Systém pre správu lekárne

Matej Pavlík

Zámer projektu

Používateľ sa môže prihlásiť do systému pomocou užívateľského mena a hesla. Systém overí, či sú údaje správne a ak áno tak užívateľa prihlási. Podľa toho, či používateľ je manažér alebo lekárnik má v systéme prístup k rôznym častiam.

Manažér môže nakúpiť lieky od dodávateľov. Každý liek obsahuje číslo šarže, dátum výroby, dátum expirácie, meno výrobcu, atď. Môže pridávať, odoberať zamestnancov a upravovať informácie o nich. Má možnosť vygenerovať výplatné listiny. Manažérovi sú k dispozícií rôzne reporty (napr. mesačný predaj liekov a podobne).

Manažér a lekárnik si môžu pozrieť stav skladu. Sklad sa aktualizuje nákupom a predajom liekov.

Lekárnik vydáva lieky. Pri predaji liekov sa vyplnia alebo načítajú (napr. z preukazu poistenca) údaje ako zákazníkové meno, identifikačné číslo, adresa, detail lieku, meno doktora. Systém si tieto informácie o zákazníkoch ukladá.

Lekárnikovi je umožnená organizácia liekov. Môže vytvárať sekcie (napr. podľa toho, na čo sa liek používa, atď.) a pridávať do nich jednotlivé lieky. Lekárnik (predavač) tak potom môže lieky jednoducho nájsť. Systém mu vypíše kde presne na sklade sa daný liek nachádza.

Lekárnik si môže v systéme vyhľadať detailné informácie o liekoch, ktoré sú v lekárni k dispozícií.

Hlavné kritéria hodnotenia

Dedenie

Projekt využíva dvojnásobné dedenie. Všetky admin controllers v balíku Controller.Admin dedia od abstraktnej triedy AbstractAdminController, ktorá dedí od triedy Controller.

Použitie rozhraní

V projekte sa používa rozhranie Database, ktoré implementujú všetky databázy v balíku App.Db. Toto rozhranie sa používa prevažne na serializáciu a deserializáciu. Môžeme to vidieť aj v triede Main (balík App) v metóde stop().

Polymorfizmus

V modeli LoginModel (balík Model) sa po úspešnom zadaní emailu a hesla zavolá this.loginService.getCurrentUser().login(). Metóda loginService.getCurrentUser() vracia objekt typu ManagerUser alebo PharmacistUser (ktoré dedia od triedy User) a každá z týchto tried má inú implementáciu login().

Zapuzdrenie

Takmer každá trieda používa gettery, cez ktoré možno získať údaje v privátnych alebo protected poliach.

Agregácia

Agregácia je využitá napríklad v triede OrderItem (balík App.Order). OrderItem má liek (triedu Drug).

Oddelenie používateľského rozhrania od aplikačnej logiky

Každé view je riešené pomocou fxml. Pre každé view existuje controller a model. Niektoré controllers využívajú services a databázy.

Ďalšie kritéria hodnotenia

Generické triedy

Abstraktná generická trieda Controller (balík Controller) slúži na inicializáciu modelov typu T. Využívajú ju všetky controllery.

Vlastné výnimky

V balíku App.Validation sa nachádzajú vlastné výnimky, ktoré sú vyhadzované pri validácií formulárov. Napríklad výnimka UserManagementError je vyhadzovaná v UserManagementService (App.Services)

Oddelenie používateľského rozhrania od aplikačnej logiky (handlers)

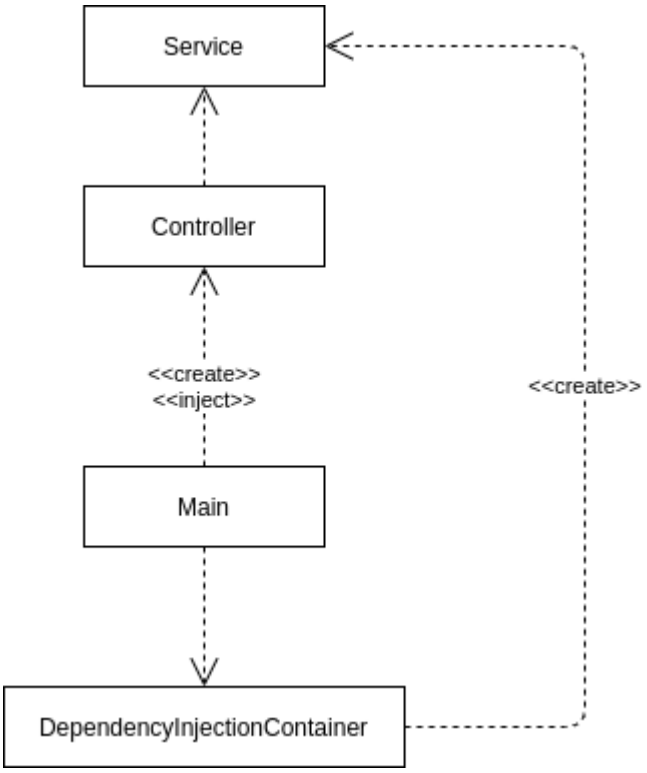
Nie vždy vytvára spracovateľov udalostí fxml. Napríklad v AdminBuyMedicineController sú vytvorené udalosti po výbere užívateľa nejak položky v zozname.

Návrhový vzor Dependency Injection

Controller má reagovať na to čo robí používateľ s View. Nemal by byť zodpovedný za napríklad za inštancovanie Services. Navyše, ak by to robilo viacero controllerov a Service by sme chceli zmeniť za iný alebo ho vymazať, museli by sme to repetetívne urobiť vo všetkých controlleroch. Takisto by sme mali v každom controlleri inú inštanciu Service, čo môže byť problém.

Pomocou návrhového vzor Dependency Injection vieme vytvoriť dependencies pre controller mimo controllera a stačí nám to urobiť len raz. Dependencies vytvára DependencyInjectionContainer a tieto dependecies sa injectujú do controllerov v triede Main (balík App).

UML diagram:



RTTI

V interfaci Database (App.Db) používam getClass().getSimpleName() na zistenie mena triedy.

Vhniezdené triedy

Trieda Comparators (balík Util) je vhniezdená. Táto trieda sa používa v databázach OrderDatbase a WarehouseDatabase.

Lambda výrazy

V FXMLLoaderCreator (Util) je použitý lambda výraz

Default method implementation

V interfaci Database (App.Db) sú použité defaultné metódy

Serializácia

Serializácia sa používa vo všetkých databázach v balíku App.Db. Na serializáciu databázy používajú jeden interface Database. Databázy sú serializované pri ukončení programu v triede Main.