



STŘEDNÍ ŠKOLA PRŮMYSLOVÁ
A UMĚLECKÁ, OPAVA

ZÁVĚREČNÁ STUDIJNÍ PRÁCE

dokumentace

Django aplikace na zaznamenávání silových tréninků

Gym Tracker

Matěj Klimeš



Obor:	18-20-M/01 INFORMAČNÍ TECHNOLOGIE se zaměřením na počítačové sítě a programování
Třída:	IT4
Školní rok:	2023/2024

Poděkování

Rád bych poděkoval panu učiteli Ing. Petru Grussmannovi za užitečné rady při zpracování projektu. Rád bych také poděkoval své rodině a svým kamarádům a spolužákům za jejich podporu a pomoc během mého studia na této škole.

Prohlašuji, že jsem závěrečnou práci vypracoval samostatně a uvedl veškeré použité informační zdroje.

Souhlasím, aby tato studijní práce byla použita k výukovým účelům na Střední průmyslové a umělecké škole v Opavě, Praskova 399/8.

V Opavě 31. 12. 2023

Matěj Klimeš, autor práce

ABSTRAKT

Výsledkem projektu je základ webové aplikace pro zaznamenávání silových tréninků. Aplikace zatím obsahuje hlavní stránku, kde lze také spustit nový trénink. Dále je funkční seznam cviků, u kterých lze nalézt jejich zaměření na svalovou partii a druh. V detailu cvičení pak lze nalézt i jednoduchý popis provedení cviku. Další částí aplikace je seznam uskutečněných tréninků. Při kliknutí na jakýkoliv trénink se zobrazí jeho detail, včetně seznamu cviků, počtu sérií u každého cviku a dalších detailů. Poslední částí aplikace je samotný start nového tréninku. Po spuštění se zobrazí rozhraní tréninku, kde lze upravovat název tréninku a přidávat cvičení z databáze. U každého cvičení pak lze přidat libovolný počet sérií, které obsahují dva parametry, váhu a počet opakování. Trénink lze zrušit a ukončit a po ukončení tréninku se zobrazí rekapitulace tréninku s podrobnostmi podobnými detailu tréninku. Aplikace je založená na jednoduchosti, a tudíž dostupnosti pro kohokoliv.

Klíčová slova: webová aplikace, databáze, silový trénink, jednoduchost

ABSTRACT

The result of the project is the basis of a web application for recording strength workouts. The application currently contains a main page where you can also start a new training session. Furthermore, there is a functional list of exercises, where you can find their focus on the muscle part and type. A simple description of the exercise can be found in the details of the exercise. Another part of the application is a list of completed workouts. Clicking on any workout will display its details, including a list of exercises, the number of sets for each exercise, and other details. The last part of the application is the very start of the new workout. After starting, the workout interface will appear, where you can edit the name of the workout and add exercises from the database. For each exercise, you can add any number of sets that contain two parameters, weight and number of repetitions. A workout can be canceled and finished, and after the workout is finished, a workout recap will be displayed with details similar to the workout detail. The application is based on simplicity and therefore accessibility for anyone.

Keywords: web application, database, strength workout, simplicity

OBSAH

ÚVOD.....	5
TEORETICKÁ A METODICKÁ VÝCHODISKA	6
1.1 ZAZNAMENÁVÁNÍ SILOVÝCH TRÉNINKŮ.....	6
1.2 PŘEDEŠLÉ ZKUŠENOSTI	6
2 VYUŽITÉ TECHNOLOGIE	7
2.1 DJANGO FRAMEWORK	7
2.2 SQLITE	7
2.3 BOOTSTRAP 5	7
2.4 DOCKER	7
3 ZPŮSOBY ŘEŠENÍ A POUŽITÉ POSTUPY.....	8
3.1 NÁVRH MODELU DATABÁZE	8
3.2 ZALOŽENÍ PROJEKTU	8
3.3 STRUKTURA ADRESÁŘE	9
3.4 VYTVOŘENÍ DATABÁZE - MODELS.PY	10
3.5 ZÁKLADNÍ POHLEDY A ŠABLONY.....	10
3.5.1 Pohledy.....	10
3.5.2 Šablony.....	11
3.5.3 Vytvoření url cest.....	12
3.6 TRÉNINKOVÉ ROZHRAŇÍ.....	12
3.6.1 Vytvoření základních pohledů a šablon	12
3.6.2 Jednoduchý formulář.....	13
3.6.3 Úprava názvu tréninku v závislosti na čase	13
3.6.4 Přidávání cvičení na stránku	14
3.6.5 Přidávání sérií ke cvičením	17
3.6.6 Ukončení tréninku, zrušení tréninku – zápis do databáze.....	18
3.6.7 Rekapitulace tréninku.....	21
3.6.8 Update URL cest	22
3.7 SPUŠTĚNÍ TRÉNINKU.....	22
3.8 STATIC FILES	22
3.9 CELKOVÁ STYLIZACE APLIKACE.....	23
4 VÝSLEDKY ŘEŠENÍ, VÝSTUPY, UŽIVATELSKÝ MANUÁL.....	24
ZÁVĚR	25
SEZNAM POUŽITÝCH INFORMAČNÍCH ZDROJŮ	26

ÚVOD

Nápadem pro tento projekt bylo pohodlné a zároveň jednoduché zaznamenávání silových tréninků. To znamená možnost zaznamenat trénink, databázi cviků, historii tréninků a sledování progresu. Prvně jsem chtěl tento nápad vložit do mobilní aplikace, z důvodu nedostatku zdrojů jsem se rozhodl začít s webovou aplikací a později ji případně na mobilní zařízení přenést. Při výběru frameworku jsem se rozhodoval mezi nástavbou flasku, Flask-AppBuilderem, a Django frameworkem. Nakonec zvítězilo Django, které mi přišlo vhodnější vzhledem k více rozšířením a větší základnou zkušených programátorů.

Mezi mé hlavní cíle bych zařadil vytvoření aplikace, která bude motivovat k vyšším výkonům díky jasně zobrazenému progresu (grafy apod.), jejíž jednoduchost a přehlednost nebude nikoho rozptylovat od svých cílů a možnost personalizace samotné aplikace.

V dalších částech dokumentace bude popsán podrobnější postup a překážky při vývoji aplikace a poukázáno na další možná zlepšení.

TEORETICKÁ A METODICKÁ VÝCHODISKA

1.1 Zaznamenávání silových tréninků

Aplikací na zaznamenávání tréninků je mnoho. I přesto mě ale lákalo si jednu takovou také vytvořit, abych hlouběji porozuměl problematice podobných aplikací a zároveň si aplikaci přizpůsobil svým preferencím. Narazil jsem na několik překážek a musel jsem v průběhu vývoje ustoupit od svých původních předpokladů, ale to se v budoucnu může změnit.

1.2 Předešlé zkušenosti

S Django frameworkem jsem se již setkal o rok dříve, když jsme ve škole vytvářeli jednoduchou databázovou aplikaci. U tohoto projektu mi tato zkušenost pomohla se lépe zorientovat ve struktuře aplikace a pochopení souvislostí. S jazykem Python jsem se seznámil také před rokem a pomohlo mi to ve vyvarování se syntaktických chyb při vývoji aplikace.

2 VYUŽITÉ TECHNOLOGIE

2.1 Django framework

Django je bezplatný a open-source webový framework založený na jazyku Python, běžící na webovém serveru. Používá architektonický vzor Model – Template (šablona) – Views (pohled) -> MTV.

2.2 SQLite

SQLite je relační databázový systém obsažený v knihovně. Na rozdíl od databází založených na principu klient–server je SQLite pouze knihovna, která je k dispozici pomocí jednoduchého rozhraní. Každá databáze je uložena v samostatném souboru .dbm.

2.3 Bootstrap 5

Bootstrap představuje svobodný a otevřený soubor nástrojů pro vytváření a stylizaci webových stránek a aplikací. Obsahuje designové šablony postavené na HTML a CSS, které umožňují úpravy typografie, formulářů, tlačítek, navigačních prvků a dalších součástí uživatelského rozhraní.

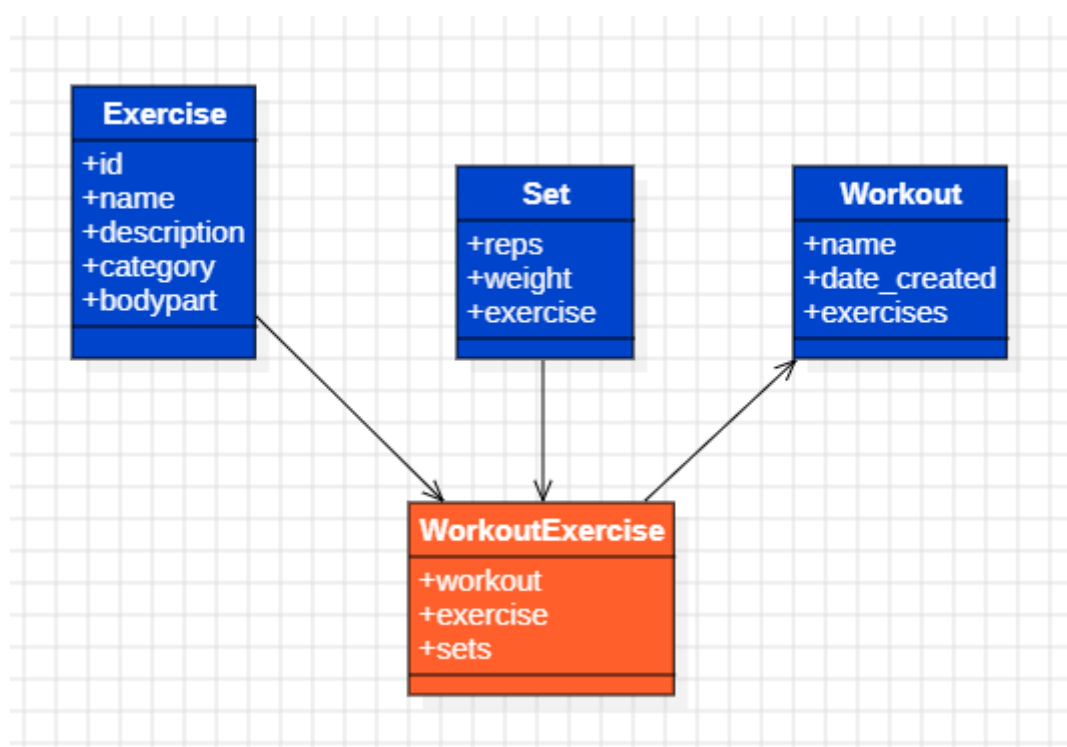
2.4 Docker

Docker představuje v oblasti informatiky open-source software s cílem poskytnout jednotné rozhraní pro izolaci aplikací do kontejnerů, které je kompatibilní s operačními systémy macOS, Linux a Windows. Tato technologie představuje formu "odlehčené virtualizace".

3 ZPŮSOBY ŘEŠENÍ A POUŽITÉ POSTUPY

3.1 Návrh modelu databáze

Prvně jsem potřeboval navrhnout databázi, která by zvládala zaznamenávat tréninky. To znamená možnost přiřadit cvičení k tréninku, série ke cvičení v určitém tréninku a tabulku pro uložení cvičení. Aby bylo tohle všechno možné, vytvořil jsem, kromě tabulek „*Exercise*“, „*Workout*“ a „*Set*“, pomocnou tabulku „*WorkoutExercise*“, která tohle zajišťovala.



Obrázek 1: Model databáze 1

3.2 Založení projektu

Poté jsem mohl začít programovat aplikaci. První jsem musel nainstalovat Python, který stačilo stáhnout z oficiálních stránek. Poté jsem potřeboval vytvořit samotný Django projekt, to znamenalo vytvoření virtuálního prostředí, nainstalování knihovny Django a zadání příkazu pro vytvoření projektu. Django framework sám vytvoří autentifikační prostředí, webovou správu databáze a hlavní strukturu adresáře.

\\\

```
D:\final_project> py -m venv .venv
D:\final_project> .\.venv\Scripts\activate
(.venv) D:\final_project> pip install django
\\\
```

3.3 Struktura adresáře

----- gym	// složka aplikace
----- migrations	// postupné migrace databáze
----- static	// static soubory
----- css	// css soubory aplikace
----- js	// javascript soubory aplikace
----- templates	// šablony html
----- __init__.py	
----- admin.py	
----- forms.py	// formuláře
----- models.py	// modely (tabulky v databázi)
----- urls.py	// url cesty
----- views.py	// pohledy
----- gymapp	// složka projektu
----- __init__.py	
----- settings.py	// nstavení projektu
----- urls.py	// hlavní url cesty
----- staticfiles	// složka static souborů (collectstatic)
----- .dockerignore	// soubory ignorované dockerem
----- .env	// prostředí pro docker
----- .gitignore	// soubory ignorované gitem
----- db.sqlite3	// soubor s SQLite databází
----- docker-compose.yaml	// soubor pro docker
----- Dockerfile	// soubor pro docker
----- manage.py	
----- requirements.txt	// soubor s knihovnamy

3.4 Vytvoření databáze - models.py

Při vytváření jednotlivých tabulek jsem vycházel z dříve navrhnutého modelu databáze. Musel jsem správně určit vztahy mezi tabulkami, kde jsem narazil na jeden problém. Když jsem v tabulce „*WorkoutExercise*“ potřeboval přiřadit tabulku „*Workout*“ do vztahu, nešlo to z důvodu pořadí tabulek v kódu, jenže jsem tabulku nemohl ani přesunout pod třídu „*Workout*“, jelikož nastával podobný problém. Nakonec jsem zjistil, že stačí ve vztahu dát třídu „*Workout*“ do uvozovek a vše bylo v pořádku.

```
'''  
class WorkoutExercise(models.Model):  
    workout = models.ForeignKey('Workout', on_delete=models.CASCADE)  
    exercise = models.ForeignKey(Exercise, on_delete=models.CASCADE)  
    sets = models.ManyToManyField(Set, blank=True)  
  
class Workout(models.Model):  
    name = models.CharField(max_length=100, blank=True)  
    date_created = models.DateTimeField(auto_now_add=True)  
    exercises = models.ManyToManyField(Exercise, through=WorkoutExercise,  
                                       blank=True)  
  
'''  
models.py
```

3.5 Základní pohledy a šablony

3.5.1 Pohledy

Při vytváření pohledů jsem používal vestavěné Django pohledy z knihovny „*django.views.generic*“ („*DetailView*“, „*ListView*“, „*View*“). Vytvořil jsem jednoduché pohledy pro seznam cviků a tréninků a jejich detaily a pohled pro domovskou stránku.

```
'''  
class ExerciseListView(ListView):  
    model = Exercise  
    template_name = 'exercise_list.html'  
    context_object_name = 'exercises'  
'''
```

views.py

3.5.2 Šablony

Před vytvořením šablon jsem si první vytvořil „*base_generic.html*“ soubor pro jednodušší práci se šablonami. Krom základních bloků jsem přidal i menu pro snazší orientaci v aplikaci. Poté jsem vytvořil jednoduché šablony pro výpis cvičení a tréninků a jejich detailů.

```
'''  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1,  
shrink-to-fit=no">  
    <title>{% block title %}My Site{% endblock %}</title>  
    {% block head %}{% endblock %}  
</head>  
<body>  
    <div id="menu">  
        {% if request.path != '/gym/running_workout/' %}  
            <a href="{% url 'index' %}">Home</a>  
            <a href="{% url 'exercise_list' %}">Exercises</a>  
            <a href="{% url 'workout_list' %}">Workouts</a>  
            <a href="{% url 'start_new_workout' %}">Start New Workout</a>  
        {% endif %}  
    </div>
```

```
<div class="container">
    {% block content %}{% endblock %}
</div>

{% block scripts %}{% endblock %}
</body>
</html>
'''
```

base_generic.html

3.5.3 Vytvoření url cest

V souboru „*urls.py*“ jsem samozřejmě přidal cesty k seznamům a detailům. Nainportoval jsem pohledy z „*views.py*“ a ke všem cestám jsem přidal parametr „*name*“ pro snadné přístupu k jejich URL pomocí Django syntaxe.

```
'''
urlpatterns = [
    path('exercises/', ExerciseListView.as_view(),
         name='exercise_list'),
    path('exercises/<int:pk>/', ExerciseDetailView.as_view(),
         name='exercise_detail'),
    path('workouts/', WorkoutListView.as_view(), name='workout_list'),
    path('workouts/<int:pk>/', WorkoutDetailView.as_view(),
         name='workout_detail'),
]
'''
```

urls.py

3.6 Tréninkové rozhraní

3.6.1 Vytvoření základních pohledů a šablon

K vytvoření pohledu pro samotné tréninkové rozhraní jsem použil vestavěný jednoduchý pohled z knihovny „*django.views.generic*“, „*View*“. Použil jsem metodu „*get*“ k zavolání formuláře a následně jsem nechal vykreslit šablonu „*running_workout.html*“.

```
'''  
class RunningWorkoutView(View):  
    def get(self, request):  
        form = WorkoutForm()  
        return render(request, 'running_workout.html', {'form': form})  
'''
```

views.py

3.6.2 Jednoduchý formulář

V souboru „*forms.py*“ jsem potřeboval vytvořit nový formulář vykreslující parametry tréninku. Potřeboval jsem dvě informace, jméno tréninku a seznam cvičení v něm. Použil jsem tedy opět Django knihovnu, ze které jsem naimportoval „*forms*“. Poté jsem jen určil model a které pole z něj potřebuji.

```
'''  
class WorkoutForm(forms.ModelForm):  
    class Meta:  
        model = Workout  
        fields = ['name', 'exercises']  
'''
```

forms.py

3.6.3 Úprava názvu tréninku v závislosti na čase

Jako další, doplňková, ale efektní, část mě napadla změna názvu tréninku v závislosti na reálném čase. Navíc mě lákala myšlenka se seznámit s prací s reálným časem. Importoval jsem tedy knihovnu „*datetime*“, kterou jsem dále zakomponoval do mého formuláře. Prvně jsem načel do proměnné čas ve chvíli spuštění formuláře, a s tímto časem jsem pak dále pracoval – nastavil jsem podmínky, podle kterých se výchozí jméno tréninku mění („*Morning workout*“, „*Afternoon workout*“...).

```
'''
def __init__(self, *args, **kwargs):
    super().__init__(*args, **kwargs)
    current_time = datetime.now().time()

    # Set default workout name based on the current time
    if current_time >= datetime.strptime('05:00:00',
        '%H:%M:%S').time() and current_time <= datetime.strptime('11:59:59', '%H:%M:%S').time():
        self.fields['name'].initial = 'Morning Workout'
    elif current_time >= datetime.strptime('12:00:00',
        '%H:%M:%S').time() and current_time <= datetime.strptime('17:59:59', '%H:%M:%S').time():
        self.fields['name'].initial = 'Afternoon Workout'
    // rest of conditions
'''
```

forms.py

3.6.4 Přidávání cvičení na stránku

Dále jsem potřeboval, ať uživatel může do běžícího tréninku přidávat jakýkoliv počet cvičení. Proto jsem první potřeboval načíst všechna cvičení, aby z nich později mohl uživatel vybírat a po jednom jakýkoliv cvik přidávat. Když jsem se o tohle snažil, použil jsem funkci „*fetch()*“ a narazil jsem na problém, kdy mi pohled „*ExerciseList*“ vracel odpověď ve formátu HTML. Tento formát nešel zpracovat a tudíž jsem musel přijít s jiným řešením. Nakonec jsem přišel na to, že udělám nový pohled „*ExerciseJSONView*“ pouze pro tento specifický účel, který bude vracet data v JSON formátu.

```
'''  
class ExerciseJSONView(View):  
    def get(self, request, *args, **kwargs):  
        exercises = Exercise.objects.all()  
        data = list(exercises.values())  
        return JsonResponse(data, safe=False)
```

```
'''
```

views.py

Po vytvoření tohoto pohledu již funkce „*fetchExercises()*“ fungovala správně, a po úpravě v šabloně jsem na stránce zprovoznil možnost vybrání si jakéhokoliv cvičení z databáze. Dále jsem ve funkci přidal kód pro vypsání informací o chybě, kdyby nějaká nastala.

```
'''  
function fetchExercises() {  
    fetch('{% url "exercise_list_json" %}')    .then((response) => response.json())  
    .then((data) => {  
        const selectElement = document.getElementById("exercise-select");  
        selectElement.innerHTML =  
            '<option value="" disabled selected>Select an exercise</option>'  
        data  
            .map(  
                (exercise) =>  
                    `<option value="${exercise.id}">${exercise.name}</option>`  
            )  
            .join("");  
    })  
    .catch((error) => console.error("Error fetching exercises:", error));  
}
```

```
'''
```

running_workout.html

Adresa URL Žádosti:	http://127.0.0.1:8000/gym/exercises-json/
Metoda Žádosti:	GET
Stavový Kód:	● 200 OK
Vzdálená Adresa:	127.0.0.1:8000
Zásada Odkazujícího:	same-origin

Obrázek 2: Spuštěná fetch() funkce

Poslední fází bylo přidat script, který po kliknutí na tlačítko „Add exercise“ přidá vybrané cvičení na stránku. K tomu jsem využil dynamické přidávání prvků na stránce. Vytvořil jsem blok stránky s id, na které jsem poté odkazoval ve scriptu.

```
'''  
<div id="exercise-list-placeholder"></div>  
'''  
  
running_workout.html
```

Ve scriptu jsem vytvořil funkci „addSelectedExercise()“, ve které jsem uložil do proměnné obsah pole pro výběr cvičení. Pak jsem si uložil do další proměnné místo, kde se budou jednotlivá cvičení vkládat. Poté jsem pomocí „innerHTML“ sepsal HTML kód, který se vkládal na místo cvičení (v dalším bodě).

```
'''  
  
function addSelectedExercise() {  
    const selectElement = document.getElementById("exercise-select");  
    const selectedExerciseIndex = selectElement.selectedIndex;  
  
    if (selectedExerciseIndex !== -1) {  
        const selectedOption = selectElement.options[selectedExerciseIndex];  
        const exerciseId = selectedOption.value;  
        const exerciseName = selectedOption.textContent;  
        const exerciseListPlaceholder = document.getElementById(  
            "exercise-list-placeholder"  
        );  
    }  
}
```

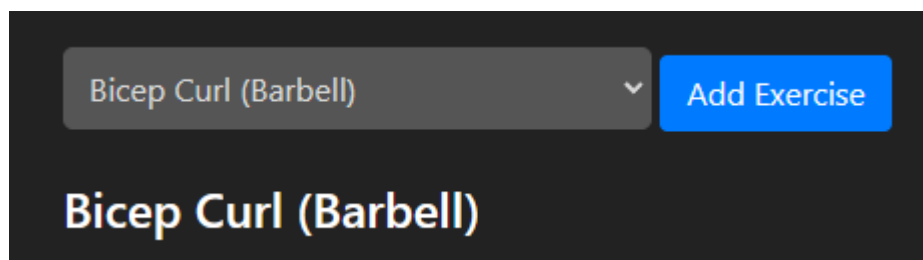


```
const newExerciseElement = document.createElement("div");
newExerciseElement.setAttribute("data-exercise-id", exerciseId);
newExerciseElement.innerHTML = `// ...html code //`;
exerciseListPlaceholder.appendChild(newExerciseElement);

const selectedExercisesInput =
  document.getElementById("selected-exercises");
selectedExercisesInput.value += exerciseId + ",";

updateSelectedExercisesInput();
}
}
```
```

running\_workout.html



Obrázek 3: Funkční tlačítko Add Exercise 1

### 3.6.5 Přidávání sérií ke cvičením

Dalším krokem k praktičnosti a funkčnosti aplikace bylo moci přidat libovolný počet sérií ke cvičení. Série musí obsahovat dvě naprosto fundamentální hodnoty, váhu a počet opakování. Mým prvním krokem tedy bylo přidání jedné série automaticky, ihned po přidání cviku, a poté možnost přidat vždy jednu další sérii příslušným tlačítkem. Začal jsem tedy sepsáním kódu pro přidání první série a přidání tlačítka (viz. předešlý bod). Poté jsem přidal „*EventListener*“ do funkce „*addSelectedExercise*“ a připravil si tedy místo, kde budu volat další funkci „*addSetToExercise*“.

```
```\nconst addSetButton = newExerciseElement.querySelector(".add-set-button");\n    addSetButton.addEventListener("click", addSetToExercise);\n```\n\nrunning_workout.html
```

Posledním krokem k funkčnosti bylo vytvoření funkce „*addSetToExercise*“. Prvně jsem si zavolal „*id*“ od cvičení, ke kterému jsem následně přiřadil jednu sérii, řádek, navíc. Nakonec jsem definoval HTML kód.

```
```\nfunction addSetToExercise() {\n    const exerciseId = this.getAttribute("data-exercise-id");\n    const exerciseDiv = document.querySelector(\n        `[data-exercise-id="${exerciseId}"]`\n    );\n    // rest of the code\n}\n```\n\nrunning_workout.html
```

### 3.6.6 Ukončení tréninku, zrušení tréninku – zápis do databáze

V tréninkovém rozhraní nemůžou chybět dvě tlačítka, a to pro dokončení tréninku a zrušení tréninku. Dokončení tréninku je tedy provedeno kliknutím na tlačítko „*Finish workout*“, zde jsem musel udělat hned několik úprav, aby se trénink uložil do databáze. Prvním krokem bylo vytvoření pohledu. Upravil jsem pohled „*RunningWorkoutView*“, aby postupně zapisoval vše do databáze. U téhle části jsem se musel podívat do více zdrojů, jelikož se mi pořád nedařilo zapsat vše včetně sérií. Nakonec jsem musel přidat neviditelné „*input*“ elementy do „*running\_workout.html*“ a brát z nich data. Před koncem ukládání do databáze jsem přiřazoval série ke cvičení (tréninku).

```
'''
def post(self, request):
 form = WorkoutForm(request.POST)
 if form.is_valid():
 # Save the workout without committing to the database
 workout = form.save(commit=False)
 # ... python code
 # Retrieve set details from the hidden input field
 set_details_json = request.POST.get('set_details', '[]')
 set_details = json.loads(set_details_json)

 # Create sets and associate them with the workout
 for detail in set_details:
 exercise_id = detail['exerciseId']
 exercise = Exercise.objects.get(pk=exercise_id)
 reps = detail['reps']
 # ... python code
 return redirect('workout_summary', workout_id=workout.id)
'''
```

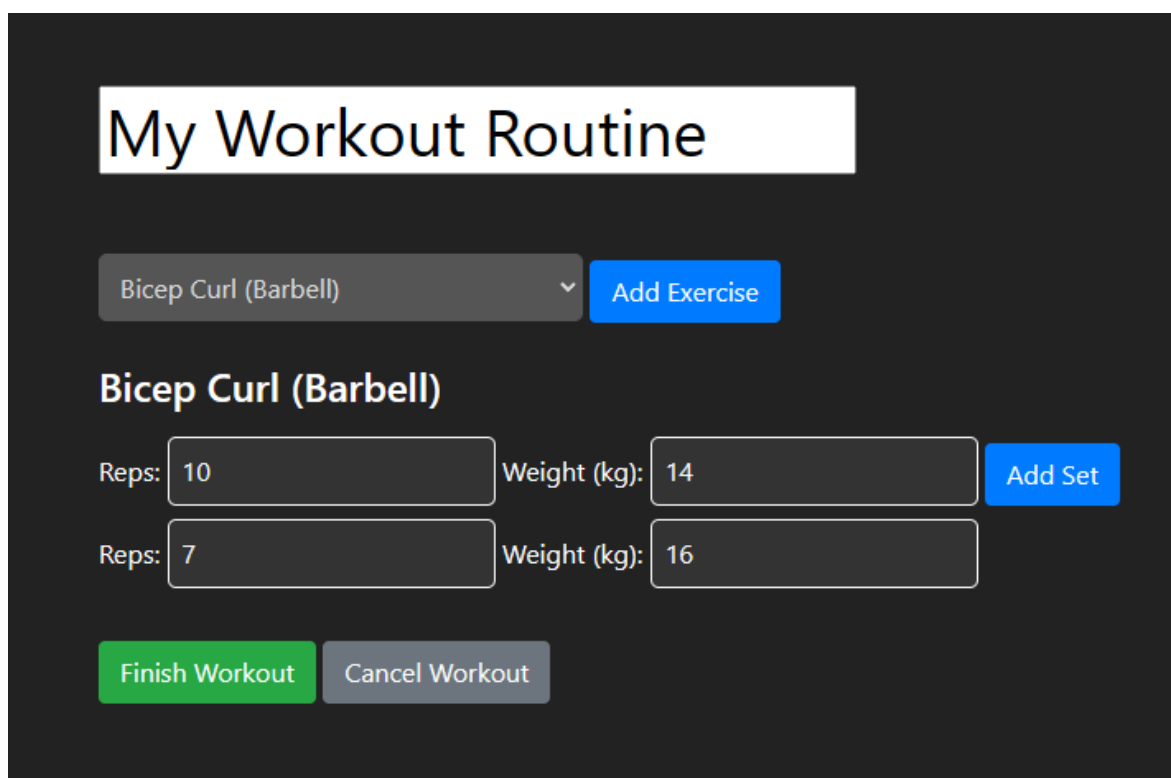
views.py

Dalším krokem bylo vyvolání funkcí v „*running\_workout.html*“, kde jsem navíc ošetřil případ, kdy chce uživatel ukončit trénink, ale není přidáno jediné cvičení. Poté jsem přečetl data z „*div*“ elementu, kde se nacházejí všechny potřebné informace.

```
'''
document.getElementById("workout-form")
 .addEventListener("submit", function () {
 if (document.getElementById("selected-exercises").value === "") {
 alert("Please add at least one exercise to the workout.");
 event.preventDefault();
 }
 })
// ... javascript kód
```

```
document.querySelectorAll('[id^="exercise-list-placeholder"].set-row')
 .forEach((setRow) => {
 const exerciseId = setRow
 .closest("[data-exercise-id]")
 .getAttribute("data-exercise-id");
 // ... javascript kód
 })
```

running\_workout.html



Obrázek 4: Tréninkové rozhraní 1

Poslední tlačítko na stránce tréninkového rozhraní má pouze funkci vrácení se na stránku s výpisem tréninků. Běžící trénink do databáze nezapíše.

```
```\n\ndocument.getElementById('cancel-workout').addEventListener('click',\nfunction () {\n\n    window.location.href = '/gym/workouts/';\n\n});\n```\n
```

cancel_workout.js

3.6.7 Rekapitulace tréninku

Po ukončení tréninku se zobrazí krátká rekapitulace, včetně seznamu cvičení, sérií a všech podrobností. Šablona vypadá téměř identicky s tréninkovým detailem.



Obrázek 5: Rekapitulace tréninku 1

3.6.8 Update URL cest

Jako poslední věc jsem přidal cesty k dosud vytvořeným pohledům.

```
'''  
urlpatterns += [  
    path('running_workout/', RunningWorkoutView.as_view(),  
        name='running_workout'),  
    path('workout_summary/<int:workout_id>/',  
        WorkoutSummaryView.as_view(), name='workout_summary'),  
]  
'''
```

urls.py

3.7 Spuštění tréninku

Spuštění tréninku jsem vyřešil pomocí URL cesty, která ihned uživatele přesměruje na adresu „*running_workout.html*“.

```
'''  
class StartNewWorkoutView(View):  
    def get(self, request, *args, **kwargs):  
        return redirect('running_workout')  
'''
```

views.py

3.8 Static files

V Django frameworku je běžnou praxí umístit např. CSS soubory, JavaScript soubory (dále „*static soubory*“) do složky „*/static/*“. Aby však Django našlo cestu, musí se přidat nastavení do „*settings.py*“. Pak už jen stačí do šablony napsat „*{% load static %}*“.

```
'''  
STATIC_URL = '/static/'  
STATICFILES_DIRS = [os.path.join(BASE_DIR, 'gym', 'static')]  
// collectstatic folder  
STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')  
'''  
  
settings.py  
  
'''  
{% load static %}  
<!DOCTYPE html>  
<html lang="en">  
// ... html kód  
'''  
  
base_generic.html
```

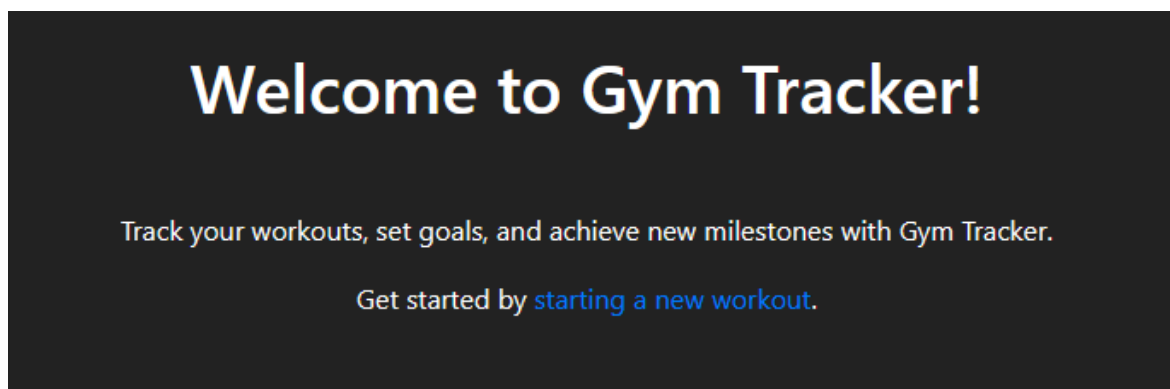
3.9 Celková stylizace aplikace

K hlubším úpravám vzhledu aplikace jsem použil kaskádové styly (CSS), ve kterých jsem upravoval celkový vzhled (pozadí aplikace, barvy...). K těm povrchovým jsem použil soubor nástrojů Bootstrap 5, který vyniká svou jednoduchostí (odsazení...). Celkově jsem zvolil tmavší tematiku a kombinace šedé a modré mi přijde pěknou, a zároveň jednoduchosti přívětivou, volbou.

4 VÝSLEDKY ŘEŠENÍ, VÝSTUPY, UŽIVATELSKÝ MANUÁL

4.1 Funkční část aplikace

Po spuštění aplikace se ocitneme na hlavní stránce, odkud se můžeme dostat v podstatě kamkoliv, díky hlavnímu menu. Zde najdeme seznam cviků, seznam tréninků a možnost spuštění nového tréninku. Na stránce s cviky můžeme vidět jméno cviku, jeho typ a zaměření na svalovou skupinu. Po kliknutí se dostaneme na detail cviku, kde je i krátký popis jeho provedení. Na stránce s tréninky máme možnost si trénink rozkliknout a podívat se na všechny detaily, včetně jména tréninku, data, cvičení i sérií. Stejný náhled můžeme vidět také po dokončení tréninku. Při samotném startu nového tréninku se dostaneme do tréninkového rozhraní, kde můžeme měnit jméno tréninku, přidávat libovolný počet cvičení a taktéž sérií. To jsou tedy základní funkce aplikace.



Obrázek 6: Hlavní stránka 1

4.2 Splněné cíle a výhledy do budoucna

Aplikace je nyní v použitelném stavu, lze zapisovat a analyzovat své tréninky. Co bych ale určitě v budoucnu chtěl přidat, tak je přehledné zobrazování postupu. To znamená různé grafy, například pro každý cvik. Pak taky možnost sledovat intenzitu tréninků a tím pádem i pomoc s optimalizací tréninkových plánů. Rozhodně možnost vytvořit různé šablony tréninků, kdy bude pro uživatele mnohem pohodlnější jen kliknout na personalizovanou šablonu a mít zde již připravené cviky. Je toho mnohem víc, ale tohle jsou takové hlavní nápady na vylepšení aplikace Gym Tracker.

ZÁVĚR

Jako maturitní projekt jsem vypracoval webovou aplikaci na zaznamenávání silových tréninků. Aplikace dostala aspoň základním požadavkům, které jsem si ve fázi plánování stanovil, jak z hlediska funkčnosti, tak jednoduchosti a přehlednosti. Lze zde zapisovat tréninky, prohlédnout si jednotlivé cviky a sledovat svou historii tréninků.

Při tvorbě aplikace jsem narazil na spoustu problémů, které jsem musel vyřešit, tudíž mě to posunulo jako programátora, ale i jako člověka s lepším kritickým myšlením a touhou nevzdávat se. Je toho ještě spousta, v čem se mohu zlepšovat, a tak můžu jen doufat, že tímto projektem mé studium zdaleka nekončí.

Repozitář projektu na Githubu: https://github.com/matejklimes/maturita_project.git

SEZNAM POUŽITÝCH INFORMAČNÍCH ZDROJŮ

- [1] Bootstrap: Build fast, responsive sites with Bootstrap. [online]. [cit. 2024-01-08]. Dostupné z: <https://getbootstrap.com/>
- [2] Bootstrap 5 CheatSheet [online]. [cit. 2024-01-08]. Dostupné z: <https://bootstrap-cheatsheet.themeselection.com/>
- [3] Django Documentation [online]. [cit. 2024-01-08]. Dostupné z: <https://docs.djangoproject.com/en/5.0/>
- [4] Django: The web framework for perfectionists with deadlines. [online]. [cit. 2024-01-08]. Dostupné z: <https://www.djangoproject.com/>
- [5] Docker: Make better, secure software from the start. [online]. [cit. 2024-01-08]. Dostupné z: <https://www.docker.com/>
- [6] ITNetwork: Učíme národ IT. [online]. [cit. 2024-01-08]. Dostupné z: <https://www.itnetwork.cz/>
- [7] Python Documentation [online]. [cit. 2024-01-08]. Dostupné z: <https://docs.python.org/3/>
- [8] Python: Python is powerful...and fast. [online]. [cit. 2024-01-08]. Dostupné z: <https://www.python.org/>
- [9] Stack Overflow: Everyone has a tab open to Stack Overflow. [online]. [cit. 2024-01-08]. Dostupné z: <https://stackoverflow.com/>
- [10] Wikipedie [online]. [cit. 2024-01-08]. Dostupné z: <https://cs.wikipedia.org/wiki/>
- [11] W3Schools: Online Web Tutorials [online]. [cit. 2024-01-08]. Dostupné z: <https://www.w3schools.com/>

