Memo

Závěrečná zpráva

Autoři: Daria Kuznetsova, Matěj Charousek

Popis aplikace

Naše aplikace Memo umožňuje ukládat vzpomínky a časové kapsle. Vzpomínka je běžný příspěvěk, časová kapsle se otevře po určitém čase, který uživatel nastaví. Kromě ukládání příspěvků (vzpomínek nebo časových kapslí) také umožňuje vytvářet skupiny a sdílet ve skupinách vzpomínky a společné časové kapsle. Vytvářet skupiny ale smí pouze privilegovaní uživatelé, tedy uživatelé s rolí premium. Ostatní uživatelé smí být členy skupin, ale musí je tam přidat tzv. premium uživatelé. Detailní popis aplikace včetně funkčních požadavků je k nalezení v dokumentu MemorSRS.pdf.

Struktura aplikace – vrstvy

Naše aplikace je dělena do těchto vrstev: **persistentní vrstva, vrstva business logiky, service, rest**. S aplikací komunikujeme pomocí HTTP dotazů skrze aplikace Postman.

Persistentní vrstva: je v našem případě package dao. Informace, jaké atributy jakých tříd má na databází namapovat získává pomocí anotací @Column, @Entity apod. Stará se o CRUD operace entit, propojení aplikace s databází. Využíváme zde @OrderBy, @NamedQuery a kaskádní create, remove.

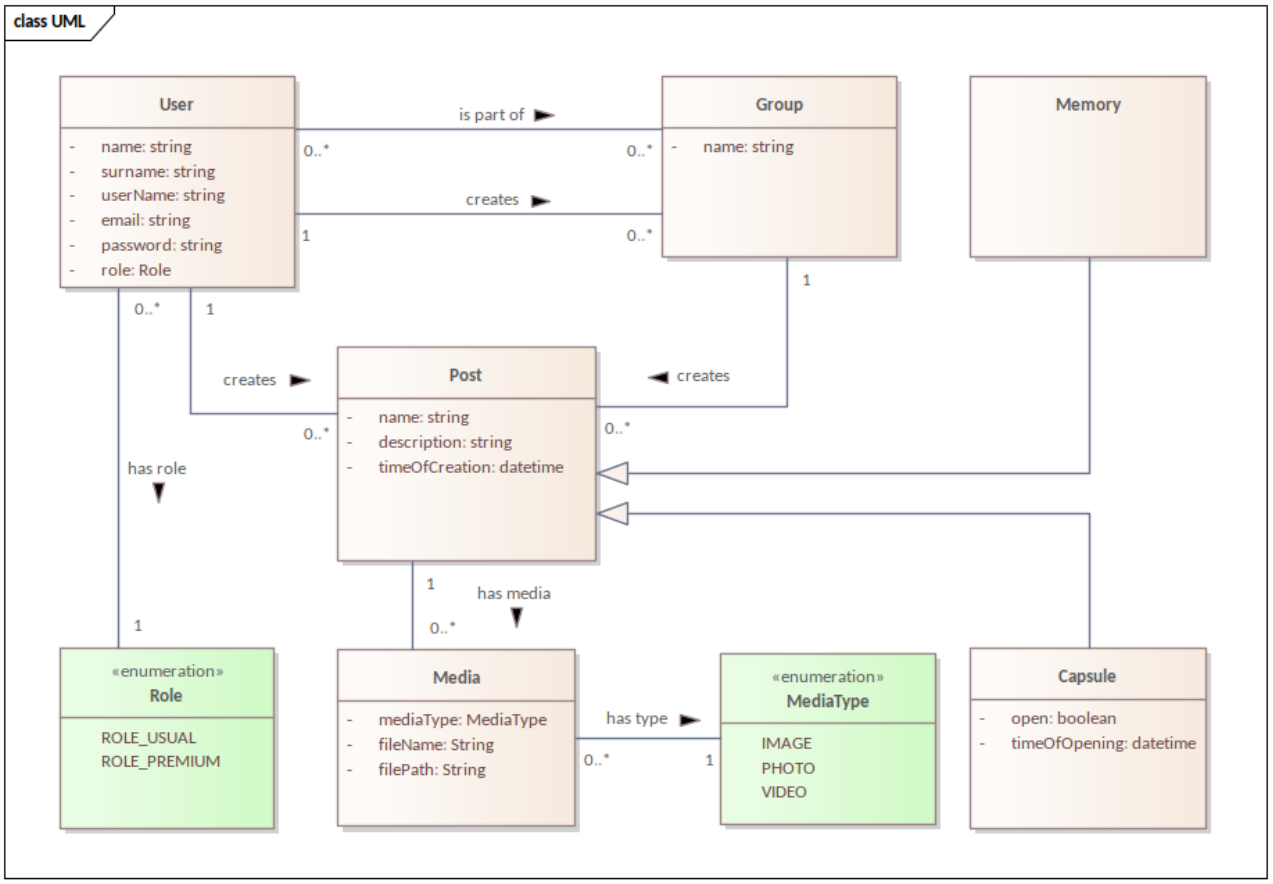
Vrstva business logiky: package model. Zde jsou naimplementované entity (viz. UML diagram).

Service vrstva: package service. Získává požadavky od rest vrstvy, konkretizuje požadavky a předává je dál do vrstvy business logiky, nebo přímo do DAO vrstvy a následně vrací výsledky zpět do rest vrstvy. Mám 3 service: PostService, GroupService, UserService,

Rest vrstva: package rest. Tato vrstva má v sobě konkrétní controllery, které reagují na http dotazy od uživatele. Z http dotazu zjistí požadavky a ty pak delegují na service vrstvu. Máme 3 controllery: PostController, GroupController a UserController. PostController zajišťuje vytváření příspěvků, takže vzpomínek i časových kapslí. GropController vytváří skupiny, umožňuje přidávání členů do skupin, odebírání ze skupin, mazání skupin a vypisování skupin. Tento controller nejvíce omezuje přístup, protože vytváření skupin je povoleno pouze uživatelům s premium účtem a zároveň pouze uživatel, kterou skupinu vytvořil, může přidávat a odebírat členy. UserController zajišťuje registraci, přihlašování, změnění role na premium, tedy zajišťuje, že omezení v GroupControlleru mohou fungovat.

Struktura aplikace – entity

Toto je UML diagram, na němž jsou vidět všechny zásadní entity naší aplikace.



Návod jak nainstalovat

Je potřebné stáhnout celou složku z git labu a následně spustit soubor TimeKeep/src/main/java/sir/timekeep/TimeKeepApplication.java. Po nějaké době se aplikace rozběhne (z naší zkušenosti zhruba 15-30 sekund). Potom už se dá využít aplikace postman a vykonat jednotlivé http dotazy. To, že se příkazy provedly lze kromě http dotazů get ověřit i nahlédnutím přímo do databáze. My jsme využívali databázi H2. V souboru TimeKeep/src/main/resources/application.properties je nastavení H2 databáze.

HTTP dotazy

Registrace uživatele:

POST http://localhost:8080/rest/register

```
{
"name": "Test",
"surname": "User1",
"username": "testuser",
"email": "testuser@seznam.cz",
"password": "testuser"
}
```

Přihlášení uživatele:
POST http://localhost:8080/login?username=testuser&password=testuser
Změna role na Premium:
PUT http://localhost:8080/rest/users/{user_id}/changeRole

Vytvoření skupiny:
POST http://localhost:8080/rest/{user_id}/groups

```
{
"name": "testGroup1",
"groupCreator" : { "id": 1}
}
```

Přidání uživatele do skupiny:
PUT http://localhost:8080/rest/{user_id}/groups/{group_id}/add_user/{add_user_id}

Odebrání uživatele ze skupiny:
PUT http://localhost:8080/rest/{user_id}/groups/{group_id}/remove_user/{remove_user_id}

Vypsání skupin, kterých je uživatel členem:
GET http://localhost:8080/rest/{user_id}/groups

Přidání vzpomínky:
POST http://localhost:8080/rest/{user_id}/memory

```
{
"name": "New Memory",
"description": "Me beautiful new memory.",
"postCreator": {
        "id": 1
        }
}
```

Přidání kapsle:
POST http://localhost:8080/rest/{user_id}/capsule

```
{
"name": "New Capsule",
"description": "Me beautiful new capsule.",
"timeOfOpening": "2025-01-06"T15:22:56.43",
"postCreator": {
        "id": 1
        }
}
```


Upravení vzpomínky (kapsle se nedá upravovat):
PUT http://localhost:8080/rest/{user_id}/memory

```
{
"name": "New Memory",
"description": "Updated description.",
"postCreator": {
        "id": 1
        }
}
```


Vymazání vzpomínky/kapsle (obojí stejná metoda):
DELETEhttp://localhost:8080/rest/{post_id}/delete

Testovací scénáře

Všechny následující scénáře vužívají registraci a přihlášení uživatelů, proto pokud následující scénáře byly úspěšně provedeny, pak registraci i přihlačování uživatelů zcela jistě funguje.

Ověření funkčnosti vytváření skupiny:

Registrace uživatele U1 – přihlášení U1– změna role na PREMIUM – opětovné přihlášení U1– registrace dalšího uživatele U2 – vytvoření skupiny SA – vytvoření další skupiny SB – přidání uživatele U1 do skupiny SA – přidání uživatele U1 do skupiny SB – přidání uživatele U2 do skupiny SA.

Uživatel U1 při vypsání skupiny uvidí skupinu SA, jejíž členem je on a uživatel U2 a skupinu SB, jejíž členem je pouze on sám. Uživatel U2 uvidí skupinu SA, jejíž členem je on a uživatel U1.

Ověření funkčnosti mazání skupin:

Registrace uživatele U1 – přihlášení U1– změna role na PREMIUM – opětovné přihlášení U1– vytvoření skupiny SA – registrace dalšího uživatele U2 – přihlášení U2– změna role na PREMIUM – opětovné přihlášení U2 – vytvoření skupiny SB.

Uživatel U2 smí smazat skupinu SB, ale nemůže smazat skupinu SA. Uživatel U1 smí smazat skupinu SA, ale nemůže smazat skupinu SB.

Ověření vytváření vzpomínek:

Registrace uživatele U1 – přihlášení U1– přidání vzpomínky V1 – přidání vzpomínky V2.

Dotaz na vypsání vzpomínek vypíše vzpomínku V1 a V2.

Ověření mazání vzpomínek:

Registrace uživatele U1 – přihlášení U1– přidání vzpomínky V1 – přidání vzpomínky V2 – smazání vzpomínky V2.

Dotaz na vypsání vzpomínek vypíše pouze vzpomínku V1.

Nabité zkušenosti

Daria Kuznetsova: v rámci tohoto projektu jsem se naučila pracovat s frameworkem Spring Boot. Práce s Spring Bootem mi přišla trochu složitá, zejména v části týkající se controllerů a autentifikace. Na druhou stranu, práce s databází a logikou byla relativně jednoduchá. Tento projekt mi poskytl cenné zkušenosti, které mi pomohly lépe pochopit strukturu a fungování webových aplikací postavených na Spring Bootu.

Matěj Charousek: v rámci této semestrální práce jsem vytvořil aplikaci pomocí Javy. V minulosti jsem vytvářel webovou aplikaci pomocí jazyka PHP. Vytváření Java aplikace mi přišlo přirozenější a mnohem strukturovanější. Tím, že každá třída musí být v Javě jako jeden soubor mě to tlačilo k tomu mít jasnou strukturu. PHP mě k tomuto tolik nevedlo a přišlo mi, že jsem se občas ztrácel v tom, kde vlastně v rámci své práce jsem, což se mi s Javou nestalo. Také jsem ocenil jednoduché napojení na databázi skrze persistentní vrstvu a jednoduché reagování na http dotazy díky rest controllerům. Obecně mám rád OOP a Javu, takže i díky tomu mě práce na semestrální práci velmi bavila a přijde mi, že jsem se více naučil Javu, například jsem začal používat JavaStreamy, které mi často usnadnily práci.