

# CV04 – GraphQL

Cílem cvičení je seznámit studenty s GraphQL. Alternativou k REST API.

## Předpoklady












- ☕ Znalost jazyka Java a OOP (minimálně Java 1.8).
- 🐙 Verzovací systém Git (orientace v GitHub nebo GitLab výhodou).
- 🧠 Pokročilejší orientace v modernějším IDE.
- 🖱️ Znalost Spring Boot z předchozích cvičení.
- 🐳 Funkční Docker (ideálně s Docker Desktop).
- ? Teoretická znalost REST API z předchozího cvičení.

## Výstupy




- Student je schopen napsat jednoduchý schéma soubor s query a mutacemi které následně namapuje do Spring ekosystému.
- Nastavit GraphQL.
- Jednoduché volání query dotazů.

## Zadání

1. 🖋️ Přidejte do projektu následující závislosti:
  - a. 🖋️ Spring for GraphQL.
2. Upravte **application.yaml** tak aby Spring Boot při inicializaci:
  - a. Načtl graphql schéma umístěné v rootu projektu pod **resources/graphql/schema.graphqls**.
    - i. Tento soubor vytvořte.
  - b. Pod URI **/api/v1/graphql** bylo možné volat POST a GET pro ovládání GraphQL.
  - c. Pod URI **/api/v1/graphiql** byl dostupný nástroj „**GraphiQL**“ pro testování query dotazů.
3. Vytvořte nový GraphQL controller s názvem **AppUserQLController.java/AppUserQLController.kt**:
  - a. Definujte v **schema.graphqls** nové query, které na základě **id v argumentu** vrátí daného uživatele.
    - i. Bude nutné definovat nový typ **AppUser**.
  - b. Vytvořte query mapping v **AppUserQLController**, které bude obsluhovat výše zmíněné query.
  - c. Otestujte přes GraphiQL.
4. 🧑🏫 Teoretická 5 minutovka.
5. Upravte typ **AppUser** ve **schema.graphqls** tak, aby obsahoval množinu entity **Task**.
  - a. Tedy tak, aby bylo možné se dotazovat i na úkoly u kterých je uživatel autorem.
  - b. **Novou funkcionalitu založte za pomoci anotace @SchemaMapping a nové metody v TaskQLController. Vyhněte se implementování této funkcionality do již existujících metod!**
  - c. ? Do komentáře nad třídu **AppUserQLController** napiště, co to je tzv. **N+1 problém** a jak jej řešit pomocí anotace **@BatchMapping**.
    - i. ? Uveďte **rozdíl** mezi **@SchemaMapping** a **@BatchMapping** a co musíme dodržet, aby anotace správně fungovala.
6. 🖋️ Založte novou mutaci, pro vytvoření nového uživatele:

- a.  Mutaci a nový **input** typ definujte v **schema.graphqls** souboru.
  - b.  V **AppUserQLController** namapujte novou mutaci a implementujte funkcionalitu.
    - i.  Setkáte se s komplikací pro atributy s datovým typem `LocalDateTime`. GraphQL neumí deserializovat `String` do `LocalDateTime`. Jsou dvě možnosti, jak to řešit:
      1.  Založíte nové DTO zvláště pro GraphQL. Zde budou datумы reprezentovány jako textové řetězce. Při převodu do entity pak převedete text na instanci `LocalDateTime`.
      2.  Prostudujte si, co to jsou tzv. [Scalars](#).
  - c.  Nastavte Spring Boot tak aby [reagoval na výjimky](#) vložení zprávy do **error fieldu** v odpovědi na GraphQL query. Místo toho, aby aplikace chybu jenom zalogovala.
7.  Do komentáře nad třídu `AppUserQLController` napište, co to je tzv. „**GraphQL subscription**“. U svých odpovědí se zaměřte zejména na:
- a.  Jak probíhá komunikace. Kdo je příjemcem a kdo je odesílatelem.
  - b.  Jaké anotace jsou použity pro její založení.
  - c.  Uveďte alespoň dva příklady využití v jakékoliv aplikaci.
  - d.  Co to je tzv. „web socket“ a jaký má vztah ke GraphQL subscription.

## Vysvětlivky

-  Samostatná práce s podporou vyučujícího.
-  Teoretická otázka. Odpověď studenti napíší do komentáře v kódu.
-  Teoretická 5 minutovka.

## Odkaz pro odevzdání úlohy:

<https://forms.gle/Y2qqZdcbvdP6xXFZA>

## Vzorová data

```
INSERT INTO public.app_user (id, active, creation_date, password, update_date, username) VALUES (0, true, '2023-03-07 14:14:33.000000', 'sdadsads', '2023-03-07 14:14:42.000000', 'dsadasasd');
```

```
INSERT INTO public.task (id, creation_date, description, title, update_date, author_id) VALUES (0, '2023-03-07 14:31:59.000000', 'task1', 'Title 1', '2023-03-07 14:32:20.000000', 0);
```

```
INSERT INTO public.task (id, creation_date, description, title, update_date, author_id) VALUES (1, '2023-03-07 14:32:01.000000', 'task2', 'Title 2', '2023-03-07 14:32:23.000000', 0);
```