

Vylepšený task manager

Předmět: Testování softwaru

Datum: 21.2.2026

Autor: Lucie Mazánková

Cílem projektu bylo rozšířit původní konzolovou aplikaci pro správu úkolů tak, aby data nebyla ukládána pouze do paměti programu, ale do relační databáze MySQL. Aplikace nyní pracuje s databází a podporuje plné CRUD operace (Create, Read, Update, Delete). Součástí řešení je také sada automatizovaných testů vytvořených pomocí knihovny pytest.

Projekt propojuje programovací jazyk Python s databázovým systémem MySQL a demonstruje práci s databázovým připojením, SQL dotazy i základní testování aplikace.

Použité technologie

- Programovací jazyk: Python 3
- Databáze: MySQL
- Knihovna pro připojení: mysql-connector-python
- Testování: pytest
- Nástroj pro správu databáze: MySQL Workbench

Nejprve je potřeba mít nainstalovaný balíček:
`pip install mysql-connector-python`

Databázová struktura

Byla vytvořena databáze `ukoly_db`, která obsahuje tabulku `ukoly`.

Struktura tabulky:

- `id` – primární klíč, `AUTO_INCREMENT`
- `nazev` – název úkolu (povinný údaj)
- `popis` – popis úkolu (povinný údaj)
- `stav` – ENUM ('Nezahájeno', 'Probíhá', 'Hotovo'), výchozí hodnota 'Nezahájeno'
- `datum_vytvoreni` – datum a čas vytvoření (automaticky generované)

Tabulka je vytvořena automaticky při spuštění programu, pokud ještě neexistuje.

Funkcionalita aplikace

Aplikace je rozdělena do jednotlivých funkcí:

pripojeni_db()

Zajišťuje připojení k MySQL databázi. V případě chyby zobrazí chybovou hlášku.

vytvoreni_tabulky()

Vytvoří tabulku `ukoly`, pokud ještě neexistuje.

hlavni_menu()

Zobrazuje uživateli nabídku možností:

1. Přidat úkol
2. Zobrazit úkoly
3. Aktualizovat úkol
4. Odstranit úkol
5. Ukončit program

Program kontroluje neplatné vstupy a vyžaduje opakovanou volbu.

pridat_ukol()

- Kontroluje, že název i popis nejsou prázdné.
- Úkol je uložen do databáze.
- ID a datum vytvoření se generují automaticky.
- Výchozí stav je nastaven na „Nezahájeno“.

zobrazit_ukoly()

- Zobrazuje seznam úkolů.
- Obsahuje filtr – zobrazí pouze úkoly se stavem „Nezahájeno“ nebo „Probíhá“.
- Pokud nejsou žádné úkoly, zobrazí informaci o prázdném seznamu.

aktualizovat_ukol()

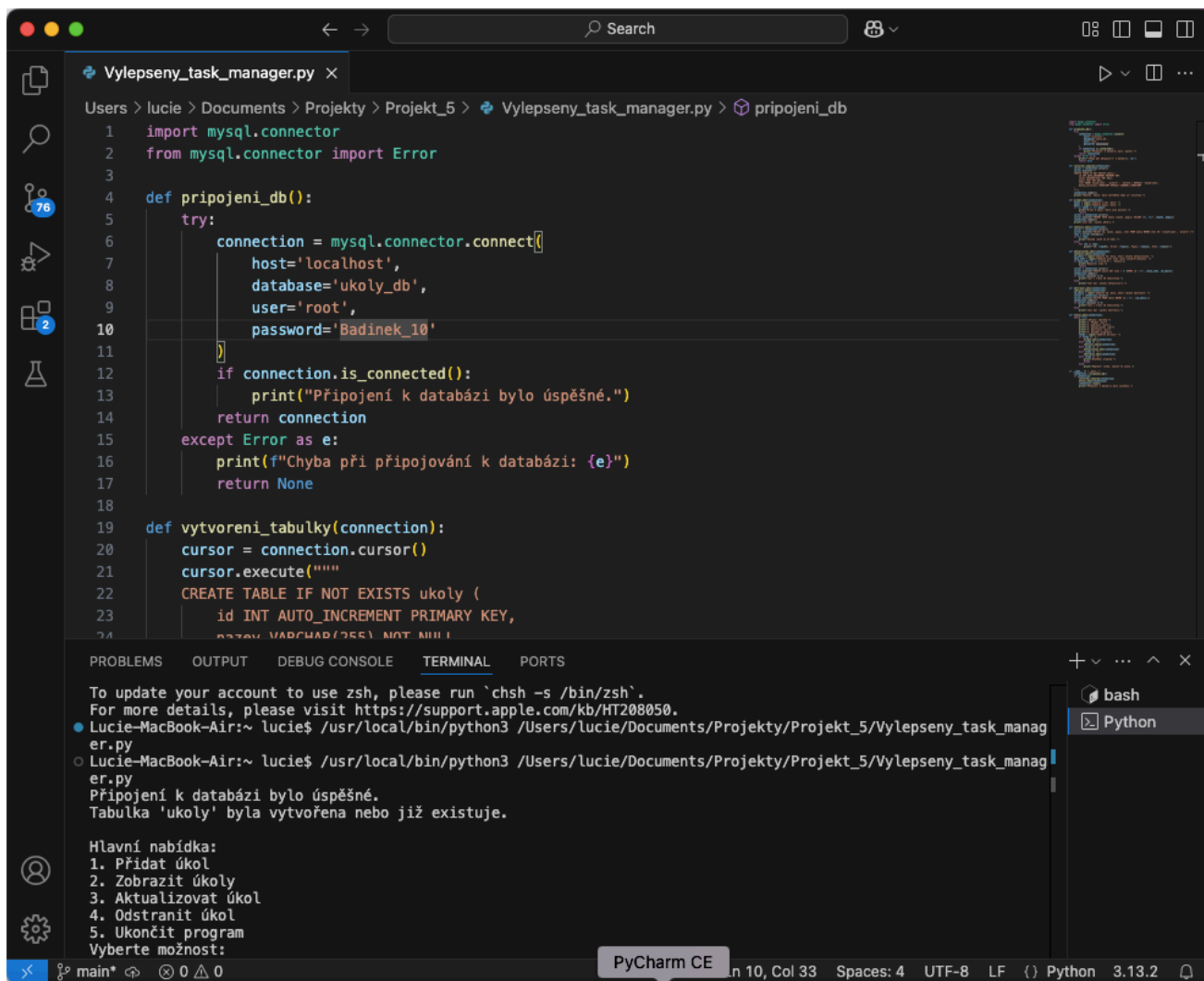
- Uživatel vybírá úkol podle ID.
- Lze změnit stav na „Probíhá“ nebo „Hotovo“.
- Program kontroluje existenci zadaného ID.

odstranit_ukol()

- Uživatel vybírá úkol podle ID.
- Úkol je trvale odstraněn z databáze.
- Při zadání neexistujícího ID je uživatel upozorněn.

Toto řešení představuje jednoduchého správce úkolů s využitím Pythonu a databáze MySQL. Program je rozdělen do jednotlivých funkcí, které zajišťují připojení k databázi, vytvoření potřebné tabulky a provádění základních CRUD operací – přidání, zobrazení, aktualizace a odstranění úkolu. Uživatel s aplikací interaguje prostřednictvím textového menu v konzoli. Data se ukládají do tabulky ukoly, která obsahuje sloupce pro název, popis, stav úkolu a datum vytvoření. Pro práci s databází je použit modul mysql.connector a připojení je řešeno přes lokální server.

Tabulka ukoly v databázi byla nejprve vytvořena ručně ve vizuálním nástroji MySQL Workbench, kde byla definována její struktura – sloupce, datové typy i výchozí hodnoty. Následně je program schopen tabulku znovu vytvořit (pokud neexistuje), aby byl kód plně funkční i samostatně bez závislosti na předchozí ruční konfiguraci.



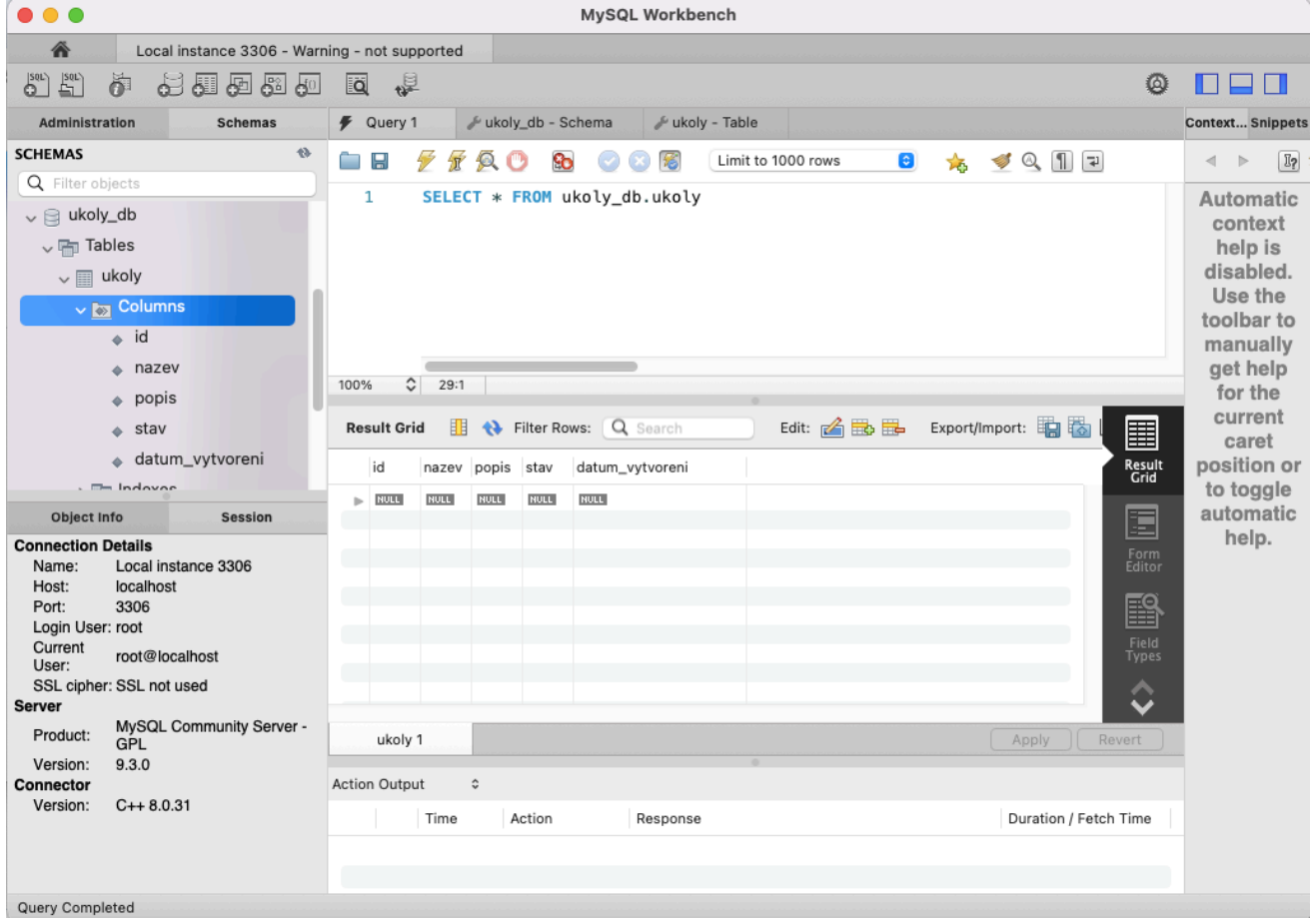
The screenshot displays the PyCharm CE IDE interface. The main editor window shows the file `Vylepseny_task_manager.py` with the following Python code:

```
1 import mysql.connector
2 from mysql.connector import Error
3
4 def pripojeni_db():
5     try:
6         connection = mysql.connector.connect(
7             host='localhost',
8             database='ukoly_db',
9             user='root',
10            password='Badinek_10'
11        )
12        if connection.is_connected():
13            print("Připojení k databázi bylo úspěšné.")
14            return connection
15    except Error as e:
16        print(f"Chyba při připojování k databázi: {e}")
17        return None
18
19 def vytvoreni_tabulky(connection):
20     cursor = connection.cursor()
21     cursor.execute("""
22     CREATE TABLE IF NOT EXISTS ukoly (
23         id INT AUTO_INCREMENT PRIMARY KEY,
24         name VARCHAR(255) NOT NULL,
25         description VARCHAR(255) NOT NULL,
26         status VARCHAR(20) NOT NULL,
27         created_at TIMESTAMP NOT NULL
28     )
29 """)
```

The bottom panel shows the terminal output, which includes a message about updating the account to use zsh, followed by the execution of the Python script. The output confirms the successful connection to the database and the creation of the 'ukoly' table. Below the terminal output, a main menu is displayed:

```
Hlavní nabídka:
1. Přidat úkol
2. Zobrazit úkoly
3. Aktualizovat úkol
4. Odstranit úkol
5. Ukončit program
Vyberte možnost:
```

The status bar at the bottom indicates the file is at line 10, column 33, using UTF-8 encoding and LF line endings, with the Python 3.13.2 interpreter selected.



Automatizované testování

Pro testování byla použita knihovna pytest.

Každá hlavní funkce (přidání, aktualizace, odstranění) obsahuje:

- 1 pozitivní test (ověření správné funkčnosti)
- 1 negativní test (ověření reakce na neplatný vstup)

Testy:

- Dynamicky vytvářejí testovací data.
- Po dokončení testu jsou testovací záznamy odstraněny.
- Nedochozí k trvalé změně databáze.

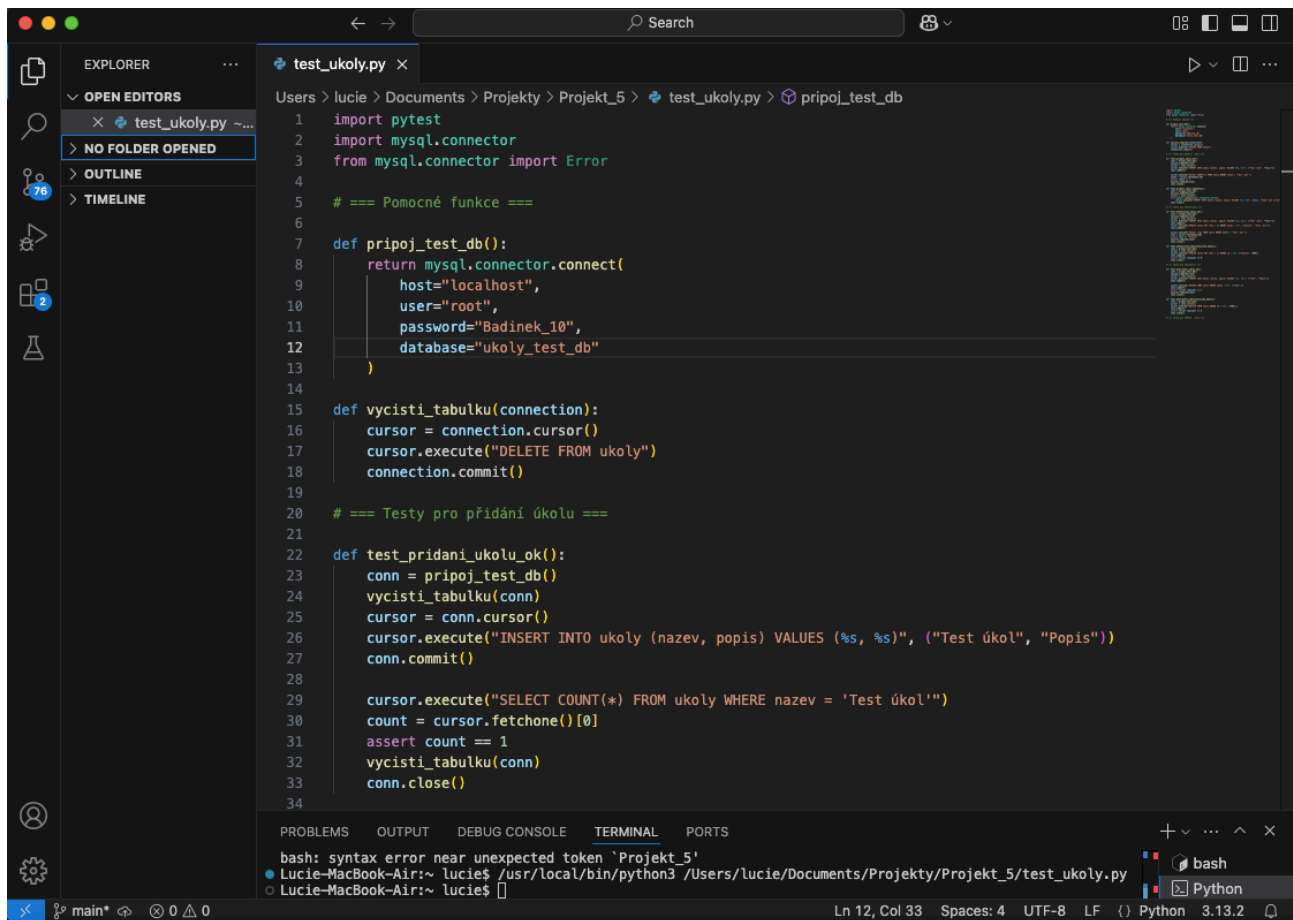
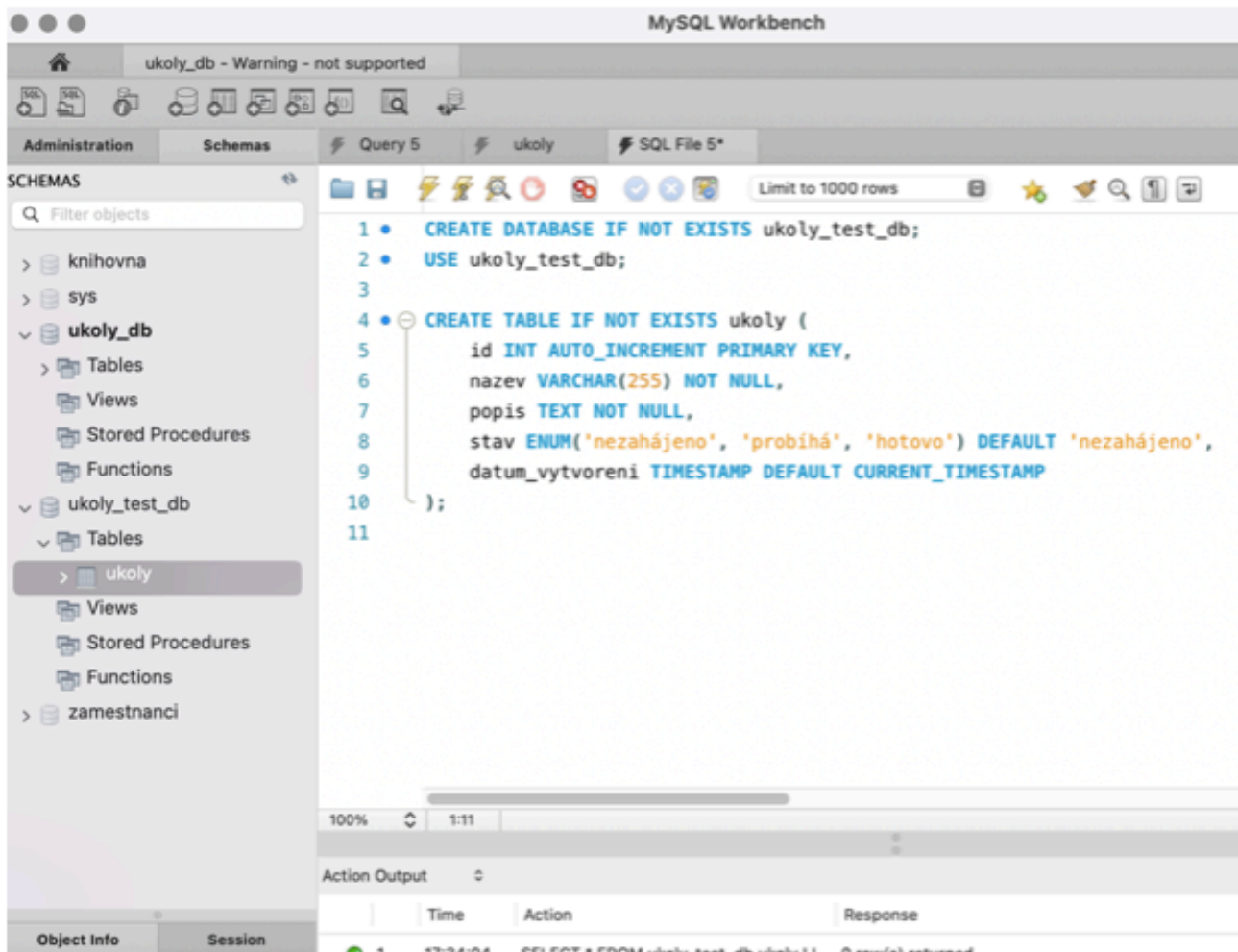
Byla použita samostatná testovací databáze, aby nedošlo k ovlivnění produkčních dat.

Vytvoření testovací databáze ukoly_test_db

Důležité (aby testy prošly)

Musíš mít:

- vytvořenou databázi ukoly_test_db
- vytvořenou tabulku ukoly
- běžící MySQL server
- správné heslo k root uživateli



```
test_ukoly.py::test_pridani_ukolu_ok PASSED
test_ukoly.py::test_pridani_ukolu_neplatny PASSED
test_ukoly.py::test_aktualizace_ukolu_ok PASSED
test_ukoly.py::test_aktualizace_neexistujiciho_ukolu PASSED
test_ukoly.py::test_odstraneni_ukolu_ok PASSED
test_ukoly.py::test_odstraneni_neexistujiciho_ukolu PASSED
```

Závěr

Projekt demonstruje praktické využití propojení Pythonu s relační databází MySQL. Byla implementována plnohodnotná CRUD aplikace s ošetřením chybových stavů a automatizovaným testováním.

Díky oddělení databázové logiky od uživatelského rozhraní je aplikace přehledná, snadno rozšiřitelná a testovatelná. Projekt přispěl k lepšímu pochopení práce s databázemi, SQL dotazy a testováním aplikací v Pythonu.