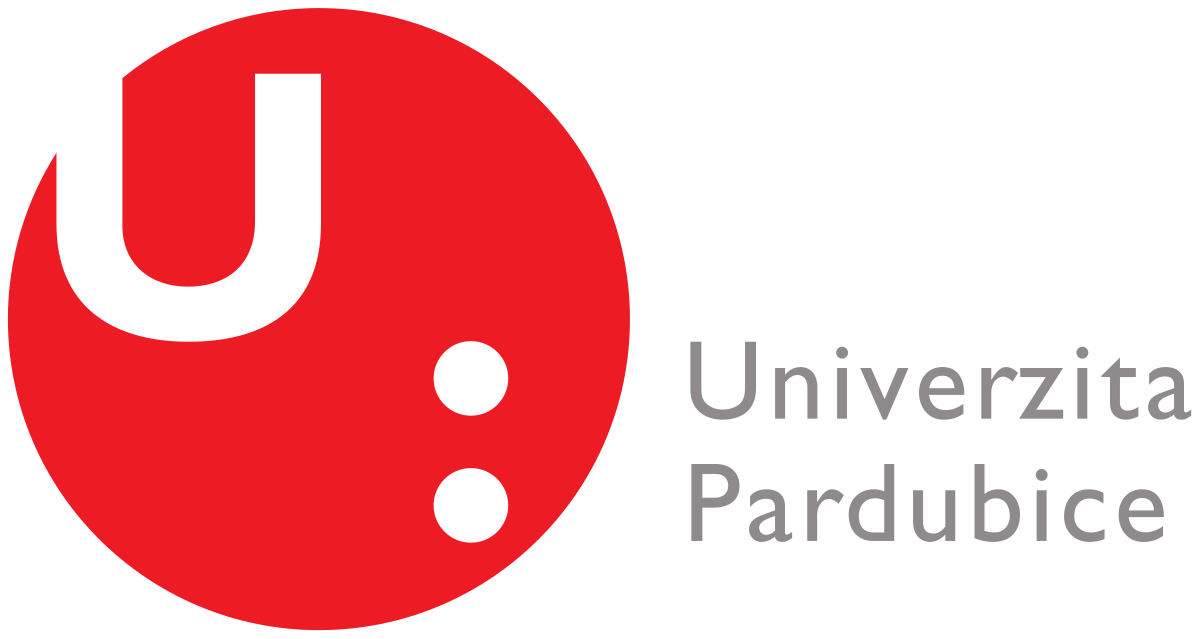
Univerzita Pardubice

**Semestrální práce**



Jan Mlejnecký

Mladá Boleslav 2024

Univerzita Pardubice  
Fakulta Elektrotechniky a Informatiky

**Semestrální práce**

**Autor: Jan Mlejnecký  
Studijní obor: Webové technologie**

Mladá Boleslav 2024

# Obsah

[Obsah 3](#_Toc167615432)

[1 Úvod 4](#_Toc167615433)

[2 Obsah práce 5](#_Toc167615434)

[2.1 Funkční požadavky 5](#_Toc167615435)

[2.2 Nefunkční požadavky 5](#_Toc167615436)

[2.3 Front End 5](#_Toc167615437)

[2.3.1 Technologie 5](#_Toc167615438)

[2.3.2 Web design a funkcionality front endu 6](#_Toc167615439)

[2.4 Back End 8](#_Toc167615440)

[2.4.1 Technologie 8](#_Toc167615441)

[2.4.2 Důležité implementace 9](#_Toc167615442)

[3 Závěr 11](#_Toc167615443)

# Úvod

Smyslem této semestrální práce bylo vytvořit webovou aplikaci, která bude provolávat Instagram API a z ní následně sbírat obdržená data, které zobrazí na stránce. Uživatel následně může sledovat vývoj sledovaného příspěvku v čase z hlediska počtu jeho komentářů a počtu líbí se, které příspěvek obdržel.

Dále bylo mým osobním cílem se tímto projektem blíže seznámit s Javou 17 a novým Springem 6 a porovnat si jeho případné výhody či nevýhody oproti Javě 8 se Springem 4, na který jsem osobně zvyklí, ale jeho životnost se čím dál více přibližuje svému konci.

# Obsah práce

## Funkční požadavky

* Uživatelé se po registraci mohou přihlásit
* Přihlášený uživatel může sledovat instagramové příspěvky
* Přihlášený uživatel může vybírat aktuálně zobrazovaný příspěvek
* Přihlášený uživatel může ručně provolat aktualizaci sledovaného příspěvku
* Aplikace zobrazuje vývoj počtu komentářů a líbí se v čase v grafu

## Nefunkční požadavky

* Uživatelské jména uživatelů musí být unikátní
* Uživatel přidává sledovaný příspěvek pomocí URL
* Aplikace je lokalizované v angličtině
* Formát datumů je v americkém formátu
* Aplikace bude v technologickém stacku Java, Spring, PostgreSQL, React

## Front End

### Technologie

Front end aplikace je vytvořen zapomocí knihovny React. Pro tento framework jsem se rozhodl z důvodu mých předešlých zkušeností, jeho stavové kontroly a možnosti vytváření dynamického reaktivních webových aplikací. Další klíčovou vlastností pro toto rozhodnutí bylo široké množství knihoven a dodatečných balíčků, které výrazně urychlují a usnadňují práci.

Rozhodl jsem se aplikaci navrhnout jako SPA, single page application, abych si dále prohloubil znalosti v tvorbě front endů v tomto principu, i z tohoto důvodu není v aplikaci využit react router a vše je dynamicky kontrolováno. Zároveň si ale uvědomuji jeho benefity a při vytváření dalšího podobného projektu bych jej využil.

Mezi využité balíčky patří například js-cookie, který napomáhá při práci s webovými cookies umožňuje jejich snadnou modifikaci, vytváření a čtení.

Dalším důležitým balíčkem webu je recharts, který tvoří jednu z hlavních komponent webu, a to právě graf, ve kterém se zobrazuje počet liků a komentářů v čase

Pro provolávání back endu slouží využitá knihovna Axios, která usnadňuje skládání zasílaných requestů, úpravu odesílaných hlaviček, v případu této aplikace hlavně z důvodu autorizace a autentifikace uživatelů. Dále Axios dovoluje například plnit i tělíčka requestů snadným a intuitivním přístupem.

### Web design a funkcionality front endu

#### Navigační pruh nepřihlášeného uživatele

#### 

Navbar nepřihlášeného uživatele obsahuje v levé části název aplikace, který zároveň slouží jako navigátor zpět na hlavní stranu, tato funkcionalita by byla převážně využita v budoucnu, kdy by byly doimplementovány funkcionality pro jednotlivé separátní stránky.

#### Navigační pruh přihlášeného uživatele



Navbar přihlášeného uživatele obsahuje v levé části název aplikace, která slouží jako proklik na stránku.

Dále v pořadí z leva je v navbaru uživatelské jméno přihlášeného uživatele s tlačítkem Logout, které slouží odhlášení momentálního uživatele přemazáním cookies a je stylizováno do modrého pozadí s bílým textem a zakulacenými rohy.

Dále v prostředku navbaru najdeme select box pro vybrání jednoho ze sledovaných příspěvků, kdy vybráním zobrazíme uložená data o příspěvku.

Dále v navbaru napravo od select boxu najdeme tlačítko Update Post stylizované stejně jako tlačítko Logout. Toto tlačítko slouží k aktualizaci momentálně zobrazeného příspěvku a aktualizaci zobrazených dat.

Další tlačítko obsahuje textaci Track New Post a slouží k zobrazení vyskakovacího okna, které následně dovoluje začít sledovat další příspěvek.

#### Hlavní obsah

Hlavní obsah aplikace je tvořen ze 7 dlaždic, každá obsahující nějaká užitečná data nebo proklik na instagramový post.

V pořadí prvního řádku a prvního sloupce najdeme nejaktuálnější počet komentářů na příspěvku, další dlaždice ve vedlejším sloupci zobrazuje nejaktuálnější počet líbí se, který příspěvek obdržel. V následující dlaždici najdeme počet aktualizací neboli počet kolikrát byl Instagram provolán, aby se do aplikace nahrály data.

Následující dlaždice v již druhém řádku obsahuje datum v americkém formátu, kdy byla vytvořena poslední aktualizace. Ve vedlejší dlaždici najdeme uživatelské jméno instagramového účtu, který nahrál sledovaný příspěvek. Dále máme dlaždici o velkosti 2x1, která obsahuje vložený instagramový element pro přímé prokliknutí do instagramu možnost zobrazení profilu nebo například rovnou po prokliknuti přidat líbí se ze svého účtu či přidat komentář. Poslední dlaždici v rozměru 1x2 tvoří graf, kde může uživatel sledovat počet líbí se a komentářů v čase.

#### Modální okno

Obsah obrázku snímek obrazovky, text

Popis byl vytvořen automaticky

F

Vyskakovací okno pro přidání sledování nového příspěvku obsahuje input a dvě tlačítka, input slouží pro zadání URL adresy na příspěvek na instagramu. Tlačítko Cancel zavře okno a neodešle požadavek pro sledování. Naopak tlačítko Track zašle požadavek na sledování a obdržení dat Instagramového příspěvku a následně zavře modální okno.

## Back End

### Technologie

Technologie využité na back endu, serveru, aplikace jsou Java 17, Spring 6 a Hibernate, tedy standardní a aktuální technologický stack pro vývoj v back endů javě. Rozhodl jsem se pro Javu 17 převážně z důvodu již zastaralosti Javy 8 hlavně ze strany bezpečnosti a to i na úkor některých dnes již historických knihoven, které ale velmi dobře plnily svoji funkci.

Spring 6 je momentálně nejnovější verzí Java Spring a s tím se objevují i problémy z hlediska chybějící zpětné kompatibility pro mnoho funkcionalit ze Spring 4, který byl standardem k Javě 8, ale i chybějcí zpětné kompatibility mezi verzí 6.2.2 a 6.1.x, které tvořili problém z hlediska nedostatku informací na internetu.

Hibernate jsem zvolil z důvodu jeho elegantnosti z hlediska kontroly entit a jeho podporu HQL, který preferuji oproti Criteria queries, které jsou velmi populární u dalších vývojářů. Hibernate slouží primárně ke komunikaci a provolávání databázové vrstvy a dovoluje nám snadno mapovat tabulky na jednotlivé Java classy/objekty.

Aplikace dále využívá knihovny jako Lombok, pro zvýšení čitelnosti a zaručení předvídatelnosti pro jednotlivé classy (DTO, Entity a podobně), tato knihovna také výrazně zrychluje práci při vytváření velkého objemu DTO či entit.

Pro snadnější udržování database také aplikace obsahuje knihovnu Flyway, která se stará o řízení migračních skriptů database, zachování jejich správné posloupnosti a umožňuje snadnější spolupráci vice vývojářů.

### Důležité implementace

Mezi důležité implementaci patří například způsob nakonfigurování bezpečnosti aplikace, tedy autentizace a následná autorizace příchozích requestů.

*public SecurityFilterChain* securityFilterChain(HttpSecurity http) *throws* Exception {  
 http.cors();  
  
 http.csrf().disable()  
 .authorizeRequests()  
 .requestMatchers("/users/\*\*").permitAll()  
 .anyRequest().authenticated()  
 .and().sessionManagement().sessionCreationPolicy(SessionCreationPolicy.***STATELESS***)  
 .and().addFilterBefore(jwtAuthorizationFilter, UsernamePasswordAuthenticationFilter.*class*);  
  
 *return* http.build();  
}

Na úryvku kódu můžeme sledovat způsob filtrace, jakýsi firewall, příchozích requestů na back end aplikace. Můžeme pozorovat že pro cestu /users/\*\* filtrace neprobíhá a to z důvodu možnosti neověřených uživatelů se zaregistrovat a přihlásit a tak obdržet token, který následně využívají pro přístup do ostatních koncových bodů.

Dále můžeme pozorovat i povolení CORSu, ten je nastavený následovně.

*CorsConfigurationSource* corsConfigurationSource() {  
 CorsConfiguration configuration = *new* CorsConfiguration();  
 configuration.setAllowedOrigins(Arrays.*asList*("http://127.0.0.1:5173", "http://localhost:5173"));  
 configuration.setAllowedMethods(Arrays.*asList*("GET","POST"));  
 configuration.setExposedHeaders(Arrays.*asList*("Authorization", "content-type"));  
 configuration.setAllowedHeaders(Arrays.*asList*("Authorization", "content-type"));  
 UrlBasedCorsConfigurationSource source = *new* UrlBasedCorsConfigurationSource();  
 source.registerCorsConfiguration("\*\*/\*\*", configuration);  
 source.registerCorsConfiguration("/\*\*", configuration);  
 *return* source;  
}

Na zmíněné konfiguraci momentálně můžeme sledovat, že pro účely tohoto projektu je komunikace omezená na lokální smyčku, tedy localhost popřípadě 127.0.0.1 a defaultní port ViteJs. Omezili jsme možnost zasílání requestů pouze na GET a POST také z důvodu bezpečnosti a zobtížnění dolování veškerých koncových bodů. Cors je dále nastaven na kompletně všechny cesty, jak je viděno na posledních dvou řádcích před return.

# Závěr

Závěrem nejsem příliš spokojený s prací, a to z několika důvodů a chybných kroků, které jsem učinil z důvodu bláhovosti, a i nezkušenosti s určitými technologiemi.

Jako první důvod nespokojenosti bych uvedl volbu verze Springu pro back end aplikace, kdy jsem využil Spring Security 6.2.2, která se opět velmi změnila od 6.1.x a to i v místech, které byly zcela nově přepsány v minulé verzi. Vedlo to k problémům neexistujících diskusí na internetu a velmi neobjemné oficiální dokumentaci. Spoustu implementací je tedy zcela vymyšlených mnou a nemohu zaručit správnost a jejich bezpečnost.

Dalším důvodem je výpadek na straně Mety, kdy nelze vytvářet oficiální testovací uživatele pro Instagram API, byl jsem tedy nucen Instagram spíše scrapovat než využívat oficiální podporované cesty.

Posledně bych negativně zhodnotil i přístup, který jsem vybral pro vývoj front endu v Reactu. Zvolil jsem čistě SPA přístup, který se mi zdá limitující asi i z důvodu mých nedostatečných předešlých zkušeností. Vedlo to k tomu, že aplikace má poměrně “špagetový” kód a jsem si jistý že z dlouhodobého hlediska je neudržitelný. Momentálně bych asi aplikaci přepsal v Angularu nebo jiném frameworku, který není tak svobodný jako React a nutí vývojáře udržovat určité konvence.

Z celkové retrospektivy mi ale projekt přinesl další zkušenosti hlavně z oblasti FE vývoje, které u mě zaostávají hlavně z důvodu zaměstnání jako BE vývojář.

Pro další projekt bych si definoval jiné zadání a snažil bych se najít zajímavost v jiné oblasti než sociálních sítí.