Platforma pro správu zaměstnanců

Dokumentace

NNPRO 2024/2025

Obsah

1 Použité technologie backend

Aplikace je napsána v jazyce Java verze 23 a využívá framework Spring Boot. Kombinace těchto technologií zajišťuje efektivní, škálovatelný a moderní vývoj.

1.1 Spring boot

Spring Boot je použit jako hlavní framework pro vývoj a správu aplikace. Umožňuje automatickou konfiguraci a efektivní tvorbu RESTful API. Verze frameworku je 3.3.4.

1.2 Spring security

Spring Security je použit pro autentizaci a autorizaci uživatelů s definovanými rolími (ROLE_ADMIN, ROLE_LEADER, ROLE_EMPLOYEE) a zabezpečení přístupu k endpointům. Verze knihovny je 6.1.4.

1.3 JWT

JWT je využito pro autentizaci uživatelů pomocí přístupových a resetovacích tokenů. Používá se knihovna java-jwt ve verzi 4.4.0.

1.4 Spring Data JPA a Hibernate

Spring Data JPA a Hibernate jsou využity pro práci s databízí, včetně CRUD operací a mapování objektů Java na databázové tabulky.

1.5 Postgre SQL

PostgreSQL je používán jako databázový server pro ukládání dat. Verze použítého ovladače je 42.7.4.

1.6 Flyway

Flyway zajišťuje migrace a správu verzí databáze. Používá se verze 10.20.1.

2 Použité technologie frontend

2.1 Frameworky a knihovny

- **React**: Hlavní knihovna pro vytváření uživatelského rozhraní (verze 18.3.1).
- **React Router DOM**: Pro správu směrování v aplikaci (verze 6.27.0).
- **Formik**: Pro správu formulářů a jejich validaci (verze 2.4.6).
- **Yup**: Pro validaci formulářových dat (verze 1.4.0).

2.2 UI a stylování

- **MUI (Material-UI)**: Pro komponenty uživatelského rozhraní (verze 6.1.4) a ikony (**@mui/icons-material**, verze 6.1.4).
- **@emotion/react a @emotion/styled**: Pro stylování komponent (verze 11.13.3 a 11.13.0).
- **Styled-components**: Alternativní knihovna pro práci se styly (verze 6.1.13).

2.3 Kalendář a vizualizace

- **FullCalendar**: Pro implementaci kalendáře s podporou událostí (verze 6.1.15 pro moduly **core**, **daygrid**, **interaction**, a **react**).
- **MUI X Charts**: Pro vytváření grafů a vizualizaci dat (verze 7.23.0).

2.4 Ostatní nástroje

- **Axios**: Pro komunikaci s API (verze 1.7.7).
- **React Toastify**: Pro zobrazení notifikací (verze 10.0.6).
- **React Helmet Async**: Pro správu hlaviček HTML dokumentu (verze 2.0.5).
- **HTTP Status Codes**: Pro práci se stavovými kódy HTTP (verze 2.3.0).

2.5 Vývojové nástroje

- **Vite**: Pro sestavení a spuštění aplikace (verze 5.4.8).
- **TypeScript**: Pro statickou typovou kontrolu (verze 5.5.3).
- **ESLint**: Pro kontrolu kvality kódu a dodržování standardů (verze 9.11.1).

- **@vitejs/plugin-react**: Plugin pro Vite na podporu Reactu (verze 4.3.2).

3 Databázový model

Databázový model aplikace je navržen tak, aby podporoval hierarchickou správu uživatelů, projektů, úkolů a souvisejících operací, jako jsou docházka, komentáře nebo výkazy výkonu.

3.1 Databázové entity

3.1.1 Tabulka app_user (Uživatelé)

Tabulka obsahuje informace o uživatelích aplikace. Každý uživatel má přiřazenou roli, která určuje jeho oprávnění.

- **Primární klíč:** id
- **Cizí klíč:** role_id odkazuje na tabulku role.
- **Popis hlavních sloupců:**
    - first_name, surname, username, email: Identifikační údaje uživatele.
    - password: Hash hesla.
    - phone_number: Telefonní kontakt.

3.1.2 Tabulka role (Role)

Tabulka obsahuje seznam rolí, které definují oprávnění uživatelů.

- **Primární klíč:** id
- **Popis hlavních sloupců:**
    - name: Název role (např. ROLE_ADMIN, ROLE_LEADER, ROLE_EMPLOYEE).

3.1.3 Tabulka project (Projekty)

Tabulka obsahuje seznam úkolů přiřazených k projektům.

- **Primární klíč:** id
- **Cizí klíč:** project_id odkazuje na tabulku project.
- **Popis hlavních sloupců:**
    - title: Název úkolu.

- description: Popis úkolu.
- creation_date, due_date, finish_date: Časová metadata.
- difficulty: Obtížnost úkolu (např. Easy, Medium, Hard).
- status: Stav úkolu.

3.1.4 Tabulka attendance (Docházka)

Tabulka slouží k zaznamenávání docházky uživatelů na konkrétních úkolech.

- **Primární klíč:** id
- **Cizí klíče:**
  - task_id odkazuje na tabulku task.
  - user_id odkazuje na tabulku app_user.
- **Popis hlavních sloupců:**
  - start_time, end_time: Časový interval docházky.
  - work_description: Popis odvedené práce.

3.1.5 Tabulka comment (Komentáře)

Tabulka slouží k ukládání komentářů uživatelů k úkolům.

- **Primární klíč:** id
- **Cizí klíče:**
  - task_id odkazuje na tabulku task.
  - user_id odkazuje na tabulku app_user.
- **Popis hlavních sloupců:**
  - text: Obsah komentáře.
  - created_date: Datum vytvoření komentáře.

3.1.6 Tabulka performance_report (Hodnocení výkonu)

Spojovací tabulka mezi uživateli a projekty, která umožňuje přiřazení více uživatelů k jednomu projektu.

- **Primární klíče:** project_id, user_id.
- **Cizí klíče:**
    - project_id odkazuje na tabulku project.
    - user_id odkazuje na tabulku app_user.

3.1.7 Tabulka user_task (Vztah uživatel-úkol)

Spojovací tabulka mezi uživateli a úkoly, která umožňuje přiřazení více uživatelů k jednomu úkolu.

- **Primární klíče:** task_id, user_id.
- **Cizí klíče:**
    - task_id odkazuje na tabulku task.
    - user_id odkazuje na tabulku app_user.

3.1.8 Tabulka user_project (Vztah uživatel-projekt)

Tato tabulka představuje vztah mezi uživateli a projekty. Umožňuje přiřazení více uživatelů k jednomu projektu a současně jednoho uživatele k více projektům (M: N vztah).

- **Primární klíče:**
    - Kombinace project_id a user_id.
- **Cizí klíče:**
    - project_id odkazuje na tabulku project.
    - user_id odkazuje na tabulku app_user.

3.2 Relace mezi entitami

3.2.1 Tabulka app_user

app_user.role_id -> role.id

- **Typ:** M:1 (Many-to-One)
- Každý uživatel (app_user) má přiřazenou jednu roli (role).
- **Význam:** Rola určuje úroveň oprávnění uživatele v systému.

app_user.id -> user_project.user_id

- **Typ:** 1: N (One-to-Many)
- Jeden uživatel může být členem více projektů (relace v tabulce user_project).

app_user.id -> user_task.user_id

- **Typ:** 1: N (One-to-Many)
- Jeden uživatel může být přiřazen k více úkolům (relace v tabulce user_task).

app_user.id -> attendance.user_id

- **Typ:** 1: N (One-to-Many)
- Jeden uživatel může mít více záznamů docházky v tabulce attendance.

app_user.id -> comment.user_id

- **Typ:** 1: N (One-to-Many)
- Jeden uživatel může vytvořit více komentářů.

app_user.id -> performance_report.user_id

- **Typ:** 1: N (One-to-Many)
- Jeden uživatel může mít více výkonnostních reportů.

app_user.id -> project.user_id

- **Typ:** 1: N (One-to-Many)
- Jeden uživatel může vytvořit více projektů.

3.2.2 Tabulka role

role.id -> app_user.role_id

- **Typ:** 1: N (One-to-Many)

- Jedna role může být přiřazena více uživatelům.

3.2.3 Tabulka project

project.id -> task.project_id

- **Typ:** 1: N (One-to-Many)
- Jeden projekt může mít více úkolů.

project.id -> user_project.project_id

- **Typ:** 1: N (One-to-Many)
- Jeden projekt může mít více členů v tabulce user_project.

project.user_id -> app_user.id

- **Typ:** M:1 (Many-to-One)
- Každý projekt má jednoho vlastníka (např. vedoucího).

3.2.4 Tabulka task

task.project_id -> project.id

- **Typ:** M:1 (Many-to-One)
- Každý úkol patří jednomu projektu.

task.id -> user_task.task_id

- **Typ:** 1: N (One-to-Many)
- Jeden úkol může být přiřazen více uživatelům v tabulce user_task.

task.id -> attendance.task_id

- **Typ:** 1: N (One-to-Many)
- Jeden úkol může mít více záznamů docházky.

task.id -> comment.task_id

- **Typ:** 1: N (One-to-Many)
- Jeden úkol může mít více komentářů.

task.id -> performance_report.task_id

- **Typ:** 1:N (One-to-Many)
- Jeden úkol může mít více výkonnostních reportů.

3.2.5 Tabulka user_project

user_project.user_id -> app_user.id

- **Typ:** M:1 (Many-to-One)
- Každý záznam odkazuje na jednoho uživatele.

user_project.project_id -> project.id

- **Typ:** M:1 (Many-to-One)
- Každý záznam odkazuje na jeden projekt.

3.2.6 Tabulka user_task

user_task.user_id -> app_user.id

- **Typ:** M:1 (Many-to-One)
- Každý záznam odkazuje na jednoho uživatele.

user_task.task_id -> task.id

- **Typ:** M:1 (Many-to-One)
- Každý záznam odkazuje na jeden úkol.

3.2.7 Tabulka attendance

attendance.user_id -> app_user.id

- **Typ:** M:1 (Many-to-One)
- Každý záznam docházky odkazuje na jednoho uživatele.

attendance.task_id -> task.id

- **Typ:** M:1 (Many-to-One)
- Každý záznam docházky odkazuje na jeden úkol.

3.2.8 Tabulka comment

comment.user_id -> app_user.id

- **Typ:** M:1 (Many-to-One)
- Každý komentář patří jednomu uživateli.

comment.task_id -> task.id

- **Typ:** M:1 (Many-to-One)
- Každý komentář je přiřazen k jednomu úkolu.

3.2.9 Tabulka performance_report

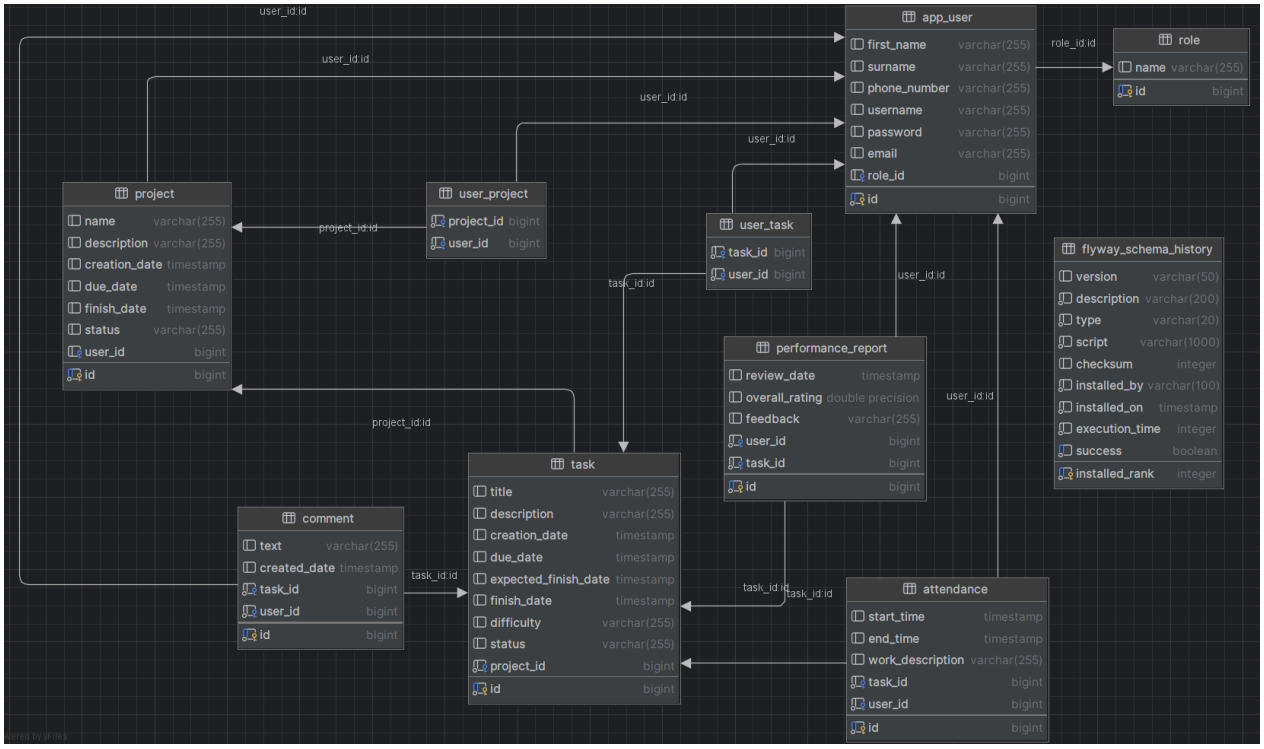
performance_report.user_id -> app_user.id

- **Typ:** M:1 (Many-to-One)
- Každý výkonnostní report patří jednomu uživateli.

performance_report.task_id -> task.id

- **Typ:** M:1 (Many-to-One)
- Každý výkonnostní report je přiřazen k jednomu úkolu.

3.3 ERD diagram



4 Role a oprávnění přístupu

4.1 Autentizace uživatelů

Aplikace používá JWT (JSON Web Token) pro autentizaci a autorizaci uživatelů.

- Expirace JWT tokenu je defaultně nastavena na 1 den
- Při každém požadavku na zabezpečený endpoint musí být JWT token přiložen v hlavičce Authorization jako Bearer <token>.
- Pro resetování hesla je použít JWT token s dobou expirace 30 minut

Hesla jsou před uložením do databáze šifrována pomocí BCrypt.

4.2 Správa rolí

Aplikace disponuje hierarchickou strukturou rolí, která zajišťuje, že různí uživatelé mají přístup k různým funkcionalitám. Tento přístup je spravován pomocí Spring Security, která omezuje přístup k jednotlivým endpointům REST controlleru na základě rolí uživatele. Existují tři hlavní role, které ovládají přístup:

- **ROLE_ADMIN** (Administrátor): Tato role má nejvyšší úroveň přístupu. Administrátoři mohou provádět všechny operace na uživatelských datech, včetně jejich vytváření, aktualizování, mazání a přiřazování úkolů a projektů.
- **ROLE_LEADER** (Vedoucí): Role vedoucího projektu umožňuje vedoucím přiřazovat uživatele k projektům a úkolům, ale bez schopnosti mazat nebo vytvářet uživatele. Vedoucí mohou provádět operace týkající se úkolů a projektů, na kterých se podílejí.
- **ROLE_EMPLOYEE** (Zaměstnanec): Role zaměstnanců umožňuje uživatelům přístup pouze k informacím a operacím, které souvisejí s jejich přiřazenými úkoly. Zaměstnanci mohou například vidět svůj seznam úkolů nebo projektů, ale nemohou provádět žádné změny v rámci správy uživatelů, nemohou vytvářet projekty apod.

4.3 Použití AuthService pro validaci práv

Kromě jednoduchého přiřazení rolí je v aplikaci implementována i validace práv prostřednictvím AuthService. Tato servisní třída slouží kromě samotné autentizace uživatelů také k ověřování, zda konkrétní uživatel má práva vykonat akci, která souvisí s konkrétním projektem nebo úkolem.

Pro každou operaci je prováděn kontrolní mechanismus, který ověřuje ID projektu, úkolu apod. a pomocí tohoto údaje zjistí, zda přihlášený uživatel má přístup k dané operaci.

Tato kombinace rolí a autentizace zajišťuje, že uživatelé mají přístup pouze k těm operacím, na které mají právo podle své role a konkrétního kontextu (projektu nebo úkolu).

Přiřazení úkolu uživateli (assignTaskToUser):

1. **ROLE_ADMIN:**
    - Uživatel má plné oprávnění a operace proběhne bez dalších kontrol.
2. **ROLE_LEADER:**
    - Před přiřazením úkolu je zkontrolováno:
        - Je přihlášený uživatel vedoucím projektu, ke kterému úkol patří?
        - Pokud ano, úkol může být přiřazen členovi projektu.

Přístup k úkolům v projektu:

- **ROLE_EMPLOYEE:**
    - Zaměstnanec může zobrazit pouze úkoly, které mu byly přiřazeny.
- **ROLE_LEADER:**
    - Vedoucí projektu má přístup ke všem úkolům v rámci svého projektu.

5 REST API Dokumentace

Pro dokumentaci API bylo využito Swaggeru.

Dokumentace a přehled všech dostupných endpointů, kterými aplikace disponuje je možné zobrazit na adrese https://editor-next.swagger.io, zde stačí soubor „api-docs.json", který se nachází ve stejném adresáři jako tato dokumentace, jednoduše importovat, po importu souboru se vám v editoru zobrazí kompletní přehled všech dostupných endpointů aplikace, včetně popisů a parametrů, které podporují.

6 Flyway

V aplikaci používáme Flyway migrace pro správu databázových schémat. První migrace je určena pro vytvoření databázových tabulek. Další dvě migrace slouží pro vložení testovacích dat do těchto tabulek.

Ačkoliv víme, že vkládání dat přímo v migracích není ideální praxe, rozhodli jsme se tuto strukturu použít z důvodu testování frontendových částí aplikace. Frontend totiž potřebuje pracovat s reálnými daty, aby správně fungoval a ukázal realistické chování aplikace. Tento přístup nám umožňuje simulovat realistické prostředí pro testování, aniž bychom museli přistupovat k externím nástrojům nebo složitějším procesům pro generování dat. V budoucnu plánujeme tento způsob vylepšit, ale pro aktuální potřeby je to nejefektivnější řešení.

7 Docker

V projektu je využit Docker, kde máme vytvořeny jednotlivé kontejnery pro správu prostředí pro frontend, databázi, což výrazně zjednodušuje správu závislostí, vytváří konzistentní vývojové prostředí a zajišťuje, že všechny komponenty aplikace běží v oddělených, izolovaných prostředích. Při případném nasazení budue dockerizován i backend.

8 Testování

Testování aplikace je implementováno s cílem zajistit správnou funkčnost, bezpečnost a spolehlivost systému. Pro unit testy je využito JUnit, které pokrývá testování logiky aplikace na úrovni jednotlivých komponent, jako jsou služby a pomocné třídy. Pro testování Spring Security a autentifikace uživatelů je využito spring-security-test, což umožňuje simulovat různé scénáře autentizace a autorizace, například ověřování rolí a práv uživatelů při přístupu k REST API.

Dále je využito frameworku Mockito pro mockování závislostí a simulaci chování komponent, jako jsou služby pro správu uživatelů, úkolů a projektů, což umožňuje testování izolovaných funkcí bez potřeby skutečné interakce s databází nebo externími systémy. Pro integrační testy aplikace je využito Spring Boot testování, které ověřuje komunikaci mezi vrstvami aplikace, včetně propojení s databází přes JPA. Integrační testy také pokrývají REST API endpointy, kde je validována správnost HTTP odpovědí v závislosti na požadavcích a autentifikaci uživatele.

Pro integrační testy je využito spring-boot-starter-test, který zahrnuje nejen JUnit, ale i další jako MockMvc pro simulaci HTTP požadavků. Tyto testy pomáhají ověřit, že všechny endpointy fungují správně, a to jak pro administrátory, vedoucí, tak i pro běžné zaměstnance.

Pro testování byla zvolena H2 databáze, i přestože v produkčním prostředí je nasazena PostgreSQL. Vzhledem k jednoduchosti databázového schématu nebyly očekávány žádné komplikace v tomto směru.

8.1 Testovací scénáře

8.1.1 Test zobrazení seznamu uživatelů (testFindAppUsers)

Scénář:

- Testuje správné načítání seznamu uživatelů s použitím stránkování a třídění.
- **Předpoklady:** Existují uživatelé v databázi.
- **Očekávaný výsledek:** Server odpoví statusem 200 OK a vrátí seznam uživatelů podle specifikovaného stránkování a třídění.

8.1.2 Test vytvoření nového uživatele (testCreateAppUser)

Scénář:

- Testuje vytvoření nového uživatele s platnými údaji.
- **Předpoklady:** V databázi je již definována role "ROLE_EMPLOYEE".
- **Očekávaný výsledek:** Server odpoví statusem 201 Created, uživatel je uložen v databázi a jeho údaje jsou správně uloženy.

8.1.3 Test aktualizace uživatele (testUpdateAppUser)

Scénář:

- Testuje aktualizaci existujícího uživatele.
- **Předpoklady:** Uživatel existuje v databázi.
- **Očekávaný výsledek:** Server odpoví statusem 200 OK, údaje uživatele jsou aktualizovány v databázi (jméno, email, telefonní číslo).

8.1.4 Test smazání uživatele (testDeleteAppUser)

Scénář:

- Testuje smazání uživatele z databáze.
- **Předpoklady:** Uživatel existuje v databázi.
- **Očekávaný výsledek:** Server odpoví statusem 204 No Content, uživatel je odstraněn z databáze.

8.1.5 Test přiřazení úkolu uživateli (testAddTaskToUser)

Scénář:

- Testuje přiřazení úkolu k uživateli.
- **Předpoklady:** Projekt a úkol existují, uživatel existuje a má oprávnění k přiřazování úkolů.
- **Očekávaný výsledek:** Server odpoví statusem 201 Created, úkol je přiřazen k uživatelskému účtu a je viditelný v seznamu úkolů uživatele.

8.1.6 Test přiřazení uživatele k projektu (testAddUserToProject)

Scénář:

- Testuje přiřazení uživatele k projektu.

- **Předpoklady:** Projekt existuje a uživatel není ještě přiřazen k žádnému projektu.
- **Očekávaný výsledek:** Server odpoví statusem 201 Created, uživatel je přiřazen k projektu a tento projekt je zobrazen ve seznamu projektů uživatele.

8.1.7 Test přiřazení uživatele k projektu, když už je členem (testAddUserToProjectWhenUserIsAlreadyMebmer)

Scénář:

- Testuje situaci, kdy se uživatel pokusí přidat k projektu, ke kterému už patří.
- **Předpoklady:** Uživatel je již přiřazen k projektu.
- **Očekávaný výsledek:** Server odpoví statusem 409 Conflict, operace není provedena.

8.1.8 Test odstranění uživatele z projektu (testRemoveUserFromProject)

Scénář:

- Testuje odstranění uživatele z projektu.
- **Předpoklady:** Uživatel je již přiřazen k projektu.
- **Očekávaný výsledek:** Server odpoví statusem 200 OK, uživatel je odstraněn z projektu a projekt již není uveden v seznamu jeho projektů.

8.1.9 Test validace při vytváření uživatele (testCreateAppUserValidationError)

Scénář:

- Testuje validaci, kdy jsou odeslány neplatné údaje (např. prázdné heslo).
- **Předpoklady:** Zadané údaje nejsou validní (např. prázdné heslo).
- **Očekávaný výsledek:** Server odpoví statusem 400 Bad Request, dojde k chybové odpovědi kvůli neplatným údajům.

8.1.10 Test zobrazení všech docházkových záznamů (testFindAllAttendances)

Scénář:

- Testuje zobrazení všech docházkových záznamů.
- **Předpoklady:** Systém obsahuje alespoň jeden docházkový záznam.

- **Očekávaný výsledek:** Server odpoví statusem 200 OK a vrátí seznam všech docházkových záznamů.

8.1.11 Test vytvoření docházky (testCreateAttendance)

Scénář:

- Testuje vytvoření nového docházkového záznamu.
- **Předpoklady:** Uživatel má přiřazený úkol a je přihlášen.
- **Očekávaný výsledek:** Server odpoví statusem 201 Created a nově vytvořený docházkový záznam je uložen do databáze.

8.1.12 Test úpravy docházky (testUpdateAttendance)

Scénář:

- Testuje úpravu existujícího docházkového záznamu.
- **Předpoklady:** Existuje již docházkový záznam, který patří uživateli.
- **Očekávaný výsledek:** Server odpoví statusem 200 OK a docházkový záznam je aktualizován v databázi.

8.1.13 Test odstranění docházky (testDeleteAttendance)

Scénář:

- Testuje odstranění docházkového záznamu.
- **Předpoklady:** Existuje docházkový záznam, který patří uživateli.
- **Očekávaný výsledek:** Server odpoví statusem 204 No Content a docházkový záznam je odstraněn z databáze.

8.1.14 Test zobrazení docházky podle ID úkolu (testFindAttendancesByTaskId)

Scénář:

- Testuje zobrazení docházkových záznamů pro konkrétní úkol.
- **Předpoklady:** Systém obsahuje alespoň jeden úkol a několik docházkových záznamů, některé přiřazené k danému úkolu.

- **Očekávaný výsledek:** Server odpoví statusem 200 OK a vrátí seznam docházkových záznamů, které odpovídají zadanému úkolu.

8.1.15 Test úspěšného přihlášení (testLoginSuccess)

Scénář:

- Testuje úspěšné přihlášení uživatele.
- **Předpoklady:** Uživatel s uživatelským jménem "username" a heslem "Passw0rd*" existuje v systému.
- **Očekávaný výsledek:** Server odpoví statusem 200 OK a vrátí JSON s informacemi o uživatelském jménu, roli a e-mailu uživatele.

8.1.16 Test neúspěšného přihlášení (testLoginFailure)

Scénář:

- Testuje neúspěšné přihlášení uživatele s nesprávným heslem.
- **Předpoklady:** Uživatel s uživatelským jménem "username" existuje v systému, ale heslo je zadáno chybně.
- **Očekávaný výsledek:** Server odpoví statusem 401 Unauthorized.

8.1.17 Test úspěšné registrace (testRegisterSuccess)

Scénář:

- Testuje úspěšnou registraci nového uživatele.
- **Předpoklady:** Uživatel s požadovanými údaji není v systému ještě zaregistrován.
- **Očekávaný výsledek:** Server odpoví statusem 200 OK a vrátí JSON s informacemi o nově zaregistrovaném uživateli, včetně jména, e-mailu, telefonního čísla a role.

8.1.18 Test registrace selhala, kvůli již existujícímu uživatelskému jménu (testRegisterFailureUsernameAlreadyExists)

Scénář:

- Testuje selhání registrace kvůli již existujícímu uživatelskému jménu.

- **Předpoklady:** Uživatel s uživatelským jménem "username" již existuje v systému.
- **Očekávaný výsledek:** Server odpoví statusem 409 Conflict a vrátí zprávu "Username already exists".

8.1.19 Test registrace selhala kvůli neplatnému e-mailu (testRegisterFailureEmailNotValid)

Scénář:

- Testuje selhání registrace kvůli neplatnému formátu e-mailu.
- **Předpoklady:** E-mail ve formuláři registrace není platný (např. "justEmail").
- **Očekávaný výsledek:** Server odpoví statusem 400 Bad Request a vrátí zprávu "email: Email should be valid".

8.1.20 Test získání všech komentářů (testGetAllComments)

Scénář:

- Testuje získání všech komentářů z databáze.
- **Předpoklady:** V systému existuje alespoň jeden komentář.
- **Očekávaný výsledek:** Server odpoví statusem 200 OK a vrátí seznam komentářů, přičemž první komentář má text "Initial comment".

8.1.21 Test získání komentáře podle ID (testGetCommentById)

Scénář:

- Testuje získání komentáře podle jeho ID.
- **Předpoklady:** Komentář s daným ID existuje v systému.
- **Očekávaný výsledek:** Server odpoví statusem 200 OK a vrátí komentář s textem "Initial comment".

8.1.22 Test získání komentářů podle ID úkolu (testGetCommentsByTaskId)

Scénář:

- Testuje získání všech komentářů, které jsou přiřazeny k úkolu.

- **Předpoklady:** Úkol s daným ID existuje v systému a má přiřazený alespoň jeden komentář.
- **Očekávaný výsledek:** Server odpoví statusem 200 OK a vrátí seznam komentářů pro tento úkol, přičemž první komentář má text "Initial comment".

8.1.23 Test získání komentářů podle ID uživatele (testGetCommentsByUserId)

Scénář:

- Testuje získání všech komentářů, které patří uživateli podle jeho ID.
- **Předpoklady:** Uživatel s daným ID má alespoň jeden komentář v systému.
- **Očekávaný výsledek:** Server odpoví statusem 200 OK a vrátí seznam komentářů tohoto uživatele, přičemž první komentář má text "Initial comment".

8.1.24 Test vytvoření komentáře (testCreateComment)

Scénář:

- Testuje úspěšné vytvoření nového komentáře.
- **Předpoklady:** Uživatel má přístup k úkolu, ke kterému může přidat komentář.
- **Očekávaný výsledek:** Server odpoví statusem 201 Created a vrátí vytvořený komentář s textem "New comment". Po vytvoření komentáře bude v databázi 2 komentáře.

8.1.25 Test aktualizace komentáře (testUpdateComment)

Scénář:

- Testuje úspěšnou aktualizaci existujícího komentáře.
- **Předpoklady:** Komentář s daným ID existuje v systému.
- **Očekávaný výsledek:** Server odpoví statusem 200 OK a vrátí aktualizovaný komentář s textem "Updated comment". Komentář v databázi bude aktualizován na "Updated comment".

8.1.26 Test smazání komentáře (testDeleteComment)

Scénář:

- Testuje úspěšné smazání komentáře.

- **Předpoklady:** Komentář s daným ID existuje v systému.
- **Očekávaný výsledek:** Server odpoví statusem 204 No Content a komentář bude odstraněn z databáze.

8.1.27 Test aktualizace neexistujícího komentáře (testUpdateNonExistentComment)

Scénář:

- Testuje pokus o aktualizaci komentáře, který neexistuje.
- **Předpoklady:** Komentář s daným ID neexistuje v systému.
- **Očekávaný výsledek:** Server odpoví statusem 404 Not Found.

8.1.28 Test neautorizovaného uživatele při pokusu o přidání komentáře (testUnauthorizedUserCannotCommentOnTask)

Scénář:

- Testuje, zda neautorizovaný uživatel nemůže přidat komentář k úkolu.
- **Předpoklady:** Uživatel s daným uživatelským jménem není autorizován pro přidávání komentářů.
- **Očekávaný výsledek:** Server odpoví statusem 401 Unauthorized.

8.1.29 Testování získání všech performance reportů (testGetAllPerformanceReports)

Scénář:
Testuje, zda se správně vrátí všechny performance reporty.

Předpoklady:

- Uživatel je autentizován s rolí "ADMIN".
- V systému existují nějaké performance reporty.

Očekávaný výsledek:

- Server vrátí status 200 OK a seznam performance reportů.

8.1.30 Testování získání performance reportů podle task ID (testGetPerformanceReportsByTaskId)

Scénář:

Testuje, zda se správně vrátí performance reporty pro daný task.

Předpoklady:

- Uživatel je autentizován s rolí "ADMIN".
- Task, na který se odkazuje, má přiřazený performance report.

Očekávaný výsledek:

- Server vrátí status 200 OK a seznam performance reportů pro daný task.

8.1.31 Testování získání performance reportů podle user ID (testGetPerformanceReportsByUserId)

Scénář:

Testuje, zda se správně vrátí performance reporty pro daného uživatele.

Předpoklady:

- Uživatel je autentizován s rolí "ADMIN".
- Uživatelský účet má přiřazený performance report.

Očekávaný výsledek:

- Server vrátí status 200 OK a seznam performance reportů pro daného uživatele.

8.1.32 Testování vytvoření performance reportu (testCreatePerformanceReport)

Scénář:

Testuje, zda uživatel s rolí "LEADER" může vytvořit nový performance report.

Předpoklady:

- Uživatel je autentizován s rolí "LEADER".
- Task a uživatel existují.

Očekávaný výsledek:

- Server vrátí status 201 Created a nový performance report bude uložen v databázi.

8.1.33 Testování aktualizace performance reportu (testUpdatePerformanceReport)

Scénář:

Testuje, zda uživatel s rolí "LEADER" může aktualizovat existující performance report.

Předpoklady:

- Uživatel je autentizován s rolí "LEADER".
- Performance report existuje v databázi.

Očekávaný výsledek:

- Server vrátí status 200 OK a performance report bude aktualizován s novými údaji.

8.1.34 Testování smazání performance reportu (testDeletePerformanceReport)

Scénář:

Testuje, zda uživatel s rolí "LEADER" může smazat performance report.

Předpoklady:

- Uživatel je autentizován s rolí "LEADER".
- Performance report existuje v databázi.

Očekávaný výsledek:

- Server vrátí status 204 No Content a performance report bude smazán.

8.1.35 Testování aktualizace neexistujícího performance reportu (testUpdateNonExistentPerformanceReport)

Scénář:

Testuje, zda server vrátí správnou chybu, když se pokusíme aktualizovat neexistující performance report.

Předpoklady:

- Uživatel je autentizován s rolí "LEADER".
- Neexistující ID performance reportu.

Očekávaný výsledek:

- Server vrátí status 404 Not Found, protože report s daným ID neexistuje.

8.1.36 Testování získání všech projektů (testFindAllProjects)

Scénář:

Testuje, zda uživatel s rolí "ADMIN" může získat všechny projekty.

Předpoklady:

- Uživatel je autentizován s rolí "ADMIN".
- V systému existují nějaké projekty.

Očekávaný výsledek:

- Server vrátí status 200 OK a seznam projektů.

8.1.37 Testování vytvoření projektu (testCreateProject)

Scénář:

Testuje, zda uživatel s rolí "LEADER" může vytvořit nový projekt.

Předpoklady:

- Uživatel je autentizován s rolí "LEADER".
- Uživatelský účet má přiřazený projekt.

Očekávaný výsledek:

- Server vrátí status 201 Created a nový projekt bude uložen v databázi.

8.1.38 Testování aktualizace projektu (testUpdateProject)

Scénář:

Testuje, zda uživatel s rolí "LEADER" může aktualizovat existující projekt.

Předpoklady:

- Uživatel je autentizován s rolí "LEADER".
- Projekt existuje v databázi.

Očekávaný výsledek:

- Server vrátí status 200 OK a projekt bude aktualizován s novými údaji.

8.1.39 Testování smazání projektu (testDeleteProject)

Scénář:

Testuje, zda uživatel s rolí "LEADER" může smazat projekt.

Předpoklady:

- Uživatel je autentizován s rolí "LEADER".
- Projekt existuje v databázi.

Očekávaný výsledek:

- Server vrátí status 204 No Content a projekt bude smazán.

8.1.40 Testování aktualizace neexistujícího projektu (testUpdateNonExistentProject)

Scénář:

Testuje, zda server vrátí správnou chybu, když se pokusíme aktualizovat neexistující projekt.

Předpoklady:

- Uživatel je autentizován s rolí "LEADER".
- Neexistující ID projektu.

Očekávaný výsledek:

- Server vrátí status 404 Not Found, protože projekt s daným ID neexistuje.

8.1.41 Testování vytvoření projektu s neexistujícím uživatelským ID (testCreateProjectWithNonExistentUserId)

Scénář:

Testuje, zda server vrátí správnou chybu, když se pokusíme vytvořit projekt s neexistujícím uživatelským ID.

Předpoklady:

- Uživatel je autentizován s rolí "LEADER".
- Neexistující uživatelské ID.

Očekávaný výsledek:

- Server vrátí status 404 Not Found, protože uživatel s tímto ID neexistuje.

8.1.42 Testování získání projektů podle uživatelského ID v úkolech s více projekty (testFindProjectsByAppUserIdInTasksWithMultipleProjects)

Scénář:
Testuje, zda uživatel s rolí "EMPLOYEE" může získat projekty, které jsou přiřazeny k úkolům, které tento uživatel má.

Předpoklady:

- Uživatel je autentizován s rolí "EMPLOYEE".
- V systému existují projekty a úkoly, které jsou přiřazeny tomuto uživatelskému účtu.

Očekávaný výsledek:

- Server vrátí status 200 OK a seznam projektů, které jsou přiřazeny k úkolům daného uživatele.

8.1.43 Testování získání projektů podle uživatelského ID v úkolech bez projektů (testFindProjectsByAppUserIdInTasksWithNoProjects)

Scénář:
Testuje, zda uživatel s rolí "EMPLOYEE" dostane prázdný seznam projektů, pokud nemá žádné přiřazené projekty.

Předpoklady:

- Uživatel je autentizován s rolí "EMPLOYEE".
- Tento uživatel nemá žádné projekty přiřazené v úkolech.

Očekávaný výsledek:

- Server vrátí status 200 OK a prázdný seznam projektů.

8.1.44 Testování získání projektů podle uživatelského ID (testFindAppUserProjects)

Scénář:
Testuje, zda uživatel s rolí "EMPLOYEE" může získat své projekty podle svého uživatelského ID.

Předpoklady:

- Uživatel je autentizován s rolí "EMPLOYEE".
- Tento uživatel má přiřazené projekty.

Očekávaný výsledek:

- Server vrátí status 200 OK a seznam projektů přiřazených tomuto uživatelskému účtu.

8.1.45 Testování získání všech úkolů (testFindAllTasks)

Scénář:
Testuje, zda uživatel s rolí "ADMIN" může získat seznam všech úkolů.

Předpoklady:

- Uživatel je autentizován s rolí "ADMIN".
- V systému existují úkoly.

Očekávaný výsledek:

- Server vrátí status 200 OK a seznam úkolů.

8.1.46 Testování získání úkolů podle ID uživatele (testFindTasksByUserId)

Scénář:
Testuje, zda uživatel s rolí "EMPLOYEE" může získat úkoly přiřazené konkrétnímu uživateli.

Předpoklady:

- Uživatel je autentizován s rolí "EMPLOYEE".
- Uživatel má přiřazené úkoly.

Očekávaný výsledek:

- Server vrátí status 200 OK a seznam úkolů přiřazených uživateli.

8.1.47 Testování vytvoření úkolu (testCreateTask)

Scénář:
Testuje, zda uživatel s rolí "ADMIN" může vytvořit nový úkol.

Předpoklady:

- Uživatel je autentizován s rolí "ADMIN".
- Projekt je již vytvořen.

Očekávaný výsledek:

- Server vrátí status 201 Created a nový úkol bude uložen v databázi.
- Úkol bude mít správný název a detaily.

8.1.48 Testování aktualizace úkolu (testUpdateTask)

Scénář:
Testuje, zda uživatel s rolí "ADMIN" může aktualizovat existující úkol.

Předpoklady:

- Uživatel je autentizován s rolí "ADMIN".
- Úkol existuje v databázi.

Očekávaný výsledek:

- Server vrátí status 200 OK a úkol bude aktualizován s novými detaily.

8.1.49 Testování smazání úkolu (testDeleteTask)

Scénář:
Testuje, zda uživatel s rolí "ADMIN" může smazat úkol.

Předpoklady:

- Uživatel je autentizován s rolí "ADMIN".
- Úkol existuje v databázi.

Očekávaný výsledek:

- Server vrátí status 204 No Content a úkol bude smazán.

8.1.50 Testování získání úkolů podle ID projektu (testFindByProjectId)

Scénář:
Testuje, zda uživatel s rolí "ADMIN" může získat úkoly přiřazené konkrétnímu projektu.

Předpoklady:

- Uživatel je autentizován s rolí "ADMIN".
- Projekt obsahuje úkoly.

Očekávaný výsledek:

- Server vrátí status 200 OK a seznam úkolů přiřazených projektu.

8.1.51 Testování filtrace úkolů podle minimálního data splatnosti (testFindByProjectIdWithMinDueDateFilter)

Scénář:
Testuje, zda uživatel s rolí "ADMIN" může filtrovat úkoly podle minimálního data splatnosti.

Předpoklady:

- Uživatel je autentizován s rolí "ADMIN".
- Úkoly mají různá data splatnosti.

Očekávaný výsledek:

- Server vrátí status 200 OK a seznam úkolů, které mají datum splatnosti po zadaném minimálním datu.

8.1.52 Testování filtrace úkolů podle maximálního data splatnosti (testFindByProjectIdWithMaxDueDateFilter)

Scénář:
Testuje, zda uživatel s rolí "ADMIN" může filtrovat úkoly podle maximálního data splatnosti.

Předpoklady:

- Uživatel je autentizován s rolí "ADMIN".
- Úkoly mají různá data splatnosti.

Očekávaný výsledek:

- Server vrátí status 200 OK a seznam úkolů, které mají datum splatnosti před zadaným maximálním datem.

8.1.53 Testování filtrace úkolů podle minimálního a maximálního data splatnosti (testFindByProjectIdWithMinAndMaxDueDateFilter)

Scénář:

Testuje, zda uživatel s rolí "ADMIN" může filtrovat úkoly podle minimálního a maximálního data splatnosti.

Předpoklady:

- Uživatel je autentizován s rolí "ADMIN".
- Úkoly mají různá data splatnosti.

Očekávaný výsledek:

- Server vrátí status 200 OK a seznam úkolů, které odpovídají zadaným filtrům.

8.1.54 Testování filtrace úkolů bez výsledků pro filtr splatnosti (testFindByProjectIdWithNoResultsForDueDateFilter)

Scénář:

Testuje, zda uživatel s rolí "ADMIN" dostane prázdný seznam úkolů, pokud žádný úkol nevyhovuje zadaným filtrům splatnosti.

Předpoklady:

- Uživatel je autentizován s rolí "ADMIN".
- Neexistují žádné úkoly, které by splňovaly zadané filtry.

Očekávaný výsledek:

- Server vrátí status 200 OK a prázdný seznam.

8.1.55 Testování vytvoření úkolu s neplatným projektem (testCreateTaskWithInvalidProject)

Scénář:

Testuje, zda uživatel s rolí "ADMIN" obdrží chybu při pokusu o vytvoření úkolu s neplatným ID projektu.

Předpoklady:

- Uživatel je autentizován s rolí "ADMIN".
- Projekt s ID 9999 neexistuje.

Očekávaný výsledek:

- Server vrátí status 404 Not Found.

8.1.56 Testování aktualizace neexistujícího úkolu (testUpdateNonExistentTask)

Scénář:
Testuje, zda uživatel s rolí "EMPLOYEE" obdrží chybu při pokusu o aktualizaci úkolu, který neexistuje.

Předpoklady:

- Uživatel je autentizován s rolí "EMPLOYEE".
- Úkol s ID 9999 neexistuje.

Očekávaný výsledek:

- Server vrátí status 404 Not Found.