Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Deník atleta

2021

Lukáš Střelecký

Vedoucí práce:
RNDr. Arnošt Večerka

Studijní obor: Aplikovaná informatika,
prezenční forma

Bibliografické údaje

Autor:	Lukáš Střelecký
Název práce:	Deník atleta
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2021
Studijní obor:	Aplikovaná informatika, prezenční forma
Vedoucí práce:	RNDr. Arnošt Večerka
Počet stran:	58
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliograhic info

Author:	Lukáš Střelecký
Title:	Athlete's diary
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2021
Study field:	Applied Computer Science, full-time form
Supervisor:	RNDr. Arnošt Večerka
Page count:	58
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

V rámci bakalářské práce byla vyvinuta aplikace pro správu a zobrazení tréninkových jednotek sportovce. Aplikace ukládá data ze tří hlavních oblastí sportovního tréninku (silové, běžecké a nutriční) a umožňuje tak sportovci mít přehled o své kondiční přípravě. Data jsou následně prezentována v podobě grafů a tabulek.

Synopsis

The aim of this bachelor's thesis is to develop a web application for managing and visualization of sports training for athletes. Application is able to store three main areas of sports training such as strength, stamina, and nutrition. This allows the athlete to have an overview of their current fitness level. The data are presented in the form of graphs and tables.

Klíčová slova: sportovní aplikace; deník; atlet

Keywords: sports application; diary; athlete

Děkuji vedoucímu práce, RNDr. Arnoštu Večerkovi, za cenné rady, testování a konstruktivní připomínky během tvorby této bakalářské práce. Dále bych rád poděkoval všem svým přátelům, kteří mě podpořili, inspirovali, a bez jejichž pomoci by nebylo možné práci dokončit.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam zdrojových kódů

1 Úvod

Sport patří mezi pohybovou aktivitu, kterou se lidé baví celá staletí. Za prvopočátek sportovních soutěží můžeme považovat Starověké Řecko, ve kterém se pohybová aktivita stala součástí kulturního života. Člověk se postupně zdokonaloval a výsledky mu přinášely obohacení každodenního života. Postupem času začaly vznikat různé úrovně provozování sportu. V dnešní době se můžeme setkat se třemi druhy.

- **Vrcholová:** sportovci denně trénují a často mívají smlouvu na plný úvazek u svých sportovních klubů.
- **Výkonnostní:** pravidelný trénink a účast na soutěžích, osoba je registrována v některém sportovním svazu.
- **Rekreační:** příležitostné sportování bez registrace a stanovených pravidel.

Jedním z nástrojů, který slouží ke zdokonalení fyzických zdatností a dosažení sportovních cílů, je tréninkový deník. Využívají jej nejvíce lidé provozující sport na vrcholové úrovni. Obsahuje detailní záznamy o dané tréninkové jednotce a poskytuje zpětnou analýzu prováděné činnosti v určitém časovém rozmezí. Za tréninkový deník můžeme částečně považovat i aplikace, které jsou používány na chytrých mobilních zařízeních. Tyhle aplikace jsou stavěny převážně pro rekreační sportovce, kterým nabízí zaznamenávání dostatečného množství informací. Uživatelé si na nich můžou nastavit mód pro cyklistiku, běh, bruslení apod. a systém na základě senzorů (akcelerometr, gyroskop, snímač tepové frekvence, ...) zaznamenává prováděnou aktivitu. Výsledkem je analýza prováděné činnosti za určitý časový interval. Tyto aplikace mívají pro vrcholové sportovce značná úskalí. Pokud bychom vzali tréninkovou jednotku zaměřenou na běh, aplikace bude zaznamenávat data naměřená za celý trénink v jednotné podobě a budou omezená vlastnostmi systému. V praxi ale můžeme mít běžeckou tréninkovou jednotku rozdělenou na několik fází a v každé fázi měřit různé úseky. Pokud praktikujeme sport zaměřený na silově rychlostní disciplíny, budeme muset zařadit do tréninku i posilování a tady už si nevystačíme s předdefinovanými možnostmi aplikace. Každý sportovec používá různé metody a cviky pro zvýšení své výkonnosti. V rámci této bakalářské práce byla vyvinuta webová aplikace umožňující uživateli ukládat záznamy podle jeho předdefinovaných možností a porovnávat jednotlivé tréninkové jednotky včetně stravování. Uložené záznamy se dělí do tří kategorii.

- **Silová část:** ukládá a zobrazuje data pro posilovací tréninkovou jednotku
- **Běžecká část:** ukládá a zobrazuje data pro běžeckou tréninkovou jednotku
- **Nutriční:** ukládá a zobrazuje data o nutriční hodnotách stravování

2 Teoretická část

Teoretická část práce se v úvodu zaměřuje na důvody vzniku tréninkového deníku a motivaci pro tvorbu sportovních aplikací. Jsou zde uvedeny příklady a srovnání aplikací využívaných v rámci sportu, které jsou obdobou vyvíjené aplikace. Do teoretické části byly zahrnuty i uživatelské požadavky znázorněné pomocí diagramu případu užití a přehled základní funkcionality, kterou vyvíjená aplikace disponuje.

2.1 Motivace

Využití chytrých aplikací pro sportování je velkým trendem dnešní doby. Je pravdou, že tento trend převládá spíše u začínajících sportovců, jelikož se snaží být moderní a objevují více nové věci. Pro zkušenější sportovce taková aplikace nemusí být dostatečně chytrá. Pokud bychom uvažovali sportovce, kteří se připravují poctivě na nějaké závody, budou si zaznamenávat údaje o své výkonnosti (časy při běhání, nutriční hodnoty jídel, ...) do sportovního deníku, který jim umožní analyzovat své tréninky podle zaznamenaných dat. Rozdíl mezi tréninkovým deníkem a chytrou aplikací bude popsán v následující kapitole. Tréninkový deník používají sportovci se zaměřením na různé sporty. Můžou to být běžci, kulturisté nebo hokejisté. Všichni patří do skupiny komplexních sportovců, kterým se někdy také říká „atleti“. Jsou tak nazývání podle sportovního odvětví „Atletika“ zaměřující se na základní pohybové vlastnosti člověka (běhy, skoky, ...). Odtud vznikl název „Deník atleta“ pro aplikaci, která cílí na skupinu komplexních sportovců.

2.2 Srovnání sportovních aplikací

Aplikací pro zaznamenávání sportovních aktivit je celá řada. Obvykle se jedná o mobilní aplikace zdarma dostupné na Google Play. Ty můžou být dvojího typu.

- **Aplikace pro měření aktivity:** ukládá a analyzuje informace o tréninku podle naměřených dat ze zařízení. Data jsou získávány během činnosti jako je běh, lyžování nebo cvičení pomocí senzorů umístěných v blízkosti těla. Uživatel si v aplikaci nastaví typ činnosti, kterou bude vykonávat, spustí začátek tréninkové fáze a na konci ji zase ukončí. Během této doby budou data zaznamenávány do aplikace a na konci tréninku vyhodnoceny.

- **Tréninkový deník:** ukládá a analyzuje data z údajů vložených uživatelem. Aplikace nejsou závislé na aktuálně prováděné činnosti a poskytují větší svobodu zápisu dat. Některé placené verze nabízí i plánování tréninků na určité časové období a zobrazení dat ve skupině sportovců. V takovém případě jde o rozsáhlejší systém využívaný například sportovními kluby.

Ke srovnání byly vybrány aplikace Strava, Runkeeper a GymRun. První dvě spadají do kategorie aplikací pro měření aktivity. GymRun patří do skupiny tréninkových deníků.

2.2.1 Strava

Název je odvozen od švédského slova „sträva“, v překladu znamená „usilovat“. Nejedná se tedy o aplikaci, která se stará o naše stravování. Slouží primárně pro zaznamenávání a analytiku dat běžeckých a cyklistických tréninků. Postupným vývojem v aplikaci přibyly další typy aktivit pro zaznamenávání dat jako jsou plavání, lyžování nebo bruslení. Základní funkcionalita umožňuje během tréninku zaznamenávat čas, vzdálenost, tempo a zobrazovat aktuální mapu s místem, kde se sportovec nachází. Strava[1] je oblíbená i mezi profesionálními sportovci k vůli své přesné citlivosti na zastavení při běhu nebo jízdy na kole. Tím zaručuje minimální zkreslení statistik z prováděného tréninku. Dále nabízí možnost srovnání sportovních výkonů s ostatními uživateli a je kompatibilní s mnoha zařízeními.

2.2.2 Runkeeper

Podobně jako Strava vznikl Runkeeper[2] především pro běh. Nyní obsahuje přes 30 dalších aktivit. Informace o běžeckém tréninku lze zaznamenávat pomocí GPS na mapě. Výsledné statistiky zobrazují celkový čas, vzdálenost, spálené kalorie a v případě běhu na horách i přehled stoupání na jednotlivých úsecích. Placená verze je rozšířena o možnost plánování tréninků, poskytuje podrobnější analýzu a uživatel může sdílet svoji aktuální pohybovou aktivitu v reálném čase ostatním uživatelům.

2.2.3 GymRun

Aplikace typu tréninkového deníku. Uživatel si definuje tréninkové plány obsahující rutiny, ze kterých jsou vyhodnocovány statistiky. Rutinou může být seznam posilovacích cviků, běh nebo kruhový kardio trénink. Seznam cviků je možno vybírat podle kategorii. Ukládání záznamů je mnohem detailnější než v předchozích dvou aplikacích. Uživatel si může plánovat tréninky, vytvářet přesné posilovací sekce i typy běžeckých tréninků. Nevýhodou je ruční zadávání údajů do aplikace. GymRun[3] je aplikace zaměřená na posilování a oproti aplikacím Strava a Runkeeper neumožňuje zaznamenávat údaje v rámci tréninku.

2.2.4 Závěr srovnání

V kapitole byl popsán rozdíl mezi aplikací typu tréninkového deníku a aplikací pro měření aktivity sportovců. Dále byly vybrány tři typy aplikací z těchto skupin Strava, Runkeeper a GymRun. Výsledným srovnáním nelze říci, který typ je lepší nebo horší. Záleží na požadavcích uživatelů. Pro využití v rámci sportovní činnosti se více hodí aplikace pro měření aktivity, která bude zaznamenávat prováděnou činnost. Pro detailnější informace o tréninku zase aplikace typu tréninkového deníku. Dá se ale říci, že tréninkové deníky budou sportovce vést k

[1] https://www.strava.com/
[2] https://runkeeper.com/
[3] https://www.gymrun.app/

lepším výkonům díky množství uložených informací a aplikace pro měření aktivit na základě dat poskytovaných během pohybové aktivity.

2.3 Cíle

Před zahájením vývoje webové aplikace byly stanoveny požadavky, které by aplikace měla splňovat. Jsou to použitelnost, jednoduchost, bezpečnost a rychlost. Stanovené požadavky jsou popsány v následujících podkapitolách.

2.3.1 Použitelnost

Aplikace by měla najít využití v praxi mezi vrcholovými i rekreačními sportovci. Obsahuje rozdělení na hlavní tréninkové jednotky, které jsou využívány v mnoha typech sportů. Oproti aplikacím v mobilních zařízení její analytická část bude poskytovat dostatečnou zpětnou vazbu pro zlepšení fyzických vlastností jednotlivců.

2.3.2 Jednoduchost

Při vývoji byl kladen důraz na grafické uživatelské rozhraní s minimálním počtem komponent. Práce s prostředím by měla být přehledná, jednoduchá a vkládání záznamů do aplikace intuitivní. Uživatel pro zadání údajů do aplikace bude potřebovat minimální počet kroků. Uložené záznamy budou přehledně zobrazovány na úvodních sekcích jednotlivých typů tréninku.

2.3.3 Bezpečnost

Využívání aplikace pro velké množství uživatelů musí být zabezpečené. O správu a zabezpečení uživatelů se stará cloudový software od společnosti Okta integrovaný do aplikace. Registrací se vytvoří uživateli záznam na vzdáleném úložišti s vygenerovaným jedinečným klíčem pomocí kterého následně zobrazuje, ukládá a mění data lokální databáze.

2.3.4 Rychlost

Pro rychlou interakci mezi uživatelem a aplikací byla na webové části zvolená technologie Angular, která eliminuje načítání celkového obsahu webové stránky. Vhodně navržená databázová struktura a optimalizované dotazy budou zaručovat rychlou odezvu pro zobrazení uložených dat a práci s nimi.

2.4 Funkce webové aplikace

Návrh aplikace předpokládá co nejkomplexnější podporu sportovců, jak začínajících, tak profesionálních. Aplikace spadá do kategorie tréninkových deníků, jejichž základním principem je evidence dat a zobrazování přehledové statistiky. Podrobnější popis jednotlivých funkcionalit aplikace bude rozepsán níže.

- **Ukládání dat**
  Aplikace eviduje vstupní hodnoty zadané uživatelem a kategorizuje je do třech částí. K uložení dat týkajících se provedených posilovacích cviků s počtem opakování, zátěží a zaměřením na konkrétní svalovou partii slouží část silová. Další částí je běžecká, ve které uživatel ukládá data o jednotlivých typech běhů, vzdálenostech a uběhnutých časech. Po vyplnění těchto informací jsou jednotlivé tréninky uloženy uživatelem do aplikace. V nutriční části aplikace eviduje množinu nutričních hodnot potravin (celých jídel), které uživatel vyhledal nebo sám vytvořil.

- **Statistika**
  Statistické údaje jsou zobrazeny na stránce Dashboard v podobě sloupcových a kruhových grafů. Zdrojem pro vyhodnocení statistiky jsou data uložená v rámci silového a běžeckého tréninku. Statistika je vyhodnocena ve dvou grafech, první graf zobrazuje počty silových a běžeckých tréninků. Druhý graf se zaměřuje na procentuální poměr složek silových tréninků a procentuální poměr typů běhu v rámci běžeckých tréninků.

- **Analýza**
  Aplikace uživateli poskytuje vyhodnocování dat na základě údajů uložených v jednotlivých sekcích. Mezi tyto údaje patří například výpočet rychlosti běhů, celková uzvednutá váha v rámci jednotlivých cviků nebo příjem nutričních hodnot podle výběru potravin v rámci jednoho dne.

2.5 Uživatelské požadavky

V rámci vývoje aplikace proběhl sběr požadavků od budoucích uživatelů, kterými byli sportovci působící od rekreační až po profesionální úroveň z různých sportovních odvětví. Sběr požadavků probíhal konzultacemi a byl podkladem pro tvorbu diagramu případu užití.

2.5.1 Analýza uživatelských požadavků

Základním požadavkem byla komplexnost aplikace, tak aby aplikace nebyla vázána na konkrétní skupinu sportovců a zbytečně je nelimitovala. Další požadavky sportovců obsahovaly tvorbu jídelníčku na základě výpočtu nutričních hodnot potravin, přidávání poznámek k jednotlivým tréninkům, evidenci typů tréninků a charakteristiky jednotlivých složek tréninků. V neposlední řadě byl vznesen požadavek na možnost vytvoření a editaci uživatelského nastavení v rámci individuálních potřeb sportovce.

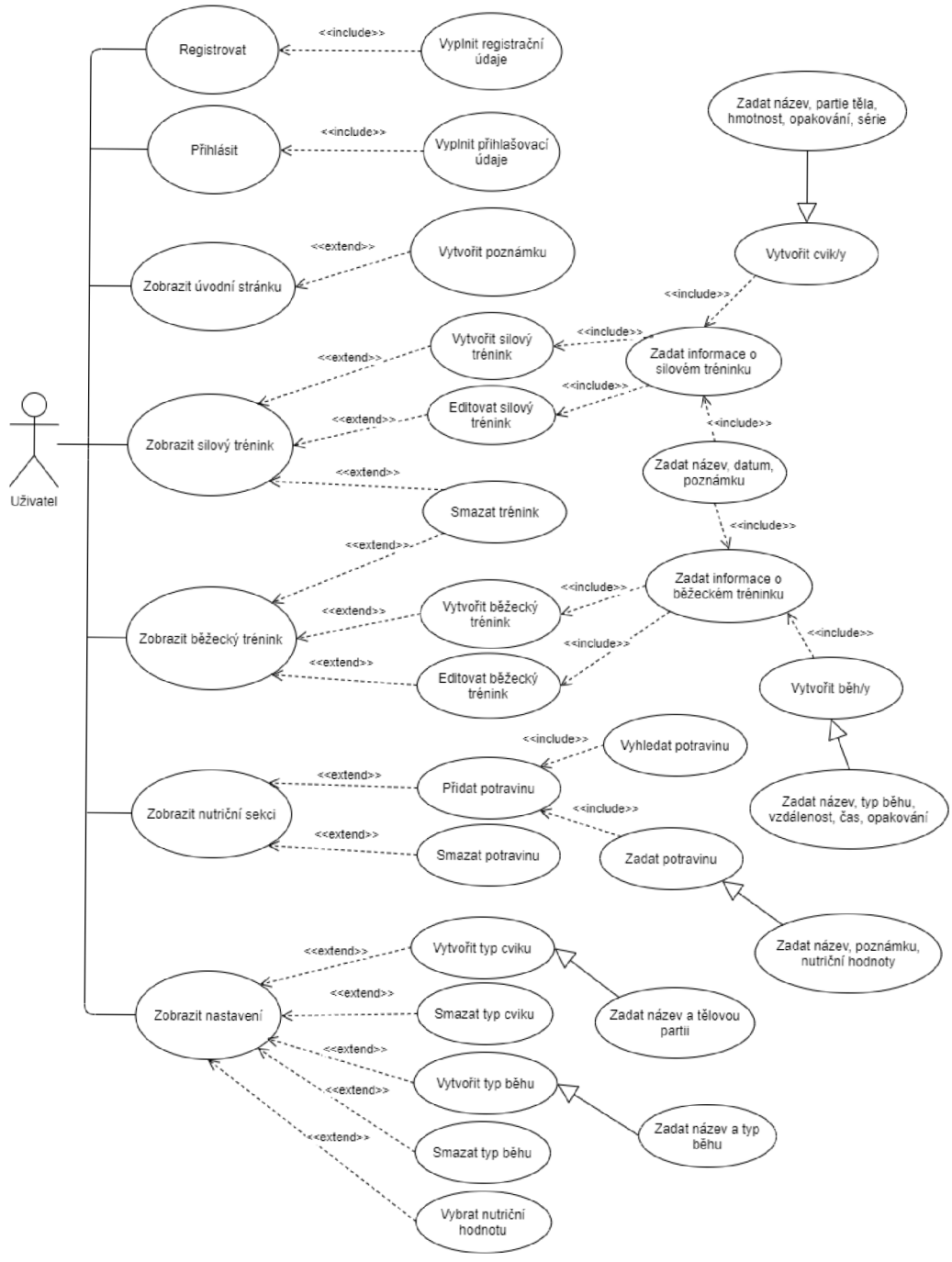
2.5.2 Případy užití

Případy užití znázorňují seznam kroků, které můžou být prováděny uživatelem v aplikaci. Jednotlivé kroky jsou zde popsány a graficky znázorněny na UseCase diagramu.

- **Registrovat:** Prvotní interakce mezi uživatelem a aplikací. Uživatel si vytváří účet, kterým se bude přihlašovat do aplikace.
- **Vyplnit registrační údaje:** K registraci do aplikace musí uživatel vyplnit registrační údaje jako jsou e-mail, heslo, jméno a příjmení.
- **Přihlásit:** Pro vstup do aplikace se uživatel musí přihlásit. Přihlášení nemusí být nutné vždy, pouze pokud uživateli vyprší doba platnosti.
- **Vyplnit přihlašovací údaje:** K přihlášení uživatele do aplikace je potřeba vyplnit přihlašovací e-mail a heslo.
- **Zobrazit úvodní stránku:** Umožňuje uživateli zobrazit stránku „Dashboard“ zobrazující grafy se statistikou tréninků a „Timeline“ pro přehled uložených poznámek.
- **Přidat poznámku:** Úvodní stránka poskytuje uživateli možnost přidat obecnou poznámku do tréninkového deníku.
- **Zobrazit silový trénink:** Jedná se o sekci „Power training“, kde si uživatel zobrazuje uložené posilovací tréninky. V této sekci může také vytvářet nové záznamy a editovat nebo mazat již existující uložené.
- **Vytvořit silový trénink:** Umožňuje založit nový záznam pro silový trénink. Silový trénink obsahuje základní informace o vytvoření, poznámku a seznam cviků.
- **Zadat informace o silovém tréninku:** Vytvoření silového tréninku vyžaduje vyplnění informací týkajících se tréninku jako jsou název, datum absolvování tréninku, poznámku a seznam cviků.
- **Vytvořit cvik/y:** Umožňuje uživateli v rámci definice silového tréninku vytvořit seznam cviků. Seznam cviků je definován vytvořením a přidáním nového cviku definovaného uživatelem.
- **Zadat název, partie těla, hmotnost, opakování, série:** Vyplnění údajů uživatelem pro přidání nového cviku v rámci silového tréninku.
- **Editovat silový trénink:** Umožňuje měnit data a spravovat seznam cviků pro již vytvořené posilovací tréninky.
- **Smazat trénink:** Umožňuje uživateli smazat silový nebo běžecký trénink.

- **Zobrazit běžecký trénink:** Jedná se o sekci „Run training“, kde si uživatel zobrazuje uložené běžecké tréninky. V této sekci může také vytvářet nové záznamy a editovat nebo mazat již existující uložené.
- **Vytvořit běžecký trénink:** Umožňuje založit nový záznam pro běžecký trénink. Ten obsahuje informace o vytvoření, poznámku a seznam nadefinovaných běhů.
- **Zadat informace o běžeckém tréninku:** Vytvoření běžeckého tréninku vyžaduje vyplnění několika základních informací. Jsou to název, datum absolvování tréninku, poznámka a seznam absolvovaných běhů.
- **Zadat název, datum, poznámku:** Vyplnění základních údajů uživatelem při vytváření posilovacího nebo běžeckého tréninku.
- **Vytvořit běh/y:** Umožňuje uživateli v rámci definice běžeckého tréninku vytvořit seznam běhů. Ten vzniká vytvořením a přidáním nového běhu definovaného uživatelem.
- **Zadat název, typ běhu, vzdálenost, čas, opakování:** Vyplnění údajů uživatelem pro definici nového běhu.
- **Zobrazit nutriční sekci:** Jedná se o sekci „Nutrition“, kde si uživatel zobrazuje uložené potraviny a jejich nutriční hodnoty. V této sekci také vytváří nové záznamy podle vlastní definice nebo dle vyhledání v databázi, které lze následně smazat.
- **Přidat potravinu:** Umožňuje uživateli přidat novou potravinu do deníku skrze vyhledání v databázi nebo vlastním vyplněním údajů o potravině.
- **Vyhledat potravinu:** Uživatel vyhledá potravinu v databázi.
- **Zadat potravinu:** Uživatel si sám definuje informace o potravině.
- **Zadat název, poznámku, nutriční hodnoty:** Uživatel vyplní název, doplňující informace a nutriční údaje o potravině.
- **Smazat potravinu:** Uložené potraviny uživatel může smazat.
- **Zobrazit nastavení:** Jedná se o sekci „Settings“, kde si uživatel zobrazuje a mění aktuální uživatelské nastavení aplikace.
- **Vytvořit typ cviku:** Uživatel v aplikaci vytváří typy cviku, který mu bude nabízen v rámci tvorby posilovacího tréninku při vytváření nového cviku.
- **Zadat název a tělovou partii:** U tvorby nového typu cviku musí uživatel zadat název a svalovou partii pro kterou je cvik určen.

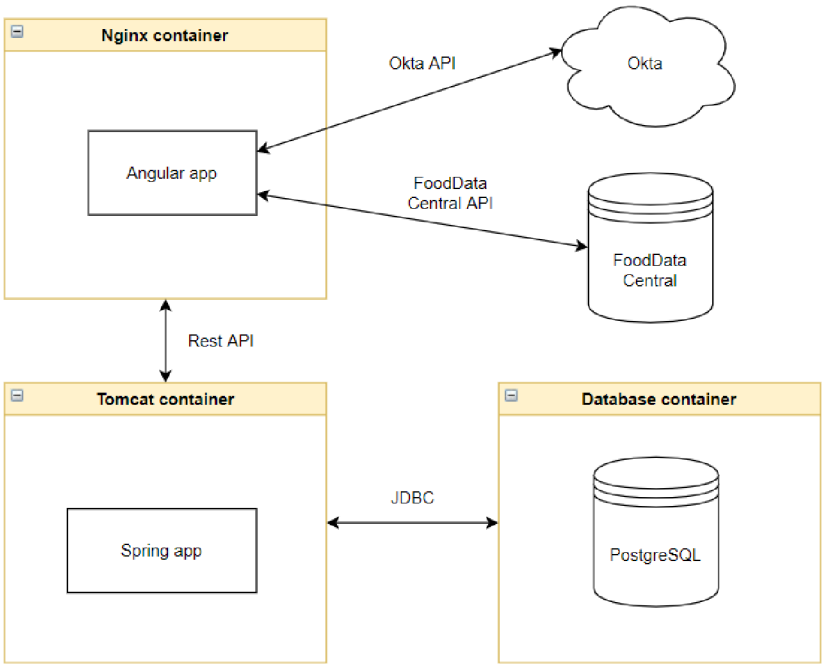
- **Smazat typ cviku:** Uživatel smaže uložený typ cviku v uživatelském nastavení.
- **Vytvořit typ běhu:** Umožňuje uživateli vytvořit typ běhu, který mu bude nabízen při vytváření běhu v rámci tvorby běžeckého tréninku.
- **Zadat název a typ běhu:** Při vytváření typu běhu v uživatelském nastavení musí uživatel zadat název a typ běhu.
- **Smazat typ běhu:** Uživatel smaže uložený typ běhu v uživatelském nastavení.
- **Vybrat nutriční hodnotu:** Uživatel vybírá ze seznamu nutričních hodnot ty, které chce zobrazovat v aplikaci.



Obrázek 1: Diagram případu užití aplikace

2.6 Použité technologie

Ke splnění výše stanovených cílů bylo potřeba zvolit vhodné technologie k sestavení aplikace. Tyto technologie jsou znázorněny v následujícím obrázku.



Obrázek 2: Architektura aplikace

- **Web:** Funkcionalitu webové části obstarává framework Angular, který využívá zabezpečení cloudového softwaru od společnosti Okta. Software kontroluje přístupy na jednotlivé sekce tréninkového deníku a v případě, kdy uživatel neprojde autentizací, zamezí se přístupu do aplikace. Okta také poskytuje úložiště pro registrované uživatelé a na požadavky vrací odpovědi s jejich informacemi. Nutriční sekce tréninkového deníku obsahuje vyhledávač v databázi FoodData Central pro vyhledání potravin s širokou škálou nutričních hodnot.
- **Backend:** Část, kterou tvoří aplikace Spring boot běžící na Tomcat serveru a PostgreSQL databáze. Spring boot aplikace odesílá a přijímá data skrze Rest API z webového obsahu. Data jsou následně zpracovány a uloženy do PostreSQL databáze, která je napojena na Spring boot aplikaci pomocí JDBC.

Angular aplikace, Spring boot aplikace a databáze jsou na provozním prostředí spuštěny v oddělených běhových prostředích pomocí docker kontejnerů.

3 Angular

Vývojová platforma postavená na programovacím jazyce TypeScript. Vytvořila ji společnost Google v roce 2009, která je v dnešní době největším přispěvatelem k rozvoji této platformy. Angular využívá více než 1,7 milionů vývojářů, autorů knihoven a tvůrců obsahu, kteří platformu dále rozvíjí o své vlastní knihovny a používají ji pro tvorbu svých produktů. Své využití najde mezi jednotlivci i rozsáhlými podniky. Soubor knihoven pokrývá velkou škálu funkcí jako je např. přesměrování, správa formulářů nebo komunikace mezi klientem a serverem.

Slouží pro vytváření jednostránkových aplikací interagujících s uživatelem dynamickým přepisováním aktuálního webového obsahu na základě dat přijatých z webového serveru. Prohlížeč tak nemusí načítat celou webovou stránku a přechody mezi stránkami jsou mnohem rychlejší. Základními stavebními bloky aplikace vytvořenou tímto rámcem (dále frameworkem) jsou komponenty organizované do modulů (NgModules).

3.1 Moduly

Moduly slouží jako kontejnery, který sdružují logické bloky kódů. Obsahují komponenty, služby a další soubory kódu. Mohou importovat funkce z jiných modulů a exportovat funkce pro další moduly. Díky modulárnímu systému je možno aplikaci logicky strukturovat a rozšiřovat ji o další funkcionalitu. Aplikace vždy obsahuje alespoň jeden kořenový modul, který ji spouští. Takovýto modul se nazývá „AppModule“ a nachází se v souboru *app.module.ts*. Modul je definován pomocí dekorátoru třídy*@NgModule()*, což je funkce, která bere objekt metadat popisujících vlastnosti modulu. Soubor definující modul se skládá z importů knihoven, definice metadat a exportu třídy. V následujícím příkladu je zobrazen kořenový modul s konfigurací metadat dekorátoru *@NgModule()*.

```
1  import { NgModule } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3  @NgModule({
4     imports:   [ BrowserModule ],
5     providers: [ Logger ],
6     declarations: [ AppComponent ],
7     exports:   [ AppComponent ],
8     bootstrap: [ AppComponent ]
9  })
10 export class AppModule {
11 }
```

Zdrojový kód 1: Definice kořenového modulu **AppModule**

- **Declarations:** definují sadu komponent, direktiv a kanálů, které patří tomuto modulu.

- **Exports:** jedná se o sadu komponent, direktiv a kanálů deklarovaných v tomto modulu. Exportované deklarace jsou rozhraním aplikace pro programování (API). Lze je tedy použít v šabloně jakékoli komponenty, jenž je součástí modulu, který importuje tento modul.
- **Imports:** sada modulů, jejichž exportované třídy jsou k dispozici pro šablony v tomto modulu.
- **Providers:** poskytovatelé (třídy), kteří budou k dispozici pro injekci do jakékoli komponenty, direktiv, kanálu nebo služby, která je podřízeným prvkem tohoto injektoru
- **Bootstrap:** zobrazení hlavní kořenové komponenty aplikace

Moduly můžou obsahovat libovolný počet komponent, kterým poskytují kontext pro jejich kompilaci.[1]

3.2 Komponenty

Stejně jako každá Angular aplikace obsahuje jeden kořenový modul, musí obsahovat i jednu kořenovou komponentu. Tahle komponenta spojuje hierarchii komponent s objektovým modelem webové stránky (DOM). Jejich úkolem je ovládat část obrazovky nazývané „view“ (pohled). View tvoří třída komponenty a její přidružená šablona. Interakce mezi view a třídou probíhá prostřednictvím API a příslušných metod, které můžou být definovány uvnitř třídy nebo poskytovány ze služeb injektovaných pomocí závislostí.[2]

3.2.1 Definice tříd a metadat

Pro vytvoření komponenty je zapotřebí definovat dekorátor označený anotací *@Component()* nad příslušnou třídou. Dekorátor slouží pro identifikaci třídy a specifikaci metadat komponenty. Pokud tedy neoznačíme třídu tímto dekorátorem, Angular ji bude považovat pouze za obyčejnou třídu a neumožní získat komponentě hlavní stavební prvky k jejímu vytvoření. Pro příklad je uvedena třída *AppComponent* s konfigurací metadat dekorátoru *@Component()*.

- **selector:** definuje název HTML tagu pro komponentu. Angular na základě selektoru vytvoří instanci a vloží šablonu komponenty na místo určené tímto tagem.
- **templateUrl:** relativní adresa HTML šablony nebo přímo vložený HTML kód.
- **providers:** pole poskytovatelů služeb, které komponenta využívá.
- **styleUrls:** pole relativních cest k souboru s definovanými CSS styly pro vytvářenou komponentu

```
@Component({
   selector: 'app-component',
   templateUrl: './app.component.html',
   styleUrls: ['./app.component.css'],
   providers: [ AppService ]
})
export class AppComponent implements OnInit {
/* . . . */
}
```

Zdrojový kód 2: Definice třídy komponenty **AppComponent**

3.2.2 Šablony a pohledy

Pohledy (view) jsou seskupením prvků, které lze společně vytvořit a odstranit. Tvoří je třída komponenty dohromady se šablonou. Tyto šablony komponent jsou jen částmi HTML, které se vkládají do vykresleného obsahu webové stránky. Rozšířením o syntaxi Angular je umožněno aplikaci dynamicky měnit strukturu obsahu na základě stavu dat, logiky a DOM. Jelikož se jedná o část webové stránky, nikoli o celou stránku, nemusí obsahovat prvky *<html>*, *<body>*, apod..[4]

V následujícím příkladu je uvedena šablona, která na základě dat komponenty zobrazí seznam poznámek. Tvoří ji základní HTML elementy *<ul>*, *<li>*, * *, **, *<h3>* a *<p>*. Elementy mají přiřazené třídy sloužící pro kaskádové styly provázané s komponentou šablony. Direktiva **ngFor* se stará o iteraci nad atributem *notes*, který je definován jako pole objektů typu *Note*. Ten obsahuje atributy *createdDate* znázorňující datum vytvoření poznámky a *text* obsahující text poznámky. Syntaxe dvojitě složených závorek umožní programu navázat data programu do DOM (funkcionalita vázání dat bude více přiblížená v kapitole 3.2.3).

```
<ul class="timeline">
   <li class="timeline-item" *ngFor="let note of notes">
      <div class="timeline-info">
         <span> {{ note.createdDate | date:'dd.MM.yyyy'}} </span>
      </div>
      <div class="timeline-content">
         <p> {{note.text}} </p>
      </div>
   </li>
</ul>
```

Zdrojový kód 3: Šablona pro výpis poznámek

[4]Pro bezpečnost aplikace a zamezení útoků s vložením skriptu, Angular nepodporuje prvek *<script>*. Aplikace tento prvek ignoruje a zahlásí varování do konzole prohlížeče.

3.2.3 Vazba dat

Jedná se o mechanismus sloužící k synchronizaci dat mezi částmi šablony a komponenty[3]. Uživatel webové stránky tak bude mít vždy zobrazen aktuální obsah. K synchronizaci slouží mechanismus detekce změn, který prochází strom komponent a aktualizuje DOM. Podle směru odesílání dat rozlišujeme komunikaci:

1. **Jednosměrnou od zdroje k zobrazení v šabloně**
   Používá se například pro zobrazení textu v šabloně pomocí interpolace *{{expression}}* nebo přiřazení vlastnosti elementu šablony *[value]="expression"*.

2. **Jednosměrnou od zobrazení v šabloně ke zdroji**
   K odesílání událostí z šablony ke komponentě. Příkladem události může být stisk tlačítka *(click)="showDialog()"*, které vyvolá zobrazení dialogu. Změna hodnoty ve výběrovém menu *(selectionChange)="onChange()"* nebo i posun obrazovky.

3. **Obousměrnou**
   Je kombinací dvou předešlých. Obvykle se používá k provázání hodnoty prvku v DOM a atributu komponenty. Syntakticky se zapisuje složením *[ ]* závorek do kterých se vkládají *( )* závorky. Příkladem takového zápisu může být *[(target)]="expression"*.

3.3 Služby a vkládání závislostí

Služba je třída obsahující kód, který lze využít i v několika různých komponentách. Poskytuje konkrétní funkce, které přímo nesouvisejí s pohledy. V ideálním případě by komponenta měla obsahovat pouze metody obstarávající funkcionalitu stránky. Samotná logika aplikace má být odstíněná a poskytována formou služby vybrané komponentě. Ty můžou delegovat určité úkoly na služby, jako je načítání dat ze serveru, ověřování vstupu uživatele nebo k zaznamenávání dat do logu.[5]

Poskytování služeb komponentám probíhá formou vkládání závislostí "Dependency injection (DI)". Aby framework poznal, že se jedná o službu, musí být třída označená dekorátorem *@Injectable()*, který umožní vložení třídy služby ke komponentě pomocí závislosti.

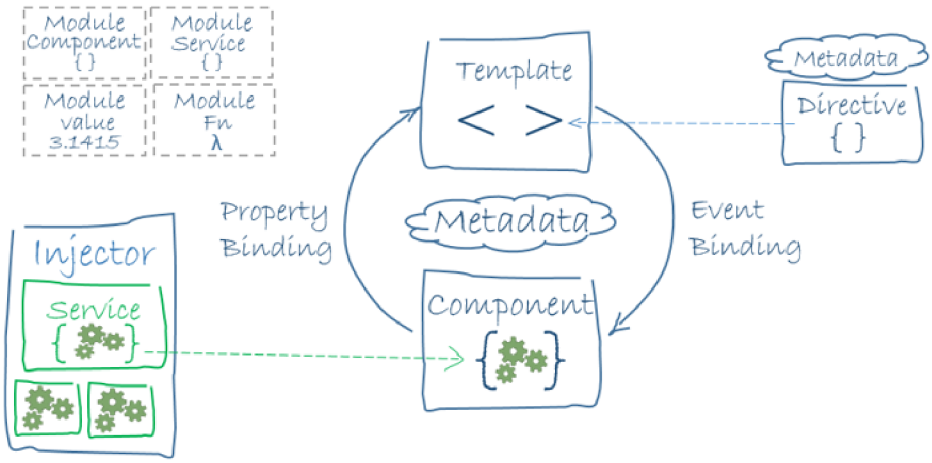
[5]Log (též žurnál) je v informatice obecné označení pro záznam nějaké činnosti nebo pro soubor se záznamy (často s příponou .log), které některé programy vytvářejí pro záznam informací o své činnosti a běhu.

```
@Component({
  selector: 'app-setting',
  templateUrl: './setting.component.html'
})
export class SettingComponent implements OnInit {
    constructor(private settingService: SettingService) {
    }
}
```

Zdrojový kód 4: Injektace služby pro komponentu

3.4 Koncept

Pro ucelený náhled na architekturu Angular aplikace jsou v následujícím obrázku zobrazeny jeho základní stavební bloky. Šablona znázorněná jako *Template* tvoří spolu s komponentou *Component* pohled. Direktivy *Directive* upravují pohled šablony a injektor závislostí *Injector* poskytuje služby komponentě. Množina těchto prvků dohromady tvoří modul, ke kterému můžou být importovány další knihovny.[4]



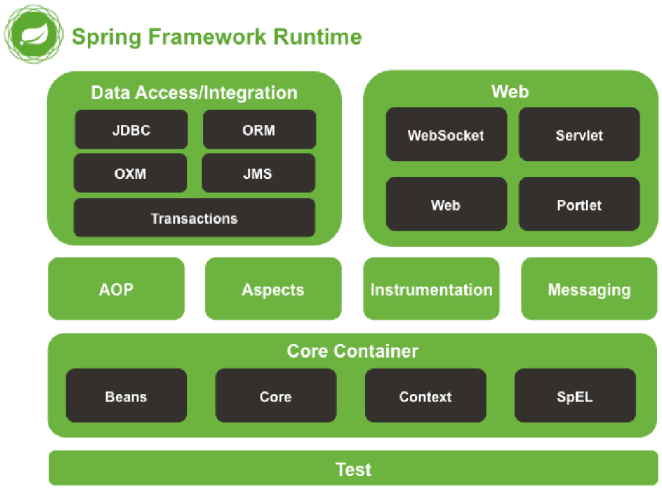
Obrázek 3: Základní stavební bloky Angular aplikace

4 Spring

V této kapitole je popsán Spring framework, který se používá k vývoji podnikových aplikací a informačních systémů postavených na programovacím jazyce Java. Vznikl v roce 2003 jako alternativa k Java EE,[6] která byla v té době velmi těžkopádná a konfiguračně složitá. V posledních letech se rozdíl prakticky smazal, ale Spring se již stihl etablovat jako plnohodnotná alternativa. Programovací model neobsahuje celou specifikaci platformy Java EE, ale integruje se s vybranými částmi specifikací. Hlavními důvody vzniku jsou:

- Odstranění POJO (Plain old Java object) pomocí návrhové vzoru Inversion of Control.
- Možnost volby implementace business vrstvy pro aplikační architekturu.
- Řešení aplikačních domén bez nutnosti použití EJB (transakční zpracování, RMI).
- Podpora implementace komponent pro přístup k datům skrze JDBC i ORM.
- Usnadnění používání a psaní unit testů.
- Správa a konfigurační management business komponent.

Spring je složen z funkcí uspořádaných do přibližně 20 modulů. Tyto moduly jsou seskupeny do jednotlivých částí (Core Container, Data Access/Integration, Web, AOP - Aspect Oriented Programming, Instrumentation, Messaging a Test).[5]



Obrázek 4: Spring - běhové prostředí

[6] J2EE (Jakarta EE) je sada specifikací rozšiřujících javu SE.

4.1 Core kontejner

Moduly Beans a Core jsou základními bloky tohoto frameworku. Poskytují funkcionalitu inverze kontroly 4.6 a vkládání závislostí 4.7. Beans také umožňuje oddělit konfiguraci a specifikaci závislostí od skutečné logiky programu.

Context je prostředek pro přístup k objektům ve stylu rozhraní podobný JNDI. Své funkce dědí z modulu Beans. Podporuje také funkce Java EE, jako je EJB, JMX a základní vzdálená komunikace. Hlavním bodem modulu je rozhraní „ApplicationContext“.

Výrazový jazyk (SpEL) slouží pro dotazování a manipulaci s objektovým grafem[7] za běhu.

4.2 AOP a instrumentace

Aspektově orientované programování (AOP) je rozšířením objektového programování. Dělí program na části, které se mezi sebou co nejméně překrývají svou funkcionalitou.

Instrumentace umožňuje vkládat do programu instrukce pro získání dat k analýze. Je to jeden ze způsobů profilování programu. Příkladem může být počítání volání metody nebo měření času.

4.3 Zprávy

Spring obsahuje podporu pro integraci se systémy pro zasílání zpráv. Informace jsou asynchronně zpracovávány a posílání dat mezi různými komponentami probíhá ve formě zpráv. Jedním z možných řešení může být JMS.

4.4 Přístup a integrace dat

Slouží pro přístup k datům a interakci mezi vrstvou pro přístup k datům a obchodní nebo servisní vrstvou. Skládá se z modulů JDBC, ORM, OXM, JMS a transakčního modulu.

- **Java Database Connectivity (JDBC)**
  API programovacího jazyka Java, které definuje jak může klient přistupovat k databázi. Poskytuje metody pro dotazování a aktualizací dat nad relační databází.

- **Objektově relační zobrazení (ORM)**
  V relační databázi bývá entita reprezentována jako řádek v tabulce, případně více řádků v několika tabulkách. V objektově orientovaném jazyce je entita reprezentována jako instance nějaké třídy. ORM zajišťuje automatickou konverzi dat mezi relační databází a objektově orientovaným programovacím jazykem, včetně jejich datových typů a dědičnosti.

[7] Objektový graf je pohled na objektový systému v určitém časovém okamžiku

- **Mapování objektů XML (OXM)**
  Modul poskytující mapování objektů do XML a z XML. Proces je také známý jako „XML Marshalling“ nebo „XML Serialization“. Spring poskytuje několik frameworků, které umožňují vývojářům mapovat třídy do reprezentace XML jako například JAXB, XMLBeans nebo JiBX.

- **Služba zpráv Java (JMS)**
  Služba umožňuje vytvářet, posílat, přijímat a číst zprávy posílané mezi aplikacemi. Díky tomu můžou aplikace fungovat společně jako jeden systém. Zpráva je složena z hlavičky pro identifikaci zprávy, vlastní určujících nastavení a těla pro obsah zprávy. Spring poskytuje JMS framework, který usnadňuje práci s JMS API.

- **Transakce**
  Řeší transakce do databáze, které ji převedou z jednoho konzistentního stavu do druhého. Musí splňovat následující vlastnosti:

  - **Atomicita:** příkaz se musí provést jako celek nebo vůbec
  - **Konzistence:** převádí databázi z jednoho konzistentního stavu na druhý
  - **Izolovanost:** operace jedné transakce by neměla ovlivnit operaci druhé transakce.
  - **Trvalost:** změny v databázi, úspěšně provedenou transakcí, nemohou být ztraceny

4.5 Webová vrstva a testovací modul

Webová vrstva poskytuje funkcionalitu pro práci s webovou částí aplikace. Obsahuje základní webově orientované integrační funkce, HTTP klienta a podporu vzdálené komunikace. V tréninkovém deníku je webová vrstva nahrazena Angular aplikací popsanou v kapitole 3.

Testovací modul slouží pro testování aplikace (Unit testing, Integration testing).[6]

4.6 Inverze kontroly (IoC)

Princip, který přenáší kontrolu nad objekty nebo části programů na kontejner. Místo toho, aby aplikace volala implementace poskytované knihovnou, volá rozhraní implementace poskytované aplikací. Používá se u objektově orientovaného programování. Ve Springu reprezentuje kontejner IoC rozhraní ApplicationContext. Je zodpovědné za vytváření instancí, konfiguraci a sestavování objektů známých jako beans a za správu jejich životních cyklů. Nejznámnější z forem IoC je vkládání závislostí (Dependency injection).

4.7 Vkládání závislostí (DI)

Anglicky také „Dependency injection“ je proces, při kterém objekty definují své závislosti. To znamená, že ostatní objekty jsou vkládány prostřednictvím argumentů konstruktoru, argumentů metody nebo vlastností, které jsou nastaveny na instanci objektu po jeho vytvoření nebo vrácení. Vytvořením beany[8] kontejner injektuje tyto závislosti. Díky tomuto obrácenému procesu je pojmenován IoC (viz nahoře popis rozdílů DI a IoC). Vkládání závislostí poskytuje výhody v podobě čistoty kódu, větší modularity programu a snadnějšího testování. Vkládání závislostí může probíhat několika způsoby.

- **Vkládání závislostí pomocí konstruktoru**
  Kontejner zavolá konstruktor s argumenty, z nichž každý představuje závislost, kterou chceme nastavit. Na příkladu je znázorněna beana *NutritionRepository*, která je injektována do třídy *NutritionServiceImpl* pomocí konstruktoru.

```
1  public class NutritionServiceImpl implements NutritionService {
2
3      private final NutritionRepository nutritionRepository;
4
5      public NutritionServiceImpl(NutritionRepository
           nutritionRepository) {
6          this.nutritionRepository = nutritionRepository;
7      }
8  }
```

Zdrojový kód 5: Vkládání závislostí pomocí konstruktoru

[8] JavaBean je standart pro třídy napsané v programovacím jazyce Java. Třída musí mít soukromé vlastnosti, veřejný konstruktor bez argumentů a musí implementovat rozhraní Serializable.

- **Vkládání závislostí pomocí setter metody**
  Provádí se pomocí kontejneru volajícího setter metodu po vyvolání konstruktoru nebo statické metody bez argumentů. V následujícím příkladu je zobrazena obyčejná Java třída *SimpleMovieListener*. Do této třídy lze injektovat beanu *MovieFinder* pouze pomocí setter metody *setMovieFinder*.

```
1  public class SimpleMovieLister {
2
3    private MovieFinder movieFinder;
4
5    public void setMovieFinder(MovieFinder movieFinder) {
6       this.movieFinder = movieFinder;
7    }
8
9  }
```

Zdrojový kód 6: Vkládání závislotí pomocí setter metody

- **Vkládání závislostí pomocí @Autowired anotace**
  Používá se v případě, kdy neexistuje konstruktor ani setter metoda, ve které by byla injektována beana. Tento přístup je náročnější jelikož kontejner využívá reflexi pro injektování beany. K injektování beany se používá anotace *@Autowired* nad atributem beany ve třídě. V příkladu je injektována beana *Item* do třídy *Store*.

```
1  public class Store {
2
3  @Autowired
4  private Item item;
5  }
```

Zdrojový kód 7: Vkládání závislostí pomocí @Autowired anotace

4.8 Spring Boot

K usnadnění vývoje aplikace byla použita nadstavba pro Spring, která se jmenuje Spring Boot. Implementuje veškeré funkce Springu a začínajícímu programátorovi ve Springu umožní mnohem rychleji nahlédnout do vývoje aplikace. Jeho velkou výhodou je minimální konfigurace včetně již přednastaveného serveru.[7]

4.9 Architektura aplikace

Programovatelný kód aplikace je možné rozdělit do několika částí, aby se co nejméně překrýval svou funkcionalitou. K tomu Spring využívá komponentový systém. Za komponentu můžeme považovat jakoukoliv třídu ve Springu, která je označená anotací *@Component* nebo jinou specializovanou anotací zahrnující stejnou funkcionalitu. Spring tuhle třídu detekuje pomocí *@ComponentScan* jako beanu, do aplikačního kontextu si uloží její instanci a injektuje ji kdykoli je potřeba. Pro lepší modularitu programu slouží speciální anotace jako jsou *@RestController*, *@Service* a *@Repository*. Díky ním je možné lépe organizovat strukturu aplikace a odlišit tak jednotlivé vrstvy podle logiky.

4.9.1 @SpringBootApplication

Anotace pro hlavní komponentu aplikace definující 3 hlavní funkce:

- *@EnableAutoConfiguration*
  Automaticky nakonfiguruje aplikaci podle vložených závislostí .jar.
- *@ComponentScan*
  Skenování komponent v balíčku, kde je aplikace umístěna.
- *@Configuration*
  Povoluje registraci bean v kontextu (např. pomocí DI) nebo import dalších konfiguračních tříd.

Komponenta označená touto anotací je vstupním bodem pro spuštění aplikace. Obsahuje metodu *main*, která voláním Spring metody *run* spustí automaticky nakonfigurovaný web server Tomcat.

```
1 @SpringBootApplication
2 public class BackendApplication extends SpringBootServletInitializer
      {
3
4    public static void main(String[] args) {
5       SpringApplication.run(BackendApplication.class, args);
6    }
7
8 }
```

Zdrojový kód 8: Třída pro spuštění Spring Boot aplikace

4.9.2 @RestController

Označení komponenty pro zpracování příchozích HTTP požadavků. Metody uvnitř komponenty jsou mapovány na HTTP pomocí anotace *@RequestMapping*. Někdy se také používají anotace přímo určené jednotlivým HTTP požadavkům GET, POST, PUT, DELETE a PATCH. V následujícím příkladu je uvedeno použití dvou takových anotací *@GetMapping* a *@PostMapping*.

Obě anotace mají definovanou adresu *value*, kde budou zpracovávat požadavky a v jakém tvaru budou zpracovávat data *produces*. *@RequestParam* a *@RequestBody* slouží pro získání dat v požadavku.

```
1  @RestController
2  public class PowerTrainingController {
3
4      @GetMapping(value = "/power-training/trainings", produces =
           MediaType.APPLICATION_JSON_VALUE)
5      public List<PowerTrainingDTO> getPowerTrainings(@RequestParam("
           userId") String userId) {
6          // ...
7      }
8
9      @PostMapping(value = "/power-training/save-training", produces =
           MediaType.APPLICATION_JSON_VALUE)
10     public PowerTrainingDTO saveTraining(@RequestBody
           PowerTrainingToUserDTO training) {
11         // ...
12     }
13 }
```

Zdrojový kód 9: *@RestController* pro zpracování HTTP požadavků

4.9.3 @Service

Definuje komponentu, která slouží pouze pro oddělení business logiky aplikace.

4.9.4 @Repository

Komponenta označená anotací *@Repository* umožňuje definovat třídu pro manipulaci s daty uložené v databázi. V aplikaci tak oddělí business logiku a data. Jedním ze způsobů pro práci s daty je využití Hibernate frameworku, který je implementací JPA a umožňuje objektově relační mapování. Objekty mapované na záznamy v databázi se nazývají entity. Pro specifikaci objektu třída využívá anotace jako *@Entity*, *@Id*, *@Column* a další určující mapované hodnoty. Na příkladu je znázorněna entita *NutrientItem* s atributy *id, name, usdaNutrientId, unitName a defaultItem* s použitím anotací:

- *@Entity*: označuje třídy reprezentující Entitu.
- *@Table*: vlastnosti tabulky jako název, schéma nebo omezení. Pokud nejsou vyplněny jsou použity výchozí hodnoty.
- *@Id*: vlastnost identifikátoru.
- *@Column*: anotace pro mapování sloupců. Vlastnost *name* definuje název sloupce. Dále definuje vlastnosti jako je například *nullable*, která určuje, zda sloupec obsahuje nebo neobsahuje *null* hodnoty.
- *@GeneratedValue*: strategie generování identifikátoru. Vlastnost *strategy* pro automatické generování hodnoty.

```
1  @Entity
2  @Table
3  public class NutrientItem {
4
5      @Id
6      @Column(name = "id")
7      @GeneratedValue(strategy = GenerationType.AUTO)
8      private Long id;
9
10     @Column(name = "name", nullable = false)
11     private String name;
12
13     @Column(name = "usda_nutrient_id")
14     private Integer usdaNutrientId;
15
16     @Column(name = "unit_name", nullable = false)
17     private String unitName;
18
19     @Column(name = "default_item", nullable = false)
20     private boolean defaultItem;
21 }
```

Zdrojový kód 10: Definice entity - hibernate

5 Okta

Veřejně obchodní společnost pro správu identit a přístupu se sídlem v San Francisku. Poskytuje cloudový software, který společnostem pomáhá správu a zabezpečení autentizace uživatelů aplikace. Vývojářům dává možnost integrovat prvky pro kontrolu identity, webových stránek, webových služeb a zařízení. Založena byla v roce 2009 spoluzakladateli Todd McKinnon a Frederic Kerrest. V roce 2015 vzrostl kapitál této společnosti na 1,2 miliardy dolarů. Poslední údaje ukazují, že zaměstnává přes 2000 zaměstnanců.

Software od společnosti Okta má širokou škálu užití. Ověřování uživatelů může probíhat i skrze sociální sítě jako Facebook, Google a další. Pokud společnost využívající Okta software vyvíjí více aplikací, uživatelé se mohou přihlásit jednou a získat přístupy ke kompletní sadě aplikací. Více faktorové ověření poskytuje možnosti využití pokročilejších technologií jako je ověřování pomocí biometriky nebo hlasu.

5.1 Datový model

Entity uvnitř okty se označují jako „zdroje“. Každý zdroj obsahuje:

- Atributy jako je název nebo vytvoření.
- URL odkazy popisující vztahy nebo akce, které lze podniknout se zdrojem.
- Profil, který umožňuje ukládat informace o objektu (email, jméno, ...)

Registrací v Okta dojde k vytvoření soukromého datového prostoru označovaného jako „organizace“ obsahujícího aplikace, adresáře, uživatelé, skupiny, a další. Uživatelé v organizaci jsou uživatelé používající aplikaci, kteří mají nastavenou e-mailovou adresu a uživatelské jméno.[8]

5.2 Komunikace s aplikací

Jak již bylo zmíněno výše, Okta se využívá pro autentizaci uživatelů. Aplikace může nechat přihlašování uživatelů, registraci nových uživatelů a provádění obnovení účtu na Okta hostitelských službách. Aby bylo možné použití těchto služeb, musí proběhnout přesměrování z aplikace na Okta. To se obvykle provádí pomocí protokolu OpenID Connect. Přihlášení pak probíhá v několika krocích:

- Uživatel klikne na tlačítko v aplikaci.
- Aplikace přesměruje prohlížeč na přihlašovací stránku hostovanou na Okta, kde proběhne autentizace uživatele.
- Okta přesměruje prohlížeč zpět na konkrétní stránku pomocí zpětného volání spolu s informací o uživateli.

- Aplikace zpracuje odpověď z Okta a může odeslat další požadavek pro získání více informací o uživateli.
- Proběhne přesměrování na adresu určenou přihlášením uživatele do aplikace.

6 Infrastruktura a nasazení

Pro dostupnost webové aplikace z internetu bylo třeba zvolit odpovídající hosting a s tím související i nejjednodušší způsob nasazovaní aplikace.

6.1 DigitalOcean

Jako poskytovatel cloudové infrastruktury byl zvolen DigitalOcean[9] a to zejména pro jeho příznivou cenu a jednoduché uživatelské rozhraní, ve srovnání s ostatními velkými poskytovateli jako AWS, Google, Azure. Pro účel prezentace bakalářské práce byl vybrán tarif „Basic“ s jednojádrovým procesorem, 2GB RAM a 2TB datovým přenosem za měsíc. Tato konfigurace je pro prezentační účely dostačující, v případě produkčního provozu aplikace by musel být tarif navýšen.

6.2 Droplet

Jako Droplet DigitalOcean označuje virtuální stroj, který je možno využívat samostatně nebo v rámci clusteru a je mu přidělena veřejná IP adresa. Zvolený droplet má předinstalované Ubuntu 20.04 a běží jako samostatný server.

6.3 SSH

Pro připojení na server bylo využito protokolu SSH, který slouží jako zabezpečený kanál pro komunikaci se serverem. Pro zvýšení zabezpečení byla vygenerována dvojice klíčů (soukromý a veřejný). Veřejný klíč byl přenesen na vzdálený server a přihlašování k serveru probíhalo bez nutnosti zadávání hesla.

6.4 Docker

Docker byl využit pro běh aplikací v kontejnerech, které se postaraly o jednotné rozhraní a izolaci jednotlivých aplikací. Jedná se o virtualizační nástroj na úrovni procesů, tedy nevirtualizuje se celý hardware (jako například u VirtualBoxu), ale pouze jednotlivé procesy. Tím je docíleno takřka nulových režijních nákladů pro běh jednotlivých kontejnerů.

Další výhodou využití Dockeru je fakt, že si program nese v kontejneru celé své běhové prostředí. Pokud tedy vše funguje v kontejneru na testovacím serveru, poběží vše správně i na produkčním serveru.[9]

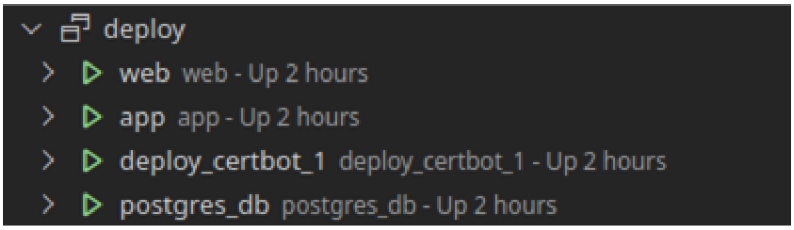
[9] https://www.digitalocean.com/

Rozlišujeme dva hlavní pojmy.

- **Docker Image:** je neměnný soubor, který obsahuje zdrojový kód, knihovny, závislosti, nástroje a další soubory potřebné pro spuštění aplikace. Předpis pro image se nazývá Dockerfile.
- **Docker kontejner:** spuštěný image se nazývá container. Je to virtualizované běhové prostředí, které je odděleno od operačního systému hostitelského serveru.

Jednotlivé image mohou vycházet z jiných imagů a tím dědí veškerou jejich funkcionalitu. Aplikace využívá čtyř bežících kontejnerů.

- **postgres_db:** kontejner s běžící PostgreSQL databází
- **app:** kontejner s běžící backendovou aplikací. Image pro tento kontejner vychází z image pro aplikační server Tomcat, ve kterém je provozována samotná aplikace
- **web:** kontejner s běžícím Nginx webovým serverem, který obsahuje build Angular aplikace
- **deploy_certbot_1:** kontejner pro obnovu ssl certifikátu (platnost certifikátu je omezena na 3 měsíce)



Obrázek 5: Docker kontejnery

6.4.1 Docker compose

Nástroj sloužící pro usnadnění práce s více kontejnery. Jelikož se aplikace skládá ze 4 kontejnerů, musely by být spouštěny postupně pomocí příkazu *docker run* s danými parametry. Při použití *docker-compose* lze napsat jako předpis pro spouštění soubor ve formátu YAML, který definuje jednotlivé kontejnery a jejich parametry. *Docker-compose* také automaticky vytváří lokální síť a zapojuje do nich všechny obsažené kontejnery, ty potom spolu mohou komunikovat zabezpečeně v rámci této lokální sítě. V následujícím příkladu je ukázána část použitého *docker-compose.yml* s definicí atributů pro spouštění kontejneru s databází.

```
version: '3'
services:
   postgres_db:
      image: postgres:latest
      ports:
         - "5432:5432"
      environment:
         - POSTGRES_USER=ad_user
         - POSTGRES_PASSWORD=test
         - POSTGRES_DB=athletes-diary-db
      volumes:
         - ./data/:/var/lib/postgresql
      container_name: postgres_db
```

Zdrojový kód 11: Část docker-compose.yml

- **image:** obraz, ze kterého kontejner bude vycházet
- **ports:** z důvodu bezpečnosti nejsou žádné porty, mimo portů uvedených v této sekci, publikované z kontejneru ven. Dále lze porty přemapovat na jiné číslo pro běh venku a uvnitř kontejneru.
- **environment:** definuje proměnné pro prostředí uvnitř kontejneru. V tomto případě tedy jméno a heslo pro databázového uživatele a jméno databáze
- **volumes:** slouží pro mapování složek z kontejneru do hostitelského počítače mimo kontejner. V tomto případě slouží pro persistentní uložení dat z databáze. Pokud by totiž data nebyla uložena na hostitelském počítači, došlo by při rebuildu image ke smazání všech dat
- **container_name:** jméno kontejneru

7 Programátorská dokumentace

V této kapitole je popsána aplikace z pohledu vývojáře, která se dělí na 3 podkapitoly. Webová část popisuje strukturu Angular a napojení na autentizační modul Okta. Server část obsahuje ukázku vytvoření Spring boot aplikace na jejím počátku, adresářovou strukturu s přehledem komponent a entitní model pro definici dat. V poslední kapitole je vysvětlen postup pro nasazení aplikace na provozní prostředí.

7.1 Webová část

Kapitola popisující adresářovou strukturu webové aplikace Angular, seznam vytvořených komponent, autentizaci uživatele a základní používané třídy a soubory.

7.1.1 Adresářová struktura

Vytvořením Angular projektu a jeho následným buildem vznikla adresářová struktura pro vývoj aplikace.

- **dist/** - výsledný build Angular aplikace
- **node_modules/** - instalované knihovny aplikace
- **src/** - zdrojový adresář aplikace se souborem *index.html* a adresářem app/
- **app/** - obsahuje veškeré komponenty a soubory aplikace rozdělené do dalších podadresářů

7.1.2 Komponenty a jejich rozdělení

Jelikož komponenty definují pouze jednotlivé bloky aplikace, musejí být vkládány do plnohodnotné HTML stránky v prohlížeči. Z tohoto důvodu obsahuje webová část aplikace jeden soubor *index.html* umístěný ve zdrojovém adresáři src/. Uvnitř adresáře se pak nachází podsložka app/, která obsahuje jednotlivé komponenty tvořící web. Celá aplikace vlastní pouze jeden modul *AppModule*, umístěný také ve složce app/ s definicí všech komponent, služeb a importů. Každá komponenta má svůj vlastní adresář pojmenovaný podle názvu komponenty a obsahující hlavní soubory:

- *.component.css* poskytující sadu stylů
- *.component.html* pro definici šablonu
- *.component.ts* pro definici komponenty

a přídavných nepovinných souborů:

- *service.ts* poskytujících služby komponentě

- *.ts* obsahující datovou strukturu

Po načtení souboru *index.html* je zavolána kořenová komponenta *AppComponent* tvořící vstupní bránu do aplikace. Komponenta kontroluje pomocí Okta, zda je uživatel přihlášen a podle odpovědi z Okta přesměrovává uživatele na přihlašovací stránku nebo na požadovanou adresu URL v aplikaci.

Podle tréninkových jednotek, obecného přehledu a nastavení dále rozlišujeme hlavní komponenty,

- **dashboard:**
  Zobrazuje data na webu z silové a běžecké části pomocí grafů. Je do ní vložena přídavná komponenta *noteOverview* pro zobrazení všech poznámek.
- **power-training:**
  Reprezentuje silovou sekci tréninkového deníku. Slouží pro zobrazení dat pomocí grafů a tabulek. Vytvoření tréninku obsahuje přídavnou komponentu *exercise-dialog* pro definici cviků.
- **run-training:**
  Běžecká sekce tréninkového deníku. Zobrazuje data pomocí grafů a tabulek. Vytvoření tréninku obsahuje přídavnou komponentu *run-section-dialog* pro vytvoření běžeckých sekcí.
- **nutrition:**
  Nutriční sekce zobrazující seznam jídel a jejich nutričních hodnot. Přídavná komponenta *nutrition-dialog* slouží k vyhledání stravy a vytvoření záznamů s nutričními hodnotami.
- **settings:**
  Komponenta uživatelského nastavení. Obsahuje přídavné komponenty *power-training-setting*, *run-training-setting* a *nutrition-setting* pro nastavení jednotlivých sekcí.

a přídavné, které jsou umístěny ve složce component/.

- ***exercise-dialog*:** dialog pro vytvoření cviku
- ***navigation*:** komponenta navigace
- ***noteDialog*:** dialog pro vytvoření poznámky
- ***noteOverview*:** přehled poznámek v časové ose
- ***nutrition-dialog*:** dialog pro vytvoření záznamu v nutriční sekci
- ***nutrition-setting*:** nastavení nutriční sekce
- ***power-training-setting*:** nastavení silové sekce
- ***run-section-dialog*:** dialog pro vytvoření běžecké sekce
- ***run-training-setting*:** nastavení běžecké sekce

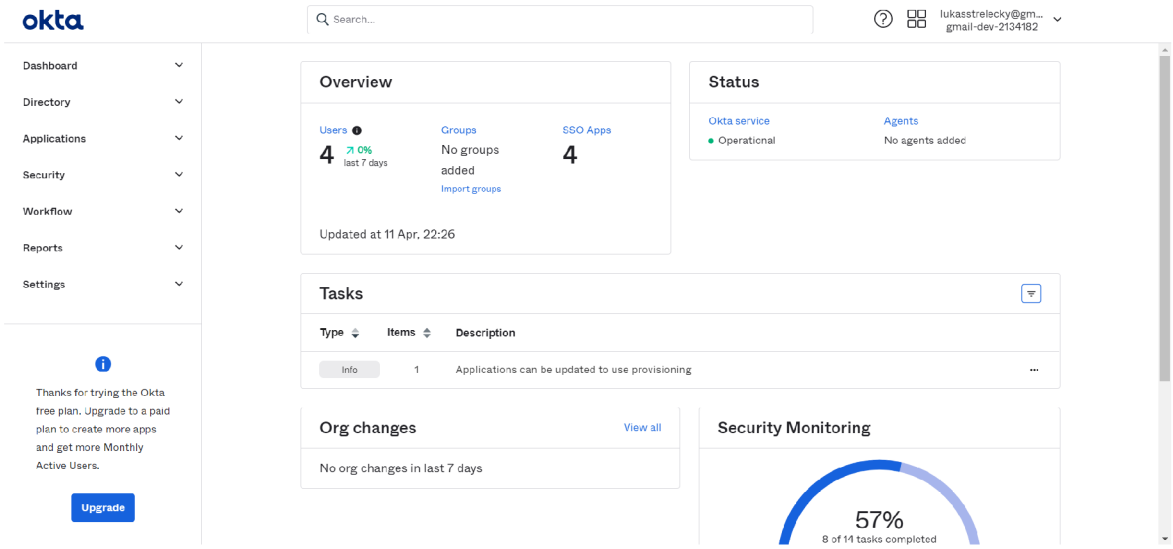
7.1.3 Výčtové typy (Enums)

Jsou umístěny v adresáři enums/ a rozděleny do TypeScript souborů podle samostatných částí tréninkového deníku. Jeden soubor může obsahovat více výčtových typů. Například soubor *foodNutrition.ts* obsahuje výčtové typy id hodnot v USDA databázi, název nutričních hodnot a název nutričních jednotek.

7.1.4 Autentizace a služby Okta

Pro autentizaci uživatelů byla implementována služba *AuthenticationService*, která je provázána se službami Okta a vkládána do hlavních komponent. Ke samotné správě uživatelů a nastavení aplikace poskytuje Okta administrátorskou konzoli s velkým množstvím funkcí. Administrátor může například spravovat role uživatelů, definovat víceúrovňové ověřování nebo kontrolovat aktivitu uživatelů. Navigační menu je rozděleno na:

- **Dashboard:** úvodní obrazovka
- **Directory:** slouží pro správu uživatelů
- **Applications:** nastavení aplikace
- **Security:** zabezpečení aplikace a uživatelů
- **Workflow:** nastavení akcí
- **Reports:** statistiky a log
- **Settings:** obecné nastavení



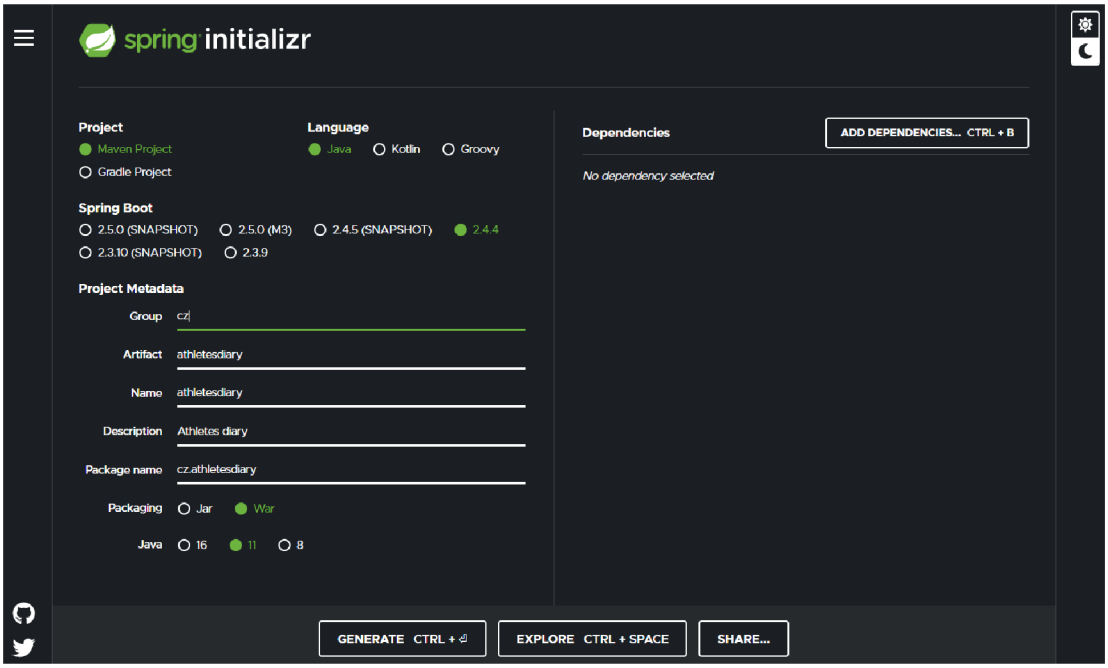
Obrázek 6: Okta administrátorská konzole

7.2 Server část

Obsah této kapitoly tvoří seznámení se způsobem, jakým byla aplikace Spring boot vytvořena, popis adresářové struktury a entitní model představující uložená data v databázi a jejich vazby.

7.2.1 Vytvoření aplikace Spring boot

Pro běh aplikace slouží servlet kontejner Tomcat 9.0, ale je možno nasazení na jakýkoliv servlet kontejner kompatibilní s verzi 3.1 a vyšší. Spring Boot verze 2.4.4 vyžaduje alespoň verzi Java 8 a Spring Framework 5.3.5. K sestavení aplikace slouží Apache Maven. Pro vytvoření nového projektu bylo využito stránky https://start.spring.io/. Po vyplnění údajů stránka automaticky vygenerovala celou strukturu aplikace Spring boot včetně souboru pom.xml s vloženými závislostmi pro sestavení projektu pomocí vybrané technologie Maven.



Obrázek 7: Spring Boot - vytvoření projektu

7.2.2 Adresářová struktura aplikace Spring boot

Vytvořená aplikace se nachází v adresáři athletesdiary/. Hlavními částmi jsou:

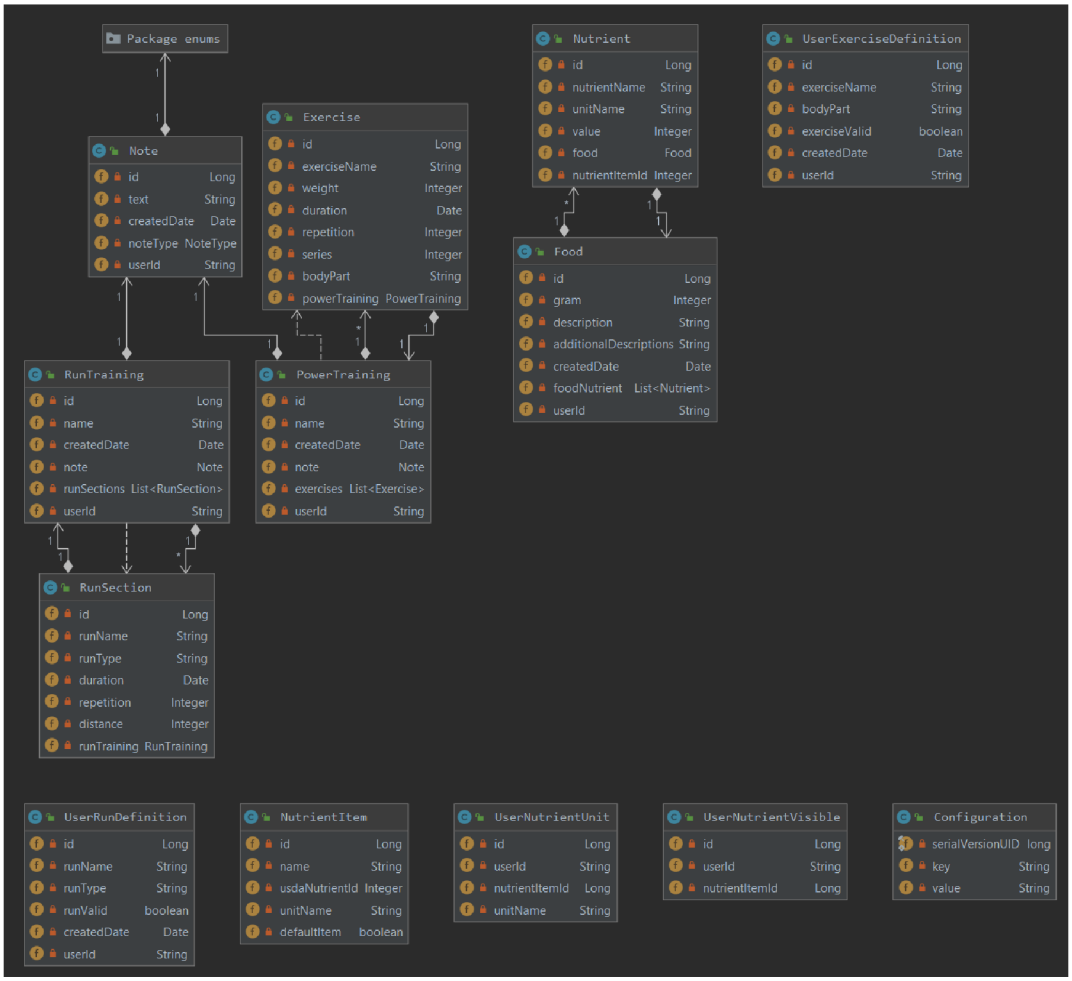
- **src/** - zdrojový adresář aplikace Spring boot. Obsahuje všechny Java třídy a konfigurační soubor *application.properties*.
- **target/** - výsledný adresář po sestavení aplikace skrze Maven.

- **pom.xml** - konfigurační soubor k sestavení aplikace. Obsahuje informace o aplikaci, verzi Java a závislosti.

Třídy umístěné v adresáři java/, který se nachází uvnitř src/, jsou rozděleny podle logických celků do čtyř podadresářů. Každý z těchto podadresářů obsahuje seznam tříd pro jednotlivé sekce tréninkového deníku (dashboard, poznámky, nutriční sekce, silová sekce, běžecká sekce a uživatelské nastavení).

- **controller/**
  Kontrolery pro práci s REST API. Metody uvnitř těchto tříd zpracovávají příchozí požadavky z URL.
  - ***DashboardController***
  - ***NoteController***
  - ***NutritionController***
  - ***PowerTrainingController***
  - ***RunTrainingController***
  - ***UserSettingController***
- **service/**
  Služby pro vytvoření business vrstvy. Metody pouze volají metody tříd umístěných v adresáři repository/.
  - ***DashboardService***
  - ***NoteService***
  - ***NutritionService***
  - ***PowerTrainingService***
  - ***RunTrainingService***
  - ***UserSettingService***
- **repository/**
  Repositáře pro manipulaci s daty uložené v databázi. Třídy obsahují metody pro vytváření dotazů do databáze a transformaci dat do DTO.
  - ***DashboardRepository***
  - ***NoteRepository***
  - ***NutritionRepository***
  - ***PowerTrainingRepository***
  - ***RunTrainingRepository***
  - ***UserSettingRepository***

- **model/** - Definice entitního modelu a tříd známých jako Data Transfer Object (DTO)[10]. Pro lepší přehlednost je entitní model zobrazen na diagramu v následujícím obrázku.



Obrázek 8: Entitní model

[10]Datová struktura (třída) pro přenos dat. Neobsahuje žádnou logiku a slouží pouze k zapouzdření dat.

7.3 Postup nasazení na provozní prostředí

Jak již bylo zmíněno v kapitole 6 o infrastruktuře a nasazení, aplikace využívá čtyř běžících docker kontejnerů. Kontejnery poskytují běhové prostředí pro web, backend a databázi. Na produkčním prostředí vznikl adresář deploy/ pro nasazení aplikace a spuštění těchto kontejnerů. Aby posun aplikace na produkční prostředí proběhl v pořádku, je stanoven základní postup pro nasazení.

1. Sestavení aktuální verze aplikace
    - **Angular:** pomocí správce balíčků „npm“ je výsledkem sestavená aplikace ve složce dist/.
    - **Spring:** sestavení probíhá pomocí příkazu maven. Výsledkem je soubor „backend-0.0.1-SNAPSHOT.war“ umístěný v adresáři target/.
2. Pokud sestavení proběhlo v pořádku, je potřeba přenést adresář dist/ a soubor „backend-0.0.1-SNAPSHOT.war“ do adresáře deploy/ na provozním prostředí.
3. Po dokončení přenosu dat je za potřebí všechny kontejnery vypnout, případně provést změny nad databázi. Vypnutí kontejnerů probíhá pomocí příkazu „docker stop web app postgres_db deploy_certbot_1“.
4. Posledním krokem je sestavení kontejnerů a jejich spuštění pomocí příkazu „docker-compose up --build“.

S nasazením nové verze je povinností vývojáře zkontrolovat aplikované změny na produkčním prostředí. V průběhu nasazování může dojít k neočekávaným chybám, které se projeví až při práci s aplikací.

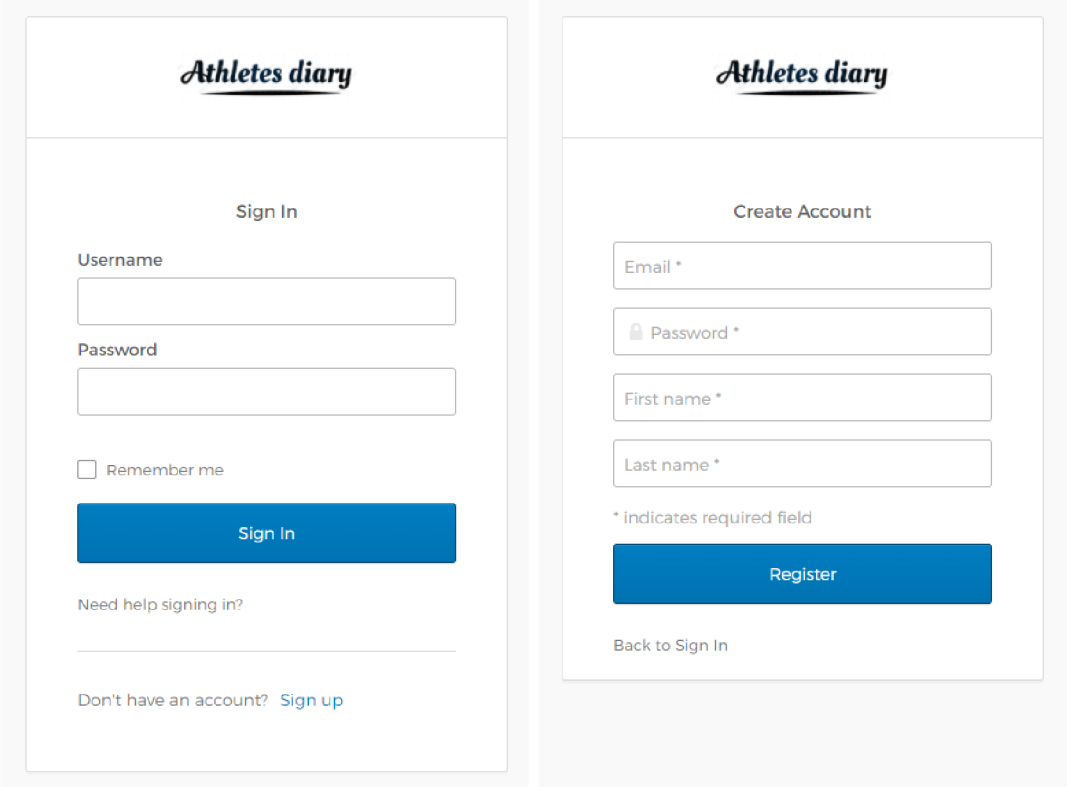
8 Uživatelská dokumentace

Obsahem této kapitoly je seznámení s aplikací z uživatelského úhlu pohledu.

8.1 Regitrace a přihlášení

Registrace uživatele je prvním krokem pro zahájení práce s aplikací. V úvodu je nejdříve načtena stránka pro přihlášení, ve které je umístěn odkaz „Sign up“ pro zobrazení registračního formuláře. Vyplněním povinných položek (e-mail, heslo, jméno a příjmení) označených hvězdičkou a stiskem tlačítka „Register“, je odeslán aktivační e-mail na vyplněnou e-mailovou adresu. Potvrzením je uživateli povolen přístup do aplikace.

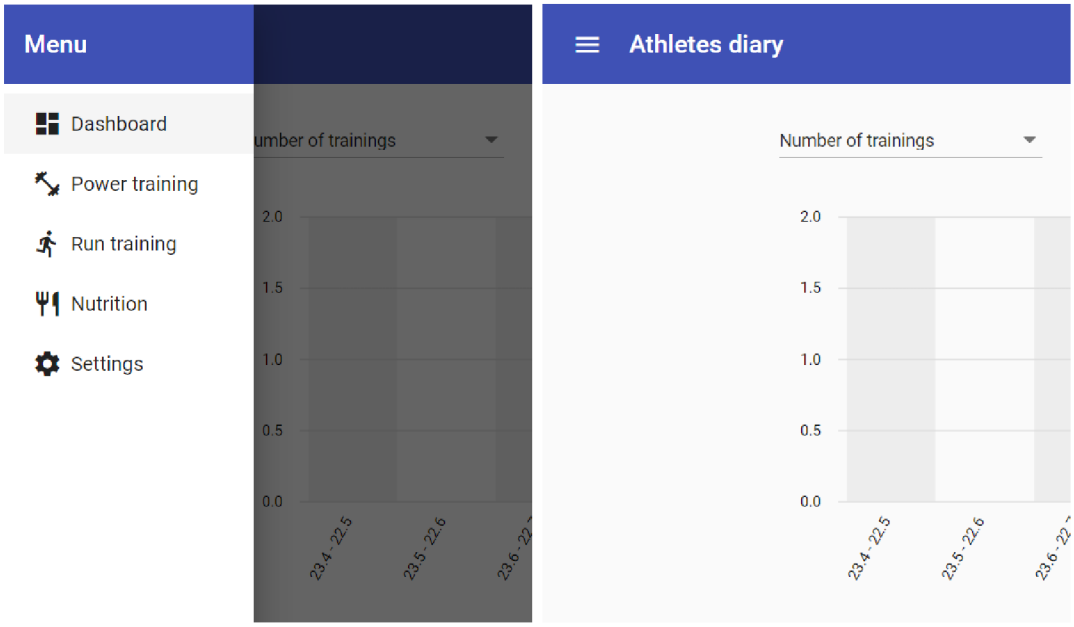
Pro přihlášení a vstup do aplikace je nutné vyplnit uživatelské jméno „Username“ a heslo „Password“ v přihlašovací obrazovce. Potvrzením tlačítka „Sign in“ proběhne ověření uživatelského účtu a následné přesměrování na úvodní stránku v aplikaci.



Obrázek 9: Přihlášení a registrace

8.2 Navigační menu

Slouží pro přepíná mezi hlavními moduly aplikace (Dashboard, Power training, Run training, Nutrition a Setting). Menu je vyvoláno stiskem ikony umístěné v levé části záhlaví stránky. Pro uživatele je navigační menu ve výchozím stavu skryto. Aplikace tak může využít celou šířku webové stránky pro práci s jejím obsahem.



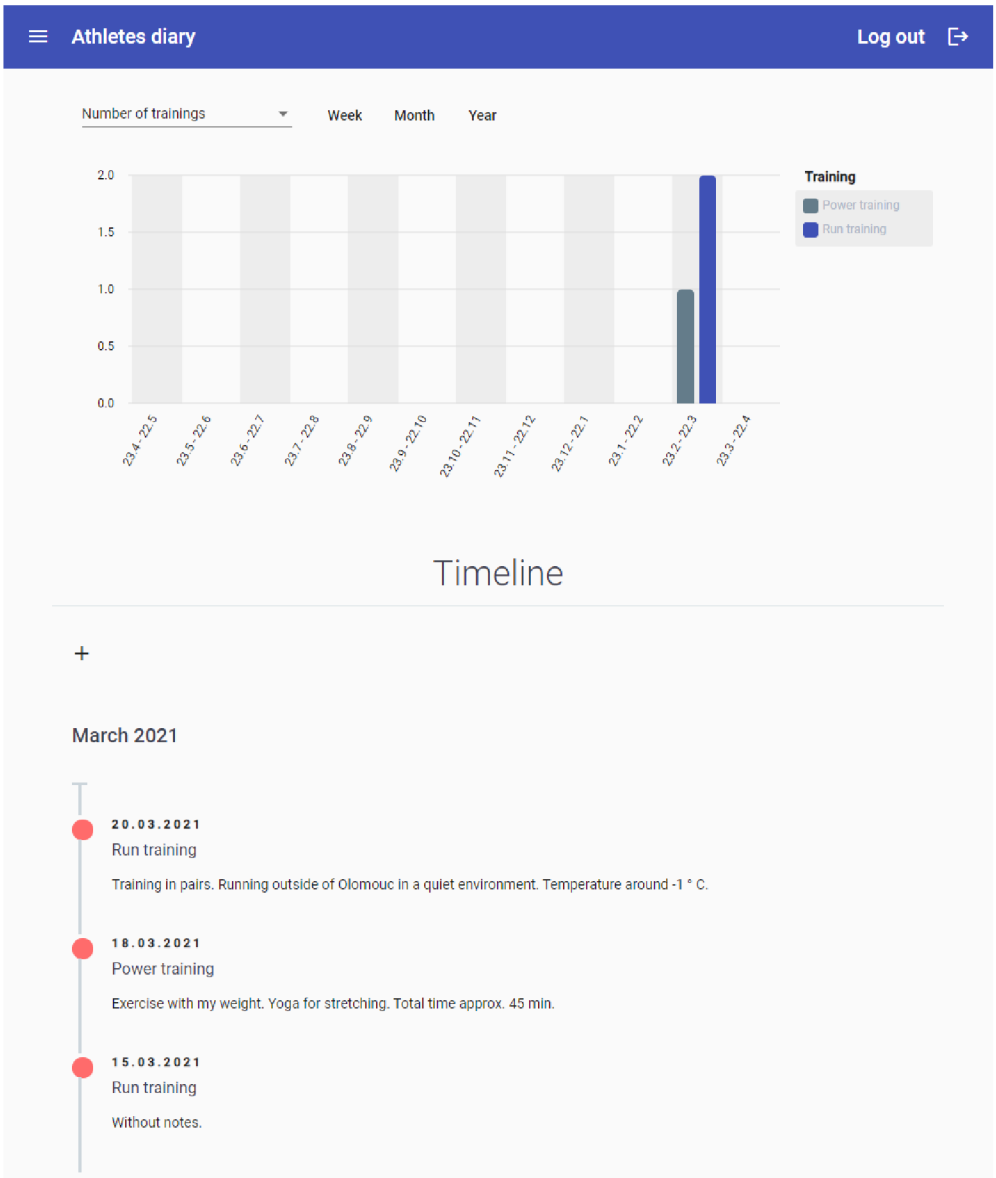
Obrázek 10: Navigační menu

8.3 Úvodní stránka (Dashboard)

V horní části dashboard jsou umístěny grafy zobrazující statistiky pro silový a běžecký trénink. Statistiky jsou vyhodnocovány podle vybraného časového období, a to vždy za poslední týden, měsíc nebo rok.

- **Number of trainings:** výchozí statistika počtů běžeckých a silových tréninků v časovém horizontu. Graf zobrazuje poměr mezi jednotlivými tréninky.
- **Performance:** výkonnostní statistika za zvolené časové období pro silový a běžecký trénink. K zobrazení slouží dva kruhové grafy. Graf pro silový trénink poskytuje procentuální informace pro různé typy zaměření (silový, objemový, ...). Graf pro běžecký trénink zobrazuje procentuální informace o typech běhů.

Následující část stránky tvoří „Timeline“ umístěná přímo pod grafy. Na vertikální časové ose jsou zobrazeny poznámky, které uživatel zadal do aplikace v rámci jednotlivých tréninků nebo přímo v Dashboard pomocí ikony „+“.

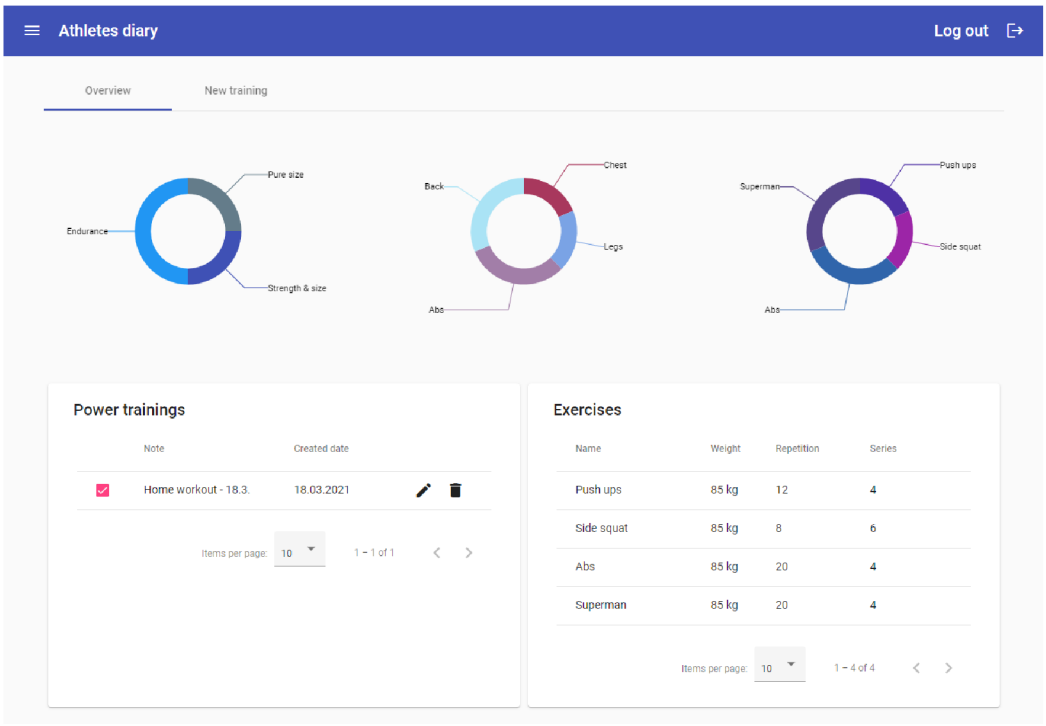


Obrázek 11: Dashboard

8.4 Silový trénink (Power training)

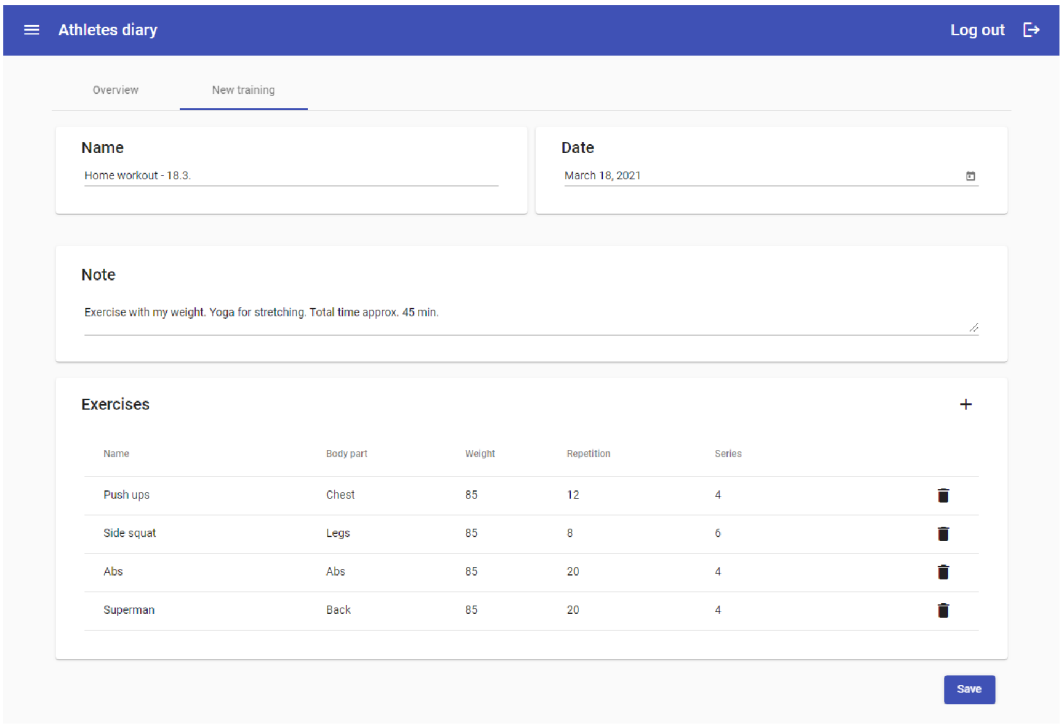
Část tréninkového deníku zaměřená na posilování. Uživateli dává možnost nadefinovat nové tréninkové jednotky se seznamem cviků a zobrazovat tyto uložené informace v přehledu. Přesměřováním na stránku silového tréninku je uživateli zobrazen přehled obsahující dvě části.

- **Overview:** První záložka obsahující dvě tabulky a tři grafy. Tabulka „Power trainings“ zobrazuje uložené tréninkové jednotky seřazené sestupně podle data uložení. Výběr konkrétní tréninkové jednotky pomocí zaškrtávacího pole (checkbox) aktualizuje data v přehledu. Aktualizací dojde k zobrazení seznamu cviků v tabulce „Exercises“ a načtení statistických údajů do grafů pro vybraný trénink. Grafy, tak jak jsou umístěny, zobrazují procentuální poměr o typech posilování, procentuální poměr posilovacích partii na těle a poměr uzvednuté váhy pro jednotlivé cviky.



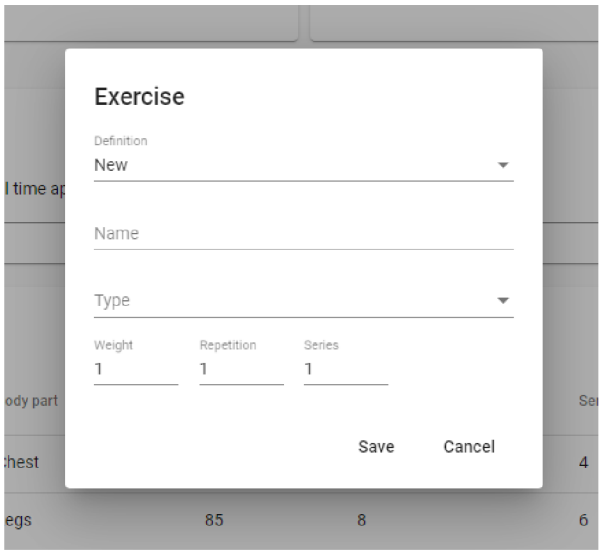
Obrázek 12: Silový trénink - přehled

- **New training:** Druhá záložka sloužící k vytvoření nové tréninkové jednotky. Uživatel zde vyplní údaje jako jsou název, datum, poznámka a seznam cviků. Tlačítkem „Save“ proběhne uložení nové tréninkové jednotky a její přidání do přehledové tabulky v první záložce. Přidávání cviků do tabulky „Exercises“ probíhá formou dialogu, který je zobrazen po stisku tlačítka „+“. K odstranění uložených cviků slouží řádková ikona koše.



Obrázek 13: Silový trénink - vytvoření

Dialog obsahuje formulář pro vyplnění názvu cviku, tělové partie, váhy, počet opakování a počet sérii. Uložení cviku do tabulky se provádí stiskem tlačítka „Save“, v jakémkoliv jiném případě proběhne uzavření dialogu.

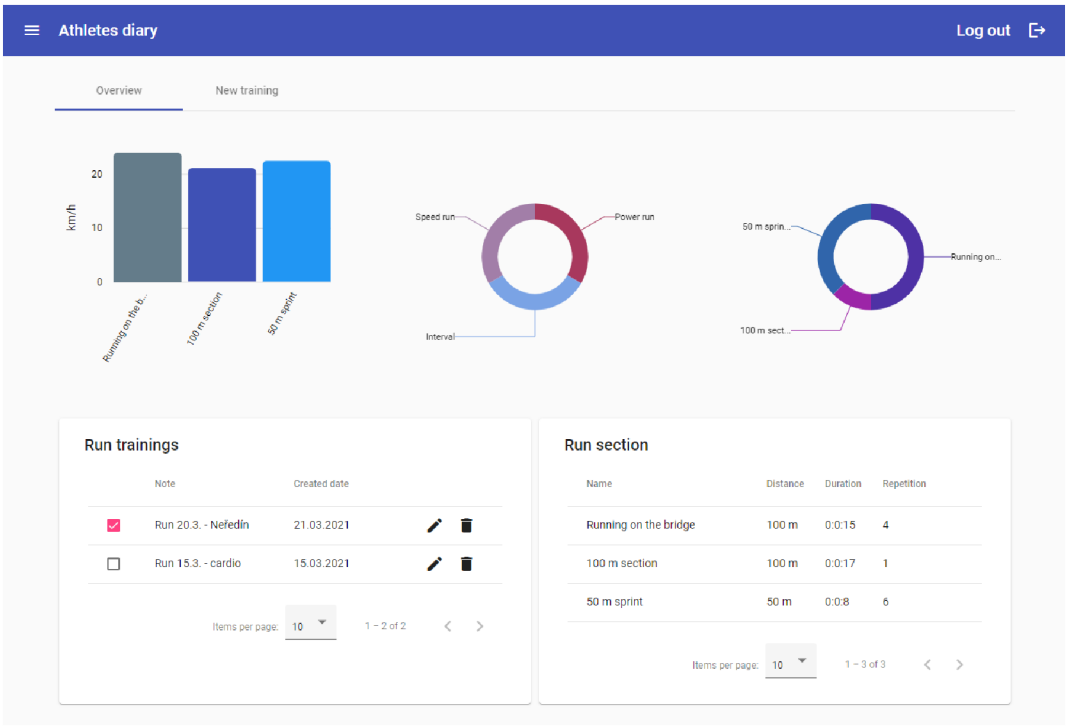


Obrázek 14: Silový trénink - vytvoření cviku

8.5 Běžecký trénink (Run training)

Část tréninkové deníku zaměřená na běh a jeho fáze. Stejně jako silový trénink je běžeckých trénink považován za tréninkovou jednotku. Běžecká tréninková jednotka je ale rozdělena na jednotlivé sekce (běhy). Uživatel tak může vytvořit tréninkovou jednotku obsahující jeden nebo více běhů. Přesměrováním na stránku běžeckého tréninku je zobrazen přehled obsahující dvě části.

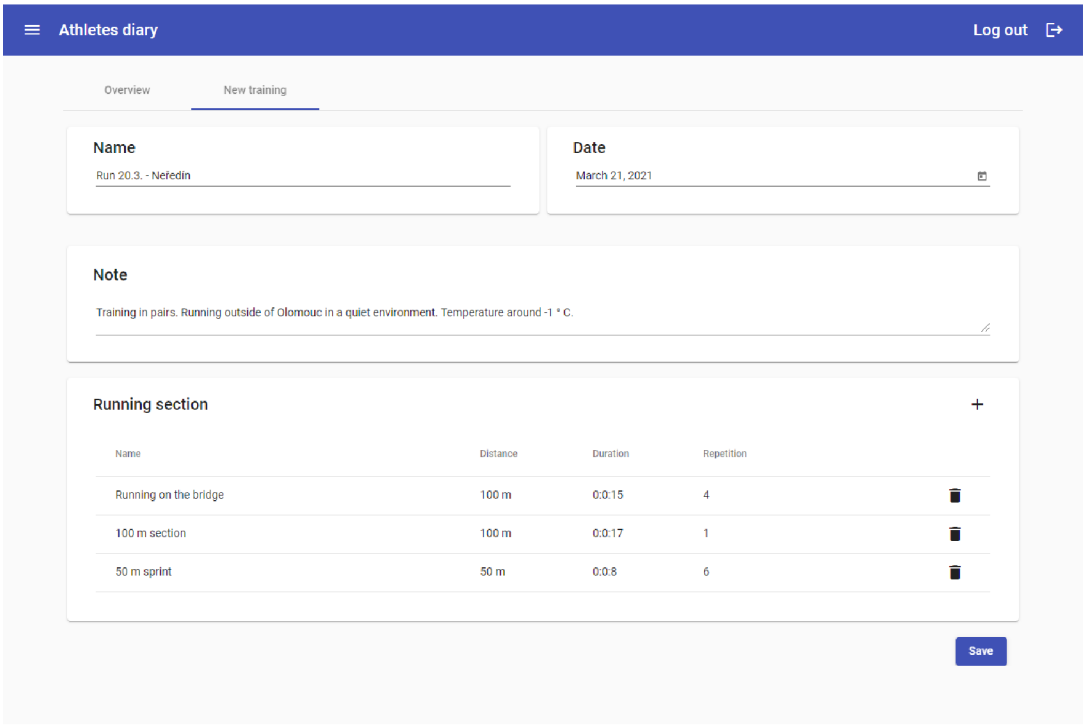
- **Overview:** První záložka, která obsahuje tabulku „Run trainings“ s přehledem uložených tréninkových jednotek seřazených sestupně podle data uložení. K ní náleží druhá tabulka „Run section“ pro zobrazení uložených běžeckých sekcí a tři grafy prezentující rychlost jednotlivých běhů v km/h, procentuální poměr typů běhů a uběhnutou vzdálenost v metrech pro každou běžeckou sekci. Výběrem tréninkové jednotky přes checkbox proběhne aktualizace dat v tabulce „Run section“ a v grafech. Uživatel má také možnost celou tréninkovou jednotku smazat pomocí řádkové ikony koše nebo editovat skrze ikonu tužky. Editace probíhá obdobně jako vytvoření nové tréninkové jednotky. V případě editace je ale uživatel přesměrován na druhou záložku s již předvyplněnými hodnotami tréninkové jednotky.



Obrázek 15: Běžecký trénink - přehled

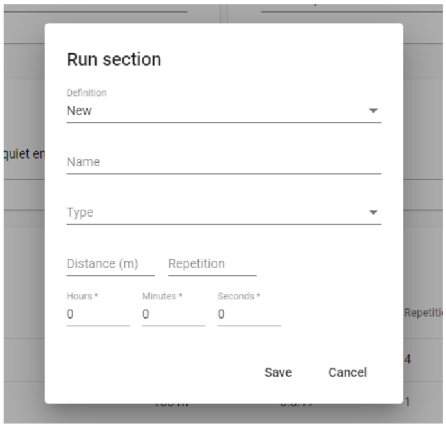
- **New training:** Záložka pro vytvoření nové tréninkové jednotky. Obsahuje pole pro vyplnění názvu tréninkové jednotky, datumu, poznámky a tabulku „Running section“ se seznamem běžeckých sekcí. Tlačítkem „Save“ dojde

k uložení nové tréninkové jednotky, která je následně zobrazena v první záložce. K přidání a vytvoření běžecké sekce slouží ikona „+“ otevírající dialog s definicí běžecké sekce.



Obrázek 16: Běžecký trénink - vytvoření

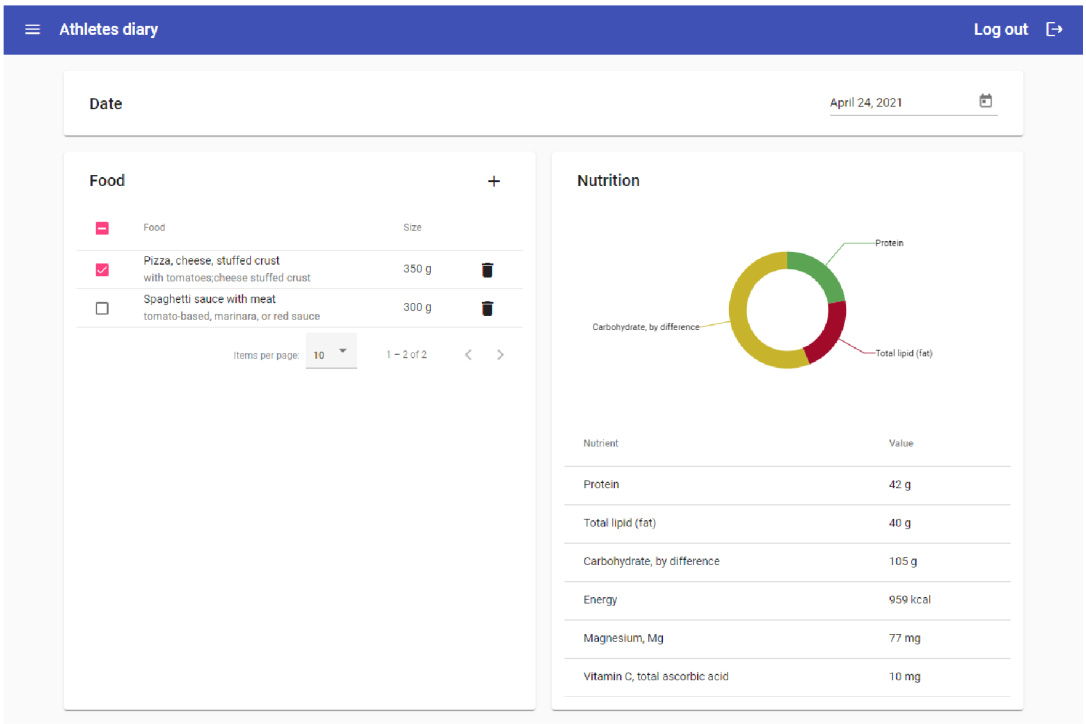
Dialog obsahuje formulář pro vyplnění údajů o běžecké sekci (název, typ běhu, vzdálenost, počet opakování, čas). Stiskem tlačítka „Save“ proběhne uložení běžecké sekce do tabulky. V případě špatně zadaných údajů může uživatel běžeckou sekci smazat přes ikonu koše.



Obrázek 17: Běžecký trénink - vytvoření běžecké sekce

8.6 Výživa (Nutrition)

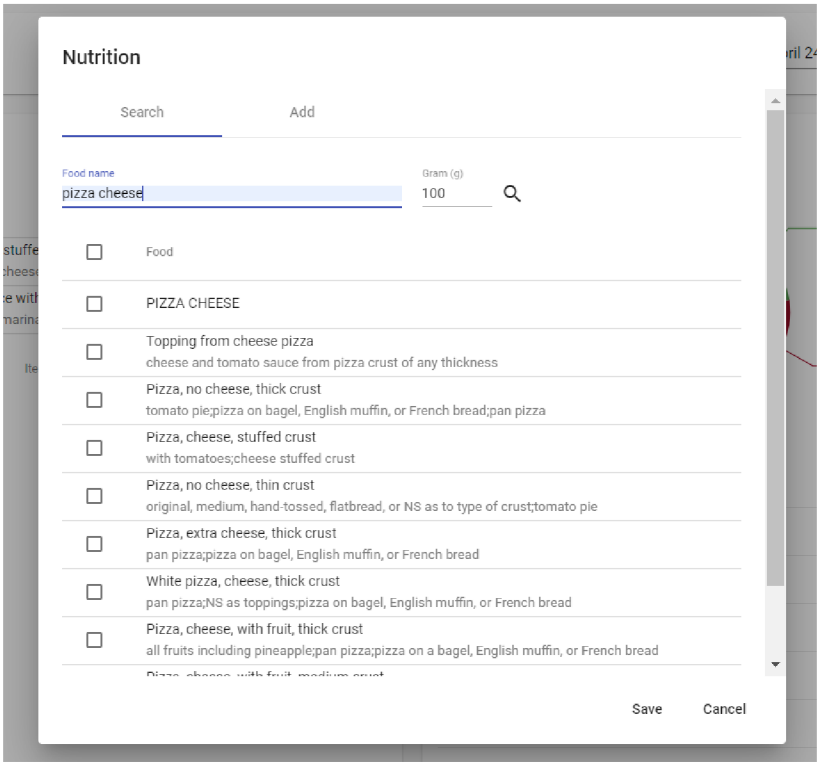
Nedílnou součástí tréninkové přípravy sportovce tvoří i jeho stravování. Sekce „Nutrition“ tvoří třetí hlavní část aplikace zaměřenou na ukládání a zobrazování nutričních informací. Tabulka „Food“ obsahuje seznam uložených potravin (jídel) a jejich hmotnost. Zaškrtnutím checkboxu u potraviny dojde k aktualizaci druhé tabulky „Nutrition“ s nutričními hodnotami pro vybrané potraviny. Pokud uživatel vybere více potravin, provede se součet jejich nutričních hodnot. Díky tomu lze vypočítat nutriční hodnoty zkonzumované během celého dne nebo pouze u vybraných jídel. Graf umístěný nad tabulkou zobrazuje poměr nutričních hodnot pro bílkoviny, sacharidy a tuky. Potraviny a jejich nutriční hodnoty jsou zobrazeny pro vybraný den v měsíci pomocí komponenty kalendáře. K při-



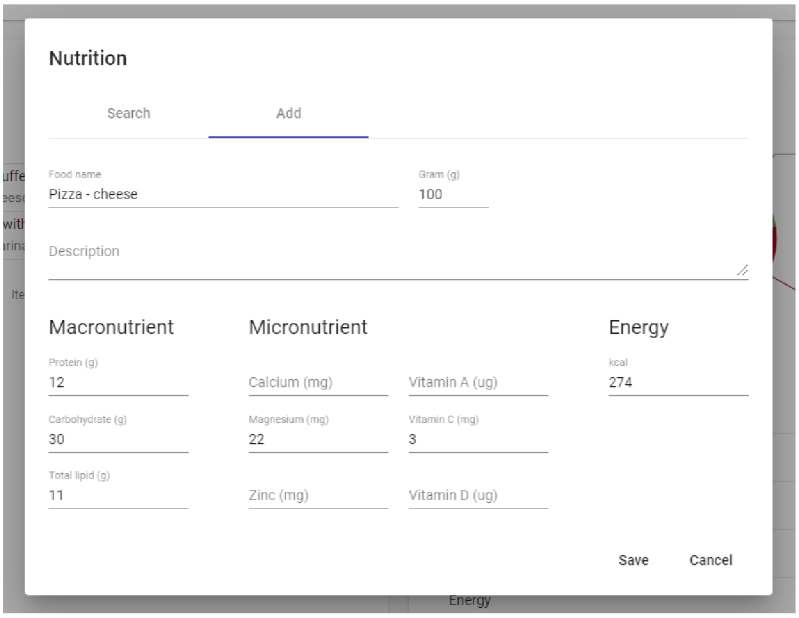
Obrázek 18: Výživa - přehled

dání potravin a jejich nutričních hodnot do aplikace slouží tlačítko „+“ jenž zobrazí dialog „Nutrition“ se dvěma záložkami. Záložka „Search“ dává možnost uživateli vyhledat jakoukoliv potravinu v databázi nutričních hodnot USDA. Databáze vrací prvních 10 nalezených výsledků pro vyhledávanou hodnotu, které jsou zobrazeny v tabulce pod vyhledáváním. Pomocí checkboxu může uživatel vybrat jakoukoliv potravinu a tlačítkem „Save“ ji uložit do aplikace. Druhá záložka „Add“ slouží k vlastnímu nadefinování hodnot potraviny. Veškeré údaje o potravině si zde uživatel může nadefinovat podle sebe. Nutriční hodnoty jsou rozděleny na tři části „Macronutrient“, „Micronutrient“ a „Energy“. Tyto hodnoty platí pro hmotnost potravin na 100 gramů, jelikož se nejčastěji uvádí na obalů v obchodech, restauracích a na internetu. Uložení se provádí opět tlačítkem

„Save“. Potraviny jsou následně zobrazeny v tabulce „Food“ odkud je uživatel může odstranit pomocí řádkové ikony koše.



Obrázek 19: Výživa - vyhledání potraviny

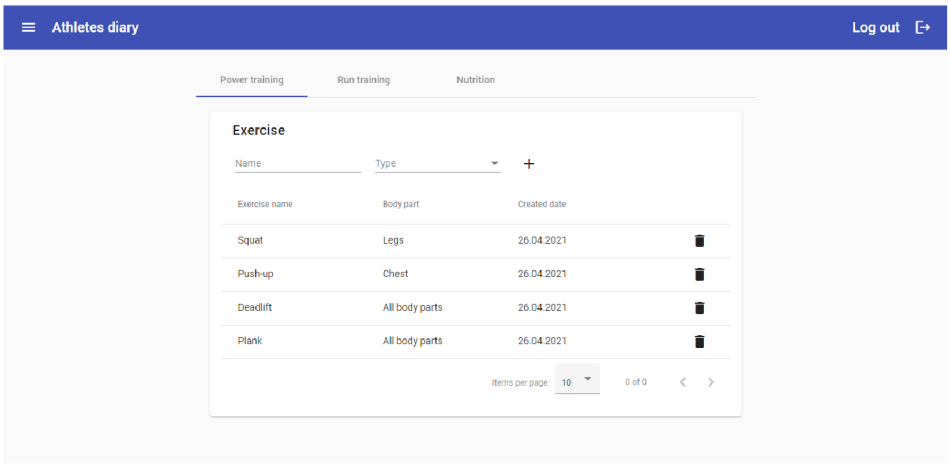


Obrázek 20: Výživa - definice potraviny

8.7 Uživatelské nastavení (Settings)

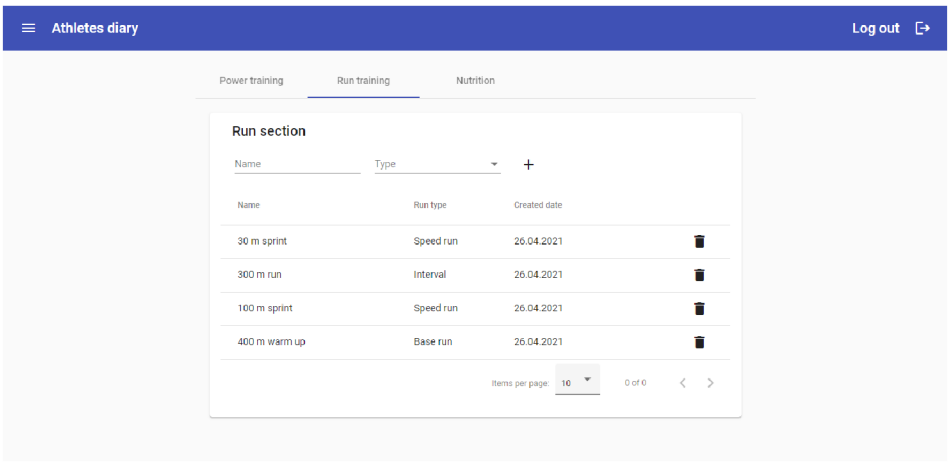
Aplikace obsahuje nastavení, které uživateli usnadní práci se záznamy týkající se silové, běžecké a nutriční sekce. Nastavení je rozděleno do tří záložek.

- **Power training:** Uložení uživatelsky předdefinovaných cviků pro silovou část tréninkového deníku. V případě vytvoření nové tréninkové jednotky se v dialogu přidání nového cviku uživateli zobrazí seznam uložených cviků podle tohoto nastavení. Uložené cviky lze smazat pomocí ikony koše.



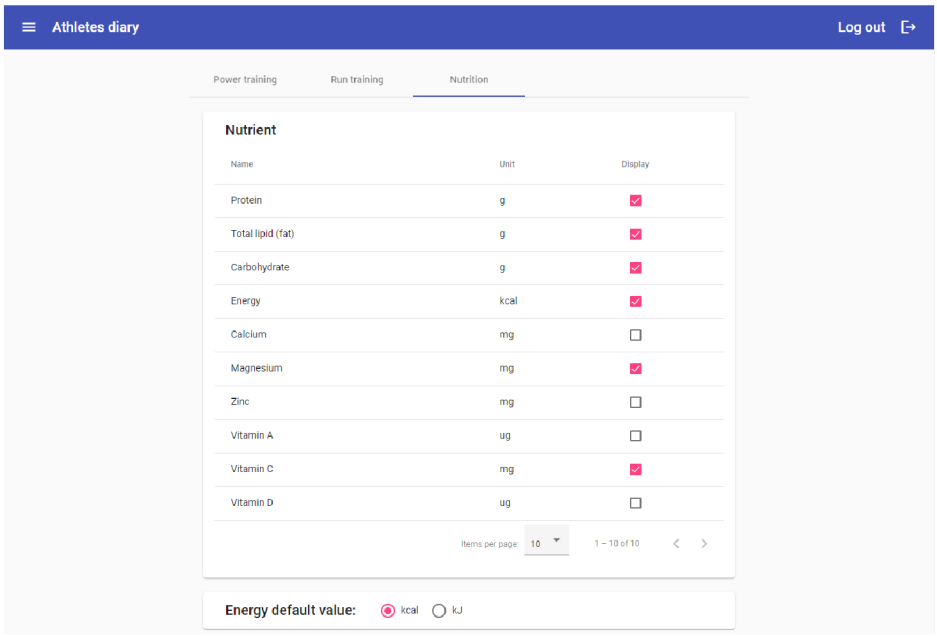
Obrázek 21: Nastavení - silový trénink

- **Run training:** Uložení předdefinovaných běhů pro běžecký trénink. Seznam běhů prezentovaný v tabulce „Run section“ bude uživateli nabízen i při vytváření nové tréninkové jednotky v dialogu přidání běžecké sekce. Uložené záznamy lze smazat pomocí ikony koše.



Obrázek 22: Nastavení - běžecký trénink

- **Nutrition:** Nastavení pro nutriční sekci tréninkového deníku. V tabulce „Nutrient“ je zobrazen seznam všech nutrientů potravin ukládaných do aplikace. Uživatel si výběrem nutrientu přes checkbox nastaví nutriční hodnoty, které chce v aplikaci zobrazovat. Přepínač „Energy default value“ slouží pro nastavení výchozí energetické hodnoty potravin.



Obrázek 23: Nastavení - nutriční hodnoty

9 Plány do budoucna

Původním záměrem bylo vytvoření aplikace umožňující sportovci ukládat data a poskytovat jejich zpětnou analýzu pro zlepšení tréninkového výkonu. Během vývoje se ukázalo, že tato vize je velice omezená z pohledu vývojáře a poskytuje širokou škálu možností pro další rozvoj. Aplikace by mohla obsahovat například:

- **Trenérskou část:** jednalo by se o další komplexní systém, který by propojil sportovce s jejich trenéry a umožnil tak sportovním klubům organizovat tréninky.
- **Mobilní aplikaci:** ačkoli existuje mnoho verzí mobilních aplikací, jen málo jich poskytuje dostatečnou funkcionalitu, kterou by mohl využít profesionální sportovec v rámci tréninku. Zápis informací během tréninku do mobilní aplikace je mnohem detailnější a záleží jen na sportovci, jak tuto funkcionalitu využije.
- **GPS a mapy:** pro sekci běžeckého tréninku je využití GPS a zobrazení trasy na mapách zajímavým rozšířením. Mobilní aplikace ji v dnešní době běžně poskytují. Funkcionalita rozlišující nadefinované typy běhů v nastavení z dat získaných přes GPS do aplikace a jejich zobrazení na mapě, je dalším krokem pro vylepšení takovýchto sportovních aplikací.
- **Repositář potravin a nutričních hodnot:** Aktuálně je aplikace navázána na USDA databázi nutričních hodnot, čímž se aplikace stává závislou na druhé straně. Vlastní databáze by uživatelům poskytovala větší komfort pro získávání nutričních hodnot, které budou více přiblíženy určitým zemím.

Závěr

Cílem práce bylo vyvinout aplikaci umožňující sportovcům ukládat data z tréninků a provádět nad těmito daty základní analýzu ke zlepšení výkonnosti. Vznikla tak aplikace, kterou budou moci využívat lidé vykonávající sport na profesionální i rekreační úrovni. Aplikace rozděluje trénink na část silovou, běžeckou a nutriční. Uložená data jsou zobrazena v podobě grafů a tabulek. Může se jednat například o přehled zatížení tělových partií při silovém tréninku, uběhnuté kilometry a množství nutričních hodnot zkonzumovaných potravin. Úvodní stránka uživateli poskytuje průběžný přehled o svých trénincích. Dále je možné využít nastavení, které uživateli usnadní práci při zadávání nových údajů do aplikace.

Během implementace této práce jsem se naučil mnoho nových technologii, které mi rozšířily obzor nejen v programování, ale i v dalších oblastech vývoje a distribuce softwaru. Jelikož jsem v minulosti neměl žádnou zkušenost s vývojem rozsáhlejších projektů, považuji tuhle zkušenost za velice přínosnou.

V budoucnu je jedním z prvních cílů rozšíření aplikace o trenérskou sekci, ve které trenér bude mít možnost evidovat data všech svých sportovců a vytvoření verze pro Android zpřístupněné na Google Play. Aplikace se tak stane více využitelná právě v profesionálních sportech, kde bude také ulehčovat práci mnohem více, jelikož jsou zde zaměstnaní trenéři a sportovci na plný úvazek.

Conclusions

The aim of this work was to create a web application that allows athletes to store and analyze training data to improve their performance. Therefore, Athlete's diary was developed and can be used by any athlete on a professional or recreational level. The application is able to store strength and stamina training as well as nutrition. Stored data are visualized and presented in form of graphs and tables. The athlete is, for example, able to get an overview of stressed body parts during strength training, distance covered during stamina training, or amount of nutrition consumed each day.

During the implementation of this work, I learned plenty of new technologies not only in the programming field but also in other areas of web development and software deployment. Since I didn't have any previous experience with the development of large projects, I consider this experience to be very beneficial.

One of my first goals in the future is to extend the application with a coaching section, in which the coach will be able to track the data of all his athletes. The next step is to create a mobile application and publish it in Google Play. This should make the application more usable on a professional level where it will make work much easier as there are full-time coaches and athletes.

A Obsah přiloženého DVD

Součástí práce je DVD s následující adresářovou strukturou:

`bin/`
: Kompletní adresářová struktura webové aplikace (v ZIP archivu) pro zkopírování na webový server.

`doc/`
: Text práce ve formátu PDF, vytvořený s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce, včetně všech příloh, a všechny soubory potřebné pro bezproblémové vygenerování PDF dokumentu textu (v ZIP archivu), tj. zdrojový text textu, vložené obrázky, apod.

`src/`
: Kompletní zdrojové texty webové aplikace se všemi knihovnami a dalšími soubory potřebnými pro bezproblémové vytvoření adresářové struktury ke zkopírování na webový server.

`readme.txt`
: Webová adresa, na které je aplikace nasazena pro účel testování při tvorbě posudků práce a pro účel obhajoby práce.

Literatura

[1] *Introduction to modules.* [online]. 2021 [cit. 2021-5-3]. Dostupný z: ⟨https://angular.io/guide/architecture-modules⟩.

[2] *Introduction to components and templates.* [online]. 2021 [cit. 2021-5-3]. Dostupný z: ⟨https://angular.io/guide/architecture-components⟩.

[3] *Binding syntax.* [online]. 2021 [cit. 2021-5-3]. Dostupný z: ⟨https://angular.io/guide/binding-syntax⟩.

[4] *Introduction to Angular concepts.* [online]. 2021 [cit. 2021-5-3]. Dostupný z: ⟨https://angular.io/guide/architecture⟩.

[5] *Introduction to the Spring Framework.* [online]. 2020 [cit. 2021-5-3]. Dostupný z: ⟨https://docs.spring.io/spring-framework/docs/4.3.x/spring-framework-reference/html/overview.html⟩.

[6] FOWLER, Martin. *IntegrationTest* [online]. 2018 [cit. 2021-5-3]. Dostupný z: ⟨https://martinfowler.com/bliki/IntegrationTest.html⟩.

[7] *Using Spring Boot.* [online]. 2021 [cit. 2021-5-3]. Dostupný z: ⟨https://docs.spring.io/spring-boot/docs/current/reference/html/using-spring-boot.html⟩.

[8] *Okta Data Model.* [online]. 2021 [cit. 2021-5-3]. Dostupný z: ⟨https://developer.okta.com/docs/concepts/okta-data-model/⟩.

[9] *Docker overview.* [online]. 2021 [cit. 2021-5-3]. Dostupný z: ⟨https://docs.docker.com/get-started/overview/⟩.

[10] PŘISPĚVATELÉ. *Log* [online]. 2020 [cit. 2021-5-3]. Dostupný z: ⟨https://cs.wikipedia.org/wiki/Log⟩.

[11] PŘISPĚVATELÉ. *Spring Framework* [online]. 2021 [cit. 2021-5-3]. Dostupný z: ⟨https://en.wikipedia.org/wiki/Spring_Framework⟩.

[12] PŘISPĚVATELÉ. *Java Database Connectivity* [online]. 2021 [cit. 2021-5-3]. Dostupný z: ⟨https://en.wikipedia.org/wiki/Java_Database_Connectivity⟩.

[13] PŘISPĚVATELÉ. *Java Message Service* [online]. 2019 [cit. 2021-5-3]. Dostupný z: ⟨https://cs.wikipedia.org/wiki/Java_Message_Service⟩.

[14] PŘISPĚVATELÉ. *Databázová transakce* [online]. 2020 [cit. 2021-5-3]. Dostupný z: ⟨https://cs.wikipedia.org/wiki/Datab%C3%A1zov%C3%A1_transakce⟩.

[15] *Web on Servlet Stack.* [online]. 2021 [cit. 2021-5-3]. Dostupný z: ⟨https://docs.spring.io/spring-framework/docs/current/reference/html/web.html⟩.

[16] PŘISPĚVATELÉ. *Transmission Control Protocol* [online]. 2021 [cit. 2021-5-3]. Dostupný z: ⟨https://cs.wikipedia.org/wiki/Transmission_Control_Protocol⟩.

[17] PŘISPĚVATELÉ. *Java Portlet Specification* [online]. [cit. 2021-5-3]. Dostupný z: ⟨https://en.wikipedia.org/wiki/Java_Portlet_Specification⟩.

[18] *Developing Your First Spring Boot Application.* [online]. 2021 [cit. 2021-5-3]. Dostupný z: ⟨https://docs.spring.io/spring-boot/docs/current/reference/html/getting-started.html#getting-started-first-application⟩.

[19] SIMPSON, Jordan. *Spring @Component Annotation* [online]. [cit. 2021-5-3]. Dostupný z: ⟨https://www.baeldung.com/spring-component-annotation⟩.

[20] PŘISPĚVATELÉ (comp.). *Mapping Entities* [online]. 2004 [cit. 2021-5-3]. Dostupný z: ⟨https://docs.jboss.org/hibernate/annotations/3.5/reference/en/html/entity.html⟩.

[21] CRUSOVEANU, Loredana. *Intro to Inversion of Control and Dependency Injection with Spring* [online]. 2021 [cit. 2021-5-3]. Dostupný z: ⟨https://www.baeldung.com/inversion-control-and-dependency-injection-in-spring⟩.

[22] *The IoC container.* [online]. 2016 [cit. 2021-5-3]. Dostupný z: ⟨https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/beans.html⟩.

[23] *Okta-hosted flows.* [online]. 2021 [cit. 2021-5-3]. Dostupný z: ⟨https://developer.okta.com/docs/concepts/okta-hosted-flows/⟩.

[24] *Orientation and setup.* [online]. 2021 [cit. 2021-5-3]. Dostupný z: ⟨https://docs.docker.com/get-started/⟩.