Studiul și implementarea unei arhitecturi bazate pe Microservicii

Eduard Cosmin Valache

Rezumat

Tema acestei lucrări este studierea, compararea și crearea unei aplicații care se bazează pe o arhitectură de tip “Microservicii”.

Acest tip de arhitectură presupune dezvoltarea unei aplicații ca **o suită de servicii**, unde fiecare serviciu are rolul de a se ocupa de o arie specifică (*business module*). Aceste module trebuie să comunice prin intermediul unor mecanisme simple (ex. HTTP REST) și să fie construie în jurul unor domenii logice bine definite și delimitate. Ele trebuie să fie ușor de lansat în producție, independente unele de altele, iar managementul acestoră să se facă la un nivel cât mai decentralizat.

Microserviciile vin ca o alternativă la un stil de **dezvoltare monolitică** a aplicațiilor, unde acestea sunt dezvoltate ca o singură unitate. Majoritatea aplicațiilor sunt constituite deseori din 3 mari părți: o parte ce conține interfața cu utilizatorul, o bază de date și o aplicație de tip server, ce se ocupă cu logica domeniului, extragerea, întroducerea de informații în baza de date și oferirea paginilor HTML către utilizatori. Astfel, orice schimbare a acestui sistem va duce la lansarea unei noi versiuni a întregii aplicații de tip server. Pe lângă acest lucru, toate părțile sunt interconectate, aplicața este greu de modificate și greu de întreținut, greu de mutat (portat) pe alte tehnologii și nu se scalează într-un mod ușor și eficient.

Într-o arhitectura de tipul microserviciilor, se urmărește o **componentizare** a aplicației cu ajutorul unor servicii WEB. Astfel, fiecare componentă va fi reprezentată de un serviciu care (respectând principiul SRP [1]) se va ocupa de o unitate logică, bine definită, a aplicației. Acest lucru permite realizarea ușoară a modificărilor în cadrul aplicației, o mentenață mai eficientă a acestora și un proces de lansare în producție ușor de realizat și fără un impact major.

Scalabilitatea este principalul atuu al acestui tip de servicii, întrucât anumite module care sunt mai des folosite pot fi **ușor replicate** pentru a oferi performanțe mai bune. Pe lângă toate aceste beneficii, serviciile pot fi construite folosind **tehnologii diferite**, astfel putându-se alege limbajul sau tehnologiile care sunt cele mai recomandate pentru construirea acelui modul.

Obiectivul acestei lucrări este de a studia principiile care stau a baza microserviciilor, de a compara cu alte tipuri de arhitecturi asemănătoare și de implementa un astfel de tip de arhitectură bazată pe microservicii, urmărind rezolvarea problemelor și dificultăților pe care le presupune, urmând apoi enunțarea unei concluzii cu privire la alegerea făcută.

Pentru atingerea acestui obiectiv, am ales dezvoltarea unei aplicații care s-ar potrivi cu acest stil arhitectural, realizând un portal prin intermediul căruia diversele departamente, împreună cu angajații unei firme, pot realiza management-ul deplasărilor în străinătate. Ideea a pornit de la o problema reală, întrucât întreg procesul de la creare, agregare de informații și până la aprobarea și efectuarea propriu-zisă este unul anevoios, care necesită confirmări și colaborări din partea mai multor departamente. Aceste departamente pot avea propria infrastructură informatică (aplicație) care la rândul ei poate suferi modificări majore, iar modul de procesare a unei delegații poate fi schimbat sau modificat des, în funcție de cerințele organizației.

Pentru proiectarea acestei aplicații am pornit prin a defini principalele module care fac

1 Single Responsability Principle (http://www.oodesign.com/single-responsibility-principle.html)

parte atât din aplicație cât și din structura organizațională. S-a ajuns astfel la definirea următoarelor module: HR.Employee (modul ce ține de resursele umane și se ocupă cu managementul informațiilor angajaților), FNN.FundCalculator (modul de management al finanțelor), OM.FrontOffice (modulul responsabil de agregare de informații si rezervare a biletelor de avion, cazare), OM.AccommodationSnitcher (modul ce oferă informații cu privire la cazare) și OM.FlightSnitcher (modul ce oferă informații cu privire la biletele de avion). Toate aceste module reprezintă departamente bine definite din organizație, care sunt nevoite să comunice între ele pentru a duce la planificarea și realizarea unei delegații.

Una dintre principalele probleme ale acestui tip de arhitectură este lipsa standardizării. Întrucât este un model de dezvoltare software nou apărut, nu există standarde clar definite. Asta înseamnă că alegerea tehnicilor, protocoalelor, librăriilor, uneltelor și mecanismelor acestei aplicații presupun studierea, compararea și realizarea unei concluzii clare cu privire la ceea ce trebuie folosit în aplicație.

În dezvoltarea acestei aplicații, s-a ales un protocol de tip HTTP REST, o comunicație sincronă între servicii, un mecanism de tip *gateway-proxy* pentru accesarea individuală a serviciilor, un **mecanism de înregistrare a serviciilor** în cadrul întregului sistem (service registry), un **mecanism de auto-descoperire a serviciilor** (self-service discovery), un **mecanism de autentificare** de tipul OAuth, unde server-ul de resurse și cel de autorizare sunt bine definite și separate, un **mecanism de monitorizare** a tuturor instanțelor unui serviciu și un **mecanism de agregare a tuturor erorilor** diverselor componente.

Mecanismele au fost dezvoltate folosind tehnologii Microsoft, utilizând .NET 4.5, Web API 2 pentru serviciile web și AngularJS pentru partea de interfață. S-au folosit mai multe tipuri de baze de date (SQL și NoSQL), împărțite pe domeniile serviciilor pe care le deservesc, obținând astfel 2 instanțe de SQL Server și 3 instanțe de MongoDB.

Au mai rezultat astfel încă trei componente ale aplicației: Front-end (modul ce conține logica interfeței cu utilizatorul), ServiceRegistry (componentă ce are rolul de a oferi informații cu privire la alte servicii disponibile și adresele acestora), Service Authentication (modul ce ofera logica de autentificare întregii arhitecturi).

S-a ajuns astfel la o aplicație modulară, unde componentele sunt independente unele de altele, servesc un rol bine definit, se pot modifica sau schimba (ca întreg modul) foarte ușor, pot fi lansate în producție independent unele de celalalte și se pot scala independent, prin replicare, fără mecanisme adiționale. Deși sunt independente, toate au rolul de a colabora pentru a face posibilă lansarea unei delegații în cadrul unei companii.