****

**Документација за семинарска работа на тема *WorkoutApplication***

**по предемтот Веб програмирање**

***Ментор:***

***проф. Ана Тодоровска***

*Изработила:*

*Ана Марјановиќ - 211024*

*Скопје, 2024*

**Документација за семинарска работа на тема WorkoutApplication**

**по предемтот Веб програмирање**

Како голем ентузијаст и љубител на вежбањето, секојдневно поминувајќи го времето во програмирање успеав да ги обединам овие две нешта во оваа семинарска работа создавајќи Веб апликација на тема **WorkoutApplication**. Со целото знаење кое го стекнав во изминатава (трета) година од моите додипломски студии, по предметот Веб програмирање, следејќи ги предавањата и вежбите, направив апликација за вежбање која ќе ми помогне мене,но и на сите оние кои имаат пасија кон вежбањето.

Почнувајќи од основната идеја за тоа како ќе изгледа апликацијата, па се до функционалностите, моделите,креирањето на postgress базата на податоци, формата за логирање и регистрирање, HTML и CSS темплејтите, ја добив крајната верзија на апликацијата за чии што детали ќе објаснам подолу.

**1. Model**

Целиот процес на изработка на оваа апликација го започнав со креирање на моделите, поточно на **Exercise** и **User** моделот. Exercise моделот е составен од атрибутите id (Long), name, description, videoURL, category, imageUrl (String) при што како примарен клуч го поставив атрибут id со додавање на анотацијата @Id по кој би се препознавала секоја вежба засебно. Додатно за овој модел ги креирав потребните конструктори, како и set и get методи кои ќе ни бидат потребни во понатамошните функции.



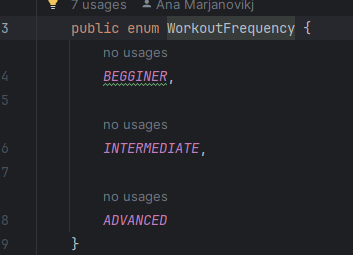
***Слика 1. Приказ на моделот Exercise***

По креирањето на моделот Exercise, го креирав моделот User, кој ги содржи следниве атрибути: id (Long), username (String), password (String), height (Integer), weight (Integer), при што повторно како примарен клуч кој ќе служи за идентификување на корисниците го поставив атрибутот id. Дополнително за овој модел покрај анотацијата @Id, ја употребив и анотацијата @Column со која направив ограничување дека атрибутите кои корисникот ќе ги внесува за себе не смеат да имаат null вредност и мораат да имаат должина помала од онаа која произволно сум ја внела. За двата модела кои се користат во оваа апликација употребив заедничка анотација @Entity и @Table со кои се поврзуваме со дата базата и креираме посебна релација (табела) за секој модел посебно, соодветно.



***Слика 2. Приказ на моделот User***

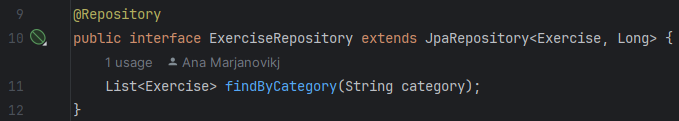
Покрај Exercise и User моделот, креирав и енумерација **(WorkoutFrequency)** со која корисникот при регистрација може да го избере нивот на спремност за вежбање така што поставив таа да може да прими некоја од следниве три вредности: BEGINNER, INTERMIDIATE,ADVANCED. Оваа енумерација се поврзува со моделот User бидејќи при регистрацијата самиот корисник ќе одбере на кое ниво смета дека се наоѓа.



***Слика 3. Приказ на енумерација***

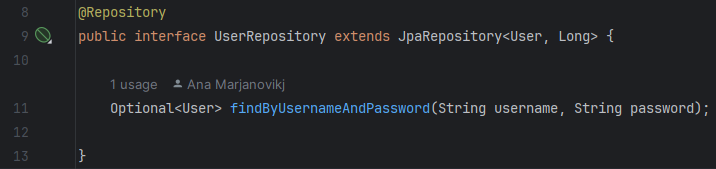
**2. Repository**

Откако ги креирав моделите, за секој од нив соодветно создадов репозитори, односно ги креирав **ExerciseRepository** и **UserRepository**. Овие два типа на репозитори наследуваат од класата JPARepository на SpringBoot која содржи соодветни методи.



***Слика 4. Приказ на ExerciseRepository***

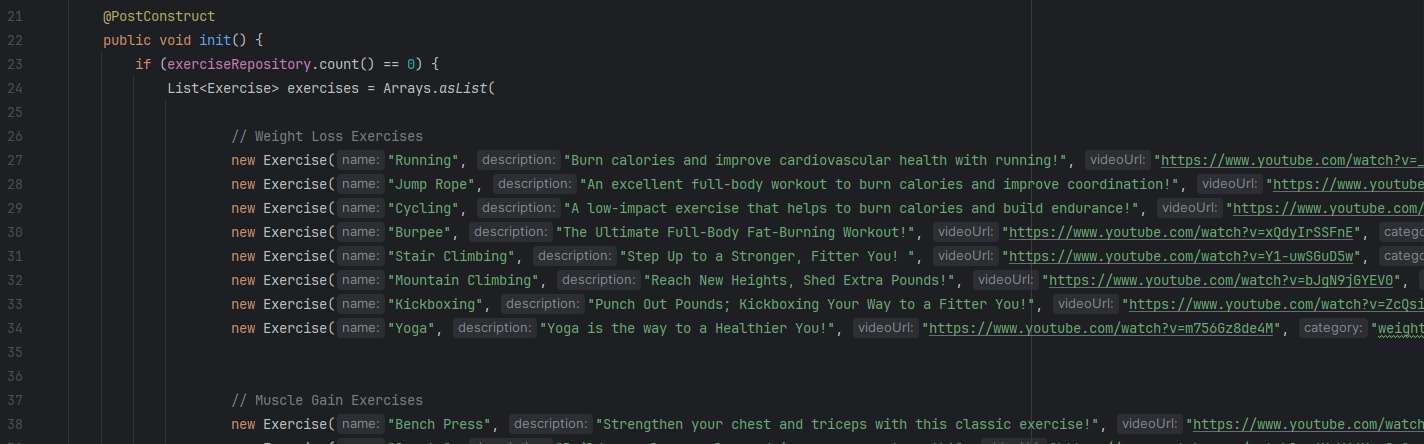
Дополнително во ExerciseRepository креирав додатен метод findByCategory кој ќе ги пребарува вежбите по категорија, додека кај UserRepository креирав додатен метод findByUsernameAndPassword кој ќе ги пребарува корисниците по корисничко име и лозинка.

******

***Слика 5. Приказ на UserRepository***

**3. DataInitializer**

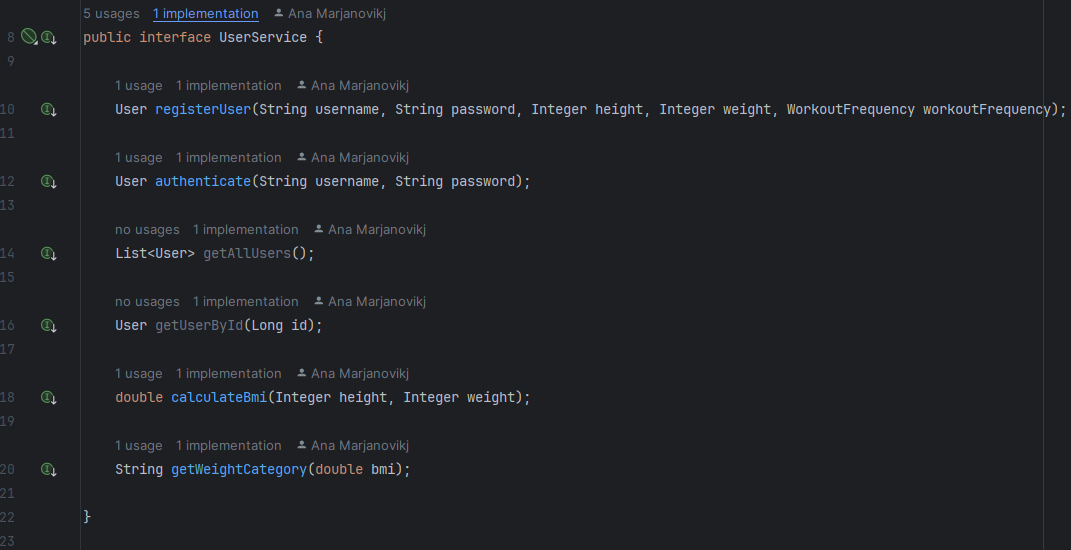
Откако ги креирав репозиторито за двата модела, ја создадов јава класата DataInitializer.Таа е класа која се наоѓа во config пакетот и има за задача да ја пополни дата базата со иницијалните податоци за вежбите. Оваа класа користи @PostConstruct анотација која ја извршува init() функцијата (методот) која воедно проверува дали ExerciseRepository e празно. Ако е празно тогаш прави листа која ја пополнува со предефинирана листа од вежби кои се категоризирани во една од 4-те секции: weightloss, musclegain, body posture and health. Воедно за секоја вежба се чуваат името на вежбата, дескрипција за вежбата, линк до YouTube видео за демонстрација на вежбата, категорија во која спаѓа и image file name.



***Слика 6. Приказ на јава класата DataInitializer***

**4. Service (ServiceImpl)**

По јава класата DataInitializer, го креирав UserService во кој создадов интерфејси од методите кои ги обработувам во ServiceImpl. Тие методи се: **registerUser**, **authenticate**, **calculateBmi** и **getWeightCategory**.

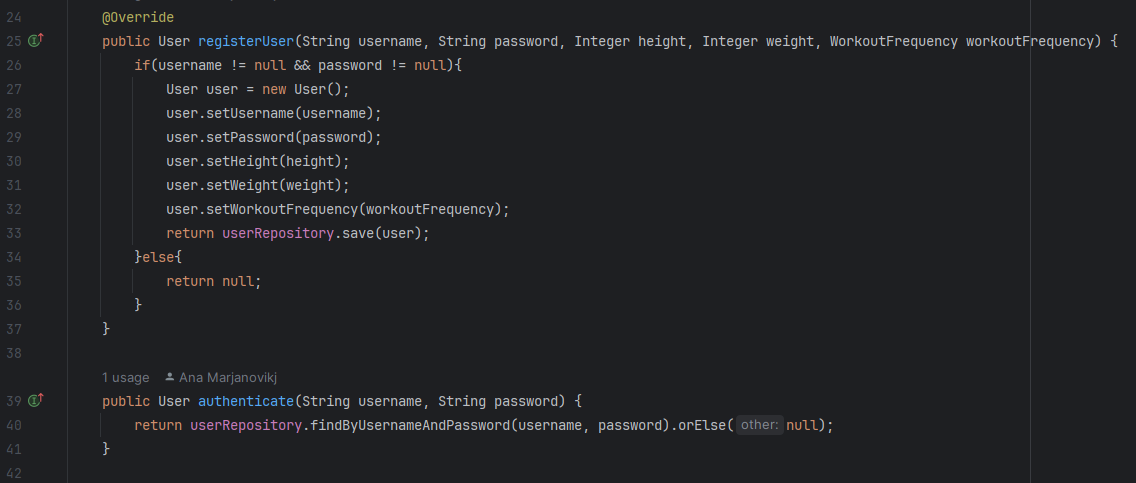


***Слика 7. Приказ на UserService***

По UserService, го креирав UserServiceImpl кој е составен од методите:

- **registerUser** : тоа е метод кој проверува дали веќе сме ги внеле корисничко име и лозинка што корисникот моментално ги внесува во формата за регистрација, па доколку се уникатни го креира новиот корисник и го зачувува во дата базата.

- **authenticate** : тоа е метод кој ја користи функцијата findByUsernameAndPassword преку userRepository при што пробува да го препознае дали постои корисник со тоа корисничко име и лозинка.



***Слика 8. Приказ на методите од UserServiceImpl (registerUser и authenticate)***

- **calculateBmi** : тоа е метод кој со однапред дадени вредности за висина и тежина го пресметува BMI на корисникот.

- **getWeightCategory** : тоа е метод кој врз база на пресемтаниот BMI го класифицира корисникот во соодветната тежинска група.



***Слика 9.*** ***Приказ на методите од UserServiceImpl (calculateBMI и getWightCategory)***

**5. Controller**

По креирањето на сите претходни јава класи, ја креирав класата контролер составена од следните методи:

- **showHomePage:** тоа е **GET** метод кој ја прикажува почетната страница на веб апликацијата при што не пренасочува до *homepage.html.*

*-* **showAboutPage**:тоа е **GET** метод кој ја прикажува страната About Us на веб апликацијата при што не пренасочува до *aboutpage.html.*

- **showRegisterForm**: тоа е **GET** метод кој ја прикажува формата за регистрација на веб апликацијата при што не пренасочува до *registerpage.html.*

**

***Слика 10. Приказ на методите showHomePage, showAboutPage, showRegisterForm***

*-* **processRegistrationForm**: тоа е **POST** метод кој проверува преку if услов дали корисникот направил грешка при пополнување на формата за регистрација, при што доколку направил го враќа повторно на страната каде што се наоѓа формата, а доколку се регистрирал коректно ја проверува уникатноста на корисничкото име и лозинка со if услов, при што доколку двата податока се прв пат внесени во базата преку метод од userService се регистрира новиот корисник.

- **showLoginForm**: тоа е **GET** метод кој ја прикажува формата за најавување на корисникот при што оваа функција не пренасочува кон *loginpage.html.*

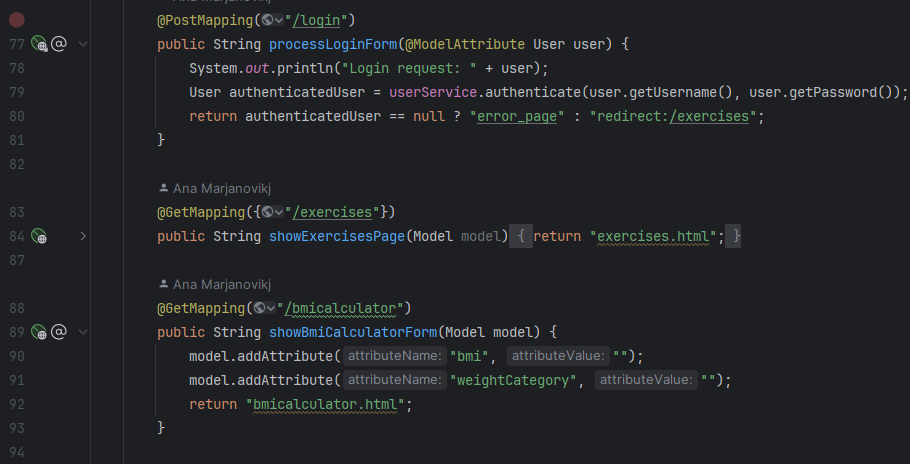


***Слика 11. Приказ на методите processRegistrationForm и showLoginForm***

*-* **processLoginForm**: тоа е **POST** метод кој преку userService ја користи функцијата authenticate при што пробува да провери во базата на податоци дали постои веќе регистриран корисник со тоа корисничко име и лозинка, па доколку постои го најавува корисникот.

- **showExercisesPage**: тоа е **GET** метод кој ја прикажува страната со категории и при тоа не пренасочува до *exercises.html* во која се креираат четири картички различни по категорија.

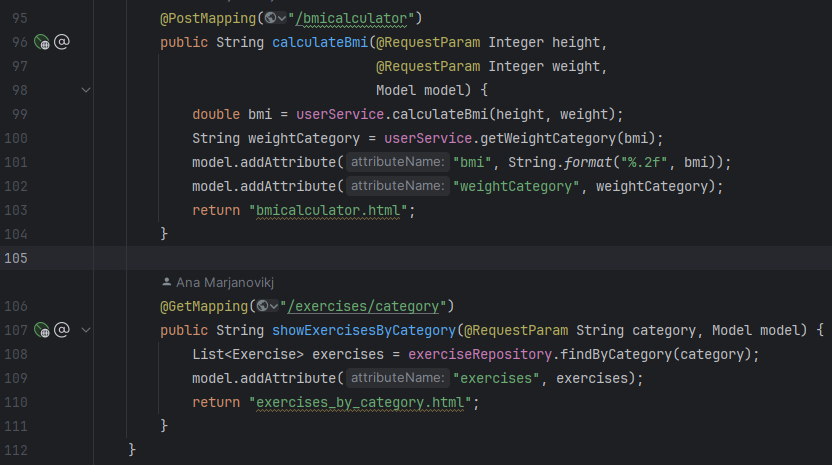
- **showExercisesByCategory** : тоа е **GET** метод кој креира листа за секоја категорија при што со помош на ExerciseRepository, односно методот findByCategory ги класифицира вежбите по категории според вредноста на атрибут category. Дополнително не пренасочува до страната exercises\_by\_category.html каде со помош на thymeleaf покажува соодветни картички за вежбите како и нивните атрибути: image, exercise name, description…



***Слика 12. Приказ на методите processLoginForm, showExercisePage,showBMICalculator***

- **showBmiCalculatorForm**: тоа е **GET** метод кој ја прикажува странат со BMI калкулаторот и не пренасочува до страната *bmicalculator.html.*

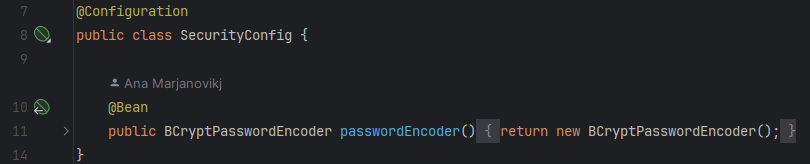
*-* **calculateBmi**: тоа е **POST** метод кој ни овозможува кога ќе кликнеме на копчето Calculate BMI да ни ја прикаже тежинската група на корисникот како и неговиот BMI.



***Слика 13. Приказ на методите calculateBMI и showExercisesByCategory***

**6. SecurityConfig**

На самиот крај ја креирав SecurityConfig класата која служи за безбедносно нагодување на апликацијата. Во оваа класа го повикав јава методот **BCryptPasswordEncoder** кој дозволува енкодирање на лозинките на корисниците во дата базата.



***Слика 14. Приказ на SecurityConfig***