UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Budget Planner

Name: Tinca Andrei
Group: 30234

Table of Contents

Deliverable 1

Project Specification

Această aplicație web de tip CRUD (Create, Read, Update, Delete) permite gestionarea utilizatorilor printr-o interfață prietenoasă. Este dezvoltată cu Spring Boot, folosind arhitectura MVC și stocare în memorie (List). Utilizatorii pot fi adăugați, editați, șterși și vizualizați într-un tabel. Interfața este realizată cu Thymeleaf și stilizată cu Bootstrap. Aplicația include validări de bază și oferă navigare intuitivă între funcționalități. Aplicația simulează un sistem simplu de management al datelor, ideal pentru prototipuri sau testare. Codul este modular, ușor de extins pentru integrarea cu o bază de date reală sau autentificare. Structura proiectului respectă principiile bune de practică din Spring, facilitând mentenanța și dezvoltarea ulterioară.

Functional Requirements

1. Aplicația trebuie să permită afișarea unei liste cu toți utilizatorii existenți.
2. Utilizatorul trebuie să poată adăuga un nou utilizator completând un formular.
3. Utilizatorul trebuie să poată edita datele unui utilizator existent.
4. Utilizatorul trebuie să poată șterge un utilizator din listă.
5. Datele introduse în formular trebuie să fie validate (ex: nume și email obligatorii).
6. După fiecare operație (adăugare, editare, ștergere), aplicația trebuie să redirecționeze utilizatorul către lista actualizată.
7. Formularele trebuie să fie precompletate cu datele utilizatorului în cazul editării.
8. Interfața trebuie să fie intuitivă și responsive, utilizând Bootstrap pentru stilizare.

Use Case Model

Use Cases Identification:

Use-Case 1: Add User

- **Level:** User goal
- **Primary Actor:** Admin
- **Main Success Scenario:**
    1. Adminul accesează aplicația.
    2. Adminul navighează către pagina „Add New User”.
    3. Adminul completează formularul cu datele noului utilizator (nume, email).
    4. Sistemul validează datele introduse.
    5. Sistemul salvează noul utilizator în memorie.
    6. Sistemul redirecționează către lista actualizată cu utilizatori.
- **Extensions:**
    - Dacă datele introduse sunt invalide, sistemul afișează mesaje de eroare și nu permite salvarea.

Use-Case 2: Edit User

- **Level:** User goal
- **Primary Actor:** Admin
- **Main Success Scenario:**
    1. Adminul accesează lista de utilizatori.
    2. Adminul selectează opțiunea „Edit” pentru un utilizator existent.
    3. Sistemul încarcă formularul cu datele utilizatorului selectat.
    4. Adminul modifică datele (nume, email) și trimite formularul.
    5. Sistemul validează noile date și actualizează informațiile în memorie.
    6. Sistemul redirecționează către lista actualizată cu utilizatori.
- **Extensions:**
    - Dacă formularul este trimis fără un ID valid, sistemul nu efectuează actualizarea.
    - Dacă datele sunt invalide, se afișează mesaje de eroare și se cere corectarea acestora.

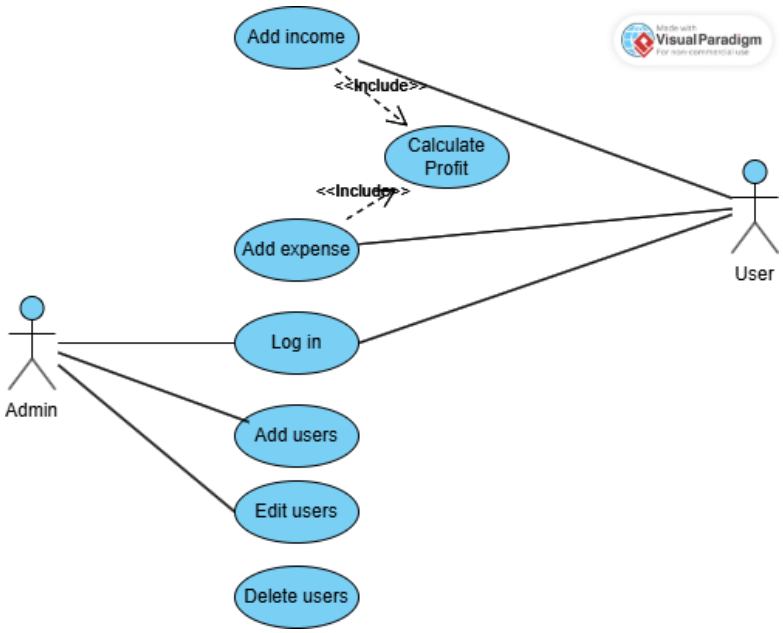
Use-Case 3: Delete User

- **Level:** User goal
- **Primary Actor:** Admin
- **Main Success Scenario:**
    1. Adminul accesează lista de utilizatori.
    2. Adminul selectează opțiunea „Delete” pentru un utilizator.
    3. Sistemul șterge utilizatorul din lista internă.
    4. Sistemul redirecționează către lista actualizată.
- **Extensions:**
    - Dacă utilizatorul nu există (ID invalid), sistemul nu face nicio modificare și poate afișa un mesaj de eroare.

Use-Case 4: View User List

- **Level:** User goal
- **Primary Actor:** Admin
- **Main Success Scenario:**
    1. Adminul accesează ruta /users.
    2. Sistemul preia toți utilizatorii din memorie.
    3. Sistemul afișează lista utilizatorilor într-un tabel.
- **Extensions:**
    - Dacă nu există utilizatori, sistemul afișează un mesaj informativ.

UML Use Case Diagrams



Supplementary Specification

Non-functional Requirements

1. Usability

- **Descriere**: Interfața aplicației trebuie să fie intuitivă și ușor de utilizat atât pentru utilizatori obișnuiți, cât și pentru admin. Navigarea între funcționalități (adăugare venituri, cheltuieli, ștergerea acestora, calculul bugetului) trebuie să se facă rapid și fără confuzie.
- **Justificare**: Aplicația este destinată utilizatorilor obișnuiți, care trebuie să poată gestiona bugetul personal fără dificultăți tehnice. O bună uzabilitate asigură o experiență plăcută și crește șansele ca aplicația să fie folosită constant.

2. Performance

- **Descriere**: Sistemul trebuie să răspundă la acțiunile utilizatorului (adăugare, ștergere, calcul buget) în maximum 2 secunde.
- **Justificare**: Deoarece aplicația implică operații simple asupra datelor (sume, liste), timpul de răspuns trebuie să fie rapid pentru a nu frustra utilizatorii. Performanța bună contribuie la o experiență fluentă.

3. Security

- **Descriere**: Toate datele personale și financiare ale utilizatorilor trebuie protejate prin autentificare securizată și stocate criptat. Admin-ul trebuie să aibă acces doar la funcțiile administrative.
- **Justificare**: Bugetul personal este o informație sensibilă, iar protecția datelor este esențială. Implementarea unor măsuri de securitate previne accesul neautorizat și crește încrederea utilizatorilor în aplicație.

4. Scalability

- **Descriere**: Sistemul trebuie să fie capabil să gestioneze un număr crescut de utilizatori și date financiare fără a afecta performanța.
- **Justificare**: Pe măsură ce aplicația devine populară, este important să poată susține mai mulți utilizatori simultan. Scalabilitatea permite extinderea ușoară a aplicației în viitor, fără modificări majore în arhitectură.

Design Constraints

1. **Model Arhitectural – Client-Server**
   - **Constrângere**: Aplicația trebuie să urmeze o arhitectură **client-server**, unde interfața utilizator este separată de logica de server și bază de date.
   - **Motivare**: Această separare permite o mai bună organizare a codului, ușurează întreținerea sistemului și permite scalabilitate.
2. **Autentificare și Autorizare**
   - **Constrângere**: Utilizatorii trebuie să se autentifice înainte de a accesa funcționalitățile aplicației, folosind un sistem de login bazat pe token JWT.
   - **Motivare**: Este o cerință de securitate care asigură că doar utilizatorii autorizați pot accesa sau modifica datele financiare personale.

3. **Instrumente de Dezvoltare**
    - **Constrângere**: Trebuie utilizate anumite instrumente recomandate, precum **InteliJ** pentru editare, **Git** pentru versionare, și **Postman** pentru testarea API-urilor.
    - **Motivare**: Aceste instrumente sunt standard în industrie și permit o dezvoltare colaborativă eficientă și testare facilă a componentelor aplicației.

Deliverable 2

Domain Model

[Define the domain model and create the conceptual class diagrams]

Architectural Design

Conceptual Architecture

[Define the system's conceptual architecture; use an architectural style and pattern - highlight its use and motivate your choice.]

Package Design

[Create a package diagram]

Component and Deployment Diagram

[Create the component and deployment diagrams.]

Deliverable 3

Design Model

Dynamic Behavior

[Create the interaction diagrams (2 sequence) for 2 relevant scenarios]

Class Diagram

[Create the UML class diagram; apply GoF patterns and motivate your choice]

Data Model

[Create the data model for the system.]

System Testing

[Describe the testing methides and some test cases.]

Future Improvements

[Present some features that apply to the application scope.]