NNPIA – Zadání semestrální práce 2026

Cílem semestrální práce je vytvořit webovovou aplikaci, která demonstruje znalosti studenta získané v předmětu NNPIA - Programování internetových aplikací. Zaměření aplikace si student volí sám.

Organizační požadavky

- **Důležité termíny:**
  - 📆 **8. 3. 2026** – Konzultovat téma semestrální práce a odevzdat README.md soubor s popisem tématu do repozitáře.
    - Přes Microsoft Teams nebo osobně.
  - 📆 **12. 4. 2026** – Navrhnout databázový model a vložit ho s popisem entit do README.md.
- **Termín odevzdání:** minimálně **48 hodin před termínem zápočtu**, který si student zvolí.
- **Odevzdání repozitáře:** GitHub Classroom.

Povinné požadavky

Pro uznání semestrální práce a získání zápočtu je nutné splnit následující požadavky:

Backend (Spring Boot)

1. Použití **Spring Boot** jako backendového frameworku.
2. **Minimálně 3 datové entity** včetně vizualizace datových entit. Schéma může být vygenerováno pomocí IDE na základě již existujícího kódu.
3. **Spring JPA** pro práci s databází.
4. **Spring Security** pro autentizaci a autorizaci pomocí JWT tokenů.
5. **Vícevrstvá architektura** (Model, DAO/Repository, Service, Controller).

Frontend

1. Použití **React.js** nebo jiné knihovny pro tvorbu single page aplikace.
   - Mobilní aplikace boudou akceptovány pokud bude využit React Native pro implementaci.
2. Aplikace musí obsahovat **minimálně 7 komponent** a ideálně hooky.
3. Implementace alespoň jedné **znovupoužitelné komponenty** (např. data-grid, tabulka, filtr, modal apod.).
   - Znovupoužitelnou komponentou se rozumí komponenta, která je navržena a použita alespoň na 2 místech v aplikaci s různými parametry.
4. Aplikace by měla po designové stránce alespoň trochu odrážet moderní trendy.
   - Chybějící nebo minimální stylování nemusí být uznáno.
   - Tento bod můžete snadno dodržet aplikací UI knihoven jako je `React MUI`, `Hero UI` nebo `Tailwind`.

Dodatečná netriviální funkcionalita

Student si zvolí jednu dodatečnou netriviální funkcionalitu dle svého uvážení. Funkcionalita může být vlastní, ale musí být konzultována předem. Možnosti zahrnují:

- Automatická tvorba Open API 3 dokumentace.
- Cashování REST API odpovědí za pomocí Redis.
- Napojení na OpenAI API nebo jinou službu poskytující jazykové modely.
- Přihlašování pomocí SSO (Google, Microsoft...).
- Backend v jazyce Kotlin.
- Nasazení aplikace na produkční prostředí.
- Použití jiné knihovny než React.js pro tvorbu frontendu (musí být zachována single page aplikace).

Obecné požadavky

1. Dodržování **principů čistého kódu** (čitelnost, struktura, názvy, formátování).
    - U backendu a frontendu musí být struktura projektu členěna na balíčky/podsložky.
    - Kód v jedné komponentě/souboru nebude uznán.
2. Semestrální práce musí splňovat single page application rendering pattern.
3. Studenti si sami vybírají vhodný typ databáze (relační, grafová, dokumentová). Výběr by měl odpovídat potřebám aplikace.

Testování

1. **Minimálně 2 unit testy** na netriviální logiku/algoritmus.
2. **Minimálně 2 unit testy** controlleru (např. s využitím **MockMvc**).
3. **Minimálně 1 integrační test**, kde spolupracují alespoň 2 třídy servisní vrstvy (datová vrstva není mockovaná).

Doporučený postup implementace

1. Promyslete si aplikační logiku.
2. Navrhněte datový model.
3. Maximálně využívejte AI služby, copilota a chatboty:

- Obzvláště pro repetativní úkoly, generování testů a psaní dokumentace.
- Ideálně si pro každý problém vytvořte nový chat, zadejte chatbotovi vhodnou roli a poskytněte mu dostatek kontextu.
- Nezapomínejte ale validovat výstupy a snažte se porozumět dané problematice.

4. Ideálně pracujte průběžně. Můžete použít kód ze cvičení jako základ.
5. Při řešení problémů aplikujte dekompozici.