Java alapú webes keretrendszerek
Házi feladat Dokumentáció:

InvoiceManager

(Számlakezelő app)

Biró Bence
SM5P0J
Dátum: 2023.05.16

Feladat:

A webalkalmazás feladata, hogy felhasználók jogaiktól függően képesek legyenek a számláikat megnézni, módosítani, törölni és létrehozni. Valamint a felhasználókat is lehet benne törölni és jogaikat módosítani, valamint lehet regisztrálni a rendszerbe.

Környezet:

IntelliJ IDE fejlesztői környezetben, Windows 10 operációs rendszeren, Java 17-es verzióval, Maven 4.0.0 és Spring Boot 3.0.2 használva.

Forráskód struktúrája:

A Maven könyvtárstruktúráját használom. Az alkalmazás 3 rétegre van bontva. A rétegek a következőek : Repository, Service, Controller. Ezek mindegyike, csak az alatta lévőre hivatkozik, tehát Controller csak a service-re(megeshet, hogy a Repositroy-ra), Service csak a Repository-ra, Repository pedig az adatbázissal kommunikál.

- /pom.xml – projekt build fájl
- /src/main/java – java forráskódok
    - Fő package név : com.exmaple.invoicemanager
    - Alkönyvtárak:
        - DTO – Egyes entitásokhoz tartozó Data Transfer Object-ek
        - /libs.Error – Egyes Exception-ök találhatóak itt
        - Model – Entitások definiciói
        - Repository – Entitásokhoz tartozó JpaRepository-k
        - Security – Authentikációhoz és Authorizációhoz tartozó configurációk és modellek
        - Service – Entitásokhoz tartozó üzleti logikák
        - Thymeleaf – Thymeleaf-hez tartozó configuráció
        - Web – Oldalak és jogok alapján meghatározott Kontrollerek és handler-ek
- /src/main/resources – nem forditandó de csomagolandó fájlok
    - /static/js – templétekhez tartozó .js fájlok
    - /static/styles – temlétekhez tartozó .css fáljok
    - /templates/layout – oldalakhoz tartozó közös sablon
    - /templates – oldalak html fálja
    - application.ymal – egyes statikus értékek beállítása
    - data.sql – sql kód az adatbázis default feltöltéséhez

Maven függőségek:

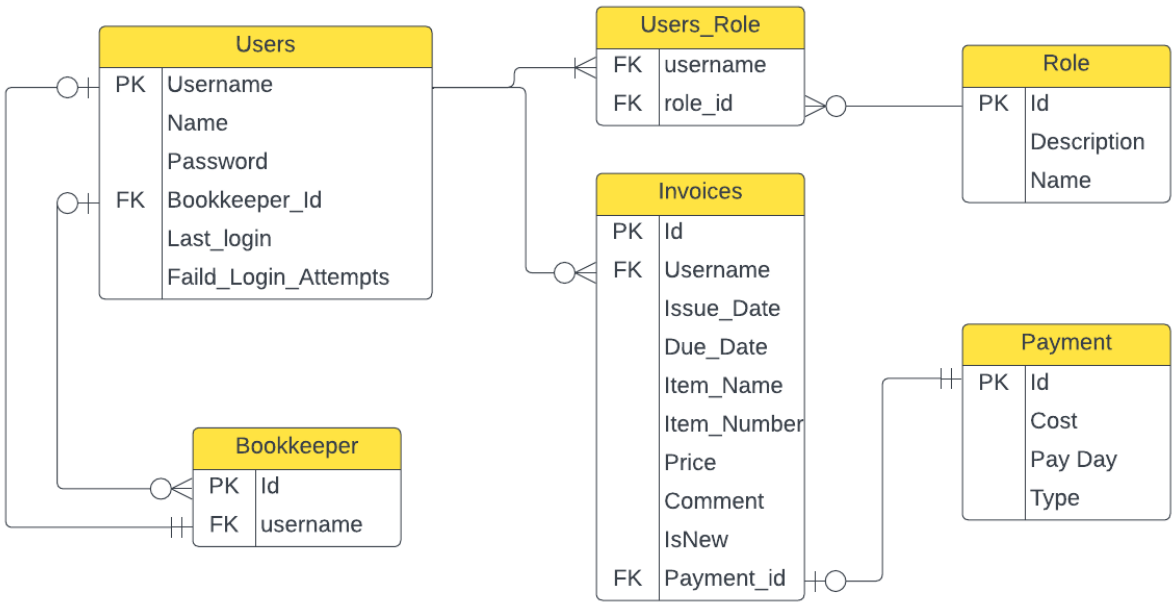
A forráskódban a pom.xml fájlban találhatóak a függőségek, a „dependencies” tag alatt ebben a sorrendben. Ezek további implicit függőségeket hozhatnak magukkal. Fejlesztés kezdetén legfrissebb verziókat használtam.

Spring Boot-hoz kapcsolódó függőségek verziói megegyeznek a Spring Boot verziójával(3.0.2).

- Spring Boot JPA – 3.0.2
- PostgreSQL – 42.5.1
- Spring Boot Web – 3.0.2
- Lombok – 1.18.24
- Spring Boot Security – 3.0.2
- Spring Boot Thymeleaf – 3.0.2
- Thymeleaf SpringSecurity6 – 3.1.1.RELEASE
- Thymeleaf Layout Dialect – 2.4.1
- Spring Boot Test – 3.0.2
- Spring Security Test – 6.0.1

Adatbázis felépítése:

PostgreSQL adatbázist használtam. Adatbázis neve „invoice”. Az URL-je a application.yml-ben található.



Tanulság:

- Primary Key csak nagyon kivételes esetben ne Id legyen, kód olvashatóság és egyszerűség szempontjából
- Figyelni bidirectional kapcsolatokra az adatbázisban, ne legyen redundáns megvalósítás az adatbázisban

Entitások

User:

- username azonosítja
- eltárolja a felhasználó adatait
- rámutat a könyvelőre, aki kezeli a felhasználó számláit
- szerepköröket egy set-ben tárolja
- eltárolja mikor jelentkezett be utoljára
- eltárolja, hány sikertelen bejelentkezési próbálkozása volt a felhasználónak
    - nem működik
    - plus funkcióhoz segítene(captcha)

Role:

- az egyes szerepkörök is adatbázisban vannak tárolva
- 3 szerep van: USER, BOOK, ADMIN
- szerepek képességei:
    - USER
        - megtudja nézni a számláit listában és részletesen is és tudja törölni is őket
    - BOOK
        - saját és klienseinek a számláit megtudja nézni, módosítani és törölni, valamint új számlát is tud létrehozni
    - ADMIN
        - megtudja nézni a saját számláit
        - felhasználókat tud törölni
        - módosíthatja a felhasználók szerepeit

Invoice:

- eltárolja a számlák adatait
- rámutat a felhasználóra, akié a számla
- a státusz is elmenti, hogy volt-e már részletesen megnyitva

Payment:

- számlához tartozó fizetési módot tárolja el
- a fizetés idejét és teljes költségét

Bookkeeper:

- BOOK szereppel rendelkező felhasználók vannak benne, saját könyvelő Id-val
- valamit elérhetőek a könyvelőhöz tartozó klienseket

Az Invoice mellett még van 3 DTO. Az InvoiceCreateDTO ebben a számla létrehozásához szükséges adatok vannak. Az InvoiceUpdateDTO, ami a számla frissítéséhez szükséges adatokat

tárolja. InvoiceDTO-t pedig a View-nak van küldve és ezt jeleníti meg, így megelőzve a rekurzív hivatkozásokat.

User-nek 2 DTO-ja van. A UseDTO a megjelenítéskor van használva, a UserCreateDTO a felhasználó regisztációjakor.

A Payment-nek csak egy van, amit az InvoiceDTO-ban használok, tehát ez is csak megjelenés miatt. Számla frissítéskor az InvoiceUpdateDTO-ban csak a payment_id kerül tárolásra.

Repository-k:

Minden entitáshoz tartozik egy Repository, amely megvalósítja a JpaRepository és ezen keresztül tudunk kommunikálni az adatbázissal.

Service-k:

Service-k publikus és bonyolultabb logikát megvalósító függvényei JavaDoc-al vanna lelátva.

UserService – Felhasználóval kapcsolatos műveleteket végzi, mentés, törlés, lekérdezés és a szerepkör változás lekezelése(dinamikus szerepkör váltás).

RoleService – Szerepkörök lekérése és konvertálás a feladata.

InvoiceService – A számlákkal kapcsolatos műveletek elvégzése. Mentés, törlés, frissités és lekérdezés.

PayService – A számlához tartozó fizetési adatokat lehet létrehozni. A fizetés létrehozás után nem lehet módosítani a számlát.

BookkeeperService – A könyvelőhöz tartozó adatok lekérése valamint a könyvelő személyének vizsgálata.

Controller-ek:

A controller-ek két csoportra szedhetőek. Az első csoportba azok tartoznak, amik konkrét oldalnak a funkcióit valósítják meg. Ezek közé tartozik a többség: Admin, Auth(Ez a login és a registration oldalt), Home, Invoice, List. A másik csoport, ami funkciókat valósít meg pl.: MyError és Pay. A Pay az List oldalon van használva.

Hibakezelés:

A libs.Error mappában vannak a saját kivételeim.

Ezeket a Web mappában található MyErrorController kezeli le, mely az ErrorController-t valósítja meg. Hiba esetén a saját hibajelző oldalamra ugrik a és kiírja , hogy milyen hibába ütközött és innét visszalehet navigálni a home oldalra. Pl:

Security:

A felhasználók authentikációt és authirizációját Spring Security valósítja meg. Első lépésként a User entitásokat MyUserDetails-be csomagolom, ami megvalósítja a UserDetails interfészt és ezzel a SecurityContext Authentication-ben feltudom használni.

A MyLoginSuccessHandler elmenti, hogy a felhasználó mikor lépett be utoljára. Ezt a SimpleUrlAuthenticationnSuccesHandler megvalósításával érem el.

A SecurityConfig-ban vannak a szükséges konfigurációs adatok.