Střední průmyslová škola elektrotechnická

a Vyšší odborná škola Pardubice

**STŘEDNÍ** **PRŮMYSLOVÁ** **ŠKOLA** **ELEKTROTECHNICKÁ**

**MATURITNÍ** **PRÁCE** **–** **PROGRAMOVÁNÍ**

**Piškvorky**

březen 2021 Jan Najman 4.D

*„Prohlašuji,* *že* *jsem* *maturitní* *práci* *vypracoval(a)* *samostatně* *a* *použil(a)* *jsem* *literárních*

*pramenů,* *informací* *a* *obrázků,* *které* *cituji* *a* *uvádím* *v* *seznamu* *použité* *literatury* *a* *zdrojů*

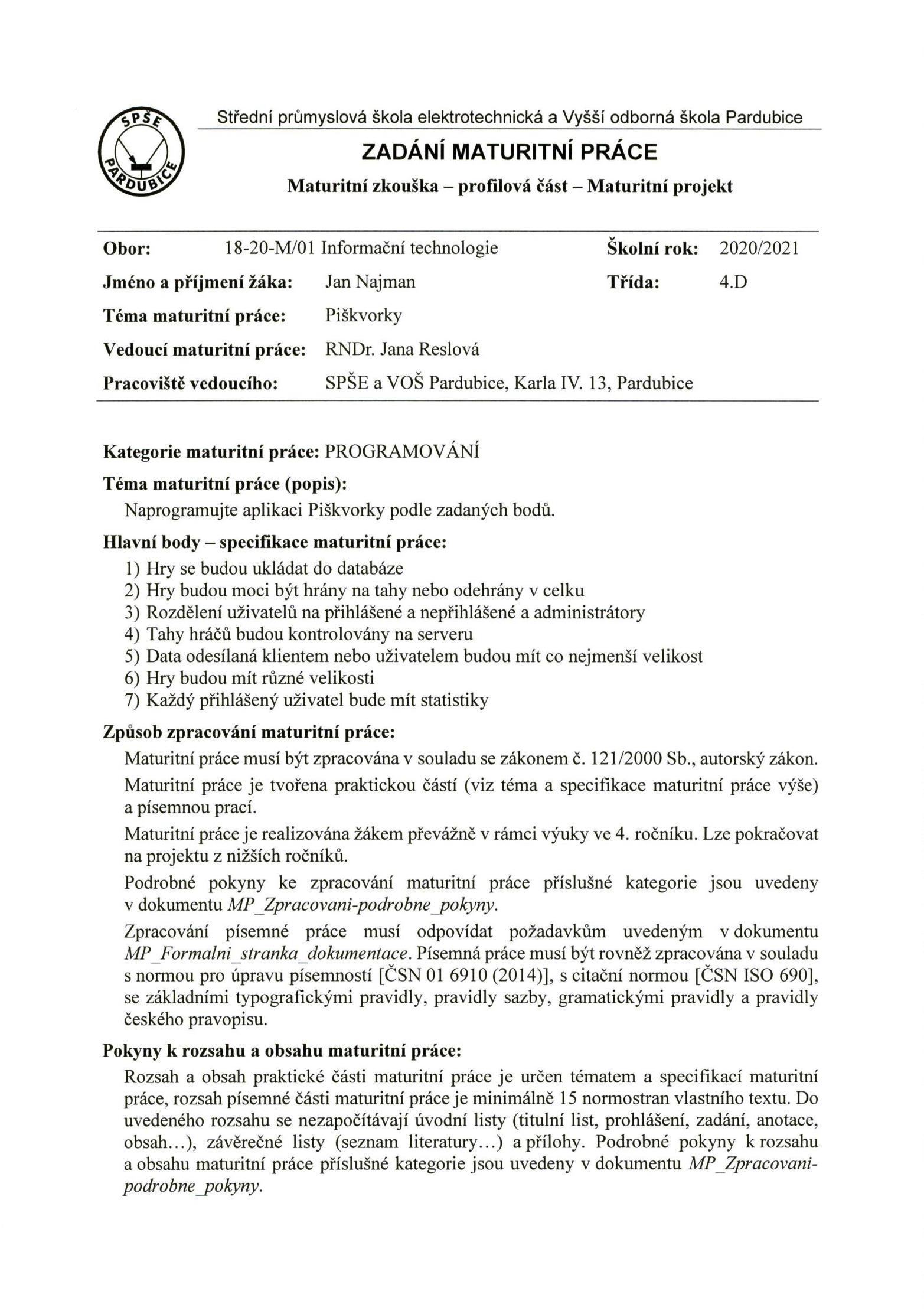
*informací* *a* *v* *seznamu* *použitých* *obrázků* *a* *neporušil* *jsem* *autorská* *práva.*

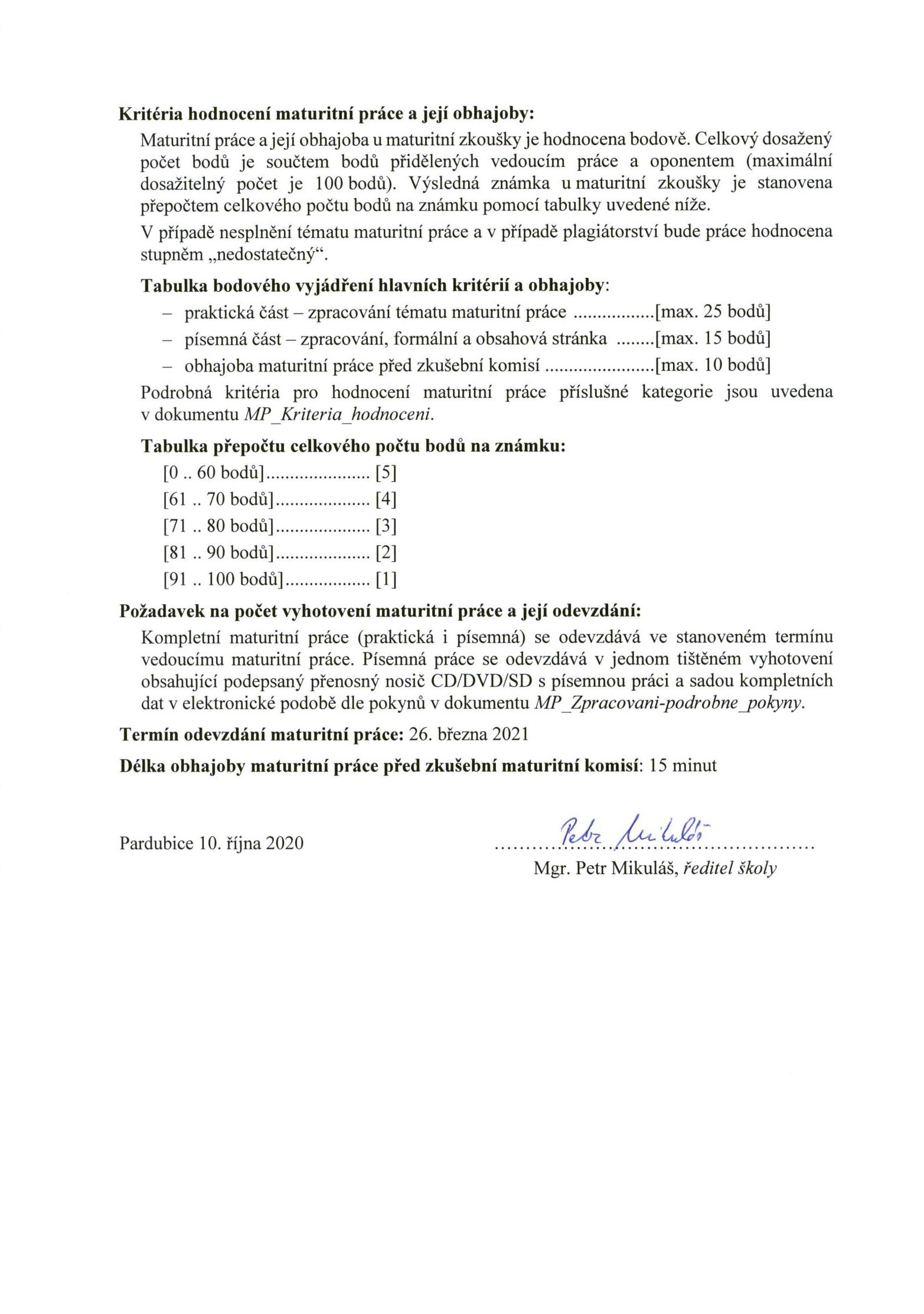
*Souhlasím* *s* *umístěním* *kompletní* *maturitní* *práce* *nebo* *její* *části* *na* *školní* *internetové*

*stránky* *a* *s* *použitím* *jejích* *ukázek* *pro* *výuku.“*

*V* *Pardubicích* *dne* *...........................* *................................................*

*podpis*





**Anotace**

Práce se zabývá rychle rostoucím a čím dál více populárním systémovým jazykem Rust. Ukazuje, jak je tento systémový programovací jazyk vyspělý a jaké má výhody oproti ostatním systémovým jazykům.

Tato práce konkrétně se zabývá programováním RESTful API serveru v programovacím jazyku Rust pomocí frameworku Actix Web a programováním front-endové webové aplikace pomocí frameworku Yew.

Klíčová slova: Rust, API, RESTful, Actix, Actix Web, Web, Server, Aplikace, Front-end, Front-end aplikace, programování, framework, Yew

**Annotation**

This work deals with the rapidly growing and increasingly popular system programming language Rust. It shows how advanced this system programming language is and what its advantages are over other system languages.

This work specifically deals with programming RESTful API server in Rust programming language using the Actix Web framework and programming a front-end web application using framework Yew.

Keywords: Rust, API, RESTful, Actix, Actix Web, Web, Server, Application, Front-end, Front-end applications, programming, framework, Yew

**Obsah**

**Úvod** **9**

**1** **Analýza** **obdobných** **aplikací** **10** 1.1 turtlediary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10 1.1.1 Kladné stránky . . . . . . . . . . . . . . . . . . . . . . . . . . 10

1.1.2 Záporné stránky . . . . . . . . . . . . . . . . . . . . . . . . . 10 1.2 Ultimate Tic Tac Toe . . . . . . . . . . . . . . . . . . . . . . . . . . . 11

1.2.1 Kladné stránky . . . . . . . . . . . . . . . . . . . . . . . . . . 11 1.2.2 Záporné stránky . . . . . . . . . . . . . . . . . . . . . . . . . 11

1.3 gametable . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12 1.3.1 Kladné stránky . . . . . . . . . . . . . . . . . . . . . . . . . . 12 1.3.2 Záporné stránky . . . . . . . . . . . . . . . . . . . . . . . . . 12

**2** **Návrh** **projektu** **13** 2.1 Obecná struktura . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13 2.2 API . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13 2.3 Redis . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13

2.4 Databáze

2.5 Front-end

. . . . . . .

. . . . . . .

. . . . . . . .

. . . . . . . .

. . . . . . . . .

. . . . . . . . .

. . . . . . . . 13

. . . . . . . . 15

2.6 Administrace . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15 2.7 Design a responzivita . . . . . . . . . . . . . . . . . . . . . . . . . . 15

**3** **Zpracování** **praktické** **části** **16** 3.1 Použité technologie . . . . . . . . . . . . . . . . . . . . . . . . . . . 16 3.1.1 Knihovna roles . . . . . . . . . . . . . . . . . . . . . . . . . . 16

3.1.2 Back-end . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16 3.1.3 Front-end . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17

3.2 Databáze

3.3 Back-end

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18

3.3.1 Správa uživatelů . . . . . . . . . . . . . . . . . . . . . . . . . 18 3.3.2 Vytváření žádostí o hru . . . . . . . . . . . . . . . . . . . . . . 19 3.3.3 Vytvoření hratelné hry . . . . . . . . . . . . . . . . . . . . . . 19 3.3.4 Hraní hry . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19

3.4 Front-end . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20 3.4.1 Registrace . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20

3.4.2 Profil . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20 3.4.3 Výpis uživatelů . . . . . . . . . . . . . . . . . . . . . . . . . . 20 3.4.4 Výpis her . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20 3.4.5 Hraní hry . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20 3.4.6 Výpis pozvánek . . . . . . . . . . . . . . . . . . . . . . . . . . 21 3.4.7 Vytváření pozvánky . . . . . . . . . . . . . . . . . . . . . . . . 21 3.4.8 Úprava uživatele . . . . . . . . . . . . . . . . . . . . . . . . . 21

**4** **Manuál** **22** 4.1 Instalace potřebných nástrojů . . . . . . . . . . . . . . . . . . . . . . 22 4.1.1 Kompilace . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22

4.1.2 Použití kontejnerů . . . . . . . . . . . . . . . . . . . . . . . . 22 4.2 Příprava . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22

4.2.1 Kompilace . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22 4.2.2 Použití kontejnerů . . . . . . . . . . . . . . . . . . . . . . . . 24

4.3 Spuštění . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25 4.3.1 Kompilace . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25 4.3.2 Použití kontejnerů . . . . . . . . . . . . . . . . . . . . . . . . 25

**5** **Závěr** **26**

**6** **Seznam** **použité** **literatury** **a** **zdrojů** **informací** **27**

**7** **Seznam** **použitých** **zkratek** **28**

**8** **Seznam** **obrázků,** **tabulek,** **příloh** **29**

**9** **Přílohy** **30**

**Úvod**

Poslední dobou se všichni pokouší optimalizovat své web servery již při jejich programování, kvůli náporu, který by nemusely stíhat. Kdo tak neprovede může toho litovat a snaží se tento problém obejít jinak. Tím že změní jazyk nebo vytvoří repliky své aplikace a dají na ně load-balancer, tuto možnost nakonec musí využít všichni, při velmi vysokém náporu.

Bohužel nepoužívanější jazyky k naprogramování web serveru jsou PHP, JS (Node.js) nebo Python (Flask, Django). Dá se v nich rychle udělat co potřebujete, ale mají spoustu nevýhod, a hlavně všechny tyto jazyky jsou tzv. interpretované jazyky. To znamená, že na pozadí běží nějaký engine (interpreter), který musí zpracovat daný kód za běhu. Jejich největším problémem oproti kompilovaným programovacím jazykům jako jsou Rust, C++, C# je jejich rychlost a možnost zjisti chybu při kompilaci.

Tabulka 1: Porovnání rychlosti jazyků

Jazyk

Rust

C#

Node.js

PHP

Python 3.5

Python 2.7

Pomalejší než C++ (gcc -O2)

7 %

78 %

93 %

596 %

1800 %

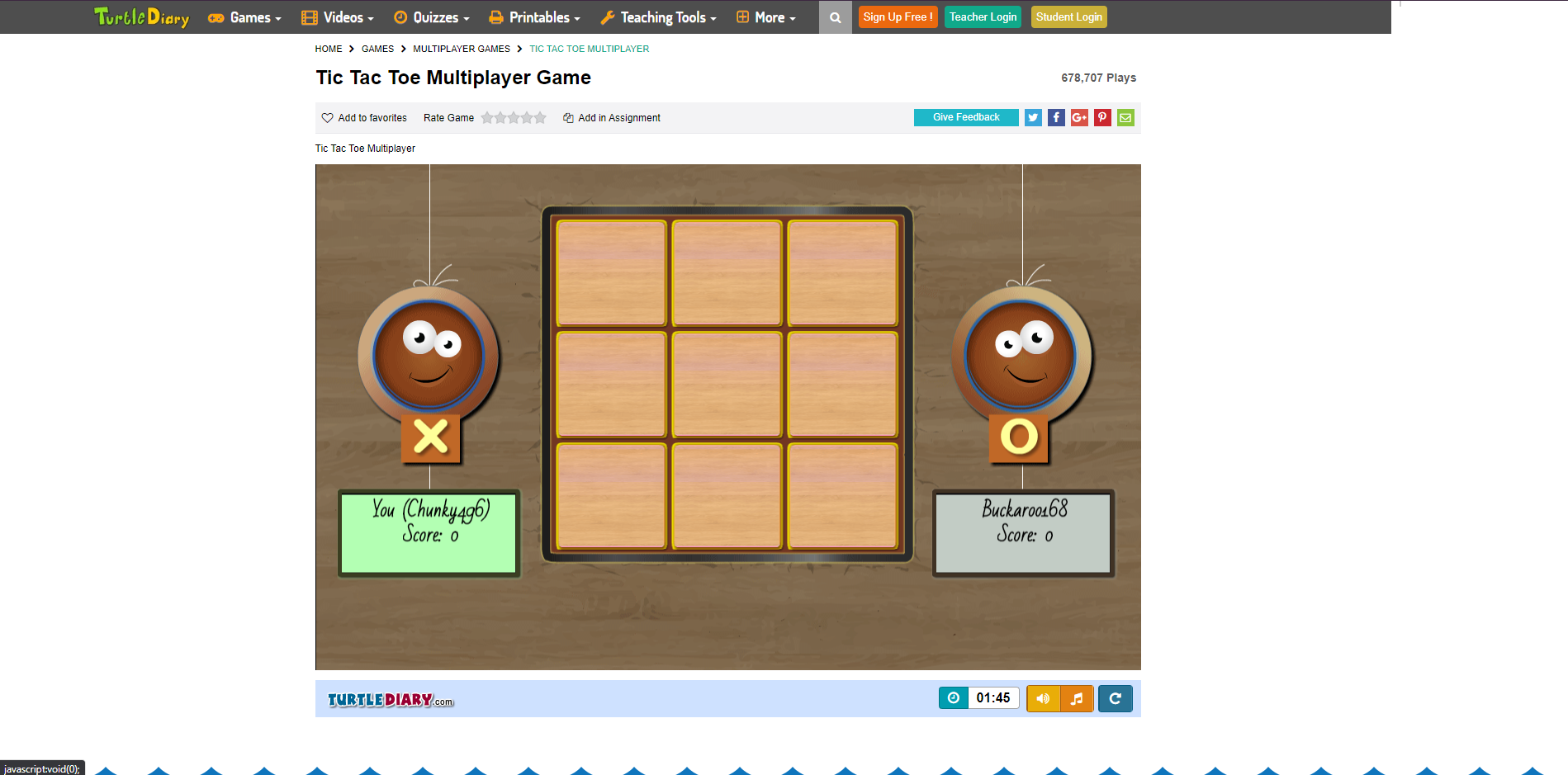
2562 %

Rozhodl jsem se ukázat, že systémový programovací jazyk Rust je na tolik vyspělý, že se nejen zvládne vše, co jiné jazyky, ale i to, že je rychlejší než konkurence.

Vybral jsem si programování RESTful API serveru a front-endové webové aplikace z důvodu, že jsou to dnes nejpoužívanější technologie a mají budoucnost.

Téma Piškvorky jsem si vybral z více důvodů. Mělo by na nich jít perfektně předvést alespoň základy obou z frameworků. Tato hra by měla být každému povědomá a nemusím se zabývat vysvětlováním pravidel.

9



**1** **Analýza** **obdobných** **aplikací**

Analýzu obdobných aplikací je dobré provádět, abyste získali představu, jak má vaše aplikace vypadat. Co chcete, aby uměla a v čem byla lepší něž ostatní aplikace.

**1.1** **turtlediary**

Adresa: <https://www.turtlediary.com/game/tic-tac-toe-multiplayer.html>

Web nabízí možnost hrát piškvorky 3x3 s náhodnými lidmi, nebo s kamarádem.

**1.1.1** **Kladné** **stránky**

• ke hraní není potřeba registrace • vizuální rozhraní hry

**1.1.2** **Záporné** **stránky**

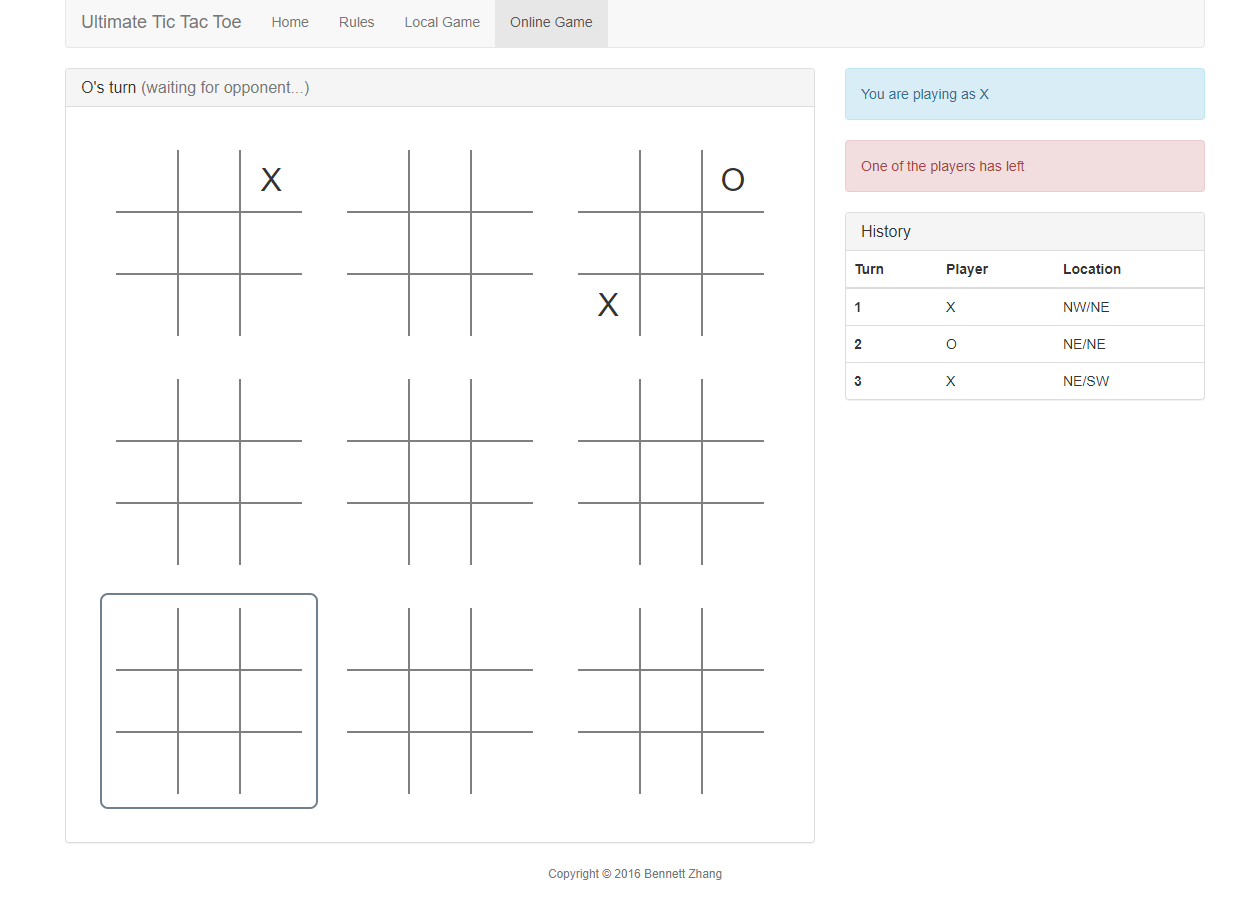
• web není responzivní

• hrací plocha je moc malá oproti zbylému volnému místu • web nemá statistiky nebo výpis nejlepších hráčů

• na mobilním telefonu není hrací plocha vidět celá

Obrázek 1: turtlediary - https://www.turtlediary.com/game/tic-tac-toe-multiplayer.html

10



**1.2** **Ultimate** **Tic** **Tac** **Toe**

Adresa: <https://ultimate-t3.herokuapp.com>

Web nabízí možnost hrát piškvorky 3x3 na více polích s kamarádem přes internet nebo lokálně.

**1.2.1** **Kladné** **stránky**

• ke hraní není potřeba registrace • čistý interface

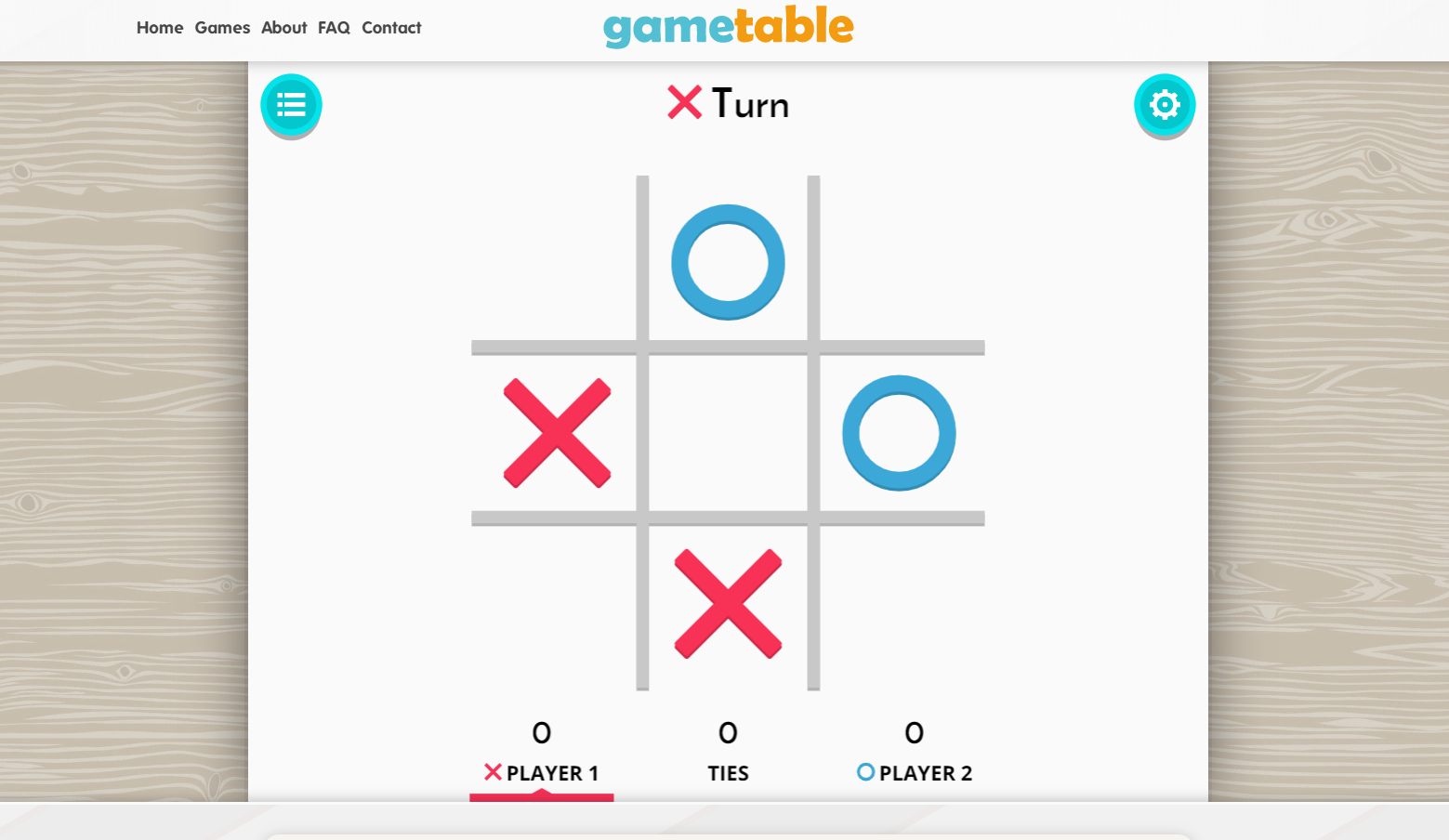
**1.2.2** **Záporné** **stránky**

• vice hracích ploch

• web nemá statistiky nebo výpis nejlepších hráčů

Obrázek 2: Ultimate Tic Tac Toe - https://ultimate-t3.herokuapp.com

11



**1.3** **gametable**

Adresa: <https://gametable.org/games/tic-tac-toe>

Web nabízí možnost hrát piškvorky 3x3 s kamarádem lokálně nebo proti AI.

**1.3.1** **Kladné** **stránky**

• ke hraní není potřeba registrace • čistý interface

• hra má vysvětleny pravidla pod hrací plochou

**1.3.2** **Záporné** **stránky**

• není možnost hrát s někým přes internet

• web nemá statistiky nebo výpis nejlepších hráčů

Obrázek 3: gametable - https://gametable.org/games/tic-tac-toe

12

**2** **Návrh** **projektu**

**2.1** **Obecná** **struktura**

Veškeré požadavky, které nebudou odkazovat na front-end zodpovídá API. API je poté napojené na databázi a redis.

Front-end nemá sám o sobě žádný přístup k databázi nebo redisu.

Celá aplikace je napsaná tak, aby bylo možné ji dát do kontejnerů. S pomocí kontejnerů je možné aplikaci libovolně a jednoduše škálovat.

**2.2** **API**

API spojuje vše dohromady. Poskytuje veškeré informace front-endu a zpracovává veškeré příchozí informace.

Ukládá a kontroluje uživatelské relace v redisu. Ukládá, upravuje a maže údaje v databázi.

**2.3** **Redis**

V redisu jsou ukládány uživatelské relace, aby k nim byl rychlý přístup a jejich ověření. Každá relace má určitou životnost, kterou je možno změnit. V relaci je uložené id uživatele.

**2.4** **Databáze**

Databáze ukládá informace o uživatelích a hrách pro dlouhodobé uložení dat.

Tabulka *users* ukládá informace o uživatelích. S vlastnostmi: • *id* - unikátní id uživatele,

• *nick* - přezdívka uživatele, • *gender* - pohlaví uživatele,

• *hash* - hashované heslo uživatele, • *salt* - sůl pro hashování hesla,

• *email* - unikátní email uživatele,

• *created\_at* - čas vytvoření účtu uživatele,

13

• *description* - popis uživatele.

Tabulka *games* ukládá informace o hrách. S vlastnostmi: • *id* - unikátní id hry,

• *name* - název hry,

• *ended* - udává jestli hra skončila,

• *last\_played* - udává kdo naposled hrál, • *data* - data hry,

• *created\_at* - čas vytvoření hry,

• *moves\_needed* - počet tahů potřebných pro výhru, • *winner* - výherce.

Tabulka *roles* ukládá role. S vlastnostmi: • *id* - unikátní id role,

• *name* - unikátní jméno role.

Tabulka *game\_requests* ukládá informace o pozvánkách na hru. S vlastnostmi • *id* - unikátní id pozvánky,

• *name* - název pozvánky,

• *last\_played* - ukládá kdy naposled hrál (určuje kdo “začal” - při stejném nastavení hry může pokaždé začínat někdo jiný)

• *created\_at* - čas vytvoření pozvánky,

• *moves\_needed* - počet tahů potřebných pro výhru.

Tabulka *roles\_to\_users* spojuje tabulky *roles* a *users*. Přiřazuje uživatelům role. S vlastnostmi:

• *user\_id* - id uživatele, • *role\_id* - id role.

Tabulka*games\_to\_users*spojujetabulky*games*a*users*.Přiřazujehrykuživatelům. S vlastnostmi:

• *game\_id* - id hry,

• *user\_id* - id uživatele.

Tabulka*users\_to\_game\_requests*spojujetabulky*game\_requests*a*users*.Přiřazuje pozvánky k uživatelům. S vlastnostmi:

• *user\_id* - id uživatele,

• *game\_request\_id* - id pozvánky,

14

• *accepted* - udává zda uživatel pozvánku přijal.

Diagram fig. 4

**2.5** **Front-end**

Slouží jako grafické zobrazení dat a druhořadá kontrola dat. Zobrazuje informace o uživatelích a hrách.

**2.6** **Administrace**

Každý uživatel co má roli *Admin* má zvýšená práva. Může upravovat ostatní uživatele, vytvářet pozvánky ve kterých nemusí být, nebo může obsahovat uživatele, kteří nemohou být normálně přidáni do pozvánky. Může také vypnout skoro všechny kontroly, jako je formát jména, hesla, popisu uživatele, atd.

**2.7** **Design** **a** **responzivita**

Design a responzivita je řešená pomocí css knihovny UIkit [(https://getuikit.c](https://getuikit.com/) [om/)](https://getuikit.com/). Několik věcí je přepsáno pro vzhled aplikace. Tyto změny se nachází v souboru *uikit\_addition.css*.

uikit\_addition.css lst. 8

15

**3** **Zpracování** **praktické** **části**

**3.1** **Použité** **technologie**

SkoroceláaplikacejenaprogramovanávjazyceRust,jendatabázesepíševjazyce sql a má několik procedur.

**3.1.1** **Knihovna** **roles**

Mnou vytvořená knihovna, která při kompilaci načte role z databáze a převede je do datového typu enum.

• *quote* - poskytuje makro pro převod datových struktur jazyka Rust na tokeny zdrojového kódu

• *syn* - je knihovna načítaní tokenů zdrojového kódu jazyka Rust do datového typy zdrojového kódu jazyka Rust

• *proc-macro2* - umožňuje definovat makra

• *dotenv* - načte .env soubor do proměnných prostředí • *postgres* - synchronní klient pro databázi PostgreSQL

Zdrojový kód knihovny roles lst. 12

**3.1.2** **Back-end**

Back-end je naprogramován celý v jazyce Rust.

• *structopt* - získává parametry pro program z příkazového řádku, nebo z proměnných prostředí (Environment variables)

• *dotenv* - načte .env soubor do proměnných prostředí • *thiserror* - slouží pro zacházení s chybami

• *actix-web* - hlavní knihovna pro web server

• *actix-redis* - používá se jako back-end pro knihovnu actix-session • *actix-session* - používá se pro zacházení s uživatelskými relacemi • *env\_logger* - slouží jako back-end k výpisu akcí serveru (logging) • *time* - používá se pro práci s časem

• *sqlx* - slouží k komunikaci s databází

• *lazy\_static* - je použit k vyhodnocování proměnných jen jednou a jen při použití • *fancy-regex* - slouží pro kontrolu dat pomocí regexu

16

• *serde* - knihovna k serializaci datových struktur • *serde\_json* - slouží k serializaci do formátu JSON

• *futures-util* - nástroje pro práci s futures (asynchronní procesy) • *futures* - implementace futures knihovny a std::future

• *rand* - používá se pro generování náhodných věcí (čísel, vybírání položky s pole, atd.)

• *argon2rs* - hashuje hesla pomocí algoritmu Argon2 • *log* - používá se k výpisu akcí serveru (logging)

• *serde\_repr* - slouží k serializaci enum data typu

• *bincode* - serializuje datové struktury do formátu bincode

• *uuid* - používá se pro práci s univerzálními unikátními identifikátory (UUID) • *tokio* - slouží jako back-end pro asynchronní procesy

• *roles* - mnou vytvořená knihovna, která při kompilaci načte role z databáze a převede je do datového typu enum

• *actix-cors* - implementace CORS (Cross-origin resource sharing) pravidel pro Actix Web

**3.1.3** **Front-end**

Front-end je naprogramován hlavně v jazyce Rust, ale využívají se tam i jiné jazyky, jako je HTML, CSS a JS.

Rust knihovny:

• *yew* - knihovna pro vytváření více vláknových front-endových webových aplikací s WebAssembly

• *wasm-bindgen* - knihovna usnadňující interakci na vysoké úrovni mezi moduly wasm (WebAssembly) a JavaScriptem

• *yew-router* - směrovací knihovna pro knihovnu yew • *wee\_alloc* - alokátor pro WebAssembly

• *wasm-logger* - slouží jako back-end pro k výpisu akcí do konzole prohlížeče • *log* - používá se k výpisu akcí serveru (logging)

• *roles* - mnou vytvořená knihovna, která při kompilaci načte role z databáze a převede je do datového typu enum

• *lazy\_static* - je použit k vyhodnocování proměnných jen jednou a jen při použití • *fancy-regex* - slouží pro kontrolu dat pomocí regexu

• *serde* - knihovna k serializaci datových struktur

17

• *serde\_json* - slouží k serializaci do formátu JSON • *serde\_repr* - slouží k serializaci enum data typu

• *time* - používá se pro práci s časem

• *bincode* - serializuje datové struktury do formátu bincode

• *strum* - poskytuje sadu maker pro snadnější práci s datovými typy enum a String

CSS a JS knihovny:

• *UIkit* - modulární front-end knihovna pro vývoj rychlých a výkonných webových rozhraní

**3.1.3.1** **Server** Speciálně vytvořený server pro správnou funkci front-endu.

• *structopt* - získává parametry pro program z příkazového řádku, nebo z proměnných prostředí (Environment variables)

• *dotenv* - načte .env soubor do proměnných prostředí • *actix-web* - hlavní knihovna pro web server

• *env\_logger* - slouží jako back-end k výpisu akcí serveru (logging) • *actix-files* - slouží k práci se statickými soubory

• *thiserror* - slouží pro zacházení s chybami

**3.2** **Databáze**

Spoustaakcí,kteréback-endpodnikájsouřešenéskrzprocedůry.Tímtozpůsobem dojde k zjednodušení kódu na back-endu a k provedení akce není potřeba dělat několik dotazů na databázi.

Procedura pro vytvoření pozvánky lst. 9

Procedura pro úpravu uživatele lst. 10

Procedura pro úpravu pozvánky lst. 11

**3.3** **Back-end**

**3.3.1** **Správa** **uživatelů**

Uživatelé jsou umístěni v tabulce *users*. Role jsou uloženy v tabulce *roles* a jsou k uživatelům přiřazovány skrz tabulku *roles\_to\_users*.

18

Pokud má uživatel roli *Admin* tak mohou upravovat kohokoli údaje bez omezení včetně rolí a hesla.

Pokud má uživatel roli *Banned* tak nemůže vytvářet pozvánky, ani nemůže být zahrnut do pozvánky jiným uživatelem.

Úprava uživatele po kontrole dat je poté řízena procedurou lst. 10.

**3.3.2** **Vytváření** **žádostí** **o** **hru**

Přihlášení uživatelé mají možnost vytvářet nové hry s různými parametry.

Nejprve se vytvoří žádost o hru, která se nachází v tabulce *game\_requests*. K dané žádosti na hru se přiřadí uživatelé skrz tabulku *users\_to\_game\_requests*.

Pozvánky nemohou vytvářet uživatelé s rolí *Banned*.

Vytvoření pozvánky po kontrole dat je poté řízeno procedurou lst. 9

**3.3.3** **Vytvoření** **hratelné** **hry**

Po vytvoření žádosti o hru jí musí všichni hráči potvrdit a hra bude vytvořena, nebo někdo z pozvaných hráčů odmítne žádost a žádost o hru bude vymazána.

Jakmile je účast všech hráčů potvrzena, tak se vytvoří nová hra v tabulce *games*, přiřadí se k ní uživatelé skrz tabulku *games\_to\_users* a žádost o hru je poté vymazána.

Úpravu pozvánky po kontrole dat je poté řízena procedurou lst. 11

**3.3.4** **Hraní** **hry**

Hrát můžete jen když jste na tahu a pokud hrané políčko ještě nebylo použito. Vyhraní hry se kontroluje na front-endu, pokud front-end usoudí, že hráč vyhrál tak výhru oznámí back-endu a ten výhru zkontroluje.

Back-end kontroluje, jestli je hráč na tahu, jestli hra neskončila, nebo jestli jeho tah je validní. V případě, že hráč ohlásí výhru, ale server zjistí, že to tak není, tak daný tah zahodí a odpoví chybou.

19

**3.4** **Front-end**

**3.4.1** **Registrace**

Všechna pole jsou kontrolována. Pokud nějaké pole není validní, tak se nepošlou data na back-end.

Back-end data zkontroluje a pokud zjistí, že nejsou validní, tak žádost zahodí a vrátí chybu.

Pro hashování hesla se používá 128 znaková sůl a algoritmus Argon2.

**3.4.2** **Profil**

Zobrazuje informace o uživateli a hry, ve kterých se nachází.

Zobrazuje také počet výher, proher a remíz. Z těchto dat poté vypočítá winrate (výhry / prohry).

Pokud je uživatel na svém profilu, nebo pokud má uživatel roli *Admin*, tak se mu také zobrazí tlačítko na upravení profilu.

**3.4.3** **Výpis** **uživatelů**

Zobrazuje všechny registrované uživatele a pár informací o nich.

Uživatelům s rolí *Admin* se navíc zobrazuje tlačítko upravení profilu.

**3.4.4** **Výpis** **her**

Zobrazuje všechny rozehrané, nebo dohrané hry s jejich hráči a stavem hry.

**3.4.5** **Hraní** **hry**

Hry jsou hrány na síti 30x30.

Uživatelé jsou zobrazováni s jejich symbolem před jménem a za jménem je napsáno, jestli jsou na tahu.

Hrát můžou jen uživatelé, kteří jsou v dané hře, ale dívat se může kdokoli.

20

Tahy uživatelů jsou kontrolovány, jestli jsou validní a jestli nastala výhra nebo remíza.

**3.4.6** **Výpis** **pozvánek**

Zobrazuje název pozvánky (později název hry), počet tahů k vítězství a id pozvánky.

Uživatel může pozvánku přijmout, nebo odmítnout.

**3.4.7** **Vytváření** **pozvánky**

Při vytváření pozvánky jsou skoro všechna pole kontrolována.

Uživatelé s rolí *Admin* mají práva na vypnutí skoro všech kontrol.

**3.4.8** **Úprava** **uživatele**

Uživatel může upravovat vše, kromě jeho rolí.

Uživatel s rolí *Admin* může upravovat vše a má možnost vypnout kontrolu, která je vyžadována po ostatních uživatelích.

21

**4** **Manuál**

Aplikace se dá spustit více způsoby. Buď kompilací ze zdrojového kódu, nebo pomocí kontejnerů.

**4.1** **Instalace** **potřebných** **nástrojů**

**4.1.1** **Kompilace**

Pro kompilaci potřebujeme nainstalovat kompilátor jazyka Rust.

Přejdeme na stránku stažení jazyka Rust <https://www.rust-lang.org/tools/install> a stáhneme exe soubor.

Otevřeme a zadáme 1 pro instalaci a stiskneme enter.

Po dokončení instalace všech komponentů nainstalujeme ještě wasm-pack pro kompilaci do WebAssembly.

Přejdeme na stránku stažení nástroje wasm-pack [https://rustwasm.github.io/was](https://rustwasm.github.io/wasm-pack/installer/) [m-pack/installer/](https://rustwasm.github.io/wasm-pack/installer/) a stáhneme exe soubor.

Po stažení jej nainstalujeme.

**4.1.2** **Použití** **kontejnerů**

Pro použití kontejnerů potřebujeme nějaký software, který to umožňuje. Já jsem zvolil docker.

Přejdeme na stránku stažení <https://hub.docker.com/editions/community/docker-ce-desktop-windows/>a stáhneme exe soubor.

Nainstalujeme a vyzkoušíme funkčnost (Hello world! kontejner).

**4.2** **Příprava**

**4.2.1** **Kompilace**

Kompilace není potřeba při použití kontejnerů.

22

Pro kompilaci aplikace je potřeba mít databázi připravenou dopředu. Stačí se přihlásit do administrace databáze a spustit sql příkazy v přiloženém sql souboru.

Zkompilujeme back-end. Vstoupíme do složky backend a spustíme příkaz:

**Výpis** **1** Příkaz pro kompilaci back-endu cargo build --release

Zkompilujeme front-end server. Vstoupíme do složky frontend/server a spustíme příkaz:

**Výpis** **2** Příkaz pro kompilaci front-end serveru cargo build --release

Zkompilujeme front-end. Vstoupíme do složky frontend a spustíme příkaz:

**Výpis** **3** Příkaz pro kompilaci front-endu

build.bat

23

**4.2.2** **Použití** **kontejnerů**

Pro jednoduché použití používám docker-compose.

Pro jednoduchou konfiguraci jsem vytvořil funkční příklad nastavení.

Tento příklad si zkopírujeme a přejmenujeme z docker-compose.yml.example na docker-compose.yml.

Potom si ho upravíme podle vlastních preferencí.

Změníme tyto položky:

**Výpis** **4** Nastavení proměnných prostředí pro kontejnery

*#* *Databáze* POSTGRES\_USER**:** root

POSTGRES\_PASSWORD**:** password

*#* *Back-end* AUTH\_TTL**:** 3600

*#* *Jméno* *uživatele* *databáze* *#* *Heslo* *uživatele* *databáze*

*#* *Jak* *dlouho* *bude* *platit* *uživatelská* *relace*

↪ *v* *sekundách*

AUTH\_KEY**:** keykeykeykeykeykeykeykeykeykeykey *#* *Klíč* *pro* *uživatelské* ↪ *relace,* *musí* *mít* *minimálně* *32* *bajtů* *(32* *znaků)*

DATABASE\_URL**:** postgres://root:password@postgres/maturitni\_projekt *#* ↪ *Změníme* *jméno* *a* *heslo* *na* *to* *které* *jsme* *zvolili* *u* *databáze*

ALLOWED\_ORIGIN**:** http://mp.loc *#* *Adresa* *front-endu* *pro* *správné* *nastavení* ↪ *CORS*

FRONTEND\_DOMAIN**:** mp.loc *#* *Adresa* *front-endu* *bez* *protokolu* *pro* ↪ *přístup* *ke* *cookies* *na* *front-endu*

*#* *Front-end*

API\_DOMAIN**:** http://api.mp.loc *#* *Adresa* *back-endu* *pro* *komunikaci* *mezi* ↪ *front-endem* *a* *back-endem*

24

Poté taky změníme překládání portů, abychom se na aplikaci vůbec dostali.

**Výpis** **5** Nastavení překládání portů pro kontejnery

backend**:** ports**:**

**-** "127.0.5.2:80:80" *#* *Toto* *znamená* *překládej* *port* *80* *na* *adresu* *a* *port* ↪ *127.0.5.2:80*

frontend**:** ports**:**

**-** "127.0.5.1:80:80" *#* *Toto* *znamená* *překládej* *port* *80* *na* *adresu* *a* *port* ↪ *127.0.5.1:80*

Samozřejmě tato aplikace má být postavená například za nginx server. Ale pokud jen testujeme na svém počítači tak můžeme do souboru hosts přidat tyto položky:

**Výpis** **6** Úprava hosts souboru pro kontejnery 127.0.5.1 mp.loc

127.0.5.2 api.mp.loc

**4.3** **Spuštění**

**4.3.1** **Kompilace**

Musíme mít spuštěnou databázi a redis.

1. Spustíme back-end. Stačí jen spustit exe soubor a popřípadě dodat další argumenty.

2. Spustíme front-end server. Stačí jen spustit exe soubor a popřípadě dodat další argumenty.

**4.3.2** **Použití** **kontejnerů**

Spustíme příkaz, který nám stáhne, nastartuje a nakonfiguruje veškeré kontejnery:

**Výpis** **7** Příkaz pro start kontejnerů docker-compose up -d

25

**5** **Závěr**

26

**6** **Seznam** **použité** **literatury** **a** **zdrojů** **informací**

27

**7** **Seznam** **použitých** **zkratek**

Tabulka 2: Seznam použitých zkratek

Zkratka

API

CORS

REST

SPŠE

UUID

VOŠ

Význam

Application Programming Interface

Cross-origin resource sharing

Representational State Transfer

Střední průmyslová škola elektrotechnická

Universally unique identifier

Vysoká odborná škola

28

**8** **Seznam** **obrázků,** **tabulek,** **příloh**

**Seznam** **obrázků**

1 turtlediary - https://www.turtlediary.com/game/tic-tac-toe-multiplayer.html 10 2 Ultimate Tic Tac Toe - https://ultimate-t3.herokuapp.com . . . . . . . . 11 3 gametable - https://gametable.org/games/tic-tac-toe . . . . . . . . . . . 12 4 ER Diagram . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 30

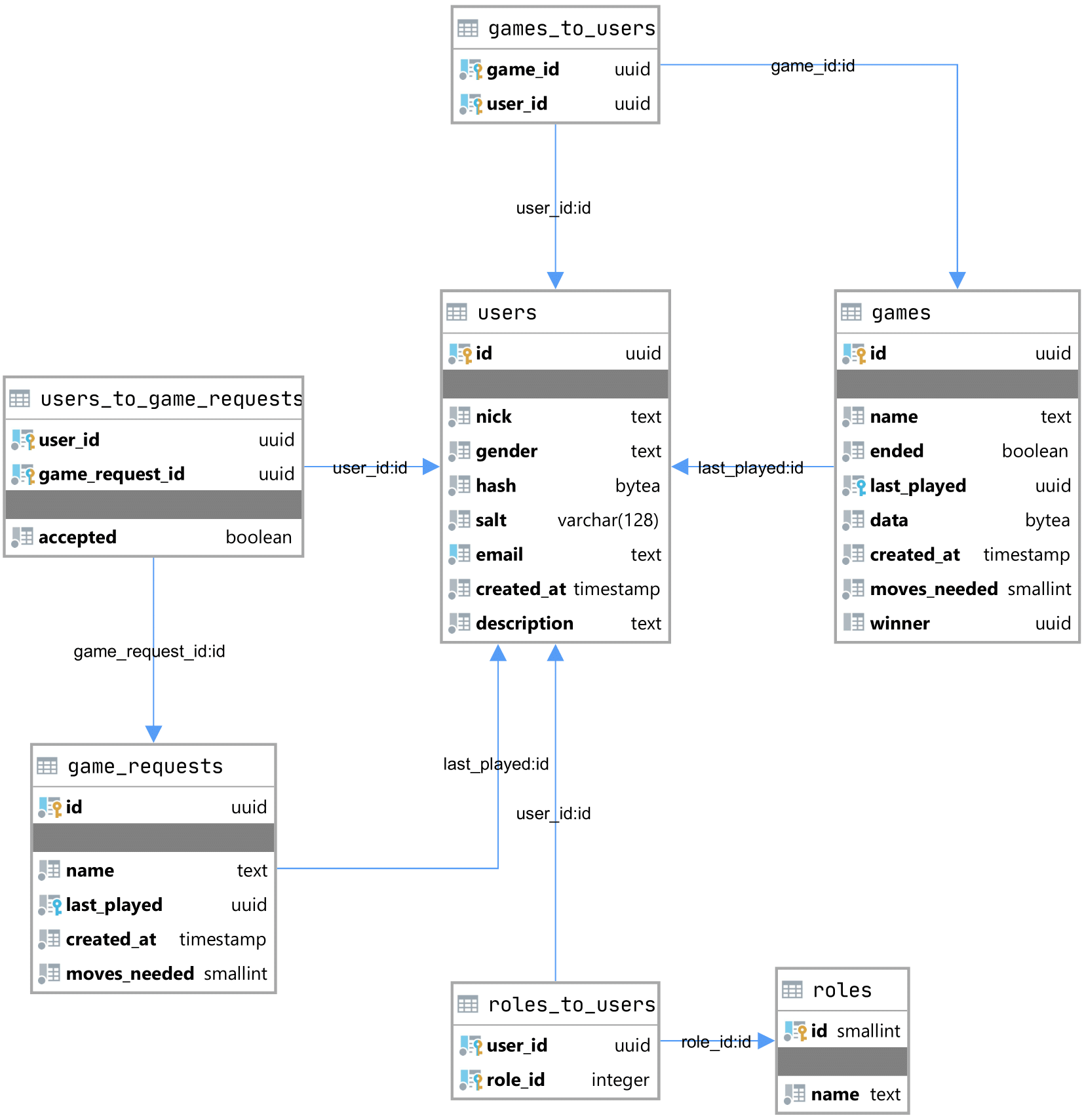
**Seznam** **tabulek**

1 Porovnání rychlosti jazyků . . . . . . . . . . . . . . . . . . . . . . . . 9 2 Seznam použitých zkratek . . . . . . . . . . . . . . . . . . . . . . . . 28

**Seznam** **výpisů**

1 Příkaz pro kompilaci back-endu . . . . . . . . . . . . . . . . . . . . . 23 2 Příkaz pro kompilaci front-end serveru . . . . . . . . . . . . . . . . . 23 3 Příkaz pro kompilaci front-endu . . . . . . . . . . . . . . . . . . . . . 23 4 Nastavení proměnných prostředí pro kontejnery . . . . . . . . . . . . 24 5 Nastavení překládání portů pro kontejnery . . . . . . . . . . . . . . . 25 6 Úprava hosts souboru pro kontejnery . . . . . . . . . . . . . . . . . . 25 7 Příkaz pro start kontejnerů . . . . . . . . . . . . . . . . . . . . . . . 25 8 uikit\_addition.css . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31 9 Procedura pro vytvoření pozvánky . . . . . . . . . . . . . . . . . . . . 32 10 Procedura pro úpravu uživatele . . . . . . . . . . . . . . . . . . . . . 33 11 Procedůra pro úpravu pozvánky . . . . . . . . . . . . . . . . . . . . . 34 12 Zdrojový kód knihovny roles . . . . . . . . . . . . . . . . . . . . . . 35

29



**9** **Přílohy**

Obrázek 4: ER Diagram

30

**Výpis** **8** uikit\_addition.css

1 body {

2 **background-color**: #545454; 3 }

4 .uk-navbar-container.uk-light:not(.uk-navbar-transparent) ↪ ***:not(***.uk-navbar-primary***)*** {

5 **background**: #222; 6 }

7 .uk-dropdown.uk-light { 8 **background**: #222;

9 }

10 .uk-dropdown li {

11 **padding-left**: 5px;

12 **border-left**: 2px solid transparent; 13 }

14 .uk-dropdown li.uk-active { 15 **border-color**: #545454; 16 }

17 #mobile-navbar li {

18 **padding-left**: 5px;

19 **border-left**: 2px solid transparent; 20 }

21 #mobile-navbar li.uk-active { 22 **border-color**: #545454;

23 }

24 .uk-navbar a {

25 **text-decoration**: none 26 }

27 .uk-form-danger {

28 **color**: #f0506e !important;

29 **border-color**: #f0506e !important; 30 }

31 .uk-notification-message { 32 **background**: #222;

33 }

34 body > div {

35 **padding-bottom**: 1px; 36 }

31

**Výpis** **9** Procedura pro vytvoření pozvánky

1 **create** **procedure** new\_game\_request(\_name text, \_last\_played uuid, \_users\_id ↪ uuid[], \_moves\_needed smallint)

2 **language** plpgsql 3 **as**

4 $$

5 declare

6 v\_game\_request\_id uuid; 7 begin

8 insert into game\_requests (name, last\_played, moves\_needed) 9 values (\_name, \_last\_played, \_moves\_needed)

10 returning game\_requests.id into v\_game\_request\_id;

11

12 insert into users\_to\_game\_requests (user\_id, game\_request\_id) 13 select user\_id\_\_, v\_game\_request\_id

14 from unnest(\_users\_id) user\_id\_\_; 15 commit;

16 end; 17 $$;

32

**Výpis** **10** Procedura pro úpravu uživatele

1 **create** **procedure** update\_user(\_id uuid, \_nick text **DEFAULT** **NULL**::text,

↪ \_gender text **DEFAULT** **NULL**::text, \_email text **DEFAULT** **NULL**::text, \_hash ↪ bytea **DEFAULT** **NULL**::bytea, \_salt character **varying** **DEFAULT**

↪ **NULL**::character **varying**, \_roles integer[] **DEFAULT** **NULL**::integer[], ↪ \_description text **DEFAULT** **NULL**::text)

2 **language** plpgsql 3 **as**

4 $$

5 begin

6 if \_nick is not null then

7 update users set nick = $2 where id = $1; 8 end if;

9

10 if \_gender is not null then

11 update users set gender = $3 where id = $1; 12 end if;

13

14 if \_email is not null then

15 update users set email = $4 where id = $1; 16 end if;

17

18 if \_hash is not null then

19 update users set hash = $5 where id = $1; 20 end if;

21

22 if \_salt is not null then

23 update users set salt = $6 where id = $1; 24 end if;

25

26 if \_description is not null then

27 update users set description = \_description where id = \_id; 28 end if;

29

30 if \_roles is not null then

31 delete from roles\_to\_users where user\_id = \_id;

32 insert into roles\_to\_users (user\_id, role\_id) select \_id, role\_id\_\_ ↪ FROM unnest(\_roles) role\_id\_\_;

33 end if;

34

35 commit; 36 end;

37 $$;

33

**Výpis** **11** Procedura pro úpravu pozvánky

1 **create** **procedure** update\_invite (\_user\_id uuid, \_game\_request\_id uuid, ↪ \_accepted boolean, \_data bytea)

2 **language** plpgsql 3 **as**

4 $$

5 declare

6 v\_ready bool; 7 v\_game\_id uuid; 8 v\_exists bool; 9 begin

10 select exists(select \* from users\_to\_game\_requests where game\_request\_id ↪ = \_game\_request\_id and user\_id = \_user\_id)

11 into v\_exists;

12 if not v\_exists then

13 raise exception 'User with id ''%'' is not part of game request with ↪ id ''%'' or game request with id ''%'' doesn''t exists', \_user\_id,

↪ \_game\_request\_id, \_game\_request\_id; 14 end if;

15 if \_accepted then

16 update users\_to\_game\_requests set accepted = true where user\_id = ↪ \_user\_id and game\_request\_id = \_game\_request\_id;

17 select not exists(

18 select \* from users\_to\_game\_requests where game\_request\_id = ↪ \_game\_request\_id and not accepted

19 ) into v\_ready; 20 if v\_ready then

21 insert into games (name, data, last\_played, moves\_needed) select ↪ name, \_data, last\_played, moves\_needed from game\_requests where

↪ game\_requests.id = \_game\_request\_id returning games.id into v\_game\_id; 22 insert into games\_to\_users (user\_id, game\_id) select

↪ users\_to\_game\_requests.user\_id, v\_game\_id from users\_to\_game\_requests ↪ where game\_request\_id = \_game\_request\_id;

23 delete from users\_to\_game\_requests where game\_request\_id = ↪ \_game\_request\_id;

24 delete from game\_requests where id = \_game\_request\_id; 25 end if;

26 else

27 delete from users\_to\_game\_requests where game\_request\_id = ↪ \_game\_request\_id;

28 delete from game\_requests where id = \_game\_request\_id; 29 end if;

30 commit; 31 end;

32 $$;

34

**Výpis** **12** Zdrojový kód knihovny roles

1 **use** postgres::{Client, NoTls}; 2 **use** proc\_macro::TokenStream;

3 **use** proc\_macro2::Span; 4 **use** quote::quote;

5 **use** syn::punctuated::Punctuated;

6 **use** syn::{parse\_macro\_input, parse\_quote, Ident, ItemEnum};

7

8 #[proc\_macro\_attribute]

9 **pub** **fn** get\_roles\_from\_db(\_attr: TokenStream, item: TokenStream) -> ↪ TokenStream {

10 dotenv::dotenv().expect("Dotenv error");

11 **let** **mut** parsed\_enum = parse\_macro\_input!(item **as** ItemEnum);

12

13 **let** database\_url = std::env::var("DATABASE\_URL").expect("Missing env ↪ variable DATABASE\_URL");

14

15 **let** **mut** client = Client::connect(&database\_url, NoTls).expect("Couldn't ↪ create pool");

16

17 **let** **mut** variants = Punctuated::new(); 18 **for** row **in** client

19 .query("select name, id from roles", &[]) 20 .expect("Couldn't get roles from db")

21 {

22 **let** name = Ident::new(row.get(0), Span::call\_site()); 23 **let** id: i16 = row.get(1);

24 **let** variant: syn::Variant = parse\_quote! { 25 #name = #id **as** i32

26 };

27 variants.push(variant); 28 }

29

30 parsed\_enum.variants = variants; 31

32 (quote! {

33 #parsed\_enum 34 })

35 .into() 36 }

35