Univerzita Karlova

Pedagogická fakulta

Katedra informačních technologií a technické výchovy

BAKALÁŘSKÁ PRÁCE

Webová aplikace pro podporu výuky algoritmizace

A web application to support the teaching of algorithmization

Petr Sysel

Vedoucí práce:	PhDr. Josef Procházka, Ph.D.
Studijní program:	Informační technologie se zaměřením na vzdělávání (B0114A140004)
Studijní obor:	B IT 20 (0114RA140004)

2024

Odevzdáním této bakalářské práce na téma **Webová aplikace pro podporu výuky algoritmizace** potvrzuji, že jsem ji vypracoval pod vedením vedoucího práce samostatně za použití v práci uvedených pramenů a literatury. Dále potvrzuji, že tato práce nebyla využita k získání jiného nebo stejného titulu.

V Praze 15.4. 2024

Děkuji PhDr. Josefu Procházkovi Ph.D. za jeho trpělivost, odborné rady a veškerou pomoc při vedení bakalářské práce. Mé poděkování patří též PhDr. Jiřímu Štípkovi Ph.D. za pomoc při ověřování praktických cílů práce.

ABSTRAKT

Bakalářská práce se zaměřuje na problematiku informatického a algoritmického myšlení, gamifikace a game-based learningu. Mapuje dostupné nástroje pro rozvoj algoritmického myšlení, jejich funkce a nástroje a snaží se nabídnou alternativu obohacenou o nástroje jako například editor vlastních úloh pro rozvoj algoritmického myšlení, který by učitelům poskytl vyšší kontrolu nad personalizací výukových materiálů. Výsledný prototyp aplikace bude otestován při hodině informatiky na prvním stupni základní školy.

KLÍČOVÁ SLOVA

Programování, algoritmizace, informatické myšlení, gamifikace, hra se vzdělávacím obsahem

ABSTRACT

The bachelor's thesis focuses on the issues of computational and algorithmic thinking, gamification, and game-based learning. It maps available tools for the development of algorithmic thinking, their functions, and features, aiming to offer an alternative enriched with tools, such as a custom puzzle editor for algorithmic thinking development, which would provide teachers with greater control over the personalization of educational materials. The resulting prototype of the application will be tested during a computer science class at the primary level of elementary school.

KEYWORDS

Programming, algorithmization, computational thinking, gamification, game with an educational content

Obsah

Úvod

V dnešní době, ve které jsou lidé obklopeni výpočetní technikou, začíná být čím dál důležitější umět těmto technologiím porozumět a správně je používat. Neustálý pokrok v oblasti informačních technologií ovlivňuje každodenní život lidí i fungování celé společnosti. Od práce přes zábavu až po běžnou komunikaci s rodinou a přáteli – digitální technologie pronikly do mnoha sfér našeho života a staly se nezbytnou součástí moderního světa. Začlenění informatického myšlení a algoritmizace do výuky se proto stává klíčovým krokem pro zkvalitnění vzdělávání v této oblasti.

Rozvíjení digitálních kompetencí již od útlého věku poskytuje žákům pevný základ pro pochopení principů digitálního světa a přípravu na budoucí výzvy, které s sebou moderní technologie přinášejí. Vzdělávání v oblasti informatického myšlení není pouze o informatice, ale také o schopnosti analyzovat a řešit problémy, logicky uvažovat a kreativně přistupovat k řešení úkolů. Tímto způsobem informatického myšlení připravuje žáky na úspěšné uplatnění v digitálním světě, který je stále složitější a dynamický (Lcom Team, 2023).

S nástupem digitálních technologií do tříd a neustále se vyvíjejícím prostředím digitálního vzdělávání vzniká stále větší potřeba inovativních a přizpůsobivých nástrojů pro podporu výuky nejen algoritmizace a informatického myšlení (Brown, 2023). I přesto, že existuje široká škála dostupných platforem a aplikací, které mají potenciál přinést nové způsoby, jak u žáků podporovat rozvoj algoritmické myšlení, stále existuje prostor pro vytvoření nových, specificky zaměřených nástrojů, které budou odpovídat konkrétním potřebám pedagogů a žáků (Repenning et al., 2016).

Cílem této práce je zmapovat dostupné nástroje pro podporu výuky algoritmizace a navrhnout alternativu obohacenou o chybějící funkce a nástroje vhodné pro použití při výuce algoritmizace na prvním stupni základní školy. Součástí práce bude návrh a implementace funkčního prototypu této aplikace, její otestování v hodině informatiky a ověření využitelnosti v praxi.

1 Algoritmizace a informatické myšlení

1.1 Informatické myšlení

Informatické myšlení navzdory názvu se nemusí nutně týkat pouze informatiky. Dnešní doba nabízí mnoho každodenních výzev, se kterými se musíme potýkat, a informatické myšlení nám v mnohých případech může poskytnout nástroje, jak se s nimi snáze vypořádat (IMyšlení: FAQ, 2018).

Jedna z definic říká, že jde o přemýšlení nad problémy tak, jak by je řešil informatik (Lessner, 2014). Ne každému musí tato definice plně vyhovovat, svádí totiž k představě něčeho vhodného či užitečného pouze informatikům. Vhodnější definice říká, že informatické myšlení je způsob uvažování nad problémem a jeho řešením založený na následujících bodech (Brdička, 2014):

- Formulace problému tak, aby při jeho řešení bylo možné využít digitální technologie
- Reprezentování dat pomocí modelů a simulací
- Automatizace řešení pomocí algoritmizace
- Identifikace, analýza a implementace možných řešení s co nejefektivnějším dosažením výsledku
- Zobecnění problému pro řešení většího množství podobných problémů

Pro osvojení algoritmického myšlení je nutné seznámit se se základními kameny celého konceptu: dekompozicí, abstrakcí, rozpoznávání vzorů a algoritmizací.

Dekompozice je umění rozdělit problém na několik menších a snáze řešitelných úkolů. Takovýmto rozdělením problému lze často zjistit, že řešení není tak komplikované, jak se na první pohled zdá (Grover, Pea, 2018).

Jako příklad ze života si lze představit přípravu obtížného pokrmu. Pokud se postup jeho přípravy rozdělí do několika dílčích kroků, získáme přehlednější pohled na řešený problém. Zvlášť bude probíhat příprava hlavního pokrmu, přílohy, omáčky, dipu a salátu. Tyto samostatné části lze znovu rozložit do menších částí, a to například rozdělením ingrediencí zvlášť pro každou část receptu.

Abstrakce se snaží v celku rozpoznat podstatné a odlišit to od méně podstatných detailů, které by jen odvracely pozornost jinam, či nás přiváděly k nesprávným závěrům. Jedná se o schopnost najít ve složitém jednoduché (Grover, Pea, 2018).

Při řešení problému je možné setkat se s podobností s něčím, co již bylo řešeno či s čím se již člověk, který úlohu řeší setkal. Proto je důležitá i schopnost rozpoznávání vzorů. Díky připodobnění k již vyřešenému dokáže tato schopnost velmi usnadnit proces hledání konečného východiska. Potom je možné se z řešení oněch problémů poučit či inspirovat a dostat se tak blíže ke konečnému rozuzlení (Grover, Pea, 2018).

Porozumění problému samo o sobě nestačí. Problémy je třeba řešit, a tak je pro informaticky smýšlejícího člověka důležité i umět správně navrhnout postup řešení a srozumitelně jej zaznamenat. Může se zdát, že se to již podobá programování – tedy sestavování postupu řešení problému v podobě programovacího jazyka tak, aby mu porozuměl počítač, ale algoritmizace má opět nezanedbatelné využití i v reálném světě (IMyšlení: FAQ, 2018). Postupy lze nalézt v kuchařkách, v krabicích se zakoupeným nábytkem a děti by jen velmi obtížně zkompletovaly model vyobrazený na krabici od LEGA, nebýt návodu, který v ní najdou spolu s kostičkami (Lcom Team, 2022).

Umět myslet informaticky se ukázalo natolik důležité, že byla výuka algoritmického myšlení zahrnuta do RVP „nové informatiky“ platné od září roku 2021 (Jak na novou informatiku v RVP ZV, 2021). Mnohé z toho, co se do té doby považovalo za výuku informatiky, bylo přesunuto pod informační gramotnost, která by měla být rozvíjena v oblastech s ní spojených (Černý, 2021; IMyšlení: FAQ, 2018).

Integrace informatického myšlení do vzdělávacího procesu nevyplývá pouze z jeho praktické důležitosti, ale také z jeho strategického začlenění do celkového konceptu vzdělávání. Důležitost rozvoje informatického myšlení se odráží i v interdisciplinárním konceptu vzdělávání STEAM (Science, Technology, Engineering, Art, Mathematics), který zdůrazňuje propojení mezi přírodními vědami, technologiemi, inženýrstvím, uměním a matematikou. STEAM představuje přístup ke vzdělávání, který podporuje kreativitu a schopnost řešit komplexní problémy. Informatika, jako klíčový prvek v této metodologii, žákům nabízí možnost propojení digitální gramotnosti s dalšími oblastmi, jako jsou umělecké a technické disciplíny. Tímto způsobem se informatika stává nedílnou součástí

celkového vzdělávacího prostředí, které podporuje rozvoj širokého spektra dovedností a kompetencí potřebných pro moderní společnost (Sieińska, Ordza, 2022).

1.2 Algoritmizace

Význam algoritmizace a to, proč má své místo v informatickém myšlení, již bylo zmíněno, nyní je potřeba definovat, co je to algoritmus. Jeho název vznikl polatinštěním jména perského matematika Muḥammad ibn Mūsā al-Khwārizmī (pro jednoduchost al-Khwarizmi), který v první polovině devátého století formuloval postupy pro počítání s arabskými číslicemi (Khnut, 1980). Definovat algoritmus je ovšem překvapivě obtížné. Některé zdroje se shodují, že jde o formální postup (dostatečně podrobný, aby byl srozumitelný pro počítač) řešení určitého problému (Singh, 2023a; Cormen et al., 2022.; Mareš, Valla, 2022).

S algoritmem pouze jako s postupem si ale vystačit nelze. Aby byl algoritmus algoritmem, musí mít následující vlastnosti (Singh, 2023a):

- Konečnost – Počet instrukcí musí být konečný a výsledek algoritmu musí být poskytnut v konečném čase. Výsledek poskytnutý za dva miliony let mnoho uplatnění nenajde.
- Hromadnost – algoritmus je možné použít k řešení obecnějších problémů stejného druhu (namísto vymýšlení algoritmu pro nalezení cesty z jednoho konkrétního bludiště lze napsat algoritmus, který bude platný pro aplikování na libovolné bludiště).
- Jednoznačnost – Algoritmus je složen z jednoduchých kroků, které na sebe navazují. Na základě stavu dat v algoritmu musí být jednoznačně určeno, který krok následuje.
- Opakovatelnost (determinističnost) – Pro stejná data na vstupu algoritmu musíme vždy obdržet stejná data na výstupu.
- Rezultativnost – Algoritmus musí vracet správný výsledek.

Samotnou Algoritmizaci potom lze označit jako proces tvorby algoritmu, tedy proces sestavování postupu za účelem řešení zadané úlohy (Baierlová, 2021).

1.3 Algoritmické myšlení

Z definice algoritmizace tak, jak byla zavedena výše, by se dalo algoritmické myšlení chápat jako způsob uvažování nad sestavováním postupu pro řešení určitého problému (Baierlová, 2021). S uměním sestavit algoritmus efektivně a tak, aby byl čitelný a proveditelný nejen jiným člověkem, ale i počítačem, nám pomáhá informatické myšlení (Shohieb et al., 2022).

Informatické a algoritmické myšlení sdílejí mnoho společných prvků jako třeba abstrakci a dekompozici. Algoritmické myšlení je ovšem více aplikované na konkrétní oblast sestavování postupu (Klement et al., 2020).

Chce-li člověk vymyslet postup pro dosažení určitého cíle, je nezbytné porozumět úloze a zjistit míru její abstrakce. Zjistit, kterými dílčími kroky se lze k požadovanému výsledku dopracovat. Zde je opět uplatňován princip dekompozice (Grover, Pea, 2018).

S algoritmickým myšlením se pojí i schopnost algoritmus vhodně vyjádřit lidským jazykem, diagramem či programovacím jazykem tak, aby byl „procesoru“ (tomu, kdo bude postup vykonávat, ať se jedná o stroj či jiného člověka) srozumitelný (Lessner, 2014).

2 Herní metody rozvoje algoritmizace na webu

Metod rozvíjejí algoritmické myšlení je celá řada. Často si lze vystačit dokonce i bez digitálních technologií. K tomu, aby se žáci seznámili se základy algoritmizace postačí tužka a papír. Vymyslet či najít v učebnicích nebo online můžeme spousty cvičení pro sestavování postupů a jejich zápis. Sice nemůžeme otestovat srozumitelnost takového postupu pro počítač, zato nám jako „procesor“ může posloužit spolužák (Klement et al., 2020).

Aby srozumitelnost pro stroj bylo možné otestovat, lze výuku algoritmizace přesunout k počítačům a využít dostupné digitální platformy pro vzdělávání v této oblasti informatiky, kterých na internetu existuje celá řada. Před prozkoumáním konkrétních platforem a jejich vlastností, je na místě specifikovat, kterými aplikacemi se práce bude zabývat.

Desktopová a mobilní řešení jsou z analýzy vynechány a práce se zabývá pouze aplikacemi webovými. Důvod upřednostnění webových aplikací je především ten, že pro využití v hodinách informatiky, nebo v zájmových útvarech mají takováto řešení rozhodující výhody. Například snadnou dostupnost bez nutnosti přímé instalace na školní zařízení. Tím se jednak zjednodušuje správa (nejsou nutné ruční aktualizace) i samotná instalace těchto

aplikací. Velmi často chybí lektorům a mnohdy i učitelům oprávnění instalovat jakýkoli software bez asistence administrátora místní sítě (K-Net Technical International Group, 2019). Webová řešení se bez tohoto kroku obejdou.

Práce se dále zaměřuje na možnosti zapojení herních prvků do vzdělávání v oblasti algoritmického a informatického myšlení. K lepšímu porozumění tématu je nutné vysvětlit některé pojmy.

2.1 Gamifikace a game-based learning

Při zapojování herních prvků do vzdělávání se můžeme setkat se dvěma zásadními pojmy, kterými jsou gamifikace a game-based learning.

Při gamifikaci jde o začleňování herních prvků a mechanismů do neherních oblastí. S pojmem gamifikace se tedy nemusíme setkat pouze ve vzdělávání. Tento princip se používá také v marketingu či firemním managementu. V marketingu může být gamifikace využita ke zvýšení zájmu klienta prostřednictvím získávání bodů, achievmentů, zvyšování úrovně a podobně. Poprvé se tento pojmem objevuje v roce 2002. Do povědomí širší veřejnosti se dostává až kolem roku 2010 (BasuMallick, 2022; Zichermann, Cunningham, 2011).

Ve vzdělávání funguje gamifikace stejně jako ve výše zmíněném příkladu marketingu. Pomocí začlenění herních prvků do výuky se snažíme zvýšit zájem studentů a žáků o probíranou látku a jejich zaangažovanost v hodinách. Díky mechanikám sbírání bodů či úrovní může gamifikace zvyšovat i soutěživost mezi žáky (Král, 2018).

Vhodnou ukázkou využití principů gamifikace v oblasti vzdělávání poskytuje aplikace Duolingo (Obrázek 1) pro učení se jazyků, která využívá například systému levelů uživatelů, průchod vzdělávacím obsahem formou úrovní, či sbírání předmětů, které mohou hráči poskytnout určitou výhodu při plnění výzev (například lektvar pro rychlejší nárůst zkušenostních bodů) (Bilham, 2021).

Obrázek 1 - Prostředí mobilní aplikace Duolingo, využívající prvků gamifikace

Nevhodně zvolená gamifikace či přílišná gamifikace ovšem může odvrátit pozornost žáků od probírané látky spíše k samotným herním prvkům. Je-li gamifikace použita správně a za správných podmínek, může pozitivně ovlivnit výsledek vzdělávacího procesu (Král, 2018).

Game-based learning, volně přeložený jako výuka založená na hře, se na rozdíl od gamifikace snaží do výuky vetknou celé hry, které jsou vytvořené či vhodné pro edukativní účely. Výhody a nevýhody jsou velice podobné gamifikaci. Game-based learning může zvyšovat zaangažovanost žáků a jejich motivaci (Plass et al., 2016; MCGonigal, 2011).

Obě metody mohou ve vyvážené podobě přispět k efektivnějšímu vyučování, a navíc navodit uvolněnou atmosféru v hodinách (MCGonigal, 2011).

2.2 Rozvoj algoritmizace pomocí aplikací k tomu určených

Webové platformy představují cenný nástroj pro výuku algoritmizace a podporu rozvoje algoritmického myšlení. Mohou sloužit jako zdroj materiálů a cvičení k výuce algoritmizace nebo poskytovat prostředí, ve kterém mají žáci možnost vyzkoušet si osvojené dovednosti v praxi při volné tvorbě vlastních projektů (CTPrimEd, 2023a).

První kategorie webových platforem zahrnuje ty, které nabízejí bohaté materiály, tutoriály a interaktivní cvičení zaměřené na různé aspekty algoritmizace. Tyto platformy poskytují studentům možnost seznámit se se základními algoritmickými koncepty, jako jsou cykly, podmínky nebo rekurze, a následně je procvičovat prostřednictvím interaktivních úkolů a příkladů. Tato fáze je klíčová pro budování pevných základů a důkladné pochopení algoritmizace (CTPrimEd, 2023a).

Druhá kategorie pak zahrnuje prostředí, kde mohou studenti vytvářet a testovat své vlastní algoritmy a programy. Typickým příkladem je platforma Scratch, která umožňuje tvorbu vizuálních programů pomocí blokového programování. S využitím takové platformy nejenom pasivně získávají znalosti, ale aktivně je využívají k vytváření a experimentování s algoritmickými koncepty (CTPrimEd, 2023a).

Obě tyto kategorie webových platforem mají své výhody, přičemž je lze ve výuce vhodně kombinovat tak, aby žáci získali dostatečné informace o probírané problematice skrze materiály a cvičení, než se přesunou k vlastní tvorbě algoritmů. Tím lze předejít pocitu frustrace a přehlcení z příliš komplexního prostředí platforem jako je Scratch. Po získání přehledu o algoritmických konstrukcích však umožňují platformy pro praktickou tvorbu algoritmů možnost kreativního projevu a radost z vlastní práce (MIT Media Lab, 2009).

Není výjimkou, že takové platformy využívají principy výše zmíněné gamifikace a zavádějí systémy bodování, úrovní a hodnocení postupu. Využití těchto principů představuje efektivní strategii k angažování žáků a vytváření prostředí, které je atraktivní a podnětné, což může výrazně zvýšit jejich výkony v hodinách i motivaci k učení (Shohieb et al., 2022).

Často využívaným způsobem „gamifikování“ úloh a cvičení je zavádění systému bodů a úrovní, kde studenti získávají body za splnění úkolů a postupují díky nim na vyšší úrovně. Tento systém slouží jako motivace ke zdokonalování a poskytuje žákům přehled o svém pokroku. Stejně je tomu při zavedení dílčích výzev a odměn za dosažení určitých cílů či splnění úkolů. Tímto způsobem se vytváří prostředí, ve kterém se studenti cítí motivováni a při učení se baví (Král, 2018).

2.3 Rozvoj algoritmizace pomocí her

I princip game-based learningu se u webových platforem objevuje v podobě bodů, úrovní, odměn a výzev. Tyto mechanismy jsou implementovány přímo do herního prostředí, ve kterém žáci získávají body za úspěšné řešení úkolů a postupují na vyšší úrovně, což slouží jako motivace k dalšímu pokroku. To se moc neliší od platforem zmíněných výše a jejich využití gamifikace. Platformy využívající game-based learning jdou při zavádění těchto principů ještě dál.

Game-based learning je nejen o získávání bodů a postupování na vyšší úrovně, ale také o zapojení žáků do interaktivních úkolů a misí, které jsou vytvořeny jako součást příběhu. Tyto úkoly jednak poskytují zábavu, jednak také rozvíjí dovednosti algoritmického myšlení. Díky interaktivnímu prostředí a postavám mají žáci možnost prozkoumat různé koncepty algoritmizace v přátelském a inspirativním prostředí virtuálních světů, se kterými mohou interagovat. Tento přístup k výuce nabízí nejen efektivní prostředí pro zlepšení algoritmických dovedností, ale také opět zvyšuje motivaci a zapojení žáků díky hernímu prvkům a příběhům (Videnovik et al., 2023).

3 Platformy pro podporu rozvoje algoritmického myšlení

3.1 Platformy s úlohami

Pro seznámení žáků se základy algoritmizace hrají platformy poskytující ucelené sady úloh klíčovou roli. Tyto úlohy bývají pečlivě navrženy tak, aby postupně učily žáky jednotlivým principům a konceptům algoritmizace. Vedou je k porozumění základních algoritmických myšlenek a umožňují rozvíjet své dovednosti při plnění praktických úloh. (Code.org, 2022)

Sady úloh nejsou pouze o základech algoritmizace, ale také se zaměřují na konkrétní dovednosti a techniky. Mnoho platforem nabízí speciální sady úloh zaměřených na učení

proměnných, funkcí, rekurze a dalších klíčových konceptů. To umožňuje učitelům a žákům flexibilnější přístup k výuce a možnost přizpůsobit materiály konkrétním potřebám třídy či jednotlivých žáků (El Mawas et al., 2022).

Většina těchto platforem využívá prvky gamifikace nebo game-based learningu ke zvýšení angažovanosti a motivace žáků. V rámci game-based learningu jsou úlohy často propojeny s příběhem, ve kterém žáci hrají roli hlavní postavy a pomocí algoritmických konstrukcí řeší výzvy a překážky, které se v příběhu objevují (El Mawas et al., 2022).

Mezi příklady platforem s úlohami patří Robomise, RunMarco, CodeMonkey, Tynker, GalaxyCodr, Ozaria a Blockly Games. Tyto platformy nabízejí různorodé sady úloh s prvky gamifikace a game-based learningu, které jsou vhodné pro různé věkové kategorie žáků a úrovně jejich znalostí algoritmizace.

Jednou z jednodušších, ale přesto v hodinách informatiky dobře využitelných platforem je webová aplikace Blockly Games[1], která obsahuje několik různorodých aktivit pro rozvoj algoritmického myšlení. Každá kategorie aktivit je koncipována jinak. Jsou to například úlohy na průchod bludištěm, pohyb postav, kreslení, programování hudby a další. Jedná se především o ukázku konkrétní implementace knihovny Blockly od společnosti Google, která vývojářům umožňuje tvorbu vlastních aktivit využívajících blokové programování.

GalaxyCodr[2] je ukázkou aplikování game-based learningu v praxi. Aplikace je zasazena do příběhu mimozemšťana putujícího vesmírem z planety na planetu a potýká se s nástrahami, které pro něj vesmír uchystal. Úlohy jsou strukturovány do úrovní (planet). Mezi některé úlohy důležité pro příběh jsou vsazena animovaná videa popisující aktuální dění, která hráčům umožňují vcítit se do příběhu a motivovat je tak při plnění úloh.

Možnosti průchodu úlohami se liší napříč platformami. Některé platformy nabízejí pevně danou strukturu průchodu, zatímco jiné umožňují žákům volnější postup a přizpůsobení úloh jejich individuálním potřebám a preferencím. Důležitým aspektem je také možnost přeskakovat obtížné úlohy nebo volit sady úloh vhodné pro konkrétní věkovou kategorii žáků (Doğan, 2020).

[1] Blockly Games https://blockly.games.
[2] GalaxyCodr dostupný z https://galaxycodr.com.

Jmenovitě lze říci, že ze zmapovaných aplikací mají GalaxyCodr a RunMarco[3] pevně daný průchod úlohami, mezi kterými se lze vracet, ale není hráčům umožněno úlohy přeskakovat. To může souviset s přítomností příběhu, který se snaží představovat hráčům úrovně v takovém pořadí, v jakém to dle příběhu dává největší smysl. Ozaria[4], která učí algoritmizaci za pomoci Javascriptu, též nabízí bohatý příběh, který se snaží hráče upoutat a motivovat jej, ale zároveň mu nechává možnost úlohy přeskakovat a vracet se k nim (El Mawas et al., 2022; Edwards, 2023).

Na platformě Robomise[5] se hráč chopí role pilota rakety, která letí vesmírem, musí se vyhýbat asteroidům a sbírat bonusové body. I když se nejedná o tak propracovaný příběh jako například ve výše zmíněném Galaxycodru, nabízí Robomise širokou škálu úloh, jejichž obtížnost pozvolna roste od těch nejsnazších až po velmi obtížné úlohy. Hráč má možnost úlohy plnit v libovolném pořadí. To poskytuje učiteli možnost stanovit žákům, které z úloh mají za úkol splnit.

Aplikace s úlohami mohou učitelům nabídnout určitou kontrolu nad obsahem, což může zahrnovat nastavování obtížnosti úloh podle individuálních dovedností a potřeb žáků, personalizaci obsahu v souladu s jejich učebními cíli a preferencemi, a také možnost sledovat postup žáků a reagovat na jejich pokrok.

CodeMonkey[6], Tynker[7], Ozaria a GalaxyCodr umožňují žáky přidávat do tříd, které jsou spravovány učitelem, a mít tak přehled o jejich postupu. Ozaria též dává učiteli možnost přiřazovat třídě konkrétní sadu úloh (level), a reagovat tím tak na potřeby svých žáků. Co se týče personalizace samotných úloh, tou z výše zmíněných platforem disponuje pouze Robomise, která učitelům poskytuje nástroj pro přípravu vlastních úloh na míru svým žákům (Akinduyo, 2023; Edwards, 2021; Edwards, 2023).

[3] RunMarco dostupný z https://runmarco.allcancode.com.
[4] Ozaria dostupná z https://www.ozaria.com.
[5] Robomise dostupná z https://robomise.cz.
[6] CodeMonkey dostupný z https://www.codemonkey.com.
[7] Tynker dostupný z https://www.tynker.com.

3.2 Platformy pro možnost volné tvorby algoritmů

Volná tvorba projektů je důležitým prvkem v procesu výuky algoritmizace, který umožňuje žákům nejen se seznámit se s jejími základy, ale také využít své kreativity a nápadů při tvorbě vlastních projektů. Při procvičování algoritmizace touto formou mají žáci často na výběr z široké palety algoritmických konstrukcí, které jim umožňují vyjádřit své myšlenky a nápady kreativními způsoby. Tato komplexnost poskytuje žákům prostor vytvářet rozsáhlejší projekty, které využívají různorodé prvky algoritmizace a programování. Může jít o animace, interaktivní příběhy či hry.

Nicméně široká paleta nástrojů algoritmizace, se kterými žák ještě nebyl zcela seznámen, přináší riziko frustrace, zejména u žáků, jež si doposud dostatečně neosvojili základy algoritmizace. Je tedy vhodné taková prostředí volit jako další krok výuky algoritmizace po tom, co si žáci osvojí základy například plněním úloh skrze výše zmíněné platformy (21K School, 2021).

Mezi zmapované platformy umožňující volnou tvorbu patří Scratch, PencilCode, Code.org, Scoolcode, MakeCode, Tinkercad, Roboblockly, Karel. Každá z nich je jinak koncipována, poskytuje jiné nástroje a zaměřuje se na jiný způsob tvorby projektů s pomocí algoritmických konstrukcí.

Nejpopulárnější mezi těmito platformami je Scratch[8]. Tato platforma poskytuje uživatelům intuitivní a interaktivní prostředí, ve kterém mohou vytvářet vlastní animace, příběhy či hry pomocí blokového programování. Scratch také obsahuje jednoduchý grafický editor, což žákům umožňuje zapojit do tvorby projektů i svou výtvarnou kreativitu. Platforma je využívána od prvního stupně základní školy až po střední školy. Množství nástrojů, které jsou v rámci Scratche k dispozici, je možné obohatit o přídavné moduly, jako je nástroj pro programování vykreslování grafiky či programování robotů LEGO Mindstorms (BasuMallick, 2024).

Alternativou ke Scratchi je platforma Scoolcode společnosti Scoolcode[9] založena na podobných principech s rozšiřujícími možnostmi v podobě textového programování

[8] Scratch dostupný z https://scratch.mit.edu.
[9] Logiscool dostupný z https://www.logiscool.com.

a několika úrovní grafického jazyka (symboly na blocích či sady bloků pro vyučovanou věkovou kategorii žáků). Scoolcode také obsahuje prvky gamifikace, které zvyšují motivaci žáků při výuce. Jedná se především o získávání bodů za plnění kvízů a odemykání nových designových prvků, jako jsou avataři postav, jejich rámečky a další (O Logiscool, 2024).

Platforma PencilCode[10] je zacílena na vykreslování grafiky na základě programování blokovým či textovým jazykem, které mezi sebou lze snadno přepínat, a vidět tak podobnost v jejich syntaxi. Vybrat si lze mezi Javascriptem a CoffeScriptem, přičemž text na blocích poté odpovídá syntaxi vybraného jazyka (Deng et al., 2020).

Roboblockly[11] je platforma zaměřená na výuku robotiky, umožňující uživatelům programovat virtuálního robota pomocí blokového programování. Tato platforma nabízí rozsáhlé možnosti a funkce, vhodné i pro výuku robotiky na vysokých školách. Blokový jazyk, který je zde k dispozici, je ale možné zjednodušit až do podoby symbolů (například šipek), a lze proto platformu využít i na prvním stupni základní školy. Aplikace lze spárovat i se skutečným robotem, což přispívá k uvědomění si provázanosti algoritmizace se skutečným světem.

Platforma Microsoft MakeCode[12] poskytuje několik oficiálních variant webového prostředí MakeCode, zaměřených na různé oblasti rozvoje algoritmického myšlení. MakeCode lze používat v kombinaci s LEGO Mindstorms, programovatelnými deskami Micro:Bit, nebo při tvorbě arkádových her v prostředí MakeCode Arcade.

Ze starších prostředí stojí za zmínku prostředí Karel[13], jehož verze, analyzovaná touto prací, byla na internetu zveřejněna poprvé v roce 2002, a tím se stala jedním z prvních nástrojů pro výuku programování u nás. V tomto postředí uživatel programuje robota Karla pomocí textového pseudokódu tak, aby se pohyboval po mapě a sbíral či přemisťoval značky. Toto prostředí je zajímavé především tím, že mapu, po které se robot pohybuje, lze umisťováním zdí a značek přizpůsobit. Tím se otevírá možnost pro učitele, jak připravit svým žákům mapu

[10] Code.org dosupný z https://code.org.
[11] Roboblockly dostupné z https://roboblocky.com.
[12] MakeCode dostupný z https://www.microsoft.com.
[13] Karel dostupný z http://karel.oldium.net.

na míru tak, aby procvičovala konkrétní dovednosti spojené s algoritmickým myšlením (Kdo je Karel, 2006).

Většina z výše zmíněných platforem stojí na velmi podobném principu, ale tím, že se zaměřují na různé oblasti využití algoritmizace, představují dohromady velmi pestrou paletu nástrojů, které lze při výuce algoritmizace využívat a s jejichž pomocí mohou žáci kromě rozvíjení svých dovedností algoritmického myšlení vyjádřit také svou kreativitu.

Některé platformy nabízejí kromě samotného prostředí pro tvorbu i nástroje využitelné učiteli během výuky. Scratch a Code.org poskytují funkci třídy, do které lze přidávat žáky. Žáci i učitelé mohou v rámci třídy sdílet své projekty a navzájem si poskytovat zpětnou vazbu. U ostatních platforem je možné předpřipravit pro žáky projekt, který bude sloužit jako odrazový můstek pro jejich další tvorbu.

3.3 Sdružující platformy

Kromě samotných platforem pro výuku algoritmizace existují také platformy, které slouží jako místa pro shromažďování edukačního obsahu z různých zdrojů. Tyto platformy se snaží poskytnout učitelům přehledné a snadno dostupné materiály pro podporu výuky algoritmizace. Jedním z hlavních cílů těchto platforem je umožnit uživatelům sdílet své připravené materiály a inspiraci s ostatními uživateli, což přispívá k bohatšímu a pestřejšímu vzdělávacímu prostředí.

Největší a nejrozšířenější platformou tohoto typu je Hour of Code[14], která představuje centrální bod pro výuku algoritmizace a programování. Tato platforma shromažďuje obsah z mnoha jiných zdrojů a umožňuje uživatelům snadný přístup k různorodým materiálům a aktivitám. S touto platformou spolupracují i ostatní platformy pro výuku algoritmizace, které disponují obsahem, který se Hour fo Code snaží shromažďovat. Na stránce Hour of Code je proto možné nalézt materiály z Code.ord, Tynkeru, CodeMonkey a mnohých dalších. Aktivity zde zveřejněné obvykle obsahují doplňující informace pro učitele, jako

[14] Hour of Code dostupný z https://hourofcode.com.

například jakému věkovému rozmezí žáků je aktivita určena, na rozvoj jakých dovedností aktivita cílí a jak lze aktivitu využít při výuce (Code.org, 2022; Hour of Code, 2024).

Další platformou sdružující edukativní materiály na jedno místo je FERTILE Community Platform[15], která se zaměřuje na sdílení připravených materiálů mezi učiteli. Projekt FERTILE cílí na rozvoj informatického myšlení při výuce robotiky ve spojení s jinými na školách běžně vyučovanými předměty, a to především s výtvarnou výchovou. Komunitní platforma tohoto projektu poskytuje prostředí pro přehlednou přípravu výukových materiálů, sdílení těchto materiálů s ostatními uživateli platformy a čerpání inspirace z materiálů ostatních uživatelů (The "FERTILE" project, 2024).

Lze zde zmínit i výše uvedený Scratch, který sice neumožňuje sdružovat materiály z jiných platforem, ale poskytuje jednoduché prostředí pro sdílení projektů s ostatními uživateli Scratche, včetně informací o projektu a jak jej lze využívat. Což opět dává možnost nejen učitelům poskytovat své projekty k inspiraci ostatních uživatelů.

Také platformy tohoto druhu poskytují nástroje, které učitelům mohou usnadnit vedení výuky. Jedná se především o možnost tvorby a správy tříd pro usnadnění poskytnutí materiálů svým žáků. Žáci mají v rámci třídy k dispozici materiály poskytnuté učitelem, který může, stejně jako například u platforem s aktivitami pozorovat postup svých žáků, má přístup k jejich práci, a může tak poskytovat okamžitou zpětnou vazbu.

3.4 Aplikace s možností správy obsahu

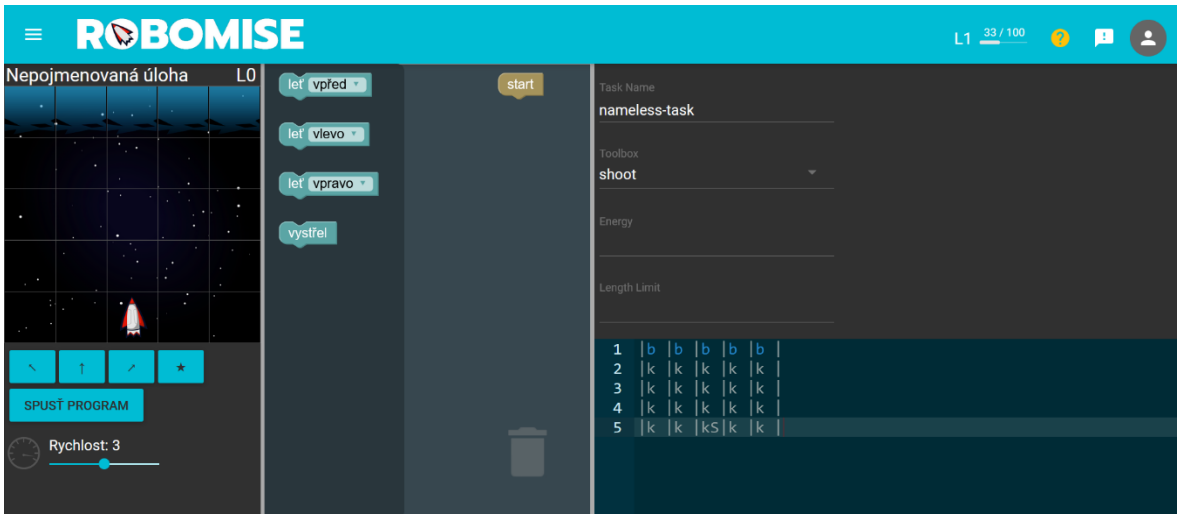
Mezi výše zmíněnými platformami si lze všimnout jednoho společného rysu. Mezi všemi třemi kategoriemi se nacházejí platformy, které učitelům poskytují správu třídy. Jedná se o šikovný nástroj, typicky umožňující přidávat a odebírat žáky z virtuální třídy, poskytovat jim potřebné materiály k výuce, mít přehled o jejich práci, a mít tak možnost lépe reagovat na individuální potřeby žáků a diferencovat svou výuku podle jejich schopností a tempa učení.

Tento nástroj v některých případech umožňuje pracovat s obsahem platformy a přizpůsobovat například pořadí, ve kterém bude žákům obsah poskytnut. Personalizace obsahuje je pro diferenciaci výuky podstatným faktorem. Platformy pro volnou tvorbu

[15] Projekt FERTILE dostupný z https://fertile.gsic.uva.es.

a sdílení materiálů jsou v tomto ohledu dostatečně pružné, aby učitelé dokázali obsah výuky vždy připravit tak, aby vyhověl potřebám svých žáků. U platforem, jejichž náplní jsou úlohy procvičující algoritmické myšlení, je míra jejich přizpůsobení omezena podobou úloh, které obsahuje a možnost přizpůsobení jednotlivých úloh či tvorby zcela vlastních, je napříč platformami dosti omezená.

Mezi všemi výše zmíněnými platformami jsou to pouze Robomise (Obrázek 2) a Karel, které umožňují učitelům upravit si úlohy dle vlastních představ a potřeb. Nicméně i ty mají tuto možnost značně omezenou. Robomise umožňuje úpravu mapy pomocí změny textového soubory a úpravy sady bloků, které mají uživatelé při řešení úloh k dispozici. Karel, jak bylo zmíněno výše, umožňuje přípravu hrací plochy, skrze kterou se bude robot Karel pohybovat, i sadu příkazů, kterou mohou žáci používat. Ovšem uživatelské rozhraní Karla může být podle dnešních měřítek považováno na zastaralé a méně uživatelsky příjemné. V obou případech lze úlohy předávat pouze mimo platformu pomocí textu, který je z aplikace exportován a následně žáky importován v jejich prohlížečích.



Obrázek 2 - Prostředí editoru vlastních úloh v aplikaci Robomise

Absence této možnosti představuje chybějící článek v nabídce dostupných aplikací. Učitelé by měli mít více prostoru a možností přizpůsobit výuku specifickým potřebám svých žáků.

Všechny zmíněné aplikace nabízí určitou formu procvičování algoritmických konstrukcí, avšak možnost tvorby či přizpůsobení úloh dle konkrétních potřeb je omezená. Platformy pro volnou tvorbu projektů, jako je například Scratch, učitelům sice umožňují připravit

výchozí stav projektů podle vlastních potřeb a nechat jej žáky dotvořit, ovšem neumožňují například přizpůsobit sadu bloků, které budou při řešení k dispozici.

Praktickým cílem práce je právě tuto mezeru vyplnit a poskytnout učitelům nástroj, který umožní tvorbu vlastních úloh přizpůsobených specifickým potřebám jejich žáků, a rozšířit tak možnost personalizace výuky algoritmizace o prostředí ve kterém mohou učitelé plně využít své kreativity a pedagogického umu k maximálnímu zapojení a rozvoji svých žáků.

4 Návrh webové aplikace pro výuku algoritmizace

Při analýze dostupných platforem pro výuku algoritmizace a programování bylo zjištěno, že dostupné aplikace nabízejí různé přístupy ke vzdělávání a mnoho nástrojů k podpoře jejich výuky. Provedená analýza těchto platforem odhalila oblast, pro kterou neexistuje dostatek nástrojů. Jedná se o možnost přípravy vlastních úloh pro rozvoj algoritmizace. S ohledem na tento fakt byla navržena aplikace, jejíž hlavní funkcí je editor pro přípravu vlastních úloh. Aplikace byla označena pracovním názvem CodeQuest. Při výběru doménového jména bylo nutné aplikaci přejmenovat. Její konečný název zní Codeblockie a nyní je dostupná na adrese https://codeblockie.com.

Případy užití dílčích částí aplikace jsou zdokumentovány pomocí Use Case diagramů viz Příloha 1[16].

4.1 Požadované funkce aplikace

Středobodem celé aplikace je editor úloh, umožňující uživatelům vytvářet a upravovat úlohy podle svých vlastních představ. Tento nástroj je navržen tak, aby poskytoval uživatelsky přívětivé rozhraní, pro snadnou a intuitivní tvorbu úloh ve stejném prostředí blokového programování, ve kterém jsou úlohy řešeny jinými uživateli.

Jeho hlavní funkce zahrnují možnost definovat objekty, se kterými bude úloha pracovat, jejich rozmístění na herním poli a vzhled, jenž budou mít. Pomocí blokového kódu lze vybraným objektům definovat jejich chování, které zahrnuje například způsob, jakým budou reagovat na události vyvolané ostatními objekty. Takovýmto definováním chování lze například stanovit podmínky pro úspěšné splnění či nesplnění úkolu. Tvůrce úlohy musí

[16] Veškeré UML diagramy byly vytvořeny pomocí nástroje Diagrams.net, dostupném na https://app.diagrams.net.

definovat, které z objektů úlohy budou upravitelné hráčem, a tudíž které bude smět hráč programovat a použít je k řešení úlohy. Bloky, které budou při řešení úlohy hráči k dispozici, bude volit tvůrce úlohy v jejím nastavení. Omezením výběru bloků pouze na ty, které jsou při řešení úlohy nezbytné, nedochází k přehlcování hráče informacemi, které by jej pouze mátly a sváděli při řešení nesprávným směrem.

Během přípravy tvorby má uživatel možnost úlohu spustit a vyzkoušet si její funkčnost z pohledu hráče.

O spuštění úlohy se stará část *Spouštěč úloh*. Tato část odpovídá za zobrazování úlohy vytvořené za pomoci *Editoru úloh*. Hráč může přepínat mezi upravitelnými postavami a měnit jejich kód pomocí dostupných bloků kódu. Pokud je hráč se svým řešením spokojený a chce vyzkoušet jeho správnost, má možnost úlohu spustit. Dojde k jejímu vyhodnocení a spustí se animovaná vizualizace, která jednak potvrzuje správné řešení úlohy, ale při chybném řešení pomáhá odhalit chybu. Zpomalením přehrávání vizualizace je možné lépe porozumět fungování aktuálního řešení a odhalit, kde se chyba nachází.

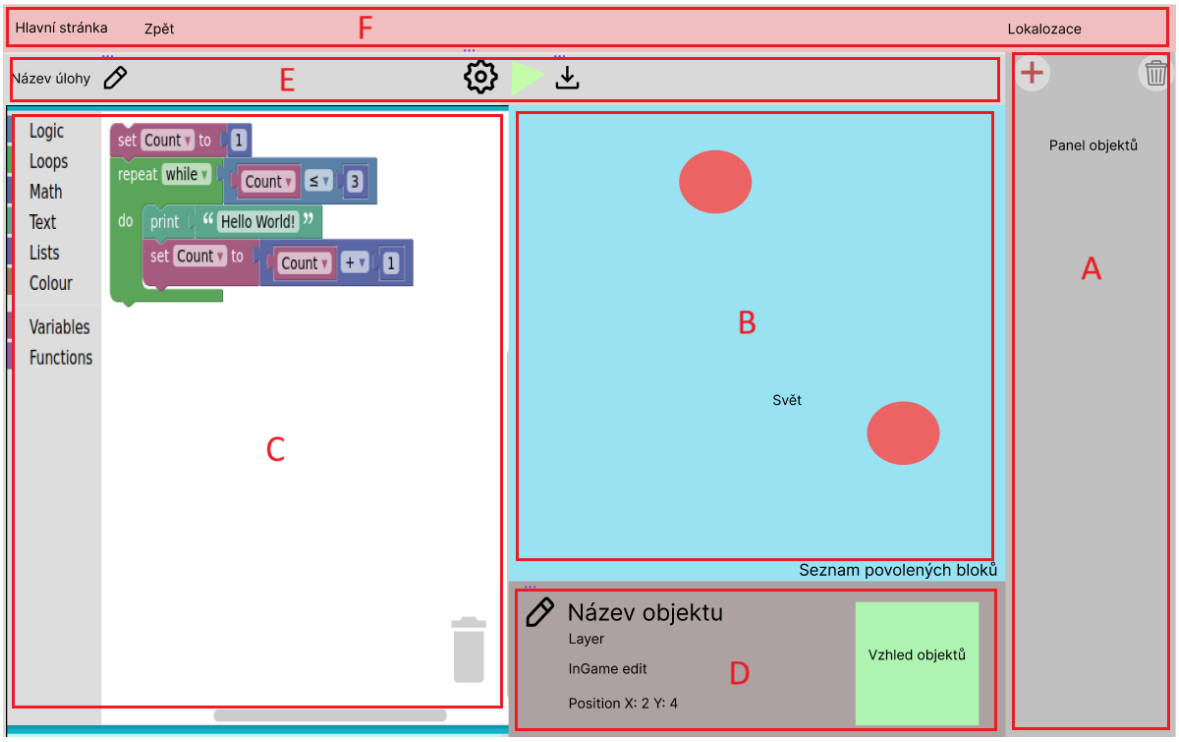
Spojení mezi *Editorem* a *Spouštěčem úloh* představuje *Spouštěč úloh*, kde mají všichni uživatelé možnost procházet veřejně dostupné úlohy, které byly vytvořeny ostatními uživateli. Pro plné využití funkcí *Panelu úloh* je možné se přihlásit či zaregistrovat. Přihlášení uživatelé získávají další možnosti, jako je tvorba vlastních úloh, editace existujících úloh a možnost duplikování úloh, včetně těch vytvořených ostatními uživateli. To umožňuje inspirovat se úlohami ostatních a dále je rozšiřovat či vylepšovat. Vlastní úlohy je možné zveřejnit pro ostatní uživatele, a to jak žáky, kteří se mohou pokusit o její vyřešení, tak pro učitele jako inspiraci pro jejich vlastní úlohy.

4.2 Grafická podoba

Před implementací aplikace bylo třeba vytvořit si představu o rozložení nástrojů jednotlivých částí aplikace. Hlavním záměrem bylo zvolit minimalistický, přehledný a uživatelsky přívětivý design, který umožní uživatelům snadnou orientaci a rychlý přístup ke všem potřebným nástrojům a funkcím. Na Obrázek 3 je vyobrazen prvotní návrh

uživatelského rozhraní *Editoru úloh*. Jednotlivé části jsou odděleny a označeny písmeny pro snazší identifikaci. Pro tvorbu grafických návrhů byl využit nástroj Figma[17].

Tento koncept byl vytvořen pouze za účelem návrhu optimálního rozložení nástrojů editoru pro zajištění co nejlepší uživatelské zkušenosti. Obrázek 3 slouží pouze jako vizuální koncept a nepředstavuje konečnou podobu editoru úloh v reálném prostředí. Barvy, fonty a další grafické prvky na obrázku se liší od výsledného prototypu aplikace.



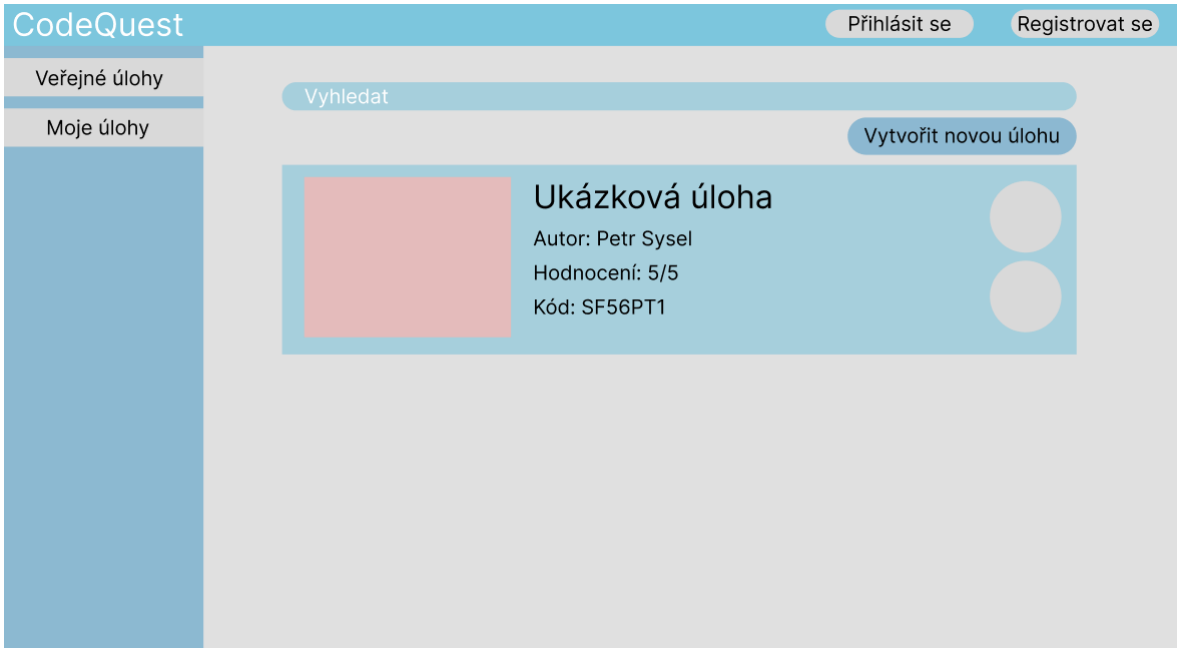
Obrázek 3 - návrh rozložení nástrojů editoru úloh s vyznačenými dílčími částmi

Během realizace projektu došlo k mírným úpravám rozložení *Editoru úloh* vyobrazeného na Obrázek 3. Tyto úpravy byly provedeny na základě zpětné vazby testovacích uživatelů. Hlavní úpravy obsahovaly přesunutí panelu s nastavením objektu (část D) do pravého panelu pod seznam objektů v úloze (část A) a přesunutí ovládacího panelu úlohy (část E) na původní místo panelu s nastavením objektu (část D).

Grafický návrh *Spouštěče úloh* vycházel z návrhu *Editoru úloh*, jelikož spolu sdílejí mnoho společných prvků. U *Spouštěče* ubyl panel objektů (část A) a na místo panelu s nastavením objektu (část D) byl umístěn panel pro výběr momentálně editovaného objektu.

[17] Nástroj Figma je dostupný z https://www.figma.com.

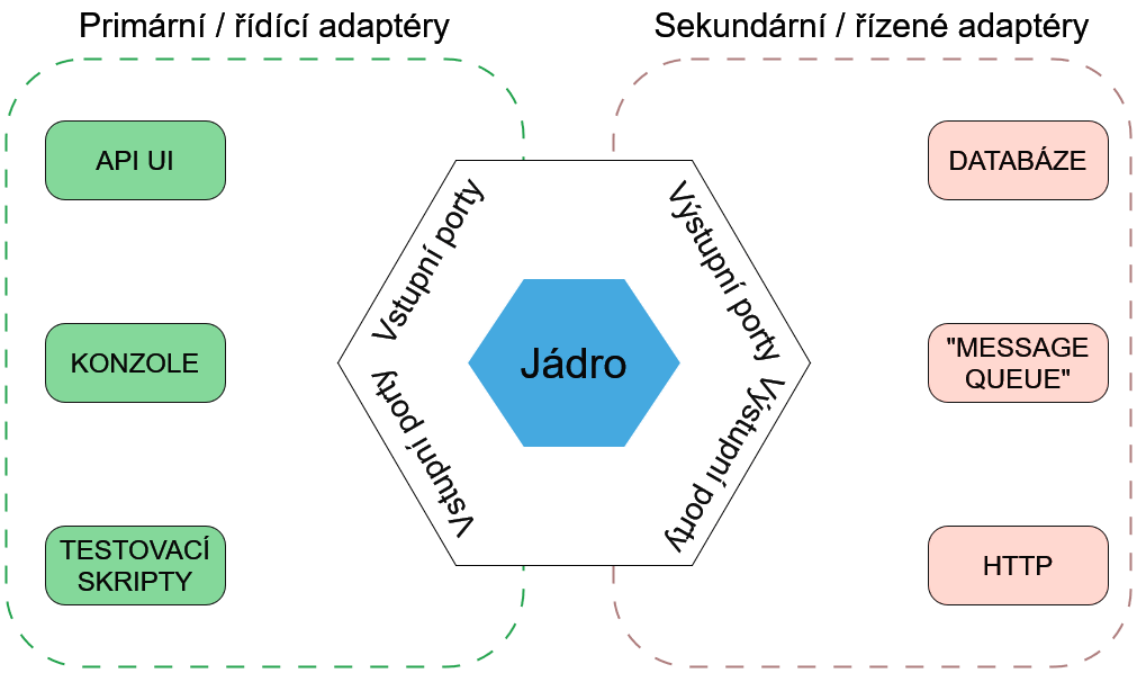
Návrh *Panelu úloh* (Obrázek 4) byl narozdíl od návrhu *Editoru* a *Spouštěče úloh*, které byly vytvářeny před zahájením vývoje prototypu, byl návrh Panelu úloh koncipován již s ucelenou představou o celkovém vzhledu a uživatelském rozhraní aplikace. Proto se návrh *Panelu úloh* finálnímu řešení blíží nejen rozložením prvků, ale i barevným schématem a celkovým dojmem.



Obrázek 4 - Návrh uživatelského rozhraní pro panel úloh

4.3 Architektura aplikace

Jako architektura aplikace byla zvolena Hexagonální architektura, známá také jako „Ports and Adapters“ nebo „Clean Architecture“, a to z důvodu své schopnosti poskytnout robustní a flexibilní základ pro vývoj aplikace. Tato architektura klade důraz na oddělení jednotlivých vrstev aplikace a zajišťuje snadnou rozšiřitelnost, testovatelnost a udržitelnost (Martinez, 2021). Schématicky je tato architektura vyobrazena na Obrázek 5.



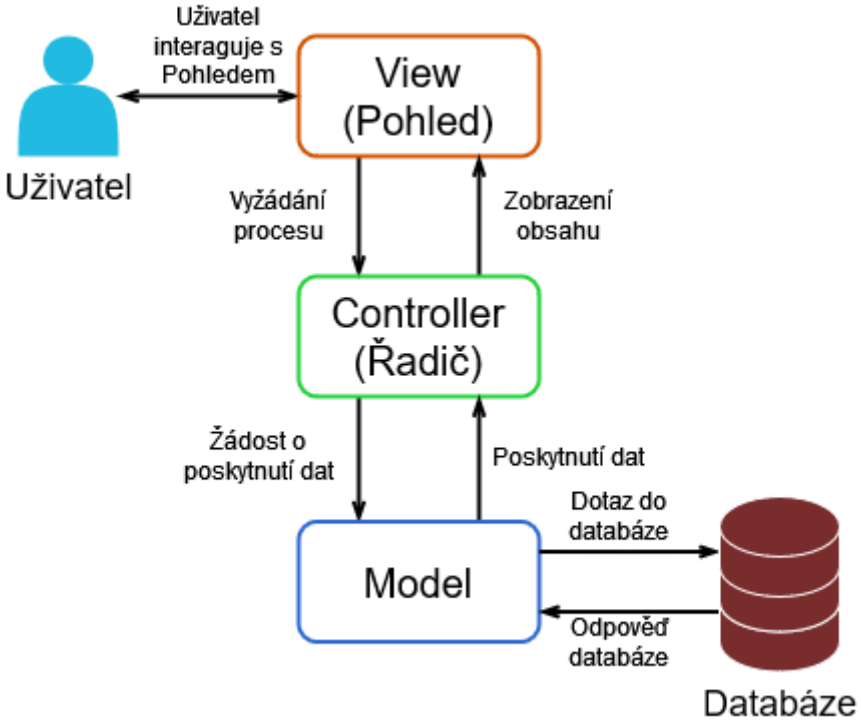
Obrázek 5 - Schéma hexagonální architektury (Dogantekin, 2020) - přeloženo autorem

Hexagonální architektura se skládá ze tří hlavních vrstev (Martinez, 2021):

1. **Doménová vrstva (Core)**: Jádro aplikace, kde se nachází veškerá doménová logika a business rules. Tato vrstva je nezávislá na konkrétní implementaci adaptérů a infrastruktury a zaměřuje se na modelování doménových objektů a jejich interakce.
2. **Porty (Ports)**: Definují rozhraní mezi doménovou vrstvou a adaptéry. Porty představují rozhraní, která umožňují doménové vrstvě komunikovat s vnějším světem nebo s infrastrukturou aplikace. Jsou to abstraktní rozhraní, která jsou implementována v adaptérech.
3. **Adaptéry (Adapters)**: Zajišťují propojení mezi doménovou vrstvou a vnějším světem nebo infrastrukturou aplikace. Tyto adaptéry implementují rozhraní definovaná v portech a provádějí konkrétní operace, jako je přístup k databázi, komunikace s externími službami nebo prezentace dat uživateli.

Při výběru architektury byla do úvah zahrnuta i architektura MVC (Model View Controller), která je u vývoje webových aplikací běžnější (Sadika, 2023).

Architektura MVC (Model-View-Controller) je rozdělena do tří hlavních částí: Model, View a Controller (schéma architektury je uvedeno na Obrázek 6). Model obsahuje data a logiku aplikace, View je zodpovědný za zobrazení uživatelského rozhraní a Controller zprostředkovává komunikaci mezi modelem a view. Když uživatel provede nějakou akci, jako je například kliknutí na tlačítko, controller přijme tuto událost, upraví stav modelu a poté informuje view, aby se aktualizovalo podle nových dat. Tato architektura je široce používaná při vývoji webových aplikací a umožňuje oddělení prezentace dat od logiky, což zlepšuje modularitu a udržitelnost kódu (Sadika, 2023).



Obrázek 6 - Schéma architektury MVC (Sadika, 2023) - přeloženo autorem

I když se jedná o odlišný přístup k vývoji softwaru, sdílejí obě architektury několik společných rysů (Fowler, 2003):

- **Rozdělení zodpovědnosti**: Obě architektury kladou důraz na oddělení zodpovědností jednotlivých částí systému. To zahrnuje oddělení doménové logiky, prezentace dat a manipulace s daty.

- **Modularita**: Obě architektury podporují modularitu a zapouzdření funkcionalit do samostatných komponent, což zvyšuje udržitelnost a znovupoužitelnost kódu.

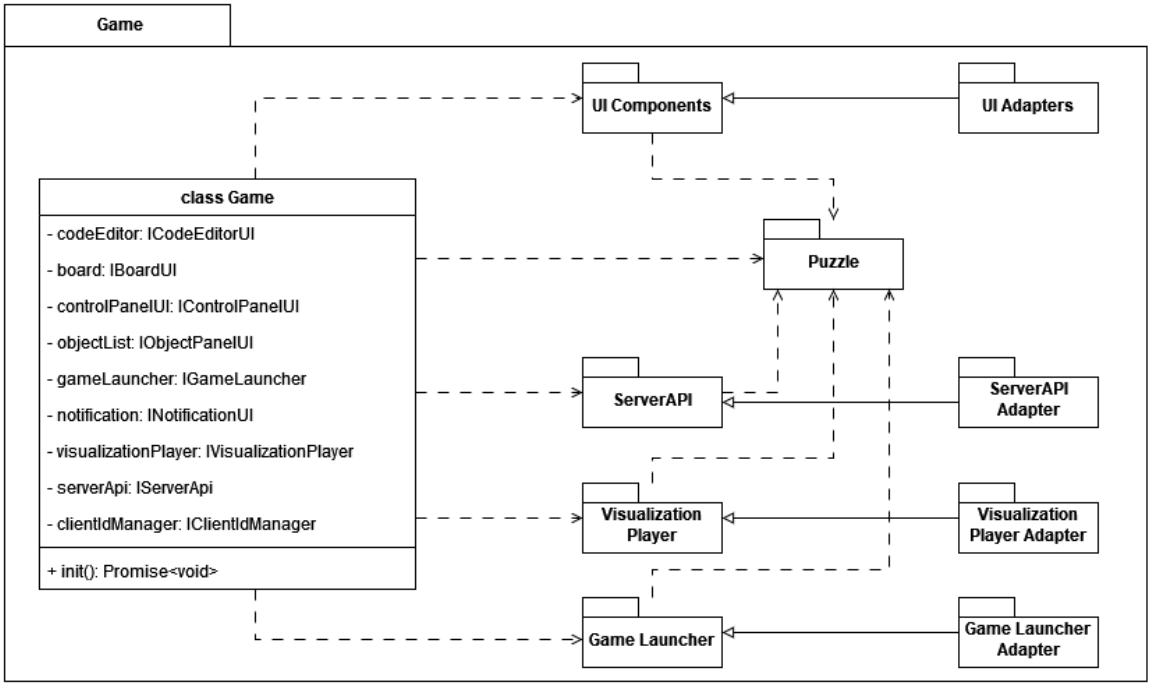
Rozdílnosti architektur by se daly shrnout do těchto bodů (Fowler, 2003):

- **Zaměření**: MVC je architektura primárně určená pro vývoj uživatelských rozhraní a webových aplikací, zatímco hexagonální architektura je obecnější a zaměřuje se na organizaci doménové logiky a její interakce s okolním světem.
- **Vrstvy a závislosti**: V architektuře MVC jsou komponenty rozděleny do tří vrstev: Model, View a Controller. Controller zpracovává uživatelské vstupy a komunikuje s modelem a pohledem. Na druhou stranu, hexagonální architektura organizuje komponenty kolem jádra aplikace (doménové vrstvy) a vnějších adaptérů. Jádro aplikace obsahuje doménovou logiku, zatímco adaptéry zajišťují komunikaci s vnějším světem.
- **Závislosti**: V architektuře MVC mohou být závislosti mezi jednotlivými vrstvami pevně zakódovány, což může vést k vyššímu propojení a obtížnějšímu testování. Naopak, hexagonální architektura klade důraz na volné vazby a rozhraní, což zvyšuje flexibilitu a snazší testovatelnost aplikace.

Hexagonální architektura byla pro tento projekt zvolena díky své obecnější povaze a modularitě, která umožňuje sdílení velkých částí kódů napříč aplikací.

S ohledem na zvolenou architekturu byl navržen kompletní Class Diagram, popisující strukturu celé aplikace. Hlavní části tohoto diagramu jsou přiloženy v podobě kombinace Class a Package diagramu v Příloha 2.

Všechny části aplikace jsou strukturovány velice podobně. Příkladem, na kterém lze strukturu aplikace ilustrovat je diagram *Spouštěče úloh* na Obrázek 7.



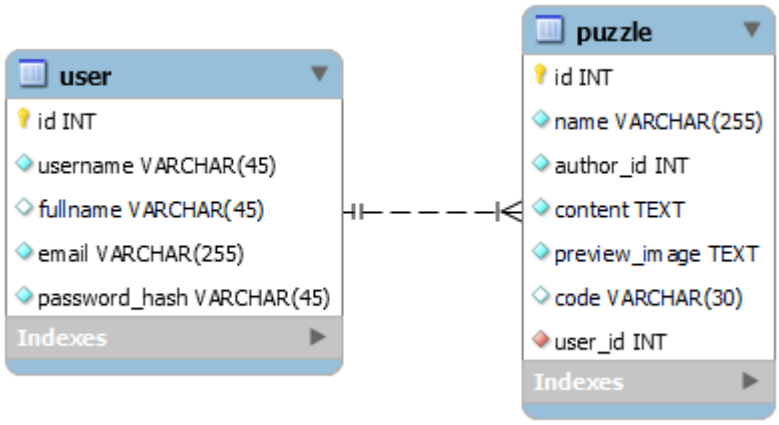
Obrázek 7 - Class a Package diagram části *Spouštěč úloh*

V kontextu hexagonální architektury představuje jádro této části třída Game. Jádro využívá řadu portů, které mu poskytují potřebné nástroje k jeho fungování. Například ke komunikaci s uživatelem jsou využity porty modulu UI Components. Tyto porty jsou následně implementovány v podobě adaptérů UI Adapters. Adaptéry je možné jednoduše využívat opakovaně v různých částech aplikace, což poukazuje na výhodu hexagonální architektury – modularitu.

Hlavní komponentou nejen *Spouštěče úloh*, ale celé aplikace je třída Puzzle, která slouží k reprezentaci úloh, k manipulaci s jejími daty, serializaci za účelem ukládání a dalších operací. Při práci v *Editoru* jsou veškeré úpravy provedené tvůrcem úlohy ukládány do této datové struktury, kterou *Spouštěč* využívá k načítání úlohy.

V rámci jednotlivých částí aplikace dochází ke komunikaci mezi jádry a jejich porty na základě zasílání událostí spolu s potřebnými daty, které si jádro vyžádá. Mezi sebou části komunikují prostřednictví serveru. Ke komunikaci se serverem je použit další port a jeho adaptér, který využívají všechny tří části frontendu aplikace.

Data, se kterými *Serverová část* pracuje, jako například informace o uživatelích a úlohách, jsou ukládána do databáze, jejíž schéma prezentuje entitně-relační diagram na Obrázek 8[18].



Obrázek 8 - Relačně entitní diagram databáze serveru

4.4 Možnosti budoucího rozšíření

Navrhovaná aplikace má potenciál pro další rozšíření a zdokonalení. Je navržena s ohledem na flexibilitu a škálovatelnost, což umožňuje snadné přidávání nových funkcí a vylepšení stávajících řešení. Tato flexibilita je zásadní pro adaptaci aplikace na různé potřeby uživatelů.

Vzhledem k rozsahu tohoto projektu není možné implementovat všechny potenciální funkce najednou. To znamená, že si práce ponechává prostor pro další rozšíření a vylepšení, která budou určována potřebami uživatelů. Některé z nich již lze jmenovat.

Jedno z možných rozšíření aplikace je implementace základní sady úloh. Tato funkcionalita by poskytovala uživatelům předdefinovanou kolekci úloh, pokrývající základní principy řešení úloh a sloužila by jako ucelený výukový materiál, který by uživatelům poskytoval přehledný úvod do algoritmizace.

Následným krokem při rozšiřování aplikace by mohlo být umožnění uživatelům vytvářet vlastní sady úloh. Tím by se především učitelům poskytl další nástroj pro personalizaci výukového obsahu s možností přizpůsobit ji potřebám svých žáků. Uživatelé by mohli

[18] Diagram byl vytvořen pomocí nástroje MySQL Workbench, dostupného z https://www.mysql.com/products/workbench/.

vytvářet tematické sady úloh, které by pokrývaly oblasti studia podle vlastních představ. Tím by aplikace poskytovala flexibilní a individuální přístup k přípravě vzdělávacích materiálů.

Aby byla zajištěna pohodlná manipulace s materiály, jejich poskytování žákům a učitel měl přehled, jak žáci materiály plní, bylo by možné aplikaci rozšířit o funkcionalitu pro tvorbu a správu tříd. Tato funkce by podobně jako u mnohých dalších platforem, včetně těch uvedených v kapitole 3, umožňovala uživatelům vytvářet virtuální třídy a přidávat do nich své žáky nebo kolegy. Každá třída by měla svůj vlastní prostor pro sdílení úlohy či celých sad. Správci tříd (učitelé) by měli možnost přidávat a odebírat členy třídy, přidávat úkoly a sledovat pokrok žáků. Tím by aplikace poskytovala učitelům a lektorům efektivní nástroj pro organizaci výuky a vedení svých žáků.

Je logické, že při poskytnutí možnosti tvorby úloh kterémukoliv uživateli aplikace se bude stávat, že některé úlohy budou kvalitněji provedené než jiné. K určení kvality úloh před vyzkoušením by mohl sloužit mechanismus hodnocení úloh ostatními uživateli. Tím by jednak docházelo k filtrování méně kvalitních úloh, a jednak k poskytování užitečné zpětné vazby tvůrcům úloh.

Kvůli možnosti využívání aplikace zahraničními uživateli by bylo vhodné provést lokalizaci do anglického jazyka, což by posílilo globální dosah aplikace a její schopnost oslovit širší uživatelskou základnu. V případě zájmu o aplikaci by bylo možné lokalizaci rozšiřovat o další jazyky.

5 Implementace prototypu navrhované aplikace

Pro implementaci navržené aplikace byl zvolen jazyk Typescript, a to jak pro frontendovou, tak serverovou část aplikace. Jazyk byl vybrán především díky své míře bezpečnosti kódu, kterou přináší statická kontrola typů, díky ní je možné některé chyby odladit již ve vývojovém prostředí. Vedlejším benefitem statických typů je vyšší čitelnost a srozumitelnost kódu (Shubel, 2022).

Díky svému překladači do Javascriptu je Typescript použitelný ve všech moderních prohlížečích i technologii Node.js, což poskytlo možnost vývoje celé aplikace v jediném jazyce (Shubel, 2022).

Během vývoje aplikace byly využity technologie, které usnadňují práci vývojáře. Na straně frontendu byl použit nástroj Vite, který se stará o automatické načtení webové stránky, bez nutnosti načítat stránku ručně po každé změně. Vite se také stará o sestavování aplikace před nasazením na server. Tento proces obsahuje automatické připojení potřebných knihoven k výsledné aplikaci a optimalizační procesy, které snižují velikost výsledného kódu (Early, 2022). Na straně serveru byl k monitorování změn a automatické restartování využit nástroj Nodemon, který navíc podporuje nativní spouštění Typescriptových aplikací bez potřeby přeložení kódu před spuštěním, což je další vlastnost, která urychluje vývoj a činí jej pohodlnějším pro vývojáře (How To Restart Your Node.js Apps Automatically with nodemon, 2024).

Veškerý zdrojový kód byl verzován pomocí technologie Git. Gitovský repozitář byl průběžně sdílen do online repozitáře GitHub[19], což umožnilo zálohování pracovních verzí aplikace (An introduction to Git: what it is, and how to use it, 2018; What Is GitHub? A Beginner's Introduction to GitHub, 2023).

5.1 Implementace návrhu

5.1.1 Klientská část aplikace

Při vývoji frontendové části bylo učiněno rozhodnutí nepoužívat robustní knihovny a frameworky, jakými jsou React, Bootstrap či Angular (Dhaduk, 2022). Místo toho byly využity nástroje HTML5 v kombinaci s menšími knihovnami. Toto rozhodnutí bylo motivováno snahou minimalizovat složitost a velikost kódu, což přispívá k rychlejšímu načítání a provozu aplikace. Do rozhodnutí se promítla i motivace autora projektu k prohloubení znalostí a zkušeností návrhu a vývoje software za pomoci základních nástrojů HTML5, které poskytují dostatečnou základnu pro vývoj robustního systému (Ashtari, 2022).

Některé nástroje HTML5 byly i přesto nahrazeny knihovnami, které pomáhají s usnadněním opakující se práce. Jednou z těchto knihoven je Hyperscript, umožňující vytvářet a manipulovat s DOM elementy pomocí jednoduché a čisté syntaxe podobné zápisu HTML.

[19] Repozitář aplikace Codeblockie, s pracovním názvem CodeQuest, je dosupný na https://github.com/petrsysel/CodeQuest.

Na rozdíl od práce s DOM elementy přímo pomocí základních nástrojů HTML5, jež často vyžaduje opakující se kód a ve větším množství se stává nepřehlednou, Hyperscript poskytuje alternativu, která tvorbu DOM usnadňuje. Využití této knihovny usnadnilo dynamickou tvorbu šablon částí uživatelského rozhraní (Croker, 2021).

Návrh aplikace zahrnoval potřebu vykreslování herního pole. Pro tuto úlohu poskytuje HTML5 vhodný nástroj pro práci s 2D grafikou – HTML5 Canvas (Juviler, 2022). Vykreslování herní plochy pomocí základních nástrojů by opět znamenalo zbytečně mnoho práce. Existuje několik nadstaveb nad HTML5 Canvasem, které poskytují jednodušší vykreslování a správu 2D grafických objektů. Pro tento účel byla vybrána knihovna Konva, která poskytuje širokou škálu nástrojů, umožňující snadnou manipulaci s grafickými prvky a objekty, což značně usnadňuje tvorbu a interaktivitu hrací plochy (Singh, 2023).

Pomocí této knihovny byl implementován adaptér starající se o vykreslování herního pole a interagování s uživatelem. Konva umožnila snadnou tvorbu objektů, vykreslování jejich grafiky, řazení těchto objektů do vrstev, i vytvoření jejich animací při přehrávání animované vizualizace řešení hráčova řešení. Tento adaptér je nakonec poskytnut jádrům *Editoru* a *Spouštěče úloh*.

Také implementace vlastního nástroje pro blokové programování by bylo zbytečně náročné. Proto byla do aplikace přidána již existující knihovna Blockly vytvořená společností Google a poskytující nástroje pro sestavování blokového kódu a generování ekvivalentního Javascriptu v podobě textového řetězce na jeho základě (Google for Developers, 2023). Na knihovně Blockly je vystavěno několik výše zmíněných technologií, jako například Blockly Games, Roboblockly či Robomise.

Kromě předdefinovaných bloků, představujících běžné nástroje pro zápis algoritmů pomocí grafického jazyka, umožňuje Blockly definování vlastních bloků. K tomuto účelu poskytuje Blockly webové prostředí Blockly Developer Tools[20], které usnadňuje přípravu bloků a toolboxů. Tento nástroj generuje tři definiční soubory, jež je třeba implementovat pro

[20] Nástroj Blockly Developer Tools je dostupný z https://blockly-demo.appspot.com/static/demos/blockfactory/index.html.

vlastního softwaru. Jsou to definice bloků v podobě JSON objektů, toolboxu v podobě XML a definice generovaného Javascriptu v Javascriptu (Google for Developers, 2023).

Po tom, co uživatel sestaví kód, je nutné úlohu vyhodnotit. Mechanismus tohoto vyhodnocování představoval největší výzvu při implementaci návrhu aplikace. Problém, kterému bylo třeba čelit, byl způsoben principem fungování knihovny blockly. Jak již bylo zmíněno výše, knihovna Blockly generuje Javascriptový kód v podobě textového řetězce. Jako příklad lze vzít blokový kódu na Obrázek 9.

Obrázek 9 - Blokový kód, vytvořený pomocí knihovny Blockly

Pokud je definován kód, který budou bloky „Jdi vpřed“ a „Otoč se“ generovat, Blockly z tohoto vygeneruje Javascript na následující Ukázka 1.

```
// příklad kódu vygenerovaného pomocí Blockly.javascriptGenerator
for(let i = 0; i < 10; i++){
    // funkce goForward a turn jsou hypotetické funkce generované bloky pro pohyb
    goForward()
    turn('left')
}
goForward()
```

Ukázka 1 - Javascriptový kód, vygenerovaný knihovnou Blockly

Takto vygenerovaný kód lze spustit jedině funkcí *eval*. Funkce *eval* přijme jako argument textový řetězec, který se pokusí interpretovat jako validní Javascript. Po dobu, co funkce *eval* vykonává svůj kód, je blokováno hlavní vlákno prohlížeče. Není tedy možnost spustit dvě a více funkcí *eval* současně, aby se vykonávaly asynchronně. Ke správnému

vyhodnocení úlohy je ovšem nezbytné vyhodnocovat kód všech objektů úlohy zároveň, aby mohli správně reagovat na svou vzájemnou pozici, vzdálenost, dotyk či přijmout událost vyvolanou jiným objektem.

Tento problém byl řešitelný spuštěním kódu každého objektu úlohy v odděleném vlákně prohlížeče za použití technologie WebWorker, což je nástroj Javascriptu pro paralelní zpracování kódu (Paralkar, 2022). Tímto způsobem se podařilo jednodušší typy úloh vyřešit. Problém nastal při pokusu implementovat zasílání zpráv mezi objekty. Při pokusu o implementaci této funkce vyvolal nežádoucí chování objektů během vyhodnocování úlohy. Proběhlo několik pokusů i debugování tohoto řešení, ovšem neúspěšně.

Správným se ukázalo být řešení, při kterém nedocházelo k přímému vyhodnocování Javascriptového kódu, generovaným Blockly, ale sestavení struktury objektů na jeho základě. Klíčovou součástí této datové struktury jsou tzv. akce. Jedná se o objekty, reprezentující určitý druh chování objektu. Pro každý blok, který bude uživatel smět využívat je třeba nadefinovat odpovídající akci. Tyto akce mohou jako parametry konstruktoru přijímat jiné akce. Takto je zajištěna možnost vytvářet komplexní struktury akcí, které obsahují spustitelné funkce, skrze které je vyhodnocováno jejich chování. Pomocí funkce *eval* jsou tedy postupně sestaveny struktury akcí všech objektů úlohy. Ty je poté možné spustit a řídit jejich vykonávání pomocí objektu *Synchronizer*, který kontroluje připravenost objektů vykonávat akce a volá události pro spuštění další akce každého z objektů.

Na Ukázka 2 se nachází příklad sestaveného kódu z Obrázek 9 za pomoci akcí.

```
// objekt ActionContainer, obsahující více instací implementací třídy Action
actionCode = new ActionContainer([
    // nový kód generovaný blokem "opakuj X krát"
    new ForLoopAction(10,
        // kód generovaný vlastními bloky pro pohyb
        new GoAction(),
        new TurnAction('left')
    ),
    new GoAction()
])
```

Ukázka 2 - Generovaný kód s úravami pro vytváření instancí akcí

Toto řešení si vyžádalo reimplementovat generování Javascriptu všech vestavěných bloků knihovny Blockly. Na Ukázka 2 si lze povšimnout vygenerovaného kódu pro blok "Opakuj X krát". I přes náročnost implementace se řešení osvědčilo jako nejlépe fungující.

U rozsáhlejších aplikací vzniká motivace implementovat vhodný mechanismus pro zprostředkování komunikace mezi částmi aplikace. V aplikaci vystavěné na hexagonální architektuře se jedná především o komunikaci jádra a adaptérů. Jedním z takových řešení je systém událostí. Zavedení tohoto mechanismu umožňuje oddělit jednotlivé části aplikace a zároveň jim umožňuje komunikovat a reagovat na události bez přímého propojení. To znamená, že změny a události v jedné části aplikace mohou být snadno detekovány a spolu s potřebnými daty odeslány ostatním částem, které na ně mohou adekvátně reagovat. Tímto způsobem se zvyšuje modularita, znovupoužitelnost a flexibilita celé aplikace, což usnadňuje správu a údržbu kódu. Implementace mechanismu událostí umožňuje jednotlivým částem aplikace pracovat nezávisle na sobě, což zlepšuje celkovou škálovatelnost systému.

V kapitole o návrhu prototypu aplikace bylo zmíněno, že jednotlivé části si vyměňují informace skrze *Server*. Jádrem adaptéru pro komunikaci se serverem je Javascriptová funkce *fetch*, která slouží k zasílání požadavků na zadanou URL adresu a poskytující odpověď z téže adresy. Při potřebě využít službu některého z endpointů serveru je za použití funkce *fetch* odeslán požadavek na route odpovídající služby *Serveru*. Tímto způsobem je zprostředkována veškerá komunikace týkající se správy uživatelů a úloh.

Pro implementaci vzhledu aplikace byl použit jeden z nástroj HTML5 – jazyk kaskádových stylů CSS. Výsledný vzhled aplikace je uveden na příkladu *Editoru úloh*, který je vyobrazen na Obrázek 10.

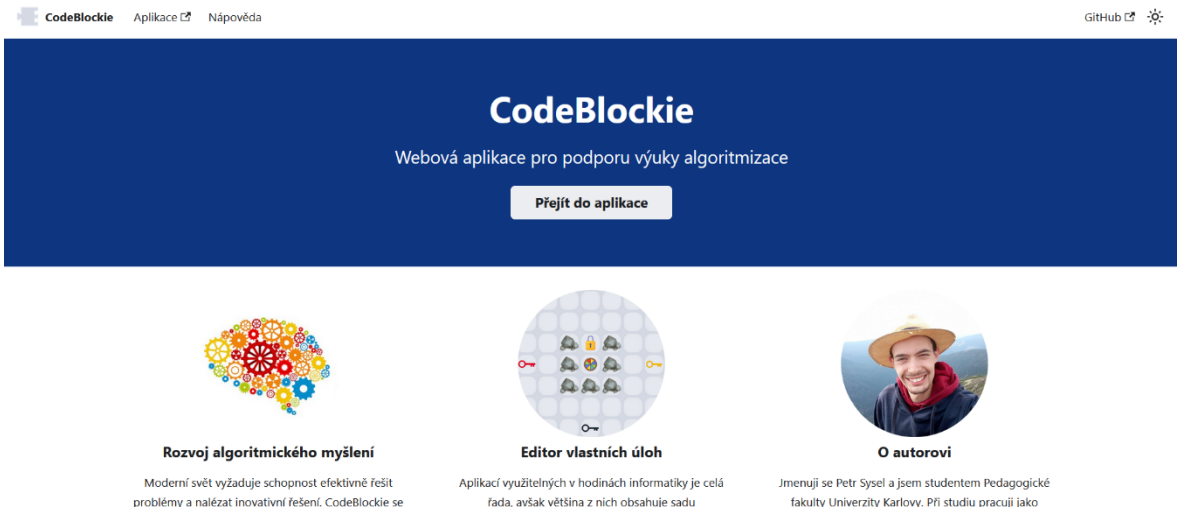
Obrázek 10 - Konečná podoba editoru úloh

Většina obrázků použitých v aplikaci byla použita z volně dostupných zdrojů. Ikony, použité pro vzhled tlačítek a favikony byly převzaty z webové stránky Icon8[21]. Grafika pro kostýmy objektů byly získány z webu Vecteezy[22]. V obou případech jsou použité obrázky licencovány pod volně šiřitelnou licencí, což umožňuje jejich bezplatné využití.

Grafická podoba aplikace je jedním z předmětů úvah o dalším rozvoji aplikace. V ideálním případě bude v budoucnu vytvořena nová grafika, včetně kostýmů obohacených o nové postavy a objekty využitelné k přípravě úloh. Tato grafika bude navržena s ohledem na získání vlastní identity v rámci designu.

Budoucí úpravy designu se týkají i hlavní stránky, která byla v pozdější fázi implementace provizorně vytvořena pomocí generátoru statických stránek Docusaurus[23]. Hlavní stránka obsahuje několik informací o jejím obsahu a stručnou nápovědu pro nové uživatele. Vzhled úvodní stránky je zobrazen na Obrázek 11.

[21] Webová stránka Icon8 je dostupná na https://icons8.com.
[22] Webová stránka Vecteezy je dostupná na https://www.vecteezy.com.
[23] Nástroj Docusaurus je dostupný na https://docusaurus.io



Obrázek 11 - Podoba hlavní stránky aplikace

5.1.2 Serverová část aplikace

Serverová část aplikace je postavena na technologii Node.js, která umožňuje spouštění Javascriptového kódu mimo prohlížeč, což stálo za motivací použít právě tuto technologii. S Node.js je možné napsat klientskou i serverovou část aplikace ve stejném programovacím jazyce. Tímto přístupem se eliminují obtíže při orientaci ve zdrojovém kódu, způsobené rozdíly v syntaxi a paradigmatu mezi klientskou a serverovou částí projektu (Semah, 2022).

Node.js díky svému balíčkovacímu systému NPM umožnilo snadné doinstalování potřebných Javascriptových knihoven. Jedním z instalovaných balíčků byl *Express* který poskytuje sadu nástrojů pro rychlé vytváření serverových aplikací. Jednou z jeho hlavních výhod je schopnost jednoduchého a intuitivního definování rout[24] a jejich handlerů, definující, jak aplikace reaguje na různé HTTP požadavky (GET, POST, PUT, DELETE) na určité URL adrese. Na těchto routách bylo vybudováno RESTful API serveru pro komunikaci s klientskou částí aplikace (Mozilla Foundation, 2024).

Pro plné fungování serveru bylo nezbytné zajistit způsob ukládání dat. Jako databáze byla k tomuto účelu vybrána MariaDB. MariaDB je open-source relační databázový systém, který je vyvíjen jako náhrada za MySQL. Vznikl v reakci na obavy ohledně budoucnosti MySQL po jeho akvizici společností Oracle Corporation. MariaDB je distribuována pod GPL (General Public License), což znamená, že je k dispozici zdarma (Jalli, 2022). Byla

[24] *Route* je cesta ke službě poskytované serverem (Routes and Endpoints, 2016)

vybrána právě díky svému open-source charakteru, možnosti dotazování za pomoci jazyku SQL, a snadné komunikaci s aplikací pomocí Javascriptového klienta mysql2 (MySQL2, 2024).

K zajištění bezpečnosti uživatelských hesel, byla do aplikace implementována knihovna Bcrypt, použita k jejich hashování před uložením do databáze, což udržuje uživatelská hesla v bezpečí i v případě, že je databáze aplikace kompromitována (Fikar, 2023).

Po dokončení implementace všech klíčových částí představovalo další krok nasazení aplikace na server. Za tímto účelem byl vybrán, zakoupen a konfigurován vhodný virtuální privátní server (VPS), který poskytoval potřebné prostředí pro běh aplikace. Byly na něm instalovány všechny technologie nezbytné pro její spuštění, včetně Node.js a MariaDB.

Součástí procesu bylo také zakoupení doménového jména *codeblockie.com*[25], které představuje adresu, pod kterou bude aplikace dostupná pro uživatele na internetu. Zakoupení a implementace SSL certifikátu na server byl též důležitý proces, který zajišťoval bezpečnou komunikaci mezi klientskou a serverovou částí aplikace, a tím chránil uživatelská data před neoprávněným přístupem (Cloudflare, 2024).

Po úspěšném dokončení procesu nasazení na server, byla aplikace připravena k použití a testování, což představovalo důležitý milník v jejím vývoji a přípravě k plnému uvedení do provozu.

5.2 Testování a ladění prototypu

Implementace aplikace se ukázala jako schopnou provozu. Po nasazení na server bylo několikrát nutné upravit některou z funkcí aplikace nebo přidat úplně novou. Díky tomu, že byla aplikace navržena a implementována podle hexagonální architektury, nevyvolaly dodatečné změny v aplikaci žádné neočekávané potíže. Tento efekt mohl být posílen rozhodnutím vyvíjet aplikaci pomocí jazyka Typescript, a využít tak jeho statickou typovou kontrolu.

S rostoucím rozsahem projektu se však začal projevovat nesprávný návrh kaskádových stylů, které se navíc postupně stávaly nepřehlednými a obtížně udržovatelnými. Pro budoucí

[25] Prototyp aplikace je umístěn na adrese https://codeblockie.com.

rozšíření a úpravy by proto bylo vhodné použít jeden z dostupných preprocesorů stylů, jako je Sass či Less Preprocesory stylů umožňují psát kaskádové styly s rozšířenou syntaxí pro zlepšení organizace a přehlednosti zdrojového kódu. Díky preprocesorům je možné využívat proměnné, modulární strukturu a další pokročilé funkce, které usnadňují tvorbu a údržbu stylů (Mozilla Foundation, 2024).

Další slabinou aplikace, týkající se její grafické podoby a související s nesprávným návrhem CSS, je její nízká míra responzivity. Aplikace byla navržena pro použití na stolních počítačích a laptopech v počítačových učebnách při výuce. Na zařízeních jako jsou tablety či mobilní telefony se aplikace nemusí zobrazovat správně, což je do budoucna v plánu napravit.

6 Ověření navržené aplikace v praxi

Tato kapitola detailně popisuje proces ověření prototypu aplikace v reálném prostředí školní třídy. Test byl zaměřen na ověření funkčnosti a uživatelské přívětivosti aplikace z pohledu jak učitele, tak i žáků prvního stupně základní školy. Testování probíhalo ve dvou fázích: nejprve aplikace podstoupila testování učitelem, který v aplikaci sestavil sadu úloh pro rozvoj algoritmického myšlení, a následně byla aplikace ověřena přímo v hodině informatiky čtvrté třídy základní školy.

Cílem ověřování bylo zjistit, zda je prototyp aplikace vhodný pro praktické využití v procesu přípravy úloh k výuce algoritmizace a následného využití těchto úloh v hodinách informatiky na prvním stupni základní školy a zda aplikace splňuje požadavky učitelů a žáků ohledně jednoduchého a intuitivního použití, efektivity při tvorbě úloh a schopnosti podpořit vzdělávací proces při výuce algoritmizace. Vedlejším cílem ověřování bylo také identifikovat případné nedostatky a oblasti k vylepšení, které by mohly vést k rozšíření a optimalizaci aplikace.

6.1 Příprava úloh učitelem

Pro otestování ze strany učitele byla aplikace poskytnuta učiteli základní školy, který ji podrobně prozkoumal, seznámil se s jejími funkcemi a možnostmi, a na základě toho si učitel připravil vzdělávací aktivity v rozsahu jedné vyučovací hodiny. Tyto aktivity zahrnovaly vytvoření úloh v testované aplikaci a následné sestavení pracovního listu (viz

Příloha 3) obsahujícího popis těchto úloh a odkazy na úlohy v prostředí aplikace. Pracovní list byl navržen tak, aby sloužil jako strukturovaný podklad pro žáky během výuky, který postupně seznamuje se základními principy algoritmizace.

Během prozkoumávání, testování aplikace a tvorby úloh byly učitelem odhaleny čtyři aspekty, které vyžadovaly pozornost. Ty byly analyzovány a následně řešeny, aby byly odstraněny ještě před představením aplikace dětem a další fází ověřování.

Prvním aspektem bylo nepropisování změn parametrů bloků do chování objektu (Obrázek 12). Při nastavování parametru bylo nutné potvrdit změnu stisknutím enteru či kliknutím myší mimo blok. To by mohlo negativně ovlivnit uživatelský zážitek žáků při práci s aplikací. Proto bylo nutné najít řešení tohoto problému.

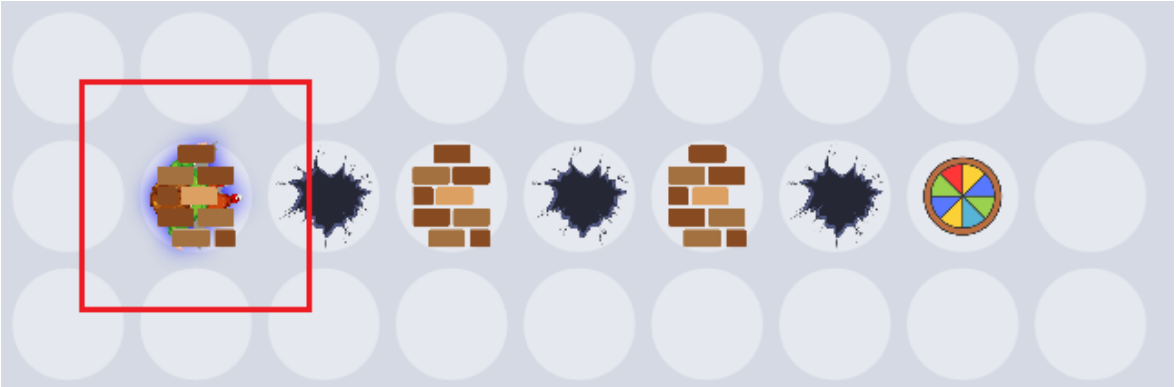
Obrázek 12 - Upravovaný parametr bloku

V momentě, kdy byla hodnota změněna z původního čísla 10 na číslo 2 a úprava nebyla potvrzena zmíněným kliknutím či stisknutím klávesy enter, nebyla úprava aplikací zaznamenána a kód byl vykonán s původní hodnotou 10 v bloku opakování.

Příčina tohoto chování byla analyzována a bylo zjištěno, že objekt Blockly.Workspace, v implementaci editoru kódu, vyvolává událost „change“, které bylo naslouchánо, pouze po opuštění upravovaného pole, stejně jako je tomu například u události „change“ na HTML elementu. Bylo třeba naslouchat události s názvem „block_field_intermediate_change“ aby bylo zajištěno předání informace o změně kódu jádru aplikace při každé změně parametrů bloků.

Dalším odhaleným potenciálním problémem bylo občasné nesprávné zobrazení vrstev objektů, což způsobovalo chybný vizuální překryv objektů. Příklad tohoto zobrazení je

červeně zvýrazněn na Obrázek 13. K tomuto jevu docházelo při spuštění úlohy nepravidelně, a při přehrávání animované vizualizace k němu nedocházelo vůbec.



Obrázek 13 - Chybné zobrazení vrstev objektů

Testy kódu zodpovědného za vykreslování hrací desky neodhalila chybu. K odhalení příčiny by bylo zapotřebí důkladnější analýzy, ke které před další fází ověřování, z nedostatku času, nedošlo.

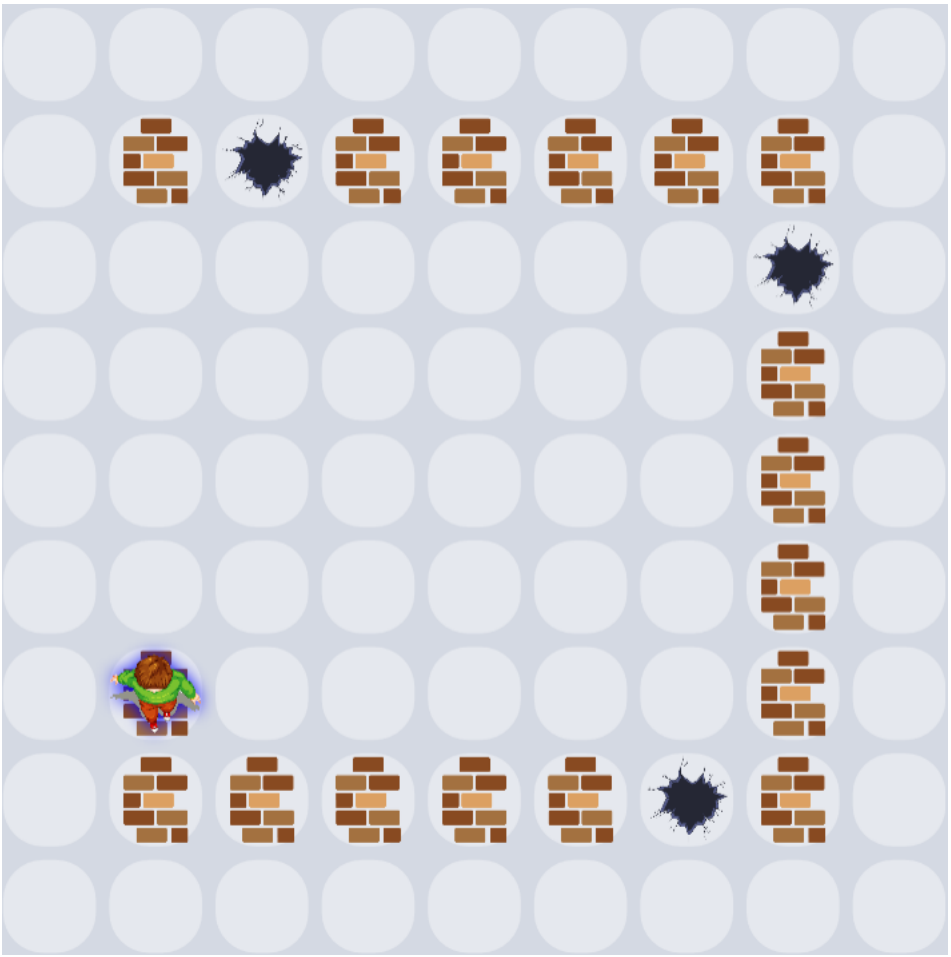
Učitel během práce s aplikací vyjádřil potřebu implementace dalšího bloku, který by byl žákům k dispozici. Jednalo se o blok pro pohyb objektu vpřed, jenž by umožňoval nastavení vzdálenosti, kterou má objekt urazit, zadáním hodnoty (v políčkách hrací desky) do parametru bloku. Obrázek 14 ukazuje kód pro pohyb objektu o pět políček vpřed pomocí původního bloku „Jdi vpřed“ (vlevo) a nově implementovaného bloku „Jdi vpřed o X“ (vpravo).

Obrázek 14 - srovnání kódu před a po přidání nového bloku "jdi vpřed o X"

Též byla projevena žádost o rozšíření grafiky kostýmů o další dva, barevně odlišené druhy cihel, symbolizující v úlohách cesty. Po obohacení kostýmů o navrhované obrázky bylo na výběr ze tří typů cihel (Obrázek 15, původní kostým je umístěn vlevo).

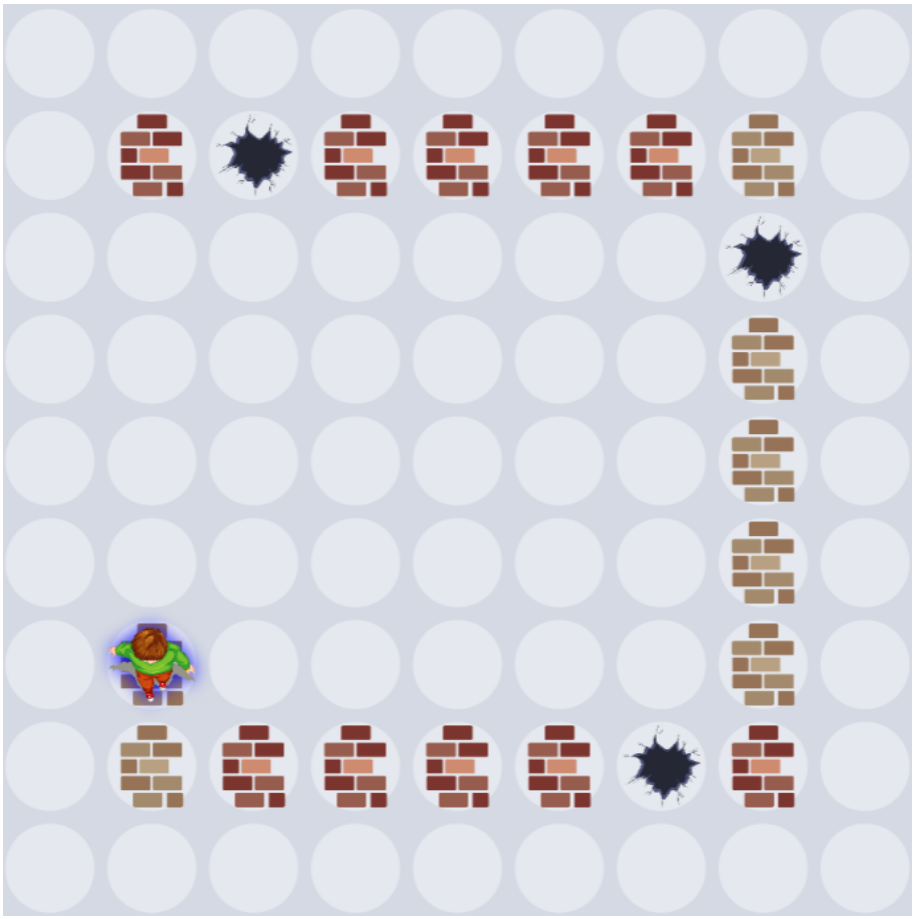
Obrázek 15 - varianty grafiky cihel (cesty)

Obohacení o další dva typy cihel pro vizualizaci cesty v úlohách bylo využito k vizuálnímu oddělení částí úlohy. Odlišná barva cest má žákovi napovědět využití cyklu, jestliže se nějaké části cest opakují. Ty pak jsou zvýrazněny odlišnými odstíny cesty. Jako příklad lze uvést úlohu, která byla vytvořena před přidáním dalších kostýmů cest. Původní podobu úlohy představuje Obrázek 16.



Obrázek 16 - Podoba jedné z úloh pracovního listu, před přidáním dodatečné grafiky cihel

Lze si povšimnout, že cesta je tvořena jedním typem cesty. Jak vzhled úlohy ovlivnilo přidání dalšího vzhledu cesty lze zhodnotit porovnáním s následující ukázkou (Obrázek 17) téže úlohy, ovšem s upravenou cestou za použití dalšího obrázku cesty.



Obrázek 17 - Podoba jedné z úloh pracovního listu po přidání dodatečné grafiiky cihel

Vizuální odlišení dílčích úseků cesty napovídá, které části lze vyřešit samostatně, a poté na ně aplikovat opakování. Jedná se tedy vizuální prvek s přesahem pro lepší porozumění úloze žákem.

Po úspěšném vytvoření úloh, sestavení pracovního listu a provedení výše popsaných úprav aplikace bylo vše připraveno k druhé fázi ověřování.

6.2 Ověření aplikace při hodině informatiky

Využitelnost aplikace byla testována při hodině informatiky žáků čtvrté třídy prvního stupně základní školy ZŠ Lupáčova. Před zahájením hodiny byly provedeny přípravy, mající za cíl zabránit technickým problémům během výuky, které by mohly způsobit narušení ověřování i samotné výuky. Byla provedena kontrola spustitelnosti aplikace na přítomných počítačích a nainstalovaných internetových prohlížečích (úlohy byly spouštěny v prohlížečích Microssoft Edge). Pracovní listy v podobě PDF souboru byly umístěny na pracovní plochu

každého z počítačů v učebně. Důvodem bylo urychlení procesu předání zadané práce žákům. Byla také provedena kontrola správného zobrazení pracovních listů v nainstalovaných aplikací pro čtení PDF souborů (jednalo se o PDF-XChange Editor).

Po zahájení hodiny byli žáci dotázáni na jejich zkušenost s algoritmizací a blokovým programováním. Ve třídě bylo celkem jedenáct žáků, přičemž z toho se s prostředím blokového programování setkali dva. Šlo o prostředí Scratch a Roblox. Ostatní žáci neměli žádné předchozí zkušenosti s programováním jakéhokoli druhu.

Dále byl žákům představen cíl hodiny – seznámení s algoritmizací pomocí řešení úloh v aplikaci Codeblockie. Aplikace byla žákům krátce představena a byl uveden způsob řešení úloh na příkladu první úlohy pracovního listu. Úloha obsahovala krátkou cestu, po které žáci museli navést postavičku do cíle pomocí bloků „Jdi vpřed o X“ a „Otoč se“.

Po tomto úvodu byla žákům přenechána volnost k řešení úloh uvedených v pracovním listě (viz Příloha 3). Žáci byli učitelem obcházeni, a v případě, že žák dlouho nemohl přijít na řešení úlohy, poskytl učitel radu, která měla žáka navést správným směrem k řešení dané úlohy.

V průběhu hodiny nebyl učitelem zaznamenán případ nesprávného zobrazení vrstev objektů popsaného výše ani jiný druh technického problému s aplikací.

Při plnění úloh žáky byl pozorován jev, při kterém se u žáků projevovala různá míra vyspělosti algoritmického myšlení. To způsobovalo odlišnou rychlost při plnění aktivit zadaných pracovním listem, a tudíž rozdílný počet úloh splněných během hodiny.

6.3 Zhodnocení výsledků ověřování

Testování aplikace v reálném vzdělávacím prostředí poskytlo cenné poznatky ohledně jejího využití při přípravě a vedení výuky algoritmizace na prvním stupni základní školy. Ověření praktické využitelnosti aplikace bylo úspěšné a aplikace splnila očekávání stanovená při jejím návrhu.

Aplikace Codeblockie získala pozitivní odezvu od učitele, který si vyzkoušel přípravu výukového obsahu pro rozvoj algoritmického myšlení, i žáky, kteří se skrze aplikaci seznámili se základy algoritmizace.

Je popsaný výše, při kterém docházelo k rozdílné rychlosti plnění úloh mezi žáky je přirozený a nebyl předmětem zkoumání této práce. Nicméně poukazuje na možnost připravit pro další vyučovací hodiny sady úloh přizpůsobených potřebám konkrétních žáků. Takovou flexibilitu při přípravě výukového obsahu výše zmíněné dostupné alternativní aplikace neobsahují.

Během testování byly identifikovány některé oblasti, ve kterých lze spatřovat potenciál pro budoucí vylepšení. Mezi ně patří možnost řazení úloh do větších celků a předávání těchto sad žákům. Nebo například možnost tvorby a administrace tříd, do kterých by mohl učitel přidávat své žáky a předávat jim skrze ni připravené úlohy či celé sady úloh. Projevily se též drobné technické vady, které ovšem jinak neomezují chod aplikace ani její využitelnost v reálném prostředí.

Testování aplikace Codeblockie potvrdilo její použitelnost pro podporu výuky algoritmizace na prvním stupni základní školy. Na základě získaných poznatků a doporučení bude aplikace dále vyvíjena a vhodně rozšiřována.

Závěr

Práce si kladla za cíl zmapovat dostupné webové nástroje a aplikace zaměřené na podporu výuky algoritmizace. Během této fáze byly identifikovány klíčové vlastnosti existujících aplikací. Analýza odhalila absenci nástrojů k tvorbě vlastních úloh.

Na základě poznatků z mapování byl vypracován návrh aplikace, který kladl důraz na vytvoření prostředí umožňující učitelům snadno vytvářet, upravovat a sdílet vlastní úlohy.

Samotná implementace prototypu aplikace Codeblockie byla provedena s cílem co nejvěrněji naplnit požadavky definované v návrhu aplikace. Důležité bylo zajistit intuitivní uživatelské rozhraní, které by umožnilo jednoduchou manipulaci s úlohami, jejich tvorbu a editaci. Při implementaci byly využity moderní technologie a metodiky vývoje softwaru, aby byla zajištěna stabilita, bezpečnost a škálovatelnost aplikace.

V implementaci prototypu existují určitá slabá místa, pro která jsou v plánu možná vylepšení. Patří sem například absence lepších nástrojů pro manipulaci s úlohami, jako je možnost jejich sdružování do sad, nedostatečná responzivita webu, s tím spojená slabá

podpora mobilních zařízení a občasné technické chyby. Žádné ze zmíněných slabých míst ale nebrání běhu aplikace či uživatelskému zážitku z jejího používání.

Výsledný prototyp byl následně podroben ověření v reálném prostředí školní třídy, kde byla zjišťována jeho využitelnost jak z pohledu učitele, tak žáků. Ověření prototypu potvrdilo jeho schopnost poskytnout efektivní nástroj pro výuku algoritmizace na základních školách a v zájmových útvarech. Úspěch v praxi naznačuje potenciál aplikace v této oblasti a otevírá cestu k dalšímu rozvoji a vylepšení.

Závěrečné zhodnocení potvrzuje splnění cílů práce a naznačuje perspektivu budoucího rozvoje. S využitím získané zpětné vazby z testování bude aplikace dále optimalizována a přizpůsobena potřebám učitelů a žáků. Bude se tak snažit přispívat k podpoře výuky algoritmizace a rozvoje algoritmického myšlení na základních školách.

Seznam použitých informačních zdrojů

21K SCHOOL, 2021. *How to Explain Algorithms to Kids*. Online. 21K School. Dostupné z: https://www.21kschool.com/in/blog/how-to-explain-algorithms-to-kids/. [cit. 2024-04-14].

About Code.org. Online. 2024. Dostupné z: https://code.org/about. [cit. 2024-04-10].

AKINDUYO, Eniola, 2023. *CodeMonkey Review: Should you enrol Your Kids?* Online. LinkedIn. Dostupné z: https://www.linkedin.com/pulse/codemonkey-review-should-you-enrol-your-kids-eniola-akinduyo-carif/. [cit. 2024-04-09].

An introduction to Git: what it is, and how to use it, 2018. Online. SRIDHAR, Aditya. FreeCodeCamp. Dostupné z: https://www.freecodecamp.org/news/what-is-git-and-how-to-use-it-c341b049ae61/. [cit. 2024-04-10].

ASHTARI, Hossein, 2022. *What Is HTML5? Meaning, Elements, and Benefits*. Online. Spiceworks. Dostupné z: https://www.spiceworks.com/tech/tech-general/articles/what-is-html-five/. [cit. 2024-04-10].

BAIERLOVÁ, Štěpánka, 2021. *Algoritmizace a programování - jedna ze 4 částí vzdělávací oblasti informatika*. Online. Npi Metodický portál RVP.CZ. Dostupné z: https://digifolio.rvp.cz/artefact/file/download.php?file=97884&view=20459. [cit. 2024-04-14].

BASUMALLICK, Chiradeep, 2022. *What is Gamification? Definition, Software, Examples, and Best Practices 2022*. Online. Spiceworks. Dostupné z: https://www.spiceworks.com/tech/devops/articles/what-is-gamification/. [cit. 2024-04-02].

BASUMALLICK, Chiradeep, 2024. *What Is Scratch Coding? Meaning, Working, and Applications*. Online. Spiceworks. Dostupné z: https://www.spiceworks.com/tech/devops/articles/scratch-coding/. [cit. 2024-04-09].

BILHAM, Jasmine, 2021. *Case study: How Duolingo Utilises Gamification to Increase User Interest*. Online. Raw Studio. Dostupné z: https://raw.studio/blog/how-duolingo-utilises-gamification/. [cit. 2024-04-10].

Blockly Games. Online. Annenberg Learner. Dostupné z: https://www.learner.org/series/project-playbook-educator-edition/blockly-games/. [cit. 2024-04-10].

BRDIČKA, Bořivoj, 2014. *Informatické myšlení jako výukový cíl*. Online. Npi. Dostupné z: https://spomocnik.rvp.cz/clanek/18689/INFORMATICKE-MYSLENI-JAKO-VYUKOVY-CIL.html?nahled=. [cit. 2024-04-10].

BROWN, Olivia, 2023. *The Complete List of ICT Tools in Education*. Online. Osbot. Dostupné z: https://www.obsbot.com/blog/e-classes/ict-tools-in-education#Part%201. [cit. 2024-04-14].

CLOUDFLARE, 2024. *How does SSL work? | SSL certificates and TLS*. Online. Cloudflare. Dostupné z: https://www.cloudflare.com/learning/ssl/how-does-ssl-work/. [cit. 2024-04-14].

CODE.ORG, 2022. *Hour of Code 2022: Explore, play, create!*. Online. Medium. Dostupné z: https://codeorg.medium.com/hour-of-code-2022-explore-play-create-4d16df85e400. [cit. 2024-04-09].

CodeMonkey. Online. Dostupné z: https://www.codemonkey.com/. [cit. 2024-04-10].

CORMEN, Thomas H.; LEISERSON, Charles Eric; RIVEST, Ronald L. a STEIN, Clifford, 2022. *Introduction to algorithms*. Fourth edition. Cambridge: The MIT Press. ISBN 9780262046305.

CROKER, Ben, 2021. *A First Look at _hyperscript*. Online. Put your lights on. Dostupné z: https://putyourlightson.com/articles/a-first-look-at-hyperscript. [cit. 2024-04-10].

CTPRIMED, 2023. *Roadmap for using Computational Thinking in schools*. Online. CTPrimEd. Dostupné z: https://computationalthinking.education/resources/Computational_thinking_roadmap_en.pdf. [cit. 2024-04-15].

ČERNÝ, Michal, 2021. *Informační gramotnost na středních školách: co, jak a proč?* Online. Revize RVP ZV. Dostupné z: https://clanky.rvp.cz/clanek/c/g/22760/INFORMACNI-GRAMOTNOST-NA-STREDNICH-SKOLACH-CO-JAK-A-PROC.html. [cit. 2024-04-02].

DENG, Wenbo; PI, Zhongling; LEI, Weina; ZHOU, Qingguo a ZHANG, Wenlan, 2020. Pencil Code improves learners' computational thinking and computer learning attitude. Online. *Computer Applications in Engineering Education*. Roč. 28, č. 1, s. 90-104. ISSN 1061-3773. Dostupné z: https://doi.org/10.1002/cae.22177. [cit. 2024-04-09].

DHADUK, Hiren, 2022. *Best Frontend Frameworks for Web Development*. Online. Simform. Dostupné z: https://www.simform.com/blog/best-frontend-frameworks/. [cit. 2024-04-10].

DOĞAN, Adem, 2020. Algorithmic Thinking in Primary Education. Online. *International Journal of Progressive Education*. 2020-08-13, roč. 16, č. 4, s. 286-301. ISSN 1554-5210. Dostupné z: https://doi.org/10.29329/ijpe.2020.268.18. [cit. 2024-04-14].

DOGANTEKIN, Tugce Konuklar, 2020. *Hexagonal (Ports & Adapters) Architecture*. Online. Medium. Dostupné z: https://medium.com/idealo-tech-blog/hexagonal-ports-adapters-architecture-e3617bcf00a0. [cit. 2024-04-14].

EARLY, Tom, 2022. *What is Vite and why do you need it?* Online. GPMD. Dostupné z: https://www.gpmd.co.uk/blog/what-is-vite-and-why-do-you-need-it. [cit. 2024-04-10].

EDWARDS, Luke, 2021. *What is Tynker and How Does It Work? Best Tips and Tricks*. Online. Tech & Learning. Dostupné z: https://www.techlearning.com/how-to/what-is-tynker-and-how-does-it-work-best-tips-and-tricks. [cit. 2024-04-14].

EDWARDS, Luke, 2023. *What is Ozaria and How Can It Be Used for Teaching? Tips & Tricks*. Online. Tech & Learning. Dostupné z: https://www.techlearning.com/how-to/what-is-ozaria-and-how-can-it-be-used-for-teaching-tips-and-tricks. [cit. 2024-04-09].

EL MAWAS, Nour; TRÚCHLY, Peter; PODHRADSKÝ, Pavol; MEDVECKÝ, Martin a MUNTEAN, Cristina Hava, 2022. Impact of game-based learning on STEM learning and motivation: Two case studies in Europe. Online. *Knowledge Management & E-Learning:*

An International Journal. S. 360-394. ISSN 20737904. Dostupné z: https://doi.org/10.34105/j.kmel.2022.14.020. [cit. 2024-04-14].

FIKAR, Jan, 2023. *Bcrypt po 25 letech*. Online. Root.cz. Dostupné z: https://www.root.cz/zpravicky/bcrypt-po-25-letech/. [cit. 2024-04-10].

FOWLER, Martin, 2003. *Patterns of enterprise application architecture*. The Addison-Wesley Signature Series. Boston: Addison-Wesley. ISBN 0321127420.

GalaxyCodr. Online. Dostupné z: https://galaxycodr.com. [cit. 2024-04-10].

GOOGLE FOR DEVELOPERS, 2023. *Blockly Developer Tools*. Online. Blockly. Dostupné z: https://developers.google.com/blockly/guides/create-custom-blocks/blockly-developer-tools. [cit. 2024-04-14].

GOOGLE FOR DEVELOPERS, 2023. *What is Blockly*. Online. Blockly. Dostupné z: https://developers.google.com/blockly/guides/get-started/what-is-blockly. [cit. 2024-04-14].

GROVER, Shuchi a PEA, Roy, 2018. Computational Thinking: A Competency Whose Time Has Come. Online. In: SENTANCE, Sue; BARENDSEN, Erik a SCHULTE, Carsten (ed.). *Computer Science Education*. Bloomsbury Academic. ISBN 978-1-3500-5714-2. Dostupné z: https://doi.org/10.5040/9781350057142.ch-003. [cit. 2024-04-09].

Hour of Code, 2024. Online. Dostupné z: https://hourofcode.com/cz. [cit. 2024-04-10].

How To Restart Your Node.js Apps Automatically with nodemon, 2024. Online. DigitalOcean. Dostupné z: https://www.digitalocean.com/community/tutorials/workflow-nodemon. [cit. 2024-04-10].

IMyšlení: FAQ, 2018. Online. IMyšlení. Dostupné z: https://www.imysleni.cz/informaticke-mysleni/imysleni-faq. [cit. 2024-04-10].

Jak na novou informatiku v RVP ZV, 2021. Online. Dostupné z: https://revize.edu.cz/nova-informatika. [cit. 2024-04-02].

JALLI, Artturi, 2022. *What Is MariaDB?* Online. Built In. Dostupné z: https://builtin.com/data-science/mariadb. [cit. 2024-04-10].

JUVILER, Jamie, 2022. *HTML Canvas: How to Get Started*. Online. HubSpot Blog. Dostupné z: https://blog.hubspot.com/website/html-canvas. [cit. 2024-04-10].

K-NET TECHNICAL INTERNATIONAL GROUP. Instalace nežádoucích programů na školní stanice. Online. K-net. 2019. Dostupné z: https://www.k-net.cz/reseni/pro-koho/it-pro-skoly/pomoc-se-skolni-infrastrukturou/instalace-nezadoucich-programu-na-skolni-stanice/. [cit. 2024-04-10].

Kdo je Karel, 2006. Online. Karel. Dostupné z: http://karel.oldium.net/napoveda.html. [cit. 2024-04-09].

KHNUT, Donald E., 1980. *Algorithms in modern mathematics and computer science*. Dostupné z: https://apps.dtic.mil/sti/tr/pdf/ADA089912.pdf. [cit. 15.12. 2023n. l.0].

KLEMENT, Milan; DRAGON, Tomáš a BRYNDOVÁ, Lucie, 2020. *Computational thinking and how to develop it in the educational process*. Olomouc: Palacký University Olomouc. ISBN 978-80-244-5796-3.

KRÁL, Filip, 2018. *Gamifikace ve vzdělávání*. Online, Diplomová práce, vedoucí Mgr. Vít Dočekal, Ph.D. Olomouc: Univerzita Palackého v Olomouci. Dostupné z: https://theses.cz/id/lyquhg/Krl_Gamifikace_ve_vzdlvn.pdf. [cit. 2024-04-14].

LCOM TEAM, 2022. *7 Examples of Algorithms in Everyday Life for Students*. Online. Learning.com. Dostupné z: https://www.learning.com/blog/7-examples-of-algorithms-in-everyday-life-for-students/. [cit. 2024-04-14].

LCOM TEAM, 2023. *Why Is Computational Thinking Important for Students?* Online. Learning.com. Dostupné z: https://www.learning.com/blog/why-is-computational-thinking-important-for-students/. [cit. 2024-04-14].

LESSNER, Daniel, 2014. *Informatické myšlení (2): různá vymezení*. Online. Učíme informatiku. Dostupné z: http://ucime-informatiku.blogspot.com/2014/09/informaticke-mysleni-2-ruzna-vymezeni.html. [cit. 2024-04-14].

Logiscool. Online. Dostupné z: https://www.logiscool.com. [cit. 2024-04-10].

Make & Code. Online. Microsoft MakeCode. Dostupné z: https://www.microsoft.com/en-us/makecode. [cit. 2024-04-09].

MAREŠ, Martin a VALLA, Tomáš, 2022. *Průvodce labyrintem algoritmů*. Druhé vydání. CZ.NIC. Praha: CZ.NIC, z.s.p.o. ISBN 978-80-88168-63-8.

MARTINEZ, Pablo, 2021. *Hexagonal Architecture, there are always two sides to every story*. Online. Medium. Dostupné z: https://medium.com/ssense-tech/hexagonal-architecture-there-are-always-two-sides-to-every-story-bc0780ed7d9c. [cit. 2024-04-04].

MCGONIGAL, Jane, 2011. *Reality Is Broken: Why Games Make Us Better and How They Can Change the World*. Penguin Books. ISBN 978-0143120612.

MIT MEDIA LAB. *Scratch 2.0: Cultivating Creativity and Collaboration in the Cloud*. Online. MIT Media Lab. 2009. Dostupné z: https://web.media.mit.edu/~mres/proposals/Scratch-CreativeIT.pdf. [cit. 2024-04-02].

MOURANT, Ron. An Introduction to Tinkercad Codeblocks. Online. Medium. 2018. Dostupné z: https://ronm333.medium.com/an-introduction-to-tinkercad-codeblocks-b896dff5f5f9. [cit. 2024-04-09].

MySQL2, 2024. Online. Npm. Dostupné z: https://www.npmjs.com/package/mysql2. [cit. 2024-04-14].

O Logiscool, 2024. Online. Logiscool. Dostupné z: https://www.logiscool.com/cz/about. [cit. 2024-04-09].

Ozaria. Online. Dostupné z: https://www.ozaria.com/. [cit. 2024-04-10].

PARALKAR, Keyur, 2022. *How Web Workers Work in JavaScript – With a Practical JS Example*. Online. FreeCodeCamp. Dostupné z: https://www.freecodecamp.org/news/how-webworkers-work-in-javascript-with-example/. [cit. 2024-04-14].

Pencilcode. Online. Dostupné z: https://pencilcode.net. [cit. 2024-04-10].

PLASS, Jan L.; HOMER, Bruce D. a KINZER, Charles K., 2016. Foundations of Game-Based Learning. Online. *Educational Psychologist*. 2016-02-06, roč. 50, č. 4, s. 258-283. ISSN 0046-1520. Dostupné z: https://doi.org/10.1080/00461520.2015.1122533. [cit. 2024-04-14].

REPENNING, Alexander; BASAWAPATNA, Ashok a ESCHERLE, Nora, 2016. Computational thinking tools. Online. In: *2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, s. 218-222. ISBN 978-1-5090-0252-8. Dostupné z: https://doi.org/10.1109/VLHCC.2016.7739688. [cit. 2024-04-14].

About RoboBlockly. Online. RoboBlockly. Dostupné z: https://roboblocky.com/about. [cit. 2024-04-09].

Robomise. Online. Dostupné z: https://robomise.cz/. [cit. 2024-04-10].

Routes and Endpoints, 2016. Online. Wordpress. Dostupné z: https://developer.wordpress.org/rest-api/extending-the-rest-api/routes-and-endpoints/. [cit. 2024-04-10].

RunMarco. Online. BrainPOP Educators. Dostupné z: https://educators.brainpop.com/bp-game/run-marco/. [cit. 2024-04-09].

SADIKA, 2023. *The MVC Architecture*. Online. Medium. Dostupné z: https://medium.com/@sadikarahmantanisha/the-mvc-architecture-97d47e071eb2. [cit. 2024-04-14].

Scratch. Online. Dostupné z: https://scratch.mit.edu/. [cit. 2024-04-10].

SEMAH, Benjamin, 2022. *What Exactly is Node.js? Explained for Beginners*. Online. FreeCodeCamp. Dostupné z: https://www.freecodecamp.org/news/what-is-node-js/. [cit. 2024-04-10].

SHOHIEB, Samaa Mohammed; DOENYAS, Ceymi a AL-ADROUSY, Waleed Mohamed, 2022. Merging Tangibles and Gamification to Teach Algorithmic Thinking to KG Children With "Gamirithmic." Online. In: BERNARDES, Oscar; AMORIM, Vanessa a MOREIRA, Antonio Carrizo (ed.). *Handbook of Research on the Influence and Effectiveness of Gamification in Education*. Advances in Game-Based Learning. IGI Global, 2022-5-20, s. 682-705. ISBN 9781668442876. Dostupné z: https://doi.org/10.4018/978-1-6684-4287-6.ch033. [cit. 2024-04-10].

SHUBEL, Meredith, 2022. *What Is TypeScript?* Online. The New Stack. Dostupné z: https://thenewstack.io/what-is-typescript/. [cit. 2024-04-10].

SIEIŃSKA, Katarzyna a ORDZA, Tomasz, 2022. *Best practice guide of STEAM methodology in eTwinning projects for future teachers*. Online. ORDZA, Tomasz. European School Education Platform. Dostupné z: https://school-education.ec.europa.eu/system/files/2022-12/ecr2022-best-practice-guide-steam-methodology.pdf. [cit. 2024-04-14].

SINGH, Jatin Kumar, 2023. *Characteristics of an Algorithm*. Online. Codingninjas. Dostupné z: https://www.codingninjas.com/studio/library/characteristics-of-an-algorithm. [cit. 2023-12-15].

SINGH, Shweta, 2023. *What is konva.js? & how to use it?* Online. Webkul Blog. Dostupné z: https://webkul.com/blog/what-is-konva-js-how-to-use-it-for-the-reach-graphic/. [cit. 2024-04-10].

The "FERTILE" project, 2024. Online. FERTILE. Dostupné z: https://fertile-project.eu/about/. [cit. 2024-04-09].

VIDENOVIK, Maja; VOLD, Tone; KIØNIG, Linda; MADEVSKA BOGDANOVA, Ana a TRAJKOVIK, Vladimir, 2023. Game-based learning in computer science education: a scoping literature review. Online. *International Journal of STEM Education*. Roč. 10, č. 1. ISSN 2196-7822. Dostupné z: https://doi.org/10.1186/s40594-023-00447-2. [cit. 2024-04-10].

Tynker. Online. Dostupné z: https://www.tynker.com/. [cit. 2024-04-10].

What Is GitHub? A Beginner's Introduction to GitHub, 2023. Online. Kinsta. Dostupné z: https://kinsta.com/knowledgebase/what-is-github/. [cit. 2024-04-10].

ZICHERMANN, Gabe a CUNNINGHAM, Christopher, 2011. *Gamification by Design*. O'Reilly Media. ISBN 978-1-449-39767-8.

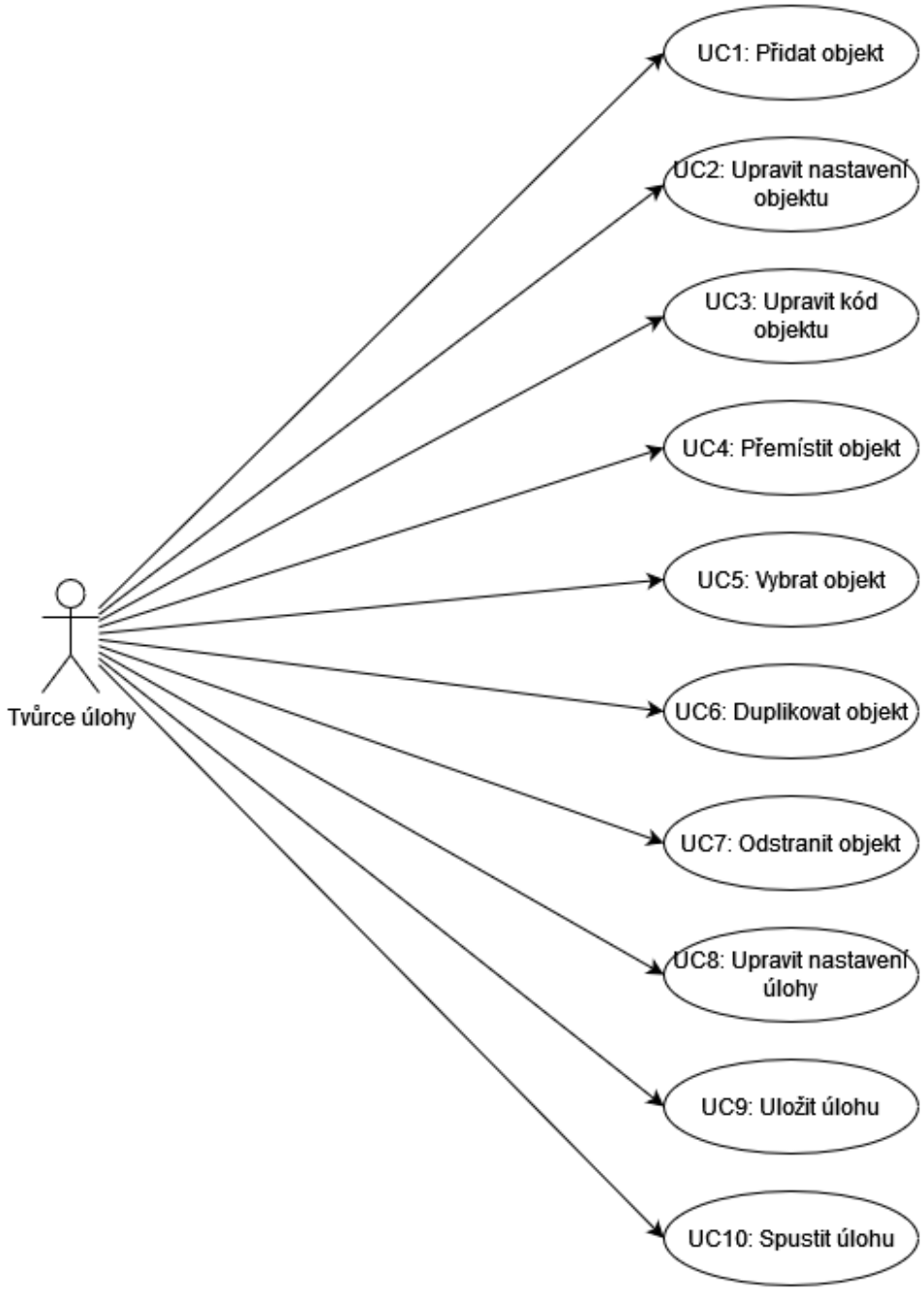
Seznam obrázků

Seznam ukázek zdrojového kódu

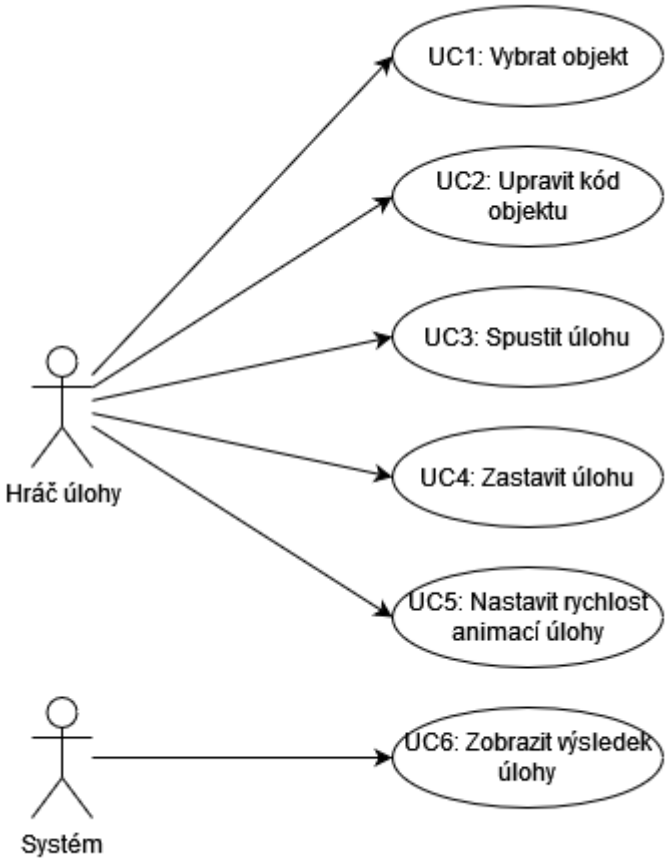
Seznam příloh

Příloha 1 – Use Case diagramy aplikace

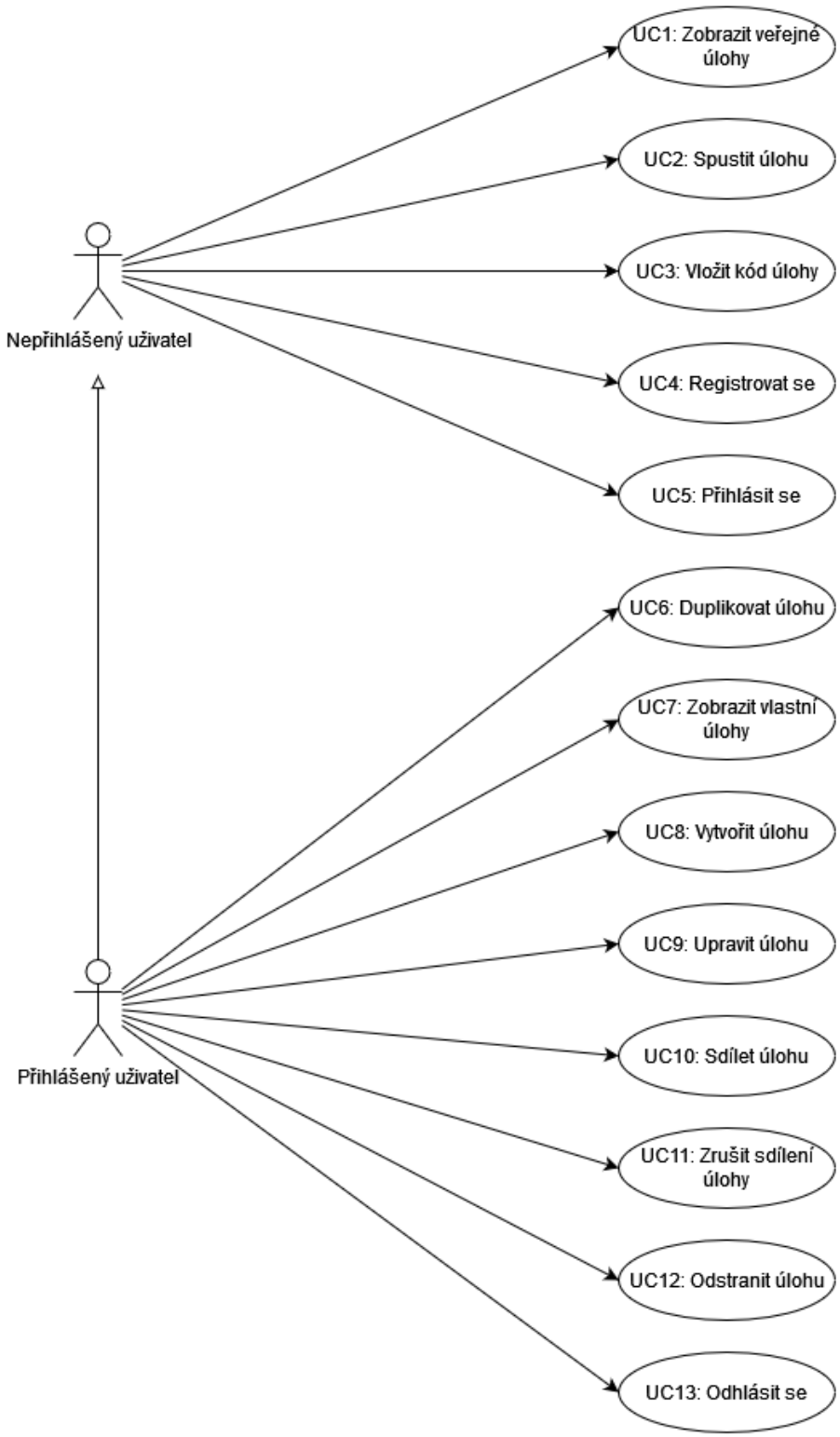
Use Case diagram *Editoru úloh*:



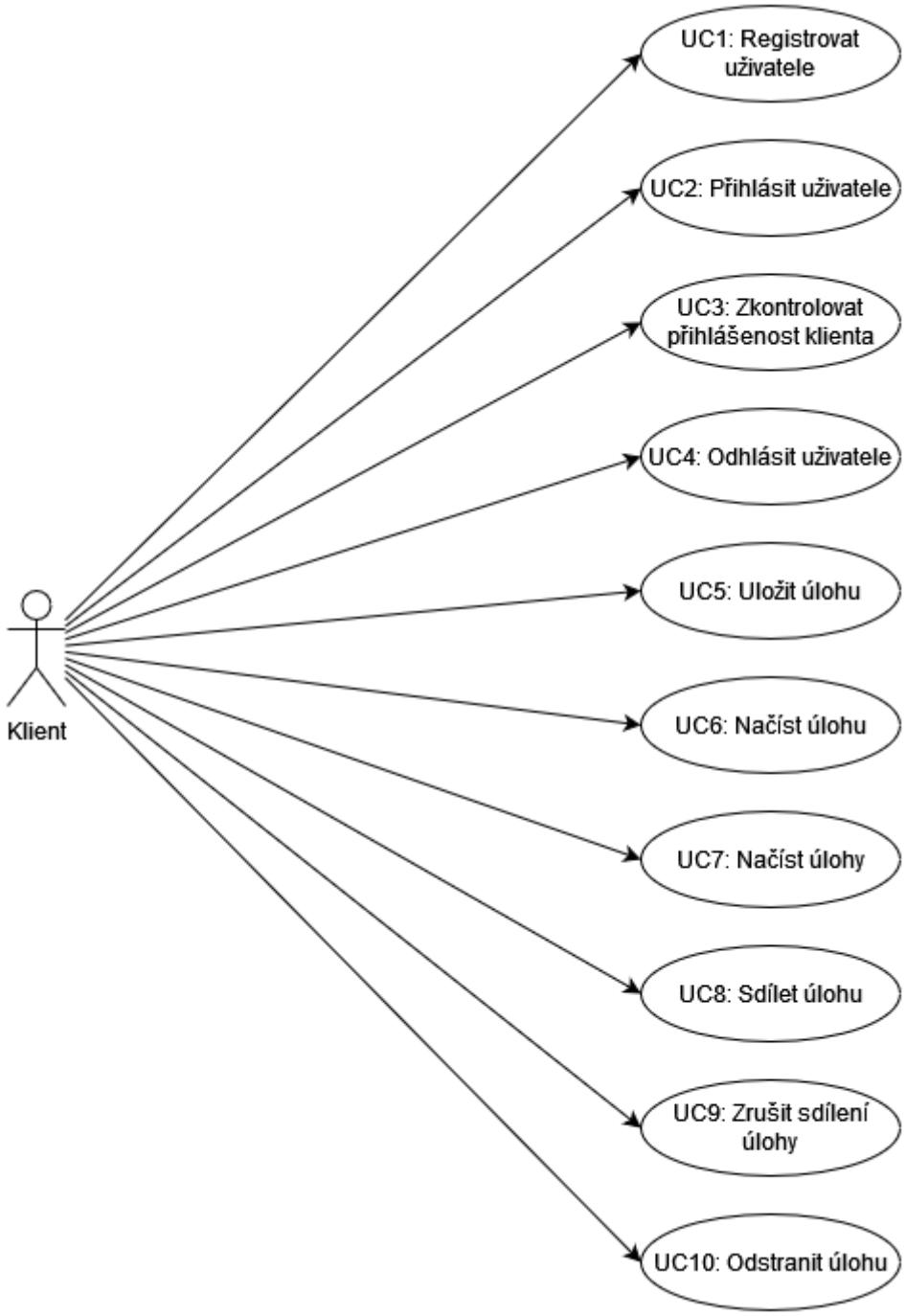
Use Case diagram části *Spouštěč úloh*:



Use Case diagram *Panelu úloh*:

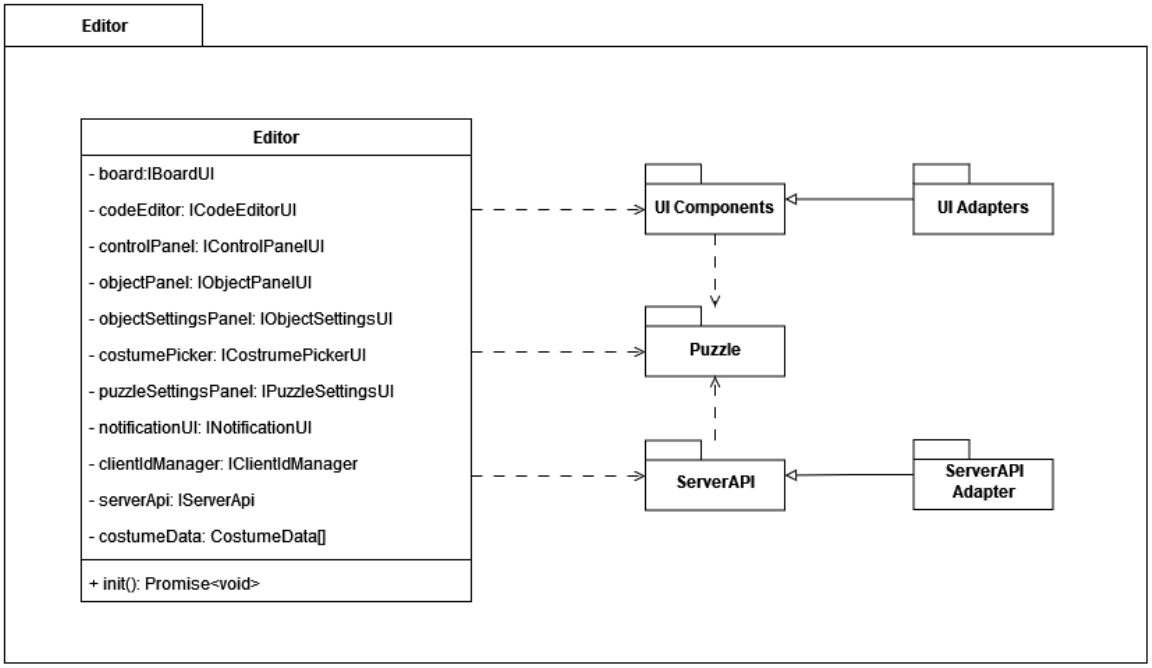


Use Case diagram *Serveru*:

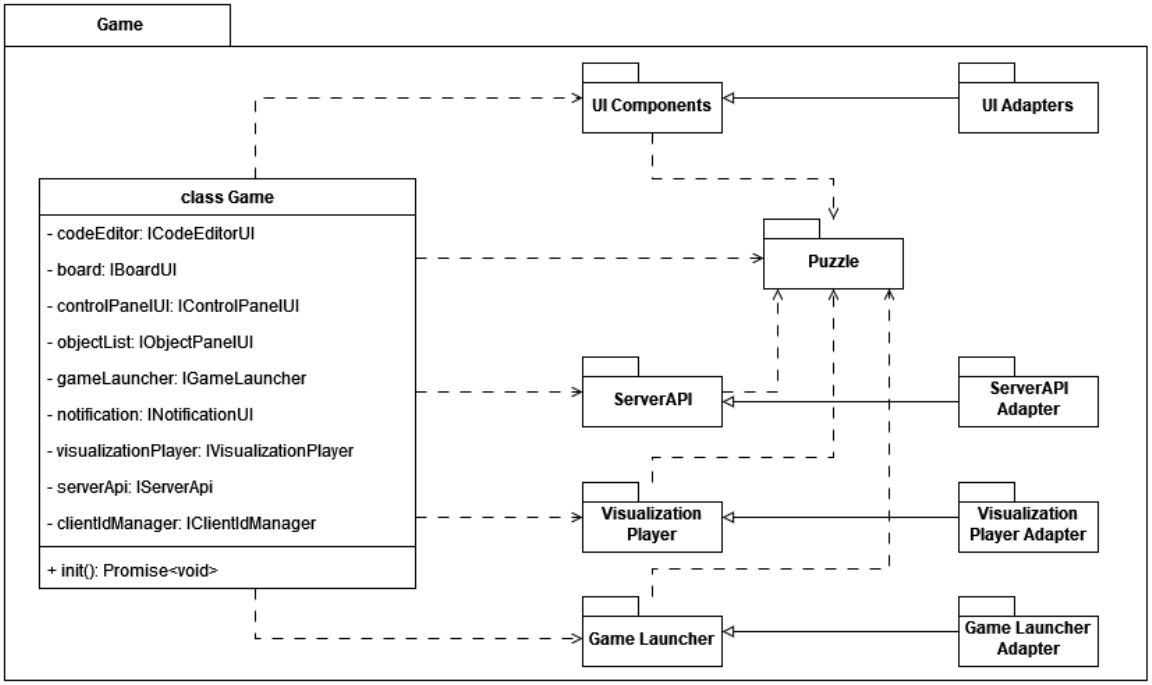


Příloha 2 – Class a Package diagramy klíčových částí aplikace

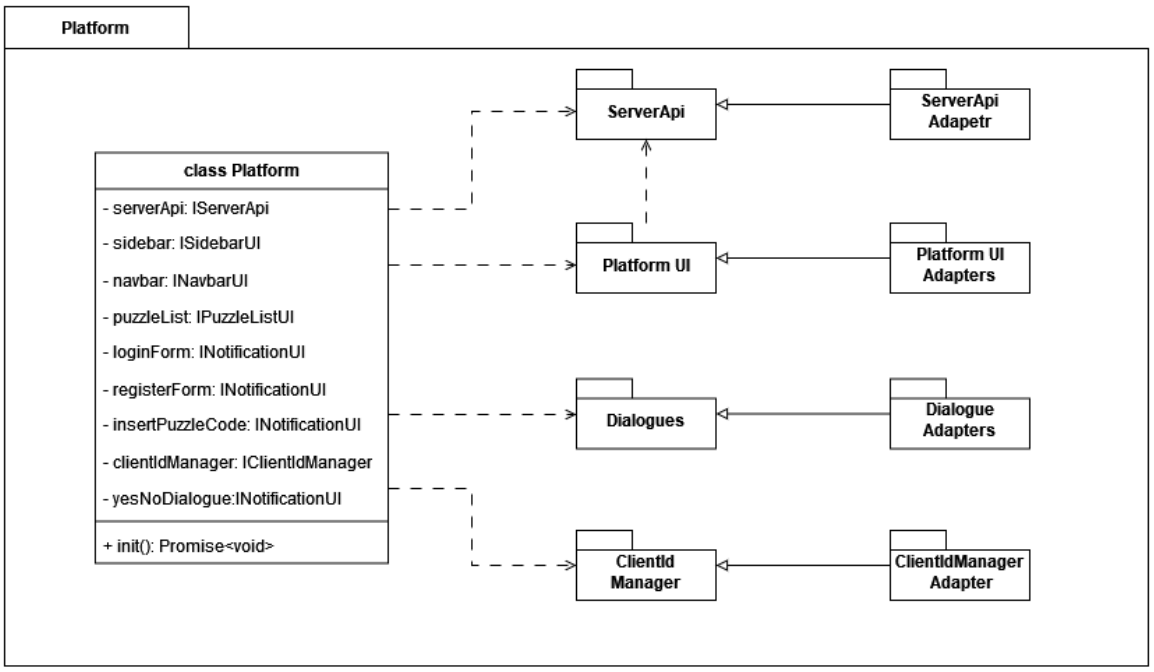
Class/Package diagram části *Editor úloh*:



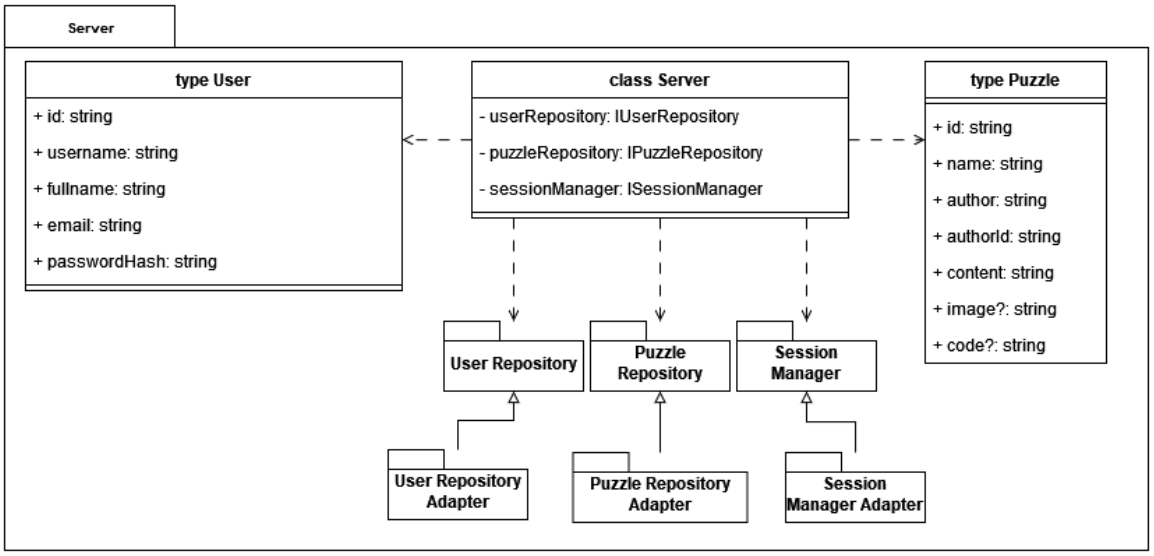
Class/Package diagram části *Spouštěč úloh*:



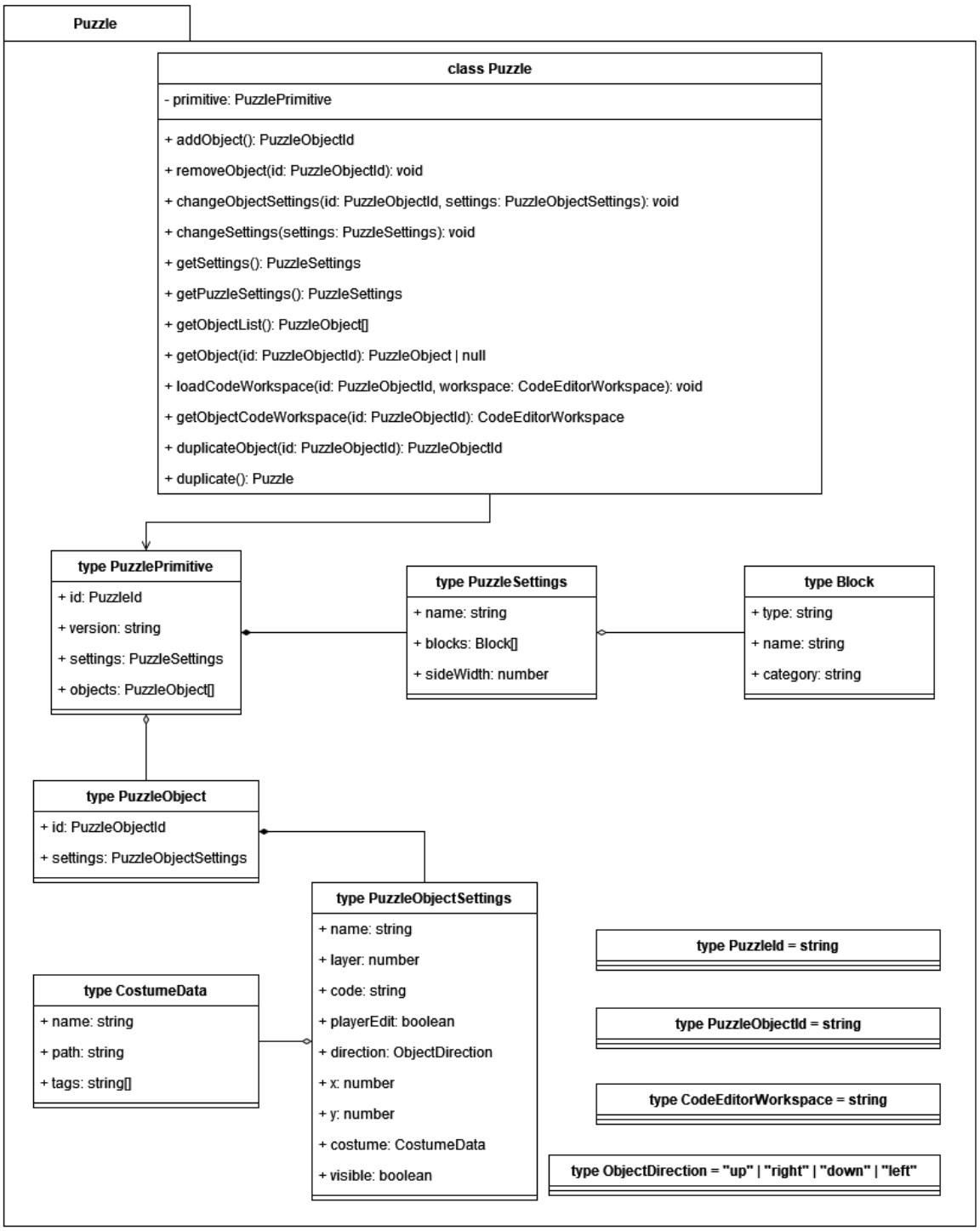
Class/Package diagram části *Panel úloh*:



Class/Package diagram části *Server*:



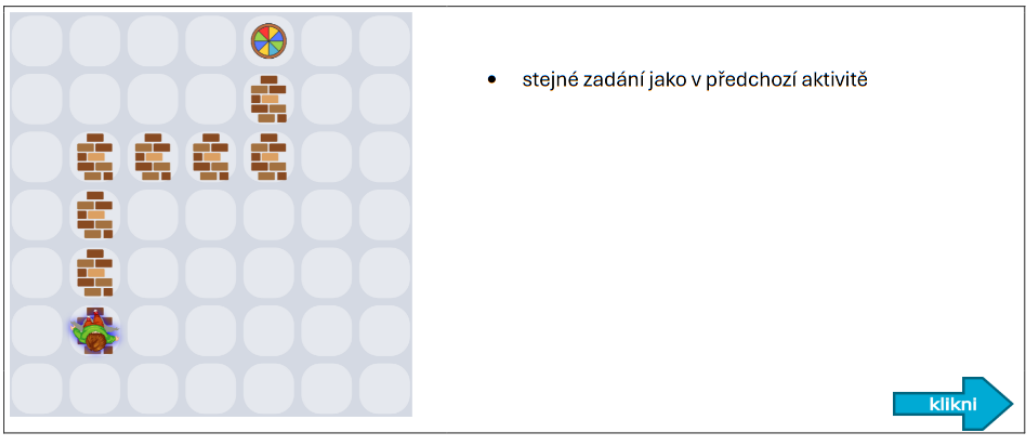
Class diagram datové struktury *Puzzle* pro reprezentaci úlohy:



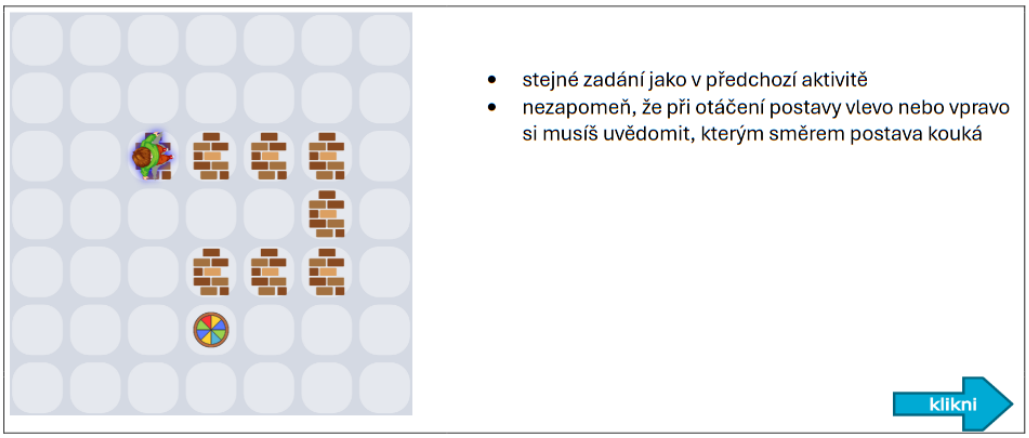
Příloha 3 – Pracovní list využitý při ověřování aplikace

1.

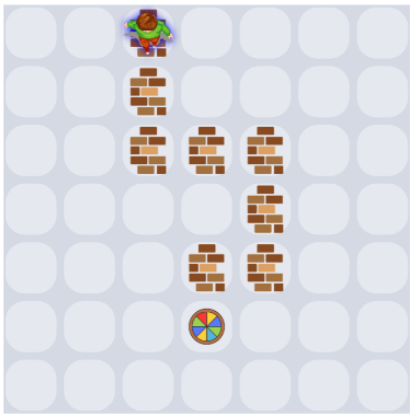
2.



3.



4.



- stejné zadání, ale trochu těžší

klikni

5.

díra – do té nesmíš spadnout

- v nabídce příkazů Pohyb najdeš nový příkaz Skoč
- a pozor, stále platí:
  - musíš jít po cihlové cestě
  - nesmíš šlápnout na světlou dlaždici

klikni

6.

- stejné zadání

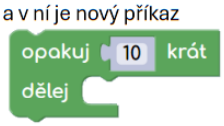
klikni

7.

- stejné zadání
- snadné řešení, stačí 3 skoky:

- zkus to ale jinak
  - vlevo přibyla nová nabídka 

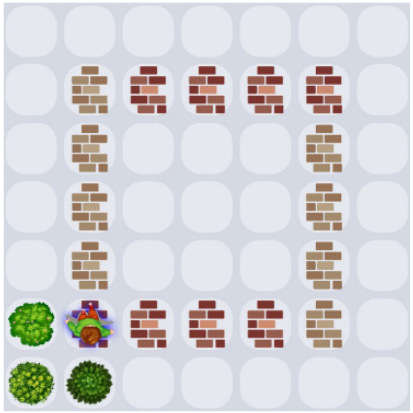
  - a v ní je nový příkaz



  - do něj můžeš vložit příkazy, které se mají opakovat (počet opakování můžeš změnit)

klikni

8.



- projdi celou cestou jednou dokola
- ***než začneš***: prohlédni si cestu pozorně a promysli, jaké by postava měla dostat příkazy
- neopakuje se něco?
- ***rada***: do příkazu opakuj můžeš vložit víc než jeden příkaz

klikni

9.

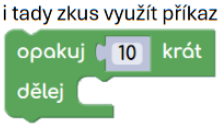
kámen – ten musíš přeskočit

- dostaň se na konec cesty do cíle
- opět si promysli, jaké by postava měla dostat příkazy
- neopakuje se něco?

klikni

10.

- dojdi na konec cesty
- i tady zkus využít příkaz opakování



klikni

11.

taky **kámen** – ten taky musíš přeskočit

- opět využij přikaz opakuj

klikni

12.

drahokamy

- nejdříve musíš posbírat drahokamy (stačí k nim dojít)
- až je posbíráš, zmizí kameny, které blokují cestu k cíli
- v této aktivitě to zkus bez příkazu opakování (*stějně by tu moc nepomohl*)

klikni

13.

- tady je to podobné: až posbíráš drahokamy, zmizí kámen, který blokuje cestu k cíli
- zkus využít přikaz opakuj

klikni

14.

- stejný úkol, jiná cesta
- využij přikaz opakuj

klikni

15.

kouzelník

postava hráče je uvězněná za zamčenými dveřmi, ale je tu ještě postava ***kouzelníka***

- naprogramuj kouzelníka tak, aby
  - nejdříve posbíral všechny klíče
  - a potom došel k zámku u dveří

  (*kouzelník pak už sám odemkne a osvobodí hráče*)

klikni

Příloha 4 - Prototyp aplikace Codeblockie

Zdrojový kód je přiložen externě.

Gitovský repozitář aplikace je dostupný na https://github.com/petrsysel/CodeQuest.