Gymnázium, Praha 6, Arabská 14

Programování

Ročníková práce

2023

Tobiáš Brňák, Ivan Merkulov, František Pešula

Gymnázium, Praha 6, Arabská 14

Arabská 14, Praha 6, 160 00

Ročníková práce

Předmět: Programování

Téma: Slovickoss

Školní rok: 2023/2024

Autor: Tobiáš Brňák, Ivan Merkulov, František Pešula

Třída: 3.E

Vedoucí práce: Mgr. Jan Lána

Třídní učitel: Mgr. Blanka Hniličková

Čestné prohlášení

Prohlašujeme, že jsme jedinými autory tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené. Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů udělujeme bezúplatně škole Gymnázium, Praha 6, Arabská 14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V Praze dne 26. dubna 2024

_____

_____

_____

Anotace

Slovickoss je webová aplikace pro učení slovíček. Tato webová aplikace umožní uživateli si zapisovat slovíčka v kterémkoliv jazyce, ze kterých se bude zároveň učit. Také obsahuje herní módy, přes které si uživatel může slovíčka procvičovat. Navíc náš učící algoritmus bude připomínat uživateli procvičovat si slovíčka.

Abstract

Slovickoss is a web application for learning vocabulary. This web application allows user to write down vocabulary in any language they want to learn. It aldo includes through which the user can practice their vocabulary. Additionally, our learning algorithm will remind the user to practice their vocabulary regularly.

Anmerkung

Slovickoss ist eine Webseite zum Vokabellernen. Diese Webseite ermöglicht dem Benutzer, Wörter in einer beliebigen Sprache aufzuschreiben und gleichzeitig daraus zu lernen. Es enthält auch Spielmodus, mit denen der Benutzer Vokabeln üben kann. Darüber hinaus erinnert unser Lernalgorithmus den Benutzer daran, Vokabeln zu üben.

Obsah

1 Úvod

Předmětem tohoto ročníkového projektu bylo vytvořit webovou stránku sloužící pro efektivní učení slovíček. Pro užití webové stránky si daný uživatel bude muset vytvořit účet. Webová stránka umožní uživateli si zapisovat slovíčka v jakémkoliv jazyce, která budou právě sloužit k jeho učivu. Uložená slovíčka se pak následně budou uživateli připomínat v různých časových intervalech podle uživatelova výběru, čímž si uživatel bude slovíčka neustále opakovat. Aby si uživatel slovíčka zopakoval, webová stránka využívá učícího algoritmu tvořící opakující se cyklus. Pro učení slovíček webová stránka uživateli nabízí několik herních módů. Webová stránka si zároveň ukládá progres naučených slovíček uživatele. Zdokumentováno bylo každé odvětví webové stránky. Bylo sepsáno, co jaká část obsahuje, co vykonává za funkci v celém projektu, k čemu slouží a případné doplňující informace.

Zadání

Zadání ročníkové práce znělo: vytvořit webovou stránku na efektivní učení slovíček, která nese název Slovickoss. Na stránce se budou moci zapisovat slovíčka na učení, která se budou následně připomínat podle učícího algoritmu. Po zapsání slovíček stránka bude sama připomínat uživateli, jaká slovíčka by si měl zopakovat. Pro učení slovíček webová stránka bude uživateli nabízet několik herních módů. A webová stránka bude ukládat progres naučených slovíček uživatele.

2 Vývojové prostředí

2.1 Django

Pro vývoj backendu jsme využili webový framework Django, který využívá MVC (model -view-controller) architekturu. Podle této architektury Django rozdělí celou aplikaci na tři komponenty. Model (v Djangu model) dostává nějaké parametry a podle nich vytahuje data z databáze vlastně propojuje naši aplikaci s databází tím, že s ní dokáže komunikovat. View (v Djangu temlate) zobrazuje uživateli data v podobě přívětivé pro uživatele. View obvykle obsahují html šablony. Controller (v Djangu views) tento komponent propojuje mezi sebou zbylé dva, kteří vlastně neví o vzájemné existenci. Controller dostane data z modelu a přeformátuje je do formátu pochopitelného pro view a nebo naopak dostane parametr z view a posle ho do modelu.

2.1.1 Django REST framework

V navázání na framework django jsme použili microframework Django REST framework a s jeho pomocí jsme vytvořili z Djanga full backend framework využívající principu **REST** (Representational State Transfer) a podporuje využívání http metod pro vytváření, čtení, aktualizaci a mazání zdrojů tzv. **CRUD** - create, read, update a delete. Názvy těchto http metod jsou **POST** - vytváření, **GET** - čtení, **PUT** - aktualizace a **DELETE** - mazání. Jednou z hlavních výhod Django REST frameworku je vestavěná podpora pro snadnou serializaci a deserializaci komplexních datových typů, které nám umožnili komunikaci s frontendovými API.

2.2 Angular

Pro vývoj frontendu jsme využili TypeScriptový framework Angular, který využívá logiku komponent pro stavbu aplikace. Základem každé komponenty je programová třída napsaná v TypeScriptu a takto framework staví na **OOP** (objektově orientované programování). K této třídě pak Angular naváže vlastní HTML šablonu a CSS. Tímto způsobem lze rozdělit webovou stránku na samostatné celky a poskládat ji právě pomocí principů OOP. Tím že každá komponenta bude mít vlastní izolovanou funkcionalitu a komponenty jsou znovupoužitelné ve více různých případech. Uveďme si příklad, mějme např. stránku jako Facebook. Menu na vršku stránky bude komponent, menu z levé a pravé strany budou též komponenty. Střední část, kde se zobrazují příspěvky je komponent a příspěvky v ní jsou samy o sobě také komponentou. Vlastně celá stránka je komponenta skládající se z dalších komponent.

2.3 MySQL

Zároveň jsme použili systém spravující relační databáze MySQL, který ukládá a spravuje strukturovaná data webové stránky. Tato databáze strukturovaná data organizuje a ukládá do tabulek, které slouží jako centrální úložiště, kde jsou informace spravovány a umožňuje uživatelům pracovat s daty. MySQL využívá strukturovaného dotazovacího jazyka (SQL) k manipulaci s daty, jejich ukládání, aktualizaci a správu relačních databází. Výhodou MySQL je jednoduchost a rychlý výkon, a proto se využívá jako samostatný server a ke skriptování v práci s daty.

2.4 Další knihovny

Dále jsme použili frontendový framework Bootstrap. Bootstrap usnadňuje vytváření funkčních webových stránek a aplikací s pomocí HTML, CSS a JavaScriptu. Také nabízí několik předpřipravených prvků a šablon, a taky podporu pro různé prohlížeče. Taktéž jsme použili knihovnu Angular Material, která poskytuje komponenty uživatelského rozhraní, jako jsou tlačítka, karty a mnoho dalšího. Tyto komponenty jsou navrženy tak, aby byly snadno použitelné na různých zařízeních.

3 Aplikace

V této části si názorně ukážeme jak funguje propojení frontendu a backendu v naší aplikaci. Pro větší názornost si to ukážeme na příkladu. Popíšeme si jak se na stránce Word Sets 4.2.4 zobrazují složky slovíček. Jako první krok musíme mít tabulku pro ukládání složek. Takovou tabulku si jednoduše vytvoříme v Djangu pomocí ORM (object relational mapping) v souboru *models.py*.

```
class Word_set(models.Model):
    id = models.AutoField(unique=True, primary_key=True)
    name = models.CharField(max_length=50)
    owner_id = models.ForeignKey(User,
    on_delete=models.CASCADE,default=None)
    created = models.DateField()
```

Listing 3.1: Word Set tabulka

Když už máme tabulku a v ní nějaká data, tak můžeme v souboru *views.py* vytáhnout potřebná data z databáze. Proto aby funkce ve *views.py* byla zavolána musí se dostat request na správnou url. Poté co se pošle request na určitou url v souboru *urls.py*, kde určujeme co se stane když přijde request na určitou url, definujeme jaká funkce z *views.py* se má zavolat (v našem případě funkce **get_user_word_sets** 3.2).

```
urlpatterns = [
    path('word-sets/', views.get_user_word_sets),
    path('word-sets/new', views.create_word_set),
    path('word-sets/new-word', views.create_word),
    path('word-sets/update', views.update_wordset),
]
```

Listing 3.2: urls.py

Funkce se zavolá a úplně prvním krokem zkontroluje jestli odpovídá typ requestu, který

má přijímat (**POST**, **GET**, **DELETE**, atd.) pokud request nebude odpovídat funkce se neprovede a server vrátí error, pokud bude opovídat, tak se metoda provede. V metodě získám pomocí Django ORM data z databáze v danném příkladě z requestu dostanu id uživatele, kterého složky chci získat z databáze. Proto podle získaného id vyfiltruji potřebnou query 3.3.

```
@api_view(['POST'])
def get_user_word_sets(request):
    user_id = request.data['userId']
    # Ziskam jmeno zakladatele setu
    user_name = User.objects.filter(id=user_id).
    values('name')
    user_name_serializer = User_name_serializer(user_name,
    many=True)

    word_set = Word_set.objects.filter(owner_id = user_id)
    # Serializuji vybrane polozky z DB
    word_set_serializer = Word_set_serializer(word_set,
    many=True)
    # Vratim JsonResponse na FE s serializovanymi daty z DB
    response_data = {
            'username': user_name_serializer.data,
            'word_sets': word_set_serializer.data,
        }
    return JsonResponse(response_data, status=200)
```

Listing 3.3: get_user_word_sets.py

Nyní máme potřebnou query, ale v momentálním formátu ji nemůžeme poslat na frontend, protože Angular přijímá data pouze v Json formátu. Proto musíme naše data serializovat a k tomu nám pomůžou serializátory z Django REST frameworku. Vytvoříme si potřebné serializátory pro data: jméno uživatele a pro složky slovíček danného uživatele 3.4. Poté co máme data serializovaná a již v potřebném formátu můžeme je v response poslat na frontend.

```python
class Word_set_serializer(serializers.ModelSerializer):
    class Meta:
        model = Word_set
        fields = '__all__'

class User_name_serializer(serializers.ModelSerializer):
    class Meta:
        model = User
        fields = ['name']
```

Listing 3.4: serializers.py

Na frontendu dostaneme data z backendu, která musíme zpracovat. K datům se pomocí metody přihlásíme a data správným způsobem uložíme do objektů. V našem případě dostaneme jméno uživatele a jeho složky slovíček. Proto si data uložím do příslušných objektů ze souboru *word-set.model.ts* 3.1 Po správném uložení dat můžeme data vypsat

```typescript
export class WordSet {
    constructor(
        public id: string,
        public name: string,
        public owner_id: string,
        public created: Date,
    ){}
}
```

Obr. 3.1: word-set.model.ts

v html. V 3.2 jsme sice dostali data z backendu a dokázali je uložit a poté zobrazit v html, ale nejdříve jsme museli poslat na backend request toho co vlastně chceme. Toho jsme dosáhli zavoláním v 3.2 funkce **getUsersSets** ze souboru *word-set.service.ts*. Tento soubor je služba, ve které se nastavují endpointy pro navázání spojení mezi frontendem a

```ts
export class WordSetListComponent implements OnInit {
  sets: WordSet[] = [];
  username: string = '';
  userSub!: Subscription;
  buttonVisibleMap: { [key: string]: boolean } = {};
  updateComponents: UpdateWordsetComponent[] = [];
  @ViewChildren('wordSetUpdateContainer', { read: ViewContainerRef }) wordSetUpdateContainers!: QueryList<ViewContainerRef>;

  constructor(
    private dataService: DataService,
    private wordSetService: WordSetService,
    private router: Router,
    private componentFactoryResolver: ComponentFactoryResolver
  ) { }

  ngOnInit() {
    const userId = this.dataService.getCurrentUserId();
    if (userId !== null) {
      this.wordSetService.getUserSets(userId).subscribe(
        (data) => {
          this.username = data.username[0].name;
          this.sets = data.word_sets;
          console.log(this.username);
          console.log(this.sets);
          this.sets.forEach(set => this.buttonVisibleMap[set.id] = true);
        },
        error => {
          console.error('Error:', error);
        }
      );
    }
  }
```

Obr. 3.2: word-set-list.component.ts

```html
<li *ngFor="let set of sets; let i = index" class="set-item">
  <div class="set-button" (click)="nvgWordSetDetail(set.id)">
    <div class="name-content" *ngIf="buttonVisibleMap[set.id]">
      <h1>{{ set.name }}</h1>
    </div>
    <div class="update-button-container">
```

Obr. 3.3: word-set-list.component.html

backendem. Zde vidíme, že metoda posílá id uživatele právě na tu url, kterou jsme měli

```ts
export class WordSetService {
  private baseUrl = 'http://127.0.0.1:7000/word-sets/';
  errorMessage: any;
  paramMap: any;

  constructor(private http: HttpClient) { }

  getUserSets(userId: number): Observable<any> {
    return this.http.post(this.baseUrl, { userId });
  }
```

Obr. 3.4: word-set.service.ts

definovanou na backendu a díky tomu se pak může na backendu zavolat funkce v souboru *views.py* 3.3.

4 Program

4.1 Backend

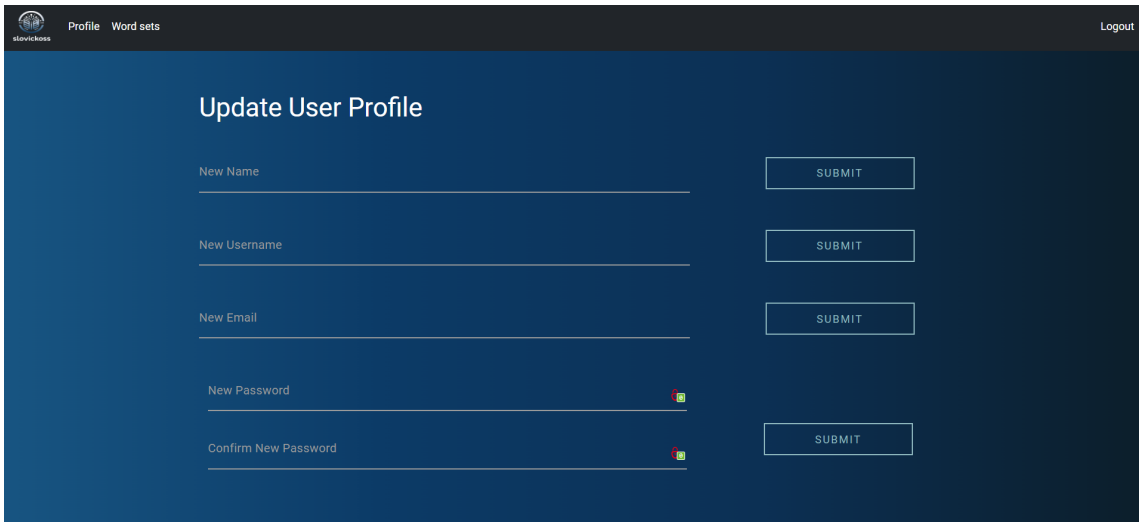
Backend postavený v Djangu má dvě aplikace **Core** a **Accounts**. Aplikace **Accounts** se stará o vytváření nových uživatelů, ověřování správnosti zadaných údajů během přihlašování a také o aktualizaci uživatelských údajů. Aplikace **Core** se stará o vytváření a aktualizaci složek slovíček, vytaváření nových slov, zobrazování dat na domovské stránce a možnost učení slovíček.

4.2 Frontend

Program jsme rozdělili na složky, které slouží k určité funkci v aplikaci. Složky obsahují komponenty, moduly, modely a služby:
- **Main** - je hlavní komponenta celé stránky, vytváří stránku a zobrazuje na ní dílčí komponenty
- **Account** 4.2 - zobrazuje uživateli data o něm a umožňuje mu je změnit
- **Auth** 4.2.1 - stará se o registraci a přihlašování uživatelů
- **Header** 4.2.2 - komponenta, která vytváří menu vrchní navigační menu
- **Home** 4.2.3 - vytváří domovskou stránku
- **Word Sets** 4.2.4 - obsahuje komponenty, které vytváří sety slovíček
- **Learning** 4.2.5 - umožňuje učit se slovíčka

4.2.1 Account

Obsahuje dvě komponenty **Account** a **Update-account** 4.1. Komponenta Account získává data o uživateli z databáze a zobrazuje je uživateli. Update-account umožňuje uživateli změnit jeho údaje uložené v databázi. Složka obsahuje modul, na který jsou napojeny obě

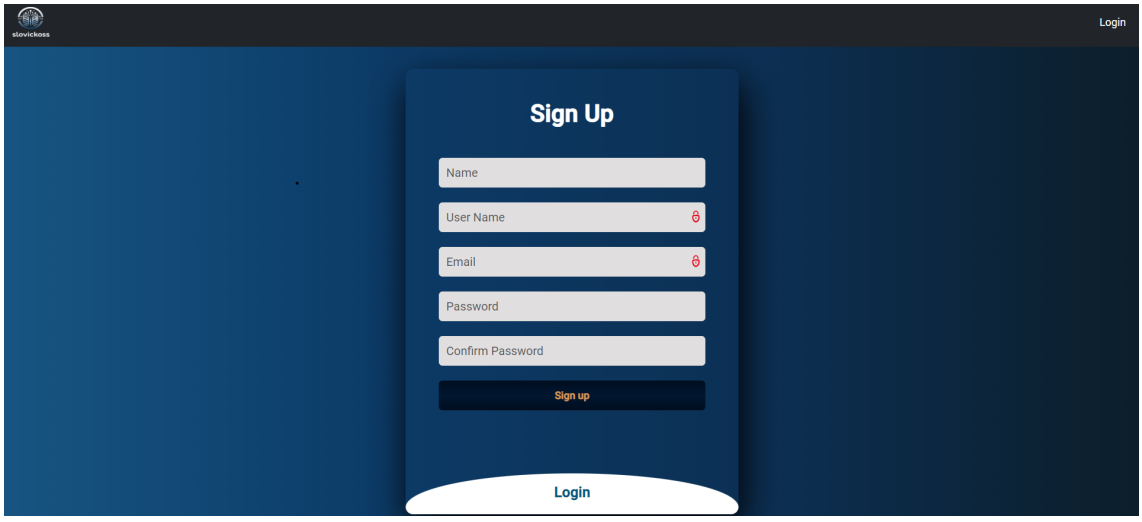


Obr. 4.1: Update-account

komponenty, model s potřebnými objekty a také obsahuje službu pro spojení s backendem.

4.2.2 Auth

Obsahuje komponent **Auth** 4.2, který umožňuje registraci a přihlášení uživatelů, také obsahuje model s objekty na přijímání a odesílání dat v potřebném formátu, module na propojení se zbytkem aplikace a službu na spojení s backendem.



Obr. 4.2: SignUp/Login

4.2.3 Header

Skládá se pouze z jedné komponenty a zobrazuje navigační menu na vrchu stránky, které je vytvořeno pomocí frameworku Bootstrap 2.3.

4.2.4 Home

Sestává z komponenty, která zobrazí domovskou stránku před a po přihlášení uživatele, modulu na propojení se zbytkem aplikace a služby pro navázání spojení s backendem. Po přihlášení najde uživatel na stránce složky slovíček, které mu stránka doporučí k procvičení a také složky slovíček, které nedávno procvičoval.

4.2.5 Word Sets

Obsahuje čtyři komponenty **New-word-set**, **Update-word-set**, **Word-set-detail** a **Word-set-list** 4.3. Nachází se zde také modul na propojení se zbytkem aplikace, model obsahující objekty na přijímání a posílání dat v určitém formátu a službu na nastavení endpointu a následné spojení s backendem. Komponent **New-word-set** umožňuje uživateli vytvořit novou složku slovíček. **Update-word-set** umožňuje uživateli přejmenovat složku, ve které se nachází slovíčka. **Word-set-list** zobrazuje uživateli všechny jeho složky slovíček. **Word-set-detail** obsahuje komponentu **New-word** a modul (*Word- set-detail.module*) na propojení s *Word-set* modulem. **New-word** umožňuje vytvořit nové slovo ve složce se slovíčky.

4.2.6 Learning

Obsahuje dvě komponenty **Learning** a **Normal-learning**, který umožňují uživateli se slovíčka učit. Dále obsahují modul na propojení se zbylým programem a službu na nastavení endpointů a propojení frontendu s backendem.

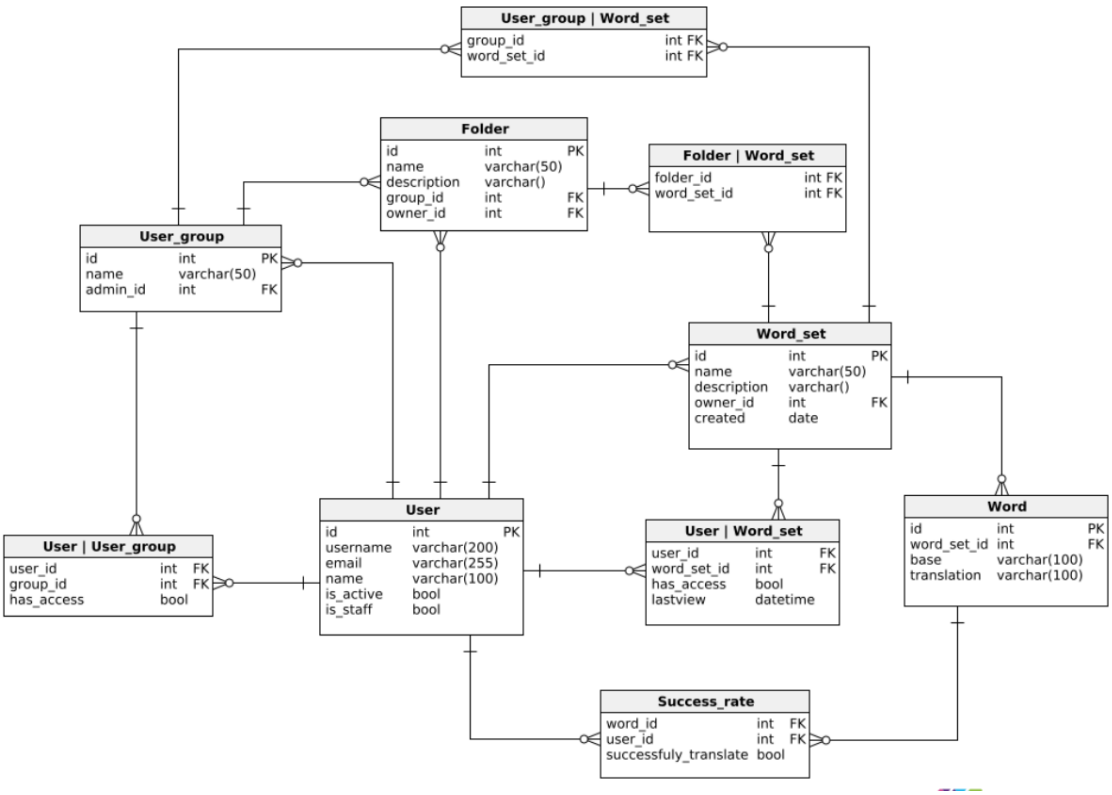
Obr. 4.3: Word Sets

4.3 Databáze

Jak už jsme zmínili dříve 2.2, tak jsme pro naši aplikaci využili systému pro relační databáze **MySQL**. Pro propojení Djanga s databází jsme využili pythonové knihovny PyMySQL [2]. Pro napojení MySQL do Djanga jsme museli změnit nastavení databáze v souboru *settings.py* a v souboru *__init__.py* importovat knihovnu PyMySQL a instalovat MySQL jako naší databázi.

```
1 import pymysql
2
3 pymysql.install_as_MySQLdb()
```

Listing 4.1: Soubor __init__.py



Obr. 4.4: Schéma databáze

5 Závěr

Téma pro tento ročníkový projekt bylo zvoleno za účelem vytvořit webovou stránku, která efektivněji zlepší učení cizojazyčných slovíček, tak aby stránku mohl používat kdokoliv, kdo se chce slovíčka naučit anebo zlepšit si svůj slovník různých jazyků. Výsledkem ročníkového projektu je webová stránka, která uživateli umožňuje učení slovíček. Webová stránka obsahuje veškeré funkce potřebné k učivu slovíček. Do webové stránky by se dalo přidat několik dalších funkcí, které by stránku zdokonalily ještě víc. Například by mohla obsahovat více herních módů, které by taktéž pomohly uživateli s učivem slovíček. Dále by stránka mohla zobrazovat statistiky uživatele: kolik slovíček se naučil, aktivita na stránce, nejpoužívanější herní mód. Během tvorby tohoto ročníkového projektu jsme získali mnoho zkušeností a nových znalostí, které by se v budoucnu dali využít v nadcházejících projektech. [3][1][4][5]

Seznam obrázků

Seznam ukázek kódu

Seznam zdrojů

[1] Learn with Castro. *Django Angular Basic Token Authentication*. Dostupné z: https://www.youtube.com/watch?v=CfS4JEAV9FY. 2021.

[2] Inada Naoki. *PyMySQL documentation*. Dostupné z: https://pymysql.readthedocs.io/en/latest. 2023.

[3] Neuvedeno. *Angular - Introduction to the Angular docs*. Dostupné z: https://angular.io/docs. 2024.

[4] Neuvedeno. *Django documentation*. Dostupné z: https://docs.djangoproject.com/en/5.0/. 2024.

[5] Neuvedeno. *Django REST framework*. Dostupné z: https://www.django-rest-framework.org/. 2024.