Nápověda k projektu SGPrusaTD.

Projekt umožnuje pro projekty (skupiny) vytvářet seznam úkolů (To-Do) a sledovat jejich plnění.

**Git repository:** <https://github.com/sgolias/sgprusatd>

**Spuštění:** java -jar sgprusatd-0.0.1.jar r --server.port=9090

- běží na http://localhost:9090/sgprusatd/info

- defaultní port je 8080.

**Projekty:**

Projekty se zadávají v rámci vkládání úkolů. Jakmile se založí úkol do neexistujícího projektu, tak se založí záznam projektu s code\_proj získaného z úkolu.

**key** projektu je **code\_proj** z úkolu, například: TD. Je to hlavní identifikátor projektu.

**id\_proj** - číslo - se přiděluje projektu automaticky a je jen pro vnitřní identifikaci v rámci databáze.

**Úkoly:**

Základními parametry úkolů jsou:

**code\_proj** - kód projektu - defaultní hodnota je TD  
**description** - text úkolu - povinný (může být "", nesmí být null)

**status** - status úkolu - nepovinný, defaultní hodnota je DRAFT

**id\_todo** - číslo - se přiděluje projektu automaticky a samo o sobě je jen pro vnitřní identifikaci v rámci databáze. Je ale součástí **key** úkolu.

**key** - úkolu se skládá z **code\_proj** a **id\_todo** úkolu ve tvaru například TD-1254 (projekt TD, id úkolu 1254). Je to hlavní identifikátor úkolu.

**completed\_at** - časový záznam, kdy byl úkol splněn, tj. je COMPLETED. Je nevyplněný do doby, kdy status úkolu není COMPLETED. Pokud dojde k další změně statusu, tak se znovu vymaže.

**Audit:**

Každý záznam, tj. projekt i úkol má auditní záznamy. Protože není registrace uživatelů, who je vždy "me".

**ts\_who\_in** - kdo záznam vložil

**ts\_when\_in** - kdy se záznam vložil

**ts\_who** - kdo naposled záznam upravil

**ts\_who** - kdy se naposled záznam upravil

**REST API Endpointy aplikace:**

/sgprusatd/help - tento dokument

/sgprusatd/info - tento dokument

/sgprusatd/projects - seznam projektů

/sgprusatd/tasks - seznam úkolů

**Endpoint /sgprusatd/projects - seznam projektů:**

Endpoint má implementovanou metodu GET

GET:

/sgprusatd/projects - vrátí všechny projekty

/sgprusatd/projects/{key} - vrátí projekt definovaný dle key

**Endpoint /sgprusatd/tasks - seznam úkolů:**

Endpoint má implementované metody GET, POST, PUT, DELETE.

GET:

/sgprusatd/tasks - vrátí všechny úkoly

Je možno doplnit parametry vyhledávání úkolů dle

- status - pro zadání statusu úkolů a

- code\_proj - pro zadání projektu, do kterého úkol patří

Nezadané parametry se nezohledňují.

Příklad: /sgprusatd/tasks?status=COMPLETED&proj=TD

Vrátí všechny úkoly se statusem COMPELTED v projektu TD.

/sgprusatd/tasks/{key} - vrátí úkol odpovídající kódu key

POST:

/sgprusatd/tasks - vkládá úkol do aplikace

code\_proj - kód projektu - nepovinný, defaultní hodnota je TD

description - text úkolu - povinný (může být "", nesmí být null)

status - status úkolu - nepovinný, defaultní hodnota je DRAFT

Povolené hodnoty jsou: CANCELLED, DRAFT, STARTED, IN\_PROGRESS, PAUSED, COMPLETED

Příklad:  
{

"code\_proj": "PRG",

"description": "Úkol pro projekt PRG",

"status": "CANCELLED"

}

PUT:

/sgprusatd/tasks/{key} - upravuje úkol odpovídající kódu key

description - nový text ukolu - nepovinný

status - nový status úkolu - nepovinný

Nezadané nebo null hodnoty se nemění.

DELETE:

/sgprusatd/tasks/{key} - smaže úkol odpovídající kódu key

**Endpoint /sgprusatd/tasks/import - hromadný import úkolů:**

Endpoint má implementovanou metodu POST.

POST:

/sgprusatd/tasks/import - hromadný import úkolů v xml formátu v struktuře

<sgprusatd>

<tasks>

<task>

<code\_proj>IMP</code\_proj>

<description>Test importu CANCELLED</description>

<status>CANCELLED</status>

</task>

</tasks>

</sgprusatd>

**Poznámky od programátora:**

Aplikace využívá interní in-memory databázi založenou na ConcurrentHashMap z důvodu umožnění více

vláknového zpracování dat a postavení aplikace na ukládání dat dle key. Jediné prohledávání celých tabulek v aplikaci je při vyhledávání úkolů dle statusu a projektu. Pokud bychom chtěli zrychlit i tuto část, dala by vystavět pomocná struktura, která by byla obdobou indexu v databázích.

Fiktívně se vytvářejí 2 tabulky   
sgproj s PK id\_proj umístěná v objektu ProjectsStorage

sgtodo s PK id\_todo umístěná v objektu TasksStorage

Pro plnění id je definována sekvence tabID, která je společná pro všechny tabulky.

**Rozšířené zadání:**

API s nahráváním CSV: :-( omlouvám, se dělal sem po paměti a byl jsem v tom, že máme implementovat XML. U csv by to bylo obdobné, jen jako vstup by byl @PostMapping s MediaType.TEXT\_PLAIN\_VALUE zpracovaný třeba s OpenCSV.

Počet vyřešených úkolů po dnech: jsem nestihl (chtěl jsem, aby aplikace měla ucelený smysl), ale už by to nebyl problém. Bylo by to jako u seznamu dle projektu a statusu, akorát by se třeba ze dnů udělala mapa a každý splněný úkol by navyšoval uloženou hodnotu +1. Nakonec by se mapa převedla do GET.

**API s uložením dat v databázi:**

Při přechodu na ukládání dat do databáze by bylo nutné nejvíce zasáhnout do tříd ProjectsStorage a TasksStorage. Tyto třídy by bylo vhodné vzhledem na Spring Boot aplikaci nahradit JPA entitami, doplněnými o potřebné anotace a atributy pro Hibernate.  
Alternativně pokud bychom pracovali přímo s JDBC drivery, tak u nás v práci máme vystavěný systém založený na Datasouces, které jsou definované v xml. Tohle je ale asi interní objekt Elanoru.

<datasource name="ZalSmis">

<query><![CDATA[select $p{columns} from ceccish

where ceccish.ciselnik = $P{ciselnik}

]]></query>

<table>ceccish</table>

<fields>

<field path="ceccish.hodnota"/>

<field path="ceccish.zkr"/>

</fields>  
</datasource>

Aplikované v java

Memory m = getForm().readMem("ZalSmis"**,** "ciselnik "**, "**lang**"**)**;**Memory je kolekce row a hromady metadat.

Vůči databázi m.Save, m.SaveAndCommit, m.Commit.  
Memory ví, které záznamy byli změněny, přidány.

Má metody, jak z už načtených dat vracet jen některé, zadáním podmínek. ... .

Ještě jsem v projektu uvažoval o použití in-memory H2 databáze, která běží v paměti, ale umí ukládat i na disk.

Používá ji interně i datasource, protože umožňuje ukládat velké množství dat a nezaplní RAM.

Z reálných databází jsou použitelné asi libovolné Oracle, MS SQL (ty jsou ale placené), takže Postgres nebo mySQL je asi dobrá volba.

O NoSQL, třeba MongoDB, databázích moc neví. Ale hodí se na ukládání třeba dokumentů. Takže kdyby úkoly měli mít u sebe návody, výkresy, ..., tak ty by se dali ukládat do noSQL DB.