Florina Danciu-Wendt

Proiect– Java Development

Wimm service

Cuprins

1. Ideea și motivația proiectului
2. Funcționalități
3. Arhitectură
4. Tehnologii folosite
5. Funcționalități viitoare
6. Ideea

Adesea ne adresăm întrebari la sfârșitul unei luni sau al unui an referitoare la direcția pe care o au finanțele proprii sau comune în familie.

În ce am investit un anumit procent din veniturile anuale?

Care sunt activitățile noastre cotidiene care ne consumă finanțele și mai ales în ce proporție?

Ce produse, activități, servicii ne sunt dispensabile sau indispensabile?

Pentru a putea dobândi o viziune de ansamblu la nivel lunar, semestrial, anual – mai exact, configurabil în funcție de preferințele și nevoile utilizatorului, am decis implementarea unei aplicației software wimm (“where is my money?”) care să ofere acest support având la bază informațiile disponibile în extrasul de cont bancar.

Argumentarea alegerii tehnologiilor folosite

Am abordat dezvoltarea aplicației într-o maniera iterativa, astfel încât în faza de produs minim viabil (MVP) rezultatele fiecărui pas din etapa de implementare să genereze plus-valoare pentru utilizator, acesta având posibilitatea vizualizării datelor într-un format user-friendly prin intermediul dashboard-urilor Kibana.

Am dorit să implementez aplicația WIMM cu un stack tehnologic “state of the art”, astfel încât REST API a fost prima opțiune atunci când am pornit cu ideea de bază a aplicației. Întrucât payload-ul obiectului JSON derivă din sintaxa object notation-ului din Javascript, motivul cel mai evident este compatibilitate cu limbajul de programare Javascript, tehnologie care va fi folosită primar viitorul apropiat în implementarea web clientului.

Pentru dezvoltarea serviciului REST am optat pentru framework-ul bazat pe tehnologie Java Spring boot având în vedere anumite considerente precum:

* efort minim de configurare inițială
* proces de deployment relativ simplu, făcând uz de application serverul Tomcat embedded
* dependency management gestionat cu ușurință mulțumită setului de dependințe standard default
* dependency injection, funcționalitate specifică spring boot care a facilitat implementarea

De asemenea, în cadrul implementării serviciul REST am dobândit și aplicat cunoștiințe în materie de design patterns, de exemplu “data transfer object” pentru a decupla informația primită și trimisă către client de restul logicii de business și maparea cu entitatea Java folosită intern la nivel de data access object layer. Practic am decuplat presentation layer-ul de persistence layer.

Librăria ModelMapper mi-a fost de un real ajutor atunci când am convertit automat obiectele dto în entități JPA (data entities).

Am extras implementarea business logic-ului într-un layer additional, service layer pentru a separa responsabilitățile în Rest Controller, acesta având doar rolul de a comunica cu clientul si de a delega procesarea către service layer, de a recepta și trimite request-uri HTTP, aplicând principiul “separation of concerns”.

Pentru a persistate entități JPA în baza de date relațională PostgreSQL am folosit implementarea JPA din Spring Data (bazată pe tool ORM Hibernate).

Folosind tool-urile open source Elastic Search și Kibana din stack-ul ELK am facilitat vizualizarea grafică a tranzacțiilor bancare în diferite formate.

Pentru a putea extrage informații din Elastic Search, concomitent cu persistarea tranzacțiilor bancare în baza de date PostgreSQL are loc indexarea obiectelor de transfer (dto) în format de Elastic Search document din format JSON.

Pe baza indexului de ElasticSearch, respectiv a documentelor persistate, Kibana oferă posibilitatea configurării și vizualizării diferitelor grafice.

Optând pentru tehnologii “state of the art” am găsit varietate în materie de documentație, support,metode de troubleshooting și alte activități adiacente dezvoltării aplicațiilor software în internet.

1. Functionalități

* Crearea și persistarea tranzacțiilor bancare
* Actualizarea tranzacțiilor bancare
* Ștergerea tranzacțiilor bancare
* Crearea automatizată a unei categorii generată în cadrul creării unei tranzacții bancare
* Actualizarea categoriilor
* Ștergerea categoriilor
* Vizualizarea datelor din tranzacțiile bancare persistate în funcție de categorie pentru un interval customizabil de timp

1. Arhitectura software

Diagram

Description automatically generated

1. Tehnologii

Test Client:

* Postman

Spring Boot:

* Java
* Spring Boot Data
  + Jackson – JSON processing
  + Jersey (JAX-RS – Restful WS)
* Spring Boot JPA
  + Hibernate
    - * JDBC
* Spring Boot ElasticSearch
* PostgreSQL
* Kibana
* Elastic Search
  + Lucene
* Docker for Desktop
* MavenDiagram

  Description automatically generated

1. Funcționalități viitoare

Diagram

Description automatically generated

* Interfață grafică pentru utilizator (web client UI)
* Securizarea serviciilor REST
* Creare unui cont nou de utilizator
* Login pentru utilizatori
* Upload de fișiere în format CSV
* Afișarea datelor per cont bancar în diferite formate vizuale (grafice, chart-uri, pie chart-uri)
* Opțiunea de download pentru graficele generate de către UI
* Integrare cu terțe interfețe ( de ex. Azure cognitive services) pentru extinderea funcționalității actuale ( transformarea in string-uri json prin metoda OCR a fișierelor .pdf)
* Configurarea de limite per categorie
* Notificare per e-mail în cazul depășirii limite configurate
* Implementarea unei versiuni de mobile app
* Integrare directă cu API-ul băncii pentru actualizare a datelor în “near real-time”
* Notificări în aplicația mobile în cazul depășirii limitelor configurate de către utilizator