

UNIVERSITATEA „BABEȘ-BOLYAI”  
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ  
SPECIALIZAREA INFORMATICĂ

LUCRARE DE DIPLOMĂ

---

**STUDIUL EFICIENȚEI FOLOSIRII UNEI  
ARHITECTURI BAZATE PE MICROSERVICII  
ÎNTR-UN SISTEM DE PLATĂ A UTILITĂȚILOR**

---

*Coordonatori științifici:*

lect. dr. Mircea Ioan Gabriel

*Absolvent:*

**Petruțiu Paul-Gabriel**

**Cluj-Napoca**

**2019**

# Cuprins

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Introducere</b>   | <b>3</b>  |
| <b>2</b> | <b>Concepte de bază</b>  | <b>4</b>  |
| 2.1      | Arhitectura unei aplicații soft . . . . .                              | 4         |
| 2.2      | Șabloane de proiectare . . . . .                                       | 4         |
| 2.3      | Serviciu Software . . . . .  | 5         |
| <b>3</b> | <b>Aplicații Monolit</b>   | <b>6</b>  |
| 3.1      | Ce este o aplicație monolit? . . . . .                                 | 6         |
| 3.2      | Avantaje și dezavantaje . . . . .                                      | 7         |
| <b>4</b> | <b>Arhitectura bazată pe servicii</b>                                  | <b>8</b>  |
| 4.1      | Fundamentele arhitecturii bazate pe servicii . . . . .                 | 8         |
| 4.2      | Sabloane de proiectare pentru arhitecturi bazate pe servicii . . . . . | 8         |
| <b>5</b> | <b>Sabloane de proiectare bazate pe microservicii</b>                  | <b>9</b>  |
| 5.1      | Sabloane de implementare a microservicilor . . . . .                   | 9         |
| 5.1.1    | Agregator . . . . .  | 9         |
| 5.1.2    | API Gateway . . . . .  | 9         |
| 5.2      | Sabloane arhitecturale de proiectare . . . . .                         | 9         |
| 5.2.1    | Service bus . . . . .  | 9         |
| 5.2.2    | Procesare asincronă . . . . .  | 9         |
| 5.2.3    | Agregarea proceselor . . . . .   | 9         |
| 5.2.4    | Alte Sabloane . . . . .  | 9         |
| <b>6</b> | <b>Studiu de Caz</b>   | <b>10</b> |
| 6.1      | Introducere . . . . .  | 10        |
| 6.2      | Tranziția către microservicii (Netflix) . . . . .                      | 10        |
| 6.2.1    | Motive . . . . .   | 11        |
| 6.2.2    | Avantaje și dezavantaje . . . . .                                      | 11        |
| 6.2.3    | Arhitectura noii aplicații . . . . .                                   | 11        |
| <b>7</b> | <b>Aplicație Practică</b>  | <b>12</b> |
| 7.1      | Cerința . . . . .  | 12        |
| 7.2      | Specificații . . . . .   | 12        |

|          |   |           |
|----------|---|-----------|
| 7.3      | Arhitectura aplicatiei . . . . .                      | 12        |
| 7.4      | Implementare . . . . .                                | 12        |
| 7.5      | Sablone bazate pe microservicii in practica . . . . . | 12        |
| <b>8</b> | <b>Concluzii</b>                                      | <b>13</b> |
|          | <b>Bibliografie</b>                                   | <b>14</b> |

# **Capitolul 1**

## **Introducere**

# Capitolul 2

## Concepte de bază

### 2.1 Arhitectura unei aplicații soft

Deoarece tot mai multe companii din diferite domenii au nevoie de o aplicație software personală pentru a-și îmbunătăți calitatea produselor, pentru prezentarea produselor sau pentru ușurarea unor activități din cadrul companiei, cererea de a crea cod este tot mai mare. Pentru ca aplicațiile să fie ușor de dezvoltat, pentru ca o îmbunătățire să poată fi implementată fără prea mult efort, este important ca aplicația să aibă la bază o structură bine definită. Acest lucru face ca o aplicație să fie ușor de întreținut în decursul timpului.

Având în vedere standardul IEEE Std 1471, arhitectura unei aplicații soft este definită ca „Organizarea fundamentală a unui sistem încorporat în componentele sale, relațiile dintre ele și mediul, precum și principiile care conduc proiectarea și evoluția sa.”((Hilliard, 2000))

Așadar, putem considera un sistem ca fiind o aplicație sau un set de aplicații care împreună rezolvă diferite probleme.

### 2.2 Șabloane de proiectare

Un șablon de proiectare este reprezentat ca fiind o soluție cunoscută pentru o problema recurentă.

„Șabloanele de proiectare pot fi văzute ca un mijloc de a reuși reutilizarea pe scară largă prin captarea unei practici de design de dezvoltare a software-ului de succes într-un anumit context” ((Alencar, Cowan, & Lucena, 1996)). Deci, un șablon de proiectare reprezintă doar în mod abstract o soluție pentru o problemă. Din acest motiv, șabloanele de proiectare pot fi aplicate în oricare limbaj de programare, în funcție de contextul problemei.

Motivul pentru care șabloanele de proiectare sunt relevante indiferent de limbajul de programare folosit, este acela că șabloanele de proiectare sunt doar concepte despre cum ar trebui să fie implementat codul și nu cod propriu zis.

## 2.3 Serviciu Software

Un serviciu este o componentă a unei aplicații soft care furnizează una sau mai multe funcționalități altor componente din sistem. Aceste componente pot să fie aplicații web, mobile sau chiar alte servicii.

Spre exemplu, putem presupune ca avem un site în care utilizatorii pot să achite facturile de curent, gaz, etc.. În momentul în care utilizatorul efectuează o plată, browser-ul va apela un serviciu care va procesa, în spate, aceasta tranzacție (verificarea dacă suma introdusă este validă, dacă suma este disponibilă, etc.)

Sistemele ce folosesc mai multe servicii, similar cu exemplul expus mai sus, sunt considerate ca fiind sisteme care au la bază o arhitectură orientată pe servicii.

# Capitolul 3

## Aplicații Monolit

### 3.1 Ce este o aplicație monolit?

O aplicație monolit este o aplicație a cărei cod este scris în cadrul unei singure unități structurale. Componentele din care este alcătuită aplicația sunt gândite în așa fel încât să funcționeze împreună și să se folosească de același spațiu de memorie și de aceleași resurse.

Aplicațiile monolit sunt printre cele mai răspândite din lume. Acest fapt se datorează felului în care oamenii abordează problemele. O soluție de tip monolit este prima soluție pe care un programator o va avea, deoarece este un mod natural de a gândi. În plus, pentru multe din aplicații, o soluție monolit va fi soluția perfectă, având în vedere că multe companii mici spre medii doresc să aibă o aplicație care să automatizeze anumite procese, procese care nu au o complexitate extrem de mare, iar numărul utilizatorilor nu este enorm.

Spre exemplu, să spunem că o firmă are un sediu destul de mare care are 5 săli de conferință, având în vedere numărul mare de întâlniri din interiorul firmei, aceștia doresc o aplicație în care să poată rezerva o sală de conferință pentru o anumită perioadă într-o anumită zi. O astfel de aplicație se poate realiza ușor și rapid, sub forma unei aplicații web de tip server client.

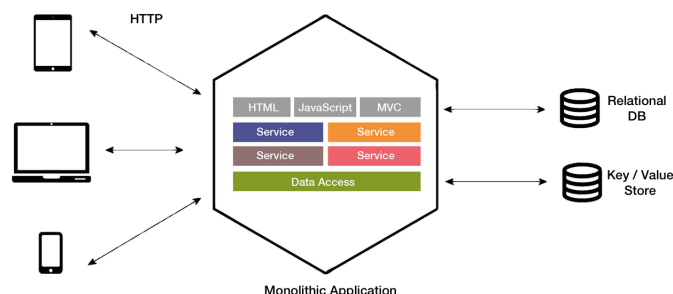


Figura 3.1: Arhitectură Aplicație Monolit(imagine : <http://bits.citrusbyte.com>)

## 3.2 Avantaje și dezavantaje

O aplicație monolit are mai multe avantaje în contextul unui business mic:

- Faza de dezvoltare durează mai puțin timp, IDE-urile moderne reușind să genereze mult cod automat, ceea ce implică scăderea costurilor de producție.
- Aplicația este ușor de testat, automatizând procesul de testare a codului folosind o combinație de unit tests și integration test.
- Procesul de lansare în producție a unei aplicații nu este complicat, deoarece este nevoie de rularea unei singure aplicații per server.

Pe de altă parte, în momentul în care aplicația va crește ca și volum al codului, un număr mai mare de programatori vor trebui implicați în procesul de dezvoltare al aplicației, acest lucru nu este un lucru rău, doar ca aceștia vor întâmpina anumite impedimente:

- Cuplajul dintre componentele aplicației crește, iar acest fapt va îngreuna procesul de dezvoltare a unor noi funcționalități, acest lucru va afecta timpul și costul necesar dezvoltării noii funcționalități.
- Alegerea tehnologiilor folosite pentru dezvoltarea aplicației este permanentă.
- Integrarea/Intrarea unui programator în proiect va fi mai dificilă. Volumul aplicației fiind mare, va fi necesar un timp mai mare pentru înțelegerea tuturor funcționalităților și a deciziilor luate pe proiect în trecut.

Acestea sunt doar câteva dintre avantajele și dezavantajele unei aplicații monolit, dar sunt suficiente încât putem sublinia o idee generală. Aplicațiile bazate pe o arhitectura monolit sunt mai ușor de implementat în primă fază cu un cost mai redus, dar în momentul în care aplicația ajunge la un nivel mai avansat, este nevoie de o reconstruire parțială sau totală a aplicației sau cel puțin este nevoie de o refactorizare care să diminueze dezavantajele create de stilul arhitectural folosit. Majoritatea aplicațiilor încep ca și aplicații tip monolit, iar mai târziu se pot transforma în aplicații cu arhitecturi bazate pe microservicii de exemplu.((Thönes, 2015))



# **Capitolul 4**

## **Arhitectura bazată pe servicii**

**4.1 Fundamentele arhitecturii bazate pe servicii**

**4.2 Sabloane de proiectare pentru arhitecturi bazate pe servicii**

# **Capitolul 5**

## **Sabloane de proiectare bazate pe microservicii**

### **5.1 Sabloane de implementare a microservicilor**

#### **5.1.1 Agregator**

#### **5.1.2 API Gateway**

### **5.2 Sabloane arhitecturale de proiectare**

#### **5.2.1 Service bus**

#### **5.2.2 Procesare asincrona**

#### **5.2.3 Agregarea proceselor**

#### **5.2.4 Alte Sabloane**

# Capitolul 6

## Studiu de Caz

### 6.1 Introducere

Dupa cum am discutat până acum, o arhitectură bazată pe microservicii nu are chiar o definiție propriu zisă, dar dupa cum susține Martin Fowler, microserviciile sunt implementarea corectă a arhitecturii bazate pe servicii.

În acest capitol vom cuprinde trecerea de la o arhitectură monolit la o arhitectura bazată pe servicii, motivele pentru care s-a facut aceasta tranziție, pașii prin care s-a facut aceasta trecere, avantajele cât și dezavantajele acestei treceri, cât și despre rezultatul final.

Netflix este o companie care a început prin a vinde sau închiria filme pe DVD-uri. Ulterior furnizând acces online la filme și seriale. Netflix fiind un gigant în industria televiziunii online. Aplicația netflix este o aplicație la scară mondială, care în momentul în care firma a hotărât schimbarea arhitecturii avea un trafic de 8 milioane de utilizatori, ajungând la finalul anul 2018 la 139 de milioane de utilizatori.

Având în vedere scala la care știm deja ca funcționează aplicația Netflix, putem deja considera că arhitectura bazată pe microservicii își servește bine scopul.

### 6.2 Tranziția către microservicii (Netflix)

În acest studiu de caz o să ne bazăm pe informațiile oferite de doua dintre personajele importante care au luat parte la tranziția către microservicii:

- Ruslan Meshenberg
- Adrian Cockcroft

### **6.2.1 Motive**

### **6.2.2 Avantaje și dezavantaje**

### **6.2.3 Arhitectura noii aplicații**

# **Capitolul 7**

## **Aplicație Practică**

**7.1 Cerința**

**7.2 Specificații**

**7.3 Arhitectura aplicației**

**7.4 Implementare**

**7.5 Sablone bazate pe microservicii în practică**

## **Capitolul 8**

### **Concluzii**

# Bibliografie

- Alencar, P. S., Cowan, D. D., & Lucena, C. J. P. d. (1996). A formal approach to architectural design patterns. In *International symposium of formal methods europe* (pp. 576–594).
- Hilliard, R. (2000). Ieee-std-1471-2000 recommended practice for architectural description of software-intensive systems. *IEEE*, <http://standards.ieee.org>, 12(16-20), 2000.
- Thönes, J. (2015). Microservices. *IEEE software*, 32(1), 116–116.
- Cockcroft, Adrian. 2015.** Talking microservices with the man who made Netflix’s cloud famous. [interview cu] Derrick Harris. 2015.
- Cockcroft, Adrian. 2014.** Migrating to Microservices by Adrian Cockcroft. YouTube. [Interactiv] 2014. <https://www.youtube.com/watch?v=1wiMLkXz26M>.
- Erl, Thomas. 2009.** SOA Design Patterns. Crawfordville Indiana : Prentice Hall, 2009.
- Fowler, Martin. 2014.** Microservices, 2014,. MartinFowler.com. [Interactiv] 2014. <https://martinfowler.com/articles/microservices.html>.
- OASIS Foundation. 2006.** Reference Model for Service Oriented Architecture. [Interactiv] 2006. <https://docs.oasis-open.org/soa-rm/v1.0/soa-rm.html>.