VSCode Open Project Rozšírenie Technický Report (Backend & API)

1 Cieľ projektu

Hlavným cieľom tohto projektu bolo vytvoriť funkčné prepojenie medzi vývojovým prostredím Visual Studio Code a nástrojom na správu projektov OpenProject. Moja časť práce sa sústredila na vytvorenie robustnej komunikačnej vrstvy, ktorá zabezpečuje autentifikáciu, sťahovanie dát a ich transformáciu pre potreby používateľského rozhrania.

2 Prehľad realizovanej práce (Backendová integrácia)

V rámci tímovej spolupráce som prebral zodpovednosť za „motor“ celého rozšírenia. Práca si vyžadovala nielen programovanie v jazyku TypeScript, ale aj hlboké pochopenie OpenProject API v3 a prácu s vývojovým prostredím v kontajneroch.

2.1 Príprava prostredia a Docker

Jednou z prvých výziev bolo správne nastavenie lokálneho vývojového prostredia.

- Musel som sa naučiť pracovať s nástrojom Docker, aby som mohol lokálne spúšťať inštanciu OpenProjectu.
- Toto riešenie nám umožnilo testovať API volania (GET, POST, PATCH) bez rizika ovplyvnenia reálnych dát a poskytlo nám plnú kontrolu nad databázou úloh.

2.2 Implementácia ApiClient.ts

Srdcom backendu je trieda ApiClient, ktorú som kompletne navrhol a implementoval. Zabezpečuje:

- Autentifikáciu: Implementoval som bezpečné overenie cez API kľúč pomocou Basic Auth (Base64 kódovanie).
- Komunikačnú vrstvu: Použil som knižnicu Axios na vykonávanie asynchrónnych požiadaviek.
- Správu dát (CRUD): * getProjects: Získavanie zoznamu projektov s podporou filtrovania (napr. len koreňové projekty).
    - getWorkPackages: Načítanie úloh pre konkrétny projekt.
    - updateWorkPackage: Implementácia PATCH požiadaviek, kde som musel vyriešiť logiku lockVersion, aby nedochádzalo ku konfliktom pri prepisovaní dát viacerými používateľmi.

2.3 Optimalizácia a Lazy Loading v TreeProvideri

V súbore workPackageTreeProvider.ts som sa zameral na efektivitu zobrazenia dát v bočnom paneli VS Code.

- Lazy Loading: Implementoval som metódu getChildren, ktorá nenačítava všetky dáta naraz. K dopytovaniu API dochádza až vo chvíli, keď používateľ rozbalí konkrétny projekt alebo úlohu.
- Hierarchická štruktúra: Navrhol som logiku, ktorá rozlišuje medzi projektmi, podprojektmi a úlohami (Work Packages). Úlohy sú následne filtrované tak, aby sa správne zobrazovali ich rodičovské a dcérske vzťahy.
- Cachovanie: Použil som Map (wpCache) na dočasné ukladanie načítaných úloh, čo výrazne znižuje počet potrebných API volaní pri opätovnom prezeraní stromu.

3 Technické detaily a dátové modely

Pre zabezpečenie typovej bezpečnosti v celom projekte som definoval rozhrania (interfaces) v súbore types.ts.

- HAL JSON Formát: OpenProject API využíva formát HAL (Hypertext Application Language). Musel som implementovať logiku na spracovanie prepojení (_links), ktoré API vracia, aby sme sa vedeli navigovať medzi zdrojmi.
- Číselníky: Implementoval som metódy pre sťahovanie metaúdajov, ako sú Priority, Statusy a Typy úloh, ktoré kolega následne využíva v dropdown menu vo frontende.

4 Záver

Práca na backende tohto rozšírenia mi priniesla cenné skúsenosti s asynchrónnym programovaním v TypeScripte a integráciou komplexných REST API. Musel som sa popasovať s autentifikáciou, spracovaním chýb a optimalizáciou výkonu cez lazy loading. Výsledkom je stabilná integračná vrstva, ktorá umožňuje plynulú prácu s projektmi priamo v prostredí editora.