**Základy webových aplikací – projekt**

Jméno: **Ondřej Dostál**

Třída: **I2A**

Školní rok: **2024/25**Název: **Mango**

Téma: **Správce projektů/úkolů**

URL: [**https://mango.kys.monster**](https://mango.kys.monster/)

# Téma

**Správce projektů/úkolů** –Interaktivní aplikace pro správu projektů nebo úkolů pro jednotlivce a týmy.Toto téma jsem si vybral z mnoha důvodů, ale toto jsou ty hlavní:

1. Může mi pomoct při tvorbě webových aplikací a dalších různých projektů.
2. Viděl jsem to jako „challenge“ – vytvořit plně interaktivní aplikaci na webu bez použití jakýchkoliv knihoven. Celá aplikace se skládá pouze z HTML, CSS, JavaScriptu a PHP.

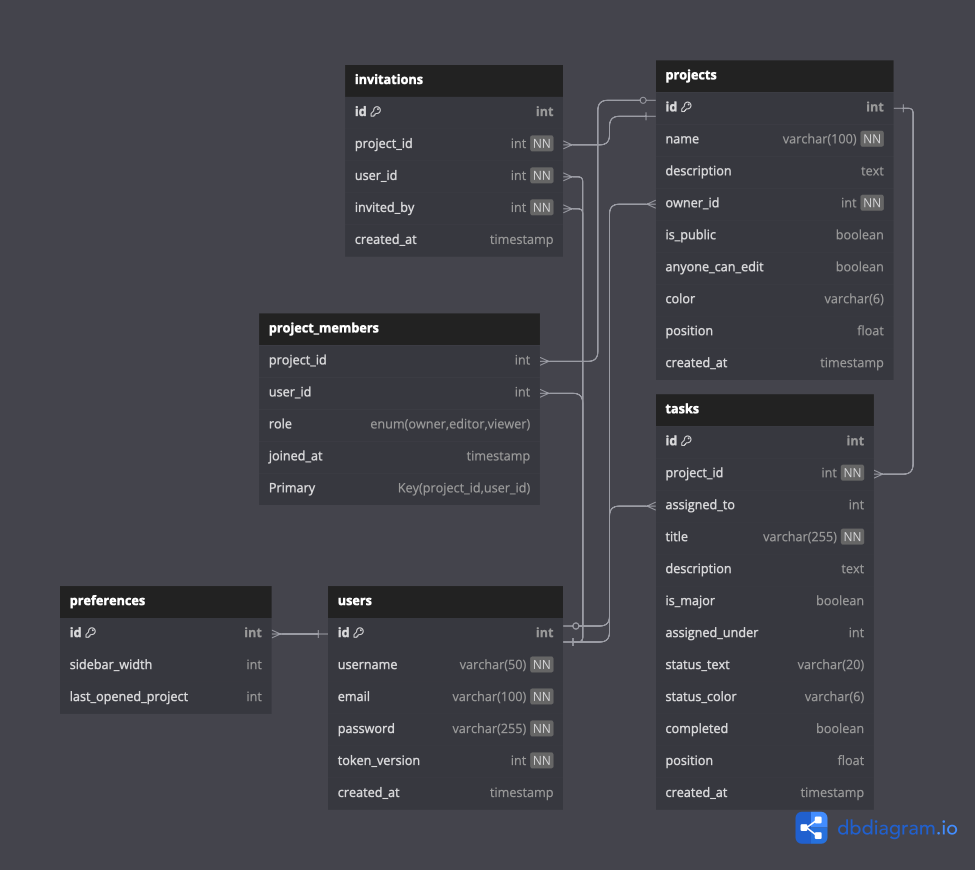
# Postup při tvorbě

Začal jsem vytvořením základní struktury projektu. Dále následovalo vytvoření jednoduchého routeru, který budu i po sléze popisovat. Poté jsem vytvořil základní komponenty, ze kterých se bude aplikace skládat (navbar, footer, …). V tuto chvíli už jsem začal řešit autentizaci uživatelů pomocí JWT. Pokud vás zajímá více, tak celý progres můžete najít na [GitHubu](https://github.com/odczik/ZWA-task-management-project).

# Databáze

Nejdůležitějším principem databáze jsou projekty a jejich úkoly. Každý úkol obsahuje id projektu, pod který spadá. Každý úkol taky obsahuje data o tom, zda je major (sloupec) a pokud ne, pod jaký major úkol spadá. Tyto data jsou poté poslány klientovi, který si je seřadí a zobrazí je. Dalším důležitým atributem je pozice úkolu, aby si je uživatel mohl seřadit a přesouvat je. Tabulky ovšem obsahují pár atributů, které zatím nebyly využity, a proto je můžete ignorovat. Na podobném principu také fungují členové projektů, kdy každý člen má záznam v tabulce obsahující údaje o tom, k jakému projektu patří a jakou v něm zaujímají roli.

Celé schéma databáze:



# Struktura aplikace

# Popis vybraného PHP kódu

Obsah obrázku text, snímek obrazovky, displej, software

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Tento kód vytváří velice základní router. Prvně se vytvoří class Router která ukládá všechny naše routy společně s callback funkcí. Poté se zavolá funkce dispatch, která se momentální request (route + method) pokusí najít v listu našich routů. Pokud ho najde zavolá jeho callback funkci (většinou render stránky nebo api endpoint) jinak vrátí chybu 404.

Body pro zlepšení:

1. Dynamické routy
2. Podpora middlewaru

# Základní HTML struktura

Všechny stránky vesměs používají tu samou HTML strukturu:

…

<**body**>

<**nav**>…</**nav**>

<**main**>

// hlavní obsah

</**main**>

<**footer**>…</**footer**>

</**body**>

# Rozbor vybraného CSS

# Manuál a screenshoty