiant chci možnost přistupovat k diskuzím a konzultacím v rámci aplikace**, kde mohu najít odpovědi na složitější otázky a sdílet zkušenosti s ostatními.

4.4.2 User Stories pro průměrného studenta

- **Jako průměrný student chci mít přístup k jasným a strukturovaným materiálům**, které mi pomohou porozumět učivu a připravit se na testy.
- **Jako průměrný student chci mít interaktivní cvičení s okamžitou zpětnou vazbou**, abych mohl/a ihned zjistit, zda je moje řešení správné, a případně jej opravit.
- **Jako průměrný student chci mít možnost vracet se k náročnějším tématům a procházet je znovu**, abych si mohl/a upevnit své znalosti podle potřeby.
- **Jako průměrný student chci mít jednoduchý přístup k nápovědě a příkladům**, které mi pomohou pochopit složitější koncepty.

4.4.3 User Stories pro pomalejšího studenta

- **Jako pomalejší student chci mít k dispozici jednoduché a srozumitelné příklady**, které mi umožní rychle splnit minimální požadavky bez zbytečné námahy.
- **Jako pomalejší student chci mít možnost snadno procházet kurzy**, abych mohl/a dokončit cvičení co nejrychleji a vyhovět požadavkům školy.
- **Jako pomalejší student chci mít automatizovanou zpětnou vazbu**, která mě jednoduše informuje o tom, zda jsem splnil/a zadání, bez nutnosti podrobnější analýzy.

Tyto user stories pomáhají při specifikaci funkcionalit a nastavení priorit tak, aby aplikace poskytovala vhodné prostředí pro různé typy uživatelů a jejich individuální potřeby a motivace.

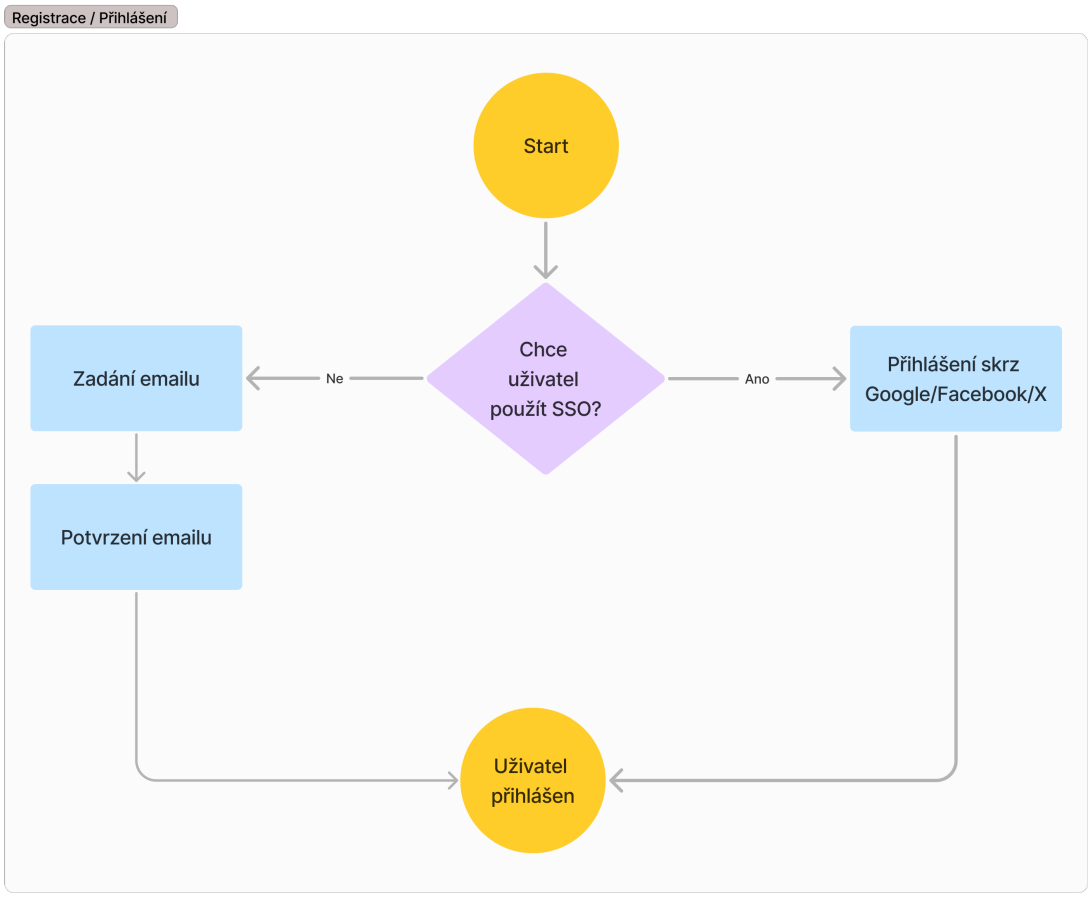
4.5 User Flows

V této kapitole jsou představeny základní scénáře uživatelských interakcí. Zaměřené na klíčové uživatelské cesty, které zahrnují přístup do aplikace prostřednictvím registrace a přihlášení, a následný průběh cvičení v rámci lekce.

4.5.1 Registrace a přihlášení

Registrace a přihlášení do aplikace představují vstupní bod pro nové a stávající uživatele. Tento user flow zajišťuje vytvoření nebo přihlášení k účtu a bezpečné ověření uživatele, aby měl přístup k personalizovanému obsahu a historii svého pokroku. Postup je popsán v seznamu níže a na obrázku 4.1.

- Uživatel navštíví web a hned uvidí možnost registrace a přihlášení.
- Pokud chce uživatel použít SSO, může tak učinit za pomocí účtu Google, Facebooku nebo X.
- Pokud nechce uživatel použít SSO, tak vyplní email, který následně potvrdí a bude odkázán zpět na web, kde už bude přihlášen.
- Po přihlášení je uživatel přesměrován na stránku obsahující seznam dostupných kurzů.

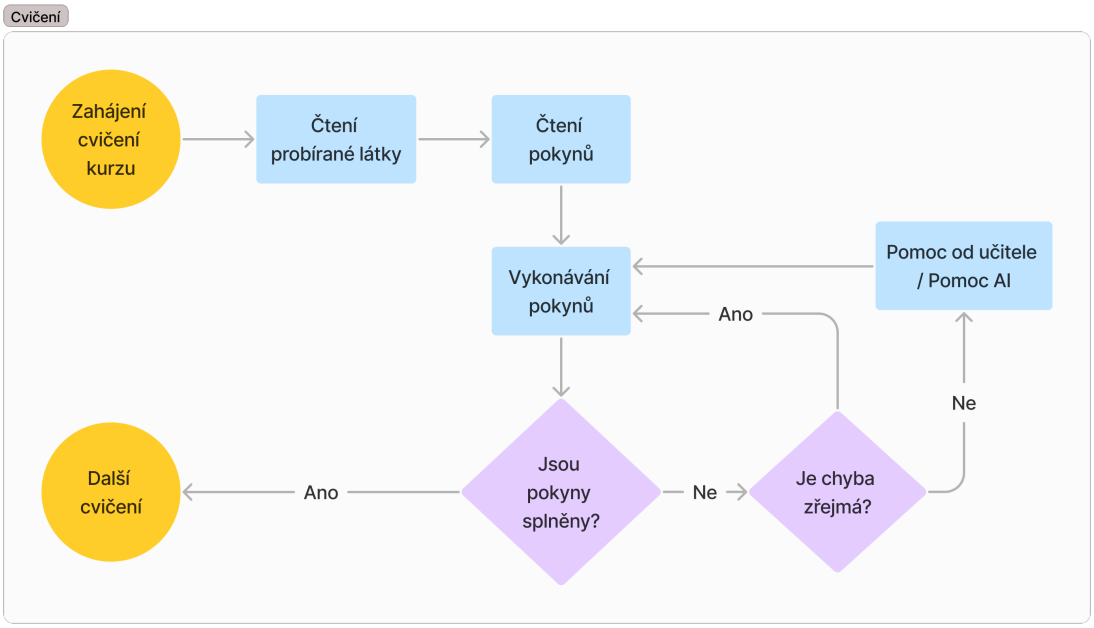


Obrázek 4.1: User flow registrace a přihlášení. Rozlišuje dvě strategie: email a SSO.

4.5.2 Průběh cvičení

Průběh cvičení zahrnuje jednotlivé kroky, které uživatel provádí během lekce či samostatného cvičení. Cílem je interaktivní učení a získávání zpětné vazby k pokroku. Průběh zahrnuje následující kroky, jak jde vidět na obrázku 4.2.

- Uživatel vybere cvičení, které chce absolvovat, a otevře ho.
- Na levé straně obrazovky si přečte vysvětlení právě probírané látky.
- Poté si přečte pokyny, které ověří, zda látce skutečně rozumí.
- Uživatel v interaktivním editoru řeší pokyny podle zadání a může jej průběžně spouštět, aby ověřil správnost.
- Při každém spuštění se řešení automaticky zkontroluje a uživatel tak získá okamžitou zpětnou vazbu.
- Pokud je řešení správné, uživatel má možnost přejít k dalšímu cvičení.
- V případě chyby má uživatel možnost požádat o pomoc AI nebo učitele.



Obrázek 4.2: User flow cvičení. Uživatel se snaží správně vykonat pokyny a pokud si neví rady, zeptá se chat bota nebo učitele.

Uvedené user flows pokrývají základní funkcionalitu potřebnou pro efektivní přístup k obsahu a interaktivnímu učení, které aplikace nabízí.

Kapitola 5

Návrh uživatelského rozhraní

Úspěšný návrh uživatelského rozhraní (UI) je klíčovým prvkem každé aplikace, neboť přímo ovlivňuje uživatelský zážitek (UX) a efektivitu práce uživatelů s aplikací. Cílem této sekce je popsat přístup a postupy, které byly použity při návrhu uživatelského rozhraní této aplikace. Zároveň budou zdůrazněny principy, jež byly při návrhu dodržovány, jako například jednoduchost, konzistence, přístupnost a vizuální hierarchie.

V kontextu vývoje interaktivní aplikace je nutné, aby rozhraní bylo intuitivní a uživatelé ho mohli používat bez složitého zaškolování. Návrh vychází z moderních zásad designu, které zahrnují minimalismus, důraz na vizuální přehlednost a přizpůsobivost pro různé typy zařízení.

Tato část práce detailně popíše proces návrhu. Zvláštní důraz je kladen na řešení konkrétních výzev, které byly při návrhu identifikovány, a na způsoby, jakými byl proces designu optimalizován k dosažení co nejlepšího uživatelského zážitku. [12]

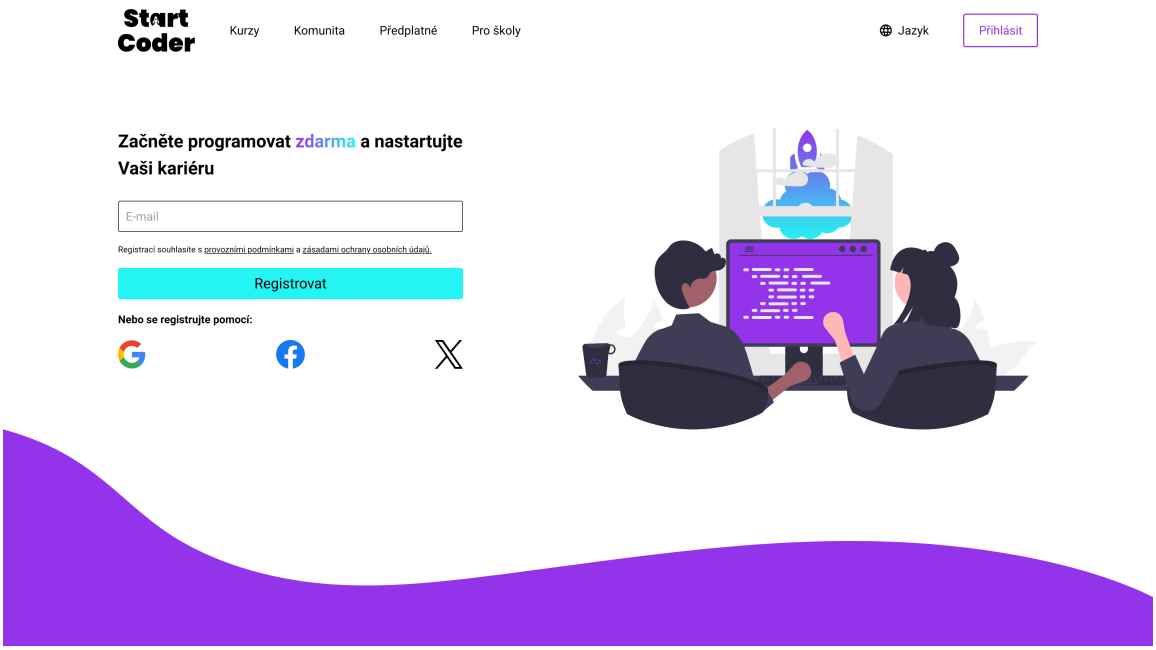
5.1 Úvodní obrazovka

Úvodní obrazovka je klíčovým prvkem aplikace, jelikož často představuje první kontakt uživatele s platformou a má zásadní vliv na jeho rozhodnutí pokračovat dále. Grafická podoba je na obrázku 5.1.

Celkově byla úvodní obrazovka navržena s důrazem na konverze a budování důvěry uživatelů, což je nezbytné pro získání nových uživatelů a jejich první krok v cestě za vzděláváním v programování.

5.2 Sekce kurzy

Tato klíčová sekce aplikace slouží jako rozcestník, kde si uživatelé mohou prohlížet a vybírat dostupné vzdělávací kurzy. Návrh této stránky, zobrazený na obrázku 5.2, ilustruje strukturu pro prezentaci různých kurzů. **Tato bakalářská práce se zabývá návrhem a strukturou pro více kurzů, včetně HTML, CSS, JavaScriptu, Pythonu a algoritmizace.** Kurz algoritmizace využívá specifické interaktivní prvky, jako je vizuální programování pomocí Blockly a 3D vizualizace, které jsou detailně popsány v dalších částech práce. Zatímco struktura a způsob prezentace jsou navrženy obecně pro všechny typy kurzů, hloubka implementace a specifické interaktivní mechanismy se v rámci této práce soustředí především na zmíněný kurz algoritmizace.



Obrázek 5.1: Návrh úvodní stránky.

Hlavním cílem této sekce je pomoci uživatelům snadno najít vhodný kurz na základě jejich znalostí a potřeb, ať už se jedná o úvod do algoritmizace nebo základy webových technologií či Pythonu. Stránka obsahuje i odkaz na kvíz, který má za úkol uživatelům pomoci s výběrem vhodného kurzu nebo programovacího jazyka pro další studium.

5.2.1 Struktura kurzů

Stránka je rozdělená na tři hlavní části:

- **moje kurzy:** uživatelé zde uvidí kurzy, které mají rozpracované nebo hotové,
- **naučit se něco nového:** v těchto kurzech zatím uživatel neudělal ani jedno cvičení,
- **již brzy:** tyto kurzy jsou teprve plánované.

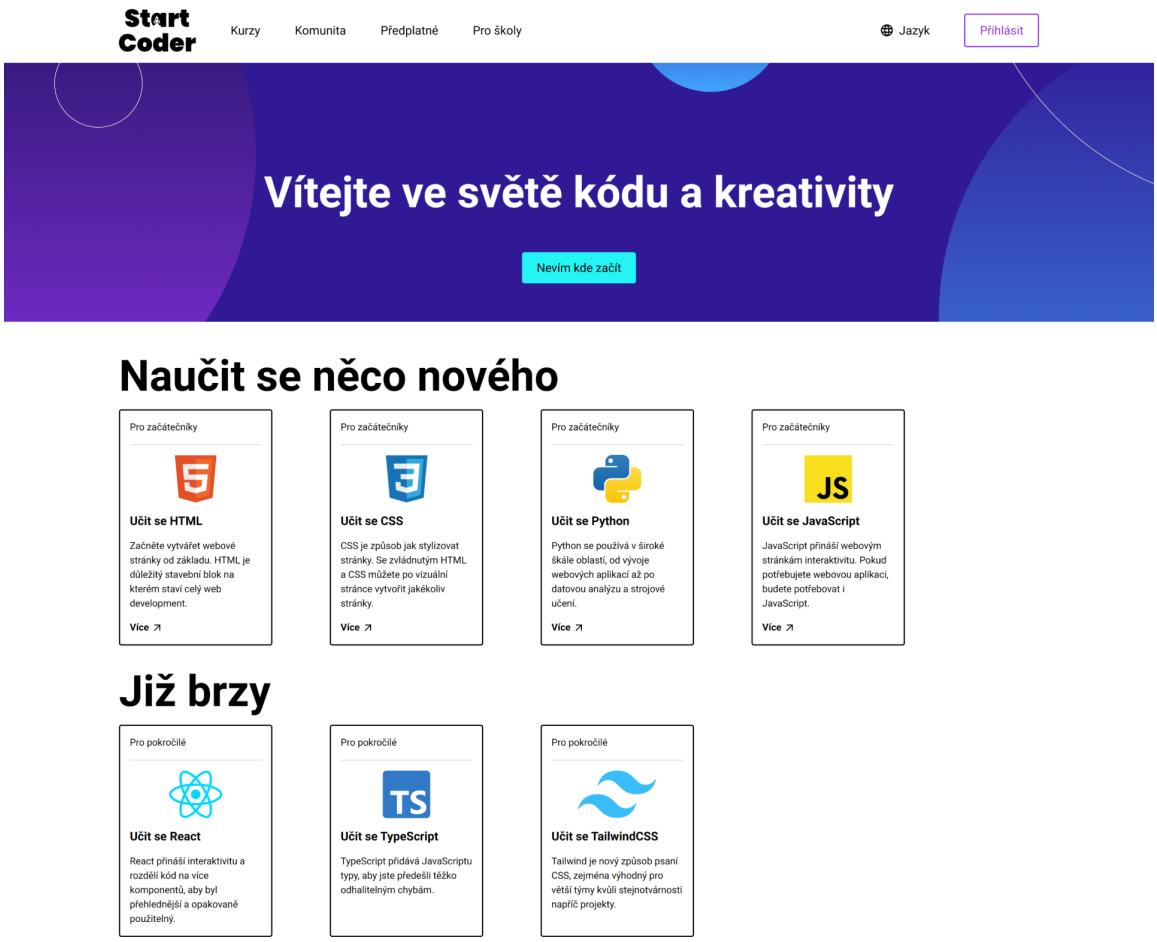
5.2.2 Interaktivní prvky

Každý kurz je prezentován jako karta obsahující:

- **název kurzu** spolu s úrovní obtížnosti,
- **krátký popis** zaměřený na hlavní přínos a využití technologie,
- **odkaz na více informací**, který uživatele přesměruje na detailní stránku kurzu.

5.2.3 Designové prvky

Pro zachování přehlednosti a moderního vzhledu byly použity konzistentní barvy a typografie, které odpovídají celkovému vizuálnímu stylu platformy. Karty kurzů mají čistý a přehledný design s jasně definovanými sekcemi, což usnadňuje rychlé prohlížení nabídky.



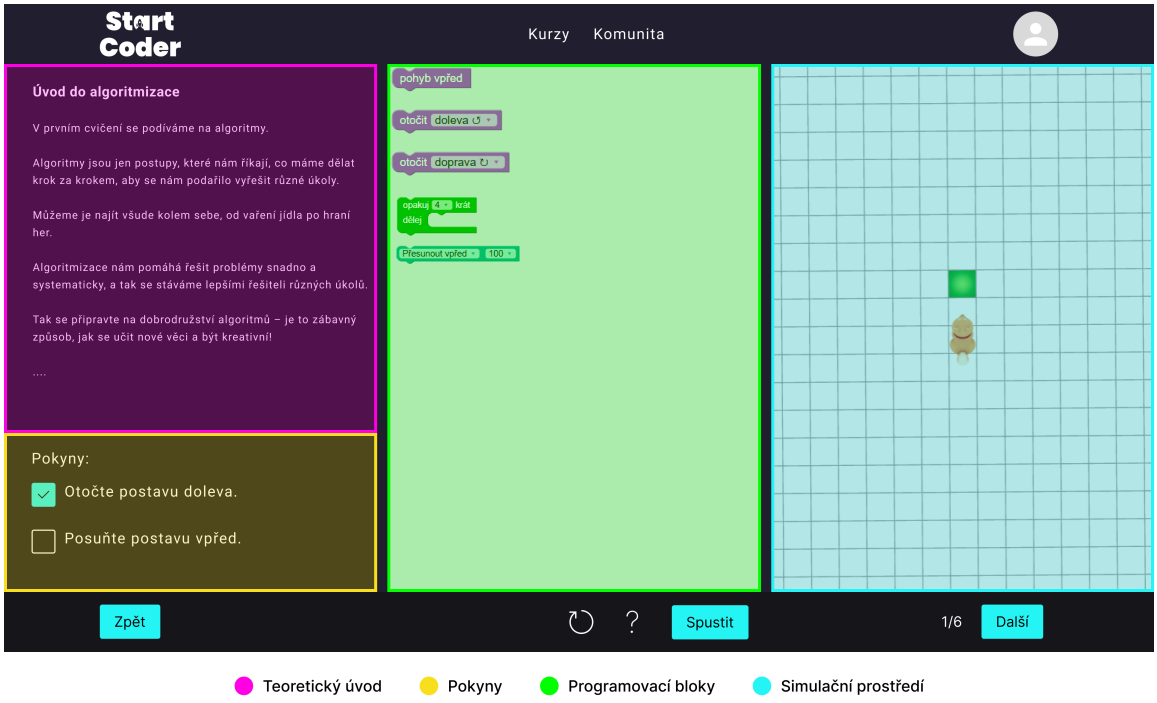
Obrázek 5.2: Návrh stránky s nabídkou kurzů. Aktuálně se zde nachází kurzy HTML, CSS, JavaScript, Python a algoritmizace.

Tato stránka hraje klíčovou roli v uživatelském zážitku, protože slouží jako rozcestník pro všechny vzdělávací aktivity na platformě.

5.3 Cvičení kurzu

Jedním z klíčových prvků aplikace je interaktivní prostředí pro výuku a procvičování programovacích dovedností. Na obrázku 5.3 je ukázáno typické cvičení, které uživatelé absolvují v rámci kurzu algoritmizace. Cvičení obsahuje několik částí.

- **Teoretický úvod:** Levá horní část obrazovky obsahuje úvodní text, který vysvětluje hlavní koncepty daného cvičení. Například v této ukázce se uživatel seznamuje se základy algoritmizace.
- **Pokyny:** V levé spodní části uživatel vidí konkrétní úkoly, které má splnit. Po splnění úkolu může pokračovat na další krok a po splnění všech úkolů může pokračovat na další cvičení.



Obrázek 5.3: Návrh cvičení kurzu.

- **Programovací bloky/Editor kódu:** V prostřední části je v případě kurzu algoritmizace vizuální editor, který umožňuje uživatelům sestavovat algoritmy pomocí bloků. Tyto bloky představují základní programovací struktury, jako je pohyb, smyčky nebo podmínky. V případě jiných kurzů (HTML, CSS, JS, Python) je zde editor kódu.
- **Simulační prostředí:** Pravá část zobrazuje simulaci, kde uživatel může sledovat chování svého algoritmu. Například v této ukázce algoritmus ovládá postavičku, která se pohybuje v mřížce s cílem dosáhnout určitého bodu.

Uživatelské rozhraní je navrženo tak, aby bylo intuitivní a podporovalo samostatné učení. Každý krok je doplněn vizuální zpětnou vazbou, která pomáhá uživatelům pochopit, jak jejich algoritmy fungují.

5.3.1 Návrh kurzu Algoritmizace

Kurz Algoritmizace je navržen jako úvodní kurz pro studenty bez předchozích zkušeností s programováním. Jeho cílem je seznámit studenty se základními principy algoritmického myšlení, logickými strukturami a řešením problémů pomocí sekvence kroků, a to bez nutnosti zabývat se syntaxí konkrétního programovacího jazyka.

Cílová skupina a cíle kurzu

Kurz je primárně určen pro studenty čtvrtých a pátých tříd základní školy a studenty středních škol jako úvod k programování. Po absolvování kurzu by student měl být schopen:

- Vysvětlit pojem algoritmus a jeho vlastnosti. Toto je vysvětleno hned v prvním cvičení.

- Vytvářet jednoduché sekvenční algoritmy pro řešení problémů. Toto je probíráno ve třech následujících cvičeních.
- Používat základní řídicí struktury: podmínky (větvení) a cykly (opakování). Toto je probíráno ve dvou posledních cvičeních.
- Rozkládat jednoduché problémy na menší kroky.
- Testovat a ladit své algoritmy ve vizuálním prostředí.

Interaktivní prvky a vizualizace

Klíčovým prvkem kurzu Algoritmizace je interaktivita:

- **Blockly Editor:** Umožňuje intuitivní sestavování algoritmů přetahováním a spojováním bloků, čímž eliminuje syntaktické chyby a umožňuje soustředit se na logiku.
- **3D Simulační prostředí:** Pravá část obrazovky zobrazuje 3D scénu s postavičkou, která vykonává kroky algoritmu sestaveného uživatelem. Toto vizuální ztvárnění pomáhá studentům okamžitě vidět důsledky jejich kódu.
- **Okamžitá zpětná vazba:** Po spuštění algoritmu uživatel vidí, zda postavička dosáhla cíle, splnila úkol, nebo zda algoritmus vedl k chybě. Systém validuje splnění podmínek definovaných v zadání cvičení.

Tento přístup kombinuje teoretické vysvětlení s praktickým experimentováním ve vizuálně atraktivním a interaktivním prostředí, což by mělo usnadnit pochopení abstraktních algoritmických konceptů.

5.3.2 Návrh kurzů pro řádkové programování (HTML, CSS, JS, Python)

Kromě úvodního kurzu Algoritmizace, který využívá vizuální programování, aplikace zahrnuje také kurzy zaměřené na konkrétní technologie a programovací jazyky založené na psaní řádkového kódu. Konkrétně se jedná o kurzy HTML, CSS, JavaScriptu a Pythonu. Přestože se každý z těchto jazyků a technologií liší svou syntaxí a účelem, návrh jejich kurzů v rámci aplikace sdílí několik společných principů a strukturálních prvků.

Cílová skupina a cíle kurzů

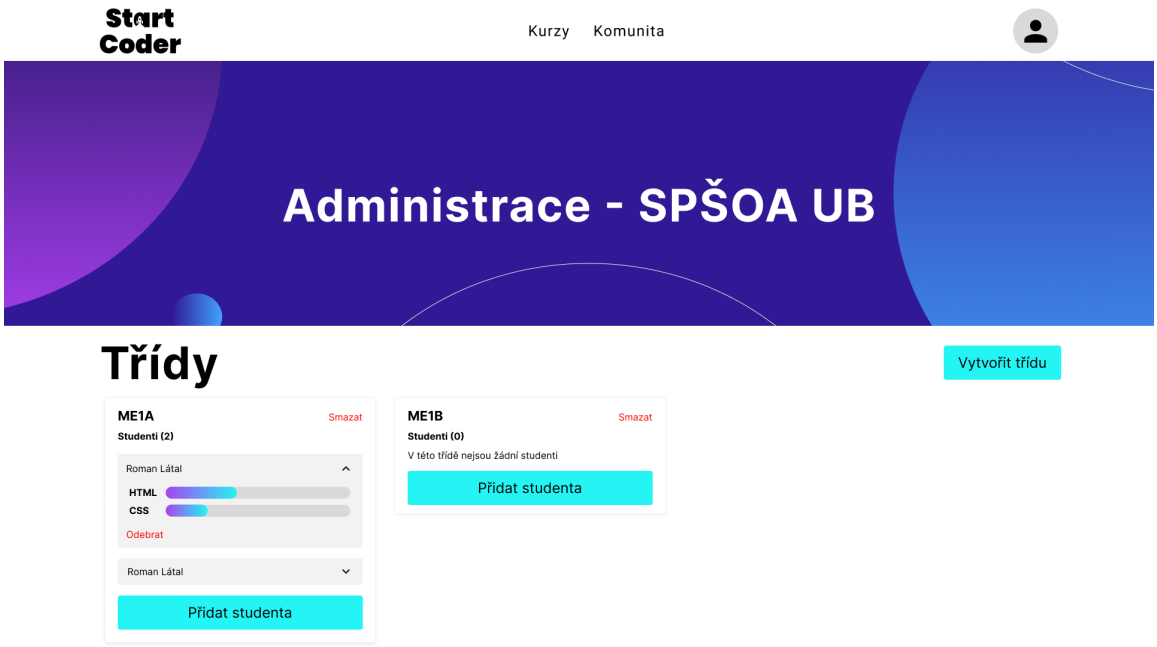
Tyto kurzy jsou primárně určeny pro studenty středních škol, případně pokročilejší žáky základních škol, kteří již mají za sebou úvod do algoritmizace nebo mají základní představu o programování. Cíle jednotlivých kurzů se liší podle specifik dané technologie, ale obecně by student po absolvování měl být schopen následujícího.

- **HTML:** rozumět struktuře webové stránky, používat základní HTML tagy pro tvorbu obsahu (nadpisy, odstavce, seznamy, obrázky, odkazy, tabulky, formuláře).
- **CSS:** stylovat HTML elementy, rozumět selektorům, vlastnostem, hodnotám a konceptům jako box model.
- **JavaScript:** pochopit základy programování v JS (proměnné, datové typy, operátory, řídicí struktury – podmínky, cykly) a pracovat s funkcemi.

- **Python:** osvojit si základy syntaxe Pythonu (proměnné, datové typy, operátory, řídicí struktury), pracovat s funkcemi a řešit jednoduché algoritmické úlohy.
- **Obecně:** testovat, ladit a spouštět svůj kód v interaktivním prostředí aplikace a aplikovat získané znalosti na řešení praktických úloh.

Struktura bude v zásadě stejná jako při kurzu algoritmizace s následujícími rozdíly. Místo blokového editoru bude přítomný řádkový editor kódu. Místo 3D výstupu zde bude v případě kurzu HTML a CSS grafický výstup a v případě JS a Pythonu konzole. Tento přístup má za cíl poskytnout studentům prostředí, kde mohou bezpečně experimentovat, okamžitě vidět výsledky své práce a dostávat konstruktivní zpětnou vazbu, což je klíčové pro efektivní učení se programování a webových technologií.

5.4 Přehled pro učitele



Obrázek 5.4: V přehledu mají učitelé přehled o všech svých třídách společně s informacemi o pokroku studentů v jednotlivých kurzech

Tato stránka slouží jako nástroj pro učitele, který jim umožňuje sledovat pokrok studentů v rámci jednotlivých tříd, skupin a kurzů (obrázek 5.4). Rozhraní poskytuje přehledné grafy znázorňující postup v jednotlivých kurzech, což usnadňuje individuální přístup a další plánování výuky. Každá třída je rozdělena do skupin, přičemž každá skupina obsahuje seznam studentů s jejich průběžnými výsledky v jednotlivých kurzech.

5.5 Databázový návrh

Tato sekce popisuje návrh databáze pro aplikaci *StartCoder*. Cílem databázového návrhu je efektivní uložení a správa dat souvisejících s uživateli, kurzy, cvičeními a dalšími klíčovými entitami systému.

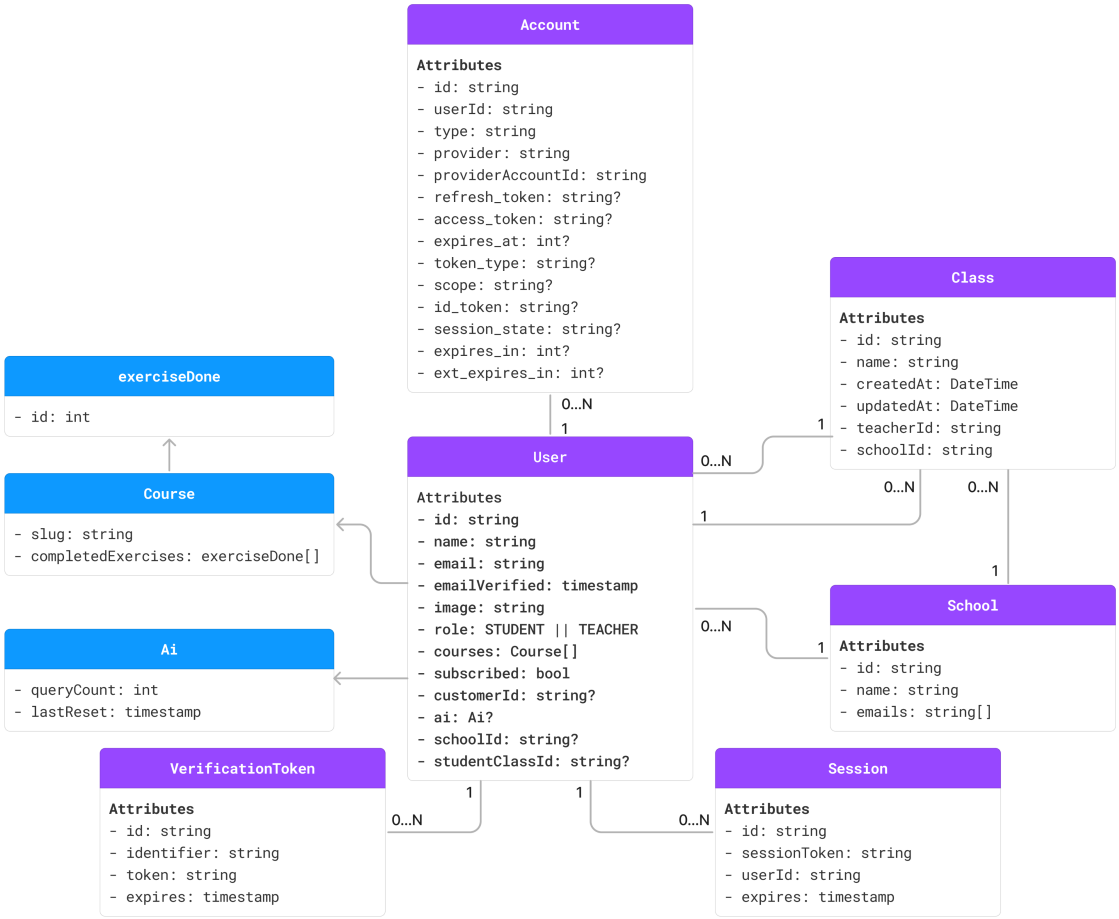
5.5.1 Analýza požadavků

Pro návrh databáze byly identifikovány následující hlavní požadavky:

- správa uživatelů (studentů a učitelů) včetně jejich údajů, jako jsou jméno, e-mail, role, a pokrok v kurzech,
- podpora rozdělení uživatelů do škol, tříd a skupin.

5.5.2 ER diagram

Pro vizualizaci návrhu databáze byl vytvořen ER diagram, který znázorňuje hlavní entity a jejich vztahy (obrázek 5.5).



Obrázek 5.5: ER diagram databázového návrhu pro aplikaci StartCoder

5.5.3 Popis hlavních entit

- **users (Uživatel):** Obsahuje informace o uživatelích, jako je jejich jméno, role (student/učitel), e-mail, a postup v kurzech.
- **accounts (Účet):** informace o způsobech přihlášení uživatelem (google účet, facebook účet atd.).
- **sessions (Sezení):** Ukládá sezení uživatelů pro funkční autorizaci.
- **verification_tokens (Verifikační token):** Slouží k přihlašování za pomocí emailu, kdy uživateli dojde magic-link, který ho přihlásí.
- **schools (Škola):** Povolí registraci a přihlášení skrze Microsoft účet jen emailovým adresám v nějaké škole v této tabulce.
- **classes (Třída):** Třída obsahuje jednoho učitele a studenty, pro organizované sledování pokroku studentů.

5.6 Rozhodování mezi 2D a 3D výstupem

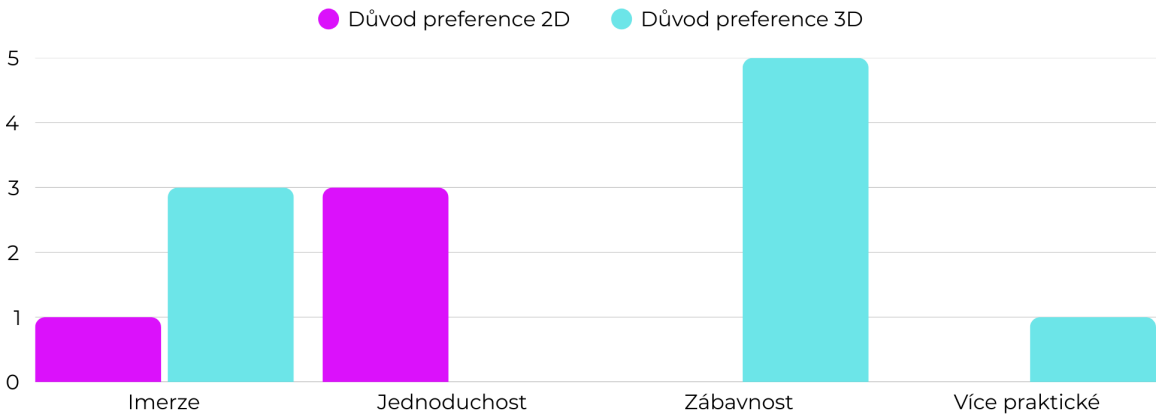
V rámci návrhu byl testován výstup jak dvourozměrný (2D), tak trojrozměrný (3D). Testování se zúčastnilo 13 studentů. Testovalo se na studentech středních škol a jednom studentovi základní školy. U trojrozměrného výstupu má uživatel možnost otáčet s kamerou, naopak u dvourozměrného výstupu je kamera fixovaná nad postavou. Volba mezi těmito přístupy závisela na několika faktorech, které ovlivňují pochopitelnost, technickou náročnost a efektivitu výuky, a také na datech z dotazníku.

5.6.1 Dotazník

Pro hlubší prozkoumání uživatelské preference byl vytvořen dotazník, který obsahoval odkaz na web, kde si uživatelé mohli vyzkoušet jak 2D, tak 3D verzi výstupu. Dále obsahoval následující 2 dotazy.

- Zda uživatel preferuje 2D nebo 3D zobrazení,
- Z jakého důvodu uživatel tuto možnost preferuje.

Obrázek 5.6: Výsledek dotazníku o preferenci 2D nebo 3D



Obrázek 5.7: Důvody preference 2D nebo 3D

Z výsledků první části dotazníku na obrázku 5.6 můžeme zpozorovat, že 3D verze je mezi studenty a učiteli výrazně preferovaná. Z druhé části dotazníku na obrázku 5.7 vychází, že nejčastěji je 3D verze preferovaná z důvodu zábavnosti a imerze.

5.6.2 Výhody a nevýhody 2D výstupu

2D prostředí bylo zejména preferováno z důvodu jednoduchosti.

- je jednodušší na pochopení, protože se neotáčí kamerou v Z souřadnici,
- umožňuje rychlejší a jednodušší tvorbu výukových materiálů.

Nicméně mezi nevýhody patří následující:

- omezené možnosti vizualizace složitějších scén a objektů,
- omezení kreativity při návrhu cvičení (např. postava nemůže skákat a pohybovat se v Z ose),
- nižší atraktivita pro žáky zvyklé na moderní 3D grafiku ve hrách.

5.6.3 Výhody a nevýhody 3D výstupu

Trojrozměrná vizualizace nabízí větší interaktivitu a realističtější prostředí, což přináší výhody jako:

- možnost hlubšího ponoření do programovacího prostředí,
- realističtější simulace fyzikálních jevů a pohybů objektů,
- atraktivita pro studenty zvyklé na 3D hry a aplikace.

Na druhou stranu je třeba zvážit i nevýhody:

- složitější implementace,
- možné ztížení pochopení algoritmů kvůli složitějším vizualizacím.

5.7 Návrh implementace umělé inteligence

Na základě diskuze s vyučujícími informatiky a programování ze SPŠOA bylo diskutováno o dvou možnostech implementace umělé inteligence do výuky. První způsob je implementace ve formě chat bota, kdy umělá inteligence pouze odpovídá na otázky studenta. Druhá varianta byla automatická zpětná vazba po kliknutí na tlačítko spustit. Toto ale vytvořilo pár problémů. Někteří studenti chtějí problém vyřešit bez pomoci a po delší době této pomoci začali někteří studenti zneužívat a stiskli tlačítko spustit ihned, aby dostali pomoc. Kvůli těmto problémům bylo nakonec rozhodnuto, že umělá inteligence bude v této výukové webové aplikaci implementována ve formě chat bota.

Cílem této implementace je poskytnout studentům interaktivní podporu během výuky, zejména při řešení cvičení nebo při obecných dotazech souvisejících s programováním a algoritmizací. Chat bot slouží jako virtuální asistent, kterého se může uživatel kdykoliv zeptat na otázky související s výukou.

Chat bot je navržen tak, aby byl schopen reagovat přirozeným jazykem, a poskytoval srozumitelné, přímé a věcné odpovědi. Jeho hlavními schopnosti jsou následující.

- Vysvětlení programovacích konstrukcí (např. podmínky, cykly).
- Pomoc při řešení algoritmických úloh.
- Odpovědi na technické dotazy ohledně syntaxe a chyb v kódu.
- Povzbuzení a motivace studentů při učení.

Použití AI formou chat bota bylo zvoleno především kvůli nízké bariéře vstupu pro studenty a přirozenému způsobu komunikace. Tento přístup umožňuje individualizované poradenství bez nutnosti stálé přítomnosti učitele, čímž zvyšuje samostatnost a sebevědomí studentů při učení programování.

Kapitola 6

Implementace

Tato kapitola se věnuje praktické realizaci výukové webové aplikace zaměřené na programování a algoritmizaci. Na základě analýzy požadavků, konzultací s vyučujícími a připomínek od cílové skupiny byla navržena a následně implementována webová aplikace, která kombinuje vizuální prvky, interaktivitu a moderní přístup k výuce.

Cílem této kapitoly je přiblížit způsob, jakým byly jednotlivé části systému navrženy a implementovány, včetně výběru vhodných technologií, struktury systému, práce s daty a integrace podpůrných nástrojů, jako je například umělá inteligence.

Implementace se zaměřuje jak na front-end, tak i na back-end a databázovou vrstvu. Důraz je kladen na srozumitelnost, přehlednost kódu a snadnou rozšiřitelnost do budoucna. Součástí implementace je také systém cvičení, mechanismus vyhodnocování úloh a uživatelské rozhraní navržené s ohledem na použitelnost a věkovou skupinu cílových uživatelů.

V následujících sekcích je popsán výběr použitých technologií a nástrojů, struktura aplikace a klíčové části implementace, které tvoří jádro výukového systému.

6.1 Výběr technologií

Při vývoji výukové webové aplikace byly vybrány moderní technologie, které umožňují rychlý a efektivní vývoj, snadnou údržbu a dobrý uživatelský zážitek. Důraz byl kladen na použití otevřených a komunitou podporovaných nástrojů, které jsou vhodné pro vývoj interaktivních a škálovatelných webových aplikací.

Next.js a React

Základem celé aplikace je framework **Next.js**, který je postaven na knihovně **React** a poskytuje server-side rendering, což zlepšuje klíčové metriky jako First Contentful Paint (FCP) a Search Engine Optimization (SEO). **React** umožňuje strukturovat uživatelské rozhraní pomocí komponent, které jsou snadno rozšiřitelné a dobře udržovatelné. Dále přináší výkonostní výhody, jelikož používá VDOM (Virtual Document Object Model), což je virtuální reprezentace UI, která je synchronizována s reálným DOM pomocí knihovny ReactDOM.

TypeScript

Celá aplikace je vyvíjena v jazyce **TypeScript**, což je nadstavba jazyka JavaScript, rozšiřující ho o statické typování. Použití TypeScriptu přináší řadu výhod, zejména v oblasti

vývoje větších aplikací — umožňuje detekci chyb už při překladu, poskytuje lepší podporu v editorech kódu, zlepšuje čitelnost a dokumentaci kódu a usnadňuje refaktoring.

Typové anotace jsou využívány v celém projektu, včetně komponent, API endpointů, práce s databází přes Prisma i s externími knihovnami. Díky TypeScriptu je také zajištěna vyšší typová bezpečnost při práci s vícejazyčnými texty, strukturou dat a datovými modely, což pomáhá předejít častým runtime chybám.

Použití TypeScriptu významně přispívá ke stabilitě a udržovatelnosti kódu, a je tak důležitým prvkem celkové architektury aplikace.

Tailwind CSS

Pro styling aplikace byla použita utility-first CSS knihovna **Tailwind CSS**, která umožňuje rychlé vytváření responzivního a konzistentního designu přímo ve struktuře HTML/JSX. Tato technologie zjednodušuje správu vzhledu a zároveň eliminuje potřebu psaní vlastních tříd v externích souborech.

MongoDB a Prisma

Pro ukládání dat byla zvolena dokumentově orientovaná databáze **MongoDB**, která je vhodná pro flexibilní práci se strukturovanými i polo-strukturovanými daty. Pro práci s databází v prostředí TypeScript/JavaScript je využita ORM (Object–Relational Mapping) knihovna **Prisma**, která poskytuje typovou bezpečnost, přehledný datový model a nástroje pro správu migrací.

Stripe

Pro zajištění monetizace aplikace a platební funkcionality byl integrován platební systém **Stripe**[1], který umožňuje bezpečné zpracování platebních transakcí.

Blockly

Specificky pro část aplikace zaměřenou na výuku algoritmizace byl zvolen nástroj **Blockly**[2] – open-source knihovna od Googlu pro vizuální programování pomocí blokového editoru. Blockly umožňuje uživatelům sestavovat programy pomocí přetahování a skládání bloků, čímž zjednodušuje pochopení základních programovacích konstrukcí, jako jsou podmínky, cykly, proměnné nebo funkce.

React Three Fiber

Pro vizualizaci výstupu algoritmických úloh ve 3D prostředí byla použita knihovna **React Three Fiber**, která umožňuje vytváření interaktivní 3D grafiky přímo v prostředí React. Tato knihovna staví na populární technologii **Three.js** a umožňuje přirozené propojení 3D grafiky s komponentovým přístupem Reactu. Díky tomu mohou studenti vidět výsledky svých algoritmů v přehledné a atraktivní formě.

Použití výše zmíněných technologií vytvořilo stabilní a moderní základ, na kterém bylo možné postavit výukovou aplikaci kombinující efektivní výuku, atraktivní vizuální prezentaci a praktické využití aktuálních webových technologií.

[1] Stripe dokumentace: https://docs.stripe.com/

[2] Blockly dokumentace: https://developers.google.com/blockly/reference/js/blockly

NextAuth

Pro správu autentizace a autorizace uživatelů byla využita knihovna **NextAuth**, která je přímo určená pro aplikace postavené na frameworku Next.js. **NextAuth** poskytuje flexibilní řešení pro přihlašování pomocí e-mailu, OAuth poskytovatelů (např. Google, GitHub) nebo vlastního poskytovatele přihlašování.

V rámci této aplikace je využita autentizace pomocí e-mailu, Googlu, Facebooku a X (dříve Twitter). NextAuth zajišťuje bezpečné ukládání relací a správu uživatelského stavu. Díky napojení na databázi prostřednictvím **Prisma** lze jednoduše pracovat s uživatelskými účty a oprávněními.

Výhodou tohoto řešení je nejen snadná integrace s Next.js, ale také podpora server-side i client-side autentizace, ochrana API rout a možnost rozšíření o vlastní zpětné volání pro pokročilou logiku ověřování.

Sentry

Pro monitoring chyb a výkonu aplikace byla integrována knihovna **Sentry**, která slouží k detekci, sledování a diagnostice runtime chyb jak na straně klienta, tak serveru. **Sentry** umožňuje v reálném čase zaznamenávat výjimky, chyby a další důležité události, které mohou ovlivnit funkčnost aplikace nebo uživatelský zážitek.

Díky integraci s **Next.js** je možné snadno sledovat chyby ve všech částech aplikace – včetně API rout, klientských komponent a serverového renderování. Sentry také poskytuje detailní informace o kontextu chyby (např. zařízení, prohlížeč, uživatel, stav aplikace), což výrazně urychluje ladění a odstraňování problémů.

6.2 Architektura aplikace

Next.js využívá file-based routing, což znamená, že adresářová struktura a velká část názvů souborů je pevně stanovena. Na druhou stranu je lehce rozšiřitelná a modulární. Základní struktura vypadá následovně:

```
root
├── app
├── components
├── public
├── prisma
│   └── schema.prisma
├── dictionaries
└── middleware.ts
```

Složka `/app`

Hlavní logika routování je umístěna ve složce `/app`, která vychází z tzv. App Routeru v Next.js. Každá podsložka v rámci této složky reprezentuje konkrétní cestu v rámci aplikace, přičemž je možné definovat soubory jako `page.tsx` (obsahuje hlavní obsah stránky), `layout.tsx` (layout pro danou část aplikace) a další specializované soubory (například `loading.tsx` – fallback při načítání obsahu nebo `error.tsx` – obsah se zobrazí, když nastane chyba). Tento přístup zajišťuje čistou a hierarchicky uspořádanou strukturu aplikace.

Složka `/components`

Znovupoužitelné React komponenty jsou organizovány ve složce `/components`. Jedná se zejména o menší vizuální nebo funkční prvky, které jsou používány napříč různými stránkami, např. formulářové prvky, hlavička, patička, tlačítka nebo modální okna. Tento přístup podporuje opakovatelnost kódu a jeho snadnější údržbu.

Složka `/public`

Statické soubory jako obrázky, ikony, písma nebo další assety jsou umístěny ve složce `/public`. Tyto soubory jsou přímo přístupné přes kořenovou URL adresu webové aplikace a nejsou transformovány buildovacím procesem Next.js.

Soubor `/prisma/schema.prisma`

Datový model aplikace je definován v souboru `schema.prisma`, který se nachází ve složce `/prisma`. Tento soubor popisuje databázovou strukturu, jednotlivé entity a jejich relace, a slouží jako základ pro generování klienta knihovny Prisma, který umožňuje typově bezpečný přístup k databázi.

Složka `/dictionaries`

Pro zajištění vícejazyčnosti aplikace byla vytvořena složka `/dictionaries`, která obsahuje soubory `cs.json` a `en.json`. Tyto JSON soubory obsahují textový obsah stránky ve dvou podporovaných jazycích – češtině a angličtině. Texty jsou strukturovány podle jednotlivých sekcí aplikace a načítány dynamicky podle aktuálně zvoleného jazyka uživatele. Jejich velký rozsah neovlivňuje velikost finální stránky, jelikož využíváme SSR, což zajistí, že se klientovi pošle jen vytvořený HTML soubor se správnými texty.

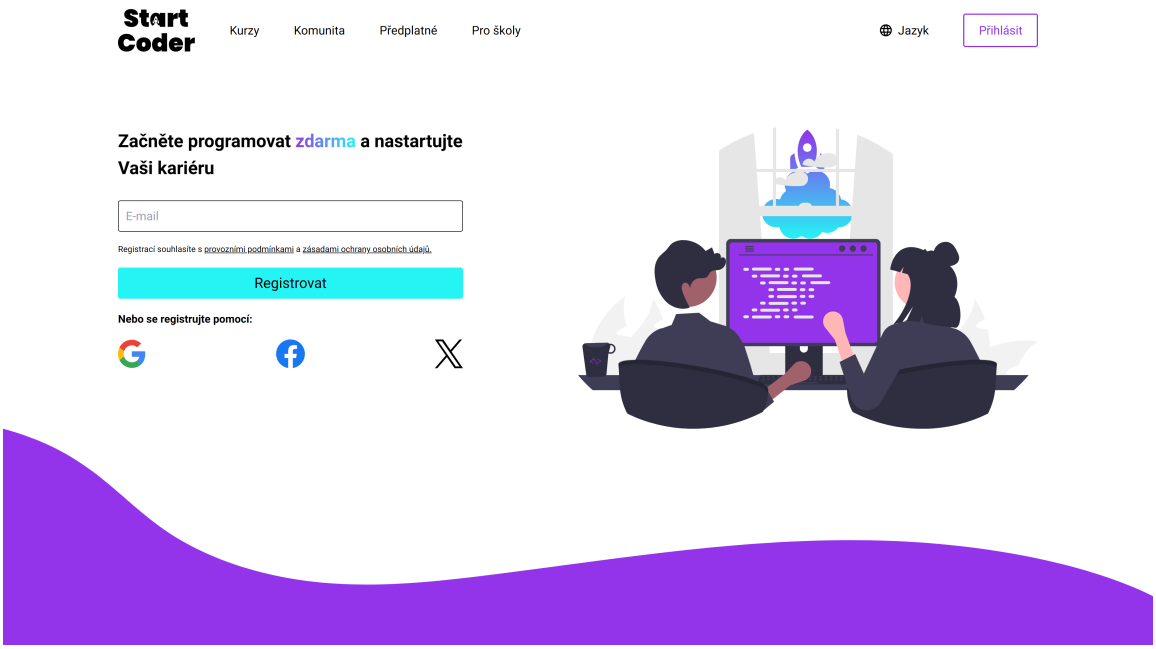
Soubor `middleware.ts`

Aplikace využívá soubor `middleware.ts` pro detekci preferovaného jazyka uživatele. Middleware zachytí každý příchozí HTTP požadavek a na základě hlavičky `Accept-Language` provede přesměrování na odpovídající jazykovou verzi stránky. Pokud má uživatel nastavenou češtinu jako preferovaný jazyk, bude přesměrován na českou verzi webu, v opačném případě na verzi anglickou. Tento mechanismus zajišťuje automatizovaný a uživatelsky přívětivý jazykový výběr bez nutnosti manuálního nastavení.

Celkově je architektura aplikace navržena s důrazem na čitelnost, modularitu a přizpůsobení různým jazykovým verzím. Kombinace statických a dynamických mechanismů v rámci Next.js umožňuje efektivní správu obsahu, komponent i uživatelského prostředí.

6.3 Registrace a přihlášení

Aplikace používá `NextAuth` pro správu ověřování identity uživatele, což poskytuje bezpečnou a flexibilní možnost pro správu uživatelských *session* (relací) a přístupu. `NextAuth` podporuje mnoho způsobů ověřování, včetně e-mailu a sociálních sítí, což se hodí pro širokou škálu uživatelů. Grafická podoba je na obrázku 6.1.



Obrázek 6.1: Implementovaná obrazovka registrace

Konfigurace

`NextAuth` je nakonfigurován tak, aby podporoval poskytovatele z následujícího seznamu.

- **Email**: Uživatelé mohou registrovat a přihlašovat se pomocí své e-mailové adresy. Po registraci je zaslán verifikační e-mail pro potvrzení identity.
- **Google**: Uživatelé se mohou přihlašovat pomocí svého Google účtu.
- **Facebook**: Podobně jako Google, uživatelé se mohou ověřovat pomocí svého Facebooku.
- **X (dříve Twitter)**: Uživatelé se rovněž mohou přihlašovat pomocí svého účtu na X.
- **Microsoft**: Možnost pouze pro školy. Pomocí účtů Microsoft se mohou přihlašovat pouze emailové adresy, které jsou v databázi přiřazeny ke škole.

Konfigurace je nastavena v souboru `auth.ts`, kde je `NextAuth` inicializován s těmito poskytovateli. Citlivé informace, jako *client ID* (identifikátor klienta) a tajné klíče pro každého poskytovatele, se ukládají v proměnných prostředí.

Proces registrace

1. **Registrace pomocí e-mailu**.
    - Uživatel zadá svou e-mailovou adresu a odešle formulář pro registraci.
    - Je zaslán verifikační e-mail obsahující magický odkaz.
    - Po kliknutí na magický odkaz se uživatel přihlásí a jeho účet je vytvořen.
2. **Registrace pomocí sociálních sítí**.
    - Uživatel klikne na tlačítko pro přihlášení daného poskytovatele sociální sítě.
    - Je přesměrován na stránku poskytovatele pro ověření.
    - Po úspěšném ověření je přesměrován zpět do aplikace a jeho účet je vytvořen nebo propojen, pokud již má účet.

Proces přihlášení

Proces přihlášení je z pohledu uživatele identický jako proces registrace.

Správa session

`NextAuth` spravuje *session* (uživatelské relace) pomocí cookies. Token relace je uložen v bezpečné cookie a data session se ukládají v databázi prostřednictvím `Prisma`. To zajišťuje, že uživatelské relace jsou spravovány bezpečně a efektivně.

Přizpůsobení

Pro splnění specifických potřeb aplikace se v `NextAuth` implementují přizpůsobené *callbacky* (zpětná volání). Tyto callbacky obsahují dodatečnou logiku, jako například:

- kontrola, zda je uživatelův e-mail ověřen před umožněním přístupu k určitým funkcionalitám,
- synchronizace uživatelských dat s databází pomocí `Prisma`.

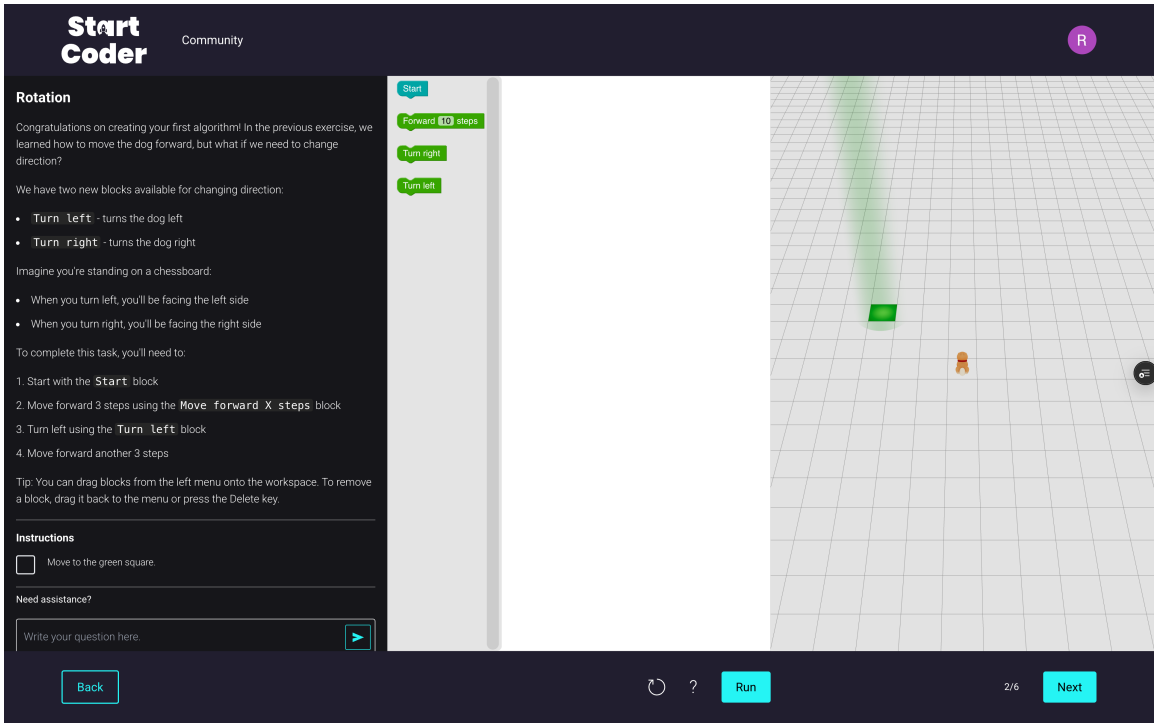
Toto přizpůsobení zajišťuje, že proces ověřování je přizpůsoben požadavkům vzdělávací platformy a poskytuje robustní a uživatelsky příjemný zážitek.

6.4 Cvičení

Tato podsekce popisuje implementaci uživatelského rozhraní a logiky pro interaktivní cvičení v rámci aplikace. Grafická podoba je na obrázku 6.2 a klíčové prvky jsou vypsány v seznamu níže.

- Zobrazení zadání a teoretického úvodu.
- Implementace editoru pro vizuální programování (Blockly) v kurzu algoritmizace.
- Integrace 3D vizualizace pomocí React Three Fiber pro zobrazení výsledků algoritmů.
- Mechanismus pro spouštění a vyhodnocování uživatelského řešení.

- Poskytování okamžité zpětné vazby uživateli.
- Komunikace s backendem pro ukládání pokroku a případnou kontrolu řešení.
- Integrace s AI (chat bot) pro nápovědu.



Obrázek 6.2: Implementovaná obrazovka cvičení.

6.4.1 Načítání a správa obsahu kurzu

Je důležité zdůraznit, že samotný **obsah a struktura kurzů** (tj. definice lekcí, texty zadání, obsah cvičení, definice testů, konfigurace Blockly toolboxu atd.) **nejsou uloženy v databázi**. Tyto definice jsou spravovány v externích **JSON souborech** (např. `cs.json`, `en.json`) umístěných v adresáři `/dictionaries` v rámci zdrojového kódu aplikace. Tato separace přináší několik výhod.

- **Oddělení obsahu od stavu uživatele:** Databáze uchovává pouze dynamický stav – *kdo* (`User`) co (`slug` kurzu) dokončil (`completedExercises`). Statická definice kurzů je mimo databázi.
- **Snadná správa a verzování obsahu:** Obsah kurzů lze upravovat přímo v JSON souborech a změny jsou sledovány systémem pro správu verzí (Git) společně se zbytkem kódu. Není potřeba provádět databázové migrace při změně obsahu kurzu.
- **Výkon načítání definic:** Definice kurzů mohou být načteny aplikací při startu nebo sestavení (build time) a drženy v paměti nebo efektivně servírovány, aniž by zatěžovaly databázi při každém požadavku na zobrazení kurzu.

Jak to funguje dohromady:

Když uživatel přistoupí ke konkrétnímu cvičení v kurzu stane se následující seznam akcí.

1. Aplikace načte definici daného kurzu a cvičení z příslušného JSON souboru (podle jazyka a `slug` kurzu).
2. Aplikace se dotáže do databáze na záznam aktuálního uživatele (`User`).
3. Z dat uživatele najde v poli `courses` záznam odpovídající danému `slug` kurzu.
4. Z tohoto záznamu zjistí, která cvičení (`completedExercises`) již uživatel dokončil.
5. Na základě kombinace statické definice cvičení z JSON a dynamických dat o postupu z databáze zobrazí uživateli správný stav cvičení (zadání, editor, případně označení dokončených úkolů).
6. Když uživatel úspěšně dokončí cvičení, tak aplikace v databázi aktualizuje pouze pole `completedExercises` v záznamu uživatele v databázi pro daný kurz.

Tento návrh efektivně kombinuje flexibilitu správy statického obsahu v souborech s robustním a výkonným ukládáním dynamického stavu uživatelů v dokumentové databázi.

6.4.2 Zobrazení zadání a teoretického úvodu.

Jak již bylo zmíněno, obsah jednotlivých cvičení, včetně teoretických úvodů a zadání, je uložen v jazykově specifických souborech formátu JSON (např. `cs.json` pro češtinu, `en.json` pro angličtinu). Ačkoliv tyto soubory mohou být rozsáhlé, aplikace využívá Server-Side Rendering (SSR). Díky tomu klientský prohlížeč nestahuje celý JSON soubor, ale obdrží již předrenderovanou HTML stránku obsahující pouze relevantní texty pro daný jazyk a cvičení.

Struktura dat pro jedno cvičení v JSON formátu je demonstrována na následujícím příkladu:

```
{
    "id": 0,
    "slug": "intro",
    "name": "Úvod do algoritmizace",
    "content": "<p>Vítejte v~první lekci algoritmizace!</p><p>Co je to
       algoritmus?</p><p>Algoritmus je jako recept. Je to přesný postup,
       jak má počítač vykonat úlohu.</p><p>Nyní vytvoříme náš první
       algoritmus!</p><p>Našim cílem je dostat psa na zelené políčko na
       zemi.</p><p>Zhruba uprostřed obrazovky vidíte kostky. Z~těchto
       kostek budeme skládat náš algoritmus. Na pravé straně uvidíme výstup
        našeho algoritmu.</p><p>Každý algoritmus musí začít kostkou <code>
       Start</code>. Přetáhněte ji do prostoru s~bílým pozadím. Nyní ke
       kostce start připojte ze spodu kostku <code>Dopředu o~10 kroků</code
       > a změňte počet kroků na <code>2</code>, jelikož zelené políčko je
       o~dvě políčka před psem.</p><p>Nyní stiskněte tlačítko spustit.</p
       >",
    "code": [
        {
```

```
            "file": "index.html",
            "content": ""
        }
    ],
    "toolbox": {
        "kind": "flyoutToolbox",
        "contents": [
            { "kind": "block", "type": "start" },
            { "kind": "block", "type": "move_forward" }
        ]
    },
    "backgroundCode": "!codeToBeReplaced!",
    "hint": "",
    "sandbox": "",
    "instructions": [
        {
            "text": "Posuňte se na zelené políčko.",
            "goalPosition": [0, 0.1, 2],
            "test": "TEST:if (position.x === goalPos[0] && position.y === 1
                && position.z === goalPos[2]) {return true;}else {return
                false;}"
        }
    ]
}
```

Popis klíčů JSON struktury.

`id` Numerický identifikátor cvičení.

`slug` Unikátní textový řetězec používaný jako součást URL adresy pro danou lekci (např. `/functions/intro`).

`name` Lidsky čitelný název cvičení zobrazovaný uživateli.

`content` Teoretický úvod a zadání cvičení ve formátu HTML. V klientské aplikaci (React) je tento obsah typicky vykreslován pomocí atributu `dangerouslySetInnerHTML`, aby byly HTML tagy správně interpretovány prohlížečem.

`code` Pole objektů definujících soubory a jejich počáteční obsah, se kterými uživatel pracuje (např. editor kódu nebo plocha pro Blockly bloky). Každý objekt obsahuje klíče `file` (název souboru) a `content` (počáteční obsah).

`toolbox` Konfigurace dostupných nástrojů nebo bloků pro dané cvičení. Toto je specifické pro kurz algoritmizace. Obsahuje `kind` (typ toolboxu) a `contents` (pole definic dostupných bloků).

`backgroundCode` Řetězec obsahující kód, který běží na pozadí (není viditelný uživateli). Kód vytvořený uživatelem je typicky vložen na místo symbolu `!codeToBeReplaced!`. Používá se pro přípravu prostředí nebo vyhodnocení.

`hint` Textová nápověda pro uživatele, pokud je pro cvičení definována. Pokud je prázdný řetězec, odkáže uživatele na komunitní Discord server.

`sandbox` Konfigurace pro spouštění kódu v izolovaném prostředí (sandboxu). Sandbox umožňuje nastavit oprávnění, např. povolení spouštění skriptů (`allow-scripts`) nebo stahování souborů (`allow-downloads`). V kontextu této práce nemusí být relevantní, pokud se iframe nepoužívá.

`instructions` Pole objektů, kde každý objekt reprezentuje jednu instrukci nebo krok, který má uživatel splnit. Každý objekt instrukce obsahuje následující.

- `text`: Textový popis instrukce zobrazovaný uživateli.
- `goalPosition` (volitelné): Pole souřadnic `[x, y, z]` definující cílovou pozici pro vizualizaci (např. kam se má postavička dostat).
- `test`: Řetězec definující podmínku pro ověření splnění instrukce. Existují následující dva režimy.
  - **Výchozí (Regulární výraz):** Řetězec je interpretován jako regulární výraz, který se aplikuje na kód vytvořený uživatelem.
  - **Test výstupu (JavaScript):** Pokud řetězec začíná prefixem `TEST:`, zbytek řetězce je interpretován jako JavaScriptový kód. Tento kód vyhodnocuje *výstup* nebo stav po spuštění uživatelského kódu (např. kontroluje aktuální pozici objektu `position` vůči cílové pozici `goalPos`). Funkce by měla vrátit `true` při splnění podmínky, jinak `false`.

6.4.3 Implementace editoru pro vizuální programování (Blockly)

Blockly[3] je nástroj pro vizuální programování, který umožňuje uživatelům sestavovat programy pomocí bloků, které reprezentují jednotlivé programové konstrukce. Tento způsob programování je vhodný zejména pro studenty, kteří se s programováním setkávají poprvé, a tím pádem by řádkové programování mohlo být příliš komplexní a matoucí. Pro implementaci vizuálního editoru byla primárně využita knihovna `react-blockly`. Ta poskytuje React komponentu `BlocklyWorkspace`, která zapouzdřuje základní funkcionalitu Blockly editoru. Komponentě se předávají zejména dvě klíčové vlastnosti (props).

- Konfigurace sady nástrojů (*toolbox configuration*): Definuje, které bloky jsou uživateli k dispozici.
- Zpětná vazba (callback) `onWorkspaceChange`: Funkce, která je volána při jakékoliv změně v pracovním prostoru editoru. Tato funkce se využívá k průběžné serializaci stavu bloků a ke generování spustitelného kódu.

Definice vlastních bloků Každý vizuální blok v Blockly je definován pomocí objektu v JSON formátu, který popisuje jeho vzhled, chování a vlastnosti. Níže je uveden příklad definice pro vlastní blok `move_forward`:

```
{
    "type": "move_forward",
    "message0": "Dopředu o %1 kroků",
    "args0": [
        {
            "type": "field_number",
```

[3] Odkaz na Blockly: https://developers.google.com/blockly

```
            "name": "step_count",
            "value": 10
        }
    ],
    "previousStatement": null,
    "nextStatement": null,
    "colour": 100,
    "tooltip": "Posune postavu dopředu",
    "helpUrl": ""
}
```

Popis polí v definici bloku.

`type` Jedinečný textový identifikátor typu bloku. Musí být unikátní v rámci všech používaných bloků.

`message0` Textový řetězec zobrazovaný na bloku. Symboly jako `%1` se zamění za hodnotu argumentu bloku.

`args0` Pole objektů definujících argumenty (vstupy, pole) bloku, které odpovídají zástupným symbolům v `message0`. Každý objekt argumentu má minimálně následující hodnoty.

- `type`: Typ pole (např. `field_number` pro číselný vstup, `field_input` pro textový vstup, `input_value` pro připojení jiného bloku vracejícího hodnotu).
- `name`: Unikátní název pole v rámci bloku, používaný pro získání jeho hodnoty v generátoru kódu.
- `value` (pro pole jako `field_number`): Výchozí hodnota pole.

`previousStatement` Určuje, zda lze k horní části tohoto bloku připojit jiný blok. Hodnota `null` znamená, že ano (lze ho zařadit do sekvence). Může obsahovat i pole typů bloků, které lze připojit.

`nextStatement` Určuje, zda lze k dolní části tohoto bloku připojit další blok. Hodnota `null` znamená, že ano.

`colour` Číselná hodnota (0-360) definující barevný odstín bloku pro vizuální rozlišení.

`tooltip` Text, který se zobrazí jako nápověda, když uživatel najede myší na blok.

`helpUrl` URL adresa stránky s podrobnější nápovědou k bloku (pokud existuje).

6.4.4 Převod Blockly bloků na kód

Převod Blockly bloků na kód je jedním z klíčových procesů pro to, aby mohl být vizuální výstup spuštěn a tím zprovozněno interaktivní blokové programovací prostředí. Tento proces je detailně popsán v následujících sekcích.

Principy převodu

Převod Blockly bloků na kód funguje na základě generátorů kódu, které jsou implementovány jako JavaScriptové funkce. Každý typ bloku má odpovídající generátor, který definuje, jak bude daný blok přeložen do textové podoby. Tento proces zahrnuje položky vypsané v následujícím seznamu.

- **Definici syntaxe:** Každý blok má specifickou strukturu, která odpovídá části kódu (např. podmínka, cyklus, příkaz).
- **Zpracování vstupů:** Bloky mohou obsahovat vstupy, které jsou převáděny jako parametry nebo součást kódu.
- **Rekurzivní generování:** Vnořené bloky jsou zpracovány rekurzivně, což umožňuje vytvořit složitější struktury, jako jsou cykly nebo funkce.

Implementace převodu

Blockly poskytuje nástroje pro implementaci generátorů kódu v různých programovacích jazycích. Například generátor pro JavaScript může být definován takto:

```
Blockly.JavaScript['controls_if'] = function(block) {
  var condition = Blockly.JavaScript.valueToCode(
    block,
    'IF',
    Blockly.JavaScript.ORDER_NONE
  );
  var statements = Blockly.JavaScript.statementToCode(block, 'DO');
  var code = 'if (' + condition + ') {\n' + statements + '}\n';
  return code;
};
```

Toto vygeneruje JavaScript kód, který může být následně spuštěn například pomocí funkce `eval()`.

Ukázkový proces převodu

Uvažujme následující příklad s vizuálním blokem reprezentujícím cyklus.

- Blok obsahuje počet opakování a vnořený příkaz.
- Generátor kódu převede blok na textový kód, například:

  ```
  for (let i = 0; i < 10; i++) {
    console.log('Hello, world!');
  }
  ```

Tento proces dělá z Blockly mocný nástroj pro výuku programování, umožňující plynulý přechod od vizuálního kódu k textovému programování.

6.4.5 Implementace řádkového editoru kódu

Pro umožnění psaní a úpravy kódu v řádkovém formátu (např. JavaScript, Python) byla do aplikace integrována knihovna `@monaco-editor/react`. Tato knihovna zpřístupňuje výkonný editor kódu Monaco Editor (vyvíjený Microsoftem, používaný např. ve Visual Studio Code) jako React komponentu. Komponentě `Editor` se předávají zejména následující vlastnosti.

- `language`: Určuje jazyk programování pro syntax highlighting a další jazykově specifické funkce (např. `'javascript'`, `'python'`).
- `value`: Řetězec obsahující aktuální kód zobrazený v editoru. Tento stav je spravován v nadřazené React komponentě.
- `onChange`: Funkce (callback), která je volána při každé změně obsahu editoru uživatelem. V této funkci se obvykle aktualizuje stav kódu v React komponentě, což umožňuje aplikaci reagovat na uživatelské úpravy a případně kód dále zpracovávat (např. pro spuštění nebo validaci).

Integrace Monaco Editoru poskytuje uživatelům pokročilé funkce známé z moderních IDE, jako je zvýrazňování syntaxe, automatické doplňování (IntelliSense pro podporované jazyky), validace kódu a další, což výrazně zlepšuje uživatelský zážitek při psaní kódu.

6.4.6 Integrace 3D vizualizace pomocí React Three Fiber

Pro vizualizaci výsledků algoritmů, zejména těch, které zahrnují pohyb v prostoru (jako v kurzu algoritmizace s postavičkou pohybující se po mřížce), byla klíčová implementace interaktivního 3D prostředí. K tomuto účelu byla v rámci frontendové části aplikace zvolena knihovna `React Three Fiber` (`R3F`).

Volba technologie `React Three Fiber` není samostatná 3D knihovna, ale jedná se o React renderer pro populární a výkonnou WebGL knihovnu `Three.js`. Zatímco přímé použití `Three.js` je často imperativní a může být složitější integrovat do deklarativního paradigmatu Reactu, `R3F` umožňuje vytvářet a spravovat komplexní `Three.js` scény pomocí známých React komponent a principů (komponenty, hooky, stav). To výrazně zjednodušuje vývoj a údržbu 3D části aplikace v rámci celkového Next.js/React projektu.

Princip implementace Základem vizualizace je React komponenta obsahující komponentu `<Canvas>` z knihovny `R3F`, která vytváří prostor pro 3D scénu. Uvnitř této komponenty jsou definovány následující komponenty reprezentující jednotlivé prvky scény.

- **Statické prostředí:** Prvky jako hrací plocha (mřížka), cílové políčko, případné překážky. Tyto jsou typicky definovány pomocí `R3F` komponent pro geometrie (např. `<planeGeometry>`, `<boxGeometry>`) a materiály (např. `<meshStandardMaterial>`).
- **Dynamický objekt (postavička):** Objekt, jehož stav (pozice, rotace) je přímo řízen výstupem algoritmu sestaveného uživatelem. Tento objekt je rovněž React komponenta, jejíž atributy (např. `position`, `rotation`) jsou navázány na stav spravovaný v Reactu.
- **Scéna a osvětlení:** Nastavení kamery (perspektiva, pozice), okolního a směrového světla pro zajištění viditelnosti a vizální kvality scény.

Propojení s logikou algoritmu Výstup algoritmu sestaveného uživatelem (typicky pomocí Blockly bloků v kurzu algoritmizace) není přímo 3D kód. Generátor kódu (např. Blockly do JavaScriptu) vytvoří sekvenci příkazů nebo funkcí, které reprezentují logické kroky algoritmu (např. `moveForward()`, `turnLeft()`). Tato sekvence je následně interpretována na straně klienta (v Reactu).

Interpretace probíhá tak, že jednotlivé kroky algoritmu postupně aktualizují React stav, který uchovává aktuální pozici a rotaci postavičky. Například volání `moveForward()` aktualizuje stavové proměnné pro souřadnice `x` nebo `z`. Jelikož jsou atributy 3D modelu postavičky v `R3F` navázány na tento React stav, jakákoli změna stavu automaticky vyvolá překreslení (*re-render*) 3D scény a postavička se vizuálně posune nebo otočí. Pro plynulé animace mezi kroky je využit hook `useFrame` poskytovaný `R3F`, který umožňuje provádět aktualizace v každém snímku animace.

Tímto způsobem `React Three Fiber` efektivně propojuje logiku algoritmů vytvořených uživateli s okamžitou a názornou 3D vizuální zpětnou vazbou.

6.4.7 Integrace s AI (chat bot) pro nápovědu

Pro zlepšení studijní zkušenosti a poskytnutí okamžité podpory studentům při řešení cvičení byla do aplikace integrována funkce nápovědy založená na umělé inteligenci ve formě chat bota. Cílem bylo nabídnout studentům interaktivního asistenta schopného odpovídat na dotazy týkající se programování a algoritmizace přímo v kontextu daného cvičení.

Použitá technologie Implementace chat bota se opírá o velký jazykový model (LLM) `GPT-4o mini`. Komunikace s modelem probíhá prostřednictvím API rozhraní poskytovaného společností `OpenAI`. Volba modelu `GPT-4o mini` byla motivována jeho schopnostmi v oblasti porozumění a generování textu relevantního pro programování a zároveň optimalizací pro rychlost, efektivitu a cenu.

Integrace do uživatelského rozhraní V rozhraní pro řešení cvičení je uživateli k dispozici pod instrukcemi textové pole, kam může student napsat svůj dotaz v přirozeném jazyce. Rozhraní zobrazuje historii konverzace a umožňuje tak dialog s AI.

Systémový dotaz odeslán s uživatelským dotazem Pro uvedení kontextu se zároveň s dotazem uživatele odesílá i systémový dotaz. Tento dotaz se liší pro kurz algoritmizace a pro ostatní kurzy, jelikož u kurzu algoritmizace je potřeba jiný formát z důvodu blokového programování. V případě kurzu algoritmizace je systémový dotaz následující:

```
'Odpovídej jen na otázky o~programování. Uživatel bude posílat otázky v~ná
    sledujícím formátu:\n' +
'Kód\n' +
'Má otázka je: Question\n' +
'Odpověz na otázku, kód je pouze kontext.\n' +
'Cílem je dostat postavu na pozici cíle (zelený čtverec) v~2D prostoru. Neř
    íkej uživateli o~souřadnicích, místo toho cíl je zelený čtverec, takže
    použij to. Postava začíná na (0,0) a je otočena na (0,1). Neříkej už
    ivateli konkrétní kód, jen se snaž je vést k~odpovědi. Překážky jsou na
     souřadnicích: [].\n' +
' Uživatel má splnit následující instrukce:\n' +
'1. Posuňte se na zelené políčko.\n' +
```

```
' na pozici: (0, 2)\n' +
'Uživatel vytváří kód pomocí následujících bloků: Start, Dopředu o~\%1 krok
    ů. Některé bloky obsahují symbol \% a číslo. Toto nahraď symbolem x.
    Toto je pouze zástupce nějaké hodnoty. Nejčastěji čísla.'
```

a v případě ostatních kurzů je systémový dotaz následující:

```
'Odpovídej pouze na otázky týkající se programování. Uživatel bude posílat
    zprávy ve tvaru:\n' +
'kód\n' +
'Má otázka je: Otázka\n' +
'odpovídej na otázku, kód je jen kontext.\n' +
'Programuje se v~javascript\n' +
'Uživatel má splnit následující instrukce:\n' +
'1. <p>V editoru vpravo použijte příkaz <code>console.log()</code> pro vý
    pis vašeho věku. Poté klikněte na tlačítko spustit.</p>\n' +
'2. <p>Na dalším řádku použijte znova příkaz <code>console.log()</code>,
    který vypíše vaše jméno. Nezapomeňte na uvozovky.</p>\n'
```

Tento systémový dotaz je v každém cvičení jiný, jelikož obsahuje instrukce a kontext daného cvičení.

Schopnosti a zaměření chatbota AI asistent je primárně zaměřen na pomoc s:

- **vysvětlením programovacích konstrukcí:** například objasnění principu podmínek, cyklů nebo proměnných,
- **řešením algoritmických úloh:** poskytnutí obecných rad nebo návodů k postupu, aniž by přímo prozradil řešení,
- **technickými dotazy:** Odpovědi na otázky týkající se syntaxe použitého jazyka (v případě kurzu algoritmizace konceptů bloků) nebo interpretace chybových hlášení (pokud jsou relevantní).

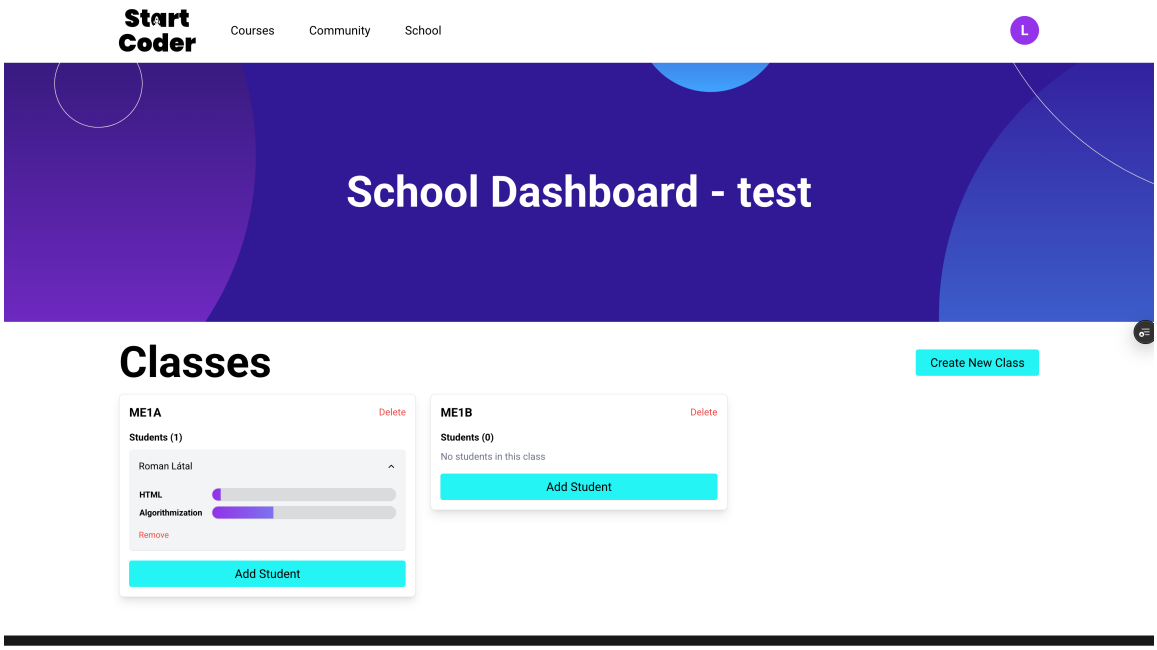
Důraz je kladen na schopnost reagovat na dotazy formulované přirozeným jazykem a poskytovat srozumitelné a věcné odpovědi.

Integrace AI chatbota představuje moderní doplněk interaktivní výuky, který zvyšuje její efektivitu a dostupnost podpory pro studenty.

6.5 Administrační rozhraní pro učitele

Klíčovou součástí aplikace pro podporu výuky je administrační rozhraní navržené speciálně pro potřeby učitelů. Tento panel slouží jako centrální místo pro správu tříd a studentů, sledování jejich pokroku a získávání přehledu o aktivitách v rámci výukových kurzů. Grafická podoba je vyobrazena na obrázku 6.3.

Účel a přístup Hlavním cílem tohoto rozhraní je poskytnout učitelům efektivní nástroje pro monitorování a řízení vzdělávacího procesu v rámci platformy. Přístup do panelu je podmíněn autentizací uživatele s rolí `TEACHER`, čímž je zajištěno, že k citlivým datům o studentech mají přístup pouze oprávněné osoby.



Obrázek 6.3: Implementovaná obrazovka administračního rozhraní pro učitele.

Klíčové funkce Administrační panel nabízí učitelům následující základní funkcionality.

- **Přehled tříd a skupin:** Učitelé mohou vidět seznam tříd, které spravují.
- **Seznam studentů ve skupině:** V rámci každé skupiny je zobrazen seznam přiřazených studentů.
- **Sledování pokroku studentů:** Pro každého studenta je možné sledovat postup v jednotlivých kurzech. Rozhraní využívá vizualizace, aby byl pokrok snadno čitelný.
- **Správa studentů:** Možnost přidávat a odebírat studenty mezi třídami.
- **Export dat:** Možnost přidávat a odebírat jednotlivé třídy.

Návrh rozhraní a technologie Návrh uživatelského rozhraní klade důraz na **přehlednost a intuitivní ovládání**. Data jsou prezentována strukturovaně, aby učitelé mohli rychle identifikovat potřebné informace. Rozhraní je navrženo s ohledem na jednoduchost a efektivitu.

Pro implementaci interakcí, které vyžadují změnu dat na serveru (vytváření, čtení, aktualizace, mazání – CRUD operace), byly zvoleny `Next.js Server Actions`. Tento přístup představuje moderní alternativu ke klasickým API routes v rámci ekosystému Next.js.

Porovnání Server Actions a API Routes.

- **Tradiční API Routes:** Při použití API routes by klientská část aplikace (React komponenty) musela explicitně vytvářet HTTP požadavky (např. `POST` pro vytvoření, `PUT` pro update, `DELETE` pro smazání) na specifické endpointy definované v adresáři `/app/api`. Každý endpoint by obsahoval serverovou logiku pro zpracování požadavku, interakci s databází a odeslání odpovědi (obvykle ve formátu JSON). Tento model

vyžaduje manuální správu `fetch` volání na klientovi, ošetření stavů (načítání, chyba, úspěch) a definici samostatných API handlerů na serveru.

- **Next.js Server Actions:** Server Actions umožňují definovat asynchronní funkce, které běží *výhradně na serveru*, ale mohou být volány přímo z klientských nebo serverových komponent (např. při odeslání formuláře nebo kliknutí na tlačítko). Next.js se postará o bezpečné vytvoření RPC (Remote Procedure Call) spojení mezi klientem a serverem. Funkce Server Action pak provede potřebné operace (např. zápis do databáze) a může přímo ovlivnit UI, například pomocí revalidace dat (aktualizace zobrazených informací) nebo přesměrování. V tomto případě revalidujeme data pomocí funkce `revalidatePath(path: string)`, která revaliduje určitou cestu. V tomto případě cestu `/[lang]/school`.

Výhody použití Server Actions v tomto kontextu.

- **Zjednodušení kódu:** Odpadá potřeba psát boilerplate kód pro API endpointy a klientské `fetch` funkce pro mutace dat. Logika pro změnu dat je zapouzdřena přímo v Server Action funkci.
- **Typová bezpečnost:** Jelikož je využíván TypeScript, tak je zajištěna typová bezpečnost end-to-end, protože voláme přímo serverovou funkci s definovanými typy argumentů a návratových hodnot.
- **Progresivní vylepšení (Progressive Enhancement):** Formuláře využívající Server Actions mohou fungovat i bez JavaScriptu na klientovi, což zvyšuje robustnost aplikace.

Volba Server Actions pro administrační panel tedy přináší zjednodušení vývoje, lepší typovou bezpečnost a efektivnější způsob provádění CRUD operací ve srovnání s tradičním vytvářením dedikovaných API endpointů.

Toto rozhraní představuje nezbytný nástroj umožňující efektivní využití výukové aplikace ve školním prostředí, jelikož umožňuje učitelům aktivně se zapojit do procesu učení a poskytovat cílenou podporu studentům.

Kapitola 7

Testování a nasazení

Po dokončení hlavní fáze implementace bylo nezbytné aplikaci důkladně otestovat a následně nasadit do produkčního prostředí, aby byla dostupná koncovým uživatelům – studentům a učitelům.

7.1 Testování

Cílem testování bylo ověřit funkčnost klíčových součástí aplikace, identifikovat a opravit chyby a zajistit co nejlepší uživatelský zážitek. Bylo provedeno uživatelské testování podle frameworku Rocket Surgery Made Easy, které se zaměřilo na následující. [11]

- **Funkčnost jádra:** Ověření správné funkce registrace a přihlášení uživatelů (včetně SSO a e-mailu), procházení kurzů, spouštění a vyhodnocování cvičení (jak s Blockly, tak s řádkovým editorem), ukládání postupu uživatele.
- **Interaktivní prvky:** Testování editoru Blockly (přidávání, spojování, mazání bloků), 3D vizualizace (správné zobrazení a reakce na kód), řádkového editoru (psaní kódu, zvýrazňování syntaxe).
- **Uživatelské rozhraní (UI) a Zážitek (UX):** Kontrola přehlednosti a intuitivnosti ovládání, vizuální konzistence napříč aplikací, responzivity designu na různých velikostech obrazovky.
- **Zpětná vazba od uživatelů:** V rámci práce byl proveden dotazník (viz sekce 5.6), který poskytl cennou zpětnou vazbu na preferenci 2D vs 3D vizualizace a sloužil i jako forma neformálního uživatelského testování použitelnosti prototypu.
- **Integrace AI:** Ověření funkčnosti chat bota pro nápovědu a jeho schopnosti poskytovat relevantní odpovědi.

Z testování vyplynulo pár opakujících se problémů.

- **Slabá zpětná vazba:** Při řešení cvičení uživatelům přišlo, že je nedostatek zpětné vazby. Toto bylo vyřešno pomocí vyskakovacího toastu v pravé dolní části obrazovky při splnění cvičení a pomocí scrollu k instrukcím při stisknutí tlačítka spustit.
- **Neočekávané chování tlačítka reset:** Při stisknutí tlačítka reset v kurzu algoritmizace uživatelé očekávali resetování jen pozice objektu u výstupu a ne jejich kódu. Toto bylo následně upraveno tak, aby to souhlasilo s uživatelskými očekáváními.

- **Matoucí tlačítko pomoc:** Při stisknutí tlačítka pomoc se zjevil dropdown, který obsahoval nápis Potřebujete pomoc?, toto bylo pro uživatele matoucí a nevěděli, že je tento nápis odkaz. Proto byl tento nápis upraven na: Zeptejte se na discordu.

Ačkoliv nebyly implementovány automatizované testy (jako unit testy nebo integrační testy), manuální ověření pokrylo nejdůležitější scénáře používání aplikace.

7.2 Nasazení (Deployment)

Pro zajištění dostupnosti, výkonu a snadné správy byla aplikace a její databáze nasazena na moderní cloudové platformy.

7.2.1 Webová aplikace (Frontend a Backend API)

Samotná webová aplikace, postavená na frameworku Next.js, je nasazena na platformě **Vercel**. Toto řešení bylo zvoleno z několika důvodů.

- **Přímá integrace s Next.js:** Vercel je vyvíjen stejnou společností jako Next.js, což zajišťuje optimální výkon a podporu všech jeho funkcí (SSR, SSG, API routes atd.).
- **Snadné nasazení a CI/CD:** Propojením s Git repozitářem (např. GitHub, GitLab) Vercel automaticky sestavuje (build) a nasazuje (deploy) každou novou verzi aplikace po commitu do hlavní větve. Tento proces Continuous Integration / Continuous Deployment (CI/CD) výrazně zjednodušuje aktualizace.
- **Globální CDN (Content Delivery Network):** Statické části aplikace jsou automaticky distribuovány prostřednictvím globální sítě serverů, což zajišťuje rychlé načítání pro uživatele po celém světě.
- **Serverless Functions:** API routes vytvořené v Next.js jsou na Vercelu provozovány jako serverless funkce, které automaticky škálují podle zátěže a nevyžadují správu serverové infrastruktury.

Nasazení na Vercel tak poskytuje robustní, výkonné a snadno spravovatelné prostředí pro provoz moderní webové aplikace.

7.2.2 Databáze

Jako databázové řešení byla zvolena dokumentová databáze MongoDB. Pro její provoz v produkčním prostředí je využita cloudová služba **MongoDB Atlas**. Jedná se o plně spravovanou databázovou platformu (Database-as-a-Service – DBaaS), která nabízí řadu výhod.

- **Snadná správa:** Odpadá nutnost instalace, konfigurace, monitorování a údržby databázových serverů.
- **Škálovatelnost:** MongoDB Atlas umožňuje snadno horizontálně i vertikálně škálovat výkon a kapacitu databáze podle aktuálních potřeb aplikace.
- **Vysoká dostupnost a spolehlivost:** Platforma zajišťuje redundanci a automatické zálohování dat pro minimalizaci rizika výpadků a ztráty dat.

- **Bezpečnost:** Poskytuje pokročilé bezpečnostní funkce, včetně řízení přístupu, šifrování dat a síťové izolace.

Použití MongoDB Atlas zajišťuje spolehlivé, bezpečné a škálovatelné úložiště pro všechna data aplikace StartCoder, jako jsou informace o uživatelích, kurzech a jejich postupu.

7.2.3 Dostupnost aplikace a prezentačního videa

Webová aplikace StartCoder je nasazena a dostupná veřejnosti na adrese:

https://startcoder-git-blockly-widewebs-projects.vercel.app/cs

Pro účely prezentace a splnění požadavků zadání bylo rovněž vytvořeno prezentační video, které demonstruje hlavní funkce aplikace. Video je dostupné na následující adrese:

https://drive.google.com/file/d/16EA1ECcLda6YLQURv5ewRKhvNIF6h63d/view

Tímto způsobem je aplikace i ukázka jejího fungování snadno dostupná pro hodnotitele a všechny zájemce o prohlídku výsledného produktu.

Kapitola 8

Závěr

Cílem této bakalářské práce bylo navrhnout a implementovat webovou aplikaci pro interaktivní výuku programování, která by lépe odpovídala potřebám českého středoškolského prostředí než stávající komerční řešení. Tento cíl byl úspěšně splněn vytvořením funkčního prototypu aplikace StartCoder.

V rámci práce byly splněny všechny body formálního zadání. Byla provedena analýza existujících výukových platforem a metod interaktivní výuky programování, na základě které byly identifikovány klíčové nedostatky pro školní prostředí, jako absence češtiny, nedostatečná interaktivita či vysoká cena. Dále byly analyzovány požadavky Rámcových vzdělávacích programů a konkrétní potřeby Střední průmyslové školy a Obchodní akademie Uherský Brod. Na základě těchto analýz byl navržen koncept a uživatelské rozhraní aplikace, včetně struktury kurzů, interaktivních cvičení a přehledu pro učitele. Následně byla aplikace implementována s využitím moderních webových technologií – frontend v Reactu s frameworkem Next.js a stylováním pomocí Tailwind CSS, backendová logika částečně zajištěna pomocí API routes v Next.js, databáze MongoDB hostovaná na Atlas a pro vizuální programování byla integrována knihovna Blockly. Byla implementována interaktivní cvičení s 3D vizualizací pomocí React Three Fiber a základní funkcionalita pro správu uživatelů a kurzů. Nakonec byla aplikace otestována a nasazena na platformy Vercel a MongoDB Atlas.

Výsledkem práce je webová aplikace StartCoder, která nabízí kurzy HTML, CSS, JavaScript, Python a algoritmizace s využitím vizuálního programování Blockly a 3D vizualizace. Aplikace podporuje registraci a přihlášení uživatelů (včetně SSO přes Google, Facebook, X a Microsoft pro školy), ukládá postup studentů a poskytuje okamžitou zpětnou vazbu při řešení cvičení. Součástí je i základní administrační rozhraní pro učitele a integrace AI chat bota pro nápovědu studentům. Celá aplikace je dostupná v češtině a angličtině.

Tato práce mi poskytla cenné zkušenosti s komplexním vývojem moderní webové aplikace od analýzy požadavků, přes návrh architektury a UI/UX, až po implementaci s využitím široké škály technologií a následné nasazení. Naučil jsem se pracovat s frameworkem Next.js, knihovnami React, Blockly, React Three Fiber, Tailwind CSS a databází MongoDB, stejně jako s principy server-side renderingu a integrací externích služeb jako NextAuth, Sentry a AI modelů.

V práci bych chtěl v budoucnu pokračovat rozšířením obsahu aplikace o další kurzy, zejména o kurz jazyka C++, který je požadován partnerskou školou. Dále bych se chtěl zaměřit na detailnější propracování přehledů pro učitele, včetně možnosti zadávat vlastní úkoly a sledovat detailnější statistiky studentů. Rovněž bych chtěl implementovat robustnější systém testování (unit a integrační testy).

Další potenciální směry rozvoje, které by mohl řešit někdo jiný, zahrnují například přidání možnosti spolupráce studentů na projektech v reálném čase, rozšíření o další programovací jazyky, gamifikaci výuky (získávání bodů, odznaků), pokročilejší integraci AI pro personalizovanou zpětnou vazbu nebo vytvoření mobilní verze aplikace.

Věřím, že vytvořená aplikace StartCoder představuje solidní základ pro moderní a efektivní nástroj podporující výuku programování na českých základních a středních školách.

Literatura

[1] Chen, C. a Guo, P. J. Improv: Teaching Programming at Scale via Live Coding. *Proceedings of the Sixth (2019) ACM Conference on Learning @ Scale*. 1. vyd., 2019, č. 9. Dostupné z: https://api.semanticscholar.org/CorpusID:150384666.

[2] *Contrast (Minimum) (Level AA)* online. 2025. Dostupné z: https://www.w3.org/WAI/WCAG21/Understanding/contrast-minimum.html.

[3] *Fundamentals of Web Application Development* online. N.d. Dostupné z: https://coreteka.com/blog/fundamentals-of-web-application-development/. [cit. 2025-04-23].

[4] Hutsulyak, O. *10 Key Reasons Why You Should Use React for Web Development* online. 2024. Dostupné z: https://www.techmagic.co/blog/why-we-use-react-js-in-the-development.

[5] Ihnatovich, D. *CSS Modules vs CSS-in-JS vs Tailwind CSS: A Comprehensive Comparison* online. 2024. Dostupné z: https://medium.com/@ignatovich.dm/css-modules-vs-css-in-js-vs-tailwind-css-a-comprehensive-comparison-24e7cb6f48e9.

[6] *Introduction • Docs • Svelte* online. 2025. Dostupné z: https://svelte.dev/docs/kit/introduction.

[7] *Understand the life cycle of Web Application Development* online. 2024. Dostupné z: https://www.iprogrammer.com/web-application-development-life-cycle-a-2024-brief-guide-for-dedicated-developers/. [cit. 2025-04-23].

[8] Jasper, A. *7 Compelling Reasons Why VueJS is the Perfect Framework for Modern Businesses* online. 2023. Dostupné z: https://www.pixelcrayons.com/blog/dedicated-teams/why-use-vuejs/.

[9] Jasper, A. *Node.js vs. Python vs. Java: Choosing the Right Back-End Technology* online. 2024. Dostupné z: https://dev.to/angelinajasper/nodejs-vs-python-vs-java-choosing-the-right-back-end-technology-5dj.

[10] Karan, M. *When to use Svelte?* online. 2020. Dostupné z: https://dev.to/mikehtmlallthethings/when-to-use-svelte-120a.

[11] Krug, S. *Rocket Surgery Made Easy: The Do-It-Yourself Guide to Finding and Fixing Usability Problems*. 1st. USA: New Riders Publishing, 2009. ISBN 0321657292.

[12] Krug, S. *Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability*. 3. vyd. New Riders, 2014. ISBN 978-0321965516.

[13] Krzywda, K. *Key Benefits of Next.js - When to Use It and What Is It Good For?* online. 2024. Dostupné z: https://naturaily.com/blog/nextjs-benefits.

[14] *MongoDB: The World's Leading Modern Database* online. N.d. Dostupné z: https://www.mongodb.com/. [cit. 2025-04-23].

[15] *Provázíme děti, učitele i rodiče digitálním světem* online. 2024. Dostupné z: https://revize.rvp.cz/digitalizace. [cit. 2024-06-10].

[16] *Relational vs Nonrelational Databases - Difference Between Types of Databases - AWS* online. 2025. Dostupné z: https://aws.amazon.com/compare/the-difference-between-relational-and-non-relational-databases/.

[17] Ruiz, D. *10 reasons to use Nuxt.js for your next web application* online. 2018. Dostupné z: https://medium.com/vue-mastery/10-reasons-to-use-nuxt-js-for-your-next-web-application-522397c9366b.

[18] Sandler, S. *Building skills through online learning: 4 data-driven best practices.* 2020. Dostupné z: https://blog.coursera.org/how-the-best-online-courses-engage-students-build-skills-and-drive-career-outcomes/.

[19] *Server-side Rendering (SSR) – Next.js* online. N.d. Dostupné z: https://nextjs.org/docs/pages/building-your-application/rendering/server-side-rendering. [cit. 2025-04-23].

[20] Singh, T. *JavaScript Frameworks in 2024: React vs. Vue vs. Svelte - Which One to Choose?* online. 2024. Dostupné z: https://dev.to/tarunsinghofficial/javascript-frameworks-in-2024-react-vs-vue-vs-svelte-which-one-to-choose-4c0p. [cit. 2025-04-23].

[21] *Static Site Generation (SSG) – Next.js* online. N.d. Dostupné z: https://nextjs.org/docs/pages/building-your-application/rendering/static-site-generation. [cit. 2025-04-23].

[22] *Tailwind CSS – A Utility-First CSS Framework for Rapidly Building Custom Designs* online. N.d. Dostupné z: https://tailwindcss.com/. [cit. 2025-04-23].

[23] UXPin. *The Importance of Consistency in UX/UI Design* online. 2024. Dostupné z: https://www.uxpin.com/studio/blog/guide-design-consistency-best-practices-ui-ux-designers/.