STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 18: Informatika

Codeventure

Michal Maděra, Josef Kahoun

Pardubický kraj

Pardubice, 2022

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 18: Informatika

Codeventure

Codeventure

Autoři: Michal Maděra a Josef Kahoun
Škola: DELTA - Střední škola informatiky a ekonomie, s.r.o., Ke Kamenci 151, 530 03 Pardubice
Kraj: Pardubický kraj
Konzultanti: RNDr. Jan Koupil, Ph.D. a akademický malíř Daniel Václavík

Pardubice, 2022

Prohlášení

Prohlašujeme, že jsme svou práci SOČ vypracovali samostatně a použili jsme pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

Nemáme závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších předpisů.

V Pardubicích dne 23. 3. 2022 .....

Michal Maděra, Josef Kahoun

Poděkování

Chtěli bychom poděkovat našim učitelům: RNDr. Janu Koupilovi, Ph.D. za vedení projektu po technické stránce a akademickému malíři Danielu Václavíkovi za vedení práce po grafické stránce. Dále bychom chtěli poděkovat všem lidem, kteří se zapojili testovací fáze našeho projektu.

Abstrakt

Práce SOČ dokumentuje návrh a vývoj hry Codeventure zaměřené na rozvoj algoritmických a programovacích dovedností u dětí mladšího školního věku. Realizace projektu zahrnuje backendovou část, mobilní aplikaci v roli ovladače hry a webový frontend v roli displeje. Součástí řešení je využití umělé inteligence pro strojové rozpoznání obrazu.

Klíčová slova

Programování; GraphQL; TensorFlow; algoritmické myšlení; 3D grafika

Abstract

This thesis documents the design and development of the Codeventure game focused on the development of algorithmic and programming skills of children, aimed at younger school age. The implementation of the project includes a backend part, a mobile application in the role of a game controller and a web frontend in the role of a display. Part of the solution is the use of artificial intelligence for machine image recognition.

Keywords

Programming; GraphQL; TensorFlow; alghoritmical thinking; 3D graphics

Obsah

1 Úvod

Celý svět kolem nás používá počítačové technologie s naprogramovanými aplikacemi. Právě kvůli tomu se s programováním setkáváme každý den, aniž bychom o tom věděli. V současné době se převádí mnoho školních činností do elektronické formy. Učebny jsou vybavovány interaktivními tabulemi, žáci pracují na počítačích či tabletech. Přesto je ale manipulace s reálnými objekty velmi důležitá pro rozvoj dětí nejen v předškolním, ale i mladším školním věku. Marie Montessori ve své koncepci vzdělávání dokonce tvrdí, že právě ruce napomáhají rozvoji intelektu. Když dítě používá své ruce, získává množství podnětů a zážitků a rozvíjí tak své vědomí a intelekt. [1]

Dle vývojových psychologů [2] již ve druhé třídě začíná být logika u většiny dětí obvyklým nástrojem k postihování reality. Přesto je dostatečně účinná pouze při úvahách o něčem konkrétním a názorném. A proto považují psychologové za nutné doplňovat výuku názornou demonstrací, nejlépe i s určitým podílem dětské aktivity. Dále se v [3] uvádí, že propojením neurologických a pedagogických výzkumů vyplývá, že kvalitní učení nastává v případě, že je zapojen doslova celý mozek, nejedná se tedy o aktivity zaměřené jen na levou či pravou hemisféru. Vhodné je zapojit u žáků všechny smysly, a to nikoli proto, aby se zvýšila zábavnost či zajímavost dané činnosti, ale protože se tím sníží kognitivní náročnost, zejména na pracovní paměť.

Na základě výše zmíněných faktů jsme se rozhodli vyvinout edukativní hru, která bude pro své fungování využívat hmatatelné objekty. Jedním z nejlepších a nejpřirozenějších způsobů, jak se dítě naučí jakékoli dovednosti, je hra, a není důvod, aby toto pravidlo neplatilo i pro algoritmické myšlení a programování. V dnešní době existují hry jako LightBot, Kodable a mnoho dalších, které děti rozvíjejí právě ve zmíněných okruzích. Často se však jedná o hry, které jsou určeny pro menší zařízení, jako tablety, či telefony a dítě prochází hrou pouze za pomoci dotykové obrazovky.

Tzv. gamifikace učení je celosvětovým trendem s pozitivními výsledky [4], proto jsme se stejnou cestou vydali i my. Cílem práce je vytvořit hru, která rozvine logické myšlení a základy programování. Celková struktura hry je navržená tak, abychom dokázali děti co nejefektivněji naučit zapojit logické myšlení a rozvinuli jejich dovednost základů programování.

2 Teoretická část

2.1 Pojmy

2.1.1 API

API (Application programming interface) je rozhraní, které umožňuje předávání dat mezi frontendem a backendem.

2.1.2 Backend

Backend, neboli zadní strana, je kód, který běží na serveru. Na tuto část se pomocí API dotazuje frontend. Uživatel není schopný se podívat na kód a nemůže měnit jeho funkcionalitu.

2.1.3 Cache

Cache, neboli mezipaměť, je úložné místo pro ukládání dočasných dat, s jehož pomocí jsou zrychleny weby, prohlížeče a aplikace. [5] Díky tomu se po určitý čas nemusí provádět složité výpočetní operace.

2.1.4 CDN

CDN, neboli content delivery network (síť pro doručování obsahu), je síť počítačů rozmístěných po celém světě, které spolupracují na poskytování rychlého doručování internetového obsahu.[6]

2.1.5 Databáze

Databáze je místo, kam si aplikace ukládá data. Dále si dokáže data zpracovávat a následovně vracet zpět do aplikace.

2.1.6 Framework

Framework je platforma, která poskytuje základ pro vývoj softwareových aplikací. [7] Jedná se o šablonu programu, kterou pak programátor rozšiřuje vlastním kódem.

2.1.7 Frontend

Frontend, neboli přední strana, je webové rozhraní, které vidí koncový uživatel. Veškerý kód, který se zde odehrává, běží u klienta, nikoliv na severu. Uživatel je schopný se podívat na kód, který u něj běží, a modifikovat jej.

2.1.8 Herní engine

Herní engine je framework pro výtváření počítačových her.

2.1.9 JSON

JSON (JavaScript Object Notation) je datový formát pro výměnu dat. Je jednoduše čitelný a je to jedna z možností, jak přes API rozhraní posílat data. [8]

2.1.10 Kompilace

Kompilace je proces, při kterém se člověkem čitelný kód převádí na kód čitelný pouze počítačem. Většinou není možné získat kód zpět do člověkem čitelné podoby po tom, co je zkompilován.

2.1.11 Komponenta

Komponenta je menší část aplikace, která poskytuje určitou funkcionalitu.

2.1.12 Kontejner

Kontejner je standardizovaná část softwaru, ve které je zabalený kód a všechny jeho potřebné závislosti, aby bylo možné aplikaci rychle spuštět a aby byla aplikace konzistentní mezi různými prostředími. [9]

2.1.13 Koprogram

Koprogram neboli Coroutine je část programu která si dokáže rozplánovat úlohu mezi více snímků.[10]

2.1.14 Open Source

Open source software je software se zdrojovým kódem, který si může kdokoliv stáhnout, upravit nebo vylepšit. [11]

2.1.15 ORM

ORM, neboli Objektově relační mapování, vytváří vrstvu mezi programovacím jazykem a databází [12]. Pomocí něj si navrhneme, jak by měla datová struktura vypadat, a on pak řeší všechny dotazy. Výhodou tedy je, že nemusíme psát dotazy přímo jazykem databáze, ale rovnou v našem zvoleném programovacím jazyce.

2.1.16 Škálování

Škálovatelnost softwaru je měřítko toho, jak lze část softwaru jednoduše zmenšovat nebo zvětšovat. [13] Například když máme moc velké využití backendu, tak jsme schopní spustit tento backend na více serverech zároveň.

2.1.17 Transpilace

Transpilace je proces, při kterém se vezme zdrojový kód napsaný v jednom programovacím jazyce a je převeden na podobný kód v jiném programovacím jazyce. [14]

2.1.18 VPS

VPS (Virtual Private Server) neboli virtuální soukromý počítač je označení pro virtuální počítač, který je prodáván jako služba.

2.1.19 WebSocket

WebSocket je komunikační protokol pro komunikaci mezi serverem a klientem [15]. Výhodou oproti HTTP požadavku je to, že komunikace je navázána na začátku a pak se dají požadavky posílat obousměrně bez jakékoliv prodlevy.

2.2 Technologie

2.2.1 Unity

Unity je herní engine určený hlavně pro nováčky, ale i pro pokročilé programátory. Jako programovací jazyk Unity využívá C# vyvíjený firmou Microsoft. Tento programovací jazyk je jedním z nejsnažších programovacích jazyků v současnosti na naučení pro začátečníky. [16] Je možné v něm vytvářet 2D i 3D aplikace. Výsledná hra je spustitelná na mnoha platformách, od počítačů až po mobilní zařízení. [17]

2.2.2 JavaScript

JavaScript je moderní programovací jazyk využívaný jak na frontendu, tak na backendu. Jazyk je tzv. interpretovaný jazyk, což znamená, že kód nekompiluje celý najednou, ale vykonává se řádek po řádku. [18]

2.2.3 TypeScript

TypeScript je pevně typový programovací jazyk, který je postaven nad JavaScriptem. Díky tomu je ve výsledné aplikaci méně chyb spojených s datovými typy, protože jsou kontrolovány během vývoje. Ve výsledku je kód transpilován zpět do JavaScriptu. [19]

2.2.4 React.js

React.js je JavaScriptový framework pro vývoj uživatelského rozhraní - UI. Slouží k tvorbě dynamických webových aplikací, nebo mobilních aplikací. Vývojářem tohoto frameworku je firma Meta (dříve známá jako firma Facebook). UI je děleno na menší komponenty, které mají svoji funkcionalitu a dají se mezi sebou jednoduše zaměňovat. Díky tomu není například potřeba načíst celou stránku při přechodu na jinou stránku, ale stačí pouze zaměnit komponenty které se na stránce změnily (např hlavička a patička zůstávají stále načteny, ale obsah stránky se změní). [20]

2.2.5 Next.js

Next.js je JavaScriptový framework stavící na frameworku React.js. Je vytvářen společností Vercel. Tento framework usnadňuje vývojáři práci se stránkami a přidává nespočet výhod nad použitím samotného Reactu.js. Hlavním lákadlem je takzvaný "Server Side Rendering", což znamená, že počáteční kód, který je poslán klientovi, je nejdříve vykreslen na serveru, pak je odeslán v neživé podobě klientovi a až po té, co je stránka načtena, se z něj stává dynamická stránka pomocí React.js. Mezi další výhody patří například automatická optimalizace velikosti obrázků podle toho, jak velké je klient potřebuje, aby se nemusely posílat zbytečně velké a neoptimalizované obrázky. [21]

2.2.6 Node.js

Node.js je backendové prostředí běžící na JavaScriptovém kódu. Výhodou oproti starším programovacím jazykům, které běží na serveru např. PHP, je, že aplikace neustále běží na pozadí. To umožňuje o dost jednodušší práci s časem (např. načasování určitých událostí neboli Cronjobů), nebo práci v reálném čase pomocí WebSocketů. [22]

2.2.7 GraphQL

GraphQL je jazyk pro API rozhraní sloužící pro jednoduchou komunikaci mezi frontendem/aplikací a backendem. Typy dotazů se dělí na 3 skupiny - Query (získání dat ze serveru), Mutation (změna dat na serveru) a Subscription (zasílání dat v reálném čase pomocí WebSocketů). [23]

2.2.8 MongoDB

MongoDB je dokumentová databáze, která je používána na velké objemy dat. Data jsou zde ukládána do dokumentů ve formátu JSON. Databáze je jednoduše škálovatelná mezi více zařízeními v tzv. „Replica Setu“ (data jsou kopírovaná mezi všemi zařízeními, aby bylo možné při výpadku některého zařízení pokračovat dál v chodu aplikace). Replica set umožňuje i další pokročilé funkce. [24]

2.2.9 Prisma

Prisma je ORM framework pro TypeScript. Hlavní částí je Prisma Client, která umožňuje komunikaci s databází. Dále je sem zahrnuta Prisma Migrate, která v případě SQL databáze dokáže zeditovat databázi tak, aby seděla podle schématu. Poslední částí je Prisma Studio, která umožňuje náhled do uložených dat.[25]

2.2.10 Redis

Redis je databáze fungující na principu klíč a hodnota. Pod nějakým klíčovým slovem si můžeme uložit hodnotu a následně si můžeme hodnotu po zadání klíče zobrazit. Databáze funguje primárně v operační paměti počítače, takže je velmi rychá a efektivní. Využívá se zejména pro akceleraci chodu aplikace. Databáze také podporuje další módy, jako například Pub/Sub[26]. Tato funkcionalita je vhodná pro použití s technologií WebSocket, nebo lépe s GraphQL komponentou „Subscription“. [27]

2.2.11 Flutter

Flutter je framework od Googlu pro vytváření nativních, multiplatformních aplikací z jednoho kódu. Jako programovací jazyk je zde použit Dart, který vyvíjí také Google. Flutter cílí primárně na zařízení s operačním systémem Android nebo iOS, ale i na Windows, Linux, MacOS nebo web. Výhodou je možnost vykreslení každého pixelu na obrazovce tak, jak chceme. [28]

2.2.12 Docker

Docker je platforma pro vytváření, správu a běh kontejnerů. Pomocí takzvaných „Dockerfile“ je možné definovat, jak bude výsledný kontejner sestaven a jak bude fungovat. Lze pomocí něj také spouštět samotné kontejnery, sestavené ať už námi, nebo komunitou. Díky němu lze spustit v podstatě jakoukoliv aplikaci pomocí jednoho příkazu. [29]

2.2.13 Kubernetes

Kubernetes je platforma pro automatickou správu kontejnerů, jejich škálovatelnosti nebo automatického nasazení při nové verzi. Platforma Kubernetes je schopna přizpůsobit počet kontejnerů podle aktuální zátěže. Původním výrobcem byl Google, kvůli jejich platformě GCP (Goole Cloud Platform) ale nyní patří pod Cloud Native Computing Foundation. Díky tomu, že je Kubernetes open-source, vznikají podobné verze, které se od původní trochu liší. Příkladem jsou: K3s, Minikube, OpenShift, Mirantis, Amazon EKS a další. [30]

2.2.14 TensorFlow

TensorFlow je platforma pro strojové učení. Slouží pro trénování umělé inteligence a vytváření neuronových sítí. Tyto neuronové sítě jsou pak při přiložení vstupních dat využívány k predikci nových dat. Pro TensorFlow se píše kód primárně v programovacím jazyce Python, ale má i další implementace, například: JavaScript (TensorFlow.js), nebo Java (TensorFlow Lite). [31]

2.2.15 Blender

Blender je open source software pro tvorbu počítačové grafiky. Lze využít na 3D modelování, 2D/3D animace a mnoho dalšího. [32]

2.2.16 Figma

Figma je grafický editor pro vytváření a prototypování grafických návrhů. Primárně se jedná o webový editor, který umožňuje offline funkce, které nabízí desktopová aplikace. Figma se zaměřuje na UX a UI s možností spolupráce v reálném čase. [33]

2.3 Umělá inteligence

Umělá inteligence, v angličtině Artificial Intelligence (zkráceně AI), je věda a inženýrství vytváření inteligentních strojů, zejména počítačových programů.[34] Toto téma je velmi rozsáhlé, a proto jej dělíme do těchto oblastí[35]:

- Rozpoznání řeči
- Zákaznické služby
- Počítačové vidění
- Doporučovací algoritmy
- Automatické obchodování s akciemi

2.3.1 Počítačové vidění

Počítačové vidění umožňuje počítači získávání smysluplných informací z vizuálních vstupů (obrázky, videa). Funguje velmi podobně jako lidské vidění - musí se učit. V tom mají, ale lidé výhodu celoživotního kontextu a tím pádem dokáží určovat více věcí zároveň. [36]

2.3.1.1 Strojové učení

K učení je potřeba velké množství dat - obrázky nebo videa, na kterých jsou objekty, které má specifiká kolekce rozlišovat. Od trénující osoby je očekáván vstup s informacemi, co za objekty je na obrázku, popřípadě i jejich souřadnice. Počítač následně odhaduje pomocí těchto vstupních dat obsah na obrázku. Takovému výstupu říkáme predikce.

Samotný proces učení probíhá tak, že počítač prochází pixely v oblastech a zdokonaluje své rozeznávání do doby, než jsou predikce dostatečně blízko vstupním datům. Tento proces může trvat několik minut i několik měsíců. Záleží na velikosti vstupních dat a výkonu počítače. Pro tyto výpočty je nejefektivnější využít grafickou kartu.

Když jsou predikce dostatečně blízko vstupním datům, je proces učení ukončen a je z něj vygenerován model, který je dále použit pro predikci, není ale nadále zdokonalován.

2.3.2 Frameworky

Vzhledem k velkému počtu oblastí umělé inteligence existuje nespočet možných frameworků implementujících toto téma. Mezi nejznámější patří:

- TensorFlow
- Caffe
- Torch
- Microsoft Cognitive Toolkit

Velké společnosti také implementují některé z výše uvedených frameworků do svých cloudových služeb. Například služba Microsoft Custom Vision podporuje export vytrénovaných modelů do formátu pro TensorFlow.

3 Praktická část

Hra Codeventure má za cíl zaujmout uživatele formou hry a představit mu svět programování a logického myšlení. Hráč se zapojuje do plnění úkolů, kterými prochází v jednotlivých sekvencích hry, a postupně získává odměny, které ho motivují ke splnění cílového úkolu.

Hra je vystavěna jako propojení mobilního zařízení, vetší zobrazovací plochy a kartiček s povely, které hráč využívá k průchodu hrou. Zapojením příkazových kartiček využívá hráč i své kreativity k dosažení určeného cíle, protože ve hře můžeme nalézt úrovně, které mají otevřené řešení. Složitost úrovní se stupňuje.

Při přenosu příkazů z kartiček do hry se používá umělá inteligence, která rozpoznává jednotlivé příkazy, a ty následně přenáší hráči na cílové zařízení. Hráč poté vidí vyhodnocení zvoleného postupu.

V momentě správného sestavení příkazových kartiček hra otevírá hráči další kolo a odmění ho.

Pokud se jedná o nesprávnou sekvenci, hra hráče upozorní na chybné řešení a pokus se musí opakovat za použití jiné kombinace kartiček.

Na konci správně vyřešené úrovně je hráč hodnocen i za nejefektivnější způsob průchodu hrou, aby měl motivaci se dále zlepšovat a rozvíjet své vnímání logických příkazů směřujících k co nejlepšímu výsledku.

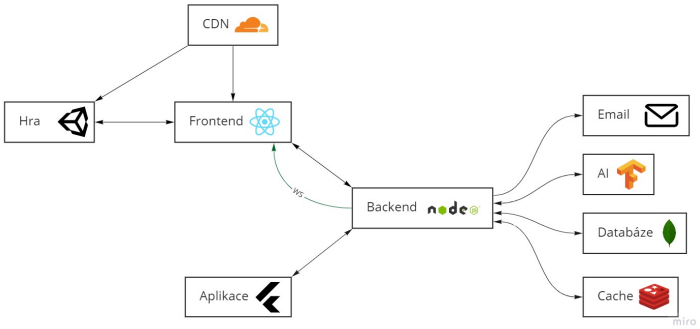
3.1 Realizace

3.1.1 Architektura

Cílem práce je vytvořit komplexní aplikaci, která využívá zařízení s větší obrazovkou jako prostředí pro běh hry, tzv. frontend, a mobilní telefon jako oddělený ovládací prvek (kontrolér).

Z tohoto rozdělení úkolů pak vychází základní nároky na architekturu systému skládající se z kontrolérové části (mobilní aplikace), zobrazovacího frontendu (webová aplikace) a ser-

verového backendu, který musí zastat většinu funkcionality – především tedy autentizaci uživatelských zařízení (frontend i kontrolér) a jejich spárování, komunikaci s nimi, analýzu zaslaného obrazu, vykonávání analyzovaného uživatelského kódu a zprostředkování výsledků pro frontend. Samozřejmými součástmi a podsystémy jsou pak databázová část, emailová komunikace či cachování (využívání dočasné vyrovnávací paměti). Architekturu názorně shrnuje schéma na obr. 3.1



Obrázek 3.1: Architektura aplikace.

3.1.2 Backend

3.1.2.1 Využité technologie

Jako stavební kámen pro backendovou část je použit Node.js s frameworkem Express.js. Na samotném Expressu běží GraphQL, které se stará o dotazy. Dále je zde také zaregistrován zpracovatel dotazů pro GraphQL Subscription, který zodpovídá za komunikaci v reálném čase.

Při použití jednoho serveru by bylo možné provozovat GraphQL Subscription pouze pomocí paměti tohoto jednoho serveru. Pokud bychom však chtěli rozdělit zátěž mezi více serverů, hrozilo by riziko, že dojde k desynchronizaci stavu kontrolérové části a frontendové části (například: Frontend si vytvořil komunikaci mezi sebou a serverem číslo 1. Další dotaz z kontroléru přišel na server 2, server tomuto zařízení odpověděl zpět bez problému, ale nebyl schopen oznámit frontendu o tom, že se něco stalo, protože tyto dva servery o sobě navzájem neví). Proto je potřeba využít funkci Redisu s názvem Pub/Sub[26], která oznámí změnu stavu všem podřízeným serverům.

Jako databáze byla zvolena databáze MongoDB, a to kvůli možnosti zpracovávat velké množství dat. Pro propojení backendu a databáze byla použita Prisma. Na dotazy, které by se často opakovaly, by nebylo efektivní se stále dokola dotazovat databáze, proto se určité, často opakované dotazy ukládají do cache paměti Redis.

3.1.2.2 Autorizace

Pro přístup k většině dotazů je potřeba být přihlášen. Uživatelé se mohou registrovat pomocí GraphQL mutace „register“ a přihlašovat pomocí „login“. Při registraci jsou uživateli zkontrolována vstupní data, je vytvořen dokument s hodnotami: „email“, „first name“, „last name“ a „password“ (heslo je zaheshováno pomocí bcrypt[37]). Po vytvoření je vygenerována session a následně vrácen hash s ukrytým ID této session a jménem uživatele. Přihlášení potom funguje velmi podobně jako registrace, ale místo vytváření nového uživatele je vyhledán uživatel se stejným emailem a přiložené heslo je zkontrolováno na základě hodnoty uložené v databázi.

K přístupu k dotazům, ke kterým je potřeba být přihlášen, je potřeba zaslat v hlavičce dotazu vlastnost s názvem „authorization“ s hodnotou „Bearer TOKEN“.

Pokud se chce uživatel odhlásit, je třeba spustit mutaci „logout“, která smaže aktuální session z databáze aby se s ním nebylo možné dálé hlásit.

Při ztrátě hesla je možné spustit mutaci „sendResetPassword“ s parametrem email. Pokud v databázi existuje účet se zadaným emailem, tak je na něj zaslán email s odkazem obsahující resetovací token. Pomocí GraphQL mutace „resetPassword“ je při přiložení tohoto tokenu do 10 minut možné heslo změnit.

Kvůli bezpečnosti je také nutné mít možnost uživatelský účet smazat. Pro účel smazání účtu slouží GraphQL mutace „deleteAccount“, při které musí být uživatel přihlášen a musí zadat aktuální heslo. Při této akci se nenávratně mažou všechna uživatelská data.

3.1.2.3 Ovládání hry

Jako jádro pro komunikaci mezi telefonem a hrou slouží GraphQL subscription s názvem „unityCommunication“. Jako argument je potřeba token s přihlášením, protože při navazování WebSocketu se nezasílá hlavička dotazu.

Pro získání ostrovů a úrovní slouží GraphQL query „islands“ a „levels“. Obě tyto query obsahují číslo ostrovu/levelu, data pro samotnou hru nebo mobilní telefon a úroveň obsahuje navíc počet hvězd, pokud již hráč tento level hrál.

K nastavení aktuálního ostrova nebo úrovně se používají GraphQL mutace „setUnityIsland“ a „setUnityLevel“. Obě nic nevrací a slouží k zasílání příkazů pro hru v prohlížeči.

Když hra potřebuje načíst ostrovy, je potřeba spustit GraphQL mutaci „getUnityIslands“. Tato mutace nic nevrátí, ale ihned posílá hráči pomocí GraphQL subscription data pro vykreslení.

Pro nastavení rycholosti přehrávání výsledné animace slouží Graphql mutace „setSpeed“, pomocí které se dá nastavit rychlost v rozmezí mezi 1x až 10x.

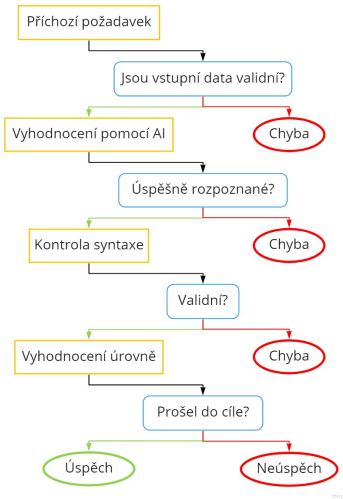
3.1.2.4 Přihlášení pomocí QR kódu

Aby uživatel nemusel psát heslo na chytré televizi nebo na zařízení bez klávesnice, nabízí server přihlášení pomocí QR kódu. První musí začít poslouchat zařízení, na kterém se chce uživatel přihlásit pomocí GraphQL subscription s názvem „qrLogin“. Tato subscription přijímá jako argument řetězec, který je pak použit pro přihlášení. Potom lze z již přihlášeného zařízení sputit GraphQL mutaci „qrLogin“ se stejným argumentem jako u subscription.

3.1.2.5 Vyhodnocování úrovně

Ještě před zobrazením zadání je potřeba požádat server o čas začátku řešení, aby bylo možné na konci zjistit, jak dlouho hráči trvalo vyřešit úroveň. K tomuto účelu slouží GraphQL query „getTime“.

Samotné vyhodnocování úrovní probíhá v několika krocích viz obrázek 3.2. Tyto kroky jsou rozepsané v následujících podkapitolách.



Obrázek 3.2: Proces vyhodnocování úrovně.

3.1.2.5.1 Validace vstupů

Před začátkem vyhodnocování je potřeba zkotrolovat, zda jsou všechny údaje validní. Backend pro tento krok očekává následující parametry: číslo ostrova a úrovně, čas, ve kterém se začínalo řešit, a fotografie kartiček. Pro zkontrolování integrity času je použit hash, kterému musí sedět kontrolní součet. Kdyby ho uživatel nějak změnil, nebyl by potom tento čas validní.

3.1.2.5.2 Vyhodnocení pomocí AI

Po zvalidování vstupů je potřeba vyhodnotit data na obrázku. Pro tento účel slouží samostatný vyhodnocovací backend, na kterém běží Express.js a TensorFlow.js. Na tomto backendu je uložen model vytrénovaný pomocí služby Microsoft Custom Vision. Tento backend není veřejně přístupný a komunikovat s ním může pouze hlavní backend.

Po přijetí obrázku jej hlavní backend zasílá pomocí API rozhraní backendu vyhodnocovacímu. Po úspěšném vyhodnocení jsou vráceny informace o tom, jak jsou kartičky průměrně velké, a pole s údaji kde a s jakou přesností se konkrétní kartička nachází.

3.1.2.5.3 Kontrola syntaxe a vyhodnocení úrovně

Po vyhodnocení obrázku je provedena kontrola syntaxe a vyhodnocení úrovně.

Nejprve je potřeba zkontrolovat, zda rozpoznané kartičky obsahují „Start“ a „Konec“. Dále je vypočítána přímka mezi dvěma body a jsou získány kartičky, které na této přímce leží (pomocí průměrné velikosti kartiček si lze vypočítat toleranci od přímky). Pak jsou pomocí pravého úhlu dopočítány přímky pro tyto kartičky a jsou k nim přiřazeny rozšiřující kartičky.

Když se někde vyskytne synkatická chyba (chybí kartička „Start“ nebo „Konec“, kartička je na místě, na kterém být nemůže), je hráči oznámeno, co udělal špatně a pokus je vyhodnocen jako neúspěšný.

Poslední částí je vyhodnocení úrovně. Pro tuto část je potřeba získat z databáze data o tom, jak úroveň vypadá. Server prochází úroveň krok po kroku a z toho zjišťuje, jak si hráč vede. Nakonec server odesílá data o výsledku zpět do kontroléru a na frontend, kde se hráč dozví celkové hodnocení.

3.1.2.6 Nasazení na server

Nasazení na server je klíčový krok, který určuje zda je celá hra hratelná. Pro vytvoření vždy identického prostředí je potřeba vytvořit kontejner, který obsahuje pouze data potřebná k běhu služby.

Pro vytvoření kontejnerů je použit Docker. Pomocí takzvaného „Dockerfile“ je zadefinován postup jak backend sestavit, který když následně sestavíme, dostaneme balík, který je možné kdekoliv sputit. Tento proces je vykonán jak u hlavního backendu, tak i u vyhodnocovacího backendu.

K následnému nasazení na server je použita platform Kubernetes, konkrétně K3s, která se stará o kontejnery, zajišťuje síťovou komunikaci a rozprostívá zátěž mezi servery.

3.1.3 Web

Web neboli frontend je hlavní částí, kterou uživatel vidí. Jeho úkoly jsou: zobrazení informací, správa uživatelského účtu a zobrazování hry. Klíčové prvky jsou rozepsané v následujících podkapitolách.

3.1.3.1 Prototyp

Celkový Frontend webové aplikace je navržen a řádně otestován softwarem „Figma“. Prvním krokem je vytvoření celkového vzhledu a designu aplikace. Rozhraní musí být co nejjednodušší a přehledné, jelikož je cíleno na mladší děti. Po navržení následovalo několik testů UX/UI. „Figma“ velice usnadní vývoj aplikace, protože jednotlivé části webu jsou navržené a celkové chování webu je nasimulované díky využitému prototypování, které „Figma“ nabízí. Z návrhu vychází kompletní vzhled a fungování webové aplikace.

3.1.3.2 Statická část

Web je postaven na frameworku Next.js, který používá pro zobrazování React.js. Cílem bylo vytvořit web co nejrychlejší a nejspolehlivější.

Aby byl web co nejrychlejší, je potřeba, aby všechny části byly předem vygenerované a aby nebylo nuté zbytečně čekat při dotazu. Next.js v základním stavu počítá s tím, že obrázky jsou optimalizované při času jejich dotazu. Tato funkce je užitečná na stránkách s proměnlivým obsahem, nikoliv na staticky generovaných stránkách. Proto jsme aplikovali trochu netradiční řešení, a to optimalizaci obrázků při generování webu. Díky tomu není potřeba žádný speciální server a je potřeba jen všechny soubory někam umístit.

Pro co nejrychlejší a nespolehlivější běh webu je potřeba umístit jej na takový server, který bude schopen rychle odpovídat velkému množství lidí. Pro tento účel byl vybrán hosting Cloudflare Pages[38] s jejich vlastní CDN[39].

3.1.3.3 Správá účtu

Důležitou částí webu je přihlašovací systém a následná správa účtu. Pro přihlášení je možné použít email a heslo (pomocí GraphQL mutace „login“) nebo QR kód, který se dá naskenovat pomocí mobilní aplikace (GraphQL subscription „qrLogin“ a stejnojmenná GraphQL mutace). Dále je zde možnost zaregistrovat se (mutace „register“) a resetovat heslo pomocí zaslání emailu (mutace „sendResetPassword“).

Když je uživatel přihlášen, je možné odhlásit se (mutace „logout“) nebo aktuální účet smazat (mutace „deleteAccount“).

3.1.3.4 Hra

Hra je vytvořena pomocí herního enginu Unity. Aby bylo možné ji spustit v prohlížeči, je zde použita možnost exportu do webového prostředí zvaná „WebGL“. Pro následnou integraci Unity do webového prostředí slouží knihovna „react-unity-webgl“[40], která poskytuje integraci kódu v Unity s kódem v Reactu.

3.1.3.4.1 Komunikace

Klíčovou funkcí hry je komunikace mezi backendem a hrou tak, aby bylo možné zobrazovat úkony hráče. Tuto práci výrazně zlehčuje již zmíněná knihovna „react-unity-webgl“[40]. Díky této knihovně není třeba komunikovat s backendem dalším způsobem a je možné použít již existující spojení mezi frontendem a backendem.

Komunikace pak může probíhat oběma směry. Směr „hra na backend“ je použit pouze jednou, a to při načtení - když hra potřebuje vykreslit ostrovy. Této akce je docíleno pomocí připravené funkce v Reactu, která prostřednictvím GraphQL mutace „getUnityIslands“ kontaktuje server, aby zaslal zpět nové ostrovy. Tuto funkci lze použít pomocí Unity metody „GetIslands“.

Druhý směr komunikace je častější, a to „backend na hru“. Pro tento směr je potřeba nejdříve navázat spojení pomocí GraphQL subscription „unityCommunication“. Backend má pak možnost provést jakoukoliv veřejnou metodu umístěnou v komunikačním skriptu. Jako argument může přiložit data, která chce předat. Tato data jsou ve formátu JSON a je třeba je po přijetí deserializovat. Používá se například pro přejetí kamery na jiný ostrov, nebo nastavení objektů do scény se zadáním.

3.1.3.4.2 Načítání modelů

Aby byla hra co nejmenší a aby se nenačítala velmi dlouho, je potřeba rozdělit hru do více částí a následně tyto části načítat podle potřeby hráče. Pro tento účel slouží balíčkový systém zvaný „AssetBundle“[41]. V následných balíčcích jsou obsaženy modely a animace k jednotlivým ostrovům, které se načítají podle toho, co uživatel zvolí na kontroléru.

3.1.3.4.3 Animace

Poslední důležitou částí jsou animace. Animace jsou přidávány na modely pomocí koster. Díky tomu je možné použít jeden model s více animacemi. Připojené animace jsou pak spouštěny pomocí koprogramů, které si navzájem předávají informace.

3.1.4 Mobilní aplikace

Mobilní aplikace je největší část, se kterou uživatel interaguje. Uživatel přes tuto část ovládá veškeré chování hry (hru jinak nejde ovládat), vybírá úrovně a skenuje svoje řešení pomocí fotoaparátu. Naprogramovaná je pomocí programovacího jazyku Dart s frameworkem Flutter.

3.1.4.1 Autorizace

Autorizace se jako u webu provádí pomocí GraphQL mutace „login“. Tento token je pak uložen v úložišti telefonu zvaném „SharedPreferences“[42] nebo na zařízeních s iOS v „NSUserDefaults“[43]. Pomocí tohoto tokenu je zařízení autentifikováno vůči serveru.

3.1.4.2 Skenování

Pro focení slouží knihovna „image_picker“[44]. Tato knihovna využívá hlavní aplikaci na focení od výrobce, díky čemuž mají fotografie vysokou kvalitu.

Nejdříve je potřeba, aby uživatel zvolil ostrov a úroveň. Po odstartování pokusu se uživateli otevírá aplikace s fotoaparátem a je potřeba vyfotit své řešení úrovně. Následovně je fotografie z kapacitních důvodů zmenšena. Takto zmenšená fotografie je zaslána na server s potřebnými parametry mutací „levelResult“. Server tento dotaz během pár vteřin zpracuje a zasílá hráči celkové hodnocení.

3.2 Bezpečnost

Zabezpečení je důležitou složkou každé webové aplikace. Kvůli rozlehlosti tohoto projektu je potřeba, aby byla každá část zabezpečená ve všech ohledech.

Nejvíce zranitelná je část webová. Nejvýznamnější bezpečností rizika zaznamenává nadace OWASP[45] s jejich seznamem „Top 10 Web Application Security Risks“[46]. V seznamu je 10 nejčastějších bezpečnostích rizik v posledních letech.

Backendová i frontendová část má většinu těchto chyb odlazených a neměly by být zneužitelné. Zde jsou konkrétní příklady některých bodů:

- **Kryptografické selhání**[47]: Všechny části předpokládají komunikaci pomocí šifrovaného protokolu (HTTPS). Certifikáty jsou podepsané od kryptografických autorit - Let's Encrypt a Cloudflare. Hesla jsou v databázi hashovaná pomocí šifrovací funkce „bcrypt“[37].
- **Injection**[48]: Rozhraní GraphQL validuje integritu datových typů, při práci s textem je tento text zkontrolován pomocí knihovny „validator“[49] pro správný formát. ORM framework Prisma vytváří bezpečné dotazy, aby nebylo možné vytvořit NoSQL inject.
- **Zranitelné a zastaralé komponenty**[50]: Všechny balíčky, které jsou použity, jsou z oficiálních zdrojů (NPM, pub.dev). Balíček musí splňovat alespoň jednu z těchto specifikací: je na dané platformě populární (vysoký počet stažení), má bezproblémový zdrojový kód (musí být manuálně prověřeno) nebo vlastní balíček. Bezproblémovost balíčků automaticky kontroluje GitHub Dependabot, který monitoruje použité balíčky a při výskytu problému automaticky vytváří opravy.

3.3 3D Modely

Každý model je složen ze tří základních složek, a to „vertices“, „edges“ a „faces“, které definují celkový tvar a vzhled. Pro tvorbu modelů do hry je využit „Low Poly“ [51] styl, díky čemuž mají výsledné objekty menší počet polygonů a jsou více optimalizované. Pro tvorbu objektů byl využit software Blender, který je ideální volbou pro tvorbu 3D grafických podkladů pro hru. Herní modely jsou využity v Unity a mobilní aplikaci.

3.3.1 Box modeling

„Box modeling“ [52] je technika modelování, při které jsou využity pouze základní objekty (krychle, koule, atd.). Tento objekt je následně použit k dosažení finálního tvaru objektu dle připraveného návrhu. Celkový cyklus je opakován s každým novým prvkem a jednotlivé modely jsou poté kombinovány k dosažení cílového tvaru.

3.3.2 Low poly

Pro tvorbu jednotlivých herních modelů je využit styl „Low poly“ [51], který nabízí vysokou optimalizaci pro herní využití. „Low poly“ modely vypadají jednoduché, přičemž představují základní tvar představované věci.

3.3.3 Jednotlivé herní modely

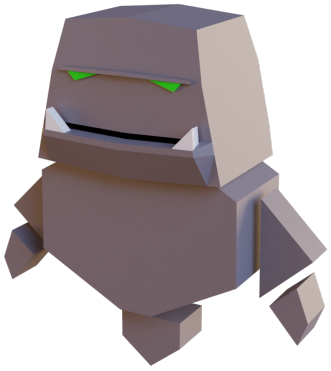
Na začátku tvoření modelů bylo hlavní vybrat časoprostor, kam celou hru zasadit. Herní modely jsou vymodelovány v prostředí příběhu o magickém světě a kouzelníkovi, který zachraňuje svoji planetu.

3.3.3.1 Postavy

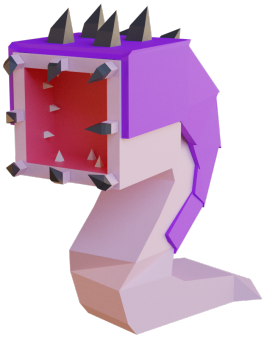
3.3.3.1.1 Hlavní postava Model hlavní postavy je klíčový, protože představuje hráče a jeho postup hrou doprovází na každém kroku. Jak můžeme vidět na obr. 3.3, tak model kouzelníka je hravý a jednoduchý. K modelu je vytvořena také sada animací pro jednotlivé pohybové operace ve hře.

Obrázek 3.3: Model hlavní postavy.

3.3.3.1.2 Nepřátelé Každý ostrov má sadu nepřátel, které hráč může nalézt v jednotlivých úrovních. Nepřátelé mají za úkol hráči ztížit průchod úrovní. Jednotlivé nepřátelské jednotky mají sadu animací. Modely nepřátel můžeme vidět na obr. 3.4, nebo obr. 3.5.



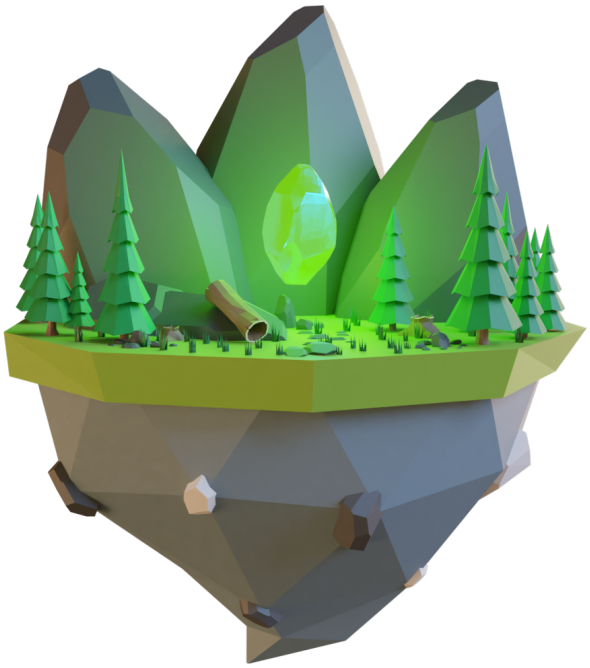
Obrázek 3.4: Model nepřítele - Golem.



Obrázek 3.5: Model nepřítele - Červ.

3.3.3.2 Ostrovy

Jednotlivé modely ostrovů představují přírodní elementy. Dle jednotlivých elementů jsou zbarvené a doplněné o modely, které jsou ve spojitosti s tímto elementem vyobrazeny. Modely ostrovů jsou využity v mobilni aplikaci, jako je vidět na obr. 3.7. Dominantou každého ostrova je krystal, který musí hráč získat pro úspěšné splnění. Ukázka zmiňovaného modelu obr. 3.6.



Obrázek 3.6: Model ostrova - Zemní ostrov.

3.3.3.2.1 Úrovně

Jednotlivé úrovně tvoří mřížku, jako je vidět na obr. 3.8. Každá úroveň představuje stejný element jako ostrov, ke kterému patří. Celá hra je umístěna na herních plošinách. Jednotlivé plošiny, které jsou rozmístněny v mřížce úrovně, slouží jako místo, na kterém se můžou nacházet modely herních postav, bariér, cest, či ukazatelů cíle. Prostředí okolo herních plošin je doplněno o modely mateřského ostrova. Úkázka herního prostředí úrovně je vidět na obr. 3.8

3.4 Popis hry

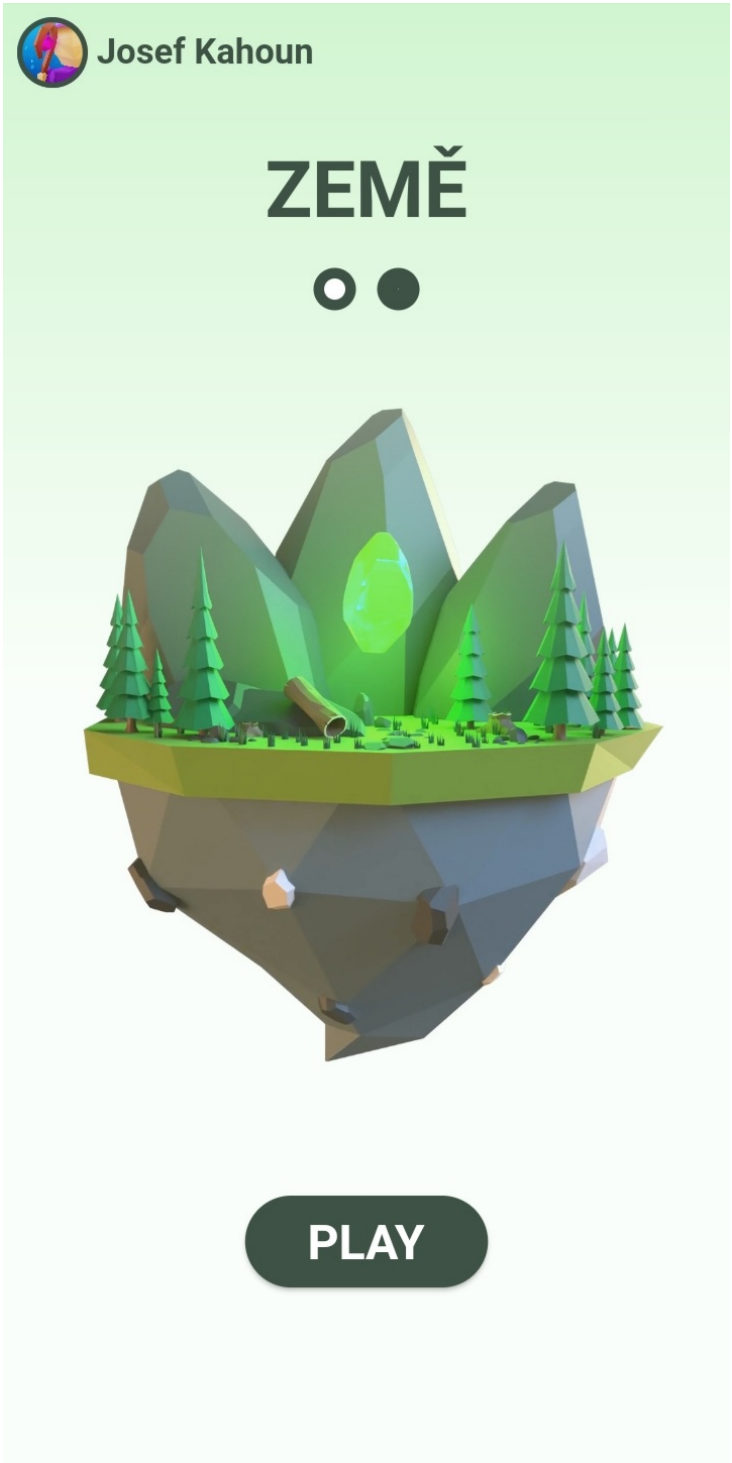
Pro hraní hry je potřeba mobilní telefon s aplikací Codeventure, herní kartičky a zobrazovací zařízení, na kterém lze spustit webová aplikace. Nejlepší herní požitek hráč získá při velké ploše zobrazovacího zařízení. Hra běží na doméně: `https://www.codeventure.cz/`.

Prvním krokem je vytvoření herního účtu, kde se ukládá jednotlivý postup hráče hrou. Následuje stažení mobilní aplikace, kde se hráč přihlásí již vytvořeným účtem. Mobilní aplikaci najdeme v obchodě Goole Play pod názvem Codeventure, `https://play.google.com/store/apps/details?id=cz.codeventure.codeventure`.

Poté se přesune ve webové aplikaci do hry, kde uvidí první ostrov. Dále hráč využívá pouze mobilní telefon, a to jako ovladač. Na mobilím zařízení si zvolí ostrov a úroveň, kterou chce splnit.

Před spuštěním levelu se hráči ukáže slovní zadání úkolu. Dalším krokem je vyobrazení herního zadání ve webové aplikaci a v mobilní spuštění fotoaparátu pro naskenování kartiček. Hráč musí pro úspěšné splnění úrovně seskládat validní a správnou sekvenci kartiček a poté ji naskenovat. Na obr. 3.9 můžeme vidět herní zadání a naskenovanou karetní sekvenci.

Po vyhodnocení se hráči na mobilním telefonu zobrazí, jestli uspěl, či nikoliv. Jednotlivé úrovně může opakovat, nebo může zvolit jinou úroveň. Hra se snaží hráče nabádat k používání sofistikovanějších řešení a díky tomu jej posouvat.



Obrázek 3.7: Model ostrova v mobilní aplikace.

Obrázek 3.8: Model úrovně.

Obrázek 3.9: Pohled hráče.

4 Závěr

V rámci práce byla navržena, implementována a graficky ztvárněna hra Codeventure zaměřená na rozvoj algoritmických a programovacích dovedností u dětí mladšího školního věku. Projekt obsahuje backendovou část, mobilní aplikaci a webový frontend v roli displeje. Součástí řešení je využití umělé inteligence pro strojové rozpoznání obrazu.

V aktuální době je hra spustitelná na adrese `https://www.codeventure.cz/` s experimentálními úrovněmi a momentálně probíhá testování s dětmi.

Literatura

[1] MONTESSORI, Maria. *Londýnské přednášky.* Praha: Portál, 2019.

[2] PROKEŠ, Josef. *Vývojová psychologie* – učební text. Studijní text. [online]. [cit. 2022-03-20]. Dostupné z `https://www.fi.muni.cz/~qprokes/socka/socka3.html`

[3] HARLEN, Wynne (ed.). ASE *guide to primary science education.* Hatfield: Association for Science Education, 2011.

[4] *Benefits of Gamification* [online]. [cit. 2022-03-22]. Dostupné z `https://ssec.si.edu/stemvisions-blog/5-benefits-gamification`

[5] *Cache* [online]. [cit. 2022-01-22]. Dostupné z `https://www.businessinsider.com/what-is-cache`

[6] *CDN* [online]. [cit. 2022-02-12]. Dostupné z `https://www.cloudflare.com/learning/cdn/what-is-a-cdn/`

[7] *Framework* [online]. [cit. 2022-01-22]. Dostupné z `https://codeinstitute.net/global/blog/what-is-a-framework/`

[8] *JSON* [online]. [cit. 2022-01-22]. Dostupné z `https://www.json.org/json-en.html`

[9] *Kontejner* [online]. [cit. 2022-01-10]. Dostupné z `https://www.docker.com/resources/what-container/`

[10] *Coroutine* [online]. [cit. 2022-02-13]. Dostupné z `https://docs.unity3d.com/Manual/Coroutines.html`

[11] *Open Source* [online]. [cit. 2022-01-22]. Dostupné z `https://opensource.com/resources/what-open-source`

[12] *What are ORMs?* [online]. [cit. 2022-03-22]. Dostupné z `https://www.prisma.io/docs/concepts/overview/prisma-in-your-stack/is-prisma-an-orm#what-are-orms`

[13] *Škálování* [online]. [cit. 2022-01-22]. Dostupné z `https://www.cyberlinkasp.com/insights/what-is-software-scalability-and-why-is-it-important/`

[14] *Transpilace* [online]. [cit. 2022-01-22]. Dostupné z `https://www.digitalocean.com/community/tutorials/javascript-transpilers-what-they-are-why-we-need-them`

[15] *WebSockets* [online]. [cit. 2022-03-22]. Dostupné z `https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API`

[16] *Nejjednoduší programovací jazyky na naučení* [online]. [cit. 2021-01-10]. Dostupné z `https://careerkarma.com/blog/easiest-programming-languages-to-learn/#Csharp`

[17] *Unity* [online]. [cit. 2022-03-22]. Dostupné z `https://unity.com/`

[18] *What is JavaScript?* [online]. [cit. 2022-03-22]. Dostupné z `https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript`

[19] *TypeScript* [online]. [cit. 2022-01-10]. Dostupné z `https://www.typescriptlang.org/`

[20] *React.js* [online]. [cit. 2022-03-22]. Dostupné z `https://reactjs.org/`

[21] *What is Next.js* [online]. [cit. 2022-03-22]. Dostupné z `https://nextjs.org/learn/foundations/about-nextjs/what-is-nextjs`

[22] *About Node.js* [online]. [cit. 2022-03-22]. Dostupné z `https://nodejs.org/en/about/`

[23] *What is GraphQL?* [online]. [cit. 2022-03-22]. Dostupné z `https://www.redhat.com/en/topics/api/what-is-graphql`

[24] *What is MongoDB?* [online]. [cit. 2022-03-22]. Dostupné z `https://www.mongodb.com/what-is-mongodb`

[25] *Prisma* [online]. [cit. 2022-01-28]. Dostupné z `https://www.prisma.io/docs/concepts/overview/what-is-prisma`

[26] *PubSub* [online]. [cit. 2022-01-28]. Dostupné z `https://redis.io/topics/pubsub`

[27] *Redis* [online]. [cit. 2022-03-22]. Dostupné z `https://redis.io/`

[28] *Flutter* [online]. [cit. 2022-01-22]. Dostupné z `https://flutter.dev/`

[29] *Docker* [online]. [cit. 2022-03-22]. Dostupné z `https://www.docker.com/`

[30] *What is Kubernetes?* [online]. [cit. 2022-03-22]. Dostupné z `https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/`

[31] *TensorFlow* [online]. [cit. 2022-03-22]. Dostupné z `https://www.tensorflow.org/`

[32] *About Blender* [online]. [cit. 2022-03-22]. Dostupné z `https://www.blender.org/about/`

[33] *Figma* [online]. [cit. 2022-03-22]. Dostupné z `https://webdesign.tutsplus.com/articles/what-is-figma--cms-32272`

[34] *What is AI* [online]. [cit. 2022-03-04]. Dostupné z `https://borghese.di.unimi.it/Teaching/AdvancedIntelligentSystems/Old/IntelligentSystems_2008_2009/Old/IntelligentSystems_2005_2006/Documents/Symbolic/04_McCarthy_whatisai.pdf`

[35] *IBM - What is AI* [online]. [cit. 2022-03-04]. Dostupné z `https://www.ibm.com/cloud/learn/what-is-artificial-intelligence`

[36] *IBM - What is computer vision* [online]. [cit. 2022-03-04]. Dostupné z `https://www.ibm.com/topics/computer-vision`

[37] *bcrypt* [online]. [cit. 2022-01-28]. Dostupné z `https://auth0.com/blog/hashing-in-action-understanding-bcrypt/`

[38] *Cloudflare Pages* [online]. [cit. 2022-02-12]. Dostupné z `https://pages.cloudflare.com/`

[39] *Cloudflare CDN* [online]. [cit. 2022-02-12]. Dostupné z `https://www.cloudflare.com/cdn/`

[40] *react-unity-webgl* [online]. [cit. 2022-02-12]. Dostupné z `https://www.npmjs.com/package/react-unity-webgl`

[41] *AssetBundle* [online]. [cit. 2022-02-13]. Dostupné z `https://docs.unity3d.com/Manual/AssetBundlesIntro.html`

[42] *SharedPreferences* [online]. [cit. 2022-02-13]. Dostupné z `https://developer.android.com/training/data-storage/shared-preferences`

[43] *NSUserDefaults* [online]. [cit. 2022-02-13]. Dostupné z `https://developer.apple.com/documentation/foundation/nsuserdefaults`

[44] *ImagePicker* [online]. [cit. 2022-02-13]. Dostupné z `https://pub.dev/packages/image_picker`

[45] *OWASP* [online]. [cit. 2022-03-03]. Dostupné z `https://owasp.org/`

[46] *OWASP Top 10* [online]. [cit. 2022-03-03]. Dostupné z `https://owasp.org/www-project-top-ten/`

[47] *Kryptografické selhání* [online]. [cit. 2022-03-03]. Dostupné z `https://owasp.org/Top10/A02_2021-Cryptographic_Failures/`

[48] *Injection* [online]. [cit. 2022-03-03]. Dostupné z `https://owasp.org/Top10/A03_2021-Injection/`

[49] *Validator* [online]. [cit. 2022-03-03]. Dostupné z `https://www.npmjs.com/package/validator`

[50] *Vulnerable and Outdated Components* [online]. [cit. 2022-03-03]. Dostupné z `https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/`

[51] *Low Poly* [online]. [cit. 2022-03-06]. Dostupné z `https://blog.displate.com/low-poly-art/`

[52] *BoxModeling* [online]. [cit. 2022-03-06]. Dostupné z `https://it-s.com/what-is-box-modeling/`

Seznam obrázků

Příloha A Karetní set


I
II
III
IV
V
START
KROK
ÚTOK
OTOČ
OTOČ
SEBER
OPAKUJ
KONEC