Tehnici Avansate de Programare

De la modularizare la arhitectură

Petru Rebeja, Marius Apetrii

21 Mai 2020

Facultatea de Matematică
Universitatea Alexandru Ioan Cuza, Iași

Introducere

Recapitulare

Recapitulare

- Elemente esențiale în proiectarea bazelor de date: `cheie primară`, `cheie străină` și `relaţie`.

Recapitulare

- Elemente esențiale în proiectarea bazelor de date: `cheie primară`, `cheie străină` și `relaţie`.
- `Normalizare` — proiectarea/restructurarea bazei de date pentru a o aduce în (cel puțin) `forma normală 3`.

Recapitulare

- Elemente esențiale în proiectarea bazelor de date: `cheie primară`, `cheie străină` și `relaţie`.
- `Normalizare` — proiectarea/restructurarea bazei de date pentru a o aduce în (cel puțin) `forma normală 3`.
- `Practicile DevOps` pentru creșterea frecvenței de livrare.

Recapitulare

- Elemente esențiale în proiectarea bazelor de date: `cheie primară`, `cheie străină` și `relaţie`.
- `Normalizare` — proiectarea/restructurarea bazei de date pentru a o aduce în (cel puțin) `forma normală 3`.
- `Practicile DevOps` pentru creșterea frecvenței de livrare.
- `Conductele de livrare` — ne ajută să implementăm practici precum:
  - `Continuous Integration` — integrarea și verificarea automată a modificărilor,
  - `Continous Delivery` — posibilitatea de a trimite oricând aplicația în producție,
  - `Continuous Deployment` — lansarea automată în producție a modificărilor noi.

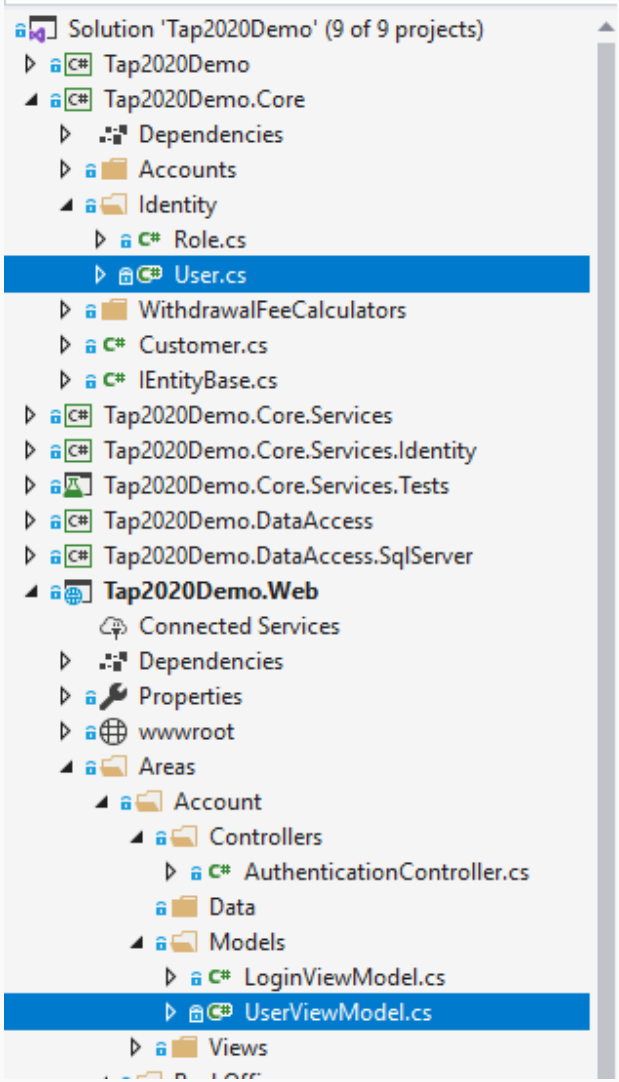
Agenda

- Studiu de caz modularizare
- Arhitectura software
- Tipuri de arhitecturi software
- Demonstrații

Modularizare — studiu de caz

Entity vs ViewModel

User vs UserViewModel

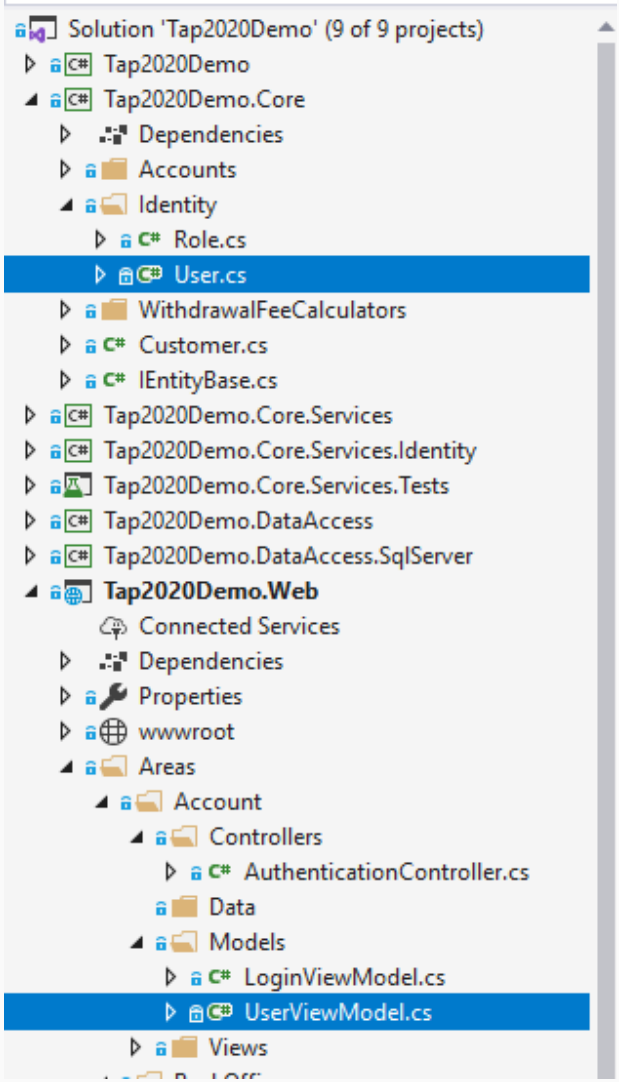


Întrebări

- De ce avem nevoie de `User` și `UserViewModel`?
- Care sunt diferențele dintre ele?

Entity vs ViewModel

User vs UserViewModel

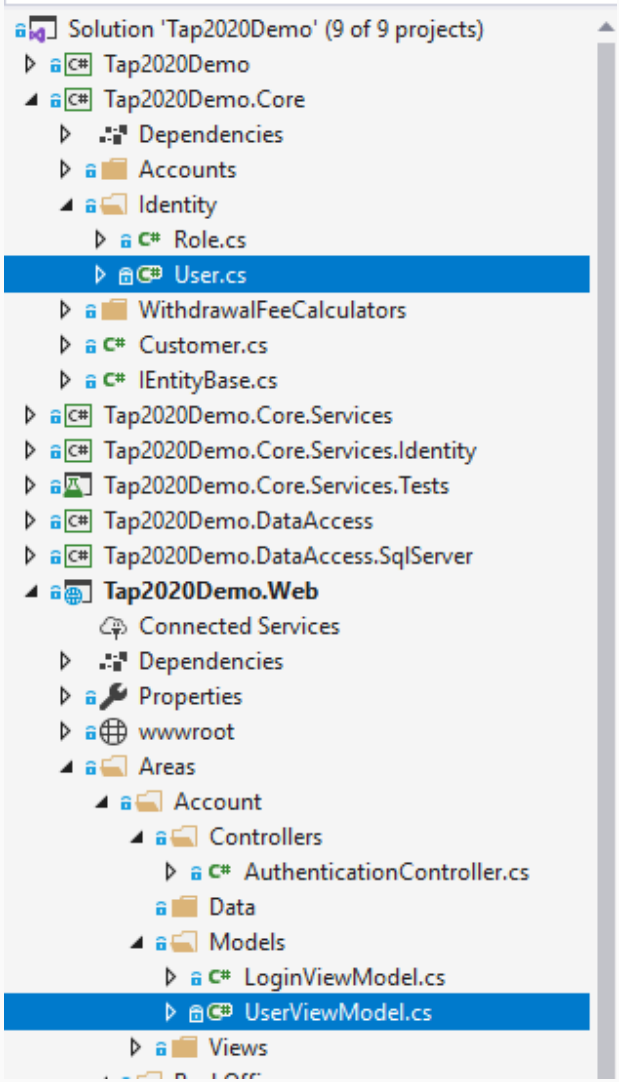


Scop

- Clasa `User`
  - Reprezintă *domeniul* aplicației
  - Se mulează pe tabelul `Users` din baza de date.
- Clasa `UserViewModel`
  - Reprezintă *modelul* pentru o anumită pagină
  - Se mulează pe câmpurile de intrare/afișare ale paginii.

Entity vs ViewModel

User vs UserViewModel

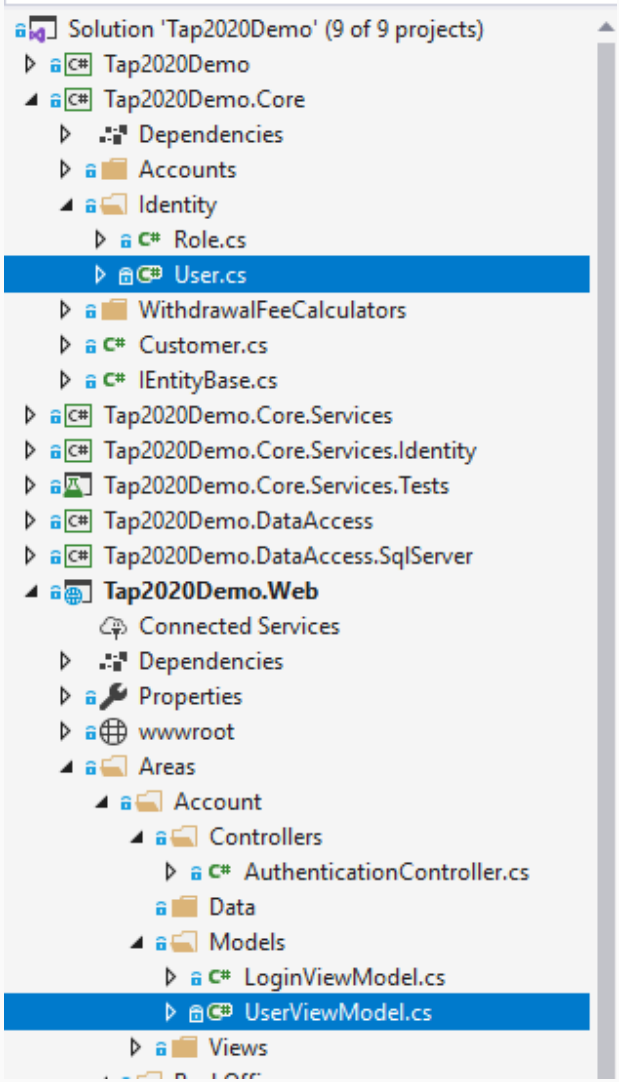


Diferențe

- `Entity` != `ViewModel` pentru multe aplicații.
- De obicei un `ViewModel` agregă date din mai multe entități.

Entity vs ViewModel

User vs UserViewModel

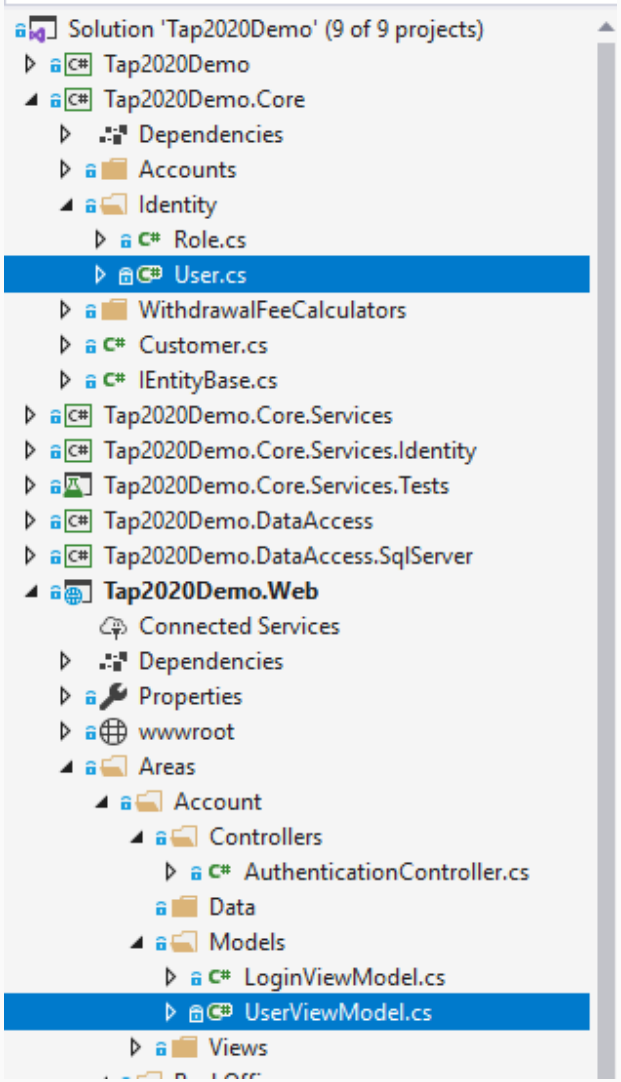


Bune practici

- Chiar dacă `ViewModel` = `Entity` se recomandă crearea unui `ViewModel`.
- Astfel decuplăm *domeniu* de *view*;
- Altfel, o modificare în entitate poate genera modificări în view.

Entity vs ViewModel

User vs UserViewModel



Securitate

- O entitate poate conține informații sensibile care nu trebuie să fie văzute de toți utilizatorii.
- La înregistrarea unei entități, utilizatorii malițioși pot suprascrie anumite atribute (`overposting`)

Arhitectura software

Definiții

Arhitectura software[1]

Modul de organizare care stă la baza unui sistem software și este încorporat în componentele acestuia, în relațiile dintre componente și mediu și în principiile care ghidează proiectarea și evoluția sistemului.

[1] IEEE Computer Society, IEEE Recommended Practice for Architectural Description of Software-Intensive Systems: IEEE Std 1472000. 2000.

Definiții

Arhitectura software[2]

Arhitectura se referă la chestiile importante. Oricare ar fi acelea.

— Ralph Johnson

[2]Ford, Neal, Rebecca Parsons, and Patrick Kua. Building evolutionary architectures: support constant change. " O'Reilly Media, Inc.", 2017.

Definiții

Arhitectura software[3]

Constă în toate deciziile importante despre structurile care alcătuiesc sistemul și interacțiunile dintre ele. Aceste decizii suportă o mulțime de caracteristici ale sistemului pe care acesta trebuie să le demonstreze pentru a deveni un sistem de succes.

[3] James McGovern, et al., A Practical Guide to Enterprise Architecture. Prentice Hall 2004

Caracteristici[4, 5]

- Performanță — timpul de răspuns per acțiune.
- Securitate — rezistență la atacuri și furt/pierdere de date.
- Scalabilitate — adăugarea de resurse noi pentru a face față cererii sporite.
- Rezistență la erori (`fault tolerance`) — posibilitatea de a continua operarea la apariția erorilor.
- etc.

[4] `https://en.wikipedia.org/wiki/List_of_system_quality_attributes`
[5] `https://syndicode.com/2018/05/03/12-software-architecture-quality-attributes/`

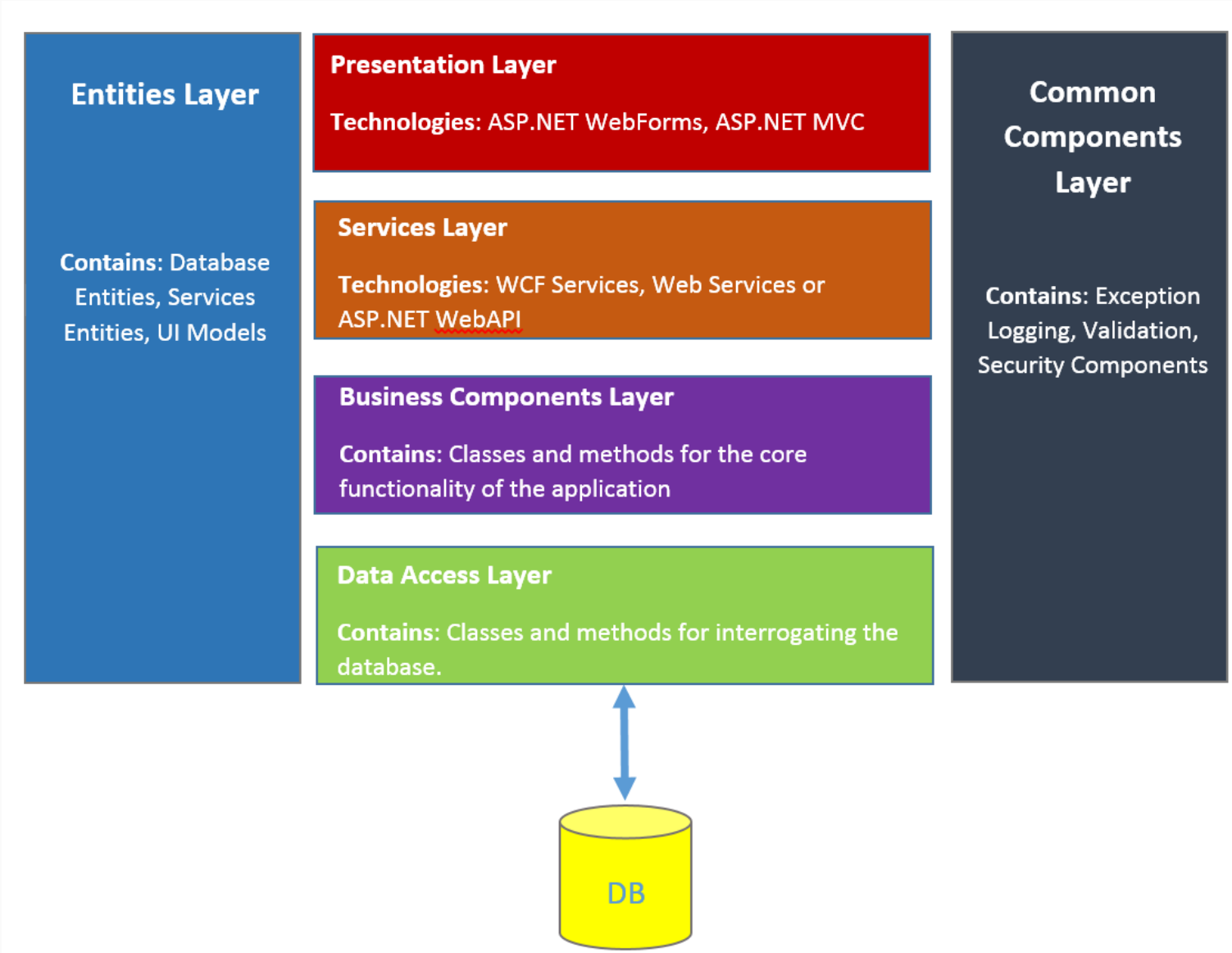
Deciziile arhitecturale

- Țin cont de caracteristicile dorite,
- Nu includ prea multe caracteristici — arhitectura generică este un anti-șablon,
- Tind spre obținerea celui mai puțin prost model[6].

[6] Ford, Neal, Rebecca Parsons, and Patrick Kua. Building evolutionary architectures: support constant change. " O'Reilly Media, Inc.", 2017.

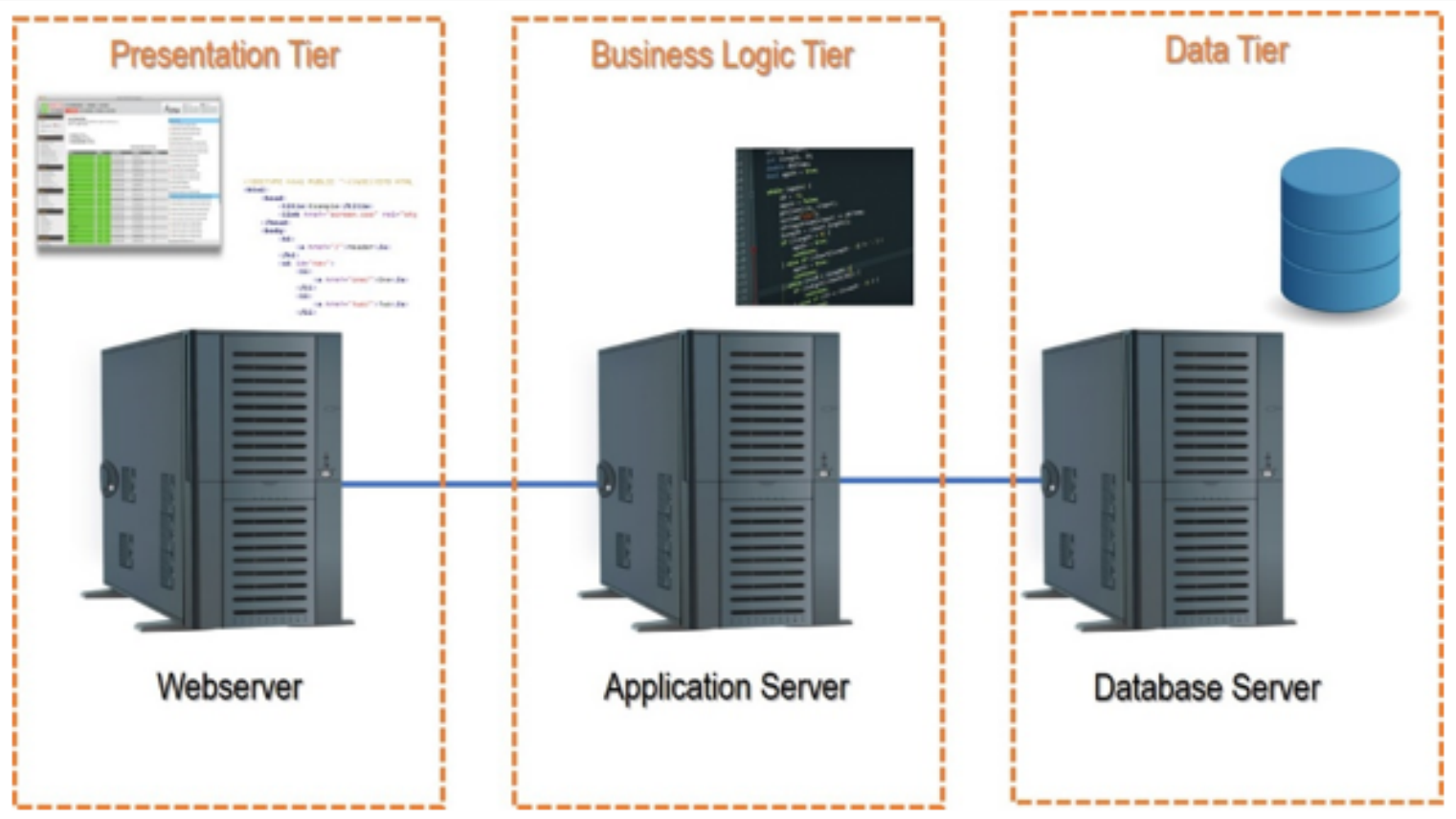
Tipuri de arhitecturi software

Arhitectura bazată pe N niveluri[7]



[7] https://dotnetdaily.net/featured/n-tier-architecture-asp-net

Arhitectura bazată pe N niveluri — organizare fizică[8]



[8] https://www.bmc.com/blogs/n-tier-architecture-tier-2-tier-3-and-multi-tier-explained/

Arhitectura bazată pe N niveluri

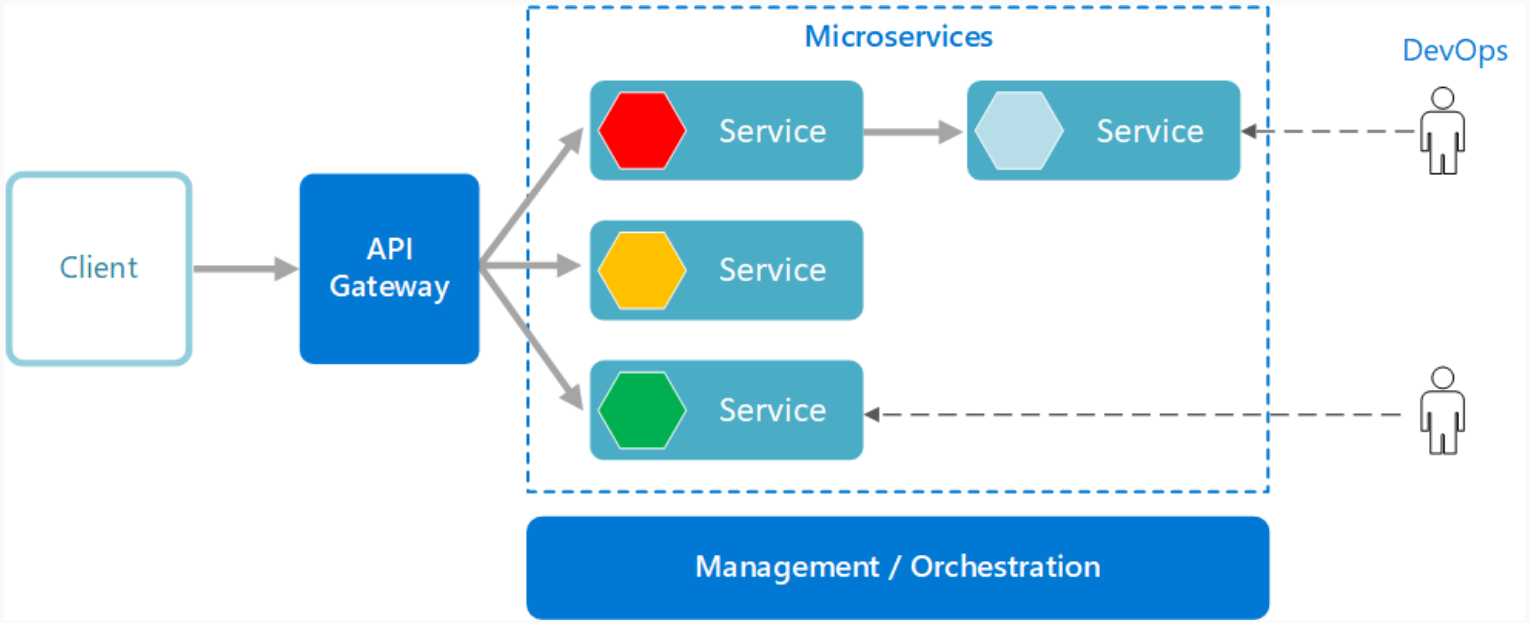
- Avantaje:
  - Complexitate redusă,
  - Ușor de portat pe alte servere.
- Dezavantaje:
  - Nu se pot aloca resurse per modul.
  - Modularizarea deficitară poate duce la latențe.

Microservicii

Definiții

1. Microserviciu — proces separat care răspunde de funcționalitatea legată de un singur aspect al domeniului unei aplicații. De obicei este apelat prin protocolul `HTTP`.
2. API Gateway — punctul de acces al metodelor expuse de microservicii.

Microservicii — arhitectură[9]



[9] https://docs.microsoft.com/nb-no/azure/architecture/guide/architecture-styles/microservices

Microservicii

- Avantaje:
  - Alocare de resure pe modul => scalabilitate îmbunătățită.
  - Ușor de întreținut — pot fi dezvoltate și instalate independent.
- Dezavantaje:
  - Complexitate sporită dată de interconectarea mai multor servicii și instalarea acestora în producție.
  - Întreținere și evoluție — fiecare versiune nouă trebuie să păstreze compatibilitatea cu una sau mai multe versiuni vechi.

Alte tipuri de arhitecturi

1. Monolit modular — sistem instalat pe un singur nivel dar organizat astfel încât modulele să fie independente și să expună funcționalitatea prin interfețe.
2. Arhitectură bazată pe evenimente — sistem în care modulele comunică prin declanșare și procesare de evenimente. Modulele sunt distribuite și au un grad maxim de decuplare.
3. Etc.

Recapitulare

Arhitectura software

Arhitectura software

- Arhitectură software — structurarea sistemului software a. î. acesta să corespundă caracteristicilor cerute (cel mai puțin prost model care corespunde cerințelor).

Arhitectura software

- Arhitectură software — structurarea sistemului software a. î. acesta să corespundă caracteristicilor cerute (cel mai puțin prost model care corespunde cerințelor).
- Există mai multe tipuri de arhitecturi: N-niveluri, microservicii, monolit modular etc.

Arhitectura software

- Arhitectură software — structurarea sistemului software a. î. acesta să corespundă caracteristicilor cerute (cel mai puțin prost model care corespunde cerințelor).
- Există mai multe tipuri de arhitecturi: N-niveluri, microservicii, monolit modular etc.
- Fiecare model vine cu avantajele și dezavantaje proprii; acestea trebuie analizate înainte de a implementa modelul ales.

Demonstrații

Încheiere

Vă mulțumesc!

Mulțumesc pentru atenție!