Webová aplikace pro vědecké pracovníky a podnikatele v oblasti biotechnológií v regionu CEE

Mgr. Barbara Klimeková

Bakalářská práce
2023

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Mgr. Barbara Klimeková**
Osobní číslo:	**A20066**
Studijní program:	**B0613A140020 Softwarové inženýrství**
Forma studia:	**Kombinovaná**
Téma práce:	**Webová aplikace pro vědecké pracovníky a podnikatele v oblasti biotechnologií v regionu CEE**
Téma práce anglicky:	**A Web Application for Biotech Researchers and Entrepreneurs in the CEE Region**

Zásady pro vypracování

1. Prozkoumejte současný stav v řešené oblasti a definujte stávající informační a datové zdroje, které budete využívat.
2. Definujte případy užití pro potřeby dané webové aplikace vyplývající z průzkumu trhu a specifikujte funkční a nefunkční požadavky na prototyp aplikace.
3. Popište některé z populárních frameworků pro implementaci responzivních web aplikací.
4. Vyberte frameworky pro implementaci backendové struktury aplikace a pro responzivní rozhraní aplikace pro vědecké pracovníky a podnikatele v oblasti biotechnologií v regionu CEE.
5. Implementujte funkce pokrývající definované požadavky ve vybraném frameworku. V rámci praktické části popište architekturu prototypu webové aplikace a dále také implementačně zajímavé části. Věnujte také pozornost zabezpečení aplikace.
6. Na prototypové aplikaci vyhodnoťte budoucí možnosti rozšiřování o další funkcionality.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. CANZIBA, Elis. Hands-on Ux design for developers: Design, prototype, and implement compelling user experiences from scratch. Birmingham: Packt Publishing, 2018. ISBN 978-1788-624299.
2. HINKULA, Juha. Hands-on Full Stack Development with Spring Boot 2 and React: Build Modern and Scalable Full Stack Applications Using Spring Framework 5 and React with Hooks. Birmingham: Packt Publishing Ltd., 2019. ISBN 978-1-83882-236-1.
3. MICHÁLEK, Martin. CSS: moderní layout. Praha: Martin Michálek – Vzhůru dolů, 2022. ISBN 978-80-88253-07-5.
4. Getting started. React – A JavaScript library for building user interfaces Documentation [online]. Meta Platforms, Inc. 2022 [cit. 2022-11-29]. Dostupné z: https://reactjs.org/docs/getting-started.html
5. Get started with Netlify – A Netflify Documentation [online]. Netlify, 2022 [cit. 2022-11-29]. Dostupné z: https://docs.netlify.com/get-started/
6. SZRAJBMAN, Sallo. Set JWT with Spring Boot and Swagger UI. Baeldung. 2022. [cit. 2022-11-11]. Dostupné z: https://www.baeldung.com/spring-boot-swagger-jwt
7. Spring Boot Reference Documentation [online]. VMware, Inc., 2022 [cit. 2022-11-29]. Dostupné z: https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/
8. Securing a Web Application[online]. VMware, Inc., 2022 [cit. 2022-11-29]. Dostupné z: https://spring.io/guides/gs/securing-web/

Vedoucí bakalářské práce: **Ing. Radek Vala, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **2. prosince 2022**
Termín odevzdání bakalářské práce: **26. května 2023**

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 7. prosince 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 20.5.2023

Barbara Klimekova, v.r
…….
podpis studenta

ABSTRAKT

Táto bakalárska práca sa zameriava na návrh a vytvorenie webovej aplikácie pre prepojenie a spoluprácu vedeckých pracovníkov a podnikateľov v oblasti biotechnológií v regióne CEE. Cieľom je vytvoriť prototyp platformy, ktorá poskytne používateľom informácie o dostupnej podpore v oblasti biotechnológií v regióne, zviditeľní etablované produkty a služby v tomto odvetví vzniknuté v regióne a prepojí podnikateľské a tímy startupov s vedeckým know-how. V rámci práce budú preskúmané prípady použitia, špecifikované funkčné a nefunkčné požiadavky, vybrané vhodné technológie pre štruktúru backendovej a responzívne rozhranie aplikácie. Implementované funkcie budú pokrývať definované požiadavky a bude venovaná pozornosť na zaujímavé časti kódu. Nakoniec budú zhodnotené možnosti budúceho rozšírenia prototypu aplikácie o ďalšie funkcionality.

Kľúčové slová: React, Spring Boot, biotechnológie, veda webová platforma

ABSTRACT

This bachelor's thesis focuses on the design and development of a web application for connecting and collaborating scientific researchers and entrepreneurs in the field of biotechnology in the CEE region. The aim is to create a platform that provides users with information about available support in the field of biotechnology in the region, showcases established products and services in the industry that have emerged in the region, and connects business and startup teams with scientific researchers and available studies. The thesis includes an exploration of use cases, specification of functional and non-functional requirements, selection of suitable frameworks for the backend structure and responsive application interface. The implemented features will cover the defined requirements, with attention given to interesting code sections. Finally, the potential for future expansion of the application prototype with additional functionalities will be evaluated.

Keywords: React, Spring Boot, Biotechnologies, Science

Chcela by som vyjadriť svoje úprimné a srdečné poďakovanie všetkým, ktorí mi poskytli svoj čas, podporu a cenné rady pri vypracovaní tejto bakalárskej práce. V prvom rade svojmu skvelému školiteľovi Ing. Radekovi Valovi, Ph.D. za jeho cenné vedenie, odborné vedomosti a neustálu podporu. Jeho rady, návrhy a pripomienky mi veľmi pomohli pri vývoji tejto webovej platformy. Taktiež by som sa chcela poďakovať všetkým respondentom, ktorí venovali svoj čas na rozhovory, pripomienky a hodnotenie prvých verzií platformy. Ich názory a spätná väzba boli pre mňa neoceniteľné a významne prispeli k zlepšeniu aplikácie.

Prehlasujem, že odovzdaná verzia bakalárskej práce a verzia elektronicky nahraná do IS/STAG sú totožné.

OBSAH

ÚVOD

Bakalárska práca sa zaoberá návrhom a implementáciou webovej aplikácie pre vedeckých pracovníkov a podnikateľov v oblasti biotechnológií v regióne CEE. Cieľom práce je vytvoriť platformu, ktorá umožní prepojenie a kooperáciu medzi aktérmi v inovačnom biotechnologickom ekosystéme vrátane vedeckých pracovníkov, vzdelávacích a investorských inštitúcií a podnikateľov.

Práca sa skladá z dvoch hlavných častí: teoretickej a praktickej časti.

V teoretickej časti práce sa bude vykonávať porovnanie a výber technológii pre praktickú časť. Budú analyzované najpoužívanejšie frontendové a backendové frameworky v súčasnosti, ako aj databázy. Ďalej sa práca zameria na testovanie, vrátane funkcionálneho testovania, unit testovania a UX/UI testovania. Bezpečnosť bude tiež dôležitou súčasťou tejto časti, pričom sa budú skúmať základné princípy bezpečnosti pri vývoji frontendu, backendu a práci s databázou.

Praktická časť sa bude zaoberať potrebou a požiadavkami na webovú aplikáciu. Bude skúmať motiváciu za vytvorením tejto aplikácie a popíše cieľové skupiny. Špecifikujú funkcionálne a nefunkcionálne požiadavky na aplikáciu. Ďalej sa práca zameria na návrh a implementáciu aplikácie, vrátane architektúry a REST API modelu. Objasní sa zvolený prístup Atomic Design a odprezentujú sa základné prvky wireframe obrazoviek.

Posledné kapitoly budú venované ukážkam implementácie používateľského rozhrania a načrtnutia možného ďalšieho vývoja.

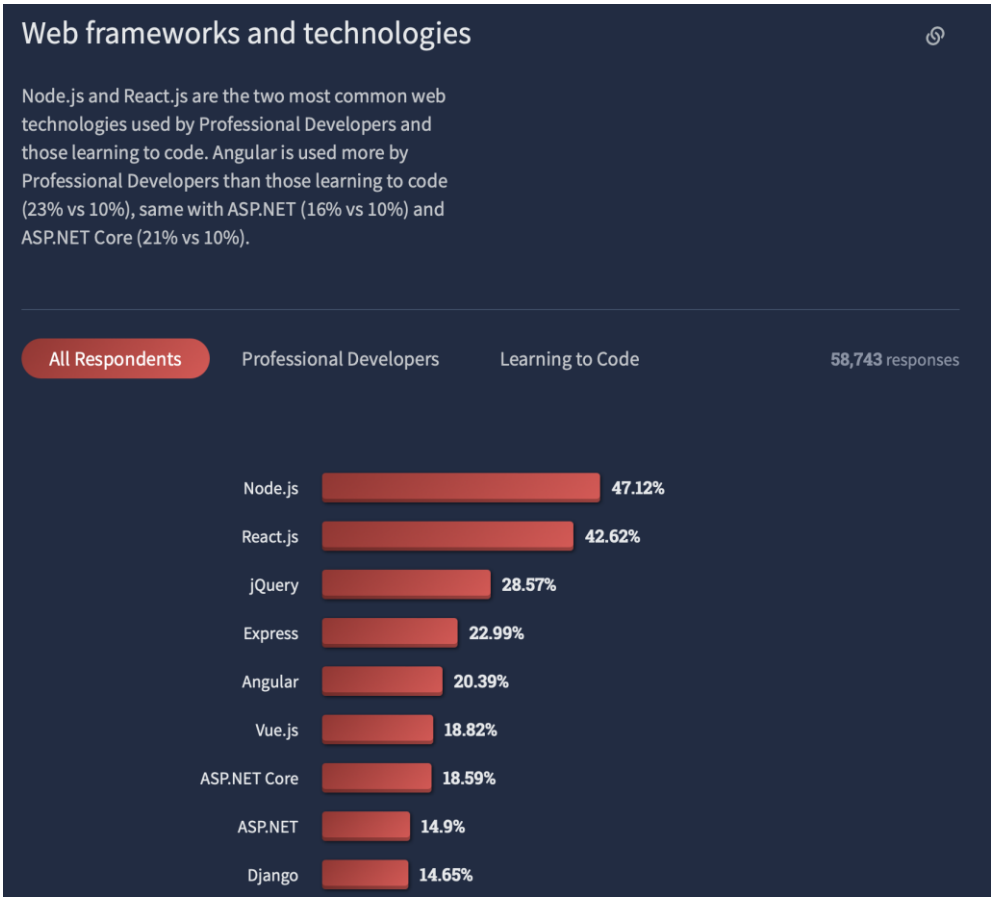
I. TEORETICKÁ ČÁST

1. POROVNANIE A VÝBER TECHNOLÓGIÍ

Úvodná kapitola je venovaná prieskumu medzi populárnymi technológiami pre tvorbu webových aplikácii a ich analýze, ktorá je podkladom pre vhodný výber výsledného technologického rámca, tzv. *tech-stack*, v praktickej časti.

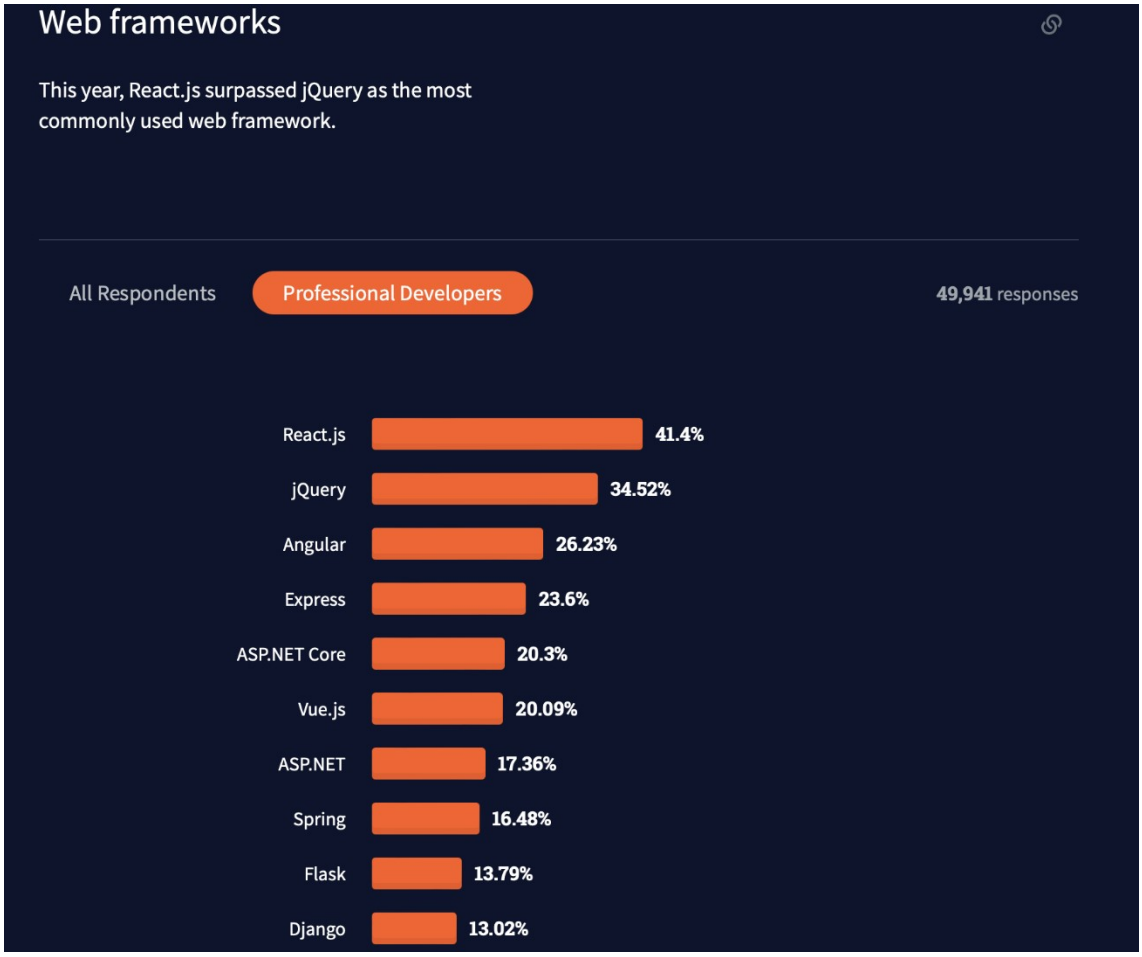
1.1 Najpoužívanejšie frontendové frameworky v súčasnosti

Frontend framework je softvérová knižnica, ktorá poskytuje štruktúru pre vytváranie webových stránok a aplikácií. Pomáha vývojárom vytvárať efektívnejšie aplikácie, ktoré sú ľahko udržiavateľné a optimalizované pre rýchle načítanie. [3] Frontendové rámce vo všeobecnosti umožňujú znova používať jednotlivé časti kódu, čo veľmi zjednodušuje prácu vývojárom a ušetrí veľa času počas vývojového procesu. Každý rok prichádzajú na trh nové frameworky a portál *Stack Overflow* zostavuje na základe hlasovania profesionálnych aj začínajúcich programátorov každoročne ich rebríček. V roku 2022 sa do dopytovania pre kategóriu *Webové frameworky a technológie* zapojilo cez 58000 vývojárov z celého sveta.[1] Ako najrelevantnejšie boli vybrané napríklad React, Angular či Vue.js.



Obrázok 1: Webové frameworky a technológie, rebríček Stack Overflow 2022 [1]

Na porovnanie uvádzame prehľad najobľúbenejších webových technológií aj z roku 2021. Rovnako v minulých rokoch si React udržal vyše 40% hlasov od 49 941 respondentov. [2] Angular aj Vue.js sa tiež znovu objavili medzi najpopulárnejšími možnosťami. Aj preto sme ich zaradili do porovnania pre účel tejto bakalárskej práce a venujú sa im nasledujúce kapitoly.



Obrázok 2: Webové frameworky a technológie, rebríček Stack Overflow 2021 [2]

1.1.1 **React**

React získal pozornosť vďaka svojmu tvorcovi - firme Facebook, ktorý vyvinul tento framework ako jednoduchú JavaScriptovú knižnicu. Medzi základné vlastnosti React frameworku patria komponenty a riešenie automatického stavu používateľského rozhrania, tzv. *UI state*

management. React prostredníctvom neho rieši správu a synchronizáciu dátového stavu aplikácie v rámci navrhnutých komponentov.[3]

Jeho dynamickosť je jeho hlavnou výhodou - prekresľuje len tie prvky, ktoré sa zmenili. Oficiálna dokumentácia opisuje základné princípy Reactu, a jedným z nich je tzv. *component-based* architektúra, ktorá umožňuje vývojárom vytvárať malé, skladateľné a znovu použiteľné komponenty. Okrem toho je React jednoduchý, čitateľný a ľahko sa v ňom orientuje. Je v ňom možné používať HTML tagy priamo v kóde a JSX umožňuje vytvárať HTML šablóny, ktoré používajú podmienené renderovanie. React používa *one-way data flow*, teda jednosmerný tok dát, ktorý umožňuje prekreslenie komponentov na základe nového stavu.[4]
Vďaka veľkému počtu knižníc a vysokému hodnoteniu od *Stack Overflow* je React stále na popredných miestach medzi najpopulárnejšími webovými technológiami.

1.1.2 **Vue.js**

Vue.js je rovnako ako React moderný JavaScriptový framework, ktorý sa používa na vytváranie interaktívnych aplikácií. Výraznou črtou je štruktúra Vue komponentov, ktoré sa vytvárajú pomocou formátu súborov známeho ako *Singe-File Component*. Tieto súbory typu **.vue* zahrňujú celkovú logiku komponentu písanú najmä v JavaScripte, HTML šablónu a aj štýlovanie pomocou napríklad CSS. Jednotlivé časti súboru sú zapuzdrené do tagov *<Template> </Template>, <Style> </Style>a <Script></Script>*. [5] Do popredia sa dostáva hlavne po vydaní jeho verzie Vue.js 3 po roku 2020.

1.1.3 **Angular**

Angular je framework vystavaný na programovacom jazyku TypeScript, ktorý je rozšírením, dá sa povedať, že nadstavbou jazyka JavaScript. Jeho výraznou aktualizáciou je statická typová kontrola. Základné stavebné bloky Angularu sú komponenty, šablóny, direktívy a *dependency injection*. Každý komponent kombinuje logiku, šablónu a štýly do jedného celku. Logika komponentu je implementovaná v TypeScripte a zodpovedá za spracovanie udalostí a dát. Šablóna je napísaná v jazyku HTML a definuje, ako sa komponent vizuálne zobrazuje. Komponenty umožňujú znovupoužiteľnosť, modulárnosť a oddelenie zodpovedností v rámci aplikácie. Direktívy sú značky v šablóne, ktoré poskytujú možnosť modifikovať správanie a vzhľad elementov DOM. Direktívy umožňujú pridávať funkcionalitu do

existujúcich HTML elementov. *Dependency injection* je v tomto prípade technika, ktorú Angular využíva na poskytovanie závislostí komponentom a službám. [6]

1.1.4 **Iné**

Medzi frontendovými technológiami, ktoré získali pozornosť v uplynulých rokoch a nachádzajú sa aj pomerne vysoko v spomenutom prieskume od *Stack Overflow* platformy je napríklad jQuery či Svelte.
jQuery je knižnica JavaScriptu, ktorá zjednodušuje manipuláciu s DOM - *Document Object Model* a spracovanie udalostí. Hoci nie je frameworkom v pravom zmysle slova, má silnú popularitu vďaka svojej jednoduchosti a rôznym funkciám.[7]
Svelte je relatívne nový frontendový framework, ktorý sa zameriava na kompiláciu kódu do optimálneho a výkonného JavaScriptu. Na rozdiel od Reactu a Angularu, Svelte nepoužíva virtuálny DOM. Okrem toho Svelte nevyžaduje zložitý systém zostavovania, pretože je implementovaný ako knižnica používateľského rozhrania, čo uľahčuje rýchle uvedenie do prevádzky. [8]

1.2 Najpoužívanejšie backendové frameworky v súčasnosti

V rámci *Developer Survey* rebríčka od *Stack Overflow* kampane sa zbierajú aj názory na backendové technológie. Podľa štatistík sú tri z najpopulárnejších backendových rámcov pre vývoj webových aplikácií Django, Spring Boot a Node.js. Django je framework založený na Python jazyku, ktorý umožňuje rýchly vývoj a nasadenie webových aplikácií. Spring Boot je framework založený na Jave. Node.js je prostredie založené na JavaScripte, ktoré umožňuje rýchly vývoj výkonných webových aplikácií. Každý z týchto rámcov ponúka jedinečné výhody, vďaka čomu sú obľúbenou voľbou pre vývoj webových aplikácií.

Obľúbenosť a používateľnosť týchto technológií sa dá podložiť aj napríklad prehľadom z portálu *Github*. V sekcii *Topic* sú dostupné informácie napríklad o kategóriách *Most Stars* alebo *Most Forks*, ktoré odzrkadľujú preferencie užívateľov tejto medzinárodne známej platformy pre správu repozitárov. V kategórii *Most Stars* sa aktuálne nachádza Django či Spring Boot, ale aj ďalšie Spring frameworky.[9]

1.2.1 **Spring Boot**

Spring Boot vedie v *Github* štatistikách aj v kategórii *Most Forks* [10], čo znamená, že z jeho repozitárov si vývojári najčastejšie robia fork verzie a môžeme to brať za znak toho, že s ním naozaj aktívne pracujú. Je to moderný framework pre rýchle vytváranie aplikácií v jazyku Java. Je postavený na Spring frameworku a teda na základoch technológií, ako sú Spring Framework, Spring MVC a Spring Data. To umožňuje vývojárom využiť existujúci výkon a flexibilitu týchto rámcov a zároveň využiť nové funkcie a nástroje, ktoré poskytuje rámec Spring Boot. Tento framework sa snaží minimalizovať nutnosť ručnej konfigurácie tým, že poskytuje prednastavené hodnoty a automatické konfigurácie. [11]

Využíva širokú škálu anotácií, ktoré poskytujú rôzne metadáta a konfigurácie pre rôzne časti aplikácie. Okrem auto-konfigurácie ponúka aj veľa ďalších vylepšení, ako je auto-konfigurácia, bezplatná údržba, integrácia s populárnymi technológiami a jednoduchý spúšťač aplikácií či vstavané servery Tomcat alebo Jetty. Umožňuje rýchlu implementáciu aplikácií s malou alebo žiadnou konfiguráciou. Pomocou Spring Boot frameworku môžu vývojári jednoducho vytvárať robustné webové aplikácie a služby s malou námahou bez toho, aby museli špecificky konfigurovať základný rámec. [12]

1.2.2 **Django**

Django je backendový framework napísaný v Pythone a je založený na architektonickom vzore MVC - Model-View-Controller, ktorý sa však tomto frameworku nazýva ako MVT - Model-View-Template. Namiesto kontroléra sa využíva tzv. *Template*, ktorý oddeľuje obchodnú logiku od prezentačnej vrstvy aplikácie. [13]

Django framework je populárny, nakoľko má tiež mnoho funkcií, ktoré ho robia vhodným na vývoj komplexných dátovo-orientovaných webových stránok, vrátane objektovo-relačného mapovania - ORM.[14] To zjednodušuje proces pripojenia sa k databáze, rozširiteľného systému šablón na vytváranie HTML formulárov a systému autentifikácie na správu užívateľských účtov. Navyše, tento framework poskytuje veľa nástrojov na automatizáciu bežných úloh webového vývoja.

1.2.3 **Node.js**

Node.js je *open-source* multiplatformové prostredie, ktoré je primárne určené na vývoj serverovej časti webových aplikácií. Node.js umožňuje programovať celé projekty, to znamená klientsku i serverovú časť, v jednom jazyku, teda v JavaScripte. Je riešený ako *single-thread*, teda pracuje iba v jednom vlákne a v súčasnosti podporuje širokú škálu súvisiacich technológií, vrátane použitia API, databáz a serverového skriptovania.[15]
Rovnako ako u prehliadača Google Chrome je aj u Node.js jadrom interpret Chrome V8. Práve to sa berie ako jedna z jeho najvýraznejších výhod. *"K výhode rýchlosti Node.js prispieva niekoľko faktorov. Prvým je Google V8 engine, ktorý poháňa bleskovo rýchlu konverziu JavaScriptu na strojový kód. Aplikácie Node.js bežia oveľa rýchlejšie ako iné aplikácie. Pretože Node používa asynchrónny, neblokujúci programovací model, procesy môžu bežať paralelne namiesto čakania na dokončenie iných procesov. Výsledkom je vyššia rýchlosť a výkon."* [16]

1.3 Databázy

Existujú rôzne typy databáz, ktoré sa líšia svojím štruktúrovaným a spôsobom ukladania údajov. Medzi najznámejšie patria: relačné databázy, NoSQL databázy, dokumentové databázy a grafové databázy a ďalšie.

Relačné databázy používajú štruktúru tabuliek na ukladanie údajov a vzťahov medzi nimi. Tieto databázy poskytujú výhody, ako sú jednoduché a intuitívne spracovanie údajov, spoľahlivosť a schopnosť vyhľadávať a kombinovať údaje z rôznych tabuliek. [17] Aj preto môžeme považovať tento typ databáz je vhodnú voľbu pre prototypovanie webovej aplikácie s architektúrou REST API.

Najväčší význam pre tento účel poskytujú napríklad tieto výhody: [17]

Jednoduchá štruktúra: Štruktúra tabuliek vzťahy medzi nimi, ktoré používajú relačné databázy sú intuitívne a ľahko pochopiteľné pre väčšinu vývojárov.

Výkonné vyhľadávanie a filtrovanie: Relačné databázy poskytujú efektívne nástroje na vyhľadávanie a filtrovanie údajov.

Možnosť vzájomných vzťahov medzi údajmi: Relačné databázy umožňujú modelovať vzájomné vzťahy medzi údajmi, čo je pre mnoho aplikácií kľúčové

Pre objektívnosť je dôležité spomenúť aj možné obmedzenia. Relačné databázy môžu požadovať väčšiu výkonnosť hardvéru, aby boli schopné efektívne spracovať veľké objemy údajov. Nevýhodou pri ich použití môže byť aj obmedzená flexibilita. Relačné databázy sú nastavené na štruktúrovanie údajov v tabuľkách a vzťahoch medzi nimi, čo môže byť pre niektoré aplikácie obmedzujúce. A niektoré dáta sa do takejto podoby nedajú dostať.

Vo všeobecnosti teda platí, že relačné databázy sú dobrou voľbou pre prototypovanie webovej aplikácie s architektúrou REST API, pokiaľ to umožňujú požiadavky na údajovú štruktúru a vzájomné vzťahy medzi údajmi. Iný typ databázy, ako napríklad NoSQL databáza, môže byť lepšou voľbou, pokiaľ sú požiadavky na flexibilitu v údajovej štruktúre vyššie. V tejto práci sme sa zamerali na využitie relačnej databázy pre dané požiadavky na platformu.

1.4 Výber technológií pre praktickú časť

Súbor technológií, tzv. *tech-stack* našej aplikácie bude tvoriť Spring Boot, React.js, React Bootstrap a CSS štýlovanie. Z pohľadu práce s dátami bude implementovaná interná relačná databáza H2 a čiastočne služby AWS S3.

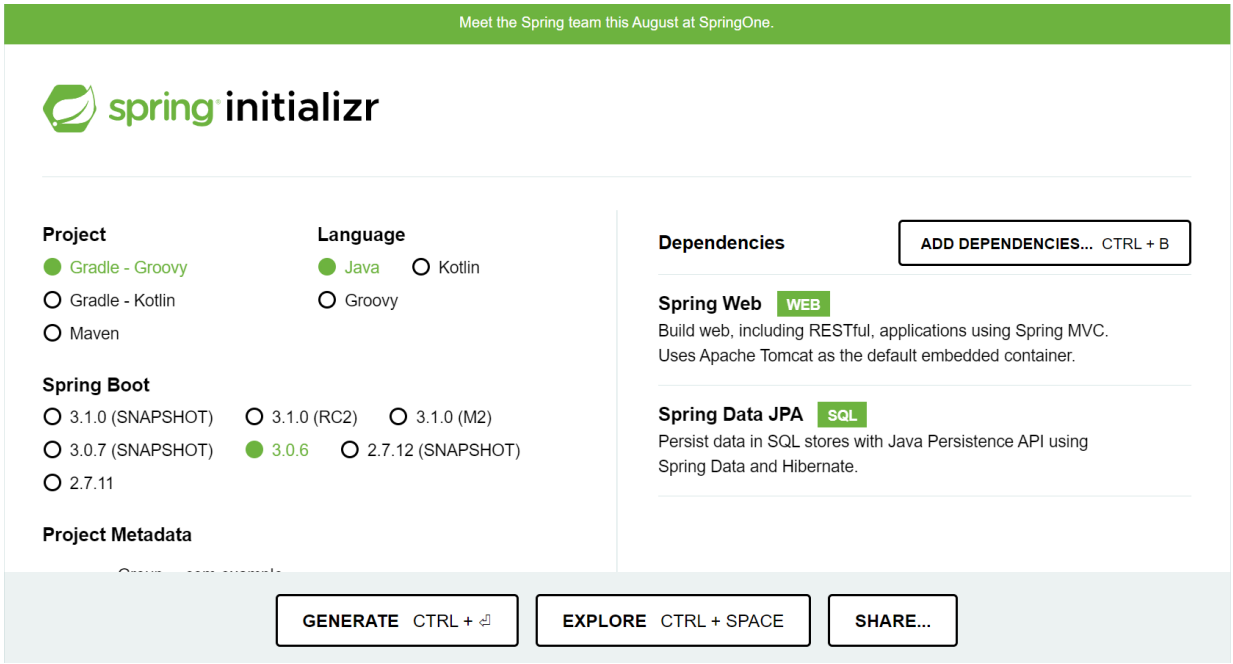
Pri zohľadnení parametrov jednotlivých technológii sme prihliadali aj na budúcu potenciálnu integráciu so systémami organizáciami stakeholderov. Tými sú z tohto pohľadu najmä vzdelávacie a vedecké inštitúcie lokálneho aj medzinárodného charakteru. Ich systémy sú častokrát robustné, nakoľko spĺňajú viacero paralelných funkcií. Mnohé z nich majú backend postavený na jazyku Java. Frontendové rozhrania sa líšili vo väčšej miere. Zároveň, keďže ide o *open-source* platformu a jej kód bude zverejnený na *Githube* s možnosťou prispievania do vývoja aj od externých dobrovoľníkov, zvolili sme technológie, ktoré majú značnú komunitu používateľov medzi vývojármi a existuje k nim kvalitná a aktualizovaná dokumentácia. Výber technológie bol definovaný aj v požiadavkách na softvérové riešenie, ktoré sú popísané detailnejšie v kapitole 2.2.

Pre backend našej aplikácie sme vybrali framework Spring Boot pre tieto dôvody: [18]

- Umožňuje vývojárom vytvoriť REST API rýchlo as minimálnymi konfiguráciami. Pri vývoji prototypu môžeme použiť funkciu *Spring initializr*, ktorý nám prednastaví východiskovú štruktúru Spring Boot aplikácie, ako je vidno na obrázku 3.
- Spring Boot navyše poskytuje vstavané bezpečnostné funkcie, ako je autentifikácia a autorizácia, a podporuje rôzne databázy a systémy zasielania správ.

- Spring Boot obsahuje default natívny Tomcat server.
- Spring Boot používa techniku *bootstrappingu* na uloženie pamäte. V tomto spôsobe je zdrojový jazyk skompilovaný pomocou boot inicializátora. Výsledkom je, že aplikácie sa načítajú oveľa rýchlejšie.

Spring Boot je postavený na Jave, rozšírenom backend jazyku, ktorý je celosvetovo a aj v CEE rozšírený medzi profesionálnymi programátormi a je predpoklad, že v prípade potreby vieme zistiť ľudské kapacity pre vývoj portálu v budúcnosti. Po prieskume a analýze dostupnej dokumentácie môžeme tvrdiť, že nové verzie Javy aj Spring Boot majú dobre štruktúrovanú a obsiahlu dokumentáciu.



Obrázok 3: UI Rozhranie Spring Boot Initializer [19]

Vzhľadom na to, že Spring Boot je softvér založený na open source, skladá sa z veľkej komunity vývojárov. Títo ľudia sú pripravení zdieľať svoje vedomosti alebo dokonca poskytovať pred-stavané kódy pre vývoj. Nezávisle od technickej odbornosti môžete nájsť rôzne učebné materiály online. Rovnako vychádzame z predpokladu, že bude pri tomto jazyku možná kontribúcia zo strany dobrovoľníkov v prípade sprístupnenia zdrojového kódu navrhovanej aplikácie na platforme Github, ako bolo spomenuté v predchádzajúcich kapitolách.

Pre frontend našej aplikácie sme vybrali framework React najmä kvôli týmto dôvodom: [20]

- Rovnako ako Spring Boot aj React má rozsiahlu dokumentáciu, množstvo moderných knižníc a aktívnu komunitu.
- Aplikácie v React.js sa vytvárajú ako séria komponentov, pričom funkcie sa prenášajú z nadradeného komponentu do podriadeného komponentu vo forme argumentov. Toto sa nazýva jednosmerné viazanie údajov alebo *one-way data binding*. Kvôli tejto vlastnosti je veľmi pohodlné vykonávať zmeny v aplikáciách postavených na React technológii. Akákoľvek zmena, ktorú vykoná v podriadených komponentoch, sa neprejaví v nadradených komponentoch, a preto je kód konzistentný aj po zmenách a aktualizáciách. [3]

Okrem knižnice React využijeme vo frontend časti aj *React Bootstrap s Material Design* princípmi - *mdbootstrap*.[21] Mdbootstrap je vystavaný na Bootsrtape, no zahŕňa aj priamu integráciu s frontentovými technológiami ako , Vue, Angular, či React a súčasne napĺňa Material Design princípy. "*Material Design je dizajnový systém vytvorený a podporovaný dizajnérmi a vývojármi Google. Material.io obsahuje hĺbkové usmernenie UX a implementácie komponentov používateľského rozhrania pre Android, Flutter a web.*"[22] Štýlovanie webovej aplikácie bude doplnené o kaskádové štýlovanie CSS.

Pre databázovú schému sme si pre prototyp vznikajúcej aplikácie vybrali databázu H2. Ide open-source relačnú databázu, ktorá sa často používa pre webové aplikácie úvode ich vývoja. Táto databáza je navrhnutá tak, aby bola ľahko integrovateľná do webových aplikácií a poskytovala jednoduchý spôsob práce s údajmi. Je určená pre aplikácie s nízkymi nárokmi na výkon a môže byť bez problémov embedovaná do aplikácie, čo umožňuje jednoduché spravovanie údajov bez nutnosti inštalovať ďalšie databázové prostredie. Tento typ databázy je vhodný pre menšie aplikácie alebo ako prototyp pre vývoj väčších aplikácií, ktoré budú vyžadovať väčšiu kapacitu a výkon. [23]
Často sa teda využíva najmä pri prototypovaní a v následnom rozvoji projektu je ju možné bez problémov nahradiť robustnejším riešením. Jej konfigurácia v rámci Spring Boot frameworku je tiež pomerne jednoduchá.

2 TESTOVANIE

Testovanie by malo byť automatickou súčasťou každého vývoja. Nutnosťou je kombinovanie rôznych typov testovania aspoň na úplne základnom rámci. V tejto časti bakalárskej práce je popísaný priebeh a potreba funkcionálneho testovania, unit testovania aj testovania používateľského rozhrania.

2.1 Funkcionálne testovanie

Funkcionálne testovanie - tzv. *Functional testing* je druh testovania, ktorý skúma funkčnosť aplikácie alebo systému podľa špecifikácie. Zameriava sa na to, či aplikácia reaguje správne na vstupy a výstupy, a či splňuje požiadavky na funkčnosť. [24]

Tento typ testovania webových aplikácií je kľúčovým krokom v procese vývoja softvéru. Cieľom je overiť, či aplikácia funguje tak, ako má, a či spĺňa požiadavky zainteresovaných strán. Počas funkčného testovania webovej aplikácie testeri testujú rôzne scenáre a vyhodnocujú reakcie systému, aby sa zabezpečilo, že aplikácia správne funguje a poskytuje očakávané výstupy. Tento typ testovania pomáha identifikovať a vyriešiť problémy s funkčnosťou, používateľským skúsenostiam a výkonom aplikácie. V praktickej časti práce sa výsledný prototyp aplikácie otestuje na základe vopred definovaných používateľských prípadov a zameriame sa na testovanie, či novo-navrhnutý prototyp webovej aplikácie spĺňa funkcionálne a nefunkcionálne požiadavky.

2.2 Unit testovanie

Unit testovanie je druh testovania softvéru, ktorý sa zameriava na testovanie jednotlivých jednotiek alebo komponentov aplikácie. Cieľom unit testovania je overenie správnosti kódu na najnižšej úrovni architektúry softvéru. Unit testy sú automatizované testy, ktoré sa spúšťajú v izolácii od ostatného systému a overenie správania jednotlivých jednotiek kódu. [26]

V rámci Spring Boot frameworku, ktorý sa bude využívať pre backendovú časť, je unit testovanie uľahčené pomocou modulu *Spring Test*, ktorý poskytuje funkcie pre testovanie Spring komponentov a aplikácií. Tieto funkcie zahŕňajú podporu pre *mockovanie*, testovanie s databázou a testovanie webových služieb. Použitím unit testovania môžu vývojári chyby včas odhaliť a opraviť v cykle vývoja, čím sa znižuje riziko chýb a zlepšuje celková kvalita

softvéru. V jazyku Java sú populárne knižnice pre unit testovanie napríklad JUnit či Mockito. Takéto knižnice poskytujú nástroje a rámce na písanie, spúšťanie a hlásenie unit testov. [25]

JUnit je open-source knižnica pre Javu, ktorá sa používa na automatizované testovanie unit častí Java kódu. JUnit poskytuje jednoduchý a prehľadný spôsob, ako písať, spúšťať a sledovať testy, a tiež poskytuje výstup, ktorý ukazuje, či boli testy úspešné alebo nie. [26]

Mockito je tiež open-source knižnica pre Javu, ktorá sa používa na tvorbu mock objektov pre testovanie. Mock objekty sú náhradné objekty, ktoré sa používajú na testovanie a simulujú správanie objektov, ktoré sú závislé na iných objektoch. [27]

Pomocou Mockito môžu vývojári testovať jednotlivé funkcie alebo metódy v aplikácii bez toho, aby museli spúšťať všetky súvisiace objekty alebo služby. To im umožňuje testovať izolovane a znižuje riziko, že sa vyskytnú problémy s kódom.

Výhody používania takýchto knižníc sú: [28]

- Umožňujú automatizovať testovanie a zrýchliť proces vývoja
- Zlepšujú kvalitu kódu a pomáhajú vyhnúť sa chybám
- Umožňujú vývojárom testovať jednotlivé funkcie alebo metódy v izolácii od ostatného kódu
- Umožňujú jednoduché a prehľadné sledovanie výsledkov testov a hlásenie chýb.

V praktickej časti pokryjeme unit testami backendovú časť webovej aplikácie.

2.3 UX a UI testovanie

User Experience - UX testovanie je proces hodnotenia produktu alebo služby zameraný na to, ako s ním používatelia interagujú a ako sa cítia. Cieľom UX testovania je zlepšiť celkový používateľský zážitok tým, že identifikuje oblasti, kde by mohol byť produkt efektívnejší pre používateľa a ako ho urobiť pre použitie ešte viac jednoduchším. UX testovanie môže zahŕňať úlohy, ako sú testovanie použiteľnosti, prieskumy a focus skupiny. [29] [30]

User Interface - UI testovanie sa naopak zameriava na overenie, že vizuálne prvky produktu sú zobrazené správne a fungujú tak, ako majú. UI testovanie overuje vzhľad, pocit a správanie používateľského rozhrania, vrátane umiestnenia tlačidiel, vzhľadu textu a funkčnosti rozbaľovacích menu a iných interaktívnych prvkov. UI testovanie sa často vykonáva s pomocou

automatizovaných testovacích nástrojov, ktoré môžu kontrolovať konzistenciu a presnosť na viacerých zariadeniach a platformách.

Hlavným rozdielom medzi UX a UI testovaním je teda ich zameranie. Oba sú dôležité pre zabezpečenie celkovej kvality produktu a poskytnutie pozitívneho používateľského zážitku. Zatiaľ čo UX testovanie sa zvyčajne vykonáva s reálnymi používateľmi, UI testovanie sa zvyčajne vykonáva vývojovým tímom pomocou automatizovaných nástrojov. [24]

3 BEZPEČNOSŤ

Bezpečnosť webových stránok je dôležitou súčasťou ich dizajnu a vývoja. Existujú tri hlavné oblasti, ktoré je potrebné zohľadniť pri zabezpečení webových stránok: bezpečnosť na frontendovej časti, bezpečnosť na backendovej časti a bezpečnosť databázy. Okrem špecifických aspektov bezpečnosti pre tieto jednotlivé 3 časti by sa mali vývojárske tímy zaoberať aj rámcovými princípmi, tzv. *best practice* ako sú: [31]

- aktualizácia systému a verzií technológii, nástrojov, databáz a ďalších častí softvéru,
- autentifikácia a autorizácia by mala byť súčasťou architektúry od samého začiatku. Overovanie, ktorý typ používateľa má práva ku ktorej časti obsahu alebo správy rozhrania či ďalším informáciám je dnes už štandardom
- zálohovanie informácií, dát, a ich prípadná rýchla obnova.
- Skenovanie kódu už počas fázy programovania. *Ide o implementáciu skenovania kódu pomocou nástroja na statické testovanie bezpečnosti aplikácií (SAST) alebo nástroja na analýzu zloženia softvéru (SCA) alebo vložiť svoju aplikáciu na vývojový server a skenovať ju pomocou nástroja na dynamické testovanie bezpečnosti aplikácií (DAST).* [31]

3.1 Základné princípy bezpečnosti pri frontend vývoji

Na strane prehliadača a strane frontend časti aplikácie sa bezpečnosť týka napríklad oblastí ako validácia vstupov vo formulároch. Táto validácia by mala byť v ideálnom prípade na oboch stranách backend-frontend. Hlavným cieľom na frontend strane je pri tejto téme minimalizácia rizík útokov typu Cross-Site Scripting (XSS).[32]

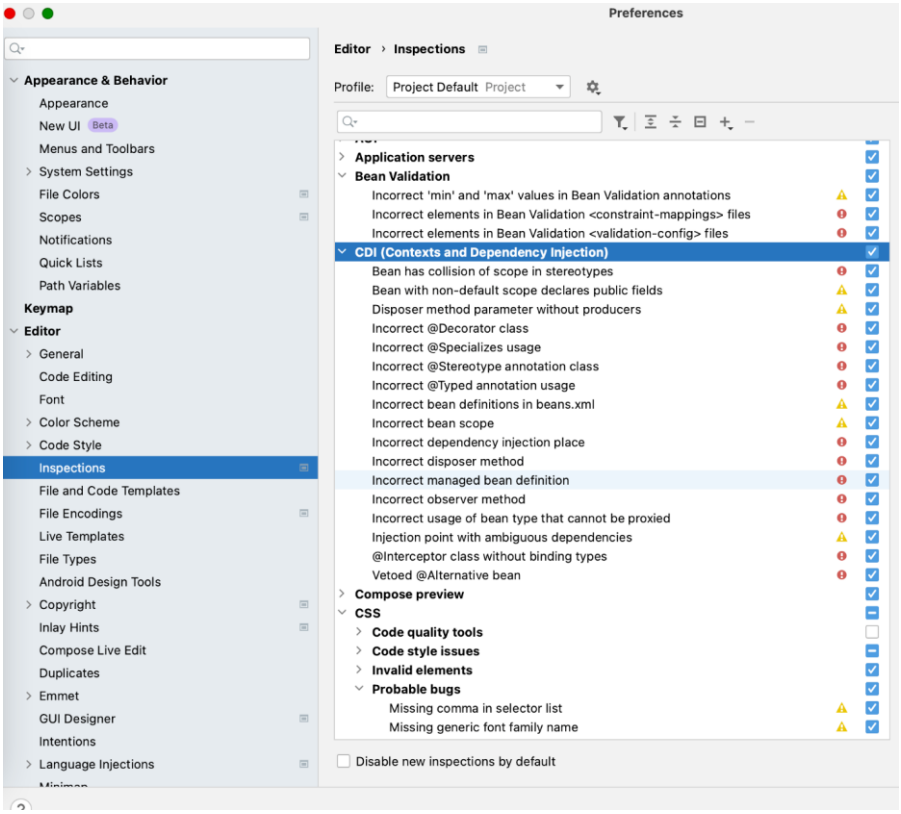
Ďalej ide napríklad o autentifikáciu a autorizáciu, aby sa zobrazoval iba želaný obsah podľa úrovne používateľa. Klasickým príkladom je prihlásený používateľ, používateľ s úrovňou registrovaného používateľa a administrátor stránky. Rovnako aj tento bod má implementačné presahy z backendu na frontend.

Zabúdať by sa nemalo ani na správu stavov obsahu, tzv. State management. Napríklad v prípade React frameworku sa o to stará Redux alebo React Context. Tie zabezpečujú jednosmerný tok dát, čo znamená, že zmeny stavu sú vykonávané iba prostredníctvom definovaných akcií a reduktorov ak hovoríme o Reduxe alebo v prípade React Contextu je to zabezpečené prostredníctvom komponentov poskytujúcich stav. Toto zaisťuje kontrolu nad tým, kto a ako mení obsah a celkový stav aplikácie. [3]

3.2 Základné princípy bezpečnosti pri backend vývoji

Okrem vyššie spomenutých oblastí sa by sa pri vývoji backendovej časti malo dohliadať na pravidelnú aktualizáciu použitých knižníc a verzií databáz a pod.

Základnou činnosťou vývojového tímu alebo jednotlivca by mala byť aj pravidelná kontrola kódu. Na to môže slúžiť aj vhodne vybrané vývojové prostredie. Napríklad v prostredí IntelliJ IDEA sú dostupné nástroje *Inspections* , ktoré slúžia na statickú analýzu a kontrolu kódu v rámci vývojového prostredia. Tieto nástroje umožňujú identifikovať potenciálne p problémy, nedostatky v kóde a poskytujú odporúčania na zlepšenie kvality a bezpečnosti kódu. [33]



Obrázok 4: Nastavenie *Inspections* funkcie v prostredí Intellij IDEA

Ďalšou významnou oblasťou sú parametrizované dotazy. To znamená, že pri práci s databázou používame na backend strane parametrizované dotazy namiesto vkladania užívateľských vstupov priamo do SQL príkazov. To umožní oddeliť vrstvu dát od samotných SQL príkazov. Napríklad Spring Boot ponúka možnosť pracovať s JPA - *Java Persistence API* a ORM - *Object Relational Mapping* nástroje sú automaticky súčasťou Spring Boot

frameworku a automaticky poskytujú parametrizované dotazy.[34] Spring Boot poskytuje tuto integráciu s Java Persistence API prostredníctvom Spring Data JPA modulu.

3.3 Základné princípy bezpečnosti pri návrhu a práci s databázou

Bezpečnosť databázy zahŕňa opatrenia, ktoré chránia dáta uložené v databáze pred útokmi a zneužitím. Patria sem opatrenia ako zabezpečenie prístupu k databáze pomocou silných hesiel a autentifikácie, šifrovanie dôverných informácií ako napríklad hesiel, a pravidelné zálohovanie dát na bezpečnom mieste.

II. PRAKTICKÁ ČÁST

4 POTREBA A POŽIADAVKY

V kapitole Potreba a požiadavky sú uvedené prvotné motivácie pre vývoj webovej platformy pre mladé vedecko-podnikateľské tímy a jednotlivcov a zadefinované požiadavky na jej funkčnosť a prípady použitia. Tie vyplývajú z viacerých diskusií s cieľovými skupinami o potrebe takej platformy a boli niekoľkokrát validované aj pri testovaní wireframe návrhov.

4.1 Motivácia

V región CEE krajín je dlhodobým problémom miera uplatnenia vedy a technologických objavov v inovačnom ekosystéme. Platí to najmä pre štáty ako Albánsko, Bulharsko, Chorvátsko, Česká republika, Maďarsko, Poľsko, Rumunsko, Slovenská republika či Slovinsko. Výnimkou z CEE krajín v tomto aspekte je napríklad Estónsko, ale i ďalšie pobaltské štáty Lotyšsko a Litva.

V krajinách CEE je potrebné zvýšiť podporu vedy a techniky s cieľom zlepšiť ich inovačnú schopnosť. Dá sa to dosiahnuť zvýšeným financovaním výskumu a vývoja, ako aj podporou spolupráce medzi výskumnými inštitúciami a priemyslom a prenosom poznatkov medzi krajinami. To si však vyžaduje rozsiahle systémové zmeny na viacerých úrovniach politického ekonomického aj spoločenského vnímania. [35]

Jednou z oblastí, na ktorú sa pri stratégiách podpory zavádzania vedeckých inovácii do praxe často zabúda, je prepojenie mladých vedeckých pracovníkov s možnosťami ďalšieho rozvoja praktických skúseností v startup systéme. Práve v prostredí rýchlo rastúcich technologických firiem - startupov - existujú metódy rýchleho prototypovania a testovania, ktoré vie využiť pre svoj budúci rozvoj aj segment vedy. Tieto príležitosti už existujú, no nie sú dobre známe medzi cieľovými skupinami a zároveň ani medzi organizáciami, ktoré ich sprostredkúvajú navzájom.

Bakalárska práca pracuje s hypotézou , že na trhu chýba jasný a prehľadný portál mapujúci možnosti pre mladých vedcov pri získavaní podnikateľských zručností alebo rozbiehaní vlastného vedeckého podnikateľského nápadu, ktorý by na jednom mieste spájal už existujúce kvalitné programy a príležitosti. Vyplýva to z mnohých aktuálnych diskusií vo vedeckej aj podnikateľskej sfére a táto potreba bola identifikovaná v spolupráci s predstaviteľmi stakeholderov z oboch segmentov ako aj samotnými študentmi vedeckých a prírodovedeckých oborov.

4.2 Popis projektu

Práca má za ciel navrhnúť prototyp možného riešenia vo forme webového portálu spájajúceho už existujúce formy získavania vedecko-podnikateľskej praxe ako aj podpory pre začínajúce vedecké inovácie. Zároveň vytvorí aktívnu databázu firiem, vzdelávacích inštitúcii aj medzinárodných programov pôsobiacich v EU, ktoré takúto podporu mladým vedeckým pracovníkom poskytujú. Inšpiráciou pre tento sektor môžu byť už existujúce informačné portály pre startup ekosystém.

Cieľom webovej aplikácie nie je vytvárať nový obsah, ale byť prehľadným nástrojom, ktorý uľahčí prístup k informáciám a vytvorí užitočný nástroj pre všetky zainteresované strany.

Keďže ide o myšlienku, ktorá bola identifikovaná na viacerých *hackathon* podujatiach či v rámci mentorovania vo vzdelávacích biznisových inkubátoroch, bude prototyp aplikácie open-source platformou sprístupnenou na *Githube* s možnosťou ďalšieho rozvoja v spolupráci s dobrovoľníkmi. Všetky dáta zozbierané a akumulované na navrhovanej platforme sú z oficiálnych verejných zdrojov a jednotlivé časti platformy riadne odkazujú na pôvodné zdroje akými sú napríklad oficiálne stránky akceleračných a inkubačných programov, verejné *LinkedIn* profily odborníkov v praxi poskytujúcich mentoring začínajúcim vedeckým tímom a pod.

V bakalárskej práci sa venujeme len výseku identifikovaných potenciálnych funkcionalít a prípadov použitia. Projekt má ambíciu reálneho nasadenia po dopracovaní ďalších častí.

Prototyp aplikácie je publikovaný vo verejnom repozitári *https://github.com/BaskaKlim/howtodobiotech*. Keďže ide o open-source, po analýze bola zvolená licencia MIT.[36] Ide o jednoduchú základnú licenciu s podmienkami vyžadujúcimi iba zachovanie autorských práv a licenčných upozornení. Licencované diela, modifikácie a väčšie diela môžu byť distribuované za iných podmienok a bez zdrojového kódu

Obrázok 5: Text licencie MIT [36]

4.3 Cieľové skupiny a stakeholderi

Navrhovaná aplikácia má niekoľko typov cieľových používateľov. Sú medzi nimi primárni konzumenti obsahu aj potenciálni rozširovatelia databázy biotechnologického ekosystému v CEE.

4.3.1 **Mladí vedeckí pracovníci a študenti vedeckých odborov**

Cieľovou skupinou pre webový portál by boli primárne mladí vedci a študenti vedeckých oblastí. Sú to osoby zaujímajúce sa o výskum, inovácie a vedu a hľadajúce príležitosti na rozšírenie svojich kariér alebo vedomostí v týchto oblastiach. Ide predovšetkým o študentoch a absolventoch a doktorandov technických a prírodovedeckých fakúlt

4.3.2 **Začínajúci podnikatelia v segmente vedeckých inovácií**

Druhou cieľovou skupinou sú začínajúce mladé kolektívy a tímy s nápadom na vedecký projekt, prípadne s už rozbehnutou vedeckou inováciou v začiatku podnikania. Tieto tímy hľadajú rôzne formy podpory pre ich produkt či službu. Môže ísť o potrebu mentoringu, potrebu právnej a finančnej pomoci či komplexného inkubačného a akceleračného

programu. Na platforme získajú prehľad o aktuálnych možnostiach ponúkaných vo svojom biotechnologickom odbore a získajú prehľad o tom, aké vedecké inovácie v našom regióne už máme a kto sú aktívni mentori v tejto sfére.

4.3.3 **Národné a medzinárodné inštitúcie podporujúce vedu**

Sekundárnou cieľovou skupinou sú samotné vzdelávacie inštitúcie, organizátori letných vedeckých a podnikateľských škôl, *hackathon* podujatí, či poskytovatelia podnikateľskej podpory ako sú inkubátory, akcelerátory aj finančné fondy. Práve tie dostanú priestor vytvoriť si na platforme účet a pridávať tieto príležitosti. Dostanú tak do pozornosti svoje aktivity ďalším možným kanálom k ich cieľovej skupine. Okrem menších lokálnych organizátorov prevažne vzdelávacích príležitostí môže ísť v neskoršej fáze aj o medzinárodné a národné inštitúcie ponúkajú mladým vedcom rôzne možnosti rozvoja v rámci grantových alebo vzdelávacích schém či tzv. *research* výziev. Tieto inštitúcie poskytujú poradenstvo vláde v oblasti výskumu a inovácií a spolupracujú s inými medzinárodnými vedeckými inštitúciami na realizácii výskumných projektov. Práve preto sú jedným zo stakeholderov aj oni. Na webovej stránke bude môcť používateľ nájsť vypísané možnosti pre kariérny rozvoj aj od týchto organizácii. Informácie budú čerpané z ich oficiálnych komunikačných kanálov a webov. Príklad týchto inštitúcií pre prototyp našej webovej stránky:

- **The European Institute of Innovation and Technology (EIT):** je medzinárodná inštitúcia, ktorá sa zameriava na podporu inovácií a technológií v Európe. Táto inštitúcia spája vedu, podnikateľský sektor a verejný sektor, aby podporila rozvoj inovatívnych riešení a projektov, ktoré prispievajú k hospodárskemu a sociálnemu rozvoju v Európe. EIT funguje tým, že poskytuje finančné a odborné zdroje, ako aj vzdelávacie programy a príležitosti pre výskumníkov, podnikateľov a inovátorov. V rámci svojej štruktúry má celé vetvy venujúce sa témam prepojenými s biotechnológiami ako EIT Food, EIT Health, EIT Raw Materials a pod.[37]
- **Slovak Academy of Sciences (SAV):** Slovak Academy of Sciences (SAV) je vedná inštitúcia zodpovedná za podporu výskumu a vedy v Slovenskej republike. SAV vedie, koordinuje a podporuje výskumné aktivity v oblastiach, ktoré sú pre Slovensko kľúčové, ako sú napríklad informatika, biotechnológie, environmentálne vedy a medicína. V prvej fáze by sme sa zamerali na zmapovanie príležitostí od UVP Biomed, ako súčasť SAV, ktorá sa zameriava na výskum a vývoj v oblasti biomedicíny. [38]

- **Czech Academy of Sciences (CAS):** Czech Academy of Sciences (CAS) je vedná inštitúcia zodpovedná za podporu výskumu a vedy v Českej republike. CAS vedie, koordinuje a podporuje výskumné aktivity v rôznych oblastiach, ako sú napríklad biotechnológie, informatika, humanitné vedy a environmentálne vedy. V rámci CAS by sme sa v prvej fáze zamerali na zmapovanie a zverejňovanie príležitostí o dotačných programoch pre aplikovaný výskum a transfer technológií. [39]
- I&I Prague - Cieľom I&I Prague je podporovať spoluprácu medzi akademickým svetom, priemyslom a investičným sektorom. Zameriava na prenos nových technológií do praxe, prioritne sa fokusuje na inovácie v oblasti objavovania liekov, diagnostiky, MedTechu a ďalších oblastí vedy o živote, ktoré vychádzajú z akademických inštitúcií. Ich tím podporuje vytváranie spin-off firiem a predaj licencií. [40]
- **CIVITTA**: je poradenská spoločnosť pre riadenie a digitálnu transformáciu, ktorá poskytuje širokú škálu služieb pre podniky a organizácie. Ich špecializáciou je rozvoj stratégie, transformácia organizácie, riadenie inovácií, trhový výskum, analýza dát a konzultácie v oblasti technológií. CIVITTA pôsobí v niekoľkých krajinách Európy , organizuje rôzne iniciatívy, vrátane inkubátorov a hackatónov, ktoré sa zameriavajú na témy zdravotníctva, zelených technológií, potravín a podobne. Tieto iniciatívy majú za cieľ podporiť inovácie, podnikateľské nápady a rozvoj v týchto oblastiach. Práve tým, že figuruje naprieč 16 európskymi krajinami sú jej programy vhodné pre mladých vedcov a odborníkov z rôznych národností aj nášho regiónu. [41]

Príležitosti ponúkané týmito organizáciami boli zmapované pre prototyp aplikácie a vložené do internej databázy platformy. No sú to len niektoré z organizácií, ktoré sú vďaka svojim aktivitám vhodnými adeptami pre prelinkovanie na novovznikajúcej platforme.

4.3.4 **Startup ekosystém**

V rámci startup ekosystému sa zameriavame na už etablované vedecko-podnikateľské subjekty s produktami alebo službami uvedenými na globálnom trhu a ktorých zakladatelia pochádzajú z niektorých z trhov CEE. V rámci platformy budú odprezentované základné informácie o ich zameraní s prelinkovaním na oficiálne weby spoločností či dané produkty. V neskoršej fáze vývoja môže pribudnúť možnosť pridania tipov na vedecké inovácie priamo od používateľov priamo do databázy. Zo začiatku však bude táto časť v manažmente admina.

4.4 Požiadavky na webovú aplikáciu

Funkcionálne a nefunkcionálne požiadavky boli v priebehu niekoľkých mesiacoch overené pri rozhovoroch s potenciálnymi užívateľmi či prispievateľmi budúcej platformy. Nižšie uvedené boli vybrané ako priorita pre takzvaný *Minimum Viable Product* - MVP prototyp aplikácie. MVP v preklade znamená minimálny životaschopný produkt je stratégia vývoja produktu, ktorá sa zameriava na vytvorenie a uvedenie základnej verzie produktu s minimálnym množstvom funkcií, ktoré sú nevyhnutné na uspokojenie potrieb a očakávaní zákazníkov. V budúcnosti sa predpokladá oveľa väčšia funkcionalita platformy.

4.4.1 Funkcionálne požiadavky

V tejto podkapitole sú uvedené funkcionálne požiadavky rozdelené podľa funkcionalít pre definované cieľové skupiny. Pre cieľovú skupinu návštevník - študenti, doktorandi, mladí vedeckí pracovníci, začínajúce biotechnologické startupy sú definované požiadavky v tabuľke 1. Pre organizácie poskytujúce príležitosti na získanie nových zručností alebo poskytujúce podporu začínajúcim vedecko-podnikateľským tímom sú požiadavky definované v tabuľke 2. Tabuľka 3 obsahuje požiadavky primárne na prácu s obsahom.

Jednotlivé funkcionálne požiadavky pre prvú skupinu sú:

Tabuľka 1. Funkcionálne požiadavky - návštevník webu

Id požiadavky	Popis požiadavky - neregistrovaný používateľ (Návštevník webu)
FP1	Návštevník aplikácie dokáže prezerať informácie ponúkané aplikáciou bez nutnosti registrácie.
FP2	Návštevník bude môcť prechádzať medzi podstránkami pomocou výberu prvku z navigačného panelu.
FP3	Verejný obsah webovej aplikácie bude štruktúrovaný do 3 hlavných obsahových blokov tvorených samostatnými sekciami: Skills, Startups, Network
FP4	Podkategória "**Skills**" bude poskytovať prehľad aktuálnych príležitostí praktického vzdelávania v oblasti vedy a inovácií.
FP5	Podkategória "**Startups support**" bude poskytovať zobrazenie aktuálne možnosti akceleračných a inkubačných programov v rámci CEE, a informovať o možnej podpore z rôznych aktuálnych finančných schémach pre mladých vedeckých inovátorov
FP6	Podkategória "**Network**" bude poskytovať zobrazenie databázy aktérov v oblasti vedeckých inovácií - expertov aj konkrétnych inovácií.

Tabuľka 2. Funkcionálne požiadavky - registrovaná organizácia

Id požiadavky	Popis požiadavky - organizácie
FP7	Webová aplikácia bude disponovať užívateľskými účtami s rolami pre organizácie a pre administrátora.
FP8	Webová aplikácia bude poskytovať rozhranie na prihlásenie organizácie alebo firmy
FP19	Webová aplikácia bude umožňovať registrovanej organizácii formulárom prihlásiť svoju vzdelávací program alebo podporu pre vedecké startupy do databázy portálu v kategórii Skills alebo Startups Support
FP10	Registrovaná organizácia bude vedieť aktualizovať pridané príležitosti a podporu
FP11	Registrovaná organizácia bude vedieť vymazať pridané príležitosti a podporu

Tabuľka 3. Funkcionálne požiadavky - obsah

Id požiadavky	Popis požiadavky - obsah
FP12	Listované informácie o príležitostiach v sekcii **“Skills”** budú s možnosťou filtrovania podľa kategórie na letné školy, stáže, Vedecko-podnikateľské workshopy a ďalšie.
FP13	Listované informácie o príležitostiach v sekcii **“Skills”** budú s možnosťou filtrovania podľa kategórie Biotechnológie, čiže zamerania.
FP14	Jednotlivé príležitosti v sekcii **“Skills”** budú zobrazovať dátum, organizátora, krajinu.
FP15	Informačná karta danej príležitosti v sekcii **“Skills”** bude riadne odkazovať na oficiálnu stránku danej príležitosti (workshop, hackathon, akadémiu...).
FP16	Listované informácie o príležitostiach v sekcii **“Startups support”** budú s možnosťou filtrovania podľa kategórie podpory na inkubátor, akcelerátor, mentoring, finančnú podporu či awards podujatie.
FP17	Listované informácie o príležitostiach v sekcii **“Startups support”** budú s možnosťou filtrovania podľa kategórie Biotechnológie, čiže zamerania.
FP18	Jednotlivé príležitosti v sekcii **“Startups support”** budú zobrazovať dátum, poskytovateľa podpory, krajinu.
FP19	Informačná karta danej príležitosti v sekcii **“Startups support”** bude riadne odkazovať na oficiálnu stránku danej príležitosti a jej prevádzkovateľa (inkubátor, VC fond...).
FP20	Databázu expertov v sekcii **“Network”** bude možné filtrovať podľa expertízy ako napríklad business development, finance, life science, clinical trial, chemistry, bioinformatics, legal a ďalšie.
FP21	Informačná karta experta v sekcii **“Network”** bude obsahovať verejne dostupné informácie o jeho expertíze a bude riadne odkazovať na jeho verejný LinkedIn profile.

FP22	Databázu inovácií vzniknutých v CEE v sekcii **"Network"** bude možné filtrovať na základe biotechnologickej kategórie
FP23	Informačná karta danej inovácie v sekcii **"Network"** bude obsahovať verejne dostupné informácie z oficiálnych webových stránok inovačných produktov a služieb a riadne prelinkovanie na ich web.

4.4.2 Nefunkcionálne požiadavky

Tabuľka 4 obsahuje definície nefunkcionálnych požiadaviek na webovú aplikáciu. Tieto požiadavky sa zameriavajú na rôzne aspekty aplikácie, ako je multiplatformovosť, responzívny dizajn, technologická kompatibilita, použitie komponentov na frontend strane, validácia vstupov, jazyková podpora a dostupnosť zdrojového kódu a dokumentácie.

Tabuľka 4. Nefunkcionálne požiadavky

Id požiadavky	Popis požiadavky
NP1	Aplikácia bude multiplatformová a zobrazená skrze webový prehliadač
NP2	Webová aplikácia bude mať responzívny dizajn a v prípade ďalšieho rozvoja bude možné prejsť na natívnu technológiu
NP3	Webová aplikácia bude vyvíjaná takou technológiou, ako disponuje väčšina stakeholderov vo svojich interných systémoch, aby bola prípadná integrácia pri rozširovaní jednoduchšia
NP4	Webová aplikácia bude využívať princíp komponent na frontend strane
NP5	Webová aplikácia bude mať implementovanú validáciu na overovanie správnosti zadaných vstupov na frontendovej aj backendovej strane
NP6	Výsledný prototyp bude v anglickej verzii s možnosťou rozšírenie ďalších jazykových mutácii
NP7	Samotný kód aplikácie bude sprístupnený v rámci Open Source licencie na platforme Github a bude sprístupnený na prípadnú kontribúciu od dobrovoľníkov.
NP8	Okrem samotného kódu aplikácie bude sprístupnená v rámci Open Source licencie aj dokumentácia vo formáte Swagger pre REST API.

4.5 Aktéri a prípady použitia

V nasledujúcich podkapitolách sú detailne opísaní aktéri, ktorí budú interagovať s aplikáciou a rovnako tak aj prípady použitia.

4.5.1 Aktéri

V aplikácii máme tri formy aktérov: návštevníka, prihlásenú organizáciu a administrátora.

Návštevník (neregistrovaný používateľ):

- Môže prezerať informácie ponúkané webovou aplikáciou bez nutnosti registrácie.
- Má možnosť prechádzať medzi podstránkami pomocou výberu prvku z navigačného panelu.
- Vidí verejný obsah aplikácie, ktorý je štruktúrovaný do 3 hlavných obsahových blokov: Skills, Startups, Network.
- V sekcii "Skills" môže vidieť aktuálne príležitosti praktického vzdelávania v oblasti vedy a inovácií.
- V sekcii "Startups support" môže zobraziť aktuálne možnosti akceleračných a inkubačných programov v rámci CEE a informácie o finančnej podpore pre vedeckých inovátorov.
- V sekcii "Network" môže vidieť databázu aktérov v oblasti vedeckých inovácií, vrátane expertov a konkrétnych inovácií.
- Má možnosť si vytvoriť účet ako organizácia.

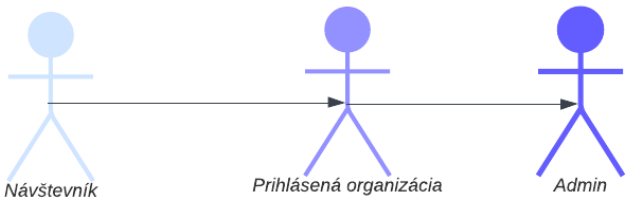
Prihlásená organizácia:

- Má vytvorený užívateľský účet so zvolenou rolou pre organizácie.
- Môže sa prihlásiť do webovej aplikácie pomocou rozhrania určeného pre organizácie a firmy.
- Má možnosť registrovať vzdelávací program alebo podporu pre vedecké startupy do databázy portálu v kategórii Skills alebo Startups prostredníctvom formulára.
- Môže aktualizovať pridané príležitosti alebo vymazať svoj záznam.
- Má prístup k funkcionalite a možnostiam priradeným organizáciám.

Admin:

- Má administrátorský účet v backend časti webovej aplikácie.
- Má prístup k rozhraniu Swagger, ktoré umožňuje správu užívateľských účtov a rolí.

- Má možnosť zmeniť rolu organizácie z ROLE_USER na ROLE_ADMIN cez rozhranie Swagger.
- Môže vyhľadať konkrétnu organizáciu v databáze účtov aj jednotlivé príležitosti a podpory.
- Môže vytvárať nové záznamy o expertoch a inováciách v databáze alebo cez Swagger, ktoré sú zobrazované v časti Network.



Obrázok 6: Hierarchia rolí aktérov

4.5.2 **Prípady použitia**

UC1: Prezeranie informácií bez registrácie

Popis: Návštevník aplikácie môže prezerať informácie ponúkané aplikáciou bez nutnosti registrácie.

Postup:

1. Návštevník otvorí webovú aplikáciu.
2. Webová aplikácia zobrazí verejný obsah, ako sú informácie o Skills, Startups a Network.
3. Návštevník má možnosť prehliadať obsah a získavať potrebné informácie bez nutnosti registrácie.

UC2: Prechádzanie medzi podstránkami

Popis: Návštevník môže prechádzať medzi podstránkami pomocou výberu prvku z navigačného panelu.

Postup:

1. Návštevník otvorí webovú aplikáciu.
2. Webová aplikácia zobrazí navigačný panel s prvkom umožňujúcim výber podstránok.
3. Návštevník vyberie požadovanú podstránku z navigačného panelu.
4. Webová aplikácia zobrazí vybranú podstránku s príslušným obsahom.

UC3: Prezeranie sekcií Skills, Startups, Network

Popis: Návštevník môže prehliadať obsah aplikácie štruktúrovaný do 3 hlavných obsahových blokov: Skills, Startups a Network.

Postup:

1. Návštevník otvorí webovú aplikáciu.
2. Webová aplikácia zobrazí hlavnú stránku so sekciou Skills, Startups a Network.
3. Návštevník vyberie jednu zo sekcií (Skills, Startups, Network).
4. Webová aplikácia zobrazí obsah príslušnej sekcie.

UC4: Zobrazenie príležitostí praktického vzdelávania v oblasti vedy a inovácií

Popis: Návštevník môže zobraziť aktuálne príležitosti praktického vzdelávania v oblasti vedy a inovácií v podkategórii "Skills".

Postup:

1. Návštevník otvorí webovú aplikáciu.
2. Webová aplikácia zobrazí hlavnú stránku s podkategóriou Skills.
3. Návštevník prejde do podkategórie Skills.
4. Webová aplikácia zobrazí aktuálne príležitosti praktického vzdelávania v oblasti vedy a inovácií.

UC5: Zobrazenie aktuálnych možností podpory v rámci CEE pre mladých vedeckých inovátorov

Popis: Používateľ môže zobraziť aktuálne možnosti akceleračných a inkubačných programov v rámci strednej a východnej Európy (CEE) pre mladých vedeckých inovátorov v podkategórii "Startups support".

Postup:

1. Používateľ otvorí webovú aplikáciu.
2. Webová aplikácia zobrazí hlavnú stránku.
3. Používateľ prejde do podkategórie "Startups support".
4. Webová aplikácia zobrazí aktuálne možnosti akceleračných a inkubačných programov v rámci CEE pre mladých vedeckých inovátorov.
5. Používateľ môže prezerať informácie o jednotlivých programoch, ich cieľoch, podmienkach a benefíciách.
6. Používateľ môže si vybrať program, ktorý ho zaujíma, a získať podrobnosti o tom, ako sa do neho zapojiť.

UC6: Zobrazenie databázy aktérov v oblasti vedeckých inovácií - expertov aj konkrétnych inovácií

Popis: Používateľ má možnosť zobraziť databázu aktérov v oblasti vedeckých inovácií, ktorá obsahuje informácie o expertoch aj konkrétnych inováciách v podkategórii "Network".

Postup:

1. Používateľ otvorí webovú aplikáciu.
2. Webová aplikácia zobrazí hlavnú stránku.
3. Používateľ prejde do podkategórie "Network".
4. Webová aplikácia zobrazí databázu aktérov v oblasti vedeckých inovácií.
5. Používateľ má možnosť vyhľadávať a prezerať informácie o expertoch v danej oblasti, vrátane ich skúseností, odbornosti a kontaktných údajov.
6. Používateľ môže tiež prezerať informácie o konkrétnych inováciách, ich popisoch, vlastnostiach a prípadne získať kontaktné údaje k tvorcom týchto inovácií.

UC7: Prihlásenie organizácie

Popis: Organizácia má možnosť prihlásiť sa do aplikácie cez rozhranie určené pre ňu.

Postup:

1. Organizácia otvorí webovú aplikáciu.
2. Webová aplikácia zobrazí rozhranie na prihlásenie organizácií alebo firiem.
3. Organizácia vyplní požadované informácie, ako meno, e-mailovú adresu a prípadne heslo.
4. Organizácia odošle prihlasovacie údaje.
5. Webová aplikácia overí prihlasovacie údaje a umožní organizácii prístup k príslušným funkciám a možnostiam v aplikácii.

UC8: Správa užívateľských účtov s rolami pre organizácie a administrátora

Popis: Administrátor má možnosť spravovať užívateľské účty s pridelenými rolami pre organizácie a administrátora v backend časti webovej aplikácie. Pri registrácii nového účtu pre organizáciu je automaticky pridelená rola ROLE_USER. Rozhranie Swagger umožňuje administrátorovi webovej stránky meniť role priradené používateľom a priradiť im rolu ROLE_ADMIN.

Postup:

1. Organizácia (používateľ) otvorí frontend časť webovej aplikácie a vykoná registráciu svojho účtu.
2. Backend časť aplikácie spracuje požiadavku na registráciu a vytvorí nový užívateľský účet pre organizáciu.
3. Pri vytváraní nového účtu je organizácii automaticky pridelená rola ROLE_USER.

UC9: Zmena role organizácie administrátorom cez rozhranie Swagger

Popis: Administrátor má možnosť zmeniť rolu organizácie z ROLE_USER na ROLE_ADMIN cez rozhranie Swagger v backend časti webovej aplikácie.

Postup:

1. Administrátor otvorí backend časť webovej aplikácie a prihlási sa do svojho administrátorského účtu.
2. Administrátor má prístup k rozhraniu Swagger, ktoré umožňuje správu užívateľských účtov a rolí.
3. Administrátor vyhľadá konkrétnu organizáciu v databáze účtov.
4. Administrátor prechádza na stránku úpravy používateľského účtu organizácie.
5. V rozhraní Swagger administrátor vidí aktuálnu priradenú rolu organizácii (ROLE_USER).
6. Administrátor môže zmeniť rolu organizácie kliknutím na možnosť zmeny role.
7. Administrátor vyberie rolu ROLE_ADMIN a uloží zmeny.
8. Backend časť aplikácie aktualizuje údaje v databáze a priradí organizácii novú rolu ROLE_ADMIN.

UC10: Registrácia vzdelávacieho programu alebo podpory pre vedecké startupy organizáciou

Popis: Registrovaná organizácia má možnosť pomocou formulára zaregistrovať svoj vzdelávací program alebo podporu pre vedecké startupy do databázy portálu v kategórii Skills alebo Startups.

Postup:

1. Registrovaná organizácia sa prihlási do webovej aplikácie.
2. Webová aplikácia zobrazí rozhranie pre správu príležitostí v kategóriách Skills alebo Startups.
3. Organizácia vyplní požadované informácie o vzdelávacom programe alebo podpore pre vedecké startupy v rámci formulára.
4. Organizácia odošle formulár s informáciami.
5. Webová aplikácia spracuje a uloží informácie o vzdelávacom programe alebo podpore pre vedecké startupy do databázy portálu

UC11: Aktualizácia a vymazanie záznamu organizáciou

Popis: Registrovaná organizácia má možnosť aktualizovať pridané príležitosti a prípadne vymazať svoj záznam v aplikácii.

Postup:

1. Organizácia sa prihlási do webovej aplikácie svojím registrovaným účtom.
2. Webová aplikácia zobrazí príležitosti, ktoré organizácia pridala do systému.
3. Organizácia vyberie príležitosť, ktorú chce aktualizovať alebo vymazať.

Ak chce organizácia aktualizovať príležitosť:

4 a. Webová aplikácia zobrazí formulár s existujúcimi údajmi príležitosti.

4 b. Organizácia upraví požadované informácie v formulári.

4 c. Organizácia odošle formulár s aktualizovanými údajmi.

4 d. Webová aplikácia spracuje požiadavku na aktualizáciu a aktualizuje údaje príležitosti v databáze.

Ak chce organizácia vymazať príležitosť:

4 e. Organizácia klikne na možnosť vymazať

4f. Webová aplikácia odstráni príležitosť zo systému a aktualizuje databázu.

UC12: Filtrovanie príležitostí v sekcii "Skills"

Popis: Používateľ má možnosť filtrovať a zobraziť informácie o príležitostiach v sekcii "Skills" podľa rôznych kategórií, napr. letných škôl, stáží alebo podľa konkrétnych biotechnológií.

Postup:

1. Používateľ otvorí webovú aplikáciu.
2. Webová aplikácia zobrazí hlavnú stránku.
3. Používateľ prejde do sekcie "Skills".
4. V sekcií "Skills" webová aplikácia zobrazí zoznam príležitostí.

5. Používateľ použije možnosť filtrovania príležitostí podľa biotechnologické kategórie a/ alebo podľa typu príležitostí ako letné školy, stáže, vedecko-podnikateľské workshopy a ďalšie.
6. Používateľ vyberie jednu alebo viac kategórií.
7. Webová aplikácia aktualizuje zoznam príležitostí a zobrazí iba tie, ktoré spĺňajú zvolené kategórie.
8. Používateľ môže prezerať podrobnosti o jednotlivých príležitostiach v zobrazenom zozname.

UC13: Zobrazenie informácií o príležitostiach v sekcii "Skills"

Popis: Používateľ má možnosť prezerať informácie o jednotlivých príležitostiach v sekcii "Skills" vrátane dátumu, organizátora, krajiny, pre ktorú príležitosť platí a tiež má možnosť kliknúť na informačnú kartu danej príležitosti a byť presmerovaný na oficiálnu stránku tejto príležitosti (workshop, hackathon, akadémia...).

Postup:

1. Používateľ otvorí webovú aplikáciu.
2. Webová aplikácia zobrazí hlavnú stránku.
3. Používateľ prejde do sekcie "Skills".
4. V sekcií "Skills" webová aplikácia zobrazí zoznam príležitostí.
5. Webová aplikácia zobrazí detailné informácie o každej\ príležitosti vrátane dátumu, organizátora a krajiny, pre ktorú príležitosť platí.
6. Na informačnej karte danej príležitosti bude poskytnutý odkaz.
7. Používateľ klikne na odkaz a webová aplikácia presmeruje ho na oficiálnu stránku danej príležitosti.

UC14: Filtrovanie podpory v sekcii "Startups support"

Popis: Používateľ má možnosť filtrovať a zobraziť informácie o príležitostiach v sekcii "Startups support" podľa rôznych typov podpory ako sú inkubátor, akcelerátor, mentoring, finančná podpora awards program alebo podľa typu biotechnológie.

Postup:

1. Používateľ otvorí webovú aplikáciu.
2. Webová aplikácia zobrazí hlavnú stránku.
3. Používateľ prejde do sekcie "Startups support".
4. V sekcií "Startups support" webová aplikácia zobrazí zoznam príležitostí.
5. Používateľ použije možnosť filtrovania príležitostí podľa kategórie podpory ako sú inkubátor, akcelerátor, mentoring, finančná podpora awards program a/ alebo podľa biotechnologické kategórie.
6. Používateľ vyberie jednu alebo viac kategórií.
7. Webová aplikácia aktualizuje zoznam príležitostí a zobrazí iba tie, ktoré spĺňajú zvolené kategórie.
8. Používateľ môže prezerať podrobnosti o jednotlivých príležitostiach v zobrazenom zozname.

UC15: Zobrazenie informácií o príležitostiach v sekcii "Startups support"

Popis: Používateľ má možnosť prezerať informácie o jednotlivých príležitostiach v sekcii "Startups support" vrátane dátumu, poskytovateľa podpory, krajiny, pre ktorú podpora platí, a tiež má možnosť kliknúť na informačnú kartu danej príležitosti a byť presmerovaný na oficiálnu stránku danej príležitosti a prevádzkovateľa (inkubátor, VC fond...).

Postup:

1. Používateľ otvorí webovú aplikáciu.
2. Webová aplikácia zobrazí hlavnú stránku.
3. Používateľ prejde do sekcie "Startups support".
4. V sekcií "Startups support" webová aplikácia zobrazí zoznam príležitostí.
5. Webová aplikácia zobrazí detailné informácie o každej príležitosti vrátane dátumu, poskytovateľa podpory a krajiny, pre ktorú podpora platí.
6. Na informačnej karte danej príležitosti bude poskytnutý odkaz na oficiálnu stránku danej príležitosti a prevádzkovateľa.

7. Používateľ klikne na odkaz a webová aplikácia presmeruje ho na oficiálnu stránku danej príležitosti a prevádzkovateľa (inkubátor, VC fond...).

UC16: Filtrovanie expertov v sekcii "Network" podľa expertízy

Popis: Používateľ má možnosť filtrovať a zobraziť databázu expertov v sekcii "Network" podľa rôznych expertíz, ako je business development, finance, life science, clinical trial, chemistry, bioinformatics, legal a ďalšie.

Postup:

1. Používateľ otvorí webovú aplikáciu.
2. Webová aplikácia zobrazí hlavnú stránku.
3. Používateľ prejde do sekcie "Network".
4. V sekcií "Network" webová aplikácia zobrazí databázu expertov.
5. Používateľ použije možnosť filtrovania expertov podľa expertízy.
6. Používateľ vyberie jednu alebo viac expertíz, ako business development, finance, life science, clinical trial, chemistry, bioinformatics, legal alebo ďalšie.
7. Webová aplikácia aktualizuje zobrazenie databázy expertov a zobrazí iba tých, ktorí spĺňajú zvolené expertízy.
8. Používateľ môže prezerať informácie o jednotlivých expertoch v zobrazenom zozname a prechádzať medzi listovanými informáciami pomocou stránkovania (pagináciou)

UC17: Zobrazenie informácií o expertovi v sekcii "Network

Popis: Používateľ má možnosť prezerať informácie o jednotlivých expertoch v sekcii "Network", vrátane verejne dostupných informácií o ich expertíze, a tiež má možnosť kliknúť na informačnú kartu daného experta a byť presmerovaný na jeho verejný LinkedIn profil.

Postup:

1. Používateľ otvorí webovú aplikáciu.
2. Webová aplikácia zobrazí hlavnú stránku.

3. Používateľ prejde do sekcie "Network".
4. V sekcií "Network" webová aplikácia zobrazí databázu expertov.
5. Webová aplikácia zobrazí detailné informácie o každom expertovi, vrátane jeho expertízy.
6. Na informačnej karte daného experta bude poskytnutý odkaz na jeho verejný LinkedIn profil.
7. Používateľ klikne na odkaz a webová aplikácia presmeruje ho na verejný LinkedIn profil daného experta. To mu umožní ho napriamo kontaktovať.

UC18: Filtrovanie inovácií v sekcii "Network" na základe biotechnologickej kategórie

Popis: Používateľ má možnosť filtrovať a zobraziť databázu inovácií v sekcii "Network" na základe

biotechnologickej kategórie.

Postup:

1. Používateľ otvorí webovú aplikáciu.
2. Webová aplikácia zobrazí hlavnú stránku.
3. Používateľ prejde do sekcie "Network".
4. V sekcií "Network" webová aplikácia zobrazí databázu inovácií.
5. Používateľ použije možnosť filtrovania inovácií na základe biotechnologickej kategórie.
6. Používateľ vyberie kategóriu biotechnológie.
7. Webová aplikácia aktualizuje zobrazenie databázy inovácií a zobrazí iba tie, ktoré patria do kategórie biotechnológie.
8. Používateľ môže prezerať informácie o jednotlivých inováciách v zobrazenom zozname.

UC19: Zobrazenie informácií o inovácii v sekcii "Network"

Popis: Používateľ má možnosť prezerať informácie o jednotlivých inováciách v sekcii "Network", vrátane verejne dostupných informácií z oficiálnych webových stránok inovačných produktov a služieb, a tiež má možnosť kliknúť na informačnú kartu danej inovácie a byť presmerovaný na jej oficiálnu stránku.

Postup:

1. Používateľ otvorí webovú aplikáciu.
2. Webová aplikácia zobrazí hlavnú stránku.
3. Používateľ prejde do sekcie "Network".
4. V sekcií "Network" webová aplikácia zobrazí databázu inovácií.
5. Používateľ vyberie konkrétnu inováciu, ktorú chce prezerať.
6. Webová aplikácia zobrazí detailné informácie o vybranej inovácii vrátane informácií z oficiálnych webových stránok inovačných produktov a služieb.
7. Na informačnej karte danej inovácie bude poskytnutý odkaz na oficiálnu stránku inovácie.
8. Používateľ klikne na odkaz a webová aplikácia presmeruje ho na oficiálnu stránku danej inovácie.

4.5.3 **Pokrytie požiadaviek**

Na overenie, či navrhnuté prípady použitia pre daný systém pokrývajú definované funkcionálne požiadavky sa používa Matica pokrytia požiadaviek, tzv. *Traceability Matrix*. V tabuľke 5 sú vyznačené pokrytia pre prototyp aplikácie *How to do biotech.*

Tabuľka 5. Traceability Matrix - pokrytie požiadaviek prípadmi použitia

Prípad použitia	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8	UC9	UC10	UC11	UC12	UC13	UC14	UC15	UC16	UC17	UC18	UC19
Id požiadavky																			
FP1	X																		
FP2		X																	
FP3			X																
FP4				X															
FP5					X														
FP6						X													
FP7								X	X										
FP8							X												
FP9										X									
FP10											X								
FP11											X								
FP12												X							
FP13												X							
FP14													X						
FP15													X						
FP16														X					
FP17														X					
FP18															X				
FP19															X				
FP20																X			
FP21																	X		
FP22																		X	
FP23																			X

4.6 Informačné zdroje

Pod pojmom informačné zdroje rozumieme zdroje dát, ktoré budú použité v databázach a ako statický obsah ako copywriting a vizuálny obsah webovej aplikácie.

Dáta

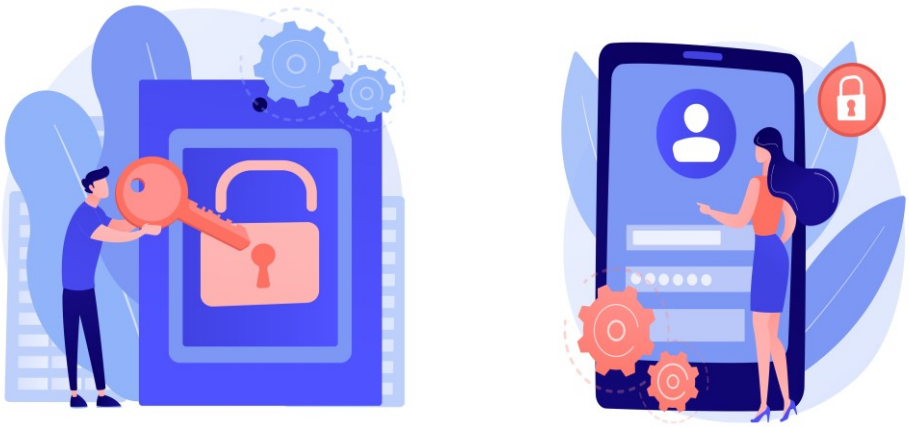
Pre účel bakalárskej práce ale aj následného prípadného spustenia aplikácie bola vytvorená databáza o vzdelávacích príležitostiach, o podpore pre vedecko-podnikateľské subjekty, o aktívnych expertoch, ktorí pôsobia vo vzdelávacích, inkubačných alebo akceleračných programoch, prípadne sú zástupcami inovačných organizácii, ktoré takúto podporu ponúkajú a o zástupcoch VC fondov. Časť databázovej schémy je venovaná aj existujúcim inovatívnym produktom a službám v oblasti biotechnológií.

Všetky dáta boli zozbierané autorkou práce, z oficiálnych zdrojov inštitúcií a firiem. V prípade expertov ide o informácie z ich verejných LinkedIn Profilov.[42] Všetky dáta sú riadne prelinkované na zdroje a uvádzajú poskytovateľa alebo organizátora daných príležitostí.

Texty a vizuálny obsah

Texty webovej aplikácie sú vytvorené autorkou bakalárskej práce. Rovnako ako názov platformy *How To Do Biotech.*

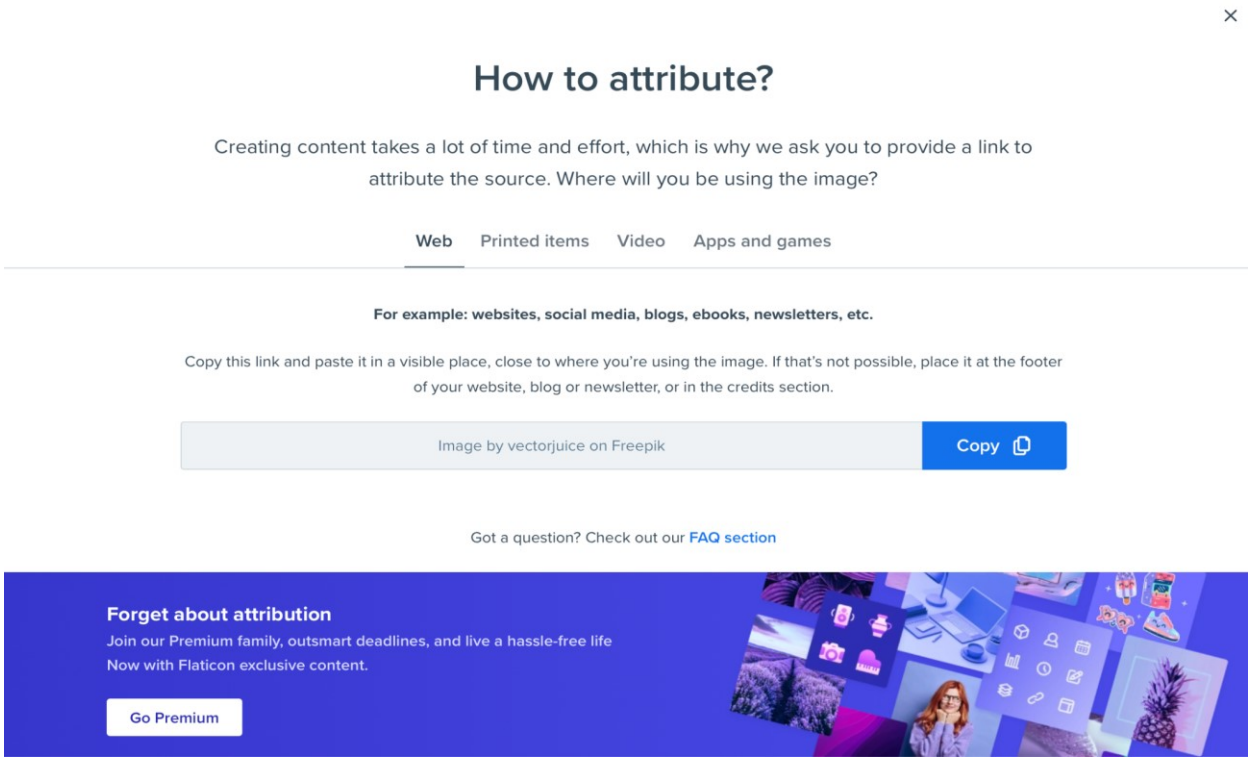
Vizuálne prvky ako ilustrácie sú čerpané z platformy Freepik [43] a sú vo forme free licence, čo umožňuje ich verejné použitie. [44]



Obrázok 7: Ukážka použitých vizualizácií pre registračný formulár [43]

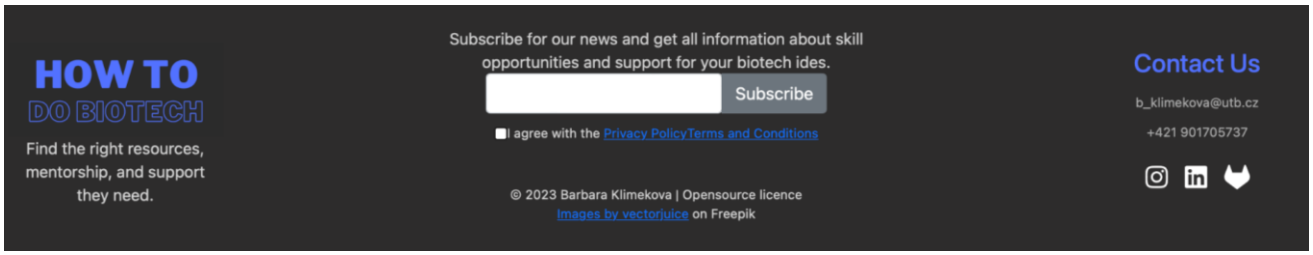
Obrázok 8: Ukážka použitých vizualizácií pre hlavné bannery na sekciách [43]

Ikony používané na webovej stránke sú súčasťou balíčka React Bootstrap s Material Design princípmi - mdbootstrap. [21] Úplný zoznam použitých ilustrácií a prvkov je uvedený v prílohe aj s konkrétnymi linkami na zdroj a licenciu.



Obrázok 9: Usmernenie ako správne odkazovať na zdroj ilustrácie s licenciou [43]

Okrem toho je komunikovaný zdroj vizuálnych prvkov aj v časti Footer na navrhovanej webovej aplikácii, ako možno vidieť na obrázku 10.



Obrázok 10: Footer aplikácie *How To Do Biotech*

Aplikácia je navrhnutá tak, aby spĺňala možnosť pre registrované organizácie publikovať ďalší obsah - v pilotnej fáze len do sekcií “Skills“ alebo “Startup Support“ pod svojim účtom,

ktorý bude automatický zverejnený ako poskytovateľ podpory alebo organizátor danej príležitosti. Sekcia “Network“ a teda databáza odborníkov - expertov a inovatívnych produktov a služieb nateraz ostáva v správe autorky stránky, aby sa zabezpečila relevantnosť a overenosť dát. V budúcnosti je možnosť dáta implementovať aj cez externé REST API endpointy tretích strán.

5 NÁVRH APLIKÁCIE

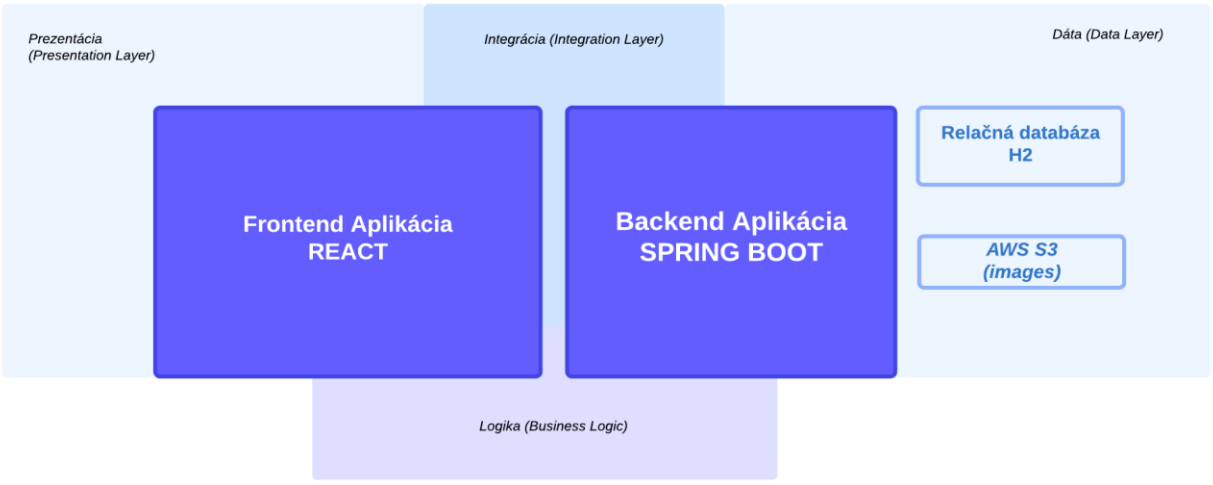
V tejto kapitole je upresnené, aké návrhové zdroje sa pri vývoji využili. Na úvod popisuje architektúru, v ďalších podkapitolách sa venuje aj vizuálnemu rozvrhnutiu. Architektúra webovej stránky zahŕňa niekoľko hlavných častí, ktoré spolupracujú na poskytnutí funkcionality a zabezpečení užívateľského zážitku. Návrh aplikácie je zobrazený na obrázku 11.

Prezentácia (Presentation Layer): Táto časť je zodpovedná za zobrazenie a interakciu so stránkou pre používateľa. Používa sa tu HTML, CSS a JavaScript (napríklad s použitím knižnice ako React) na vytvorenie užívateľského rozhrania a manipuláciu s obsahom. [45]

Logika (Business Logic): Táto časť zahŕňa rôzne funkcie a procesy, ktoré riadia a spracúvajú dáta a logiku aplikácie. Môže to zahŕňať overovanie a spracovanie vstupov, výpočty, validácie a manipuláciu s dátami. Je dôležité si uvedomiť, že dôležitá časť logiky by mala byť implementovaná na backendovej strane, aby sa zabezpečila bezpečnosť a konzistencia dát. Frontendová logika slúži na zlepšenie užívateľského zážitku, ale backend je zodpovedný za správne a spoľahlivé spracovanie dát a podnikateľskej logiky. [45]

Dáta (Data Layer): Táto časť zahŕňa rôzne zdroje a databázy, kde sú uložené a spracovávané dáta. Môže to byť relačná databáza, NoSQL databáza, súborový systém alebo externé API. Táto časť zabezpečuje prístup a manipuláciu s dátami potrebnými pre aplikáciu. [45]

Integrácia (Integration Layer): Táto časť umožňuje komunikáciu medzi rôznymi časťami aplikácie, ako je napríklad prenos dát medzi prezentáciou a logikou, integrácia s externými systémami alebo správa komunikácie s backendom pomocou API.[45]



Obrázok 11: Vrstvy a aplikácia *How To Do biotech*

5.1 Architektúra aplikácie How To Do biotech

Frontendová aplikácia:

React, Redux a Axios sú populárne technológie a knižnice, ktoré sa často používajú spoločne pri vývoji webových aplikácií.

React je opísaný v úvodných kapitolách práce. Redux je knižnica pre riadenie stavu aplikácie v Reacte. Sleduje princípy jednosmerného dátového toku a poskytuje centrálne úložisko stavu – tzv. *Store* pre celú aplikáciu. Redux umožňuje jednoduchú správu a aktualizáciu stavu pomocou akcií a reduktorov. Axios je knižnica ktorá poskytuje flexibilný spôsob na odosielanie HTTP požiadaviek z frontendu. Umožňuje vývojárom jednoduché spracovanie požiadaviek na server, vrátane získavania dát, odosielania dát a manipulácie s odpoveďami.[3]

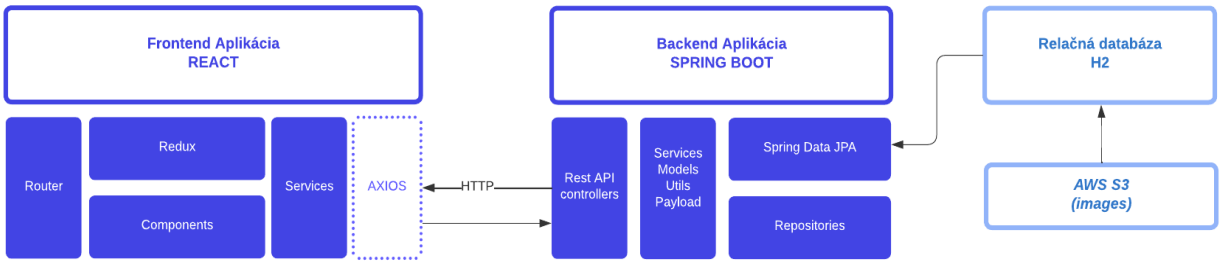
Architektúra Redux v kombinácii s Reactom je založená na princípoch jednosmerného dátového toku a poskytuje efektívny spôsob riadenia stavu aplikácie. Táto architektúra sa skladá z úložiska, akcií, reduktorov a komponentov.[3]

Úložisko je centrálne úložisko stavu aplikácie, ktoré obsahuje všetky dáta. Akcie sú JavaScriptové objekty, ktoré opisujú zmeny stavu a vyvolávajú sa v rôznych častiach aplikácie. Reduktory sú funkcie, ktoré spracúvajú akcie a aktualizujú stav v úložisku. Komponenty sú zodpovedné za vykresľovanie užívateľského rozhrania a môžu byť pripojené k Redux úložisku pomocou prepájania, čím získajú prístup k stavu a akciám.

Backendová časť aplikácie:

Pri kombinácii Spring Boot, Axios, JPA (Java Persistence API) a relačnej databázy, ako aj AWS S3, sa môže dosiahnuť výkonný a flexibilný systém pre správu dát a ukladanie súborov. Spring Boot poskytuje podporu pre RESTful API, spracovanie požiadaviek a vykonávanie obchodnej logiky. S JPA môže Spring Boot efektívne pracovať s relačnými databázami, umožňujúc rýchle a jednoduché vytváranie, čítanie, aktualizovanie a mazanie dát.

AWS S3 je služba poskytovaná Amazon Web Services, ktorá poskytuje dátové úložisko vo forme objektových "buckets". S3 buckets sú ideálne pre ukladanie a správu súborov a dát v cloude. V kombinácii s Spring Bootom a Axiosom sa dá ľahko integrovať komunikáciu a zabezpečí sa tak bezpečné ukladanie a správa súborov.



Obrázok 12: Architektúra aplikácie *How To Do biotech*

5.2 REST API Model

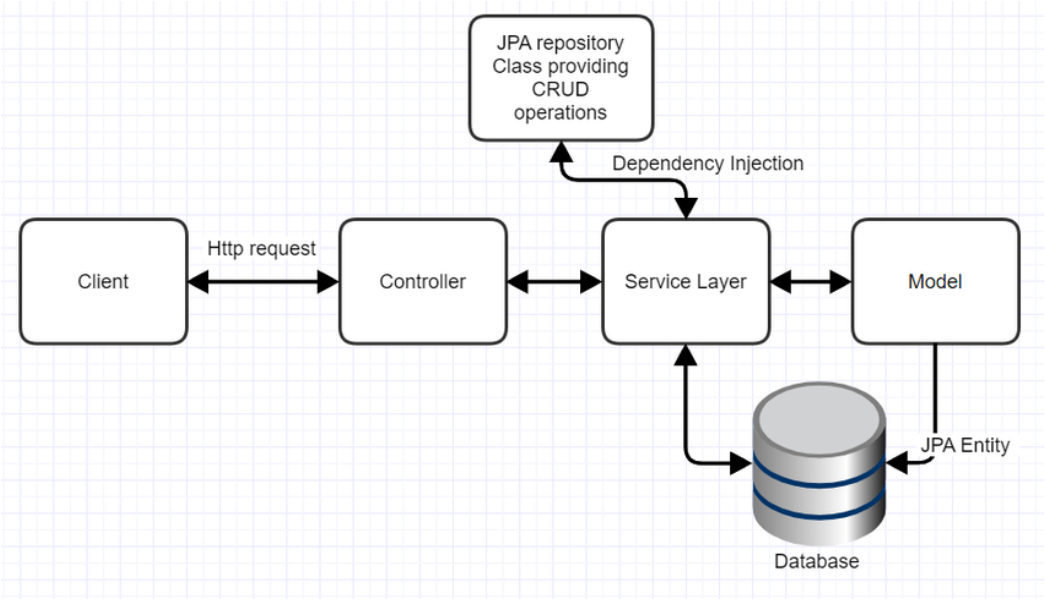
REST API (Representational State Transfer Application Programming Interface) má typicky nasledujúce vrstvy: [25]

Controller vrstva: Controller vrstva je zodpovedná za spracovanie prichádzajúcich HTTP požiadaviek a riadenie toku dát medzi klientom a serverom. Táto vrstva definuje endpointy a ich spracovanie. Spracováva vstupné požiadavky od klienta, vykonáva validáciu a autentifikáciu, komunikuje s ostatnými vrstvami a odosiela odpovede späť klientovi.

Service vrstva: Service vrstva obsahuje logiku podnikového procesu alebo operácie, ktoré sú špecifické pre danú doménu. Táto vrstva spracováva dáta prijaté od kontrolérov, vykonáva zložitejšie operácie, ako napríklad kombinovanie alebo transformáciu dát, a komunikuje s Repository vrstvou na prístup k dátam.

Repository vrstva: sa zaoberá prístupom k dátam a interakciou s databázou alebo iným úložiskom dát. Poskytuje metódy na vytváranie, čítanie, aktualizáciu a mazanie dát. Repository vrstva abstrahuje implementáciu úložiska a poskytuje jednotné rozhranie pre prístup k dátam pre Service vrstvu.

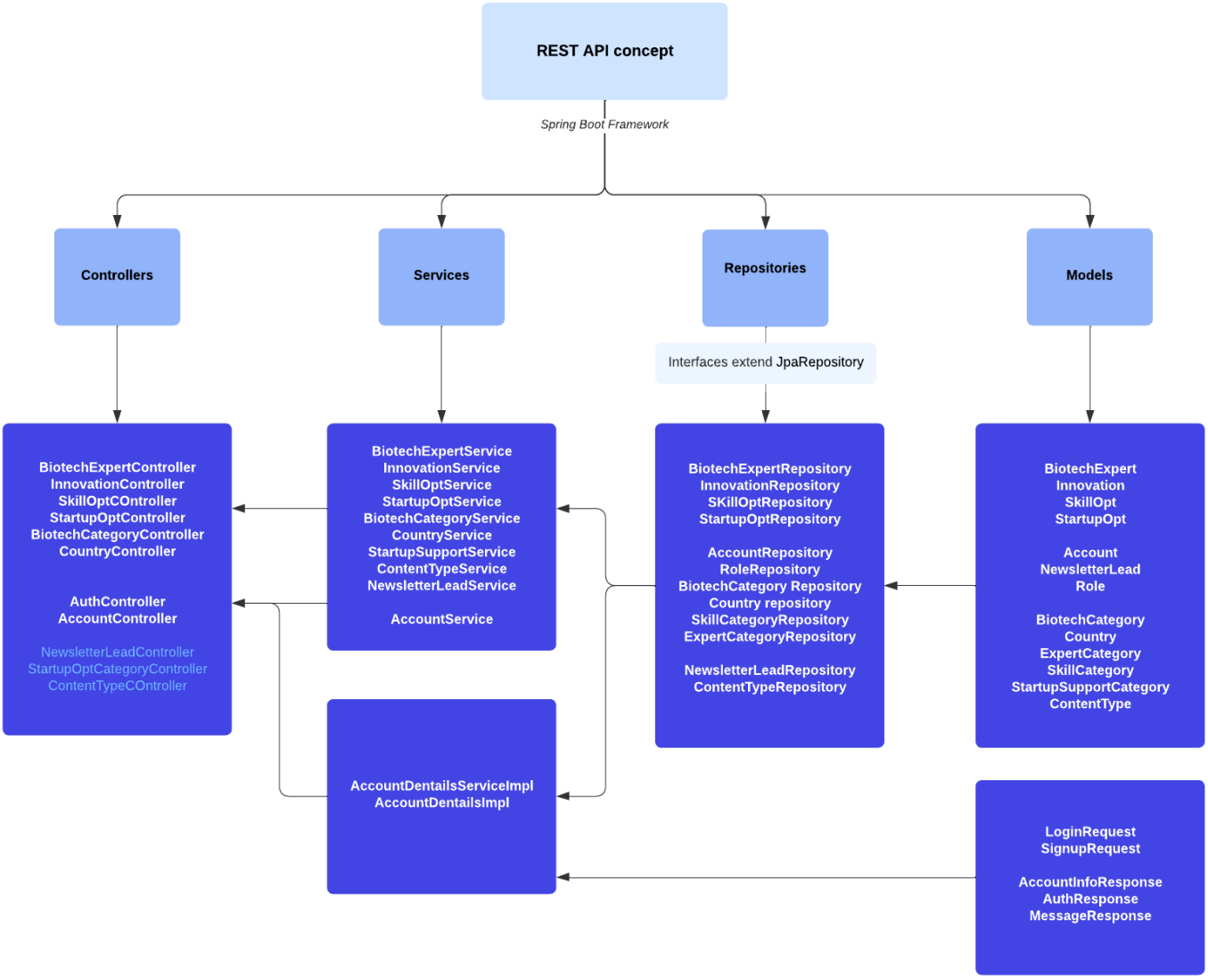
Databázová vrstva: Táto vrstva predstavuje skutočné úložisko dát, ako napríklad relačnú databázu alebo iný druh dátového úložiska. Je zodpovedná za ukladanie a získavanie dát, vykonávanie dotazov a správu transakcií.



Obrázok 13: Architektonický tok aplikácií Spring Boot [46]

Tieto vrstvy spolupracujú a komunikujú medzi sebou na správne spracovanie požiadaviek a poskytnutie odpovedí klientovi. Práve túto komunikáciu medzi vrstvami uľahčuje framework Spring Boot. Na obrázku 13 vidíme architektonický tok Spring Boot aplikácií.

"Architektonický tok aplikácií Spring Boot Spring Boot využíva všetky funkcie Springu, ako je Spring MVC, Spring Data a JPA. Všeobecná aplikácia Spring Boot sa skladá z kontrolóra, ktorý obsluhuje HTTP požiadavky klientov. Tento controller následne komunikuje s vrstvou služieb, ktorá spracováva požiadavku na úpravu modelu a databázy pomocou JPA repozitára a závislosti. Ak sa nevyskytne žiadna chyba, používateľovi sa vráti zobrazenie stránky." [46]



Obrázok 14: REST API architektúra pre *How To Do Biotech* aplikáciu

Architektúra REST API v Spring Boot frameworku pre účely webovej aplikácie *How To Do Biotech* je založená na tomto klasickom princípe Models => Repositories => Services => Controllers. Na obrázku 14 je znázornená schéma všetkých častí. Tento prístup pomáha oddeliť jednotlivé vrstvy aplikácie a udržiavať čistú a modulárnu štruktúru.

5.3 Atomic Design prístup

Atomic Design je dizajnový systém a metodika, ktorá sa často používa pri tvorbe užívateľského rozhrania v React aplikáciách. Táto metóda sa zameriava na dekompozíciu komponentov na menšie, znovu použiteľné a jednoducho upravovateľné časti.

V Atomic Design sa komponenty delia do piatich základných úrovní [47], ktoré sa navzájom dopĺňajú. Princípy sú nastavené nasledovne: [47]

Atómy (Atoms): Atómy sú najmenšie stavebné bloky, ktoré tvoria komponenty. Patria sem jednoduché elementy ako tlačidlá, textové polia, obrázky alebo iné jednoduché vizuálne prvky. Atómy by mali byť čo najviac nezávislé a opakovateľné.

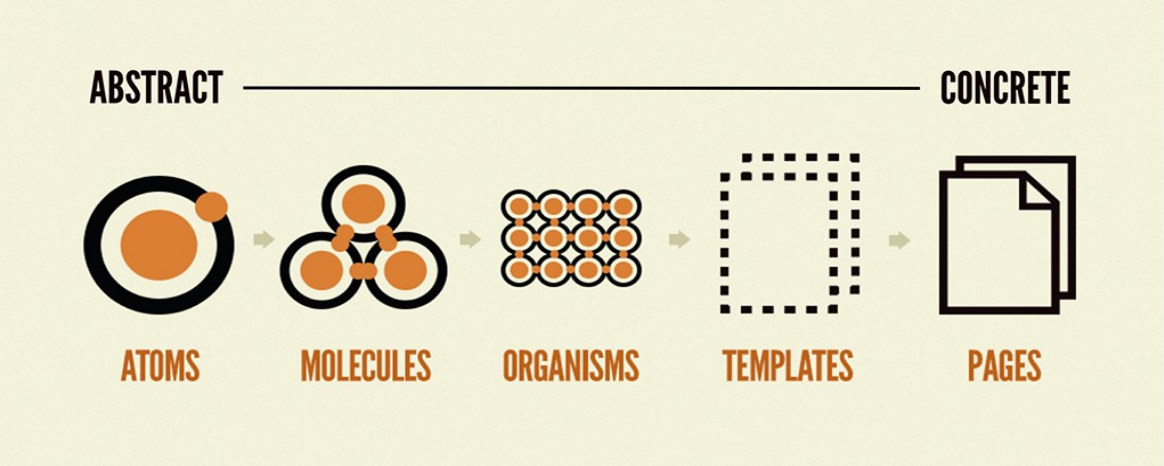
Molekuly (Molecules): Molekuly sú kombináciou atómov a zastupujú viacero prvkov, ktoré spolu pracujú a tvoria jednu funkčnú jednotku. Molekuly môžu obsahovať napríklad formuláre, karty alebo hlavičky.

Organizmy (Organisms): Organizmy sú väčšie a zložitejšie komponenty, ktoré zahŕňajú skupiny molekúl a atómov. Organizmy tvoria väčšie časti rozhrania, ako napríklad navigácie, zoznamy, stránky alebo sekcie.

Šablóny (Templates): Šablóny definujú základnú štruktúru a rozloženie komponentov v konkrétnych častiach aplikácie. Tieto šablóny môžu byť použité pre viacero stránok alebo sekcii, aby sa zachovala jednotná vizuálna a štrukturálna konzistencia.

Stránky (Pages): Stránky sú konkrétnymi inštanciami šablón, kde sú naplnené konkrétnymi obsahmi a dátami. Toto sú finálne stránky, ktoré už používatelia vidia a s ktorými interagujú.

Využitie Atomic Design v React aplikáciách umožňuje efektívne rozdelenie komponentov na menšie a znovu použiteľné časti, čo zlepšuje spravovateľnosť a opätovné použiteľnosť kódu. Taktiež umožňuje konzistentnú vizuálnu identitu a jednoduchšiu údržbu aplikácie.



Obrázok 15: Atomic Design princípy [47]

5.4 Wireframes aplikácie

Wireframe, je náčrt alebo abstraktná reprezentácia rozloženia a usporiadania prvkov na obrazovke aplikácie. Wireframe sa často používa pri prototypovaní aplikácií, pretože poskytuje jednoduchú a zrozumiteľnú vizuálnu predstavu o štruktúre a organizácii obsahu.

Pri návrhu aplikácie je dôležité vytvoriť wireframe, pretože to umožňuje navrhovať a overovať usporiadanie prvkov, navigáciu medzi obrazovkami a tok používateľskej interakcie. Pomáha to identifikovať a riešiť problémy s použiteľnosťou a navigáciou v skorých fázach vývoja, čo vedie k lepšiemu a efektívnejšiemu dizajnu aplikácie.

Pre účely bakalárskej práce a návrhu aplikácie *How To Do Biotech* boli vytvorené wireframe obrazovky v nástroji *Figma*. [48] Tieto wireframe obrazovky sú vytvorené v nízkom detaile, čo znamená, že neobsahujú konkrétne vizuálne prvky, ako sú farby, obrázky a podobne. Ich hlavným zameraním je zobraziť usporiadanie a štruktúru obsahu aplikácie, navigáciu medzi obrazovkami a interakčné prvky, ako sú tlačidlá a formuláre. Vytvorenie tzv. *low fidelity* wireframe umožňuje rýchle a flexibilné testovanie a iteráciu dizajnu, predtým než sa začne s detailnejším vizuálnym dizajnom aplikácie. [30] Tieto obrazovky boli diskutované s niekoľkými potenciálnymi užívateľmi aplikácie a zahrňujú ich preferencie na zoskupenie obsahu do jednotlivých sekcií Skills, Startup Support a Network. Sekcie Skills a Startup Support sa svojim rozložením veľmi nelíšia.

Navigácia: Umožňuje používateľom prechádzať medzi rôznymi časťami stránky. Pre návštevníka (neregistrovaného používateľa) zobrazuje relevantné informácie a možnosť

prihlásenia k odberu noviniek. Pre registrovanú organizáciu poskytuje prístup a správu ich pridaných príležitostí a podpory. Obsah navigácie sa dynamicky mení podľa role používateľa.

Carousel Banner: V hornej časti je základný vizuálny motív a všeobecné informácie o sekcie. Ide o Carousel prvok, ktorý je možný v prípade potreby rozšíriť o aktuálne kampane na konkrétne podujatia a príležitosti. Carousel je interaktívny webový prvok, ktorý umožňuje prechádzať medzi viacerými obsahovými položkami, ako sú obrázky alebo obsahové panely, pomocou pohybu vpravo a vľavo.

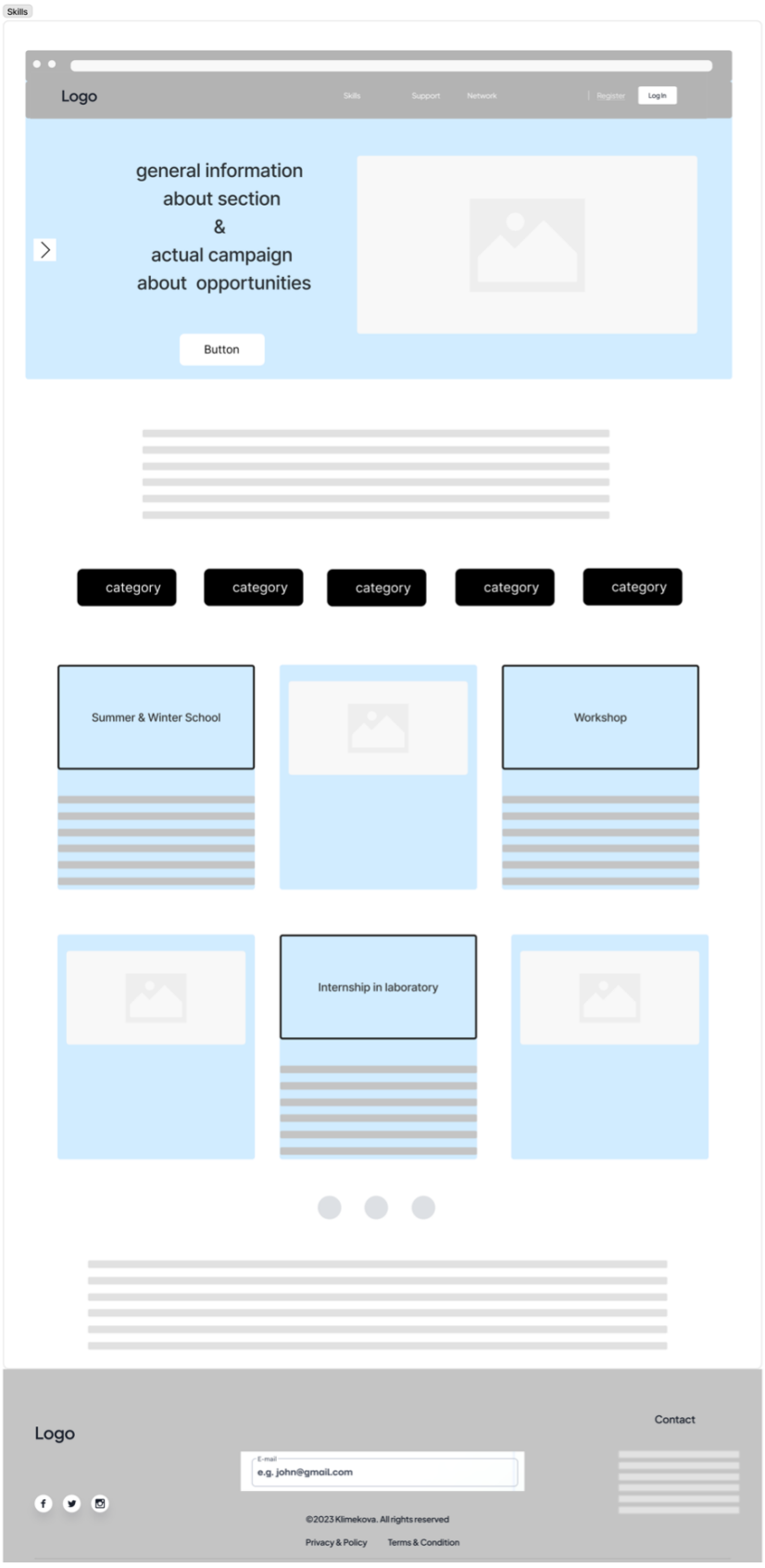
Filter kategórií: Filter na základe zvolenej kategórie - vybraného tlačidla umožňuje používateľovi vybrať konkrétnu kategóriu a pomocou tlačidla filtrovať obsah, čím zobrazuje len príslušné položky, ktoré spadajú do danej kategórie.

List položiek: List príležitosti na vzdelávanie v kategórii Skills alebo podpora v kategórii Startup Support. V liste sú zobrazené na základe aktuálneho filtra karty jednotlivých položiek. Ak je položiek v liste veľa, ich zobrazenie je obmedzené na určitý počet (napr. 6 na stránku) a užívateľ sa vie medzi nimi preklikávať pomocou tzv. *stránkovej* navigácie.

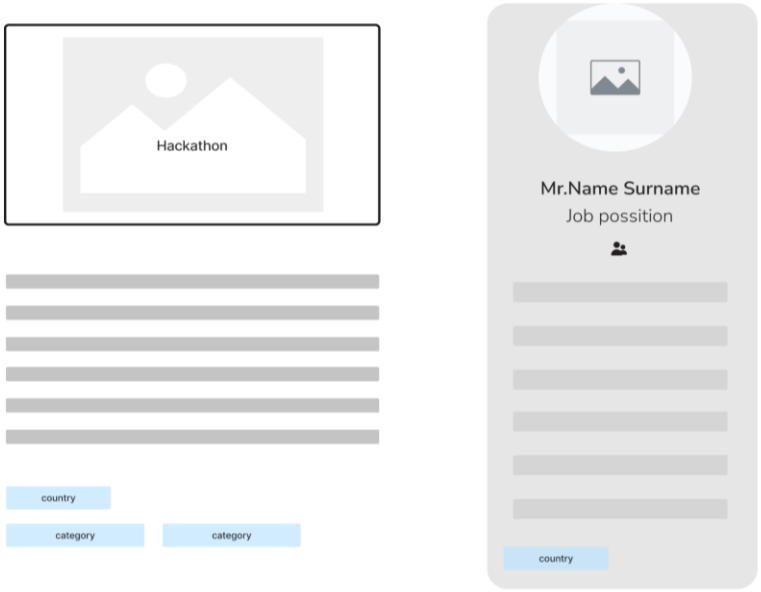
Footer: Je častou súčasťou webových stránok, ktorá sa nachádza v spodnej časti obrazovky. Obsahuje dôležité informácie a odkazy, ktoré sú prístupné po celú dobu, a poskytuje dodatočnú navigáciu a kontaktné informácie pre používateľov. *How To Do Biotech* aplikácia ponúka v časti Footer aj možnosť pre ďalšiu fázu vývoja a nové funkcionality ako prihlásiť sa do newlstettera, odkazy na *Privacy Policy* a *Terms of Use* či odkaz na zdrojový kód na Githubu skrz ikonu Githubu.

Sekcia Network má najvýraznejší rozdiel v tom, že obsahuje dva dynamické zoznamy - zoznam Experti a zoznam Inovácie. Okrem toho nemá horný dynamicky sa meniaci prvok Carousel ale statický vizuálny banner. Inak sú na nej použité rovnaké komponenty kvôli konzistencii dizajnu aj lepšiemu UX.

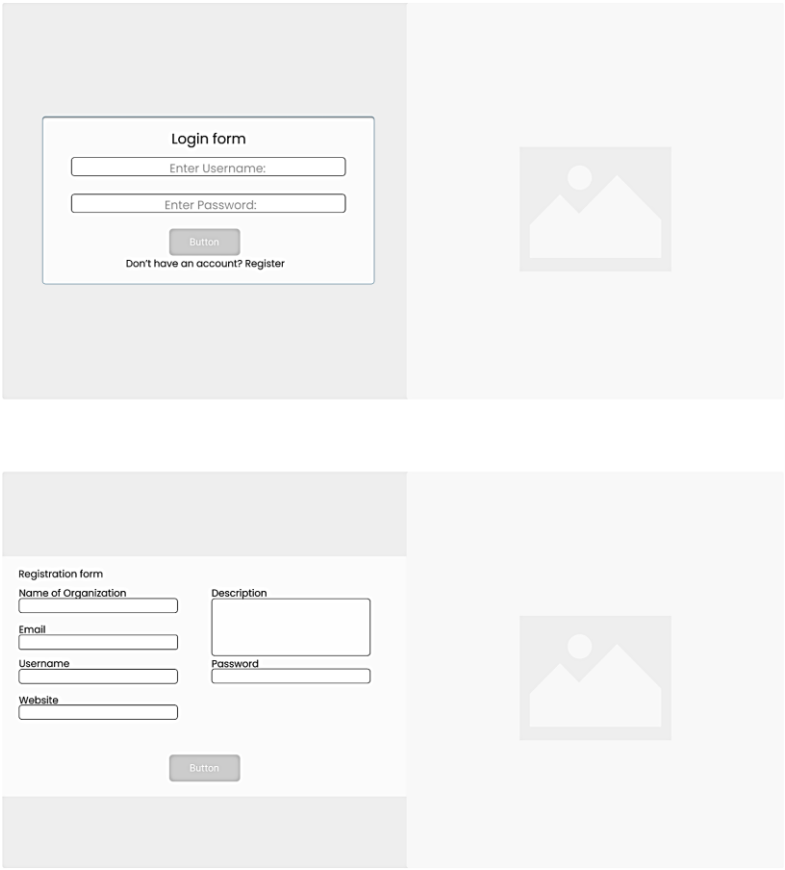
Webová aplikácia obsahuje v pilotnej fáze aj niekoľko formulárov - na registráciu, prihlásenie sa, editáciu jednotlivých položiek. Vo veľkej miere využíva aj informačné karty na zobrazovanie detailov pre daný objekt.



Obrázok 16: Wireframe sekcie *Skills*



Obrázok 17: Wireframe informačných kariet pre experta a vzdelávaciu príležitosť



Obrázok 18: Wireframe login a logout formuláre

6 IMPLEMENTACE APLIKACE

Táto časť práce sa venuje samotnej implementácii MVP prototypu aplikácie. Je obsahovo rozdelená na podkapitolu o vývojom prostredí a softvérových nástrojoch použitých pri vývoji, na backendovú implementáciu, databázu a implementáciu používateľského rozhrania prostredníctvom frontend vývoja. Záver kapitoly sumarizuje implementované prvky bezpečnosti a testovania.

6.1 Vývojové prostredie

Pre vývoj backendovej časti aplikácie bolo zvolené prostredie Intellij IDEA. IntelliJ IDEA je výkonné integrované vývojové prostredie (IDE), ktoré poskytuje špecifické vlastnosti pre vývoj aplikácií s frameworkom Spring Boot. S IntelliJ IDEA môžete jednoducho vytvárať a spravovať projekty Spring Boot, pričom sa vám poskytne podpora pre rýchle a efektívne vývojové procesy. Jeho vlastnosti zahŕňajú inteligentné dopĺňanie kódu, refaktoring, nástroje na ladenie a profilovanie, integráciu s Mavenom a Spring Boot frameworkom, podporu pre správu verzií a mnoho ďalších nástrojov, ktoré vám umožnia pohodlne vyvíjať, testovať a nasadzovať svoje Spring Boot aplikácie.[49]

Visual Studio Code [50] je populárne integrované vývojové prostredie (IDE), ktoré je špeciálne navrhnuté pre vývoj aplikácií s React frameworkom. S inteligentným dopĺňaním kódu, syntax highlight pre JSX a JavaScript.

Obidve prostredia, IntelliJ IDEA aj Visual Studio Code, poskytujú jednoduchú integráciu s verejným repozitárom na platforme Github so zdrojovým kódom. Tak sa zabezpečilo plynulé verzovanie počas celého procesu vývoja.

Pre debugovanie a testovacie účely boli požuté softvér Postman a prostredie Swagger. Postman [51] je nástroj pre testovanie a ladenie API, ktorý umožňuje vývojárom posielať HTTP požiadavky, spravovať odpovede, testovať rôzne scenáre a simulovať interakciu so serverom.

Swagger [52] je prostredie pre dizajn, dokumentáciu a testovanie API, ktoré poskytuje interaktívnu dokumentáciu, generovanie kódu a nástroje pre validáciu a testovanie API.

Práca s dátami prebiehala v dvoch prostrediach - H2 Console a AWS S3 buckets. H2 Console [53] je webové rozhranie pre správu relačnej databázy H2. Pomocou H2 Console môžu vývojári spravovať dáta v databáze, vytvárať tabuľky, vykonávať dotazy a manipulovať s

dátami. Poskytuje prístup k rôznym nástrojom na správu a administráciu databázy, vrátane prehľadu schémy, editora SQL dotazov, a možnosti importovať a exportovať dáta.

Na druhej strane, AWS S3 - Simple Storage Service) [54] je služba poskytovaná Amazon Web Services, ktorá umožňuje ukladanie a správu objektových údajov vo forme "buckets" (kôp). S3 buckets poskytujú dátové úložisko v cloude, kde je možné ukladať, sťahovať a spravovať súbory a dáta. Táto služba umožňuje vývojárom flexibilne pracovať s veľkým objemom dát, ako aj ich distribúciu a zdieľanie.

Vývoj aplikácie je robený s jazykom Java 17 a React 18.

6.2 Backend implementácia

V nasledujúcich blokoch sú prezentované najzaujímavejšie časti implementácie Spring Boot aplikácie, ktorá zabezpečuje backend pre vznikajúcu platformu.

6.2.1 **Adresárová štruktúra**

Backend How To Do Biotech platformy je rozvrhnutý do klasickej adresárovej štruktúry Spring Boot aplikácie. Na obrázku 19 vidno nasledujúce adresáre a súbory: *.idea:* Adresár obsahujúci konfiguračné súbory pre vývojové prostredie IntelliJ IDEA.

mvn: Adresár obsahujúci konfiguračné súbory pre Apache Maven (buildovací nástroj).

data: Adresár, ktorý môže obsahovať dáta alebo iné súbory potrebné pre projekt.

src: Adresár obsahujúci hlavné zdrojové súbory projektu.

main: Adresár obsahujúci zdrojové súbory pre hlavnú časť projektu. Ten sa delí na zložky main a test.

java: Adresár obsahujúci zdrojový kód v jazyku Java. V rámci neho je zadaná štruktúra pre *cz.utb.fai.howtodobiotech* Tento podadresár obsahujúci balíčky pre rôzne časti projektu.

api: Adresár obsahujúci súbory pre API rozhranie, v tomto prípade všetky triedy s anotáciou @Controller.

config: Adresár obsahujúci konfiguračné súbory.

models: Adresár obsahujúci modely, resp. objekty používané v servisoch a v kontroleroch.

payload: Adresár obsahujúci súbory pre dáta prenosné po sieti pri autentifikácii.

repositories: Adresár obsahujúci súbory s anotáciou @Repository pre správu dátových repozitárov.

security: Adresár obsahujúci súbory pre zabezpečenie aplikácie, najmä týkajúce sa aplikácie *JWT - JSON Web Tokenu.*

services: Adresár obsahujúci súbory pre služby, teda implementáciou biznisovej logiky.

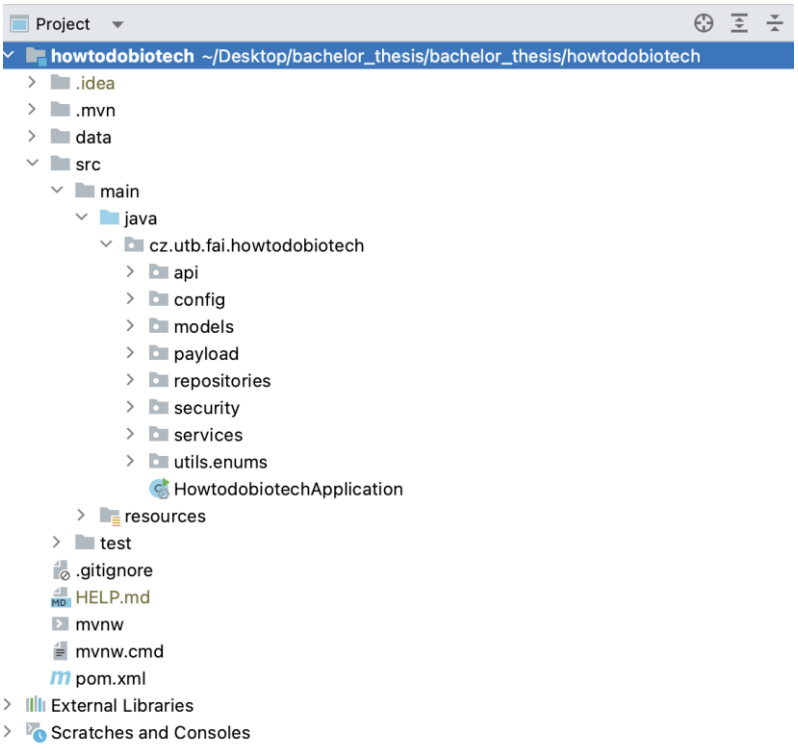
enums: Adresár obsahujúci súbory pre enumerácie.

resources: Adresár obsahujúci nezdrojové súbory, ako napríklad *application.properties*

test: Adresár obsahujúci unit testy, prípadne ďalšie súbory pre testovanie projektu

.gitignore: Súbor obsahujúci zoznam súborov a adresárov, ktoré majú byť ignorované verzovacím systémom Git.

Okrem týchto hlavných adresárov obsahuje projekt samotnú hlavnú triedu aplikácie *HowtodobiotechApplication* a *pom.xml,* čo je hlavný konfiguračný súbor pre projekt v Maven. Obsahuje závislosti, pluginy a ďalšie konfiguračné informácie potrebné pre zostavenie a správu projektu.



Obrázok 19: Adresárová štruktúra backend časti projektu

6.2.2 Dependencies a konfigurácia

Závislosti, tzv. *dependencies* definujú knižnice a moduly, ktoré sú potrebné pre správne fungovanie aplikácie. Konfigurácia Spring Boot aplikácie sa zvyčajne uskutočňuje prostredníctvom súboru *application.properties* alebo *application.yml.* Tieto súbory umožňujú nastavenie rôznych vlastností aplikácie, ako napríklad port na ktorom aplikácia beží, databázové pripojenie a ďalšie.

Hlavné závislosti projektu (dependencies):

- *spring-boot-starter-data-jpa:*poskytuje podporu pre prácu s JPA (Java Persistence API).
- *spring-boot-starter-security*: poskytuje bezpečnostné funkcie a autentifikáciu pre aplikáciu.
- *spring-boot-starter-validation:* poskytuje validáciu dát v aplikácii.
- *spring-boot-starter-web:*oskytuje podporu pre vývoj webových aplikácií s použitím Spring MVC.
- *h2:* Závislosť pre H2 Database
- *jjwt:* Závislosť pre JSON Web Token (JWT), ktorý poskytuje podporu pre generovanie a spracovanie tokenov.
- *spring-boot-starter-test:* poskytuje rôzne testovacie funkcionality pre unit aj integračné testovanie.
- *lombok:* poskytuje zjednodušenie vývoja pomocou automatického generovania kódu.
- *springdoc-openapi-starter-webmvc-ui:* OpenAPI generovanie a dokumentácia REST API
- *validation-api:* poskytuje nástroje pre validáciu dát v Java aplikáciách.

Build pluginy:

- *spring-boot-maven-plugin:* je potrebný pre spúšťanie Spring Boot aplikácie. Tento plugin zabezpečuje správne nastavenie a spustenie aplikácie.

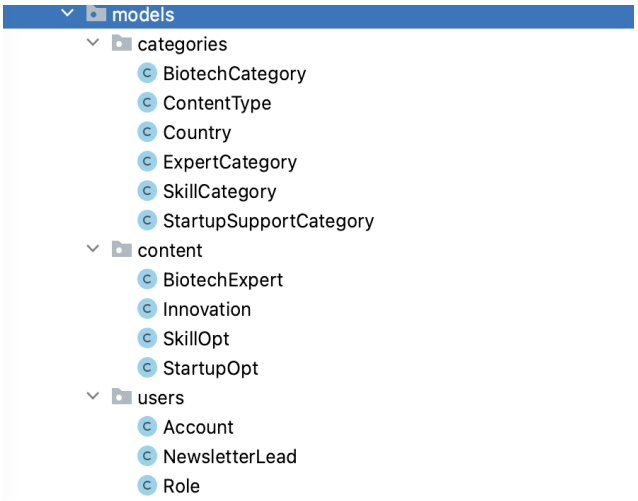
Application.properties:

V súbore *application.properties* sa nachádzajú konfiguračné nastavenia ako povolenie H2 konzoly a nastavenie jej cesty, nastavenia pripojenia k H2 databáze vrátane URL adresy, používateľského mena a hesla. Ďalej sú tu nastavenia pre Hibernate, ako je zobrazovanie generovaných SQL dotazov, dialekt H2 databázy a automatické aktualizovanie schémy pri

štarte aplikácie, je tu definovaný tajný kľúč pre generovanie a overovanie tokenov, či nastavenia pre Swagger, vrátane ciest k dokumentácii a Swagger UI a spôsobu zoradenia operácií v UI. Tieto nastavenia prispievajú k správnemu fungovaniu Spring Boot aplikácie a definujú jej vlastnosti a správanie.

6.2.3 **Models**

V rámci Spring Boot REST API aplikácie sa modely používajú na reprezentáciu dátových štruktúr alebo entít, ktoré sa používajú v systéme. Modely definujú atribúty a správanie týchto entít.



Obrázok 20: Štruktúra Models

Niektoré vlastnosti a anotácie tried sú nasledovné:

@Entity: Táto anotácia označuje triedu ako entitu, ktorá sa mapuje na tabuľku v databáze.

@Table(name = "tableName"): Táto anotácia definuje názov tabuľky v databáze, na ktorú je táto entita mapovaná.

@Setter a @Getter: Tieto anotácie generujú *setters* a *getters* pre atribúty triedy automaticky.

@NoArgsConstructor a @AllArgsConstructor: Tieto anotácie generujú konštruktory bez parametrov a konštruktory so všetkými parametrami .

@Id: Táto anotácia označuje atribút id ako primárny kľúč entity.

Na určenie vzťahov s inými modelmi - entitami slúžia anotácie ako *@ManyToMany, @OneToMany, @JoinTable a @JoinColumn* a pod.

```
1 usage
@NotNull
@ManyToMany(fetch=FetchType.EAGER)
@JoinTable(name = "SkillOpt_Countries",
        joinColumns = @JoinColumn(name = "SkillOpt_id"),
        inverseJoinColumns = @JoinColumn(name = "Country_Id"))
private Set<Country> countries = new HashSet<>();
1 usage
@NotNull
@ManyToMany(fetch=FetchType.EAGER)
@JoinTable(name = "SkillOpt_BiotechCategories",
        joinColumns = @JoinColumn(name = "SkillOpt_id"),
        inverseJoinColumns = @JoinColumn(name = "BiotechCategory_id"))
private Set<BiotechCategory> biotechCategories = new HashSet<>();

1 usage
@NotNull
@ManyToMany(fetch=FetchType.EAGER)
@JoinTable(name = "SkillOpt_SkillCategories",
        joinColumns = @JoinColumn(name = "SkillOpt_id"),
        inverseJoinColumns = @JoinColumn(name = "SkillCategory_id"))
private Set<SkillCategory> skillCategories = new HashSet<>();
```

Obrázok 21: Ukážka ošetrenia vzájomných vzťahov medzi viacerými entitami

Tento kód definuje asociácie typu *Many-to-Many* medzi entitou *SkillOpt* a ďalšími entitami *Country, BiotechCategory, SkillCategory*. Objekt *SkillOpt* je zosobnením "Skill Opportunity", teda príležitosti pre získanie nových zručností, ktoré sú zverejnené na podstránke "Skills" v aplikácii How To Do Biotech.

fetch=FetchType.EAGER: je nastavenie, ktoré určuje, že všetky súvisiace entity *Country*, *BiotechCategory* a *SkillCategory* sa načítajú s entitou *SkillOpt* okamžite.

@JoinTable: Táto anotácia definuje spojovaciu tabuľku, ktorá bude použitá na uchovávanie vzťahov medzi entitami.

- *name:* Názov spojovacej tabuľky pre danú asociáciu.
- *joinColumns:* určuje stĺpce, ktoré odkazujú na entitu *SkillOpt*.
- *inverseJoinColumns:* určuje stĺpce, ktoré odkazujú na príslušnú entitu napríklad *Country, BiotechCategory* a *SkillCategory*

Každá kolekcia (countries, biotechCategories, skillCategories) je inicializovaná typovo ako *HashSet*, čo umožňuje jednoduché pridávanie a správu súvisiacich entít.

V aplikácii je implementovaných niekoľko ENUM typov. Enum je objektový typ, ktorý predstavuje súbor preddefinovaných konštánt. Konštanty enum predstavujú konkrétne hodnoty, ktoré je možné použiť v rámci programu. Enum objekty sú následne volané v modely, kde je pridelené Id - jedinečný identifikátor každej z možností v enum.

```
8 usages  baskaklimek
@Entity
@Table(name = "SkillCategories")
@Setter
@Getter
@NoArgsConstructor
@AllArgsConstructor
public class SkillCategory {

    no usages
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    no usages
    @Enumerated(EnumType.STRING)
    @Column(length = 20)
    private ESkillCategory name;
}
```

Obrázok 22: Ukážka spracovania Enum objektu v modely

Na obrázku 22je zobrazený kód, ktorý definuje ako bude enum implementované. Anotácia *@Enumerated(EnumType.STRING)* určuje, ako bude enum hodnota *ESkillCategory* mapovaná do databázy. V tomto prípade sa používa *EnumType.STRING,* čo znamená, že enum hodnota bude mapovaná na reťazec. Anotácia @Column(length = 20) definuje konkrétne nastavenie stĺpca pre atribút *name* v databáze. *length = 20* určuje maximálnu dĺžku tohto stĺpca, ktorá je 20 znakov.

Z toho vyplýva, že enum hodnota *ESkillCategory* bude správne mapovaná na stĺpec v databáze ako reťazec s maximálnou dĺžkou 20 znakov.

6.2.4 Repositories

Trieda s anotáciou *@Repository* v Spring Boot aplikácii je zodpovedné za manipuláciu s dátami v databáze. Tento komponent poskytuje metódy na vytváranie, čítanie, aktualizáciu a mazanie záznamov v databáze. Repository trieda obsahuje anotácie, ktoré špecifikujú typ

entity, s ktorou pracuje, a prípadne aj ďalšie nastavenia. Tieto metódy sú definované pomocou štandardných konvencií alebo pomocou vlastných dotazov.

```
@Transactional
@Repository
public interface StartupOptRepository extends JpaRepository<StartupOpt, Integer> {

    1 usage  baskaklimek
    @Query("SELECT s FROM StartupOpt s JOIN s.biotechCategories sc WHERE sc.name = :categoryName")
    List<StartupOpt> findByCategoryName(@Param("categoryName") EBiotechCategory categoryName);

    1 usage  baskaklimek
    @Query("SELECT s FROM StartupOpt s JOIN s.supportCategories sc WHERE sc.name = :supportCategoryName")
    List<StartupOpt> findBySupportCategoryName(@Param("supportCategoryName") ESupportCategory supportCategoryName);

    1 usage  baskaklimek
    @Query("SELECT s FROM StartupOpt s JOIN s.countries sc WHERE sc.name = :countryName")
    List<StartupOpt> findByCountryName(@Param("countryName") ECountry countryName);

    1 usage  baskaklimek
    Optional<StartupOpt> findByTitle(String title);

    1 usage  baskaklimek
    Optional<StartupOpt> findByProvider(String organizer);

    1 usage  baskaklimek
    Optional<StartupOpt> findByStartDate(Date startDate);

    1 usage  baskaklimek
    Optional<StartupOpt> findByEndDate(Date endDate);

    1 usage  baskaklimek
    List<StartupOpt> findByAccountId(Integer accountId);
}
```

Obrázok 23: Ukážka spracovania triedy Repository a queries

Obrázok 13 ukazuje implementáciu niekoľkých dotazov na *StartupOpt* repozitár. Napríklad:

@Query("SELECT s FROM StartupOpt s JOIN s.countries

sc WHERE sc.name = :countryName").

Tento dotaz vykonáva *SELECT* operáciu na entite *StartupOpt*. Používa *JOIN* na spojenie entity *StartupOpt* s entitou *countries*, ktorá reprezentuje vzťah *ManyToMany* medzi *StartupOpt a ECountry.* V dotaze sa ďalej porovnáva hodnota vlastnosti *name* z entity *countries* s parametrom *countryName,* ktorý je poskytnutý ako argument metódy. Výsledkom je list *StartupOpt* záznamov, ktoré majú zhodnú hodnotu vlastnosti *country*.

6.2.5 Services

Anotácia @Service v Spring Boot označuje triedu ako komponentu, ktorá poskytuje služby alebo logiku aplikácie.

```
1 usage  baskaklimek
@Transactional
public void updateSkillOpt(Integer id, SkillOpt skillOptDto) {
    SkillOpt skillOpt = skillOptRepository.findById(id)
            .orElseThrow(() -> new NotFoundException("Skill opportunity not found with id: " + id));

    Set<Country> countries = skillOptDto.getCountries();
    Set<BiotechCategory> biotechCategories = skillOptDto.getBiotechCategories();
    Set<SkillCategory> skillCategories = skillOptDto.getSkillCategories();
    skillOpt.setTitle(skillOptDto.getTitle());
    skillOpt.setOrganizer(skillOptDto.getOrganizer());
    skillOpt.setDescription(skillOptDto.getDescription());
    skillOpt.setStartDate(skillOptDto.getStartDate());
    skillOpt.setEndDate(skillOptDto.getEndDate());
    skillOpt.setWebsite(skillOptDto.getWebsite());
    skillOpt.setCountries(countries);
    skillOpt.setAccountId(skillOptDto.getAccountId());
    skillOpt.setBiotechCategories(biotechCategories);
    skillOpt.setSkillCategories(skillCategories);

    skillOptRepository.save(skillOpt);
}
```

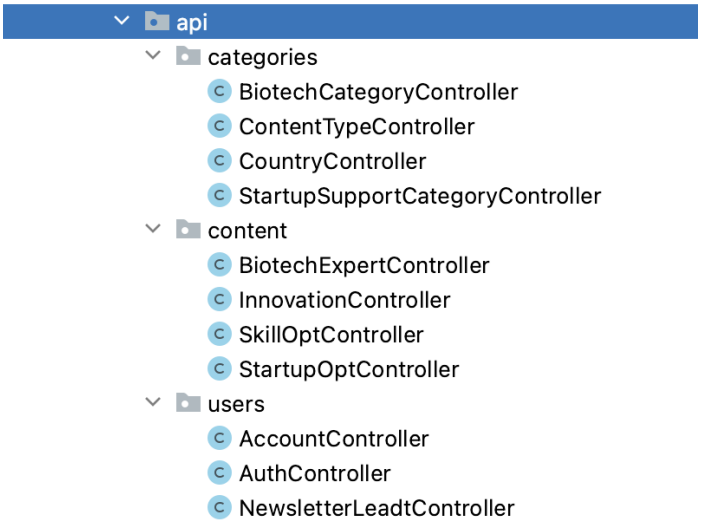
Obrázok 24: Ukážka logiky v triede typu @Service

Napríklad *SkillOptService* trieda obsahuje rôzne metódy, ktoré vykonávajú operácie nad *SkillOpt* entitami. Napríklad *selectSkillOptById(), selectAllSkillOpts(), insertSkillOpt(), updateSkillOpt(), deleteSkillOptById()* a ďalšie. Okrem toho, trieda obsahuje aj metódy na vyhľadávanie *SkillOpt* záznamov podľa rôznych kritérií, ako napríklad kategórie biotechnológie, kategórie zručnosti, krajiny, identifikátoru účtu, titulu, organizátora a dátumu začiatku a ukončenia.

Každá trieda obsahuje konštruktor, v ktorom je anotácia *@Autowired* použitá na realizáciu závislosti medzi Service a Repository triedami. To znamená, že Spring framework automaticky vytvorí inštanciu napríklad *SkillOptRepository* a priradí ju do premennej *skillOptRepository* v triede *SkillOptService*. Takto sa zabezpečuje dostupnosť repozitára v rámci Service triedy, čo umožňuje službe pristupovať k metódam a funkciám, ktoré sú definované v *SkillOptRepository* a vykonávať príslušné operácie s databázou pre entitu *SkillOpt*.

6.2.6 Controllers

Kontroléri, teda triedy s anotáciou @Controller, tvoria spojenie medzi klientmi a serverom, pričom zabezpečujú interpretáciu a vykonávanie správnych akcií na základe prijatých požiadaviek. Sú návrhovým vzorom *Model-View-Controller*, kde zabezpečujú spracovanie prichádzajúcich požiadaviek a koordináciu medzi údajmi a prezentáciou.



Obrázok 25: Štruktúra REST API

V Spring Boot backend implementácii sa označujú napríklad týmito anotáciami:

- Anotácia *@CrossOrigin(origins = "*")* umožňuje pripojenie na koncové body (endpoints) z rôznych domén a zdrojov pomocou prehliadača. V konkrétnom prípade, nastavenie origins = "*" povolí prístup ku koncovým bodom zo všetkých domén.
- Anotácia *@RestController* označuje že výstup z metód triedy bude automaticky serializovaný do formátu vhodného na odoslanie klientovi (napr. JSON) a bude sa vrátiť ako odpoveď na HTTP požiadavky.
- Anotácia *@RequestMapping("")* určuje základnú cestu pre všetky koncové body, teda endpoints v triede. Tento prefix sa pripája k všetkým metódam v triede ako napr. *@GetMapping, @PostMapping.*

Väčšina kontrolérov v aplikácii má podobnú štruktúru. Z implementačného hľadiska je zaujímavá konštrukcia triedy *AuthController*. Napríklad endpoint slúžiaci na autentifikáciu.

```
no usages  baskaklimek
@PostMapping("/authenticate")
public ResponseEntity<AuthResponse> authenticate(@RequestBody LoginRequest loginRequest, HttpServletResponse response) {
    try {
        Authentication authentication = authenticationManager.authenticate(
                new UsernamePasswordAuthenticationToken(loginRequest.getUsername(), loginRequest.getPassword())
        );

        // Generate the authentication token
        String token = jwtUtils.generateToken(loginRequest.getUsername());

        AccountDetailsImpl userDetails = (AccountDetailsImpl) userDetailsService.loadUserByUsername(loginRequest.getUsername());
        Integer userId = userDetails.getId();

        response.setHeader( s: "Authorization",  s1: "Bearer " + token);
        response.setHeader( s: "Access-Control-Allow-Origin",  s1: "http://localhost:8081");
        response.setHeader( s: "Access-Control-Allow-Methods",  s1: "GET, POST, PUT, DELETE");
        response.setHeader( s: "Access-Control-Allow-Headers",  s1: "Origin, Content-Type, Authorization");

        // Create the login response object
        AuthResponse loginResponse = new AuthResponse(token, userId);


        return ResponseEntity.ok(loginResponse);
    } catch (AuthenticationException e) {
        // Handle invalid credentials
        return ResponseEntity.status(HttpStatus.UNAUTHORIZED).build();
    } catch (Exception e) {
        return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
    }
}
```

Obrázok 26: Ukážka kontrolóra z REST API

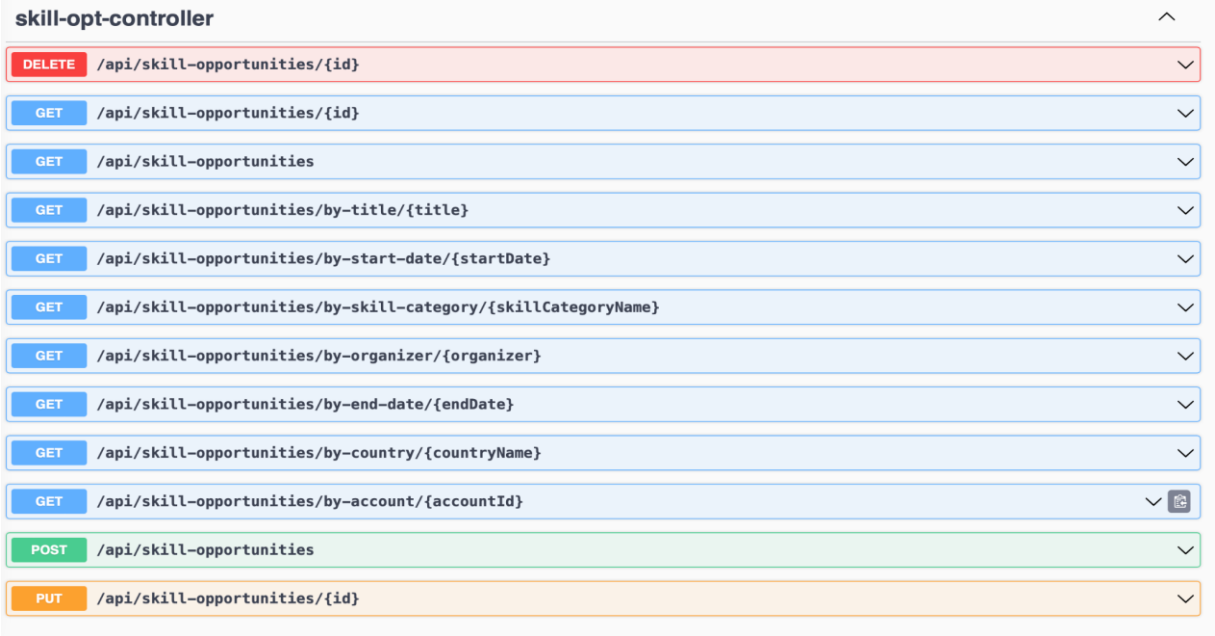
Tento kód predstavuje metódu *authenticate* označenú anotáciou *@PostMapping* v triede kontroléra. Metóda slúži na autentifikáciu používateľa. Po prijatí požiadavky s prihlasovacími údajmi sa autentifikuje používateľ a generuje sa autentifikačný token JWT. Odpoveď obsahuje autentifikačný token a identifikátor používateľa. V prípade neplatných poverení sa vráti odpoveď s kódom 401 *UNAUTHORIZED* a v prípade chyby pri spracovaní požiadavky odpoveď s kódom 500 *INTERNAL SERVER ERROR*.

V tejto metóde sa generuje JWT token pomocou *jwtUtils*. Metóda *generateToken* vytvára JWT na základe používateľského mena poskytnutého v *loginRequest*.

Následne sa nastavuje hlavička odpovede HTTP, ktorej súčasťou je aj vygenerovaný autorizačný token.

6.2.7 Swagger dokumentácia

Swagger OpenAPI je nástroj a špecifikácia na vytváranie, dokumentovanie a testovanie RESTful API.



Obrázok 27: Ukážka enpointov pre Skill-Opt-Controller

Projekt má implementovaných 11 kontrolérov s mnohými endpointami. Na prototype komunikuje frontend časť aplikácie momentálne z väčšinou z nich a priamo ich využíva v prípadoch použitia. Niektoré, ako napríklad *newsletter-lead-controller* je na backend strane implementovaný a na frontende v prípade rozvoja prototypu stačí aktualizovať aktivity, nakoľko Service trieda je navrhnutá už kompletne aj na frontende.

V projekte sú tieto kontroléri:

startup-opt-controller: sa zaoberá operáciami súvisiacimi so startupovými príležitosťami. Obsahuje metódy na vytvorenie, zobrazenie, aktualizáciu a odstránenie startupových príležitostí, ich vyhľadávanie podľa rôznych parametrov.
skill-opt-controller: sa zaoberá operáciami súvisiacimi so zručnosťami a príležitosťami na rozvoj zručností. Obsahuje metódy na vytvorenie, zobrazenie, aktualizáciu a odstránenie zručností a rovnako metódy pre ich vyhľadávanie podľa rôznych parametrov.
innovation-controller: spravuje operácie súvisiace s inováciami. Môže obsahovať metódy na vytvorenie, zobrazenie, aktualizáciu a odstránenie inovácií.
biotech-expert-controller: sa stará o operácie súvisiace s biotechnologickými expertmi. Obsahovať metódy na vytvorenie, zobrazenie, aktualizáciu a odstránenie expertov. Momentálne MVP aplikácie je navrhnuté tak, že tieto funkcie môže spravovať iba admin, teda používateľ s rolou ROLE_ADMIN. Vložiť, upraviť aj vymazať vie jednotlivých expertov v

databáze alebo aj v rozhraní Swagger. Do budúcnosti je možné pridať do aplikácie dashboard pre prihláseného admina a rozhranie pre CRUD operácie cez formuláre.

account-controller: zabezpečuje operácie s používateľskými účtami.

newsletter-lead-controller: sa stará o operácie súvisiace s užívateľmi prihlásenými na odber noviniek. Momentálne nie je implementované volanie z frontend strany.

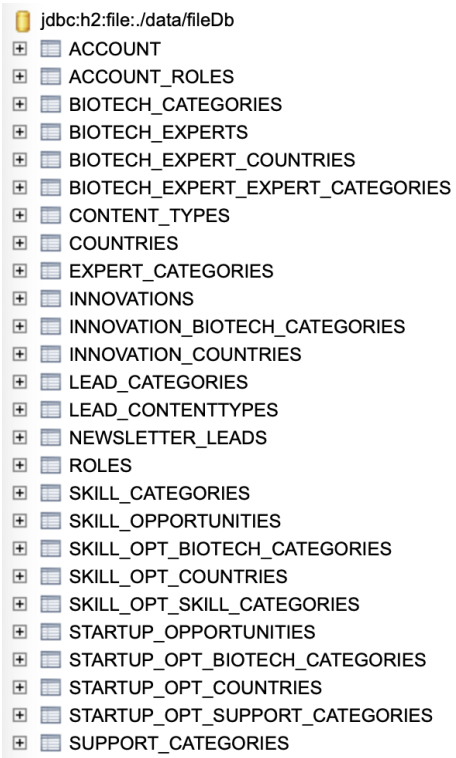
auth-controller: Tento kontrolér riadi autentifikáciu a autorizáciu používateľov. Môže obsahovať metódy na prihlásenie, odhlásenie a správu autentifikačných tokenov.

startup-support-category-controller, country-controller: Tieto kontroléri spravujú operácie súvisiace s kategóriami a krajinami.

content-type-controller: riadi operácie súvisiace s typmi obsahu.

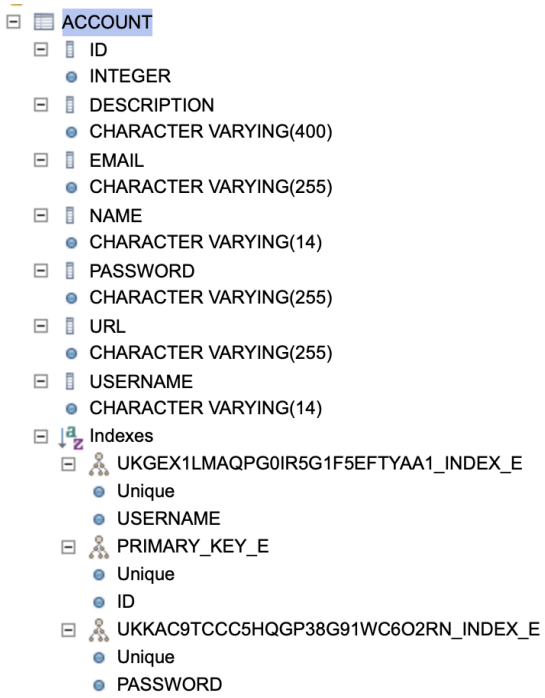
6.3 Databáza

Na obrázku 28 je znázornená databázová štruktúra tabuliek. Nachádza sa v nej niekoľko spojovacích tabuliek, ktoré riešia vzťahy *ManyToMany*. Takou je napríklad aj tabuľka *Account_Roles,* ktorá spája informácie o účte a pridelenej roly. Nachádza sa v nej *AccountId* a *RoleId*.



Obrázok 28: Databázová štruktúra tabuliek

Na ďalšom obrázku 29 je znázornená implementácia entity *Account* do tabuľky s jednotlivými vlastnosťami. Zaujímavosťou je nastavenie unikátnej hodnoty pre *username* a *password*.



Obrázok 29: Tabuľka entity Account

Modul S3 od AWS je implementovaný najmä kvôli obsahu ako sú obrázky. Momentálne sa na tomto úložisku nachádzajú fotky jednotlivých expertov, ktorí sú publikovaní v sekcii Network, do budúcna je možné rozšírením na backend časti zabezpečiť ukladanie nahraných obrázkov k vytváraným príležitostiam v sekcii Skill a Startup Support. Ku každému obrázku na AWS S3 je vygenerovaná URL adresa, na ktorú sa odkazuje tabuľka *BIOTECH_EXPERTS* a stĺpec *PROFILE_IMAGE_URL* .

6.4 Frontend implementácia

V nasledujúcich blokoch sú prezentované najzaujímavejšie časti implementácie frontendovej časti aplikácie, ktorá je vytváraná ako React aplikácia s použitím *Redux* a *Axios* podpory.

6.4.1 Adresárová štruktúra

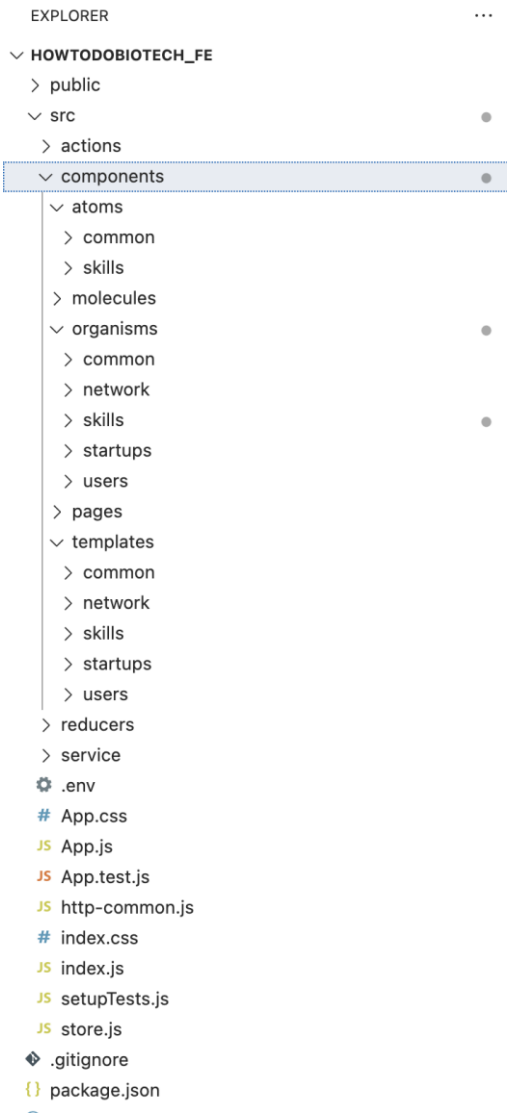
Pri zvolenej architektúre frontendu sú štandardnou súčasťou adresárovej štruktúry priečinky ako *actions, reducers, components a services*.

actions: V tomto priečinku sa nachádzajú súbory, ktoré definujú akcie pre interakciu so stavom aplikácie. Akcie sú objekty, ktoré popisujú niečo, čo sa stalo v aplikácii, napríklad požiadavka na získanie dát, aktualizácia stavu, prihlásenie používateľa a podobne. Tieto súbory obsahujú funkcie na vytváranie a spracovanie týchto akcií.

reducers: Obsahuje súbory, ktoré definujú správu stavu aplikácie. Reduktory sú funkcie, ktoré prijímajú akciu a aktuálny stav a vrátia nový stav aplikácie.

services: Ide o implementáciu servís pre komunikáciu so serverom alebo iné externe zdroje. Práve tu je definované volanie endpointov na backend strane.

components: Tento priečinok obsahuje komponenty, ktoré sú znovu použiteľné a zložené z menších častí - *atoms, molecules* - ako aj výsledné stránky - *pages*.



Obrázok 30: Adresárová štruktúra frontend časti projektu

Okrem toho sa v štruktúre nachádza priečinok *public,* ktorý obsahuje verejné súbory a aktíva aplikácie, ktoré sú dostupné priamo z webového prehliadača. Sem patria napríklad obrázky, favicony a podobne. V hlavnej štruktúre sú osobitne dôležité aj nasledujúce súbory:

store.js: ide o implementáciu *Redux Store*, ktorý slúži na správu globálneho stavu aplikácie. V ňom sa definuje, ako sú spojené reduktory a ako sa spracovávajú akcie.
index.js: slúži na inicializáciu a vykreslenie hlavného komponentu aplikácie do DOMu.

http-common.js: obsahuje konfiguráciu pre HTTP požiadavky a volanie Axiosu. Definuje sa tu základná URL adresa servera alebo API.

App.js: hlavná trieda - komponent aplikácie. Je to koreňový komponent, ktorý obsahuje ďalšie komponenty a definuje celkovú štruktúru a správanie aplikácie.

package.json: konfigurácie a zoznam závislostí, skripty na spustenie, zostavenie a testovanie aplikácie.

6.4.2 **Knižnice**

Závislosti a knižnice, ktoré sú využité pri vývoji definuje už spomínaný súbor package.json. Medzi hlavné knižnice projektu patrí:

- set knižníc React ako napríklad *"react-router-dom": "^5.3.4"* pre správu navigácie a routovania v Reacte, *react": "^18.2.0",* čo je hlavná knižnica React pre vytváranie užívateľského rozhrania, alebo *react-dom": "^18.2.0"* - React adaptér pre manipuláciu s DOM-om .
- set knižníc pre implementáciu princípov Reduxu ako *"react-redux": "^8.0.5"* - Knižnica pre správu stavu aplikácie (state) v Reacte, *"redux-thunk": "^2.4.2"* - Redux middleware pre spracovanie asynchrónnych akcií.
- knižnica *"axios": "^1.4.0"* -pre vytváranie HTTP požiadaviek.

Pre dizajn UI aplikácie je využitá knižnica *"^bootstrap": "5.1"* , čo je CSS framework Bootstrap pre rýchle a jednoduché vytváranie responzívnych webových stránok a "mdb-react-ui-kit": "^6.0.0" - UI kit pre React s implementovaným materiál design princípmi.

Pre prácu s formulármi sa využila knižnica *"formik": "^2.2.9"* pre spracovanie a validáciu formulárov v Reacte a *"yup": "^1.1.1",* knižnica pre definovanie a validáciu schémy dát v Reacte.

6.4.3 Redux

Projekt obsahuje 6 reduktorov. Každý slúži na definovanie iného setu stavov pre jednotlivé objekty. Všetky reduktory súbory sú spájané v index.js v priečinku *Reducers*, kde sa kombinujú. Na obrázku 31 je znázornený *SkilOptReducer* a detail na stav pri vytvorení nového objektu *SkillOpt*, teda príležitosti pre vzdelanie na podstránku Skills.

src > reducers > JS skills.js > ...

```
1   import {
2     CREATE_SKILL_OPT,
3     RETRIEVE_SKILL_OPTS,
4     UPDATE_SKILL_OPT,
5     DELETE_SKILL_OPT,
6   } from "../actions/types.js";
7
8   const initialState = {
9     skillOpts: [],
10  };
11
    Complexity is 17 You must be kidding
12  function skillOptReducer(state = initialState, action) {
13    const { type, payload } = action;
14
15    switch (type) {
16      case CREATE_SKILL_OPT:
17        return {
18          ...state,
19          skillOpts: [...state.skillOpts, payload],
20        };
21
22      case RETRIEVE_SKILL_OPTS:
23    >   return {…
26        };
27
28      case UPDATE_SKILL_OPT:
29    >   return {…
41        };
42
43      case DELETE_SKILL_OPT:
44    >   return {…
47        };
48
49      default:
50        return state;
51    }
52  }
```

Obrázok 31: Reducer implementácia

Nižšie je detailná implementácia akcie pre túto istú aktivitu - vytváranie novej príležitosti. Na obrázku 32 je znázornení kód akcie s názvom *createSkillOpt,* ktorá sa volá pri

vytváraní nového objektu *SkillOpt* a vykonáva asynchrónne operácie pomocou služby *SkillOptDataService.*

```
export const createSkillOpt = (
  title,
  organizer,
  description,
  startDate,
  endDate,
  website,
  accountId,
  countries,
  biotechCategories,
  skillCategories
Complexity is 5 Everything is cool!
) => async (dispatch) => {
  try {
    const res = await SkillOptDataService.createSkillOpt({
      title,
      organizer,
      description,
      startDate,
      endDate,
      website,
      accountId,
      countries,
      biotechCategories,
      skillCategories
    });

    dispatch({
      type: CREATE_SKILL_OPT,
      payload: res.data,
    });

    return Promise.resolve(res.data);
  } catch (err) {
    return Promise.reject(err);
  }
};
```

Obrázok 32: Action implementácia pre metódu *createSkillOpt*

Táto funkcia prijíma rôzne parametre, ako je názov, organizátor, popis, dátumy, webová stránka, ID účtu a ďalšie údaje potrebné pre vytvorenie *SkillOpt* inštancie. Tieto údaje odosielajú ako požiadavka na vytvorenie *SkillOpt* . Je pri tom použitá metóda s názvom *SkillOptDataService.createSkillOpt()*. Po úspešnom vytvorení *SkillOpt* sa vyvolá akcia typu *CREATE_SKILL_OPT,* ktorá obsahuje dáta získané zo servera. Táto akcia je odoslaná do reduktora, ktorý aktualizuje stav aplikácie o nový *SkillOpt* objekt. V prípade chyby pri vytváraní sa odovzdá chyba pomocou *Promise.reject().*

Pri akciách je dôležité definovať aj jednotlivé typy, tzv. *types*. Typy akcií v súbore types.js sú často definované ako konštanty, napríklad *CREATE_SKILL_OPT*, ktoré sa potom používajú pri volaní akcií v rámci komponentov alebo v iných častiach aplikácie

src > actions > JS types.js > ...

```
1   export const CREATE_INNOVATION = "CREATE_INNOVATION";
2   export const RETRIEVE_INNOVATIONS = "RETRIEVE_INNOVATIONS";
3   export const UPDATE_INNOVATION = "UPDATE_INNOVATION";
4   export const DELETE_INNOVATION = "DELETE_INNOVATION";
5
6
7   export const CREATE_STARTUP_OPT = "CREATE_STARTUP_OPT";
8   export const RETRIEVE_STARTUP_OPTS = "RETRIEVE_STARTUP_OPTS";
9   export const UPDATE_STARTUP_OPT = "UPDATE_STARTUP_OPT";
10  export const DELETE_STARTUP_OPT = "DELETE_STARTUP_OPT";
11
12  export const CREATE_SKILL_OPT = "CREATE_SKILL_OPT";
13  export const RETRIEVE_SKILL_OPTS = "RETRIEVE_SKILL_OPTS";
14  export const UPDATE_SKILL_OPT = "UPDATE_SKILL_OPT";
15  export const DELETE_SKILL_OPT = "DELETE_SKILL_OPT";
16
17
18  export const CREATE_EXPERT = "CREATE_EXPERT";
19  export const RETRIEVE_EXPERTS = "RETRIEVE_EXPERTS";
20  export const UPDATE_EXPERT = "UPDATE_EXPERT";
21  export const DELETE_EXPERT = "DELETE_EXPERT";
22
23  export const REGISTER_SUCCESS = 'REGISTER_SUCCESS';
24  export const REGISTER_FAILURE = 'REGISTER_FAILURE';
25  export const LOGIN_SUCCESS = 'LOGIN_SUCCESS';
26  export const LOGIN_FAILURE = 'LOGIN_FAILURE';
27  export const LOGOUT_SUCCESS = 'LOGOUT_SUCCESS';
28  export const LOGOUT_FAILURE = 'LOGOUT_FAILURE';
29
30
31  export const FETCH_ACCOUNTS_SUCCESS = "FETCH_ACCOUNTS_SUCCESS";
32  export const FETCH_ACCOUNTS_FAILURE = "FETCH_ACCOUNTS_FAILURE";
33  export const CREATE_ACCOUNT_SUCCESS = "CREATE_ACCOUNT_SUCCESS";
34  export const CREATE_ACCOUNT_FAILURE = "CREATE_ACCOUNT_FAILURE";
35  export const UPDATE_ACCOUNT_SUCCESS = "UPDATE_ACCOUNT_SUCCESS";
36  export const UPDATE_ACCOUNT_FAILURE = "UPDATE_ACCOUNT_FAILURE";
37  export const DELETE_ACCOUNT_SUCCESS = "DELETE_ACCOUNT_SUCCESS";
38  export const DELETE_ACCOUNT_FAILURE = "DELETE_ACCOUNT_FAILURE";
```

Obrázok 33: Types.js implementácia

6.4.4 Services a Axios

Axios je volaný v súbore http-common.js nasledovne: URL je nastavená na *"http://localhost:8080/api",* čo znamená, že všetky žiadosti odoslané touto inštanciou budú mať túto URL ako základný prefix. Na tomto porte beží lokálne backendová časť implementácie.

Riadok *instance.defaults.headers.common["Content-Type"] = "application/json";* nastavuje predvolený obsah hlavičky *"Content-Type"* pre všetky žiadosti odosielané touto inštanciou na *"application/json"*.

src > JS http-common.js > ...

```
1  import axios from "axios";  55.4k (gzipped: 20.2k)
2
3    const instance = axios.create({
4      baseURL: "http://localhost:8080/api",
5    });
6    instance.defaults.headers.common["Content-Type"] = "application/json";
7    export default instance;
```

Obrázok 34: Práca s Axios

SkillOptDataService pre spravované príležitosti na získanie vzdelania a zručností , ktorá slúži na komunikáciu s backendovým API, konkrétne pre entitu "Skill Opportunity" je definovaná nasledovne:

- obsahuje rôzne metódy, ktoré vykonávajú žiadosti HTTP na rôzne endpointy API súvisiace s "Skill Opportunities". Každá metóda vracia výsledok asynchrónnej HTTP žiadosti získanej z importovanej inštancie Axios, ktorá bola definovaná v súbore http-common.js.
- *getAllSkillOpts():* metóda na získanie všetkých "Skill Opportunities".
- getSkillOptById(id): metóda na získanie konkrétneho "Skill Opportunity" podľa ID.
- *createSkillOpt(data):* metóda na vytvorenie "Skill Opportunity".
- *updateSkillOpt(id, data):* metóda na aktualizácia existujúceho "Skill Opportunity".
- *deleteSkillOpt(id):* metóda na odstránenie "Skill Opportunity" s daným ID.
- ďalšie metódy na získanie "Skill Opportunities" na základe rôznych filtrov ako podľa kategórie, krajiny, titulu, organizátora, dátumu.

src > service > JS Skill.service.js > ...

```
1  import http from "../http-common";
2
3  class SkillOptDataService {
4    getAllSkillOpts() {
5      return http.get("/skill-opportunities");
6    }
7
8    getSkillOptById(id) {
9      return http.get(`/skill-opportunities/${id}`);
10   }
11
12   createSkillOpt(data) {
13     return http.post("/skill-opportunities", data);
14   }
15
16   updateSkillOpt(id, data) {
17     return http.put(`/skill-opportunities/${id}`, data);
18   }
19
20   deleteSkillOpt(id) {
21     return http.delete(`/skill-opportunities/${id}`);
22   }
23
24   getSkillOptsByAccountId(accountId) {
25     return http.get(`/skill-opportunities/by-account/${accountId}`);
26   }
27
28   getSkillOptByBiotechCategory(biotechCategoryName) {
29     return http.get(
30       `/skill-opportunities/by-biotech-category/${biotechCategoryName}`
31     );
32   }
33
34   getSkillOptBySkillCategory(skillCategoryName) {
35     return http.get(
36       `/skill-opportunities/by-skill-category/${skillCategoryName}`
37     );
38   }
39
40   getSkillOptByCountry(countryName) {
41     return http.get(`/skill-opportunities/by-country/${countryName}`);
42   }
43
44   getSkillOptByTitle(title) {
45     return http.get(`/skill-opportunities/by-title/${title}`);
46   }
47
48   getSkillOptByOrganizer(organizer) {
49     return http.get(`/skill-opportunities/by-organizer/${organizer}`);
50   }
51
52   getSkillOptByStartDate(startDate) {
53     return http.get(`/skill-opportunities/by-start-date/${startDate}`);
54   }
55
```

Obrázok 34: Ukážka implementácie *SkillOptDataService*

6.4.5 Components

Pri komponentovej architektúre vychádzala bakalárska práca z princípu *Atomic Design* popísaný v predchádzajúcich kapitolách. Štruktúra komponentov tomu odpovedá a sú delené do priečinkov *atoms, molecules, organisms, templates a pages*. Následne sú v nich delené do sekcií podľa obsahu na:

- *common* - ak ide o všeobecný komponent,
- *skills* - komponenty vytvárajúce výsledné UI pre sekciu Skills,
- *startups* - komponenty vytvárajúce výsledné UI pre sekciu Startup Support
- *network* - obsahujúci komponenty pre inovácie a expertov z ekosystému, ktorí sú zobrazení na podstránke Network,
- users - komponenty zabezpečujúce prácu s účtami, registráciu a prihlasovanie.

Na nasledujúcich vybraných komponentoch vidno aplikáciu v praxi.

Komponenta *ButtonAdd* je atómovým prvkom a slúži na návrh jednoduchého tlačidla s popisom Add. Pri kliknutí na tlačidlo sa spustí zadaná funkcia *onClick*. Tlačidlo je štýlované pomocou CSS triedy *styles.btnUpdate*, ktorá určuje vzhľad tlačidla. Tento vzhľad je rovnaký pre tlačidlo *Add* aj *Update*. Komponent je exportovaný pre použitie v iných častiach aplikácie.

src > components > atoms > common > JS ButtonAdd.js > ...

```
1   import React from "react";   6.9k (gzipped: 2.7k)
2   import styles from "./Buttons.module.css";
3
    Complexity is 3 Everything is cool!
4   const ButtonAdd = ({ onClick }) => {
5     return (
6       <button className={styles.btnUpdate} onClick={onClick}>
7         Add
8       </button>
9     );
10  };
11
12  export default ButtonAdd;
```

Obrázok 36: Ukážka implementácie atomického komponentu

Komponenta *Pagination* je molekulou a slúži na zobrazenie ovládacích prvkov pre stránkovanie. Komponenta prijíma tri vstupné premenné - číslo aktuálnej stránky *currentPage*, celkový počet stránok *totalPages* a funkciu *onPageClick*, ktorá sa vyvolá pri kliknutí na

niektoré tlačidlo / stránku. V implementácii komponentu sa vytvára pole *pageNumbers*, ktoré obsahuje čísla stránok od 1 po *totalPages* a tie sa následne mapujú a pre každú stránku sa vytvára tlačidlo. Pri každom tlačidle sa nastavuje identifikátor id na príslušné číslo stránky a ako funkcia je nastavená *onPageClick*. Ďalej sú v komponente volané štýly cez konvenciu *styles.pageButton* a *styles.active.* To zabezpečuje jednotný štýl implementovanej Pagination komponentne kdekoľvek na stránke.

src > components > molecules > common > JS Pagination.js > ...

```
1   import React from "react";  6.9k (gzipped: 2.7k)
2   import styles from "./Pagination.module.css";
3
    Complexity is 8 It's time to do something...
4   const Pagination = ({ currentPage, totalPages, onPageClick }) => {
5     const pageNumbers = Array.from({ length: totalPages }, (_, i) => i + 1);
6
7     return (
8       <div className={styles.pagination}>
9         <div className={styles.paginationButtons}>
            Complexity is 3 Everything is cool!
10          {pageNumbers.map((number) => (
11            <button
12              key={number}
13              id={number}
14              onClick={onPageClick}
15              className={`${styles.pageButton} ${
16                currentPage === number ? styles.active : ""
17              }`}
18            >
19              {number}
20            </button>
21          ))}
22        </div>
23      </div>
24    );
25  };
26
27  export default Pagination;
28
```

Obrázok 37: Ukážka implementácie molekulárneho komponentu

src > components > pages > JS SkillPage.js > ...

```
1   import React, { Component } from "react";  6.9k (gzipped: 2.7k)
2   import SkillOptList from "../templates/skills/SkillOptList";
3   import CenteredTextWithButton from "../organisms/common/CallToActionComponnetAccordingRole";
4   import CarouselBanner from "../organisms/skills/SkillCarousel.banner";
5   import styles from "./Page.module.css";
6
    Complexity is 8
7   class SkillOptPage extends Component {
      Complexity is 8
8     render() {
9       return (
10        <div>
11          <CarouselBanner />
12
13          <div className={styles["intro"]}>
14            <p>
15              Welcome to How To Do Biotech, your one-stop platform for finding
16              skill development opportunities in the biotech industry. Our
17              platform is designed to make it easy for you to find the courses,
18              workshops, and events that best suit your needs to improve your skills. Whether you're a
19              student, a professional, or an PhD researcher looking to enhance your
20              biotech skills, you can find the right opportunity here.
21            </p>
22          </div>
23
24          <SkillOptList />
25          <CenteredTextWithButton
26          textPublic="Are you an organizer of biotech skill development courses, workshops, or events?
27          Are you looking for interns and researchers? We invite you to sign up to our platform and create
28          an account for your organization. Add your workshops, internships, or hackathons to our database
29           and attract young talents in science through our platform. Join us today and be a part of the biotech
30           skill development revolution!"
31          urlPublic="/register"
32          buttonTextPublic="create account"
33          textLoggedIn="Great! You are logged in your organization account. Now you can create new skills opprortur
34          urlLoggedIn="/skill-opportunities/add"
35          buttonTextLoggedIn="add opportunity"
36          />
37        </div>
38      );
39    }
40  }
41
42  export default SkillOptPage;
```

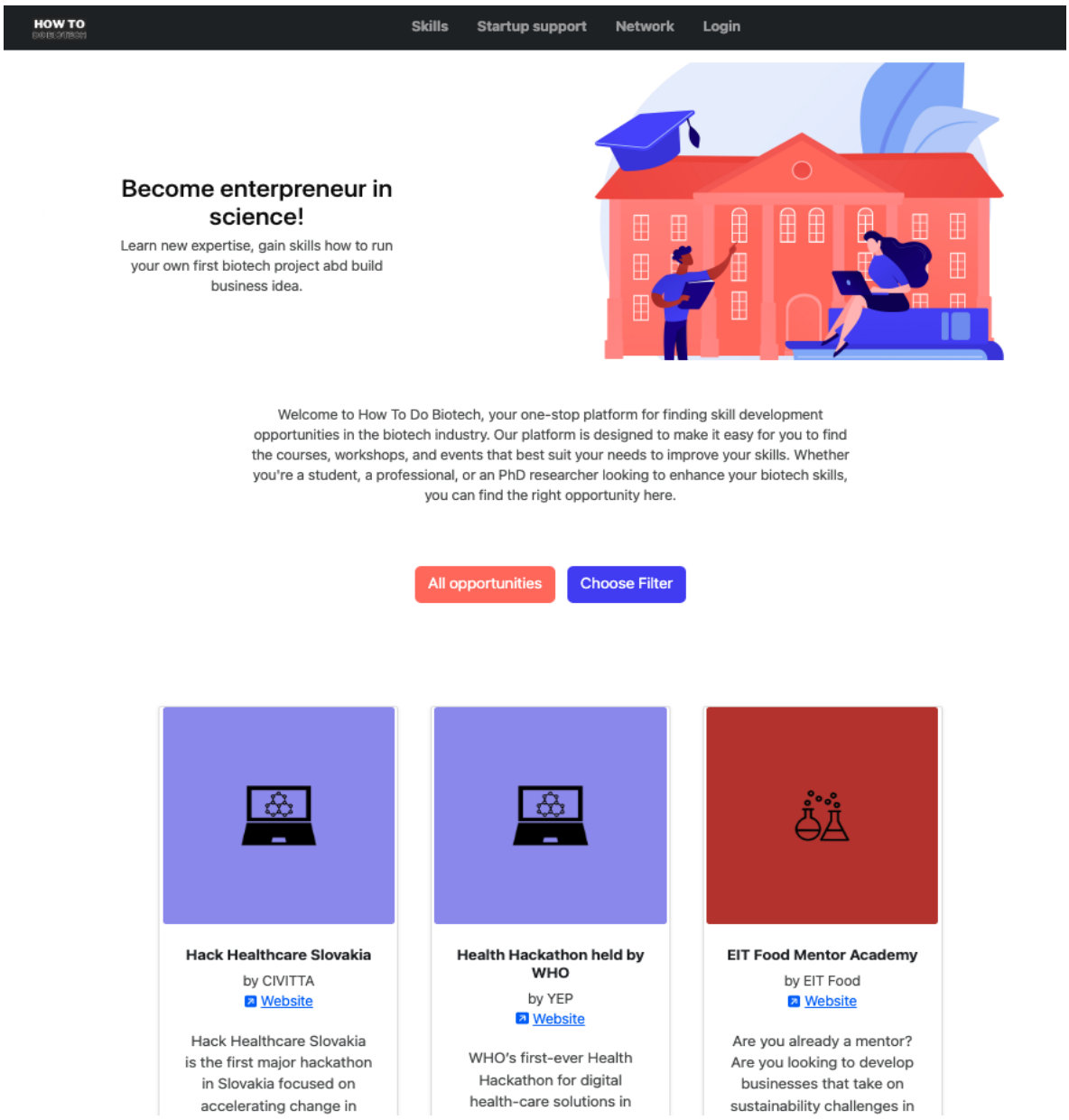
Obrázok 38: Ukážka implementácie page

6.5 UI rozhranie aplikácie

Na nasledujúcich stranách sa bakalárska práca venuje užívateľskému rozhraniu. Výber jednotlivých komponentov je uvedený podľa skupín používateľských prípadov, ktoré napĺňa. Najprv je zobrazené rozhranie s detailmi pre neprihláseného užívateľa - návštevníka. Následne je zobrazené rozhranie pre prihlásenú organizáciu.

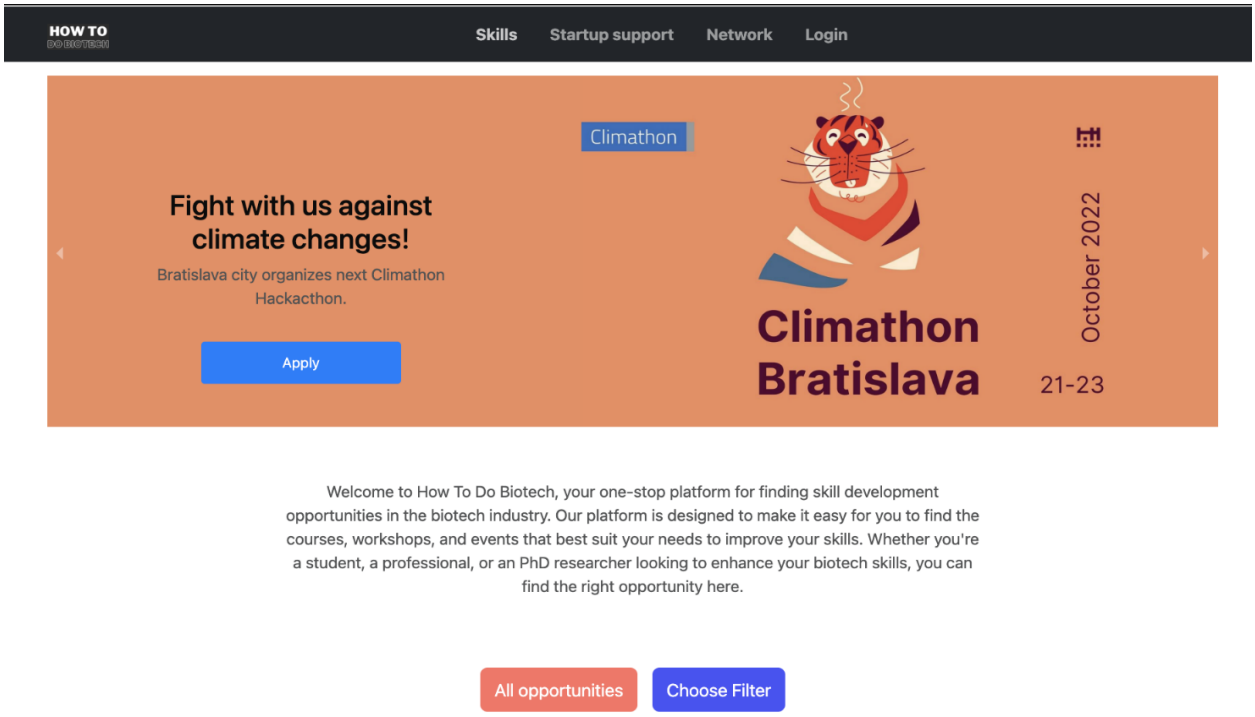
6.5.1 Návštevník

Po načítaní stránky z lokálneho prostredia vidí v prototype návštevník 3 základné možnosti - sekcie obsahu: Skills, Startup Support a Network. Medzi nimi sa môže jednoducho preklikávať na hornom navigačnom panely.



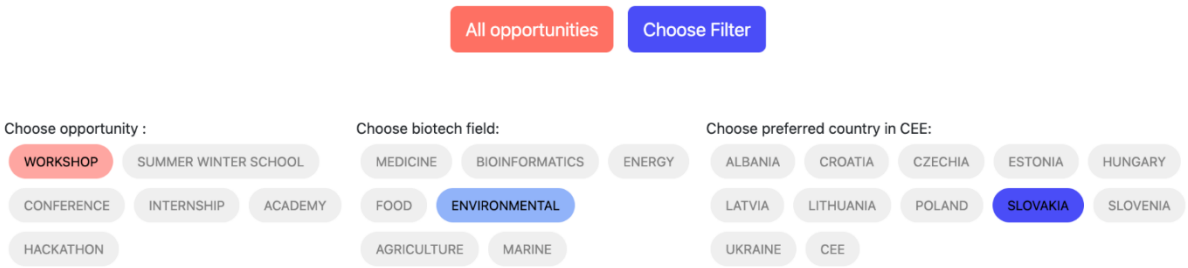
Obrázok 39: Sekcia Skills

Carousel je dizajnový prvok, ktorý je implementovaný v sekcii Skills a Startup Support. Dynamický mení obsah a dá sa využiť na zvýraznenie aktuálnych termínov na využitie konkrétnej schémy podpory pre vedecko-podnikateľské tímy a blížiace sa termíny na prihlásenie do letných škôl, workshopov či stáží.

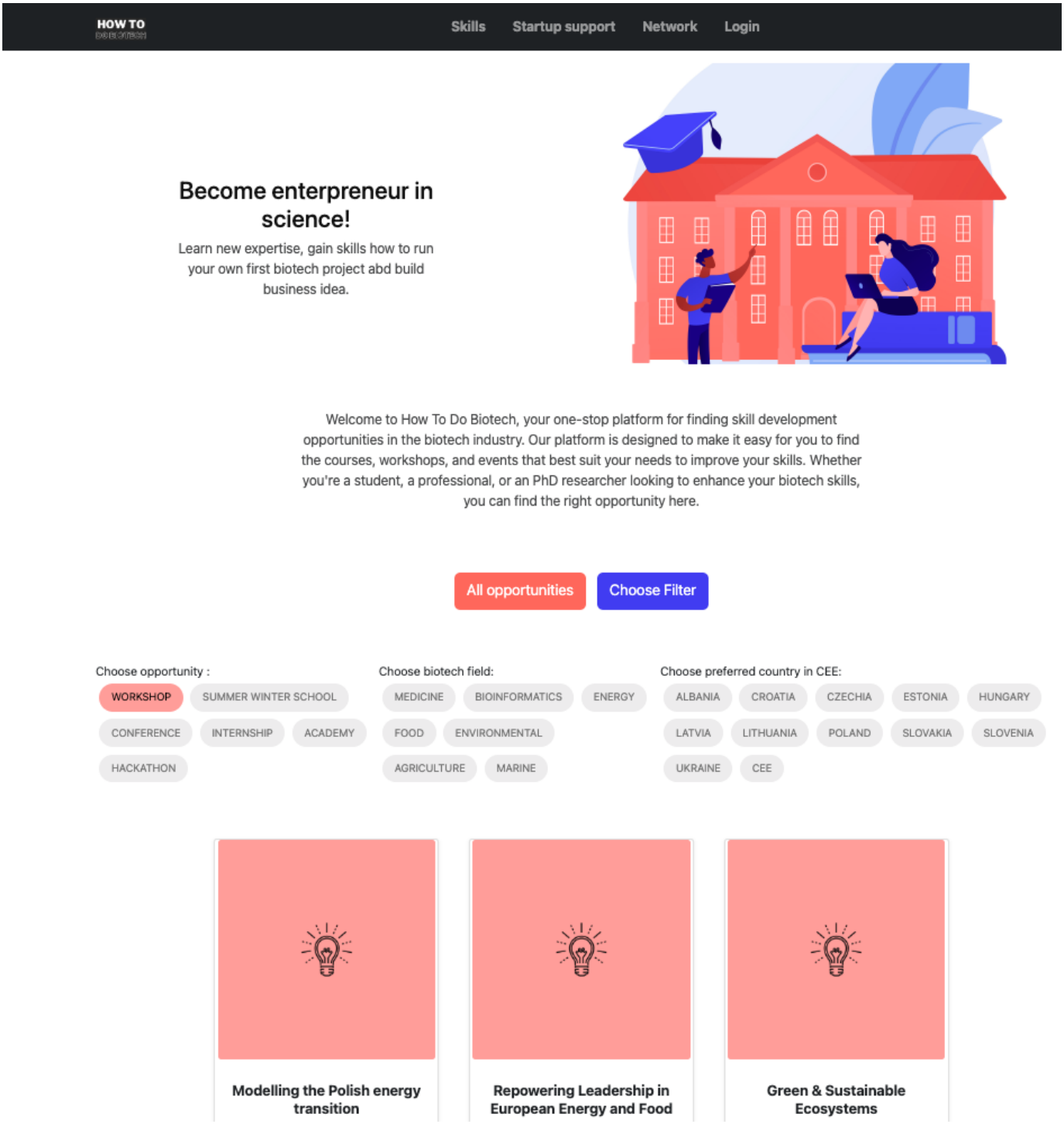


Obrázok 40: Carousel Banner - dynamická zmena

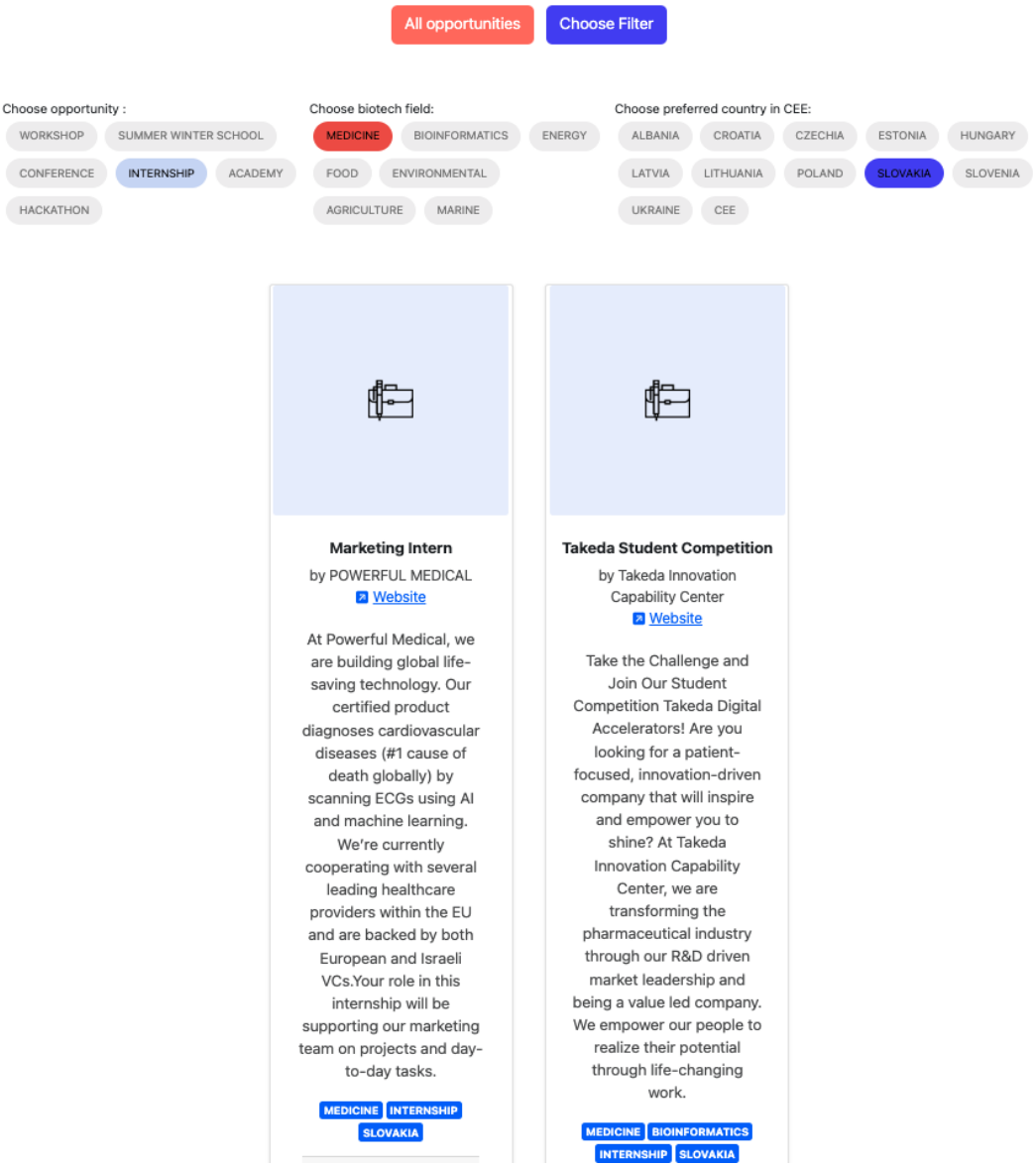
Databázy sa dajú filtrovať podľa rôznych parametrov ako sú krajiny, biotechnologické odvetvie, druh poskytovanej podpory, druh podujatia pre získanie zručností alebo podľa expertízy pri sekcie Network. Po kliknutí na tlačidlo *Choose Filter* sa zobrazia možnosti pre danú sekciu. Filtrovať sa dá vybraním jedného alebo viacerých filtrov.



Obrázok 41: Filter - podstránka Skills

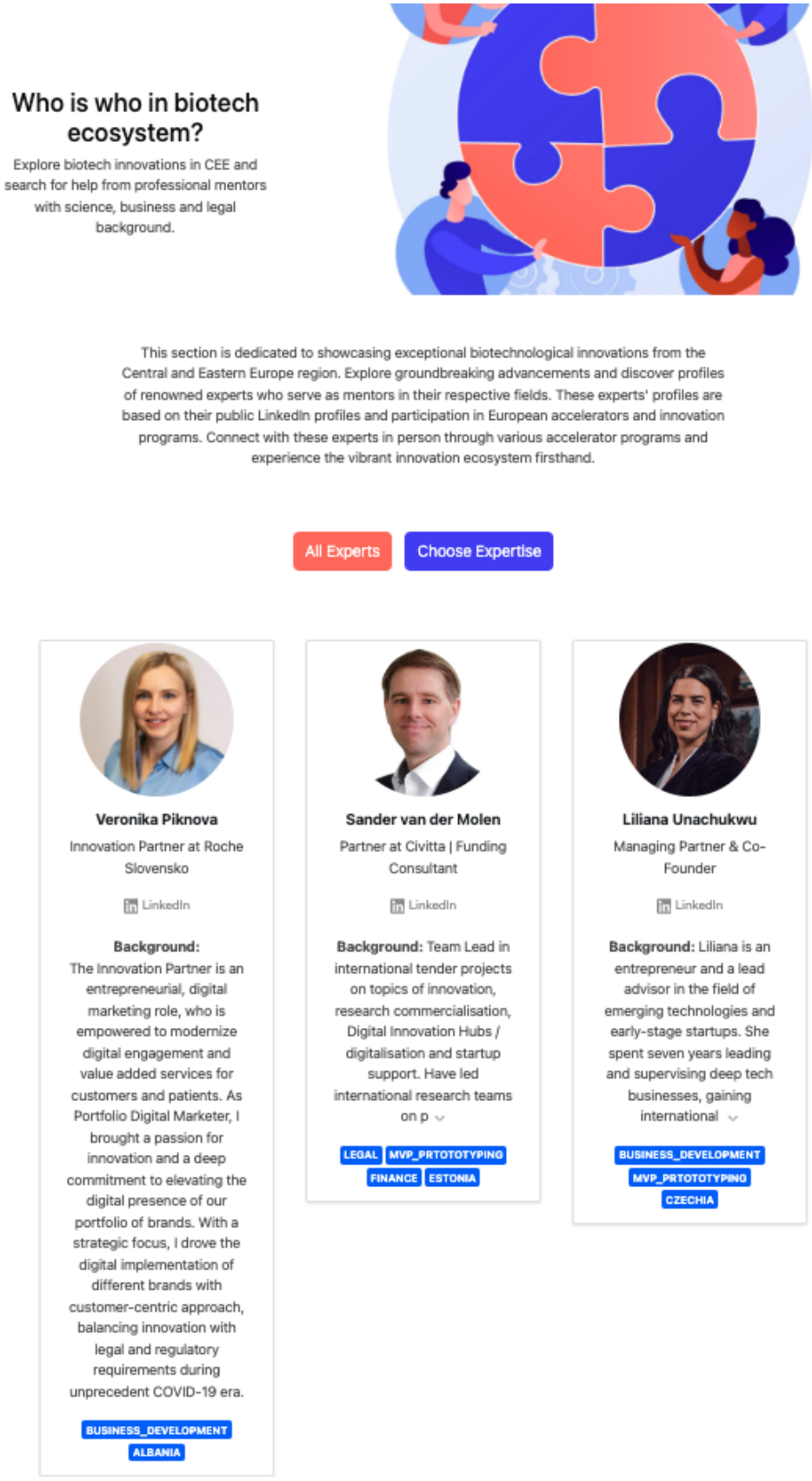


Obrázok 42: Filter - aplikácia jedného parametra



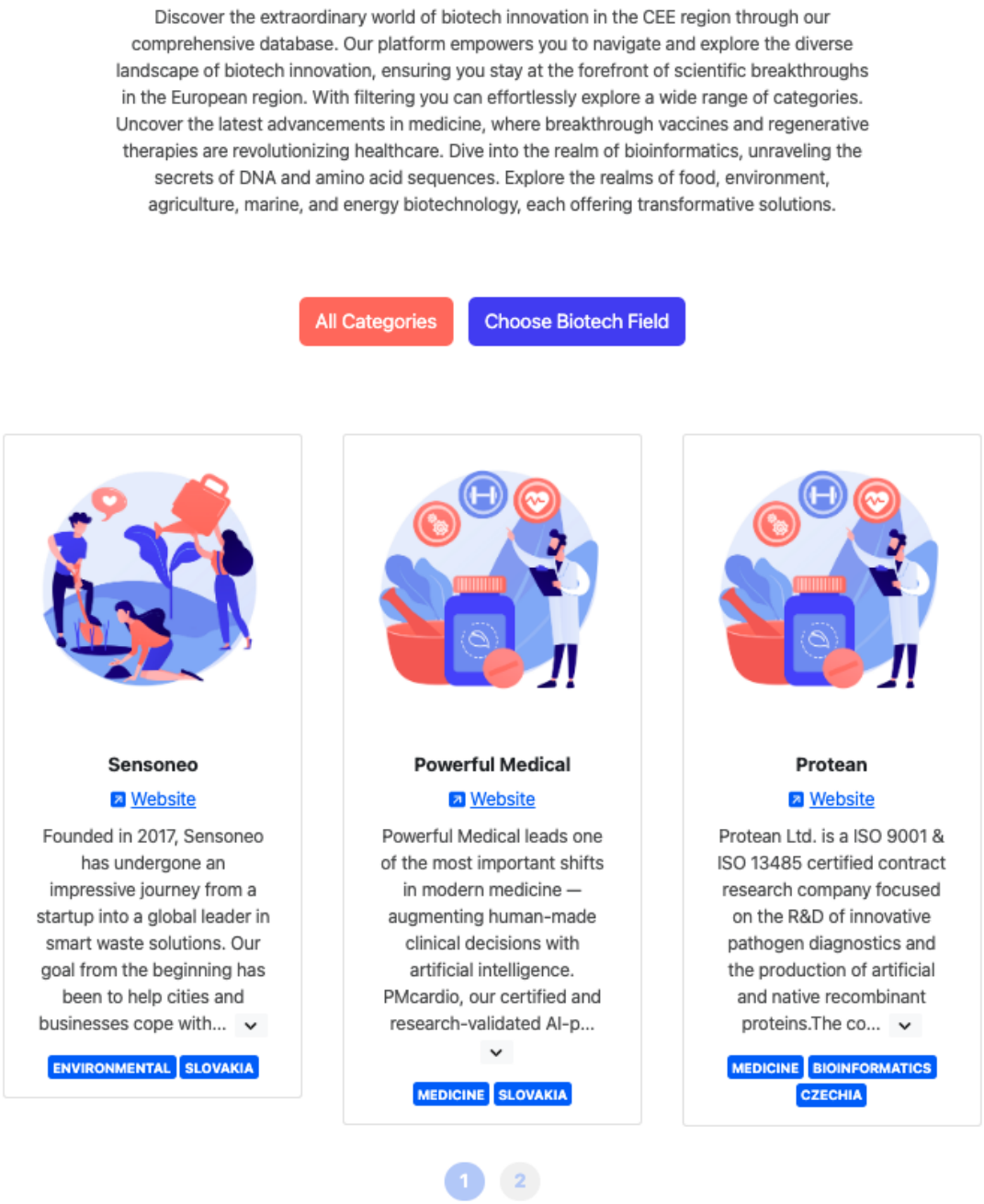
Obrázok 43: Filter - aplikácia viacerých filtrových parametrov

V aplikácii vo veľkom využívame "Cards", karty s obsahom, či už ide o profily mentorov alebo informačné karty o inováciách, podpore a príležitostiach. Obrázok 44 zobrazuje využitie kariet pre expertov s fotkami sú doťahované z AWS S3 bucket cez databázu H2. Obsahujú ovládací prvok *collaps*, ktorý umožňuje zobraziť presne daný rozsah textu a v prípade potreby ho rozrolovať.



Obrázok 44: Cards & Collaps

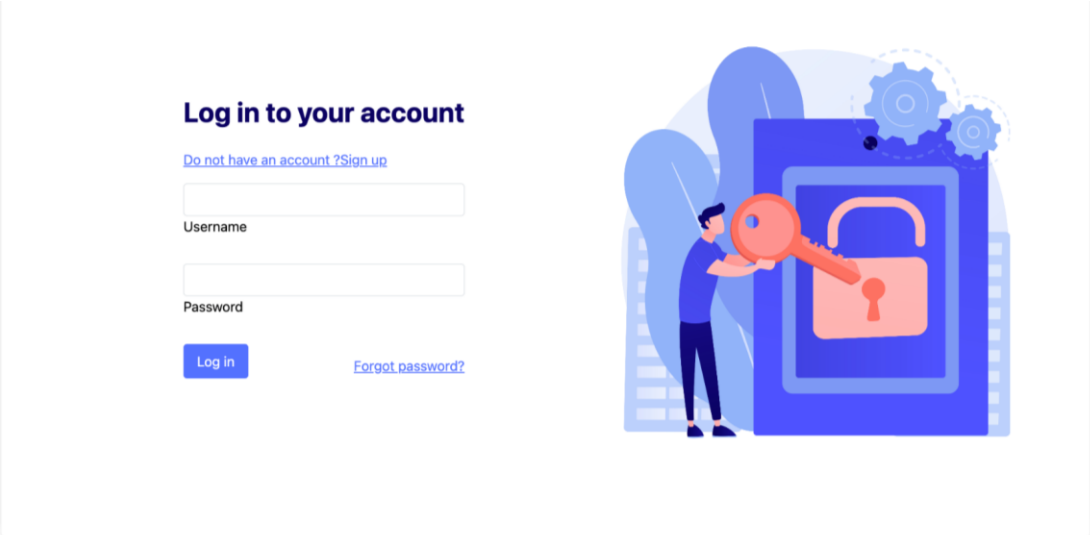
Na obrázku 45 je implementovaný celý komponent typu Template - list inovácii *InnovationList*, s úvodným textom, filtrom, jednotlivými kartami o inováciách aj *Pagination* komponentom na stránkovanie listu inovácii.



Obrázok 45: Template komponent Innovations

6.5.2 Registrácia a Prihlásenie

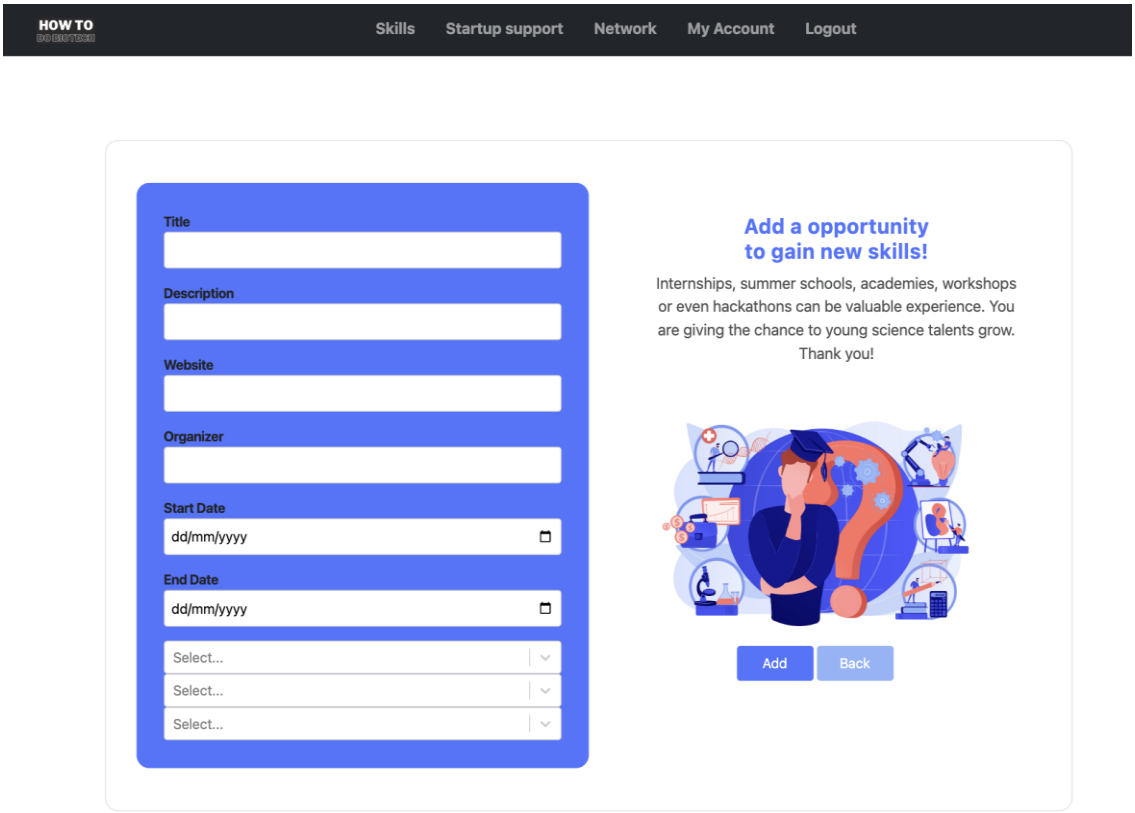
Pre prihlásenie sa a registráciu sú vytvorené dva formuláre, ktorých ukážky sú nižšie. Podľa toho, či je používateľ prihlásený alebo nie sa mu na navigačnom panely zjaví príslušná možnosť. Jednotlivé prvky ako *Input field*, tlačidla a pod. sú implementované s dizajnom knižnice *mdb-react-ui-kit.*



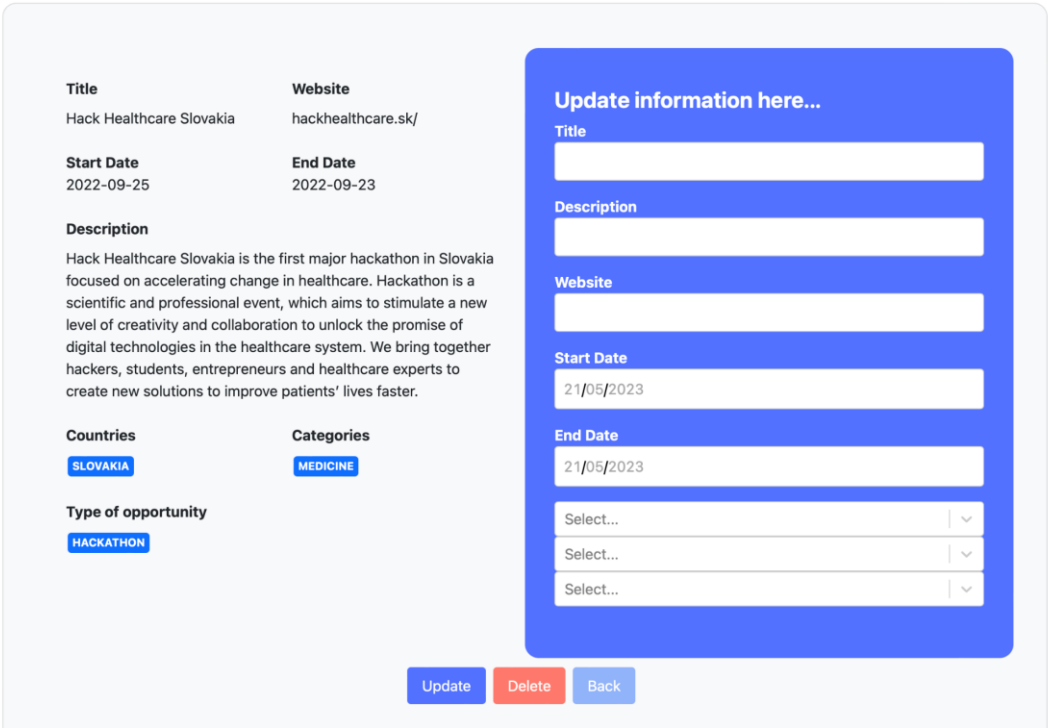
Obrázok 46: UI implementácia Prihlásenia sa

6.5.3 Prihlásená organizácia

Po registrácii a následne prihlásení do svojho účtu má možnosť organizácia pridávať, upravovať a mazať položky do databázy príležitostí a podpory. V aplikácii je zavedených niekoľko formulárov, v ktorých si používateľ vyberá dátum, konkrétne kategórie zo zoznamov, alebo vpisuje osobitné údaje do textových polí.



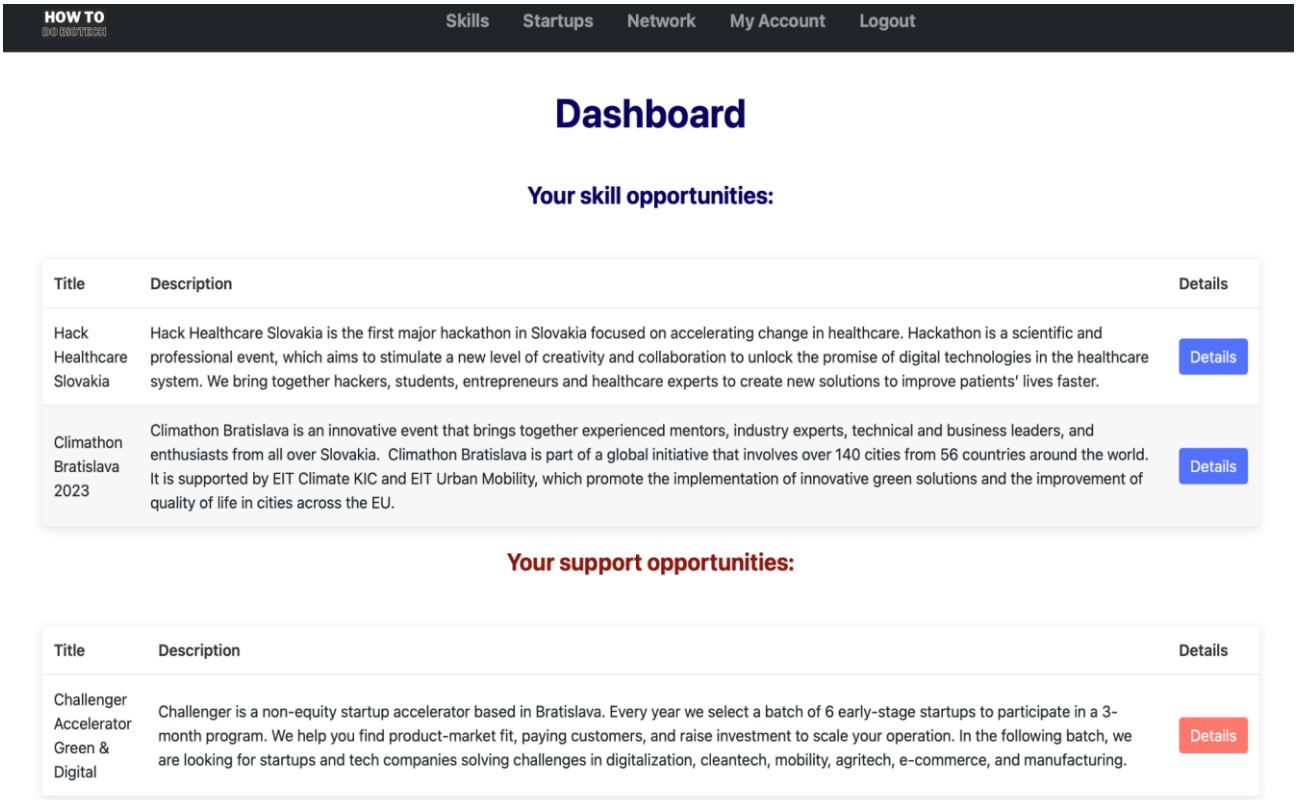
Obrázok 47: UI implementácia formuláru Add



Obrázok 48: UI implementácia formuláru Update

6.5.4 Prihlásená organizácia Dashboard

Po registrácii a následne prihlásení do svojho účtu má možnosť organizácia vidieť unikátny dashboard s informáciami o pridaných príležitostiach vytvorených daným účtom.



Obrázok 49: UI implementácia dashboardu

Vidí na ňom zoznam svojich záznamov a cez tlačidlo detail vie pristúpiť k detailnému popisu na karte danej položky. Odtiaľ sa vie dostať k editovaniu alebo mazaniu záznamu.

Každý typ používateľa zároveň vidí chybové hlášky, alert hlášky po ukončení úspešnej alebo neúspešnej editácie a vkladania nových záznamov ako aj *Page Not Found*, ak vyberie vo filtri kombináciu, na ktorú sa v databáze nenachádza žiadna položka

Obrázok 50: UI implementácia Page Not Found

6.6 Bezpečnosť

Jenou z hlavných implementácii bezpečného autorizovaného prístupu k obsahu je implementácia JWT tokenu na frontend aj backend časti aplikácie.

Na backend časti je vytvorená trieda filtra *AuthJwtRequestFilter,* ktorý slúži na overenie a spracovanie JWT autentifikácie v každom prichádzajúcom HTTP požiadavku. V úvode si trieda injektuje závislosti ako *JwtUtils* s pomocnými metódami na prácu s JWT a *UserDetailsService* rozhranie, ktoré zabezpečuje získanie detailov používateľa, v tomto prípade účtu account.

Filter obsahuje metódu *doFilterInternal* , ktorá implementuje filter a vykonáva spracovanie každej prichádzajúcej HTTP požiadavky. Získava sa JWT z hlavičky, kontroluje, či v hlavičke požiadavky existuje token a či začína s reťazcom "*Bearer*". Nasleduje overenie a spracovanie JWT tokenu. Ak je token prítomný a platný a overenie prebehlo úspešne, vytvára

objekt *UsernamePasswordAuthenticationToken* s používateľskými detailmi a nastavuje autentifikáciu v SecurityContextHolder.

```
class App extends Component {
  constructor(props) {
    super(props);
    this.state = {
      isAuthenticated: !!localStorage.getItem("authToken"),
    };
    this.logout = this.logout.bind(this);
    this.login = this.login.bind(this);
  }

  logout() {
    localStorage.removeItem("authToken");
    this.setState({ isAuthenticated: false });
  }

  login() {
    this.setState({ isAuthenticated: true }, () => {
      window.location.reload();
    });
  }
```

Obrázok 51: JWT Token spracovaný na frontende

Ďalšou oblasťou, spomínanou v predchádzajúcich kapitolách je využitie JPA pri práci s dátami. Jej implementácia je načrtnutá v kapitolách 6.2.3 a 6.2.4. Dôležitou je aj validácia na frontendovej časti.

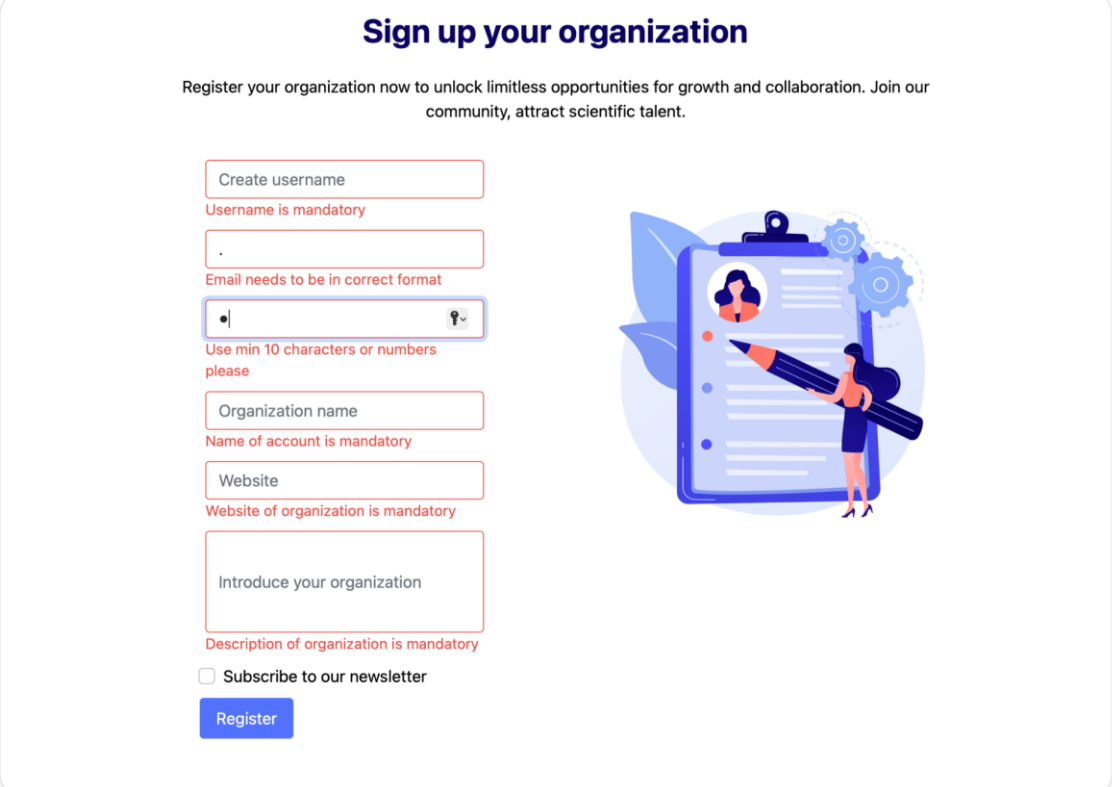
Vďaka knižniciam *formik a yup* je nastavená validácia napríklad na registračnom formulári. Či už ide o maximálny a minimálny počet znakov, o vyžadujúce informácie alebo pole pre email, ktorého výraz musí byť v korektnom emailovom formáte.

```
</div>
{formik.touched.password && formik.errors.password && (
  <div className={`${styles.error} ${styles.mt2}`}>
    {formik.errors.password}
  </div>
)}
```

Obrázok 52: Implementácia validačných hlášok

```
const validationSchema = Yup.object().shape({
  name: Yup.string()
    .required("Name of account is mandatory")
    .min(3, "Name should be at least 3 characters long")
    .max(14, "Name should not exceed 14 characters"),
  description: Yup.string()
    .required("Description of organization is mandatory")
    .max(400, "Description should not exceed 400 characters"),
  url: Yup.string().required("Website of organization is mandatory"),
  email: Yup.string()
    .required("Email is mandatory")
    .email("Email needs to be in correct format"),
  username: Yup.string()
    .required("Username is mandatory")
    .min(3, "Username should be at least 3 characters long")
    .max(14, "Username should not exceed 14 characters"),
  password: Yup.string().min(
    10,
    "Password should be minimum 10 characters or numbers long"
  ),
});
```

Obrázok 53: Ukážka schémy validácie



Obrázok 54: UI rozhranie registrácie s validáciou

Vďaka implementácii rozličných rol vie aplikácia zobrazovať obsah podľa pridelených právomocí. Na backend časti je to ošetrené anotáciou *@PreAuthorize* ktorá určuje, ku ktorým endpointom bude mať prístup iba admin.

```
no usages   baskaklimek
@PutMapping("/{id}")
@PreAuthorize("hasRole('ROLE_ADMIN')")
public ResponseEntity<?> updateInnovation(@PathVariable("id") Integer id, @RequestBody Innovation innovation) {
    try {
        innovationService.updateInnovation(id, innovation);
        return new ResponseEntity<>(HttpStatus.OK);
    } catch (Exception e) {
        return new ResponseEntity<>(HttpStatus.NOT_IMPLEMENTED);
    }
}
```

Obrázok 55: Správa obsahu na základe role

Na frontendovej časti je obsah a jeho sprístupnenie riešené práve JWT tokenom a boolean premennou *isAuthenitcated* a *PrivateRoute* komponentnou.

```
Complexity is 6
const PrivateRoute = ({ render: RenderComponent, isAuthenticated, ...rest }) => (
  <Route
    {...rest}
    Complexity is 4
    render={(props) => ■
      isAuthenticated ? (
        <RenderComponent {...props} />
      ) : (
        <Redirect to="/login" />
      )
    }
  />
);
```

Obrázok 56:: Private Route komponent

```
<Route path="/skill-opportunities/update/:id" component={SkillOptUpdateForm} />
<Route path="/startup-opportunities/update/:id" component={StartupOptUpdateForm} />
<Route path="/accounts/update/:id" component={AccountUpdateForm} />
<PrivateRoute path="/startup-opportunities/:id" component={StartupOpt} />
<PrivateRoute path="/skill-opportunities/:id" component={SkillOpt} />
<PrivateRoute path="/accounts/:id" component={Account} />

<PrivateRoute path="/experts/:id" component={Expert} />
<PrivateRoute path="/innovations/:id" component={Innovation} />
```

Obrázok 57: Routing v App.js

6.7 Testovanie

Pri prototype aplikácii sa dal dôraz najmä na testovanie wireframe rozloženia a naplnenia požiadaviek s potenciálnymi budúcimi užívateľmi s radov študentov prírodovedeckých odborov a implementáciu unit testov. Prototyp sa bude ďalej testovať, aby sa vyladila jeho funkčnosť z UX aj UI hľadiska.

Testovanie wireframe obrazoviek je dôležité, pretože umožňuje identifikovať potenciálne problémy alebo nedostatky v návrhu užívateľského rozhrania ešte pred jeho implementáciou. Toto testovanie sa zameriava na overenie zrozumiteľnosti a použiteľnosti konceptuálneho návrhu.

Pri unit testoch sa využili knižnice *Mockito* a *JUnit* ako vidno na obrázku 60.

```
@Test
void testGetAllStartupOpts() {
    when(startupOptService.selectAllStartupOpts()).thenReturn(startupOptList);

    // Call the controller's getAllStartupOpts() method
    ResponseEntity<List<StartupOpt>> response = startupOptController.getAllStartupOpts();

    // Verify that the response contains the startup opportunities
    assertNotNull(response);
    assertEquals(HttpStatus.OK, response.getStatusCode());
    List<StartupOpt> responseStartupOpts = response.getBody();
    assertNotNull(responseStartupOpts);
    assertEquals( expected: 1, responseStartupOpts.size());
    assertEquals(startupOptList.get(0), responseStartupOpts.get(0));
}
```

Obrázok 58: Implementácia unit testu

V tomto testovacom prípade sa testovala metóda getAllStartupOpts(). Použitý je k tomu mock objekt *startupOptService*, ktorý sa nastavil tak, že pri volaní metódy *selectAllStartupOpts*() vráti zoznam *startupOptList*. Následne sa zavolala testovaná metóda a overí sa, že odpoveď obsahuje správne startup príležitosti. Skontroluje sa, či odpoveď nie je null, či jej stavový kód je *HttpStatus.OK*, či obsahuje zoznam startup príležitostí a či sa zhoduje s prvým prvkom *startupOptList*.

Na začiatku testovacej triedy sme si vytvorili anotáciami *@InjectMocks a @Mock* objekty servisy a kontroléra. Následne sme v časti @BeforeEach definovali želaný objekt so všetkými vlastnosťami, s ktorým sme následne pracovali.

7 BUDÚCÍ VÝVOJ APLIKACE

Momentálne je projekt bakalárskej práce nastavený ako prototyp MVP, teda doručuje používateľ pokrytie základných potrieb. Táto práca má však ambíciu nasadenia na doménu howtodobiotech.com v najbližších mesiacoch a zapracovania zmien po odskúšaní a otestovaní užívateľmi. Na backendovej časti sú predpripravené viaceré štruktúry k použitiu. Rovnako dizajn a UX webovej stránky prejdú ešte výraznejšími zmenami tak, aby sa zabezpečila jednoduchá orientácia používateľov.

Databáza je naplnená prvými pilotnými dátami, na spracovanie však čaká omnoho väčší objem údajov, ktoré sú už zozbierané a potrebujú byť upravené do podoby vhodnej pre relačnú databázu projektu.

Predpokladá sa, že webová aplikácia sa bude stále dynamicky vyvíjať a že sa postupne zapracuje možnosť vkladať aj obrázky k uverejneným príležitostiam a podpore. Na to je nachystané prostredie AWS S3 a stačí doplniť konfiguráciu na backendovú stranu a endpoint pre vloženie dátového typu obrázok.

ZÁVER

Bakalárska práca sa zaoberala návrhom a implementáciou webovej aplikácie pre výskumníkov a podnikateľov v oblasti biotechnológií v regióne CEE. Cieľom práce bolo vytvoriť platformu, ktorá umožňuje prepojenie a spoluprácu medzi aktérmi inovačného biotechnologického ekosystému, ako sú výskumní pracovníci, vzdelávacie a investičné inštitúcie a podnikatelia. Aplikácia poskytuje užívateľom informácie o dostupnej podpore v oblasti biotechnológií v regióne CEE, zviditeľňuje etablované produkty a služby v tejto oblasti a spája podnikateľské a startup tímy s výskumníkmi a expertmi..

Práca sa venuje prípadom použitia a funkcionálnym a funkcionálnym požiadavkám na prototyp aplikácie, pričom sa berie do úvahy prieskum trhu

V teoretickej časti sú diskutované rôzne technológie, testovanie aplikácií a princípy zabezpečenia.

Analýza populárnych technológií slúži ako základ pre výber vhodných technológií pre backend a frontend aplikácie

Praktická časť sa zameriava na potrebu a požiadavky aplikácie. Analyzuje motiváciu a popisuje projekt, cieľové skupiny a zainteresované strany. Definuje funkcionálne a nefunkcionálne požiadavky a skúma aktérov a prípady použitia. Skúma existujúce informačné zdroje a navrhuje architektúru aplikácie, REST API model a dátový model. Vytvorené wireframey poskytujú predstavu o vzhľade aplikácie. Implementácia zahŕňa backend a frontend časť aplikácie vrátane konfigurácie, vývojového prostredia, modelov, služieb a komponentov.

Na základe výsledného prototypu aplikácie je možné aj naďalej rozširovať jej funkcionality a prispieť k zlepšeniu spolupráce medzi výskumníkmi a podnikateľmi v oblasti biotechnológií v regióne CEE a vzniku nových vedecko-podnikateľských inovácií.

ZOZNAM POUŽITEJ LITERATÚRY

[1] *2022 Developer Survey* [online]. 2023 [cit. 2023-01-14]. Dostupné z: https://survey.stackoverflow.co/2022/#technology-most-popular-technologies

[2] *2021 Developer Survey* [online]. 2022 [cit. 2023-01-14]. Dostupné z: https://insights.stackoverflow.com/survey/2021

[3] *Learning React:: A Hands-On Guide to Building Web Applications Using React and Redux*. 2nd. Boston: Addison-Wesley Professional, 2018. ISBN 978-0134843551.

[4] *Learn React: Describing the UI* [online]. [cit. 2022-12-15]. Dostupné z: https://react.dev/learn/describing-the-ui

[5] *The Progressive JavaScript Framework: What is Vue?* [online]. [cit. 2022-12-15]. Dostupné z: https://vuejs.org/guide/introduction.html

[6] *Introduction to the Angular docs: What is Angular?* [online]. [cit. 2023-01-14]. Dostupné z: https://angular.io/guide/what-is-angular

[7] *JQuery API* [online]. [cit. 2022-12-15]. Dostupné z: https://api.jquery.com

[8] *What is Svelte?* [online]. [cit. 2023-12-15]. Dostupné z: https://svelte.dev/tutorial/basics

[9] *Github: Most Stars Framework* [online]. [cit. 2023-05-14]. Dostupné z: https://github.com/topics/framework?o=desc&s=stars

[10] *Spring Boot* [online]. [cit. 2023-01-15]. Dostupné z: https://spring.io/projects/spring-boot

[11] Github: Most Forks Framework [online]. [cit. 2023-05-14]. Dostupné z: https://github.com/topics/framework?o=desc&s=forks

[12] HINKULA, Juha. *Hands-on Full Stack Development with Spring Boot 2 and React:: Build Modern and Scalable Full Stack Applications Using Spring Framework 5 and React with Hooks*. 2nd. Birmingham: Packt Publishing, 2019. ISBN 978-1-83882-236-1.

[13] *Django: Django appears to be a MVC framework, you call the Controller the "view", and the View the "template".*[online]. [cit. 2023-01-15]. Dostupné z: https://docs.djangoproject.com/en/4.2/faq/general/#django-appears-to-be-a-mvc-framework-but-you-call-the-controller-the-view-and-the-view-the-template-how-come-you-don-t-use-the-standard-names

[14] *Django Documentation: Models* [online]. [cit. 2023-01-15]. Dostupné z: https://docs.djangoproject.com/en/4.2/topics/db/models/

[15] *About Node.js®* [online]. [cit. 2023-01-18]. Dostupné z: https://nodejs.org/en/about

[16] KARASAVVAS, Theodoros. Node.js makes fullstack programming easy with server-side JavaScript. *Stackoverflow.blog* [online]. 2021-10-25 [cit. 2023-01-15]. Dostupné z: https://stackoverflow.blog/2021/10/25/node-js-makes-fullstack-programming-easy-with-server-side-javascript/

[17] MEIER, Andreas a Michael KAUFMANN. *SQL & NoSQL databases: models, languages, consistency options and architectures for Big data management*. Wiesbaden: Springer, [2019]. ISBN 978-3658245481.

[18] *Spring Boot Reference Documentation: Web* [online]. [cit. 2023-01-22]. Dostupné z: https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#documentation.web

[19] Spring initializr. In: *Spring.io* [online]. [cit. 2023-01-25]. Dostupné z: https://start.spring.io/

[20] *React Docs: Getting Started* [online]. [cit. 2023-01-22]. Dostupné z: https://legacy.reactjs.org/docs/getting-started.html

[21] *Material Design for Bootstrap* [online]. [cit. 2023-02-01]. Dostupné z: https://mdbootstrap.com/docs/react/

[22] *Material Design: What's Material?* [online]. [cit. 2023-02-01]. Dostupné z: https://m3.material.io/get-started

[23] *H2 Database Engine* [online]. [cit. 2023-02-02]. Dostupné z: https://h2database.com/html/main.html

[24] KINSBRUNER, Eran a Gleb BAKHMUTOV. *A Frontend Web Developer's Guide to Testing: Explore leading web test automation frameworks and their future driven by low-code and AI*. 1. Birmingham, UK: Packt Publishing, 2022, 304 s. ISBN 978-1803238319., p.12

[25] BAKLIWAL, Shagun. *Hands-on Application Development using Spring Boot: Building Modern Cloud Native Applications by Learning RESTFul API, Microservices, CRUD Operations, Unit Testing, and Deployment*. 1. New Delhi: BPB Publications, 2021, 348 s. ISBN 978-9391030223. p.77-93 .

[26] *JUnit 5: The 5th major version of the programmer-friendly testing framework for Java and the JVM* [online]. [cit. 2023-02-10]. Dostupné z: https://junit.org/junit5/

[27] *Mockito: Tasty mocking framework for unit tests in Java* [online]. [cit. 2023-02-10]. Dostupné z: https://javadoc.io/doc/org.mockito/mockito-core/latest/org/mockito/Mockito.html

[28] *Best Practices for Unit Testing in Java* [online]. 2023-05-05 [cit. 2023-05-10]. Dostupné z: https://www.baeldung.com/java-unit-testing-best-practices

[29] MORAN, Kate. Usability Testing 101. In: *Nielsen Norman Group*[online]. 2019, 2019-12-01 [cit. 2023-05-15]. Dostupné z: https://www.nngroup.com/articles/usability-testing-101/

[30] CANZIBA, Elvis. *Hands-On UX Design for Developers: Design, prototype, and implement compelling user experiences from scratch*. 1. Birmingham, UK: Packt Publishing, 2018, 350 s. ISBN 978-1788-624299.

[31] TANYA, Janca. Three layers to secure a software development organization. In: *Stackoverflow blog* [online]. 2023, 2023-02-09 [cit. 2023-05-15]. Dostupné z: https://stackoverflow.blog/2023/02/09/three-layers-to-secure-a-software-development-organization/

[32] *Cross-Site Scripting in Frontend* [online]. [cit. 2023-05-05]. Dostupné z: https://knowledge-base.secureflag.com/vulnerabilities/cross_site_scripting/cross_site_scripting_frontend.html

[33] *Code inspections* [online]. [cit. 2023-02-15]. Dostupné z: https://www.jetbrains.com/help/idea/code-inspection.html

[34] *Spring Framework: Chapter 12. Object Relational Mapping (ORM) data access* [online]. [cit. 2023-02-15]. Dostupné z: https://docs.spring.io/spring-framework/docs/2.5.5/reference/orm.html

[35] MARIA PECE, Andreea a Florina Salisteanu SALISTEANU. *Innovation and Economic Growth: An Empirical Analysis for CEE Countries* [online]. 2015. Procedia Economics and Finance, 2015, 461-467 [cit. 2023-03-21]. ISSN 2212-5671. Dostupné z: https://www.sciencedirect.com/science/article/pii/S2212567115008746

[36] *MIT License* [online]. [cit. 2023-05-10]. Dostupné z: https://choosealicense.com/licenses/mit/

[37] *The European Institute of Innovation & Technology* [online]. [cit. 2023-03-10]. Dostupné z: https://eit.europa.eu/who-we-are

[38] *UVP BIOMED: Univerzitný vedecký park pre biomedicínu Bratislava Slovenská akadémia vied* [online]. [cit. 2023-03-10]. Dostupné z: http://www.biomed.sav.sk

[39] *AV ČR: Dotační programy pro aplikovaný výzkum a transfer technologií* [online]. [cit. 2023-03-10]. Dostupné z: https://www.avcr.cz/cs/veda-a-vyzkum/transfer-technologii/dotacni-prilezitosti/

[40] *I&i Prague: Bio-Innovation Center* [online]. [cit. 2023-03-10]. Dostupné z: https://www.iniprague.com

[41] *CIVITTA: The Challenge*^{r} *Advisory* [online]. [cit. 2023-03-10]. Dostupné z: https://civitta.com

[42] *Linkedin* [online]. [cit. 2023-03-01]. Dostupné z: https://www.linkedin.com

[43] *Freepik* [online]. [cit. 2023-03-01]. Dostupné z: https://www.freepik.com/

[44] *Freepik: Terms of use* [online]. [cit. 2023-03-01]. Dostupné z: https://www.freepik-company.com/legal

[45] *Software architecture patterns: Chapter 1. Layered Architecture* [online]. [cit. 2023-05-11]. Dostupné z: https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html

[46] RAO R, Rakshith a Dr S. R. SWAMY. Review on Spring Boot and Spring Webflux for Reactive Web Development. International Research Journal of Engineering and Technology (IRJET)[online]. 2020, 7(4), 34-37 [cit. 2023-05-16]. ISSN 2395-0056. Dostupné z: https://www.researchgate.net/publication/341151097_Review_on_Spring_Boot_and_Spring_Webflux_for_Reactive_Web_Development

[47] FROST, Brad. Atomic Design by Brad Frost [online]. 2016 [cit. 2023-04-17]. ISBN 978-0998296609. Dostupné z: https://atomicdesign.bradfrost.com

[48] *Figma: About* [online]. [cit. 2023-05-11]. Dostupné z: https://www.figma.com/about/

[49] *Welcome to Apache Maven* [online]. [cit. 2023-05-15]. Dostupné z: https://maven.apache.org/

[50] *Visual Studio Code* [online]. [cit. 2023-05-15]. Dostupné z: https://code.visualstudio.com/

[51] *Postman* [online]. [cit. 2023-05-15]. Dostupné z: https://www.postman.com/

[52] *Swagger Open API: Specification* [online]. [cit. 2023-05-15]. Dostupné z: https://swagger.io/specification/

[53] *H2 Database Engine* [online]. [cit. 2023-05-15]. Dostupné z: https://www.h2database.com/html/main.html

[54] *AWS S3* [online]. [cit. 2023-05-15]. Dostupné z: https://aws.amazon.com/s3/

ZOZNAM POUŽITÝCH SKRATEK

MVP Minimum Viable Product

MVC Model View Controller

MVT Model View Template

JWT JSON Web Token

JSX JavaScript XML

UX User Experience

UI User Interface

ZOZNAM OBRÁZKOV

ZOZNAM TABULIEK

ZOZNAM PRÍLOH

P1 USB kľúč

PRÍLOHA P1 USB

Priložený USB kľúč obsahuje:

Zdrojový kód aplikácie pre frontend aj backend časť

Návod na spustenie aplikácie

Použité ilustrácie