**Уеб приложение за управление на личния бюджет**

Албена Кьосева 4 курс, информатика, задочно

ФН 2401262052

**Необходимост от приложението**

Приложението за управление на личния бюджет ще помогне на потребителите да проследяват своите разходи и доходи. В днешно време много хора търсят лесни и удобни инструменти, които да им позволят да планират по-добре личните си финанси. С това приложение те ще могат:

* Да добавят информация за своите доходи и разходи.
* Да категоризират разходите си, за да получат по-ясна представа за навиците си на харчене.
* Да наблюдават месечния си баланс, което ще им помогне да вземат по-информирани финансови решения.

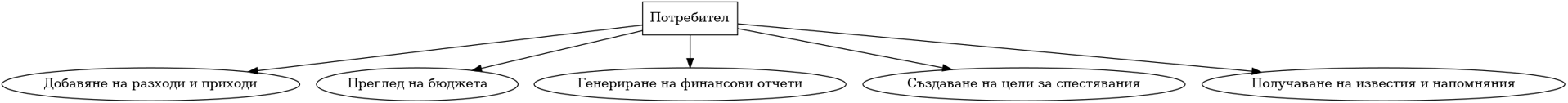
**Потребителски групи**

Приложението е предназначено за следните потребителски групи:

* **Студенти** – за да управляват бюджета си, докато са в университета.
* **Млади професионалисти** – за да проследяват първите си доходи и разходи.
* **Всички, които искат да имат по-добър контрол върху финансите си.**

**Основни функционалности:**

1. **Добавяне на приходи и разходи**: Потребителят може да добавя доходи и разходи, като избира категория и въвежда стойност.
2. **Планиране на бюджети**: Потребителите могат да задават месечни или седмични бюджети за различни категории като храна, транспорт, развлечения и др.
3. **Финансови отчети**: Приложението генерира отчети за общите разходи и приходи, представени графично, за по-лесно разбиране.
4. **Цели за спестявания**: Потребителите могат да поставят цели за спестявания (например за ваканция или голяма покупка) и да следят напредъка.
5. **Известия и напомняния**: Приложението изпраща известия при достигнати бюджети или предстоящи плащания.
6. **Препоръки**: С помощта на анализи, приложението предлага препоръки за оптимизиране на разходите.



**Ограничения**

1. Приложението ще бъде достъпно само чрез уеб браузър (без мобилни приложения).
2. Няма да бъде осигурена интеграция с банкови сметки или автоматично синхронизиране на транзакции.
3. Приложението ще изисква интернет връзка за синхронизация на данни между устройства.

**Архитектура на приложението**

Архитектурата на уеб приложението ще бъде изградена с модел на три слоя:

1. **Frontend (клиентска част)**:
   * **Технология**: **HTML, CSS, JavaScript**, с използване на библиотека **React.js** за динамично изграждане на интерфейса.
   * Основната цел на уеб приложението е да осигури интерфейс за взаимодействие с потребителите. Ще има екрани за:
     + Добавяне на разходи и приходи.
     + Преглед на бюджета.
     + Финансови отчети.
     + Цели за спестявания.
     + Настройки на потребителя.
   * За динамичност на страниците ще използваме **React.js**, който ще ни позволи да изграждаме компоненти и да управляваме състоянието на приложението.
2. **Backend (сървърна част)**:
   * **Технология**: **Java с Spring Boot** за изграждане на RESTful API.
   * Spring Boot ще се използва за сървърната част, която ще осигури RESTful API за комуникация с базата данни и обработка на заявките.
   * Основни компоненти на backend:
     + **Controller Layer**: Обработва HTTP заявки от клиентската част (GET, POST, PUT, DELETE).
     + **Service Layer**: Логика за обработка на данни (например изчисляване на баланси, генериране на отчети).
     + **Repository Layer**: За комуникация с базата данни. Ще използваме **Spring Data JPA** за свързване с релационни бази данни.
3. **База данни**:
   * Ще използваме **релационна база данни** (например **MySQL** или **PostgreSQL**) за съхранение на информация за потребителите, разходите, приходите, бюджетите и целите.
   * Структурата на базата данни ще включва таблици за:
     + **User (Потребители)**: Съхранява информация за потребителите.
     + **Transaction (Транзакции)**: Съхранява записи за приходи и разходи.
     + **Budget (Бюджет)**: Управлява лимити за различни категории.
     + **Goal (Цели)**: Съхранява цели за спестявания.
     + **Report (Отчети)**: Съхранява генерирани отчети.

A diagram of a user id

AI-generated content may be incorrect.

**Структура на приложението (Class диаграма)**

Основните класове в приложението ще включват:

1. **User**:
   * Атрибути: username, email, password
   * Методи: login(), register(), updateProfile()
2. **Transaction**:
   * Атрибути: amount, category, date, type (разход или приход)
   * Методи: addTransaction(), getTransactions()
3. **Budget**:
   * Атрибути: category, limit, spent
   * Методи: createBudget(), updateBudget(), getRemainingBudget()
4. **Goal**:
   * Атрибути: goalName, targetAmount, currentAmount, targetDate
   * Методи: createGoal(), updateGoalProgress(), getGoalStatus()
5. **Report**:
   * Атрибути: startDate, endDate, incomeTotal, expenseTotal, categoriesBreakdown
   * Методи: generateMonthlyReport(), generateWeeklyReport()

**Activity и Sequence диаграми**

1. **Activity диаграма за добавяне на разход**:
   * Потребителят избира "Добави разход".
   * Въвежда сума, категория и дата.
   * Потвърдителен екран, за да запише разхода в системата.
   * Данните се записват в базата данни.

A diagram of a process

AI-generated content may be incorrect.

1. **Sequence диаграма за генериране на отчет**:
   * Потребителят избира "Генерирай отчет".
   * Приложението изпраща заявка към backend.
   * Backend събира данни и генерира отчет.
   * Отчетът се връща на уеб приложението и се показва на потребителя.

A diagram of a diagram

AI-generated content may be incorrect.

**State диаграма (например за цели)**

Целите ще преминават през състояния като:

* **Активна**: Целта е активна и потребителят я изпълнява.
* **Завършена**: Целта е постигната.
* **Неуспешна**: Потребителят не е изпълнил целта в срока.

A diagram of a company

AI-generated content may be incorrect.

**Тестови случаи**

1. **Тест 1 - Добавяне на разход**:
   * **Вход**: Потребител добавя разход от 100 лв. за храна.
   * **Очакван резултат**: Разходът се появява в списъка с разходи и се добавя към общия баланс.
2. **Тест 2 - Генериране на отчет**:
   * **Вход**: Потребител избира да генерира отчет за последния месец.
   * **Очакван резултат**: Генерира се отчет с правилни данни за приходи и разходи.

**Заключение:**

Това уеб приложение ще предостави на потребителите лесен и интуитивен начин да управляват личните си финанси, като използват съвременни технологии за изграждане на уеб интерфейси (HTML, CSS, JavaScript с React) и мощни backend технологии като Java Spring Boot за обработка на данни и взаимодействие с базата данни.