# Návrh architektury pro projekt PraxeO

Aplikace je navržena jako vícevrstvá architektura typu klient–server, která využívá moderní webové technologie:

* Frontend: React + Next.js
* Backend: Java Spring Boot
* Databáze: PostgreSQL
* Komunikace: REST API s formátem JSON
* Autentizace: JWT (JSON Web Token)
* Nasazení: Docker – 3 samostatné kontejnery (frontend, backend, databáze)
* Certifikace: HTTPS pro šifrovanou komunikaci

Systém bude rozdělen do tří hlavních vrstev:

* Prezentační vrstva (Frontend)
  + Realizována pomocí React/Next.js – umožňuje server-side rendering i klientskou interaktivitu.
  + Komunikuje s backendem přes REST API.
  + Zajišťuje přihlášení, zobrazení praxí, úkolů, hodnocení a reportů.
  + Obsahuje dynamické routování (Next.js Pages/Routes).
  + Kontejner běží v samostatném Docker image.
* Aplikační vrstva (Backend)
  + Implementována v Java Spring Boot.
  + Obsahuje logiku systému, řízení přístupových práv a zpracování požadavků z frontendu.
  + Poskytuje API endpointy pro entity:

Uživatel, Role, Praxe, Úloha, Komentář, Posudek, Report.

* + Odesílá e-maily (např. při registraci) pomocí modulu Spring Mail.
  + Běží ve vlastním Docker kontejneru.
* Datová vrstva (Databáze)
  + Relační databáze PostgreSQL.
  + Spravuje data o uživatelích, praxích, úlohách, posudcích a exportech.
  + Přístup zajištěn přes ORM Spring Data JPA.
  + Databáze běží ve třetím kontejneru.
* Zabezpečení a ochrana dat
  + Všechny požadavky běží přes HTTPS.
  + Autentizace a autorizace přes JWT tokeny.
  + Role: student, učitel, externista, admin.
  + GDPR: uživatelé při registraci potvrzují souhlas se zpracováním osobních údajů; podporováno „právo být zapomenut“ (anonymizace).
  + Nahrávání souborů omezeno na bezpečné přípony (.txt, .docx, .pdf, .png, .jpg) a maximální velikost.
* Mailová komunikace
  + Systém zasílá ověřovací e-mail při registraci uživatelů:

Student a učitel se mohou registrovat pouze s univerzitním e-mailem (@student.osu.cz, @osu.cz).

* + Uživatel potvrdí registraci kliknutím na odkaz zaslaný na uvedený e-mail.
  + Možnost rozšíření o notifikace (např. přidání úkolu, změna stavu praxe).

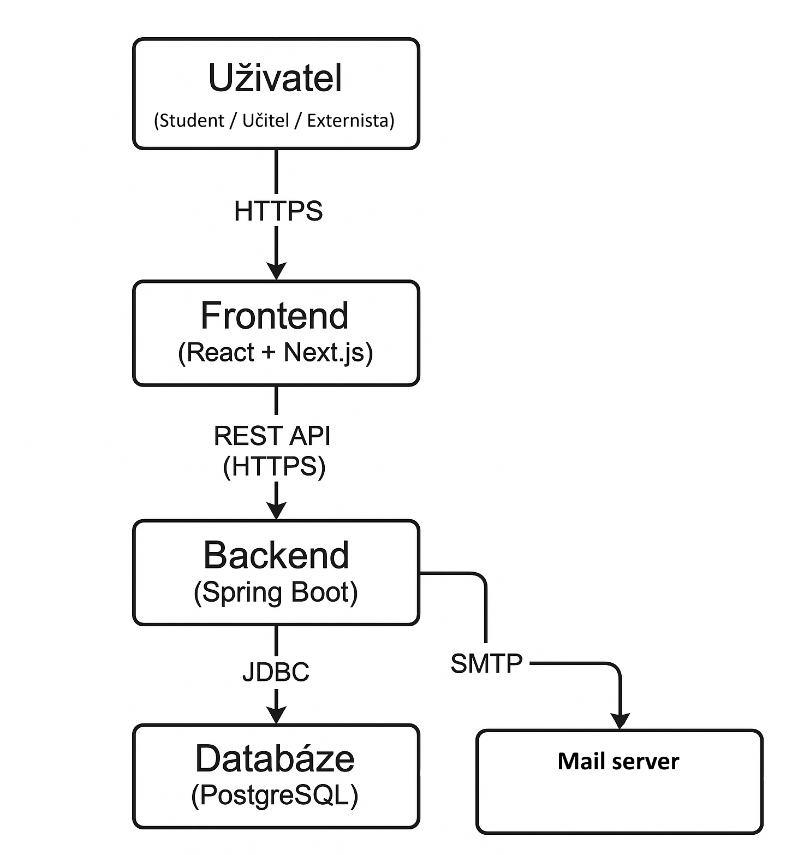
## Výhody a nevýhody architektury

**Výhody:**

* Jasné oddělení vrstev a přehledná struktura kódu.
* Snadné rozšiřování o nové moduly.
* Ověřený technologický stack (Java + React + PostgreSQL).
* Možnost nasazení do Dockeru a provozu na libovolném serveru.
* Podpora bezpečného přístupu pomocí JWT a řízení rolí.

**Nevýhody:**

* Vyšší komplexita při vývoji (více technologií).
* Vyšší nároky na prostředky (3 samostatné služby).
* Nutnost udržovat konzistenci mezi frontendem a API.



## Použité nástroje

Pro vývoj projektu PraxeO byl zvolen soubor moderních vývojových nástrojů, které podporují týmovou spolupráci, verzování kódu a efektivní řízení vývoje:

1. IntelliJ IDEA  
   Prostředí IntelliJ IDEA od společnosti JetBrains bylo zvoleno jako hlavní vývojové IDE pro implementaci backendu v jazyce Java.  
   Nabízí rozsáhlou podporu pro framework Spring Boot, správu závislostí pomocí Maven či Gradle a integrované nástroje pro testování a verzování kódu.

Oficiální stránka: <https://www.jetbrains.com/idea/>

1. Azure DevOps  
   Pro řízení vývoje, backlogu, sprintů a přiřazování úkolů využíváme platformu Microsoft Azure DevOps, která poskytuje přehledné rozhraní pro správu epiků, funkcí, user stories a tasků.  
   Zde probíhá také sledování postupu prací a verzování pomocí napojení na GitHub.

Projekt DevOps: <https://dev.azure.com/okonal23/PraxeO>

1. Git a GitHub  
   Pro verzování zdrojových kódů je použit systém Git s hostovaným repozitářem na GitHubu.  
   Git umožňuje sledovat změny v kódu, vytvářet větve (branching), provádět sloučení (merge) a zajišťuje tak bezpečnou spolupráci více vývojářů bez ztráty dat.

Oficiální Git: <https://git-scm.com/>  
GitHub repozitář projektu: <https://github.com/Krizkova/PraxeO>

## Závěr návrhu architektury

Pro zvolenou architekturu jsme se rozhodli na základě dosavadních znalostí a zároveň s cílem naučit se efektivně spolupracovat v týmu a osvojit si nové technologie. Vícevrstvé řešení oddělující frontend, backend a databázi nám umožní rozdělit práci tak, aby se jednotliví členové mohli soustředit na konkrétní část systému, aniž by si navzájem zasahovali do kódu. Zároveň nám tento přístup poskytne možnost porozumět celému vývojovému procesu — od návrhu uživatelského rozhraní přes implementaci serverové logiky až po práci s databází. Architektura tak představuje nejen vhodné technické řešení pro realizaci projektu PraxeO, ale i příležitost k rozvoji praktických dovedností v moderním webovém vývoji.