VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

WEBOVÝ NÁSTROJ PRO TVORBU, SPRÁVU A VYUŽITÍ DATABÁZE SPORTOVNÍCH POZIC

WEB TOOL FOR CREATION, MANAGEMENT, AND USE OF A DATABASE OF SPORTS POSES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **TOMÁŠ KŘIVÁNEK**
AUTHOR

VEDOUCÍ PRÁCE **prof. Ing. ADAM HEROUT, Ph.D.**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Křivánek Tomáš**
Program: Informační technologie
Název: **Webový nástroj pro tvorbu, správu a využití databáze sportovních pozic**
Web Tool for Creation, Management, and Use of a Database of Sports Poses
Kategorie: Uživatelská rozhraní

Zadání:

1. Seznamte se s problematikou vývoje webových aplikací.
2. Seznamte se s problematikou evidence sportovních pozic, zaměřte se na jógu, zvažte i další sporty.
3. Navrhněte prvky uživatelského rozhraní pro správu databáze sportovních pozic.
4. Iterativně testujte prvky vyvíjené aplikace s uživateli a iterativně je vylepšujte.
5. Navrhněte a implementujte vytvářenou aplikaci a na základě testování ji iterativně vylepšujte.
6. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu; vytvořte plakátek a krátké video pro prezentování projektu.

Literatura:

- Tidwell et al.: Designing Interfaces: Patterns for Effective Interaction Design, O'Reilly, 2020
- Steve Krug: Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability, ISBN: 978-0321965516
- Steve Krug: Rocket Surgery Made Easy: The Do-It-Yourself Guide to Finding and Fixing Usability, ISBN: 978-0321657299
- Joel Marsh: UX for Beginners: A Crash Course in 100 Short Lessons, O'Reilly 2016

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3, značné rozpracování bodů 4 a 5.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Herout Adam, prof. Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 1. listopadu 2021

Abstrakt

Tato práce je zaměřena na dvě metody vyhledávání v databázi sportovních pozic, které se dají ovšem použít naprosto v jakémkoliv vyhledávání ve webové aplikaci. Nejdůležitější vlastnosti pro běžné uživatele je samozřejmě rychlost a přesnost vyhledávání. Jednou z nejlepších metod pro rychlé vyhledávání je tzv. livesearch, který uživateli nabízí výsledky vyhledávání přímo při zadávání textu. To často zapříčiní, že se výsledek objeví již při zadávání textu tzn. uživatelé nemusí dokončit vyhledávaný výraz. Přesnosti vyhledávání je často velmi těžké dosáhnout. Pro toto jsou využity úseky textu, klíčová slova nebo jiné technologie. Metoda, kterou využívá valná většina sociálních sítí jsou tzv. tagy, což je obdoba klíčových slov. Tyto tagy mají, ale jednu velkou nevýhodu, co se týče vyhledávání na sociálních sítích. Můžeme pomoci nich vyhledat příspěvky, které daný tag mají připsaný a to je vše. Mnohem přirozenější by bylo, kdybychom tyto tagy mohli mezi sebou kombinovat, vytvářet logické výrazy za pomocí těchto tagů, specifikovat, které tagy má náš vyhledávaný výsledek mít a které naopak nikoliv. Pro řešení tohoto problému byl navržen jednoduchý logický jazyk, které dostal pojmenování „Queries“, protože fungují velmi podobně jako queries, které se využívají pro vyhledávání v databázi. Výsledkem této práce je webová aplikace, která využívá dříve zmíněné metody (livesearch, queries). Webová aplikace pracuje s databází sportovních jógových pozic, ve kterých uživatel může rychle a jednoduše vyhledávat, ale také specifikovat logické výrazy, které vyjadřují vlastnosti daných pozic.

Abstract

This paper is focused on two searching methods and how they can be used in any searching and in any web application. Two most important qualities of any search are speed and accuracy. One of the best methods in terms of speed is so-called live search. This method provides results immediately when the user starts typing their search expression. That is why in many cases, the user does not even have to finish their expression, because when seeing the result, the user can immediately access it. Search accuracy is often achieved by using keywords or some part of text. Another common way to organize and fetch documents are tags: a short string which identifies a post or a photo. Multiple tags are then used similarly to plain keywords. This work proposes to use tags as a new searching logical language, where presence and/or absence of tags can be required. For this purpose I created a simple logical language “Queries”, because they can be used very similarly to database queries. In my case, the language and the user interface is made very simple so that a normal user can specify the query himself. This search tool is showcased in a web application focused around yoga poses. Both methods are used there – the user can search in the yoga poses database by name or by the queries language.

Klíčová slova

python, web, aplikace, livesearch, tag, query, mongodb, flask, backend, frontend, javascript, jQuery, css, html, asynchronní

Keywords

python, web, application, livesearch, tag, query, mongodb, flask, backend, frontend, javascript, jQuery, css, html, asynchronous

Citace

KŘIVÁNEK, Tomáš. *Webový nástroj pro tvorbu, správu a využití databáze sportovních pozic.* Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Ing. Adam Herout, Ph.D.

Webový nástroj pro tvorbu, správu a využití databáze sportovních pozic

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana prof. Ing. Adama Herouta Ph. D. .Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Tomáš Křivánek
2. května 2022

Poděkování

Chtěl bych poděkovat mému vedoucímu prof. Ing. Adamu Heroutovi Ph. D. za velkou pomoc při psaní bakalářské práce a za všechny připomínky a nápady k webové aplikaci. Dále bych chtěl poděkovat své přítelkyni za pomoc s testováním a za navržení loga aplikace. V neposlední řadě bych chtěl poděkovat svým rodičům, kteří mi během celého studia a během psaní bakalářské práce poskytovali velkou podporu.

Obsah

Kapitola 1

Úvod

Cílem práce je vytvořit přehlednou a uživatelsky přívětivou webovou aplikaci, která spolupracuje s databází sportovních pozic. Výsledná webová aplikace je soustředěna pouze na jógové pozice, ve kterých může uživatel vyhledávat pomocí dvou zajímavých vyhledávacích metod. První metodou je metoda livesearch. Ta uživatelům poskytuje výsledky jejich hledání přímo při zadávání vyhledávaného výrazu. Po přidání nebo odmazání znaku se tedy uživateli zobrazí nové výsledky hledání. Pokud uživatel nemůže svoji pozici najít, je mu nabídnuta možnost přidat novou jógovou pozici do databáze. Druhou metodou je vyhledávání pomocí jednoduchého logického jazyka queries, pomocí kterého uživatel může psát logické výrazy za pomoci klíčových slov a tagů. Tagy jsou jednoslovné řetězce, které blíže určují vlastnosti nějaké jógové pozice. Nejen, že uživatel může vytvářet logické výrazy a vyhledávat pomocí nich, může si také toto vyhledávání ukládat k sobě na profil. Tzn. uživatel nemusí stejnou query psát pokaždé znovu, ale může využít rychlou nabídku svých uložených queries přímo na hlavní stránce. Jelikož je databáze zaměřena na jógové pozice, byla do webové aplikace přidána i možnost vytvořit si vlastní jógovou sestavu z pozic, které již v databázi jsou. Jógové sestavy totiž hrají klíčovou roli v samotném cvičení jógy.

Práce je členěna do 6 kapitol. V první kapitole je blíže popsáno vlastní zadání výsledné aplikace: co by měla aplikace umět a co by měla uživatelům nabízet. V druhé kapitole jsou blíže popsány technologie, které byly využity pro implementaci webových stránek – mezi ně patří programovací jazyky Python, HTML, CSS, JavaScript, a také NoSQL databáze MongoDB. V třetí kapitole je popsán grafický návrh uživatelského rozhraní. Jsou zde popsány dodržované principy a příklad na domovské stránce celé aplikace. Ve čtvrté kapitole je popsána implementace celého systému, ve kterém se zaměřuji spíše na nejzajímavější části implementace. Nejsou zde popsány všechny součásti systému, ale pouze ty, které jsou klíčové pro správný chod aplikace. V šesté kapitole je popsáno testování, které probíhalo metodou do-it-yourself podle Steva Kruga. V závěru jsou shrnuty výsledky implementace a testování. Také je zde zmíněno možné budoucí rozšíření aplikace. Výsledná aplikace je dostupná na adrese: `https://yoga-poses-bp.herokuapp.com/`.

Kapitola 2

Popis webové aplikace

Úkolem je vytvořit jednoduchou a přehlednou webovou aplikaci, která bude zobrazovat databázi jógových sportovních pozic. Uživatelé by měli mít možnost vyhledávat pomocí anglických názvů pozic nebo provádět složitější vyhledávání pomocí klíčových slov (tagů). Tyto tagy budou moci každé pozici volně přidávat a následně je pomocí připsaných tagů vyhledat. Měla by zde být také možnost ukládání tohoto vyhledávání, aby uživatelé nemuseli pokaždé zadaný logický výraz znovu tvořit. Uživatelé tak budou mít k dispozici rychlou možnost jak své uložené vyhledávání znovu zadat do vyhledávacího pole.

Dalším důležitým prvkem webové aplikace bude systém pro tvorbu sestav/sekvencí. Sekvencování [46] je způsob, jakým jsou jógové pozice řazeny v určitém pořadí, aby se vytvořil logický průběh jógové praxe nebo aby se cvičení zaměřilo na určitý výsledek. Různé školy jógy mohou mít různé představy o tom, jak se má jóga nebo osobní praxe řadit.

Bude tedy potřeba systém, kde uživatelé budou moci vybírat z pozic v databázi, seřadit je a připsat jim dobu cvičení. Po uložení sestavy budou moci uživatelé za pomoci webové aplikace pohodlně odcvičit svoji sestavu.

Na interenetu se nachází velká spousta různých datasetů sportovních pozic. Nejlepší zpracování procházení jógových pozic je na webové stránce Yoga journal[1]. Zde jsou jógové pozice seskupeny podle typu: např. pozice v sedě, ve stoje nebo v leže. Design webové stránky může být trošku matoucí, jelikož se na ní nachází prakticky tři navigační lišty. Při prohlížení určité pozice je vlastní fotografie kvůli těmto navigačním lištám až ve spodní části obrazovky. Je zde ale výhoda, že každá pozice má opravdu detailní textový popis a hezké fotografie. Databáze, která byla vytvořena pro účely této práce, používá seznam pozic nabízených právě na stránce Yoga Journal. Jelikož je zde i přehledná tabulka všech pozic, bylo jednoduché vytvořit web scrapový skript, který všechny informace načetl a uložil je do CSV souboru. Nevýhodou aplikace Yoga journal je to, že zde není žádná jednoduchá cesta, jak hledat pozice – při hledání určité pozice se často stává, že místo samotné pozice aplikace vyhledá nějaký článek. Další nevýhodou je, že zde není možnost vytvářet jógové sestavy, které hrají klíčovou roli ve cvičení jógy.

Pro skládání jógových sestav existuje mnoho aplikací [22], ale všechny mají jednu nevýhodu a tou je měsíční paušál. Nabízejí za něj ale propracované mobilní nebo desktopové aplikace, většinou mají originální nákresy nebo fotografie pozic. Například aplikace Sequence Wiz používá velmi jednoduché ilustrace pozic a celá aplikace vypadá uživatelsky přívětivě. Nabízí také možnost, jak si ke každé pozici v sestavě napsat své poznámky.

[1] `https://www.yogajournal.com/`

Kapitola 3

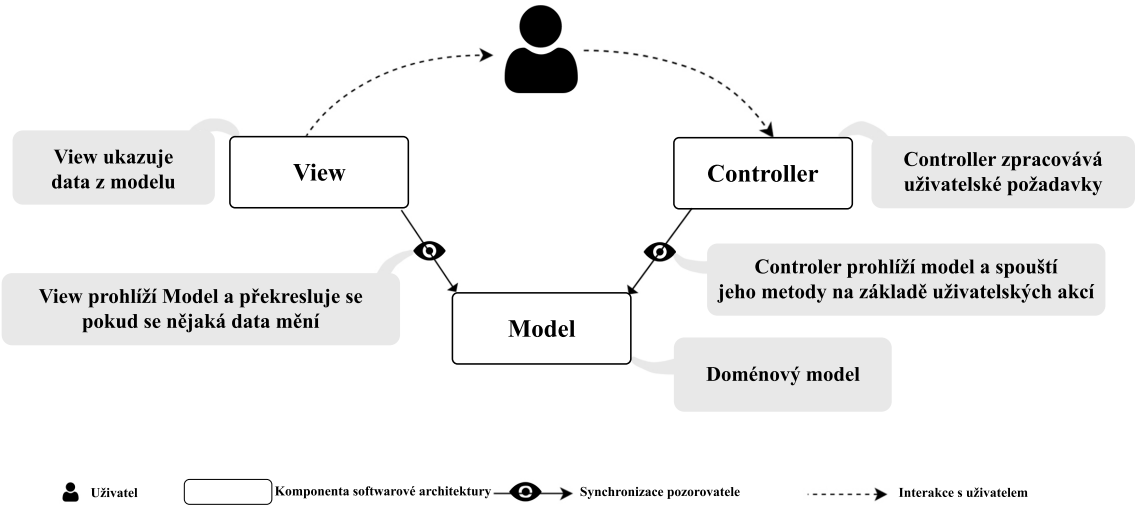
Použité webové technologie

Před započetím implementace bylo potřeba zvolit návrhový vzor aplikace. Pro efektivní výběr konkrétního vzoru musí návrhář prostudovat několik vzorů a jejich verzí, aby pochopil jejich účel, oblasti použití, silné a slabé stránky a jejich použitelnost pro daný problém [42]. Je známo, že výběr návrhových vzorů je důležitý pro srozumitelnost a udržovatelnost programového kódu. Použití nesprávného vzoru (nebo nepoužití žádného vzoru) pro integraci uživatelského rozhraní s aplikační doménou může vést ke složitému kódu, kde i malá změna vede ke značné režii na udržení konzistence mezi uživatelským rozhraním a aplikační doménou.

Jako návrhový vzor pro aplikaci byl zvolen Model-View-Controller. Jedná se o běžný návrhový vzor pro integraci uživatelského rozhraní s logikou aplikační domény [24]. MVC odděluje reprezentaci aplikační domény (Model) od zobrazení stavu aplikace (View) a řízení interakce uživatele (Controller). MVC byl poprvé představen ve Smalltalku'80 vědci Glennem E. Krasnerem a Stephenem T. Popem. Od konce 80. let, kdy bylo MVC zdokumentováno, se objevilo mnoho nových návrhových vzorů MV* [42], jejichž cílem bylo odstranit nevýhody svých předchůdců. Mezi tyto vzory patří např. Model-View-Presenter a Model-View-View-Model (nebo Prezentační model). Pro tuto jednoduchou aplikaci je ale základní Model-View-Controller ideální.

Vzor Model-View-Controller (MVC) [23] zahrnuje tedy tři části: Model, View a Controller. Model reprezentuje stav aplikace a samozřejmě nastavuje a udržuje veškerou komunikaci s databázemi a dalšími zdroji dat. Zobrazení (View) je poměrně jednoduché. Definuje, co uživatel vidí a co z aplikace dostává. To může zahrnovat uživatelské rozhraní a různé formy exportovaných dat (například soubory CSV nebo HTML). Controller přijímá události z view a předává je Modelu. Model události zpracovává a View se synchronizuje se všemi změnami, které v Modelu nastanou (viz obrázek 3.1).

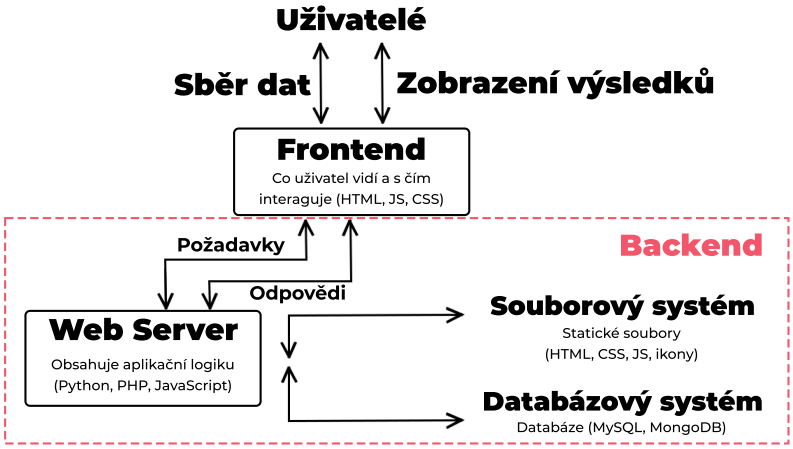
Ve srovnání s jednoduchou samostatnou aplikací telefonního seznamu, kterou v roce 1989 vytvořil Tim Berners-Lee, se webové aplikace a technologie klient-server dostaly poměrně daleko [16]. V dnešní době mají webové aplikace různé podoby a velikosti, například statické, dynamické, systémy pro správu obsahu, elektronické obchodování a hry až po portály pro sdílení živého obsahu. Společně sdílenou technologií výše uvedených typů aplikací je backendová a frontendová technologie.



Obrázek 3.1: Schéma návrhového vzoru Model-View-Controller, které ukazuje, jak mezi sebou jednotlivé části systému MVC komunikují a kde je v tomto modelu postaven uživatel. Přeloženo z [42].

3.1 Backend

Vývoj backendu se zabývá logickou stránkou webové aplikace [16]. Týká se především programovacích jazyků, architektury jádra a logiky. Tyto logiky jsou napsány hlavně v programovacích jazycích, které mohou běžet na počítačových serverech. Backend také ovlivňuje způsob ukládání dat, přístup k nim a jejich obsluhu ze serverů.

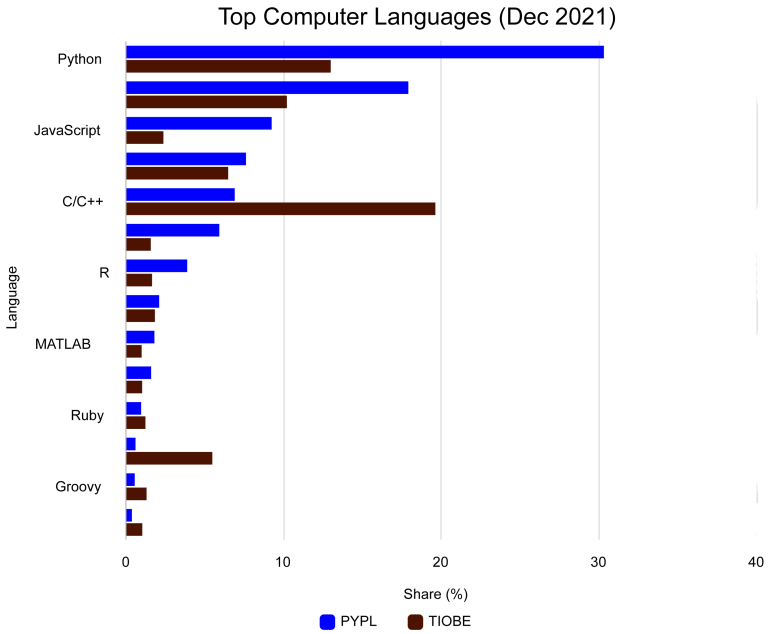


Obrázek 3.2: Uživatelé pracují s frontendem aplikace, pomocí kterého zasílají požadavky na webový server. Na něm je spuštěn backend aplikace, který dotazy zpracovává a zasílá odpovědi zpět na frontend a ten je zobrazí uživateli. Převzato z [27]

3.1.1 Python

Python je interpretovaný a objektově orientovaný programovací jazyk [37]. Obsahuje moduly, výjimky, dynamické typování, dynamické datové typy velmi vysoké úrovně a třídy. Kromě objektově orientovaného programování podporuje i více programovacích paradigmat, například procedurální nebo funkcionální programování. Python kombinuje pozoruhodnou sílu s velmi přehlednou syntaxí. Má rozhraní k mnoha systémovým voláním a knihovnám i k různým okenním systémům a je rozšiřitelný v jazycích C nebo C++. Je také použitelný jako rozšiřující jazyk pro aplikace, které potřebují programovatelné rozhraní. A konečně, Python je přenosný: běží na mnoha variantách Unixu včetně Linuxu a MacOS i na Windows.

Python je podle serveru Statistic Times [41] na prvním místě, co se týče popularity (před jazyky Javascript, C, C++ nebo R). Tento programovací jazyk je populární především proto, že má všechny správné vlastnosti pro vývoj softwaru, který je v dnešní době skutečnou hnací silou celého světa vývoje softwaru. Strojové učení, robotika, umělá inteligence a věda o datech jsou vedoucími technologiemi současnosti i dohledné budoucnosti. Python je populární hlavně proto, že v těchto oblastech již má spoustu schopností, zatímco mnoho starších jazyků v těchto technologiích zaostává [39]. Dále se pomocí Pythonu často programují webové stránky (na straně serveru), obecně jakýkoliv software, skriptování, implementace matematické logiky [36]. Největší výhoda Pythonu je ale jeho čitelnost, kód je velmi jednoduchý na porozumění a proto je ideální i pro méně zkušené programátory. Backend aplikace Yoga poses je naprogramována pomocí Pythonu verze 3.8.3.



Obrázek 3.3: Graf nejpoužívanějších programovacích jazyků [41], na kterém lze vidět, že Python byl nejpoužívanější programovací jazyk v roce 2021. Používat tento jazyk tedy znamená, že je mnohem pravděpodobnější, že programátor řeší problém, který již někdo popsal a zveřejnil. Další výhodou může být to, že čím více programátorů jazyk používá, tím méně chyb se v něm bude nacházet.

3.1.2 Nejpoužívanější frameworky pro Python

Od představení Webu 2.0[1] se webové aplikace stávají čím dál tím více populárnější [3]. Vývoj takových aplikací se stal složitějším a požadavků, které se obvykle zabývají stejným základním problémem, přibývalo. Při vývoji se programátoři často setkávali s problémy, které již byly mnohokrát řešeny. Za účelem standardizace takových problémů začaly vznikat různé šablony a vzory určené k řešení opakujících se problémů. Takové šablony vznikaly proto, aby se při tvorbě kódu zachoval daný standard a řád i při vytváření složitějších informačních systémů. Tyto šablony jsou úzce spjaty právě s navržením frameworků, které opakované základní problémy řeší.

Frameworky můžeme rozdělit do tří kategorií: Full-stack frameworky, Microframeworky a Asynchronní frameworky [40]. Full-stack frameworky představují komplexní řešení pro všechny požadavky vývojářů. Generátory formulářů a jejich validace, rozvržení šablon pro HTML, moduly pro různé databázové systémy atd. Microframeworky jsou velmi odlehčené, nenabízejí abstraktní vrstvy pro databázové systémy, validaci formulářů a v základu nenabízejí další specifické nástroje a knihovny. Vývojáři, kteří využívají Microframeworky, musí většinou tyto specifické operace sami naprogramovat. Nejnovějším typem frameworků jsou Asynchronní frameworky, což je mikrosystém, který umožňuje zpracovávat velkou sadu souběžných připojení.

Mezi nejpopulárnější full-stack frameworky pro Python se řadí např. Django nebo Pyramid [38]. Django má zaslouženě pověst vysoce produktivního nástroje pro vytváření složitých webových aplikací. Je označován jako "webový framework pro perfekcionisty s blízkými termíny odevzdání" a zaměřuje se na rychlý vývoj s velmi dobře zdokumentovanými možnostmi využití pro běžné případy.

Flask je zase nejpopulárnější microframework [38]. Původně je inspirovaný frameworkem Sinatra pro programovací jazyk Ruby. Zaměřuje se na poskytování základní sady funkcí (zpracování požadavků, směrování, soulad s WSGI, šablonování) a nabízí modulární design pro přidání dalších potřebných funkcí. Jednoduchost tohoto frameworku lze spatřit již z aplikace Hello World (Výpis 3.1). Právě kvůli možnosti manuální kontroly formulářů a jednoduchého rozšíření byl pro tento projekt vybrán microframework Flask.

```
1 from flask import Flask
2
3 app = Flask(__name__)
4
5 @app.route('/')
6 def hello():
7     return 'Hello World!'
```

Výpis 3.1: Příklad aplikace Hello World pomocí Flasku.

Každá aplikace tvořená pomocí frameworku Flask musí mít na začátku vytvořenou instanci třídy Flask. Webový server poté všechny požadavky od klientů předává právě tomuto objektu, který je zpracuje. Používá pro to protokol WSGI (Web Server Gateway Interface) [17]. Klienti (webové prohlížeče) posílají požadavky na webový server, který je přepošle této instanci. Aplikace musí vědět, jaký kód spustit pro každou URL adresu, takže si uchovává mapování každé URL adresy na příslušnou Python funkci. Toto mapování je umožněno

[1]Populární termín označující novou generaci webu (resp. služeb internetu). Za fenomény WEB 2.0 se označují např. blogy, sociální sítě, RSS, Ajax atp.. Mezi servery ztělesňující WEB 2.0 patří např. Wikipedia, YouTube, Second Life a MySpace. Velký důraz je kladen na interaktivitu s uživatelem [29]

Flask	Django
Vytvořen v roce 2010.	Vytvořen v roce 2005.
Navržen pro rychlý vývoj.	Navržen pro jednoduchý vývoj aplikací.
WSGI framework.	Full-stack framework.
Podporuje rozšíření pomocí API.	Nepodporuje rozšíření pomocí API.
Podporuje více typů databází pro projekt.	Podporuje jeden typ databáze pro projekt.
V základu nemá podporu pro zpracování formulářů.	Má podporu pro zpracování formulářů.
Nepodporuje dynamické HTML stránky.	Podporuje dynamické HTML stránky.
Umožňuje rozmanitý styl práce.	Nabízí monolitický styl práce.

Tabulka 3.1: Porovnání vlastností frameworků Flask a Django [5].

pomocí dekorátoru funkce. V případě kódu 3.1 dekorátor `app.route('/')` se funkce `hello` spustí při přístupu na domovskou stránku a zobrazí zprávu `'Hello World'`.

3.1.3 Databázové systémy

Databáze je možné rozdělit na dva typy: SQL a NoSQL databáze [30]. Jedním z nejjednodušších a nejintuitivnějších způsobů shromažďování a prezentace dat je tabulka. Většina tabulkových dat lze číst a pochopit bez dalších vysvětlení. Každá tabulka má své jméno, první řádek, který popisuje formát dat a další řádky, které obsahují jednotlivé záznamy a jejich hodnoty. Každý záznam má jednoznačný identifikující atribut – většinou se jedná o unikátní číslo pro každý záznam nebo minimální kombinaci atributů, která vede k unikátní hodnotě. Každá tabulka má tedy následující vlastnosti:

- **Jméno** – každá tabulka musí mít své jméno,
- **Jména atributů** – každý atribut musí být v tabulce jedinečný a reprezentuje jeden sloupec,
- **Sloupce nemají pořadí** – počet ani pořadí sloupců v tabulce není důležitý a nemění význam tabulky,
- **Řádky nemají pořadí** – počet ani pořadí řádků v tabulce není důležitý a nemění význam tabulky,
- **Identifikační klíč** – jeden atribut nebo kombinace atributů, které jednoznačně identifikují záznam v tabulce.

Theorem 1 (Relační model) *Relační model reprezentuje data i vztahy mezi nimi. Matematicky to znamená, že jakákoliv relace R je jednoduše podmnožinou kartézského součinu domény: $R \subseteq D_1 \times D_2 \times \cdots \times D_n$ s D_i jako doména i-tého atributu/vlastnosti. Jakákoliv n-tice r, je množina dat $r = (d_1, d_2, \cdots, d_n)$ [30].*

Podle této definice, relační model považuje každou tabulku jako neseřazenou množinu n-tic. Relační model byl základem prvního relačního databázového systému. Dnes jsou jejich

sofistikovaní nástupci pevně zakotveni v mnoha praktických aplikacích [30]. SQL jde ruku v ruce s relačním modelem, protože pro reprezentaci dat používá právě tabulky [4].

Relační databáze jsou již dvacet let úspěšnou technologií, která poskytuje perzistenci, řízení souběžnosti a integrační mechanismus. Vývojáři aplikací byli frustrováni nesouladem impedance mezi rozhraním relačního modelu a datovými strukturami v paměti. Dochází k odklonu od používání databází jako integračních bodů směrem k zapouzdřování databází v rámci aplikací a integraci prostřednictvím služeb. NoSQL je náhodný neologismus. Termín NoSQL byl použit pouze pro název schůzky ohledně této technologie, když vývojáři chtěli nějak výstižně popsat toto setkání na Twitteru. Neexistuje žádná normativní definice – jediný způsob, jak určit vlastnosti NoSQL databáze je pozorováním společných charakteristik. Mezi společné charakteristiky [15] patří:

- Nepoužívají relační model,
- skvěle běží na clusterech,
- jsou open-source,
- jsou vytvořeny pro webové aplikace 21. století,
- nemají pevně dané schéma.

Pro tento projekt byla vybrána NoSQL databáze MongoDB [7]. Tato databáze tedy není relační, ale dokumentově orientovaná. Hlavním důvodem pro odklon od relačního modelu je snazší škálování, ale existují i další výhody. Dokumentově orientovaná databáze se odklání od řádků v tabulkách k souborům. Díky možnosti vkládat dokumenty a pole umožňuje dokumentově orientovaný přístup reprezentovat složité hierarchické vztahy pomocí jediného záznamu. V této době objektově orientovaného programování je toto pojetí databáze mnohem přirozenější. Neexistují ani žádná předdefinovaná schémata: klíče a hodnoty dokumentu nemají pevně dané typy ani velikosti. Bez pevného schématu je přidávání nebo odebírání polí podle potřeby snazší. Je také snazší experimentovat. Vývojáři mohou vyzkoušet desítky modelů pro data a poté vybrat ten nejlepší, kterým se budou zabývat více.

Jednoduché škálování MongoDB lze předvést na obrázku 3.4. V tomto případě šlo o pouhé přidání dalšího e-mailu k uživateli. Tato jednoduchá operace je v SQL databázi ale mnohem složitější, než prosté přidání do pole. V SQL databázi by bylo potřeba vytvořit další tabulku, kde by byl cizí klíč, který by odkazoval na uživatele, a jeden atribut, který by specifikoval hodnotu e-mailu.

Obrázek 3.4: Příklad jednoduchého škálování databáze. Záznam s jedním uživatelským e-mailem (vlevo) a záznam s více e-maily (vpravo).

Formát dokumentů MongoDB je založen na JSON[2], populárním schématu pro ukládání libovolných datových struktur. Ve formátu JSON se data přijímají, ale interně MongoDB využívá formát BSON [2]. Rozdíl mezi nimi je přehledně popsán v tabulce 3.2.

	JSON	BSON
Kódování	UTF-8 řetězec	Binární data
Podpora dat	Řetězec, bool, čísla, pole	Řetězec, bool, číslo, pole, časová známka, holá binární data
Čitelnost	Člověk a počítač	Jen počítač

Tabulka 3.2: Rozdíl mezi formátem JSON a BSON. Převzato z [32].

Vysoký výkon je hlavním cílem MongoDB [7] a ovlivnil většinu jejího návrhu. MongoDB přidává do dokumentů dynamickou výplň a předalokovává datové soubory, aby vyměnila dodatečnou spotřebu místa za konzistentní výkon. Využívá co nejvíce paměti RAM jako mezipaměť a snaží se automaticky vybírat správné indexy pro dotazy. Stručně řečeno, téměř každý aspekt MongoDB byl navržen tak, aby udržoval vysoký výkon. Ačkoli je MongoDB výkonný a snaží se zachovat mnoho funkcí relačních systémů, není určena k tomu, aby dělala vše, co relační databáze. Kdykoli je to možné, databázový server přenáší zpracování a logiku na stranu klienta (zpracovávají ji buď ovladače, nebo aplikační kód uživatele). Zachování tohoto zjednodušeného návrhu je jedním z důvodů, proč může MongoDB dosahovat tak vysokého výkonu.

3.2 Frontend

Vývoj front-endových webových aplikací může vyžadovat změnu pohledu, protože se jedná o velmi odlišný postup vývoje od vývoje pro jiné platformy. Hlavním rozdílem je, že kód napsaný pro front-end není spouštěn na serveru, ale na klientské straně v prohlížeči [1].

Vývoj front-endu u aplikací zahrnuje více jazyků – HTML, CSS, Javascript. V poslední době jsou velmi moderní i front-endové frameworky, které lépe interagují s uživatelem jako např. React nebo Angular.

3.2.1 HTML

HyperText Markup Language (HTML) [18] je programovací jazyk, který usnadňuje tvorbu webových stránek. Tento jazyk, který má kódová slova a syntaxi jako každý jiný jazyk, je poměrně snadno pochopitelný a s postupem času stále výkonnější v tom, co umožňuje vytvářet. Jazyk HTML je pod záštitou organizace World Wide Web Consortium, která tento jazyk navrhuje a udržuje, neustále vyvíjí, aby vyhovoval požadavkům a nárokům internetu; například s přechodem na Web 2.0.

HTML je soubor značkovacích symbolů nebo kódů vložených do souboru určeného k zobrazení na internetu. Značkování říká webovým prohlížečům, jak mají zobrazit slova a obrázky na webové stránce.

HTML stránky jsou jednoduché textové soubory. Tento jazyk používá tagy [12] (znaky, které jsou mezi symboly „<“ a „>“), které těmto řetězcům přidávají speciální význam. Tagy

[2]JSON je zkratka pro JavaScript Object Notation. Struktury JSON se skládají z klíčů a hodnot a mohou se vnořovat libovolně hluboko [2].

bývají také označovány jako HTML elementy. Tyto tagy jsou většinou párové, to znamená, že jeden tag označuje začátek a druhý konec. Každý tag může mít připsány další vlastnosti pomocí atributů. Tyto atributy musí mít vždy název a hodnotu. Různých tagů/elementů HTML jazyk definuje více než 70. Nové verze HTML5 [12] podporují i multimediální obsah jako jsou videa nebo spouštění zvuku.

```
<button class="btn" type="submit" title="Hover text">Potvrdit</button>
```

Obrázek 3.5: Příklad kódu v jazyku HTML. Modře jsou zvýrazněné tagy, oranžově atributy, žluté jsou hodnoty atributů a černě je obsah elementu.

Velmi podobným jazykem je XML, které umožňuje uživatelům definovat své vlastní značky. Tento jazyk se ale používá spíš pro uchování a posílání dat a ne pro programování webových stránek.

3.2.2 Stylování HTML dokumentů pomocí CSS

V prvních letech webu (1990–1993) byl jazyk HTML poměrně chudým. Skládal se téměř výhradně ze strukturálních prvků [31], které byly užitečné pro popis věcí, jako jsou odstavce, hypertextové odkazy, seznamy a nadpisy. Neobsahoval nic, co by se alespoň vzdáleně blížilo tabulkám, rámcům nebo složitému značení, které lze vidět na dnešních webových stránkách běžně. Obecnou myšlenkou bylo, že HTML bude strukturální značkovací jazyk, který se bude používat k popisu různých částí dokumentu. O tom, jak by se tyto části měly zobrazovat, toho bylo řečeno jen velmi málo. Jazyk se nezabýval vzhledem. Bylo to jen malé, čisté značkovací schéma. Po nějaké době od vyvinutí jazyka HTML byl ale kladen důraz na větší zkrášlení každé webové stránky. To vedlo k vytvoření speciálních elementů – např. element `FONT`, který upravoval podobu písma. HTML dokumenty by ale byly velmi složité na čtení, kdyby každý element měl u sebe připsáno celé své stylování. To vedlo k vyvinutí jazyka CSS v roce 1996.

CSS je zkratkou pro Cascading Style Sheets, v češtině se používá název kaskádové styly [19]. CSS umožňuje vytvářet pravidla, která určují, jak se má obsah prvku zobrazovat. Pomocí nich lze specifikovat například pozadí HTML elementu, písmo nebo zarovnání textu.

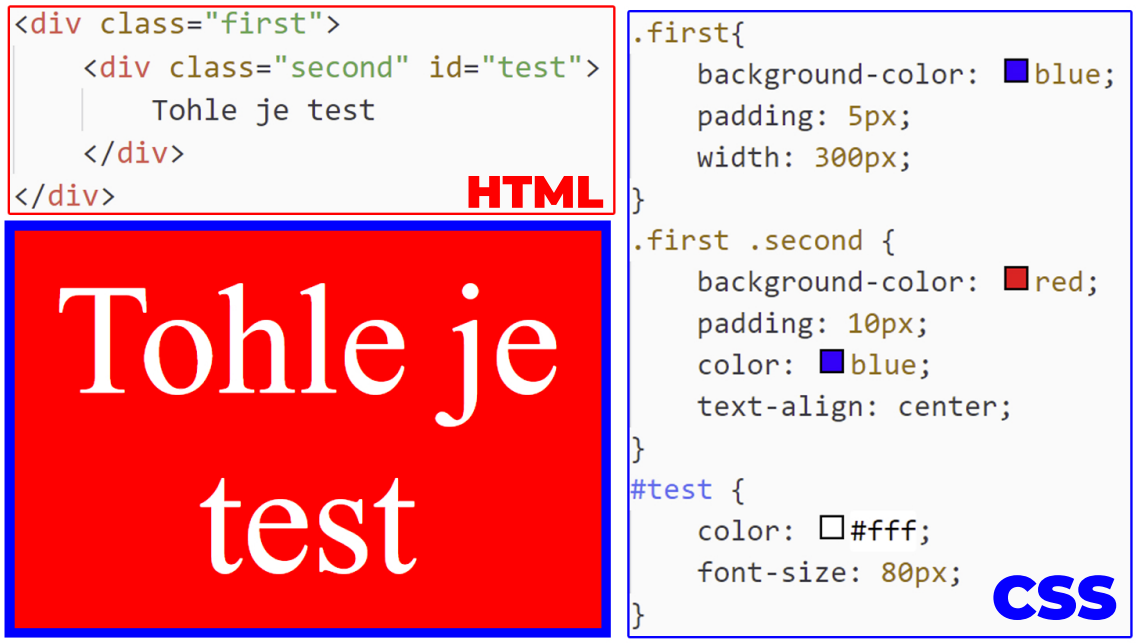
Pravidla pro stylování se skládají ze tří částí. První částí je selektor, který slouží pro adresování jednoho nebo více elementů HTML. Druhou částí jsou již jednotlivé deklarace stylů, které se dají rozdělit ještě na jméno vlastnosti a hodnotu [12]. Pro napsání selektoru je třeba znát určitá pravidla. Mějme element `div` s připsanou třídou `first-div` a id `number1`. Tento element je možné adresovat pomocí několika selektorů. Můžeme se zaměřit buď na jméno elementu, na třídu, na id nebo tyto vlastnosti použít všechny zároveň. Co liší tyto přístupy je ale specifičnost. Každý z těchto způsobů má svoji specifičnost [21], která by se od nejvyššího po nejnižší dala popsat takto:

1. styly napsané přímo v HTML dokumentu,
2. adresování pomocí atributu id,
3. adresování pomocí tříd nebo pseudo-tříd,
4. adresování pomocí jména elementu,
5. pořadí v souboru.

Pro každou kategorii se počítá počet prvků z dané kategorie, tedy pokud bychom v původním příkladě napsali selektor `div.first-div#number1`, měl by výsledný selektor u kategorie 2,3,4 poznamenanou jedničku. Tato specifičnost se využívá, pokud je jeden element adresován více selektory (připisují se mu stejné styly v jiných pravidlech). Prohlížeč stylový soubor prochází postupně shora dolů a pokud je více stejných stylů připsáno jednomu elementu, je připsán ten styl, který je u selektoru s větší specifičností. Vždy je sledován počet prvků u jednotlivých kategorií od nejvyšší po nejnižší. Pokud bychom měli selektory `#first .blue h1` a `#first .red.blue h1` a každý z nich by aplikoval jinou barvu na stejný element, vybrala by se barva z druhého selektoru.

Specifičnost je demonstrována na obrázku 3.6, kde je jednomu prvku vícekrát připsána barva písma. V tomto případě se vybere druhá barva, protože ačkoliv vrchní selektor obsahuje dvě třídy, tak spodní obsahuje jedno ID, což vede k vyšší specifičnosti a tudíž se aplikuje bílá barva na písmo.

CSS selektory mají svá pravidla, jak se jednotlivé prvky do selektoru píší. Pokud je potřeba například adresovat prvek uvnitř jiného elementu, píše se mezi jejich třídy/id/jména mezera. Pokud chceme adresovat prvek, který má připsány dvě třídy zároveň, mezera se nepíše. Pravidel je spousta a do toho se přidávají ještě tzv. pseudo-třídy, které přidávají další význam prvkům. Např. pseudo-třídy `:hover` nebo `:focus`, které aplikují styly na prvky, nad kterými je kurzor myši nebo je na něj zaměřeno. Všechny pravidla pro psaní selektorů a i veškeré CSS styly lze nalézt na W3Schools [44].



Obrázek 3.6: (Vlevo nahoře) HTML předpis pro prvek vlevo dole, (vpravo) CSS styly pro prvek vlevo dole.

3.2.3 Javascript

JavaScript je důležitý jazyk, protože je jazykem webového prohlížeče [8]. JavaScript je postaven na několika velmi dobrých a několika velmi špatných myšlenkách. Mezi velmi dobré nápady patří funkce, volné typování, dynamické objekty a expresivní objektový literální zápis. Mezi špatné myšlenky patří programovací model založený na globálních proměnných. Funkce v jazyce JavaScript jsou objekty první třídy s (většinou) lexikálním rozsahem. JavaScript je první lambda jazyk, který se stal součástí mainstreamu. Má více společného s Lispem a Scheme než s Javou[3]. Je to Lisp v oděvu jazyka C. To z JavaScriptu dělá pozoruhodně silný jazyk.

JavaScript je celosvětově rozšířený a již sedmým rokem po sobě se stal nejpoužívanějším programovacím jazykem (k roku 2019) [9], v roce 2019 jej používalo 67,8 % vývojářů. Jeho vzestup na pozici nejoblíbenějšího programovacího jazyka na světě je synonymem vzestupu samotného internetu. Více než 95,2 % webových stránek používá právě JavaScript (k roku 2019). JavaScript je skriptovací jazyk, který je jedním ze tří základních jazyků používaných při tvorbě webových stránek. Zatímco jazyky HTML a CSS dávají webové stránce strukturu a styl, JavaScript umožňuje přidávat na web funkce a chování, což návštěvníkům webu umožňuje interagovat s obsahem.

Skript napsaný v Javascriptu se dá psát přímo do HTML dokumentu. Častější je ale mít tyto jazyky oddělené v souborech a propojit je přes element script [13]. Jednotlivé elementy HTML stránky jsou zařazeny do struktury DOM – Document Object Model. Ten není součástí jazyka HTML ani Javascriptu. Je to samostatný soubor pravidel. Je implementován všemi hlavními výrobci prohlížečů a pokrývá dvě základní oblasti: vytváření modelu stránky HTML a přístup ke stránce HTML a její změny. Když prohlížeč načte webovou stránku, vytvoří její model a uloží jej do paměti. DOM specifikuje, jak tento model má být strukturován pomocí DOM stromu (viz obrázek 3.7b). Každý prvek tohoto stromu je objektem a reprezentuje nějakou část načtené webové stránky. DOM také specifikuje metody a vlastnosti, které mohou být použity při práci s tímto modelem. Metody zpřístupňují možnost, jak za běhu měnit strukturu celého DOM stromu a tím pádem i načtené webové stránky. Často se také DOM nazývá jako API. API umožňuje programům a skriptům komunikovat mezi sebou.

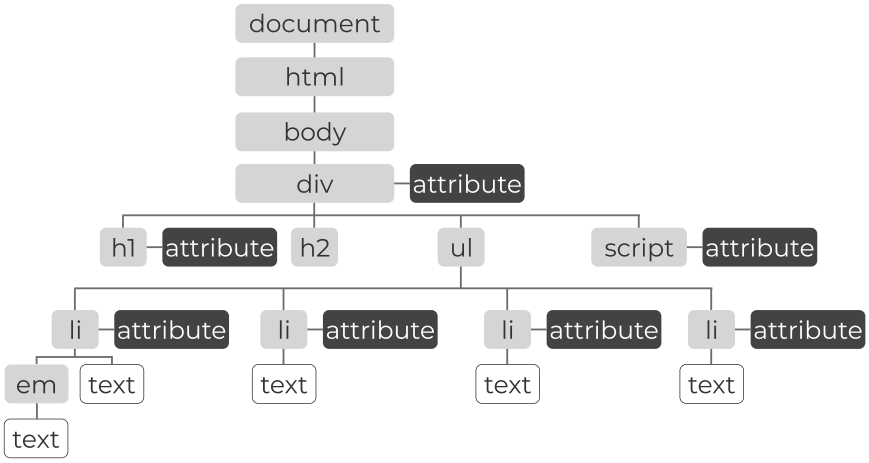
Elementy HTML dokumentu lze adresovat stejně jako v CSS selektorech, tedy kombinací jmen HTML tagů, pomocí ID nebo jmen tříd [13]. DOM specifikuje také speciální metody pro adresování elementů pouze pomocí jednoho z dříve zmíněných. Tyto metody navrací buď jeden nebo celé pole listů v DOM stromu. V celém stromu se navíc dá velmi jednoduše pohybovat. Od konkrétního listu je možné přejít na rodičovský, bratrský nebo synovské listy.

Dalším důležitým prvkem práce s DOM stromem v jazyce Javascript jsou události (events) [11]. JavaScript pro zajištění interaktivity s uživatelem využívá události (events). Události jsou vyvolány při různých akcích uživatele – kliknutí na prvek, přejetí kurzorem přes prvek, stisknutí klávesy atd.

[3] Často dochází k záměně těchto dvou jazyků, ale JavaScript a Java nemají téměř nic společného. Název JavaScript vznikl díky podpoře appletů Java v prohlížeči Netscape. Mnozí říkají, že to byla také marketingová taktika, jak odvést část pozornosti od Javy, která byla v té době nejrozšířenějším jazykem [9].

```html
<html>
    <body>
        <div id="page">
            <h1 id="header">List</h1>
            <h2>Buy groceries</h2>
            <ul>
                <li id="one" class="hot"><em>fresh</em> figs</li>
                <li id="two" class="hot">pine nuts</li>
                <li id="three" class="hot">honey</li>
                <li id="four" class="hot">balsamic vinegar</li>
            </ul>
            <script src="js/list.js"></script>
        </div>
    </body>
</html>
```

(a) HTML kód, který obsahuje všechny různé prvky HTML: tagy, atributy a jejich hodnoty.



(b) Vytvořený DOM strom pro HTML na obrázku 3.7a

Obrázek 3.7: Při tvorbě DOM stromu je důležité, aby byla zachována logická struktura kódu. Synovské uzly jsou vnořené tagy v html. Atributy obsahují všechny atributy připsané danému prvku v html a i jejich hodnoty. Textový uzel obsahuje textový obsah daného HTML elementu. Nad první HTML element `<html>` je vložen ještě kořenový uzel s názvem document, přes který se v JavaScriptu dá dotazovat na celý strom a pohybovat se v něm.

3.2.4 jQuery

jQuery je rychlá, malá a funkčně bohatá knihovna jazyka JavaScript [35]. Díky snadno použitelnému rozhraní API, které funguje v mnoha prohlížečích, usnadňuje například procházení a manipulaci s dokumenty HTML, zpracování událostí, animace a Ajax.

Knihovna jQuery poskytuje univerzální abstrakční vrstvu pro běžné webové skriptování, a je proto užitečná téměř v každé situaci skriptování. Základní funkce [6] nám však pomáhají plnit následující úkoly:

- Přístup k elementům v dokumentu – bez knihovny JavaScriptu musí vývojáři webových stránek často psát mnoho řádků kódu, aby prošli DOM strom a našli konkrétní části struktury dokumentu HTML. S nástrojem jQuery mají vývojáři k dispozici robustní a efektivní mechanismus selektorů, který usnadňuje vyhledání přesné části dokumentu, kterou je třeba zkontrolovat nebo s níž je třeba manipulovat.
- Změnit vzhled webové stránky – CSS nabízí účinnou metodu, jak ovlivnit způsob vykreslování dokumentu, ale je nedostatečná, pokud ne všechny webové prohlížeče podporují stejné standardy. Pomocí nástroje jQuery mohou vývojáři tuto mezeru překlenout a spolehnout se na stejnou podporu standardů ve všech prohlížečích. Kromě toho může jQuery měnit třídy nebo jednotlivé vlastnosti stylů aplikované na část dokumentu i po vykreslení stránky.
- Měnit obsah stránky – s jQuery je možné jednoduše změnit jak text, který je zobrazen na stránce, tak i obrázky nebo celou strukturu HTML dokumentu.
- Odpověď na interakce uživatele – knihovna také nabízí velmi jednoduchý způsob, jak naprogramovat odpovědi na různé události vyvolané uživatelem.

Tato knihovna také velmi usnadňuje asynchronní komunikaci v JavaScriptu [35].Pomocí metody `ajax` lze jednoduše specifikovat konkrétní parametry požadavku: Typ požadavku (POST, GET), url na které se má požadavek zaslat, data, která má požadavek přenést, a datový typ těchto dat. Také zde lze specifikovat, jak bude JavaScript reagovat na odpověď serveru nebo i na to, když server vůbec neodpoví.

```
$.ajax({
    type: "method",
    url: "url",
    data: "data",
    dataType: "dataType",
    success: function (response) {

    },
    error: function () {

    }
});
```

Obrázek 3.8: Metoda pro zaslání asynchronního požadavku pomocí knihovny jQuery – v metodě je možné specifikovat typ požadavku, url, kam se má daný požadavek zaslat, data, která budou přenesena v požadavku, jejich typ a také funkce, které se provedou v případě, že server odpoví (success) nebo v případě, že neodpoví (error)

3.2.5 GSAP

Knihovna pro JavaScript GSAP slouží pro jednoduchou tvorbu animací ve webových aplikacích. Pomocí metod lze naprogramovat animaci, která plynule mění CSS atributy jednotlivých prvků na stránce. Tyto animace se mohou za sebe skládat a vytvářet tzv. timelines, pomocí kterých se animace spouští jedna za druhou. Animace lze tvořit i v samotném CSS, ale pomocí této knihovny je mnohem jednodušší tyto animace ovládat.

Kapitola 4

Návrh aplikace

Před naprogramováním aplikace, bylo nutné vytvořit grafický design. Základní design hlavní stránky byl navržen v programu Adobe Photoshop 2021. Prvním velkým rozhodnutím bylo barevné schéma (viz obrázek 4.1).

Obrázek 4.1: Barevné schéma vytvořené pomocí aplikace Coolors

Oranžová v sobě spojuje teplo a žár červené s hravostí a radostí žluté. Přitahuje pozornost, aniž by byla tak odvážná jako červená, a používá se pro výstražné značky, jako jsou dopravní kužely a oblečení s vysokou viditelností. Je to energická barva, která může připomínat zdraví a vitalitu, vzhledem k jejímu zřejmému spojení s pomeranči a vitaminem C. Je to také mladistvá barva, která přináší prvek živosti a zábavy [28]. Právě kvůli tomu, jak je oranžová výrazná, zároveň nepůsobí negativně a připomíná zdraví a vitalitu, byla vybrána jako hlavní barva.

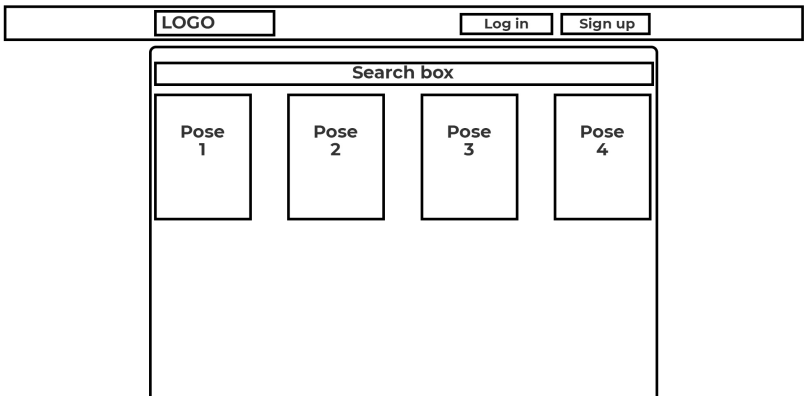
Další barvy vedle černé a bílé byly vybrány pomocí aplikace Coolors[1]. Ty byly využity pro detaily, jako je například zvýraznění obtížnosti jógové pozice nebo pro vytvoření jiného typu tlačítek a notifikací.

Po vybrání barev byl navržen jednoduchý wireframe (viz obrázek 4.2). Při tvorbě designu aplikace byla dodržována tři základní pravidla vizuální hierarchie [26]:

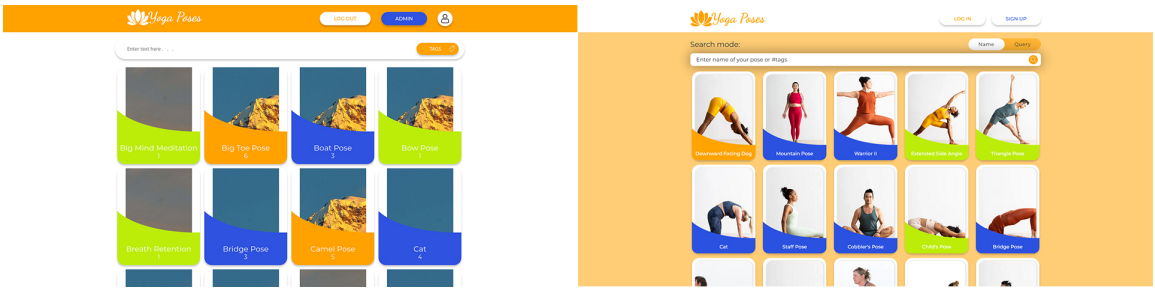
- Důležitější prvky musí být výraznější než ty méně podstatné,
- prvky, které spolu souvisí logicky, spolu musí být spojeny i vizuálně,
- u prvků, které jsou vloženy do jiných, musí být jasné, do kterých prvků vlastně patří.

Hlavní stránka byla rozdělena na tři části. Horní navigační lišta obsahuje logo aplikace na levé straně a na pravé přihlašovací a registrační tlačítko. Pod navigační lištou se nachází vyhledávací box, do kterého uživatelé budou moci zadávat jména jógových pozic,

[1] Aplikace je dostupná na adrese: `https://coolors.co/`



Obrázek 4.2: Tento wireframe sloužil jako základ/kostra pro vytvoření designu aplikace.



Obrázek 4.3: Původně navržený design (vlevo) a finální podoba aplikace (vpravo). Barevné vlny pod pozicí znázorňují obtížnost pozice – zelená znázorňuje jednoduché pozice, modrá středně těžké a oranžová pokročilé pozice.

které budou chtít vyhledat. Pod ním jsou vyobrazeny všechny jógové pozice v přehledných kartách. Každá karta obsahuje název pozice, fotografii a také obtížnost dané pozice. Karta splňuje několik důležitých faktorů [43] pro seskupené elementy. Všechny karty mají mezi sebou také určitý odstup, takže se jednotlivé karty nemohou míchat mezi sebou a uživateli se tak jógové pozice pohodlně prohlíží.

Veškerý obsah aplikace je v jejím horizontálním středu, který je široký 1200 pixelů, to zapříčiní, že webová stránka bude správně zobrazena i na menších monitorech. Velmi důležitým prvkem na každé webové stránce jsou tlačítka [26]. Z grafického návrhu tedy musí být jasné, na co uživatel může na stránce kliknout. Proto jsou všechna tlačítka zvýrazněna podbarvením a při přejetí přes ně se o pár pixelů celé zvětší. Zároveň se nad těmito tlačítky změní kurzor myši.

Po rozložení celé domovské stránky byl navržen již kompletní design, který kombinuje navržený wireframe a vygenerované barevné schéma. Tento design byl ještě dále upravován během implementace, aby se na hlavní stránku vešly všechny funkce aplikace. Porovnání původně navrženého designu a finálního designu je zobrazeno na obrázku 4.3.

Kapitola 5

Implementace

Aplikace je naprogramována pomocí 4 programovacích jazyků: Pythonu, HTML, CSS a JavaScriptu. Celý projekt je rozřazen do několika složek a souborů. Hlavní soubor `app.py` slouží pro vytvoření i spuštění webové aplikace. Všechny důležité soubory s kódem jsou ve složce `website`, kde v souboru `__init__.py` se provede potřebná konfigurace aplikace – navázání komunikace s databázovým serverem, vytvoření Flask aplikace, připsání tajného klíče aplikace, konfigurace e-mailového klienta, registrace blueprintů a konfigurace stránky, která se zobrazí při chybě 404. Další soubory a složky:

- `website/admin.py` – obsahuje funkce, které se spouští při práci s administrativními stránkami,
- `website/auth.py` – obsahuje funkce, které slouží pro registraci/přihlášení a odhlášení uživatele,
- `website/user.py` – obsahuje funkce, které jsou spouštěny převážně, když je uživatel přihlášen a spravuje své uložené informace (sestavy, vyhledávání, oblíbené pozice),
- `website/views.py` – obsahuje funkce, které jsou spouštěny z hlavní (domovské) stránky,
- `website/utils.py` – pomocné funkce, které se používají v celé aplikaci,
- `website/static/` – složka se statickými soubory jako např. ikony, JavaScript a CSS soubory,
- `website/templates/` – obsahuje HTML šablony, ve kterých je používán šablonovací styl Jinja2[1].

Výslednou aplikaci lze nalézt na adrese `https://yoga-poses-bp.herokuapp.com/`. Jelikož se jedná o webový hosting zdarma a databáze se nachází na jiném serveru, než samotná aplikace, je tudíž možné, že načítání stránek bude trvat delší dobu.

[1] Jinja je rychlý, expresivní a rozšiřitelný šablonovací engine. Speciální zástupné znaky v šabloně umožňují psát kód podobný syntaxi jazyka Python. Šabloně jsou pak předána data pro vykreslení výsledného dokumentu [20].

5.1 Struktura databáze

V této kapitole je popsána struktura NoSQL databáze. V databázi je celkem 11 typů záznamů, z čehož jich je 6 jen kvůli ukládání obrázků přímo do databáze. Tyto záznamy by mohly být také spojeny ve dva, ale při implementaci se ukázalo, že je mnohem přehlednější, když každý typ obrázků (pozice, sestava nebo uživatelský obrázek) má pro sebe vyhrazenou svoji vlastní kolekci. Pro ukládání těchto obrázků je použit modul `GridFS`, který se stará o rozkládání obrázků do databáze a také následné skládání, když o ně uživatel požádá. Každý záznam v databázi má své unikátní číslo (primární klíč), které je vygenerované vždy pomocí python modulu uuid a je uloženo v povinném argumentu každého záznamu `_id`.

Základem celé webové aplikace je samozřejmě záznam pozice. Pozice jsou na stránce zobrazovány v přehledných kartách. Každá pozice má následující atributy:

- **EngName** – anglický název pozice,
- **HindiName** – název v jazyce Sanskrt, který je často využíván jako originální název každé jógové pozice,
- **Description** – detailní popis pozice,
- **Difficulty** – obtížnost vyjádřená číslem od 1 do 5, kde 1 znamená nejjednodušší a 5 znamená nejtěžší,
- **Approved** – tato pravdivostní hodnota není v nynější verzi aplikace více využita. V databázi je hlavně kvůli budoucím aktualizacím, u kterých je možné, že se pozice prvně budou posílat na schválení administrátorům a až poté budou zveřejněny.

Dalším velice důležitým prvkem databáze jsou uživatelé. Uživatelé databázi velmi ovlivňují – přidávají do ní nové záznamy, upravují je a mažou. Každý uživatel má uložena data:

- **Email** – e-mailová adresa uživatele,
- **Username** – uživatelské jméno,
- **Password** – uživatelské jméno, respektive hash hesla, které si uživatel zvolil při registraci. Tento hash je vygenerován pomocí algoritmu sha256.
- **BornDate** – datum narození uživatele,
- **Created** – datum a čas vytvoření účtu,
- **Admin** – pravdivostní hodnota, která určuje jestli daný uživatel má připsaná administrátorská práva nebo nikoliv,
- **SubAdmin** – tato pravdivostní hodnota není v nynější verzi aplikace využita. V databázi je přítomna, protože je možné, že v budoucnu bude přidána nová role, která bude rozlišovat jiné stupně administrátorských práv.
- **Blocked** – pravdivostní hodnota, která určuje, jestli je uživatelský účet zablokovaný nebo ne. Pokud je nastavená na hodnotu `true`, uživateli je zamezeno přihlášení na webovou stránku.
- **Favourites** – pole primárních klíčů pozic, které uživatel označil jako oblíbené.

- **ProfileImage** – pravdivostní hodnota, která určuje, jestli daný uživatel má vlastní profilový obrázek.

Tag je krátký jednoslovný řetězec, který vystihuje nějakou vlastnost pozice, ke které je připsán. Tagy jsou klíčové pro vyhledávání pomocí jazyka queries (zpracování a fungování queries je popsáno v kapitole 5.6). O tomto prvku systému není třeba toho ukládat mnoho a přitom nabízí velmi silnou pomůcku při vyhledávání:

- **Name** – jednoslovný řetězec,
- **Positions** – pole primárních klíčů pozic, které mají daný tag připsaný,
- **Lower** – podoba tagu tvořena pouze malými písmeny, která je využita při vyhledávání v databázi, aby v databázi nebyli uloženy duplikáty, které se liší pouze velikostí písmen.

Pojmem queries není označen pouze jazyk použitý pro složitější logické vyhledávání. Jsou tak také nazvány jednotlivé záznamy uloženého vyhledávání. V databázi jsou uloženy tyto atributy:

- **Name** – název, pod kterým si uživatel danou query uložil,
- **User** – primární klíč uživatele, který query uložil,
- **Postfix** – pole jednotlivých prvků výrazu v postfixovém formátu,
- **Last-processed** – čas posledního vyhodnocení dané query,
- **Created** – čas uložení query,
- **Last-updated** – čas poslední úpravy query,
- **Last-count** – počet pozic, které byli nalezeny při posledním vyhodnocení,
- **Published** – pravdivostní hodnota, která určuje jestli svoji query uživatel sdílí s ostatními uživateli,
- **PlainText** – řetězec, který se využívá pro vyhledávání duplikátů. Pokud si uživatel chce uložit query, která má stejný tento atribut, je mu v tom zabráněno,
- **Infix** – výraz ve tvaru, který uživatel zadal do vyhledávacího pole. Tento atribut je využit při vkládání uložené query do vyhledávacího pole, aby byl správně zvýrazněn pomocí syntaxového zvýrazňovače,
- **Copiers** – seznam primárních klíčů uživatelů, kteří si danou query zkopírovali na svůj přehled queries.

Jak již bylo zmíněno, klíčovou součástí jógy jsou sestavy. Sestavu budou moci uživatelé složit pomocí pozic, které již v databázi jsou a připsat ke každé čas, jak dlouho v dané pozici musí sportovec setrvat. O sestavě je nadále potřeba uložit i další atributy. V databázi o každé sestavě je tedy uloženo:

- **Name** – jméno vytvořené sestavy,
- **Creator** – primární klíč uživatele, který vytvořil sestavu,

- **Created** – čas vytvoření sestavy,
- **Last_update** – čas poslední úpravy sestavy,
- **Published** – pravdivostní hodnota, která určuje jestli svoji sestavu uživatel sdílí s ostatními uživateli,
- **Last_training** – čas posledního cvičení sestavy,
- **Process** – pole, které je složeno z objektů. Tyto objekty vždy obsahují název pozice, primární klíč pozice a čas. Tím, že jsou v seřazeném poli se toto dá využít pro následnou projekci v asistentu cvičení jógových pozice (Kapitola 5.9),
- **Description** – textový popis sestavy,
- **Total** – doba celé sestavy v sekundách,
- **Difficulty** – obtížnost sestavy,
- **Time_str** – čas ve formátu řetězce (např. 6 min 7 sec),
- **Copiers** – seznam primárních klíčů uživatelů, kteří si danou sestavu zkopírovali na svůj přehled sestav.

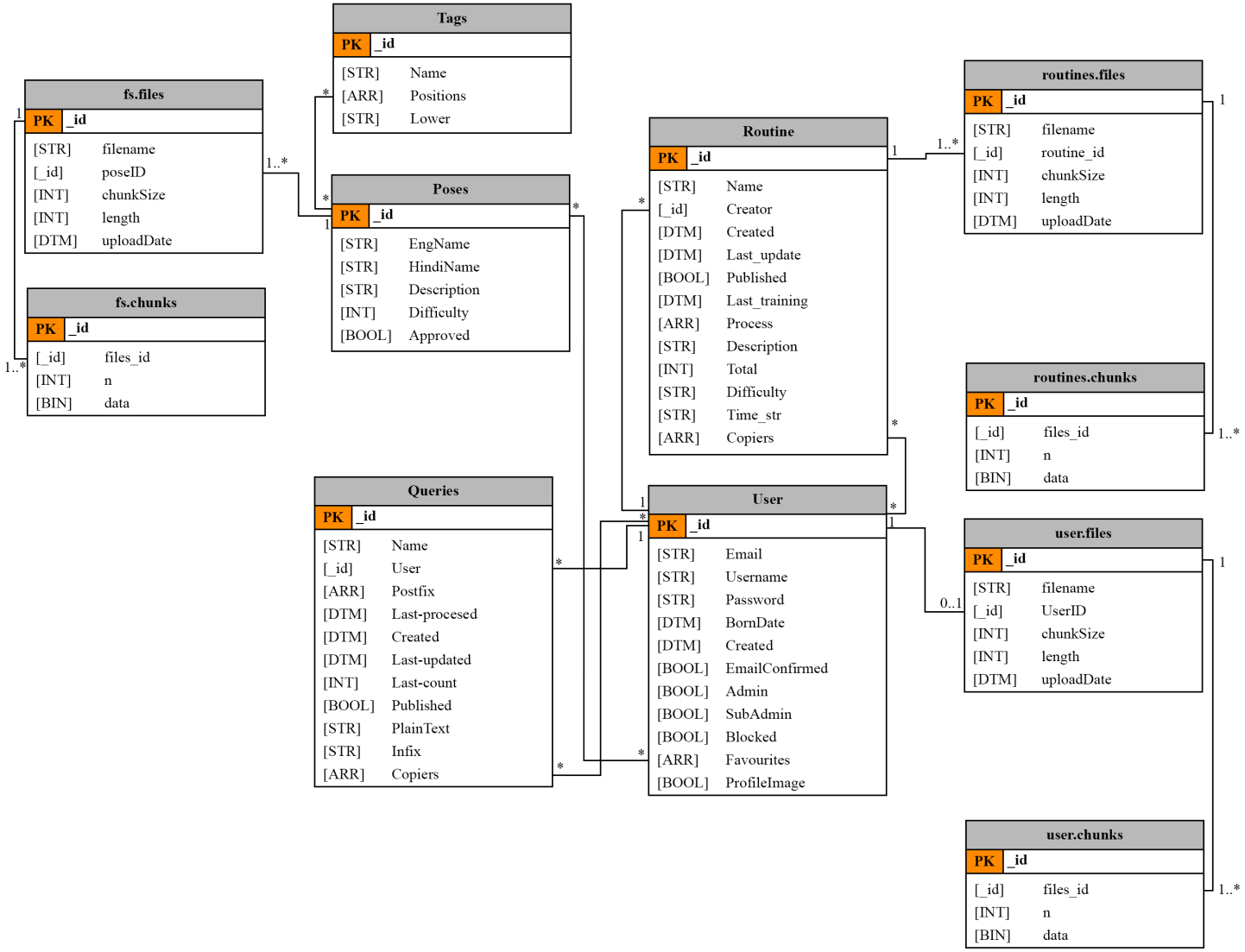
Kolekce fs.files, routines.files, users.files slouží pro ukládání obrázků různých typů. Jsou vytvořeny automaticky modulem `GridFS`. Do těchto záznamů se ukládají obecné informace o ukládaném obrázku:

- **filename** – jméno ukládaného obrázku,
- **poseID**, **routineID** nebo **userID** – primární klíč pozice, sestavy nebo uživatele, ke kterému daný obrázek přísluší,
- **chunkSize** – velikost jednotlivých kusů souboru[2],
- **length** – velikost ukládaného souboru v bytech,
- **uploadDate** – čas uložení nového souboru.

Kolekce fs.chunks, routines.chunks, users.chunks obsahují konkrétní binární data uložených obrázků.

- **files_id** – primární klíč k záznamu obecných informací o obrázku,
- **n** – pořadové číslo kusu binárních dat,
- **data** – samotná binární data.

[2]Maximální velikost BSON souboru je 16 MB, proto je potřeba větší obrázky rozdělit na části a poté je spojit [33]



Obrázek 5.1: Entity-relationship diagram pro databázi, která je použita aplikací Yoga Poses

5.2 Registrace a přihlášení

Uživatelé se do webové aplikace musí zaregistrovat, aby mohli využívat všechny její funkce, jako např. ukládat si oblíbené pozice, tvoření a cvičení jógových sestav nebo ukládání hledání. Při přístupu na stránku se uživatel dostane na registrační stránku přes tlačítko vpravo nahoře. Tyto tlačítka jsou při přihlášení uživatele nahrazena ikonou uživatele, která poté slouží jako rozcestník pro uživatelské funkce.

Při registraci musí uživatel zadat: uživatelské jméno, e-mail, heslo a datum narození. Heslo musí obsahovat alespoň jedno velké písmeno, jedno malé písmeno a číslici. Zároveň musí být alespoň 6 znaků dlouhé, ale nesmí přesáhnout délku 20 znaků. Všechny tyto podmínky jsou napsány přímo pod zadáváním hesla a postupně mizí ty, které jsou již splněny. Heslo poté musí uživatel zadat znovu, aby se předešlo překlepům v požadovaném heslu. Při zadání e-mailu se zasílá asynchronní požadavek na server s dotazem, jestli danou e-mailovou adresu někdo již používá nebo ne. Při odpovědi serveru se zobrazí buď ikonka potvrzení nebo ikonka s křížkem.

Po vyplnění všech všech polí a kliknutí na tlačítko `Register` se uživateli zašle potvrzovací e-mail. V této zprávě je potvrzovací odkaz, přes který uživatel potvrdí svoji e-mailovou adresu a může se přihlásit. E-mail je zaslán pomocí modulu `flask_mail`, který poskytuje jednoduché API pro přihlášení do e-mailu a následné odesílání zpráv.

Uživatel po přihlášení o sobě může změnit prakticky jakoukoliv informaci: uživatelské jméno, datum narození, heslo, ale i e-mail. Kvůli bezpečnosti pro změnu hesla je třeba znát heslo původní. Pokud si uživatel chce změnit e-mail, přes který se přihlašuje, zažádá si o 6 místný kód, který mu dojde na stávající e-mail. Po prokázání se tímto kódem je tedy jasné, že jde skutečně o uživatele, který vlastní i e-mail původní. Uživatel si také může přidat profilový obrázek, který se zobrazuje v pravém horním rohu.

5.3 Livesearch

Livesearch zobrazuje výsledky hledání již při zadávání vyhledáváného výrazu, tedy pokaždé, když se nějak změní vyhledávací výraz, upraví se výsledky. Toto živé vyhledávání je prováděno čistě na straně klienta, což předchází zahlcení serveru příliš častými dotazy. Livesearch se nachází na hlavní stránce a je to první způsob, kterým může uživatel vyhledávat v databázi pozic.

Algoritmus vyhledávání je implementován v souboru `liveSearchHome.js`. Pomocí JavaScript knihovny jQuery je při načtení domovské stránky aplikován naslouchač událostí na vyhledávací pole, který spustí algoritmus při zachycení události `keyup`. Tato událost je vyvolána vždy, když uživatel uvolní tlačítko na klávesnici a tedy pokaždé když vloží nebo vymaže znak z vyhledávacího pole.

První verze algoritmu postupovala následovně:

1. Uloží a pomocí metody trim() vymaže bílé znaky z textu, který je v současnosti ve vyhledávacím poli. Tento očištěný řetězec se uloží do proměnné `text`.

2. Z uloženého vyhledávacího řetězce se vytvoří regulární výraz. V JavaScriptu pro to slouží třída `RegExp`. Regulární výraz je vytvořen tak, aby byl splněn každým řetězcem, který obsahuje podřetězec `text`. Nutno podotknout, že řetězec v proměnné `text` je převeden na malá písmena, aby byly ignorovány rozdíly mezi velkými a malými písmeny.

3. Poté se pomocí metody each z knihovny jQuery iteruje přes všechny jógové karty, ve kterých se nachází název pozice. Tento název je převeden opět na malá písmena a porovnán s vytvořeným regulárním výrazem pomocí metody `match`.

   (a) Pokud metoda vyhodnotí, že název dané pozice nesplňuje regulární výraz, je pomocí jQuery metody přidána třída `textNotChecked`, která zapříčiňuje, že se daná karta pozice nezobrazí

   (b) Pokud metoda vyhodnotí, že název dané pozice splňuje regulární výraz, je třída odebrána. Pokud daná karta nemá připsanou třídu `textNotChecked`, metoda `removeClass` neprovede nic.

Toto byla první neoptimalizovaná verze algoritmu pro livesearch. Často se totiž stává, že algoritmus chce skrýt karty, které již neviditelné jsou, nebo naopak, chce zobrazit karty, které zobrazené jsou, protože se iteruje přes všechny karty pokaždé, když uživatel změní vyhledávaný výraz. Druhá a finální verze algoritmu tuto chybu nedělá. Při prodloužení vyhledávaného výrazu se dívá pouze do pozic, které jsou viditelné. To kvůli tomu, že prodloužením vyhledávaného výrazu se prakticky zpřísní kritérium, které musí jméno pozice splňovat. Při zkrácení vyhledávaného výrazu algoritmus iteruje pouze přes neviditelné pozice, protože se kritérium, které musí splňovat název jógové pozice, zmírní.

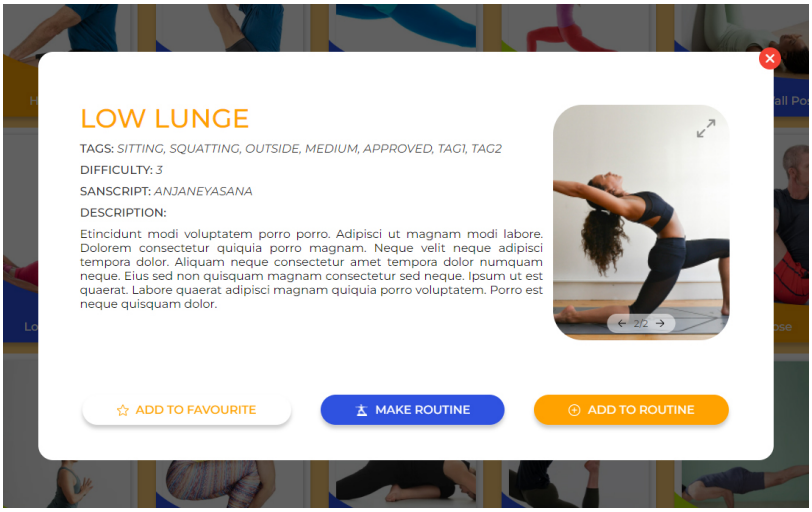
Textové pole není implementováno přes obyčejný element `<input>`, nýbrž přes `<div>`, který má zapnutou vlastnost `editableContent`. To znamená, že do tohoto elementu mohou uživatelé volně psát. Důvodem pro tento styl implementace je to, že do elementu `<input>` se nedá vložit žádná další struktura. Hodnota tohoto elementu může být čistě textová. Za to do upravitelného divu se dají vkládat další elementy. Možnost vkládat další elementy do pole, kde se píše je velmi užitečné. Vyhledávací pole na hlavní stránce totiž umožňuje i jednoduché vyhledávání pomocí tagů.

Při vložení znaku `#` do vyhledávacího pole se do něj přidá nový prázdný element `<input>`, do kterého uživatel může zadat libovolný název tagu, který je v databázi. Tento `<input>` pomocí elementu `<datalist>` poskytuje i jednoduchý našeptávač. Při stisknutí klávesy enter, se tento tag zkontroluje (jestli v databázi existuje) a poté se vyšle požadavek na server, který odpoví seznamem ID pozic, které tento tag mají připsaný. Při vložení dalších tagů se vyhledávají pozice, které mají připsány všechny zadané tagy.

5.4 Zobrazení detailu pozice

Na každou kartu s jógovou pozicí může uživatel kliknout. Při kliknutí je odeslán asynchronní požadavek na server, na který odpovídá detailními informacemi o požadované pozici. Při obdržení těchto informací je uprostřed stránky zobrazeno okno (Obrázek 5.2), které překrývá celý obsah stránky. V tomto detailním okně jsou zobrazeny prakticky veškeré informace, co jsou o pozici uloženy v databázi: název v anglickém jazyce, připsané tagy, obtížnost, sanscript název, popis pozice, všechny obrázky. Uživatelé mají možnost zvětšit obrázky po kliknutí na ikonu v pravém horním rohu fotografií.

Je zde také tlačítko, které přihlášeným uživatelům přidá prohlíženou pozici do seznamu oblíbených. Pro toto je využit asynchronní požadavek na server, který zobrazenou pozici uloží do uživatelských oblíbených pozic v databázi. Po úspěšném přidání je uživatel informován zelenou notifikací v horní části webové aplikace. Ostatní tlačítka budou popsány v kapitole 5.8, protože úzce souvisí s tvorbou jógových sestav.



Obrázek 5.2: Po kliknutí na libovolnou kartu s pozicí se uprostřed zobrazí toto okno s detaily o kliknuté pozici. Každá pozice může mít i několik obrázků, proto byl implementován obrázkový kolotoč (image corousel), který je umístěn vpravo nahoře. Obrázky se dají zvětšit kliknutím na ikonu v pravém horním rohu obrázku.

5.5 Tagy

Přihlášení uživatelé mohou volně přidávat tagy k jakékoliv pozici. Po přihlášení se uživateli na hlavní stránce objeví tlačítko v pravém dolním rohu. Po kliknutí na něj se zobrazí malé okno, kde může uživatel specifikovat, jaké tagy chce k jednotlivým pozicím přidat. Mohou to být tagy, které se již v databázi nachází nebo tagy úplně nové. Je zde také možnost přidat více tagů zároveň. Pro vložení více tagů zároveň musí uživatel jednotlivé tagy oddělit mezerou.

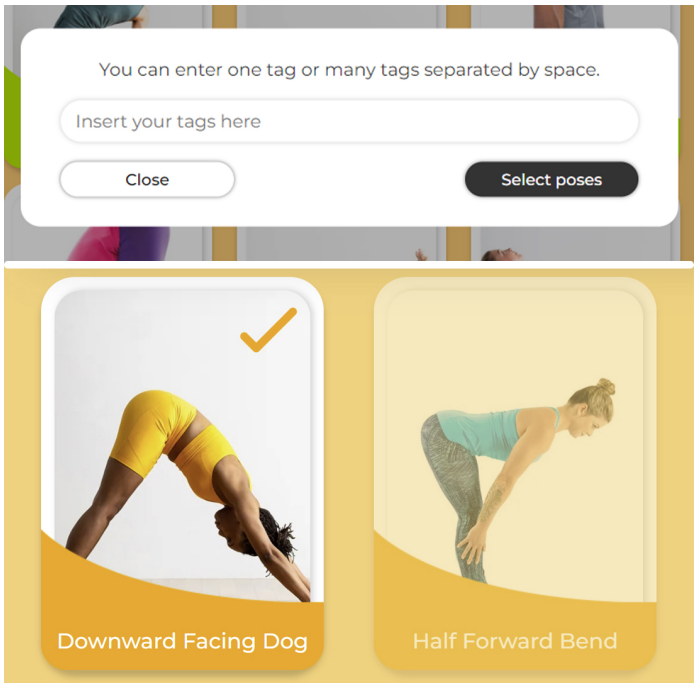
Po kliknutí na tlačítko `Select Poses` se uživateli zpřístupní možnost vybírat jednotlivé pozice, ke kterým chce zadané tagy přidat (rozdíl mezi vybranou a nevybranou pozicí lze vidět na Obrázku 5.3). Výhodou je také, že při tomto označování pozic může uživatel používat livesearch nebo i pokročilé vyhledávání pomocí queries. Je zde také tlačítko, které ulehčí uživateli vybírat více pozic. Pokud by například měl uživatel vyhledanou nějakou množinu pozic (více o ukládání hledání v kapitole 5.6) a chtěl k nim přidat zadané tagy – klikne na tlačítko v pravém dolním rohu `Select displayed poses` a označí tak všechny viditelné pozice. Pozice, které byly vybrány se od nevybraných liší připsanou třídou `selected-for-tag-configure`.

Pokud uživatel zjistí během vybírání pozic, že tagy z nějakého důvod přidat nechce, je zde tlačítko s ikonou křížku, kterým celý výběr zruší a vrátí se tak na výchozí domovskou stránku. Pokud uživatel potvrdí svůj výběr, je zaslán asynchronní požadavek se seznamem ID pozic, které byly vybrány. Tzn. při kliknutí na tlačítko potvrzení se iteruje přes všechny pozice, které mají připsánou třídu `selected-for-tag-configure` a přidá se ID této pozice do pole. Toto ID je dostupné u každé karty v atributu `data-poseid`[3]. Pole identifikačních klíčů zvolených pozic je zasláno na server pomocí asynchronní metody POST protokolu

[3]Atribut `data-*` slouží k uložení vlastních dat, která jsou privátní pro danou stránku nebo aplikaci. Název atributu by neměl obsahovat žádná velká písmena a musí být dlouhý alespoň jeden znak za prefixem „data-“ [44]. Jako hodnota tohoto atributu může být libovolný řetězec.

HTTP. Data jsou přenášena pomocí formátu JSON. V požadavku POST je také textová reprezentace zadaných tagů, které uživatel chce k pozicím přidat.

Po přijetí dat serverem se data z JSON formátu extrahují a iteruje se přes zadané tagy. Vždy se kontroluje, jestli daný tag již v databázi neexistuje. Toto hledání probíhá v case-insensitive formě, aby se předešlo stejným tagům s odlišnou velikostí písmen. Pokud tag v databázi již existuje, jsou pozice spojeny s těmi pozicemi, které má tag již připsány. To je implementováno jako funkce disjunkce mezi dvěma poli – tedy do pole, které má tag uložené jako seznam pozic, se přidají pouze ty, které v něm ještě nejsou. Pokud tag v databázi neexistuje, je vytvořen nový a jako pole pozic je nastaveno celé zaslané pole pozic. Po dokončení je zaslána potvrzovací zpráva na Front-end, která se uživateli zobrazí formou zelené notifikace v horní části obrazovky.



Obrázek 5.3: (Nahoře) Okno, které se uživateli zobrazí při vkládání tagů. (Dole) Při vkládání tagů uživatel označuje pozice, ke kterým chce nové tagy přidat. Vlevo je již označená pozice a vpravo je neoznačená.

5.6 Jazyk queries

Jednoduché vyhledávání pomocí tagů je zpracováno již v metodě livesearch, ovšem to uživateli nemusí stačit, a proto byl vyvinut logický vyhledávací jazyk queries. Do tohoto módu vyhledávání se uživatel může dostat přes přepínací tlačítko v pravém horním rohu nad standardním vyhledáváním (Obrázek 5.4).

Tato metoda vyhledávání umožňuje logicky kombinovat tagy. Uživatel tak může velmi přesně specifikovat požadovaný výsledek vyhledávání, tedy které tagy mají výsledné pozice mít a které ne. Díky tomu je snadné nalézt například pozice, které se necvičí ve stoje a zároveň jsou jednoduché.

Pro vyhodnocení výrazů byl implementován jednoduchý kompilátor, který sestává ze 4 částí:

Obrázek 5.4: Zadaný výraz v jazyce queries je uživateli syntakticky zvýrazněn. V pravé části jsou tlačítka pro potvrzení vyhledávání a pro uložení zadaného výrazu.

- Lexikální analýza (Lexer) – úkol lexeru je číst jednotlivé části výrazu (tokeny) a zasílat je překladači [14]. Jelikož výrazy jsou v případě queries velmi jednoduché, lexer vždy projde celý výraz a vytvoří z něj pole tokenů, které se dále zpracovávají v syntaktické analýze.
- Syntaktická analýza – Syntaxem jsou rozuměna pravidla, která definují strukturu jazyka. Syntax udává pravidla, jak mohou být jednotlivé tokeny uspořádány za sebou [45].
- Převod infixového tvaru výrazu do postfixového.
- Vyhodnocení postfixového výrazu.

První tři body tohoto algoritmu jsou vykonávány na straně klienta. Tedy až na vyhodnocení samotného výrazu není zatížen server. Také je důležité zdůraznit, že tato metoda vyhledávání není nijak spjatá s metodou livesearch, tzn. vyhledávání se provádí pouze při potvrzení uživatelem.

Lexikální analýza prochází výraz znak po znaku a podle prvního ve slově pozná, o který token se jedná. Pokud je prvním znakem #, jedná se o tag. Pokud je prvním písmenem A, O nebo N, může se jednat o klíčové slovo. Pokud je prvním znakem závorka, kontroluje se, zdali je toto slovo delší než jedna. Je totiž možné, že uživatel zadá výraz, kde mezi závorky nenapíše mezeru. Klíčová slova v jazyce jsou pouze tři a každý reprezentuje unikátní operaci mezi polemi pozic (Operace a jejich precedence je popsána v Tabulce 5.1). Tokeny jsou tedy trojího typu: tag, klíčové slovo, závorka. Po průchodu celým vyhledávacím výrazem je výsledkem pole tokenů, které je dále zkontrolováno pomocí syntaktické analýze.

Operace	Počet operandů	Precedence	Asociativita	Výsledek operace
NOT	1	3	zleva-doprava	Všechny pozice, které nemají připsaný daný tag
AND	2	2	zprava-doleva	Všechny pozice, které mají připsané oba dva tagy
OR	2	1	zprava-doleva	Všechny pozice, které mají připsaný alespoň jeden z tagů

Tabulka 5.1: Precedenční tabulka, která určuje pořadí jednotlivých operací. Je zde také vypsán význam jednotlivých operací a počet jejich operandů.

Syntaktická analýza pracuje s tabulkou pravidel (Tabulka 5.2), která má specifikováno, které tokeny mohou následovat nynější token. Po průchodu polem je možné pokračovat

dalším bodem algoritmu, ovšem pokud je zde nějaká syntaktická chyba, je na ni uživatel upozorněn a další body algoritmu nemá smysl provádět. Po ověření syntaxe je výraz převeden do postfixové notace.

Token	Následující tokeny
AND	NOT, TAG, (
OR	NOT, TAG, (
NOT	TAG, (
TAG	AND, OR,), @
(	NOT, TAG, (
)	AND, OR,), @

Tabulka 5.2: Pravidla, která jsou využívána v syntaktické analýze výrazu query. Při iterování přes všechny tokeny je kontrolováno, zdali za nimi následuje token z této tabulky. Symbol @ označuje, že daný token může být na posledním místě ve výrazu.

Algoritmus na převod infixové notace do postfixové je převzat z článku [10]. V tomto článku se autoři zaměřují na paralelní převod. Pro aplikaci Yoga Poses byl použit sekvenční algoritmus (Výpis 5.1). Algoritmus také využívá precedenci operátorů, která určuje pořadí vyhodnocení jednotlivých operací (precedence operací je vypsána v Tabulce 5.1) Klíčové pro postfixovou notaci je, že se výraz může vyhodnocovat zleva doprava, což je pro počítač ideální. Po převodu je postfixová notace zaslána na server asynchronním požadavkem, kde se výraz vyhodnotí.

Vyhodnocení výrazu v jazyce queries je implementováno ve funkci `process_querry` v souboru `utils.py`. Na začátku algoritmu je potřeba z databáze vybrat všechny pozice, respektive jejich ID, protože toto pole je použito při operaci negace (Význam operací je popsán v tabulce 5.1). Pokud je během algoritmu zpracováván tag, vyhledají se v databázi pozice, které tento tag mají připsaný a toto pole se dá na zásobník. Pokud algoritmus zpracovává operaci, vezme a odstraní operandy ze zásobníku a mezi nimi provede danou operaci (s vyjímkou negace, kde je potřeba operand jenom jeden). Po provedení všech operací musí na zásobník zbýt pouze jedno pole – pokud by jich zbylo více, znamená to sémantickou chybu v zaslaném výrazu. Zpět na front-end se pošle pole ID pozic a pozice, které nemají ID v tomto poli, jsou schovány připsáním třídy `querryNotChecked`.

Přihlášení uživatelé si mohou vyhodnocené queries uložit a poté je nalézt v záložce `My queries` na svém účtu, kde je mohou dále spravovat. Na hlavní stránce se uložené queries skrývají pod symbolem záložky nad vyhledávacím polem. Po kliknutí se uživateli zobrazí nabídka jeho uložených queries a po kliknutí na některou z nich se automaticky tato query doplní do vyhledávacího pole a provede se vyhledávání.

Navíc pro zvýšení uživatelského komfortu je zde implementován také syntaktický zvýrazňovač, který každý typ tokenu zobrazuje jinou barvou. V prvních verzích tohoto zvýrazňovače byl použit pouze jeden element `<div>`, do kterého se, jak uživatel psal, vkládaly elementy `<span>` s odlišnými třídami. Nevýhodou tohoto přístupu ale byl problém s kurzorem psaní. Při vložení nového elementu `<span>` totiž docházelo k posunu kurzoru psaní na nesprávné místo. Navíc manuální posun tohoto kurzoru je velice složitý, jelikož jiné prohlížeče s ním pracují odlišně. Proto byla implementována druhá verze tohoto zvýrazňovače, která používá elementy `<div>` dva. Jeden je pro uživatelský vstup a v druhém se zobrazuje zvýrazněná verze zadávaného výrazu. Tyto elementy jsou umístěny přes sebe, takže uživateli se zdá, že element je zde pouze jeden. V elementu do kterého uživatel píše je nastavená barva textu na průhlednou, ale barva ukazovátka (ukazatel, kde uživatel právě

píše další znak) je ponechána na černé barvě. To vede k tomu, že se uživateli zdá, že přímo píše do elementu, na který je zaměřen, ve skutečnosti se mu ale zobrazuje zvýrazněný text v elementu pod ním.

Zvýrazňování funguje podobně jako lexikální analýza. Při každém stisknutí tlačítka se iteruje přes každé slovo v zadávaném výrazu. Tagy a klíčová slova jsou ohraničeny elementem `<span>`, který má vždy připsanou třídu podle typu slova. Tato třída pouze přebarvuje text, který je ohraničen elementem `<span>`.

Mimo syntaktického zvýrazňování je také implementován našeptávač tagů, který se ukáže vždy při zadání znaku #. Tento našeptávač nabízí tagy, které jsou uložené v databázi. Po kliknutí na navrhovaný tag je tento tag doplněn do výrazu v textovém poli.

```
for(let i = 0; i < infix_notation.length; i++) {
    let current_symbol = infix_notation[i]
    if(current_symbol != "TAG") {
        if(current_symbol == "(") {
            stack.push(current_symbol)
        } else if(current_symbol == ")") {
            while(stack.topItem() != "(") {
                postfix.push(stack.pop())
            }
            stack.pop() // deleting left bracket
        } else if (current_operator.precedence > top_stack.operator_precedence) {
            stack.push(current_symbol)
        } else {
            while(current_operator.precedence <= top_stack.operator_precedence) {
                postfix.push(stack.pop())
            }
            stack.push(current_symbol)
        }
    } else if (current_symbol == "TAG") {
        postfix.push(current_symbol)
    }
}

while(stack is not empty){
    postfix.push(stack.pop())
}
```

Výpis 5.1: Pseudo algoritmus, který je v aplikaci použit pro převod infixové notace na postfixovou.

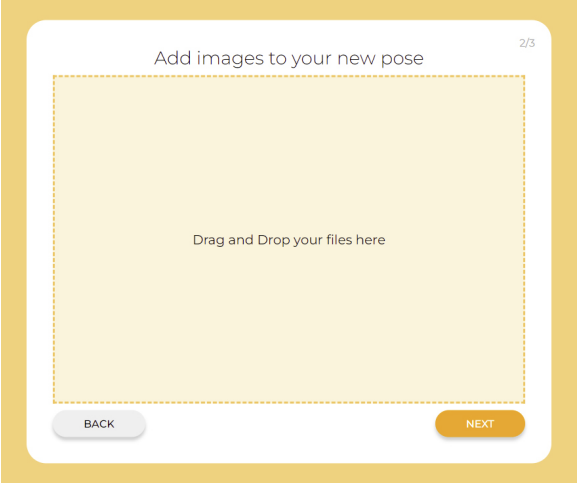
5.7 Vložení nové pozice do databáze

Přihlášení uživatelé mohou do databáze vkládat nové jógové pozice. Pokud uživatel nemůže svou pozici vyhledat pomocí metody livesearch, je mu nabídnuta možnost ji přidat. Po kliknutí na tlačítko `Add new pose` je uživatel přesměrován na stránku, kde může vložit informace o jeho pozici pomocí jednoduchého formuláře. Tento formulář je rozdělen na tři části. Část první, kde se zadávají textové informace jako název a popisek. V druhé části může uživatel přidat několik fotografií. Ve třetí části uživatel specifikuje obtížnost a tagy, které jsou připsány k této nové pozici.

První část je tvořena dvěma elementy `<input>` (anglický a sanscript název) a jedním elementem `<textarea>`, který je vhodnější pro zadávání delších textů. Pokud uživatel nevy-

plní, některý z údajů není mu dovolen postup do další části. Při této skutečnosti je uživatel informován červenou notifikací v horní části aplikace.

Hlavní prvky druhé části (Podoba druhé části je na Obrázku 5.5) jsou dva elementy `<div>` a element `<input>`, který příjímá obrázkové soubory. Obrázky se dají vložit dvěma způsoby: kliknutím na element pro vložení, nebo přetažením obrázku nad tento element (drag'n'drop). Elementy `<div>` jsou umístěny jeden přes druhý. Celá stránka, kde se přidávají pozice, musí mít připsané tři naslouchače: `dragover`, `dragleave`, `drop`. Tyto naslouchače zabraňují nechtěnému implicitnímu chování prohlížeče např. otevření obrázku při přetažení na nesprávné místo. Element `<div>`, který je výše (hloubkově), má připsán naslouchač události `drop`, který spustí připsanou funkci při dokončení přetažení souboru nad tento prvek. Funkce vytvoří kopii přetaženého obrázku, kterou umístí na stránku, aby uživatel viděl náhled svého obrázku, jak bude zobrazen na stránce. Zároveň je tento obrázek uložen do proměnné `form_data`, což je objekt JavaScriptové třídy `FormData`, která slouží pro uchování a zaslání různých typů dat zároveň (textové řetězce, obrázky, JSON soubory). Druhý element `<div>`, který je hloubkově níže, pouze zobrazuje barevné pozadí a text.



Obrázek 5.5: Druhá část formuláře pro přidání pozic. Uživatelé buď mohou kliknout na oranžový čtverec nebo do něj mohou přetáhnout požadovanou fotografii.

Ve třetí části se nachází element `<input>` typu slider, který uživateli umožňuje přidat k pozici obtížnost. A posledním elementem formuláře je textové pole, pomocí kterého se přidávají tagy k pozici. Toto textové pole má implementován i pomocný našeptávač, který zobrazuje tagy již uložené v databázi. Uživatel ale může bez problému vytvořit nový tag. Po potvrzení celého formuláře jsou všechny informace zaslány backendu aplikace a uloženy do databáze. Při úspěšném uložení je uživatel přesměrován zpět na domovskou stránku.

5.8 Tvorba a ukládání jógových sestav

Přihlášení uživatelé si mohou vlastní jógové sestavy složit z pozic, které jsou uloženy v databázi. Do systému tvorby sestav se mohou dostat několika způsoby:

- V záložce `My routines`, kterou má každý uživatel na svém účtu,
- přes detail pozice lze vytvořit novou sestavu s pozicí, která je právě zobrazena

- nebo se dá přes detail pozice přidat do již vytvořené sestavy.

Tvorba každé sestavy je rozdělena do dvou částí. V první části uživatel specifikuje, které pozice se budou cvičit, jak dlouho se v každé z nich přetrvává a pořadí ve kterém pozice budou za sebou následovat. V druhé části uživatelé musí zadat popis zadané sestavy: Jméno, obtížnost, textový popis a obrázek.

Na začátku tvorby sestav je na stránce viditelná pouze prázdná karta se symbolem plus uprostřed (viz Obrázek 5.6). Po kliknutí na tuto kartu je uživateli otevřena nabídka všech pozic v databázi, ve které je možné vyhledávat pomocí metody livesearch (viz Obrázek 5.6a). Při přejetí kurzorem se nad každou pozicí zobrazí symbol plus, který uživatelům napovídá, že po kliknutí bude tato pozice přidána do tvořené sestavy. To je implementováno tak, že se po kliknutí na kartu zkopíruje její vnitřní struktura (obrázek, jméno atd.). Tato vnitřní struktura je poté vložena do nového elementu `<div>` s jinou třídou. Třídu je potřeba napsat jinou, aby již vložené karty do sestavy nereagovaly na událost kliknutí. Navíc je k vnitřní struktuře přidán element `<input type="number">` pro vložení času setrvání v dané pozici a symbol křížku pro její odstranění. Navíc každý tento element (karta vložená do sestavy) musí mít připsané nové naslouchače událostí, které budou uživateli umožňovat pozice přesouvat: události dragstart a dragend. Tyto události pouze přidávají třídu `dragging` při začátku a odebírají ji při konci přetažení. Tato třída pouze trochu zvýší průhlednost a vypadá tedy, jako pouhé naznačení umístění přetahované pozice. Aby mohlo přetahování pozic fungovat, musí být ještě aplikován naslouchač událostí dragover[4] na celý element, kam se vkládají všechny pozice sestavy. Při této události se vždy vypočítá vzdálenost přetahované pozice od všech ostatních v sestavě. Všechny se porovnají a vybere se karta, ke které je přetahovaná pozice nejblíž. Za tuto kartu se vždy přemístí přetahovaná pozice a při upuštění pozice je vymazána třída `dragging`, tudíž pozice již nebude mít zvýšenou průhlednost a je umístěna na požadované místo. Při dokončení tvorby sestavy se uživatel dostane do další části stisknutím tlačítka v pravém dolním rohu (viz obrázek 5.6c). Při stisknutí tohoto tlačítka je kontrolováno, jestli jsou vyplněné všechny časy u všech vložených pozic, pokud nejsou, uživatel je informován červenou notifikací.

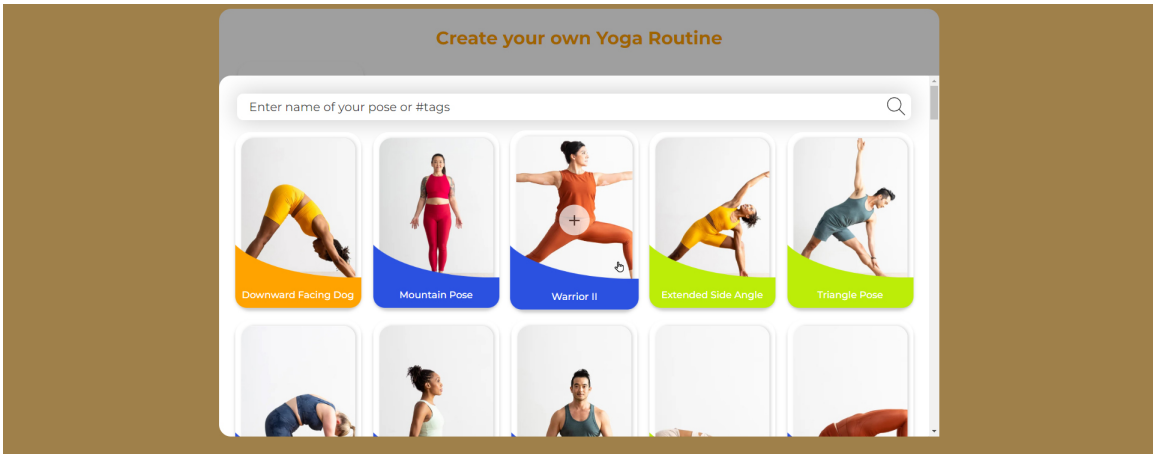
V další části je uživatel povinen zadat jméno vytvořené sestavy, obtížnost a textový popis. Musejí také přidat jednu fotografii, která bude zobrazována v přehledu sestav. Po stisknutí na tlačítko `Confirm` v pravém dolním rohu je opět zkontrolováno, jestli jsou všechny potřebné informace uživatelem vyplněny, pokud nejsou je uživatel na každou z nich upozorněn. Jinak se zasílá asynchronní požadavek na server, který nese informace o vytvořené sestavě (všechny pozice i textové informace). Co je o každé sestavě uloženo je popsáno v kapitole 5.1.

5.9 Asistent cvičení jógových sestav

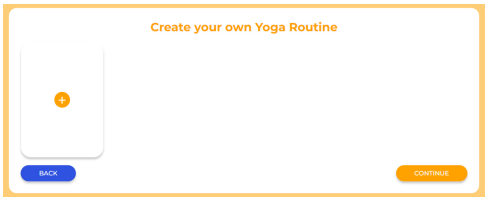
Uživatelé mohou pomocí asistenta cvičení jógových sestav odcvičit své vytvořené sestavy nebo i sestavy ostatních uživatelů (více o sdílení sestav v kapitole 5.10). Tento systém má za úkol přehledně uživateli ukázat, jak dlouho by měl v dané pozici setrvat a také zobrazit následující pozici v sestavě.

Systém je tvořen 4 prvky (viz obrázek 5.7): karta aktuální pozice, časovač, který ukazuje, kolik zbývá sekund do přechodu do další pozice. Dále je zde ovládací prvek, pomocí kterého

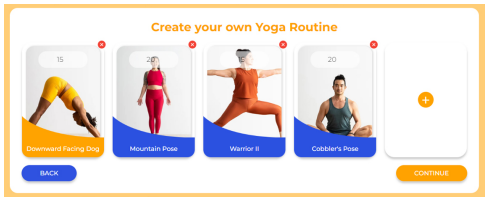
[4] Událost dragover je vyvolána každých několik set milisekund při přetahování nějakého prvku přes validní element upuštění – to může být například element `<div>` nebo `<section>` [34].



(a) Při tvorbě sestav je použita tato nabídka pozic, ve které uživatel může využívat vyhledávání metodou livesearch. Po kliknutí na libovolnou pozici se pozice přidá do právě tvořené sestavy.



(b) Při otevření tvorby nové sestavy je na stránce pouze jedna karta se symbolem plus, přes kterou se uživatel dostane na nabídku pozic (Obrázek 5.6a)



(c) Dokončená sestava v systému tvorby sestav.

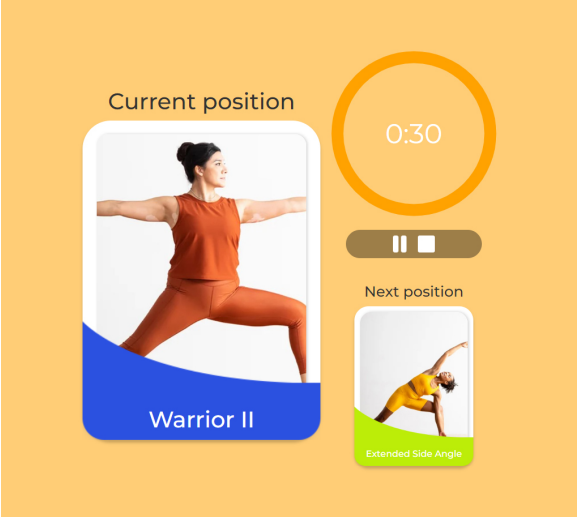
Obrázek 5.6: Systém pro tvorbu jógových sestav. Uživatel si pomocí nabídky (obrázek 5.6a) může vybrat z pozic, které jsou uloženy v databázi. Při kliknutí na pozici v nabídce se pozice přidá před kartu se symbolem plus (obrázek 5.6b). V horní části každé karty je pole, kam uživatel musí zadat, jak dlouho se daná pozice musí držet.

uživatel může cvičení sestavy pozastavit, popřípadě úplně zrušit. Posledním prvkem je karta následující pozice.

Do asistenta cvičení se uživatel dostane přes tlačítko `Start Workout`, které se nachází v přehledu sestav u každé z nich. Po kliknutí je uživatel přesměrován na stránku, kde může sestavu odstartovat. Při načtení stránky jsou načteny i informace o všech pozicích, které jsou v dané sestavě. Viditelné jsou pouze první dvě, ale další pozice jsou skryty a v průběhu cvičení sestavy se tyto pozice přesunují do viditelných oblastí. Po kliknutí na tlačítko start se spustí pěti sekundový odpočet. Ten je vytvořen pomocí časové osy (`gsap.timeline()`), která jednotlivé animace navazuje na sebe. Vždy se plynule zvětší číslo, následně se do ztracena zmenší, změní se na číslo o jedno menší a znovu se zvětší. Při posledních 4 sekundách zazní pípnutí, které signalizují start.

Časovač[5] vždy ukazuje zbývající čas setrvání v aktuální pozici. Tento čas se při dokončení jedné pozice restartuje a začne znovu odpočítávat čas pozice další. Tento údaj má každá karta pozice připsaný v atributu `data-time`. Pomocí JavaScriptové funkce `setInterval` je

[5]Časovač byl vytvořen a dále upraven podle návodu z: `https://css-tricks.com/how-to-create-an-animated-countdown-timer-with-html-css-and-javascript/`



Obrázek 5.7: Asistent cvičení jógových sestav je rozdělen na čtyři části – aktuální pozice, časovač, ovládací panel a následující pozice.

zaručeno volání funkce, která je zodpovědná za chod časovače. Tato funkce každou sekundu dekrementuje čas o jedničku a zároveň nastavuje okolní kruh, tak aby odrážel skutečnou část odcvičeného času z dané pozice. Při posledních 4 sekundách opět zazní pípnutí, které uživatelům pomáhá vnímat zbývající čas, aniž by se po celou dobu museli dívat na grafický časovač.

Když se časovač dostane na hodnotu 0, je zapotřebí zaměnit aktuální cvičenou pozici za následující a zároveň nastavit novou následující pozici. Při této skutečnosti je přepsána třída další pozice z `next-active-pose` na `active-pose` a zároveň je celá přesunuta do elementu `<div>`, který je určen pro aktuální pozici. Po tomto je potřeba doplnit následující pozici. Ta je vyjmuta z elementu `<div>`, který má připsanou třídu `positionsNext`, kde jsou připravené nevidetelné karty s dalšími pozicemi. Při dokončení celé sestavy je uživateli zobrazena zpráva o úspěšném dokončení sestavy.

5.10 Sdílení sestav a queries

Uživatelé mají také možnost mezi sebou jednotlivé sestavy a queries sdílet. V přehledu sestav má každá tlačítko se symbolem sdílení, které danou sestavu zpřístupní ostatním uživatelům. Každý má tedy možnost se rozhodnout, jestli své sestavy bude sdílet se všemi uživateli nebo nikoliv. Do nabídky sdílených sestav nebo queries se uživatel může dostat stisknutím na tlačítko `Explore shared routines` nebo `Explore shared queries`.

Uživateli se po stisknutí na tlačítko zobrazení sdílených sestav zobrazí seznam všech sestav, které ostatní uživatelé označili jako sdílené. Zároveň u každé je tlačítko, kterým uživatel může danou pozici zkopírovat a uložit si ji tak do svého seznamu sestav. Při stisknutí na toto tlačítko je vyslán asynchronní požadavek na server, který daného uživatele přidá do pole uživatelů kopírujících danou sestavu. Tento přístup má výhodu v tom, že uživatel vždy má aktuální kopii dané sestavy. Tuto sestavu nemůže nijak měnit. Pokud se ovšem majitel (uživatel co sestavu vytvořil) rozhodne v ní udělat nějaké změny, tyto změny se promítnou i do všech zkopírovaných sestav. Nevýhodou tohoto přístupu je to, že pokud uživatel, co sestavu vytvořil, danou sestavu smaže, odstraní se tato sestava i uživatelům, které ji zko-

Obrázek 5.8: (Vlevo) Nabídka sdílených sestav, které si uživatelé mohou zkopírovat do svého seznamu sestav a následně je odcvičit. (Vpravo) Nabídka sdílených queries, které si uživatel může prvně prohlédnout a až poté uložit do svého seznamu.

pírovali. Při úspěšném přidání do pole kopírujících uživatelů je uživatel informován, že vše proběhlo úspěšně notifikací a zároveň je pozice již zkopírována do jeho seznamu sestav. Zkopírované sestavy jsou vždy pod uživatelem vytvořenými sestavami. Uživatel nemá přístup k úpravám této sestavy, ale může ji pomocí asistenta cvičení jógových pozic odcvičit.

Sdílené queries používají podobný přístup jako sestavy. Při kliknutí na zkopírování určité query je vyslán asynchronní požadavek, který uživatele přidá do pole kopírujících uživatelů k dané query a uživatel ji bude moci nyní najít pod svými uloženými queries. Uživatel si může prohlížet výsledky dané query, ale nemůže v ní dělat jakékoliv změny.

5.11 Administrativní stránky

Pro administrátory jsou v aplikaci implementovány administrativní stránky, kde mají přehled o celé databázi uživatelů, pozic, sestav a uložených queries. Mimojiné zde mohou také upravovat všechny tagy (např. mazat tagy od pozic). Na hlavní stránce jsou zobrazeny počty jednotlivých prvků v databázi. V budoucnu by mohlo být doimplementováno monitorování aktivity uživatelů – kdy se uživatele nejvíce přihlašují, kdy nejvíce cvičí atd.

Na administrativní stránce uživatelů je možné vyhledávat podle atributu e-mail metodou livesearch. U každého uživatele lze provézt dvě nebo tři akce. Administrátoři mohou připisovat administrátorská práva jiným uživatelům. Po kliknutí na tlačítko se symbolem šipky nahoru je ještě vyžadováno potvrzení a následně je pomocí asynchronního požadavku požadovaný uživatel povýšen na roli administrátora. Podobně fungují i tlačítka pro blokování nebo úplné smazání uživatele. Blokovaní uživatelé se nemohou přihlásit do svého účtu, který ovšem je pořád uložen v databázi. Při smazání uživatele se smaže uživatel jako takový, ovšem pozice, sestavy, queries, které vytvořil v systému zůstávají, pouze se u nich neprezentuje autorské jméno.

I na stránce se správou pozic lze využívat metodu livesearch. Zde mohou administrátoři každou pozici upravit nebo smazat. Úprava pozice používá stejný formulář, který se využívá při tvoření pozice (Kapitola 5.7), akorát zde jsou v jednotlivých polích doplněny již uložené informace z databáze. Při uložení do databáze se textové informace jednoduše přepíší. U obrázků a tagů se musí zkoumat, které obrázky/tagy byly odstraněny a které zase naopak vloženy, aby v databázi nezůstávala data, která se nepoužívají (v případě obrázků). Administrátoři na této stránce mají také možnost jednotlivé pozice potvrzovat – tento záznam v databázi sice je, ale v nynější verzi aplikace není žádný rozdíl mezi potvrzenou a

nepotvrzenou pozicí. Tato funkcionalita ale zatím v aplikaci zůstává, kdyby se ukázalo, že je lepší přístup vložené pozice od uživatelů nejdříve poslat ke schválení administrátorům a následně je teprve publikovat všem ostatním uživatelům webové aplikace.

Administrativa sestav má pouze jedinou možnost a to sestavy vymazat, jelikož uživatelé si správu sestav provádějí sami (editace, sdílení). Administrátoři mají možnost pouze jednotlivé sestavy vymazat, pokud by se ukázalo, že jsou nějaké sestavy nevhodné nebo jsou zde pouze pro zaplnění databáze zbytečnými záznamy.

Administrátoři mají také možnost jednoduše vymazat jednotlivé pozice od připsaných tagů. Toto je implementováno podobným způsobem, jako připisování tagů na hlavní stránce (Kapitola 5.5). Při mazání tagů je možné zvolit pozice, které se od daného tagu mají vymazat a následně tento výběr potvrdit. Tím se v databázi odstraní označené pozice z pole pozic, které má daný tag u sebe uložen. Na této stránce je také možnost kompletně vymazat tag z databáze. Při smazání se tag tedy již nezobrazuje u žádné z pozic, které měl připsány.

Kapitola 6

Testování

Testování bylo inspirováno knihou Steva Kruga [25]. Ten ve své knize představuje tzv. do-it-yourself testování. Toto testování je zaměřeno na použitelnost uživatelského rozhraní (dále jen UI). Testování použitelnosti většinou zahrnuje pozorování lidí, kteří používají daný produkt s cílem odhalit, jestli se uživatelům dané rozhraní používá jednoduše nebo právě odhalit cesty, jak uživatelské rozhraní změnit, aby bylo pro uživatele co nejintuitivnější. Rozlišují se dva typy tohoto testování – kvantitativní a kvalitativní. U kvantitativního testování je cílem něco prokázat (např. je tato verze UI lepší než předešlá? nebo je toto UI lepší než ostatní?). Kvantitativní testování zahrnuje různá měření: procento úspěšnosti (kolik lidí dokončilo zadané úkoly) nebo časování úkolů (jak dlouho uživatelům trvá úkoly vyřešit). Během tohoto testování se většinou vyhýbá kontaktu uživatele s návrhářem UI – aby se ověřila intuitivnost. Uživatel sedí sám v místnosti se zadanými úkoly, které sám musí vyřešit, během čehož je nahrávána obrazovka či je uživatel sledován zpovzdálí.

Do-it-yourself testování se řadí do kvalitativního testování. Nemá za cíl prokázat, že je nějaká verze lepší, ale spíše má za úkol odhalit, jak dané UI zlepšit. To znamená, že toto testování je mnohem více neformální a může například zahrnovat i změnu testovacího scénáře přímo během testování, což při kvantitativním testováním je nemožné, protože všichni uživatelé, kteří produkt testují, by neměli stejné podmínky. Kvalitativní testování také potřebuje mnohem méně uživatelů právě kvůli tomu, že se k testování dá přistupovat způsobem: jeden uživatel otestuje UI a následně se UI může změnit. Většinou se během do-it-yourself testování neshromažďují žádná data, ale spíše se sledují myšlenkové pochody uživatele. Jsou tři důvody proč toto testování funguje velmi dobře:

- Všechny webové stránky mají problémy stejně jako existuje axiom, že v každém programu se nachází chyba, tak v použitelnosti uživatelského rozhraní tomu není jinak.
- Většinou jsou závažné problémy velmi jednoduché k nalezení. Pro člověka, který UI implementuje, jsou ovšem velmi složité pro nalezení, protože velmi dobře UI zná a ví, co v něm má dělat. Uživatelé, kteří UI vidí poprvé tuto zkušenost nemají, což se pro nalezení těchto chyb velice hodí.
- Sledování uživatelů má za cíl také zlepšení všeobecné dovednosti navrhovat použitelnější rozhraní. Časem má návrhář rozhraní větší zkušenosti a dokáže odhadnout, které prvky na jaké místo v UI dát, aby byli pro uživatele na co nejintuitivnějším místě.

Steve Krug doporučuje pro pravidelné testování 3 uživatele. To bohužel nebylo možné a během implementace, jsem měl k dispozici pouze uživatele jednoho, který mi pomáhal

s tím, abych si ujasnil, kde na stránce má co být. Nebylo to tedy testování v pravém slova smyslu, spíše byla implementována nějaká část uživatelského rozhraní a poté byla ukázána uživateli, který ji popsal a řekl, co si o ní myslí. To zahrnulo například, co si představoval pod různými tlačítky (kam se dostane? co se stane, když na tlačítko klikne?). Tento přístup se mi velice dobře osvědčil a vedl k signifikantnímu zkrácení závěrečného testování.

6.1 Testovací scénář

Po implementaci celé webové aplikace byl sepsán testovací scénář, který zahrnoval ty nejdůležitější úkoly, které by každý uživatel měl na stránce zvládnout. Během testování byl tedy kladen důraz na:

- Registrační a přihlašovací systém,
- vyhledávání pomocí live-search,
- vyhledávání pomocí jazyka query,
- tvorba a cvičení sestav,
- vytváření a připisování nových tagů k pozicím.

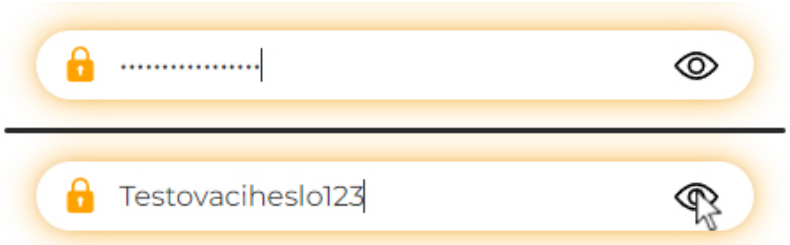
Pro každou z výše uvedených částí bylo tedy vymyšleno 3 až 5 testů, které měly za úkol zkoumat intuitivnost UI. Před testováním byl uživatel jednoduše seznámen se systémem queries a obeznámen s možnostmi tohoto vyhledávání. Nebylo mu řečeno, které tagy má využívat, pouze to, že tagy může mezi sebou kombinovat pomocí klíčových slov: AND, OR a NOT. Během samotného testování byl kladen důraz (opět podle knihy Steva Kruga [25]) na tzv. rozmluvení uživatele. Testování tedy probíhalo v přátelské atmosféře a uživatel prakticky nahlas přemýšlel – tzn. popisoval, co právě dělá, co se chystá udělat, co od stránky a tlačítek očekává. Bylo zapnuto nahrávání obrazovky, které snímalo pohyby myši po webové stránce a zároveň zaznamenávalo zvuk. Už jen během prvního testování bylo objeveno přibližně 15 chyb a sám uživatel podal návrh na zlepšení použitelnosti třemi novými návrhy. Po zhlédnutí celého záznamu z testování byli přidány ještě navíc dvě nové funkcionality.

6.2 Výsledky testů

Díky testování se přišlo na spoustu chyb, které se ve webové aplikaci během implementace přehlédly. Mezi nejjednodušší chyby, které byly opraveny prakticky hned, bych zařadil například: nevymazání testovacího řetězce z vyhledávacího pole queries (to zde bylo, aby při implementaci nebylo nutné psát po každém obnovení stránky výraz znovu), špatný font v některých prvcích na webové stránce nebo například při zkopírování sestavy/queries ze sdílených, by se měla zavřít nabídka sdílených sestav/queries.

Dále po testování byl přidán nový typ notifikace. Před testováním zde byli pouze dva druhy: úspěch (zelená) a neúspěch (červená). Je ale lepší uživatele o jeho akcích informovat častěji a proto byla přidána ještě informační notifikace (modrá). Při vytváření sestavy uživatel nevěděl, jestli se daná pozice do sestavy opravdu přidala a musel nabídku pozic zavřít a zkontrolovat přítomnost dané pozice. Proto byla přidána informační notifikace právě sem.

Na webové stránce se také hojně využívaly SVG ikony, které v sobě zahrnovaly binární data z JPG souborů. To vzniklo při tvorbě těchto ikon v programu Adobe Photoshop, který



Obrázek 6.1: Při přejetí kurzorem myši přes ikonu oka se uživateli zobrazí heslo v textové podobě.

uložení ve vektorovém formátu SVG přímo nabízí. Po uložení ve formátu SVG ale některé soubory obsahovaly binární data JPG obrázků, které například prohlížeč Firefox zobrazuje správně, ale prohlížeč Chrome tato binární data vůbec nezobrazí.

Další chyba nastala po přidání nového tagu k nějaké pozici. Našeptávač tagů v queries je naplněn daty při načtení stránky. Pokud uživatel přidává nový tag k nějakým pozicím, je zaslán asynchronní požadavek, což znamená, že uživatel by musel stránku obnovit aby došlo k opětovnému naplnění našeptávače daty. Tato chyba byla opravena tím, že po přijetí odpovědi na asynchronní požadavek byl nový tag přidán do nabídky našeptávače.

Při úkolu, který se týkal vytvoření nové sestavy, uživatel tuto možnost nemohl najít. Intuitivně přešel do přehledu svých sestav a zde očekával, že tato možnost bude nabídnuta. Další způsob, který uživatel sám zkusil, byl: vyhledání první pozice, která v plánované sestavě má být a zde uživatel hledal možnost, jak tuto pozici přidat do sestavy. Proto v poslední verzi webové aplikace byly přidány ke každé pozici dvě tlačítka. Jedno uživateli umožňuje vytvořit novou sestavu s právě prohlíženou pozicí. Druhé tlačítko uživateli umožňuje přidat novou pozici k již vytvořené sestavě. Po stisknutí na toto tlačítko se zobrazí nabídka všech uložených sestav a po kliknutí na jednu z nich se otevře stránka s editací dané sestavy, kam je přidána právě zobrazovaná pozice.

Při zhlédnutí záznamu obrazovky jsem si všiml dvou mírných nedostatků. Prvním nedostatek nastal hned při registraci uživatele. Při vytváření hesla uživatel dvakrát vkládané heslo smazal, pravděpodobně kvůli tomu, že si byl nejistý, jestli neudělal nějaký překlep. Proto byla do tohoto pole přidána nová funkcionalita. Uživatel nyní na ikonu oka (viz Obrázek 6.1) může najet myší a tím si zobrazí právě zadávané heslo (změní se typ tohoto pole na text z typu password). Toto zobrazení trvá pouze tak dlouho, jak má uživatel dlouho kurzor myši nad ikonou oka.

Druhé vylepšení, které bylo implementováno díky záznamu obrazovky, bylo jednodušší vyhledávání pomocí queries. Na videozáznamu lze vidět, že při zadávání první query uživatel intuitivně tagy zadával bez znaku **#**. Poté na tuto skutečnost byl upozorněn a dále tento znak na počátek každého tagu sám přidával. Kvůli této skutečnosti byla přidána funkcionalita, která při potvrzení vyhledávání na začátek každého slova, které není v seznamu klíčových přidá znak **#** automaticky.

Při druhém testování byly odhaleny chyby při registraci, kdy se při odeslání údajů nekontrolovalo, jestli daný e-mail již v databázi neexistuje, také se nekontroloval tvar hesla. To bylo zapříčiněno tím, že při implementaci aplikace tyto části byly zakomentovány kvůli pohodlnosti registrace. Po odkomentování všech kontrol již registrace fungovala jak má. Při tomto testování bylo také přidáno tlačítko nápovědy k používání queries. Po stisknutí na toto tlačítko se uživateli zobrazí jednoduchý manuál s příklady, které uživatele seznámí s psaním queries.

Kapitola 7

Závěr

Hlavním cílem této práce bylo implementovat jednoduché a intuitivní uživatelské rozhraní pro tvorbu, správu a využití databáze sportovních pozic. Výsledkem je aplikace Yoga Poses, která je dostupná díky hostingu Heroku na adrese `https://yoga-poses-bp.herokuapp.com/`. Aplikace je implementována pomocí jazyku Python a jeho microframeworku Flask, který nabízí jednoduchou cestu, jak implementovat aplikaci na straně serveru. Jako databázový systém byla vybrána NoSQL databáze MongoDB, hlavně kvůli podpoře uložení složitějších datových struktur přímo do databáze a také možnosti jednoduše ukládat binární data. Front-end byl naprogramován pomocí jazyku HTML s podporou šablonovacího systému Jinja2. Vzhled jednotlivých prvků byl nejprve navržen pomocí programu Adobe Photoshop a následně implementován pomocí jazyka CSS. Interakce s uživatelem byla implementována v jazyce JavaScript s knihovnami jQuery a GSAP. Aplikace velmi hojně využívá asynchronní požadavky na server, což zpříjemňuje uživateli práci s tímto systémem a celá aplikace tedy působí svižněji.

Webová aplikace Yoga Poses uživateli nabízí dvě metody, pomocí kterých může vyhledávat v databázi jógových pozic. První implementovanou metodou je livesearch. Tato metoda vyhledávání nabízí uživateli výsledky přímo při zadávání vyhledávaného výrazu. Vyhledávání probíhá pomocí regulárních výrazů, tedy tímto vyhledáváním není server nijak zatížen, jelikož vše je počítáno na straně klienta. Pro druhou metodu byly klíčové uložené tagy, které sám uživatel může volně připisovat k pozicím na domovské stránce. Tyto jednoslovné řetězce vždy nějak blíže určují danou pozici, připisují jí buď nějakou vlastnost nebo jinou charakteristiku. Poté byl navržen jednoduchý logický jazyk queries, který nabízí možnost, jak již zmíněné tagy mezi sebou kombinovat v logických výrazech. Uživatel tak jednoduše může najít pozice, které se cvičí ve stoje a zároveň jsou např. vhodné pro cvičení venku. Tento typ vyhledávání si uživatel může uložit do svého účtu, tudíž nemusí vždy danou query vždy psát znovu. Jógové pozice se většinou skládají za sebe a tvoří sestavy. Necvičí se tedy samostatně. Proto byl vytvořen systém, který uživateli umožňuje složit si vlastní sestavu pomocí pozic, které jsou již v databázi. Tuto sestavu si uživatel může uložit a následně, pomocí asistenta cvičení jógových pozic, tuto sestavu odcvičit. Uživatelé mohou mezi sebou také vytvořené queries/sestavy sdílet. Mají možnost si sdílené sestavy ostatních uživatelů zkopírovat přímo do svého účtu a následně je také odcvičit.

Aplikace byla následně otestována metodou do-it-yourself. Tato metoda se ukázala, jako velice užitečná, protože již během prvního testování poskytla nové a velmi dobré poznatky z oblasti použitelnosti uživatelského rozhraní, které byly zapracovány do aktuální verze webové aplikace. Během testování bylo také prokázáno, že metoda livesearch uživateli velice zpříjemní uživatelský zážitek, jelikož vede k mnohem kratší době vyhledávání. Uživatel

totiž většinou ani nedopsal vyhledávaný výraz a již našel výsledek svého hledání. Prvotní domněnkou bylo, že vyhledávání pomocí queries nebude intuitivní vzhledem ke komplikovanému kombinování tagů, používání klíčových slov a dodržování syntaktických pravidel. Testování ovšem potvrdilo, že po seznámení uživatele s logickými spojkami, uživatel dokáže velmi jednoduše tvořit i komplikované výrazy v jazyce queries.

Nejdůležitější změnou v budoucích verzích aplikace bude responzivita celého systému, aby se webová aplikace správně zobrazovala na zařízeních s různými velikostmi obrazovky. Webová aplikace by v budoucnu mohla být doplněna mobilní aplikací, které by spolu navzájem spolupracovaly. Tato mobilní aplikace by se hlavně měla soustředit na jógové sestavy. Pracovala by se stejnou databází jógových pozic, ale sloužila by primárně pro skládání sestav a jejich cvičení. Dále by také celá kostra mohla být použita i pro jiné sporty. Například pro cvičení HIIT kardia by tato kostra sloužila velice dobře, jelikož jsou zde podobné principy jako u jógy, pouze by se zaměnila databáze.

Literatura

[1] Aquino, C. a Gandee, T. *Front-end web development: The Big Nerd Ranch Guide.* 1. vyd. Big Nerd Ranch, 2017. ISBN 978-0134432533.

[2] Banker, e. a. *MongoDB in action.* 2. vyd. Manning, 2016. ISBN 978-1-617-29160-9.

[3] Bc. Pater, J. *Modern Web Application Frameworks.* 2015. [cit. 2022-03-29]. Diplomová práce. Masarykova Univerzita v Brně, Fakulta informačních technologií. Vedoucí práce RNDr. Petr Sojka, P. doc. Dostupné z: https://is.muni.cz/th/uz7ba/Jan_Pater_Modern_Web_Application_Frameworks.pdf.

[4] Beaulieu, A. *Learning SQL: Master SQL fundamentals.* 2. vyd. O'Reilly, 2009. ISBN 978-0-596-52083-0.

[5] Campbell, S. Flask vs Django: What's the Difference Between Flask & Django? *Guru99* [online]. Březen 2022 [cit. 2022-03-29]. Dostupné z: https://www.guru99.com/flask-vs-django.html.

[6] Chaffer, J. a Swedberg, K. *Learning jQuery: Better interaction, design, and web development with simple JavaScript techniques.* 4. vyd. PACKT PUBLISHING, 2013. ISBN 978-1-78216-314-5.

[7] Chodorow, K. *MongoDB: The Definitive Guide.* 2. vyd. O'Reilly, 2013. ISBN 978-1-449-34468-9.

[8] Crockford, D. *JavaScript: The Good Parts.* 1. vyd. O'Reilly, 2008. ISBN 978-0-596-51774-8.

[9] DeGroat, T. J. Everything You Need to Know. *The History of JavaScript* [online]. 2019 [cit. 2022-04-02]. Dostupné z: https://www.springboard.com/blog/data-science/history-of-javascript/.

[10] Dekel, E. a Sahni, S. Parallel Generation of Postfix and Tree Forms. *ACM Trans. Program. Lang. Syst.* 1. vyd. New York, NY, USA: Association for Computing Machinery. jul 1983, sv. 5, č. 3, s. 300–317. DOI: 10.1145/2166.357211. ISSN 0164-0925. Dostupné z: https://doi.org/10.1145/2166.357211.

[11] Demirov, I. *Learn JavaScript visually: Accelerated learning method that uses science and creativity to teach the right brain non-coders.* 3. vyd. Demirov, Ivelin, 2016. ISBN 978-1495233005.

[12] Duckett, J. *HTML&CSS: design and build websites.* 1. vyd. Wiley, 2011. ISBN 978-1-118-00818-8.

[13] Ducket, J. *Javascript and jQuery*. 1. vyd. Wiley, 2014. ISBN 978-1-118-53164-8.

[14] Farrell, J. A. *Anatomy of a compiler*. Srpen 1995. Dostupné z: http://www.cs.man.ac.uk/~pjj/farrell/comp3.html.

[15] Fowler, M. a Sadalage, P. J. *NoSQL distilled*. 1. vyd. Addison-Wesley, 2012. ISBN 978-0-321-82662-6.

[16] Ghimire, D. *Comparative study on Python web frameworks: Flaks and Django*. 2020. [cit. 2022-03-29]. 37 s., 0 s. příl. Bakalářská práce. Metropolia University of Applied Sciences. Vedoucí práce Salo, K. Dostupné z: https://www.theseus.fi/bitstream/handle/10024/339796/Ghimire_Devndra.pdf.

[17] Grinberg, M. *Flask Web Development: Developing web applications with python*. 1. vyd. O'Reilly, 2014. ISBN 978-1-449-37262-0.

[18] Hayes, A. *HyperText Markup Language – HTML* [online]. 2020 [cit. 2022-03-30]. Dostupné z: https://www.investopedia.com/terms/h/html.asp.

[19] Janovský, D. *Jak psát web: o tvorbě internetových stránek* [online]. 1999. 2022 [cit. 2022-03-30]. Dostupné z: https://www.jakpsatweb.cz/.

[20] Jinja. *Jinja Documentation: (3. 1. x)* [online]. 2007 [cit. 2022-04-05]. Dostupné z: https://jinja.palletsprojects.com/en/3.1.x/.

[21] Johnson, B. *How CSS works: Understanding th ecascade* [online]. 2018 [cit. 2022-03-31]. Dostupné z: https://blog.logrocket.com/how-css-works-understanding-the-cascade-d181cd89a4d8/.

[22] Julien. *Top 7 Best Yoga Sequence Builder Apps + 7 Tips for Planning Yoga Classes* [online]. Únor 2022 [cit. 2022-04-12]. Dostupné z: https://www.theyoganomads.com/yoga-sequence-builder/.

[23] Kouraklis, J. *MVVM as Design Pattern*. 2. vyd. říjen 2016. ISBN 978-1-4842-2213-3.

[24] Krasner, G. E. a Pope, S. T. A cookbook for using the model-view controller user interface paradigm in Smalltalk-80. *Journal of Object Oriented Programming*. 1. vyd. Denville, NJ, USA: SIGS Publications. 1988, č. 3, s. 26–49. ISSN 0896-8438.

[25] Krug, S. *Rocket Surgery Made Easy: The Do-It-Yourself guide to Finding and Fixing Usability Problems*. 1. vyd. New Riders, 2010. ISBN 978-0-321-65729-9.

[26] Krug, S. *DON'T MAKE ME THINK: A Common Sense Approach to Web and Mobile Usability*. 1. vyd. New Riders, 2014. ISBN 978-0-321-96551-6.

[27] Legierski, B. Frontend vs Backend. *Evertop* [online]. Březen 2021 [cit. 2022-03-29]. Dostupné z: https://www.evertop.pl/wp-content/uploads/2021/03/Front_backend_Obszar-roboczy-1-kopia-26-1024x654.pngt.

[28] Lundberg, A. Color meanings and the art of using color symbolism. *99designs* [online]. 2019 [cit. 2022-03-27]. Dostupné z: https://99designs.com/blog/tips/color-meanings/.

[29] MediaGuru. WEB 2.0. *Mediální slovník* [online]. 2022 [cit. 2022-04-03]. Dostupné z: https://www.mediaguru.cz/slovnik-a-mediatypy/slovnik/klicova-slova/web-2-0/.

[30] Meier, A. a Kaufmann, M. *SQL & NoSQL databases models,languages, consistency options and architectures for Big Data Management*. 1. vyd. Springer Fachmedien Wiesbaden GmbH, 2019. ISBN 978-3-658-24548-1.

[31] Meyer, E. A. *Cascading style sheets: The definitive guide*. 1. vyd. Safari Tech Books Online, 2000. ISBN 1-56592-622-6.

[32] MongoDB. *JSON and BSON* [online]. 2021 [cit. 2022-03-30]. Dostupné z: https://www.mongodb.com/json-and-bson.

[33] MongoDB, I. *MDN Web Docs* [online]. 2021 [cit. 2022-04-10]. Dostupné z: https://www.mongodb.com/docs/.

[34] Mozilla, F. *MDN Web Docs* [online]. 2022 [cit. 2022-04-10]. Dostupné z: https://developer.mozilla.org/en-US/.

[35] OpenJS, F. *JQuery API documentation* [online]. 2022 [cit. 2022-04-02]. Dostupné z: https://api.jquery.com/.

[36] Panchal, C. *What are the Advantages of Python Framework?* [online]. Duben 2020 [cit. 2022-03-29]. Dostupné z: https://theappideas.com/what-are-the-advantages-of-python-framework/.

[37] Python, S. F. FAQ. *Python 3.8.13 documentation* [online]. Březen 2022 [cit. 2022-03-27]. Dostupné z: https://docs.python.org/3/faq/general.html.

[38] Sanders, M. The Most Popular Python Web Frameworks in 2021. *Scout APM* [online]. Červenec 2020 [cit. 2022-03-29]. Dostupné z: https://scoutapm.com/blog/the-most-popular-python-web-frameworks-in-2020.

[39] Shovic, J. a Simpson, A. *Python all-in-one*. 1. vyd. 111 River Street, Hoboken, NJ 07030-5774: John Wiley & Sons, Inc, 2019. ISBN 978-1-617-29160-9.

[40] Singh, V. Best Python Frameworks. *Hackr.io* [online]. Únor 2022 [cit. 2022-03-29]. Dostupné z: https://hackr.io/blog/python-frameworks.

[41] StatisticsTimes.com. Top Computer Languages. *Statistics times* [online]. Prosinec 2021 [cit. 2022-03-27]. Dostupné z: https://statisticstimes.com/tech/top-computer-languages.php.

[42] Syromiatnikov, A. a Weyns, D. A Journey Through the Land of Model-View-* Design Patterns. In: Weyns, D., ed. *Proceedings - Working IEEE/IFIP Conference on Software Architecture 2014, WICSA 2014*. Duben 2014, s. 21–30. DOI: 10.1109/WICSA.2014.13. ISBN 978-1-4799-3412-6.

[43] Tidwell, J. *Designing Interfaces: Patterns for Effective Interaction Design*. 2. vyd. O'Reilly, 2011. ISBN 978-1-449-37970-4.

[44] W3Schools. *W3Schools Online Web Tutorials* [online]. 2022 [cit. 2022-04-01]. Dostupné z: https://www.w3schools.com/.

[45] Woz, U. *What is syntax in Computer Programming?* [online]. 2020 [cit. 2022-04-09]. Dostupné z: https://woz-u.com/blog/what-is-syntax-in-computer-programming/.

[46] Yogapedia. Sequencing. *Yoga Terms tagged with Yoga* [online]. Srpen 2017 [cit. 2022-03-24]. Dostupné z: https://www.yogapedia.com/definition/9749/sequencing.

Příloha A

Obsah přiloženého paměťového média

Paměťové médium obsahuje následující složky/soubory:

ZdrojoveSoubory Tato složka obsahuje nejen zdrojové soubory webové aplikace, ale také všechny, které byly při vývoji použity např. pro naplnění databáze daty nebo pro získání dat z webové stránky Yoga Journal. Složka také obsahuje soubor **README**, ve kterém je návod na zprovoznění databáze a samotné aplikace na operačním systému Windows nebo Ubuntu.

TexProject.zip Zdrojové soubory ve formátu LaTeX použité pro generaci bakalářské práce

xkriva29_bp.pdf Bakalářská práce ve formátu pdf

DemonstracniVideo.mp4 Demonstrační video

Youtube.txt Odkaz na demonstrační video na Youtube

Poster.pdf Informační plakát o vytvořené aplikaci