Iliescu Bianca-Ana-Maria
332AA

Aplicație pentru gestionarea subscripțiilor lunare

Cuprins

Iliescu Bianca-Ana-Maria
332AA

1. Specificare funcționalități de bază și suplimentare

1.1. Descrierea generală a aplicației

Aplicația permite utilizatorilor să își creeze un cont și să se autentifice, mai apoi având posibilitatea să adauge subscripțiile lor lunare prin specificarea numelui, prețului, categoriei și a datei de început. În momentul în care un utilizator are o subscripție adăugată în cont, acesta poate vizualiza numărul de zile până la următoarea plată și statistici privind costul total lunar și distribuția subscripțiilor în funcție de categorii. De asemenea, un utilizator poate modifica prețul sau data subscripției și o poate șterge dacă își dorește eliminarea acesteia din listă.

1.2. Funcționalități de bază

- Vizualizarea subscripțiilor lunare
- Adăugarea unei subscripții prin specificarea numelui, prețului, categoriei și a datei de început. Validările pe care le-am adăugat în această secțiune sunt cu privire la preț, care nu poate fi un număr negativ, și data de început a subscripției, care nu poate fi înainte de data curentă.
- Posibilitatea de a modifica prețul sau data de început a unei subscripții, cu aceleași validări ca la adăugare.
- Posibilitatea de a șterge o subscripție. Această acțiune trebuie să fie confirmată prin bifarea unei căsuțe.

1.3. Funcționalități suplimentare

- Înregistrarea unui utilizator pe platformă, prin completarea următoarelor câmpuri: username, email, parolă. Validările pe care le-am adăugat în această secțiune sunt cu privire la parolă, care trebuie să

conțină minim o majusculă, 3 cifre și un caracter special, și email, care trebuie să conțină caracterul „@” și o structură de forma „.com”.

- Autentificarea utilizatorilor în conturile proprii
- Vizualizarea statisticilor cu privire la costul total lunar al subscripțiilor, cât și distribuția subscripțiilor pe categorii în funcție de preț
- Filtrarea subscripțiilor în funcție de categorie
- Sortarea subscripțiilor ascendent sau descendent în funcție de preț și de data de început

2. Descrierea claselor/metodelor/atributelor proiectului

2.1. Subscripție

- Mapare ORM: Clasa utilizează adnotări JPA pentru a mapa obiectul Java la o entitate de bază de date.
- Gestionarea Relațiilor: Implementează o relație Many-to-One cu entitatea User, permițând asocierea mai multor abonamente cu un singur utilizator.
- Manipularea Datelor: Oferă metode pentru setarea și obținerea datelor abonamentului, inclusiv o metodă specială getUserId() pentru a obține ID-ul utilizatorului asociat.
- Formatare Date: Utilizează adnotarea @DateTimeFormat pentru a asigura formatarea corectă a datei de început.
- Calculul Următoarei Plăți: Metoda getZilePanaLaUrmatoareaPlata() calculează numărul de zile până la următoarea plată, bazându-se pe data de început și data curentă.

2.2. User

- Mapare ORM: Utilizează adnotări JPA pentru a mapa obiectul Java la o entitate de bază de date.
- Gestionarea Relațiilor: Implementează o relație One-to-Many cu entitatea Subscriptie, permițând unui utilizator să aibă multiple abonamente.
- Securitate: Folosește adnotarea @JsonIgnore pentru a preveni serializarea listei de abonamente, protejând astfel datele sensibile.
- Acces la Date: Oferă metode pentru obținerea și setarea informațiilor utilizatorului:

  - getID(): Returnează ID-ul utilizatorului

- getEmail(): Returnează adresa de email
- getNume(): Returnează numele de utilizator
- setID(int id): Setează ID-ul utilizatorului
- setNume(String name): Setează numele de utilizator

- Validare la Nivel de Bază de Date: Utilizează constrângeri de nulitate și lungime pentru coloanele din baza de date, asigurând integritatea datelor.

2.3. Repository Subscripție

- Operații CRUD de Bază: Prin extinderea JpaRepository<Subscriptie, Integer>, interfața moștenește metode standard pentru operații Create, Read, Update și Delete pe entitatea Subscriptie, utilizând Integer ca tip pentru cheia primară.
- Căutare după ID-ul Utilizatorului:

- List<Subscriptie> findByUser_UserID(Integer userID): Returnează toate abonamentele asociate unui utilizator specific.
- List<Subscriptie> findByUser_UserID(Integer userID, Sort sort): Similar cu metoda anterioară, dar permite sortarea rezultatelor.

- Căutare Specifică Abonament-Utilizator:

  - Subscriptie findBySubscriptionIDAndUserUserID(Integer subscriptionID, Integer userID): Găsește un abonament specific pentru un utilizator dat, asigurând astfel că utilizatorul are acces doar la propriile abonamente.

- Filtrare după Categorie și Utilizator:

  - List<Subscriptie> findByUser_UserIDAndCategory(Integer userID, String category): Returnează abonamentele unui utilizator filtrate după o anumită categorie.

2.4. RepositoryUser

- Operații CRUD Standard: Prin extinderea JpaRepository<User, Integer>, interfața moștenește automat metode pentru operațiile de bază Create, Read, Update și Delete pe entitatea User, folosind Integer ca tip pentru cheia primară.
- Căutare după Email:

- User findByEmail(String email): Această metodă permite găsirea unui utilizator specific bazat pe adresa sa de email. Este utilă pentru operații precum autentificarea sau recuperarea parolei.

- Verificare Existență Email:
- boolean existsByEmail(String email): Oferă o metodă rapidă pentru a verifica dacă un email există deja în baza de date. Aceasta este crucială pentru validarea înregistrării noilor utilizatori și evitarea duplicatelor.

2.5. SubscriptionServiceImpl

- Adăugare Abonament: adaugaSubscriptie(Subscriptie subscriptie): Salvează un nou abonament în baza de date.

- Ștergere Abonament: stergeSubscriptie(Integer subscriptionID, Integer userID): Șterge un abonament specific, verificând mai întâi dacă aparține utilizatorului corect.

- Obținere Abonament: getSubscriptie(Integer subscriptionID, Integer userID): Recuperează un abonament specific pentru un utilizator dat.

- Modificare Abonament: modificaSubscriptie(Integer subscriptionID, Integer userID, float price, LocalDate startDate): Actualizează prețul și data de început a unui abonament existent.

- Listare Abonamente: getSubscriptiiUser(Integer userID): Returnează toate abonamentele unui utilizator.
- getSubscriptiiUserDupaCategorie(Integer userID, String category): Filtrează abonamentele unui utilizator după categorie.
- getSubscriptiiUserSortate(Integer userID, Sort sort): Returnează abonamentele unui utilizator sortate conform criteriilor specificate.

2.6. UserServiceImpl

- Înregistrare Utilizator: inregistrareUser(User user): Înregistrează un nou utilizator în sistem, verificând mai întâi dacă emailul există deja pentru a evita duplicatele.

- Autentificare Utilizator: autentificare(String email, String password): Realizează autentificarea utilizatorului pe baza emailului și parolei. Compară direct parolele pentru utilizatorii existenți în baza de date.

- Ștergere Cont: stergeCont(Integer userID): Șterge contul unui utilizator din sistem pe baza ID-ului său.

- Obținere Utilizator:

  - getUserDupaID(Integer id): Recuperează informațiile unui utilizator specific pe baza ID-ului său.

- Listare Utilizatori:

  - getTotiUserii(): Returnează o listă cu toți utilizatorii înregistrați în sistem.

2.7. ServiceStatisticiSubscriptii

- Calculul Costului Lunar Total (calculeazaPretulLunar):

  - Primește ID-ul unui utilizator ca parametru.
  - Recuperează toate abonamentele asociate utilizatorului.
  - Calculează suma totală a prețurilor tuturor abonamentelor.
  - Returnează costul lunar total ca valoare de tip double.

- Calculul Procentajelor pe Categorii (calculeazaProcentajulPeCategorii):

  - Primește ID-ul unui utilizator ca parametru.
  - Recuperează toate abonamentele utilizatorului.
  - Calculează costul total lunar folosind metoda calculeazaPretulLunar.
  - Grupează abonamentele pe categorii și calculează suma pentru fiecare categorie.
  - Calculează procentajul fiecărei categorii din costul total.
  - Returnează un Map cu numele categoriilor ca chei și procentajele ca valori.

2.8. ControllerAutentificare

- Autentificare Utilizator:
  - Metoda autentificare gestionează cererile POST la /web/auth/autentificare.

  - Verifică credențialele utilizatorului folosind UserService

  - În caz de succes, stochează utilizatorul în sesiune.
  - Redirecționează către pagina de profil.

- Înregistrare Utilizator:
  - Metoda inregistrare procesează cererile POST la /web/auth/inregistrare.
  - Validează formatul emailului și complexitatea parolei.
  - Creează și înregistrează un nou utilizator.
  - Oferă feedback prin mesaje flash.
- Deconectare:
  - Metoda deconectare gestionează cererile POST la /web/auth/deconectare.
  - Invalidează sesiunea curentă.
  - Confirmă deconectarea prin mesaj flash.

2.9. ControllerSubscriptii

- Adăugare Abonament:
  - Metoda adaugareSubscriptie gestionează cererile POST pentru crearea unui nou abonament.
  - Validează datele de intrare (nume, categorie, preț, dată de început).
  - Creează și salvează un nou obiect Subscriptie.
- Ștergere Abonament:
  - Metoda stergereSubscriptie procesează cererile GET pentru ștergerea unui abonament specific.
  - Utilizează SubscriptionService pentru a efectua ștergerea.
- Listare Abonamente:
  - Metoda getSubscriptii returnează lista de abonamente pentru un utilizator.
  - Permite sortarea rezultatelor în funcție de diferite criterii.
- Filtrare Abonamente după Categorie:
  - Metoda getSubscriptiiDupaCategorie filtrează abonamentele unui utilizator după o categorie specifică.
- Modificare Abonament:

- Metoda modificaSubscriptie gestionează cererile POST pentru actualizarea unui abonament existent.
- Validează noile date (preț, dată de început) înainte de actualizare.

2.10. ControllerUser

- Verificarea Autentificării:
  Verifică dacă există un utilizator autentificat în sesiune.
- Afișarea Informațiilor de Profil:
  Dacă utilizatorul este autentificat, adaugă în model:
  Numele utilizatorului, ID-ul utilizatorului, Adresa de email
- Integrarea Statisticilor Abonamentelor: Pentru utilizatorii autentificați, adaugă în model: Costul lunar total al abonamentelor, Procentajele pe categorii de abonamente
- Pregătirea Modelului pentru View: Setează un flag loggedIn pentru a indica starea de autentificare în view.

2.11. HomeController

- Metoda home() este mapată la ruta root ("/") a aplicației.
- Când un utilizator accesează URL-ul de bază al aplicației, această metodă este invocată.
- Returnează string-ul "home"

2.12. Validator

- Validare Parolă (ValidareParola):
  - Verifică dacă parola are cel puțin 8 caractere.
  - Asigură prezența a cel puțin: O literă mare, Trei cifre, Un caracter special.
  - Returnează true dacă parola îndeplinește toate criteriile, altfel false.

- Validare Email (ValidareEmail):
  - Utilizează o expresie regulată pentru a verifica formatul adresei de email.
  - Acceptă adrese de email care: Conțin caractere alfanumerice, plus, punct sau liniuță înainte de @, Au un domeniu valid după @, Se termină cu ".com"

- Returnează true pentru adrese valide, false în caz contrar.

- Verificare Caracter Special (ContineCaracterSpecial):
  - Verifică prezența oricărui caracter care nu este literă, cifră sau spațiu.
  - Utilizată intern de ValidareParola și poate fi folosită independent.

2.13. SecurityConfig

- Configurare SecurityFilterChain:
  - Metoda configure definește regulile de securitate pentru cererile HTTP.
  - Permite accesul public la toate rutele ("/**".permitAll()).
  - Dezactivează protecția CSRF (Cross-Site Request Forgery).
  - Dezactivează CORS (Cross-Origin Resource Sharing).

- Configurare PasswordEncoder:
  - Metoda passwordEncoder definește un bean pentru criptarea parolelor.
  - Utilizează BCryptPasswordEncoder, un algoritm puternic de hashing pentru parole.

2.14. SubscriptionManagerApplication

- Inițializarea Aplicației Spring Boot:
  - Metoda main este punctul de start al aplicației.
  - Utilizează SpringApplication.run() pentru a porni aplicația Spring Boot.

3. Descrierea elementelor funcționale oferite prin interfața grafica cu utilizatorul

3.1. Pagina Home

Bară de navigare (navbar):

- Logo și titlu: Afișează iconița aplicației și numele "Subscription Manager".

- Buton "My Profile": Un link funcțional către pagina de profil a utilizatorului (/web/profile).

Conținut principal:

- Titlu de bun venit: "Welcome to Subscription Manager".
- Descriere scurtă a funcționalității aplicației.

Integrare Thymeleaf:

- Utilizarea namespace-ului Thymeleaf (xmlns:th="http://www.thymeleaf.org") pentru procesarea dinamică a conținutului pe server.
- Exemplu de utilizare: th:href="@{/web/profile}" pentru generarea corectă a URL-ului către pagina de profil.

3.2. Pagina Profile

Bară de navigare (navbar):

- Logo și titlu: Afișează iconița aplicației și numele "Subscription Manager".

Secțiune de conținut principal:

- Titlu "My Profile".
- Afișare mesaje de eroare sau succes.

Funcționalități pentru utilizatori neautentificați:

- Buton "Log In": Deschide un modal pentru autentificare.
- Buton "Sign Up": Deschide un modal pentru înregistrare.

Funcționalități pentru utilizatori autentificați:

- Afișare informații utilizator: nume, ID și email.
- Buton "My Subscriptions": Redirecționează către pagina de abonamente a utilizatorului.
- Buton "My Statistics": Deschide un modal cu statistici despre abonamente.
- Buton "Log Out": Permite deconectarea utilizatorului.

Modal de autentificare:

- Formular de autentificare cu câmpuri pentru email și parolă.

Modal de înregistrare:

- Formular de înregistrare cu câmpuri pentru nume utilizator, parolă și email.
- Validare client-side pentru email și parolă.

Modal de statistici:

- Afișează costul total lunar al abonamentelor.
- Prezintă distribuția procentuală a costurilor pe categorii de abonamente.

Gestionare erori și validare:

- Afișare mesaje de eroare pentru autentificare și înregistrare.
- Validare în timp real pentru formularul de înregistrare.

Interactivitate JavaScript:

- Deschiderea și închiderea modalurilor.
- Gestionarea evenimentelor pentru butoane și formulare.

3.3. Pagina user-subscriptii

Listare Abonamente:

- Tabel cu detaliile abonamentelor utilizatorului.

Adăugare Abonament:

- Buton pentru deschiderea modalului de adăugare.
- Formular pentru introducerea detaliilor noului abonament.

Actualizare Abonament:

- Buton pentru deschiderea modalului de actualizare.
- Formular pre-completat pentru modificarea prețului și datei de facturare.

Ștergere Abonament:

- Buton pentru inițierea procesului de ștergere.
- Modal de confirmare pentru prevenirea ștergerii accidentale.

Filtrare și Sortare:

- Dropdown pentru filtrarea abonamentelor după categorie.
- Modal pentru sortarea abonamentelor după preț sau dată de facturare.

Navigare:

- Buton pentru revenirea la pagina de profil.

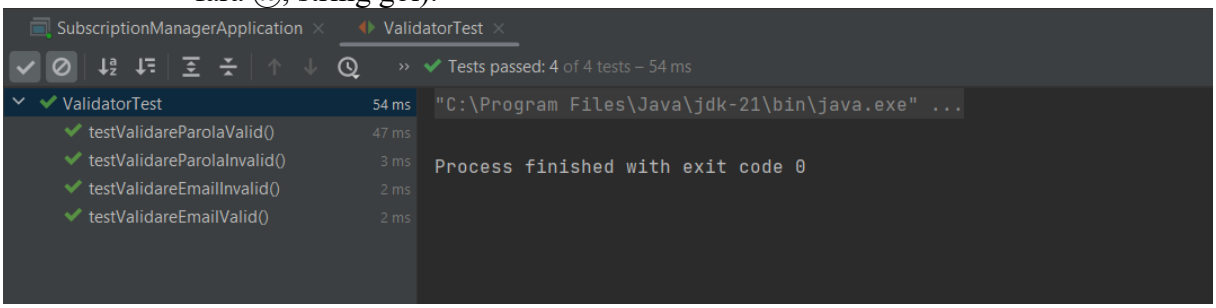
4. Aspecte privind testarea aplicației

Testele pentru aplicația de management al abonamentelor sunt organizate în mai multe clase de test, fiecare verificând funcționalități specifice.

- ValidatorTest

Testare Parole:
  - testValidareParolaValid(): Verifică parole care respectă toate criteriile (literă mare, trei cifre, caracter special).
  - testValidareParolaInvalid(): Verifică cazuri de parole invalide (fără literă mare, mai puțin de trei cifre, fără caracter special).

Testare Email:
  - testValidareEmailValid(): Verifică adrese email valide care se termină în .com.
  - testValidareEmailInvalid(): Verifică cazuri de email invalid (fără .com, fără @, string gol).



- SubscriptieTest

Testare Atribute:
  - testSetAndGetUser(): Verifică asocierea corectă între abonament și utilizator.
  - testSetAndGetStartDate(): Verifică setarea și obținerea datei de început.
  - testSetAndGetPrice(): Verifică setarea și obținerea prețului.

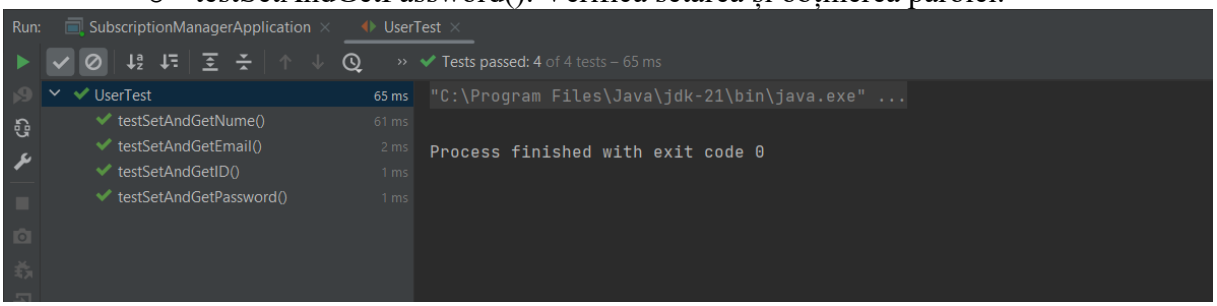
Testare Funcționalitate:
  - testGetZilePanaLaUrmatoareaPlata(): Verifică calculul corect al zilelor până la următoarea plată.

Iliescu Bianca-Ana-Maria
332AA

- UserTest

Testare Date Utilizator:

- testSetAndGetID(): Verifică setarea și obținerea ID-ului utilizatorului.
- testSetAndGetNume(): Verifică setarea și obținerea numelui.
- testSetAndGetEmail(): Verifică setarea și obținerea emailului.
- testSetAndGetPassword(): Verifică setarea și obținerea parolei.



Fiecare test folosește assertions pentru a verifica că rezultatul obținut corespunde cu rezultatul așteptat, asigurând astfel funcționarea corectă a aplicației.

5. 5 idei despre cum poate fi imbunatatita aplicatia

5.1. Integrare și Automatizare

- Sincronizare cu aplicații bancare pentru urmărirea automată a plăților
- Integrare cu servicii populare de abonamente pentru import automat
- Actualizare automată a prețurilor din surse externe
- Sistem de backup automat al datelor

5.2. Funcții de Buget

- Implementarea de limite de buget lunare/anuale
- Alerte când se apropie de limita de buget
- Sugestii pentru optimizarea costurilor
- Compararea cheltuielilor cu perioade anterioare

5.3. Recomandări

- Sugestii pentru alternative mai ieftine la abonamentele existente

- Identificarea abonamentelor neutilizate sau duplicate
- Analiză predictivă a cheltuielilor viitoare
- Sfaturi personalizate pentru optimizarea costurilor

5.4. API și Integrări

- API public pentru integrare cu alte aplicații
- Conectori pentru importul automat al datelor din servicii populare (Netflix, Spotify, etc.)
- Sincronizare cu calendare (Google Calendar, Outlook)

5.5. Sistem de Versiuni și Istoric

- Istoric complet al modificărilor pentru fiecare abonament
- Rollback la versiuni anterioare ale configurațiilor
- Comparație între diferite versiuni ale abonamentelor