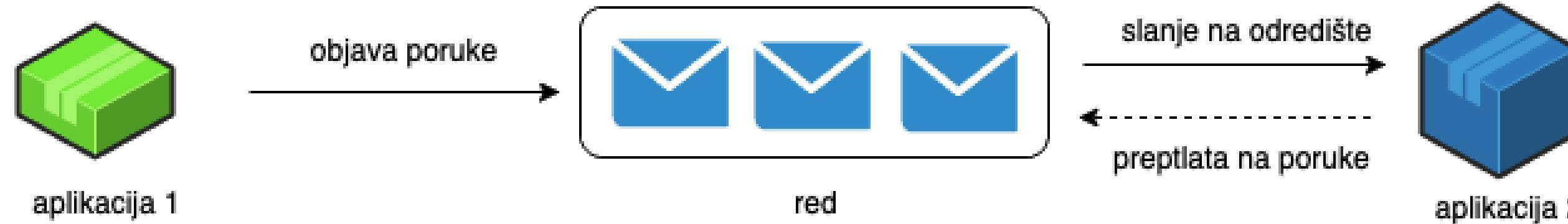


OBJEKTNO- RELACIONO MAPIRANJE

SCENARIJI TOKOVA PODATAKA

◆ TRI NAJČEŠĆA SCENARIJA

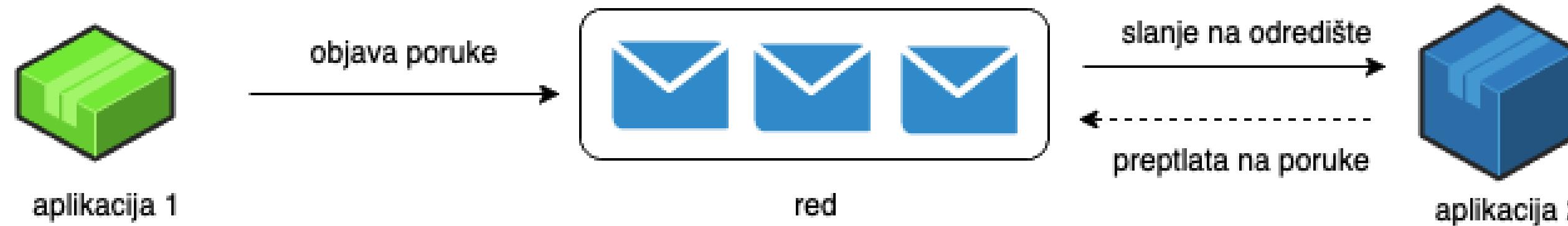
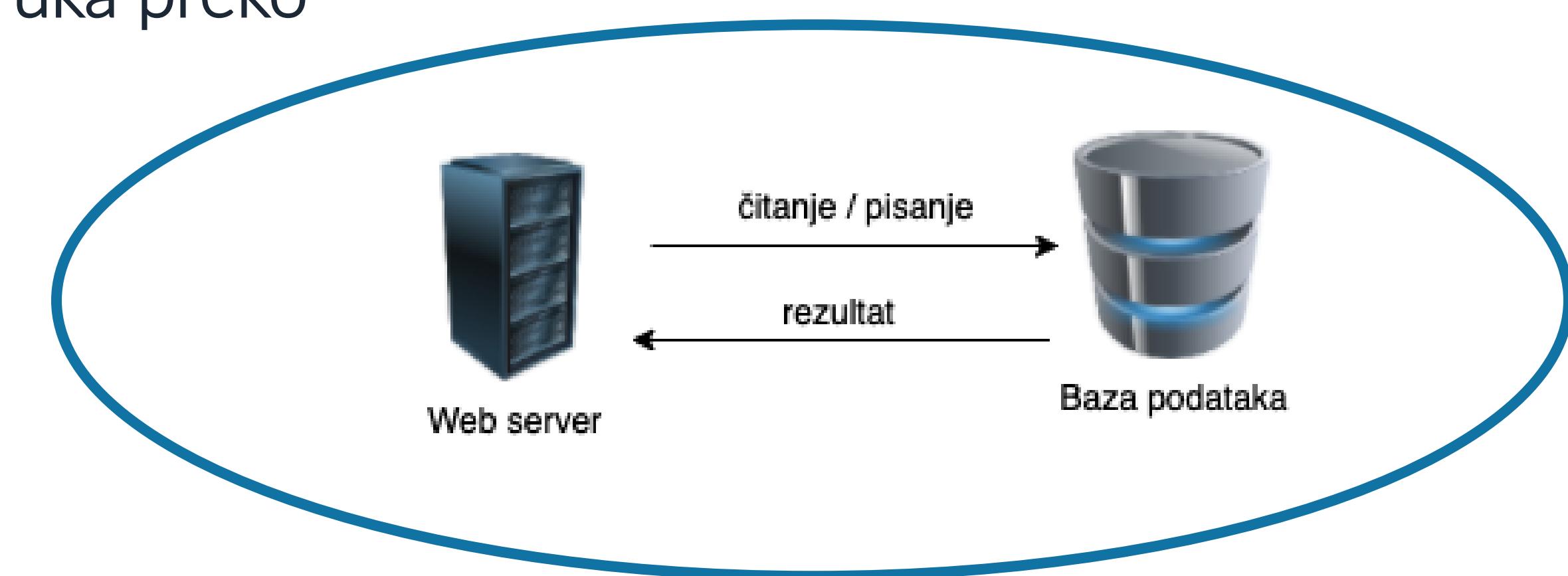
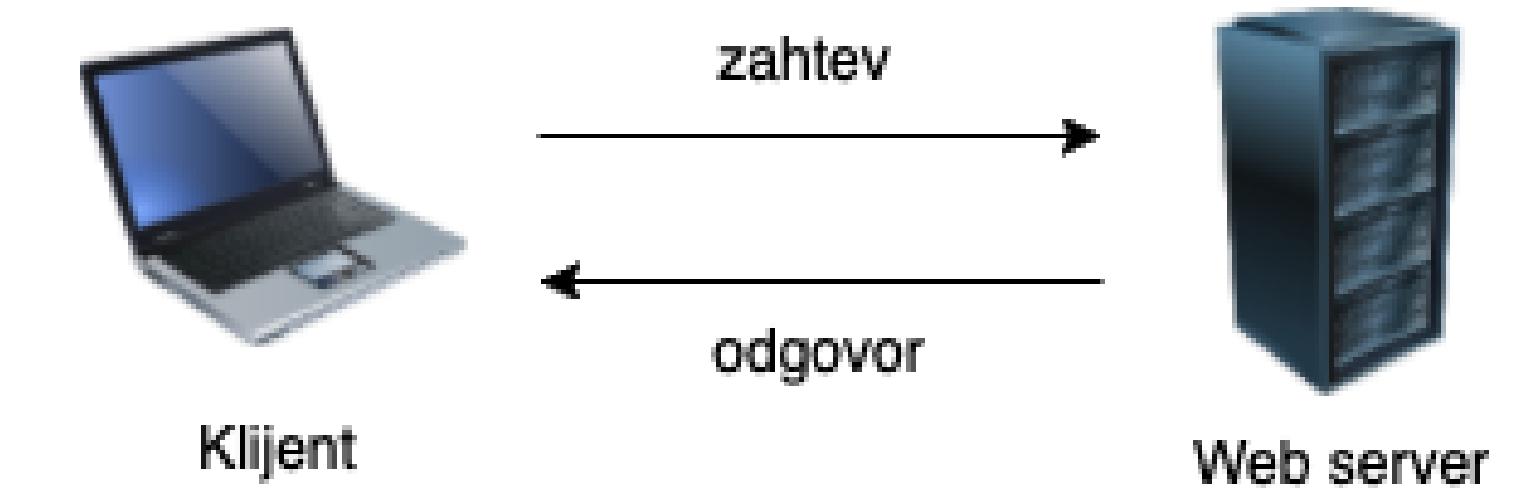
- Direktna komunikacija klijent <-> server kroz poziv servisa
- Serverska aplikacija <-> Baza podataka
- Asinhrona komunikacija razmenom poruka preko reda poruka (message queue)



SCENARIJI TOKOVA PODATAKA

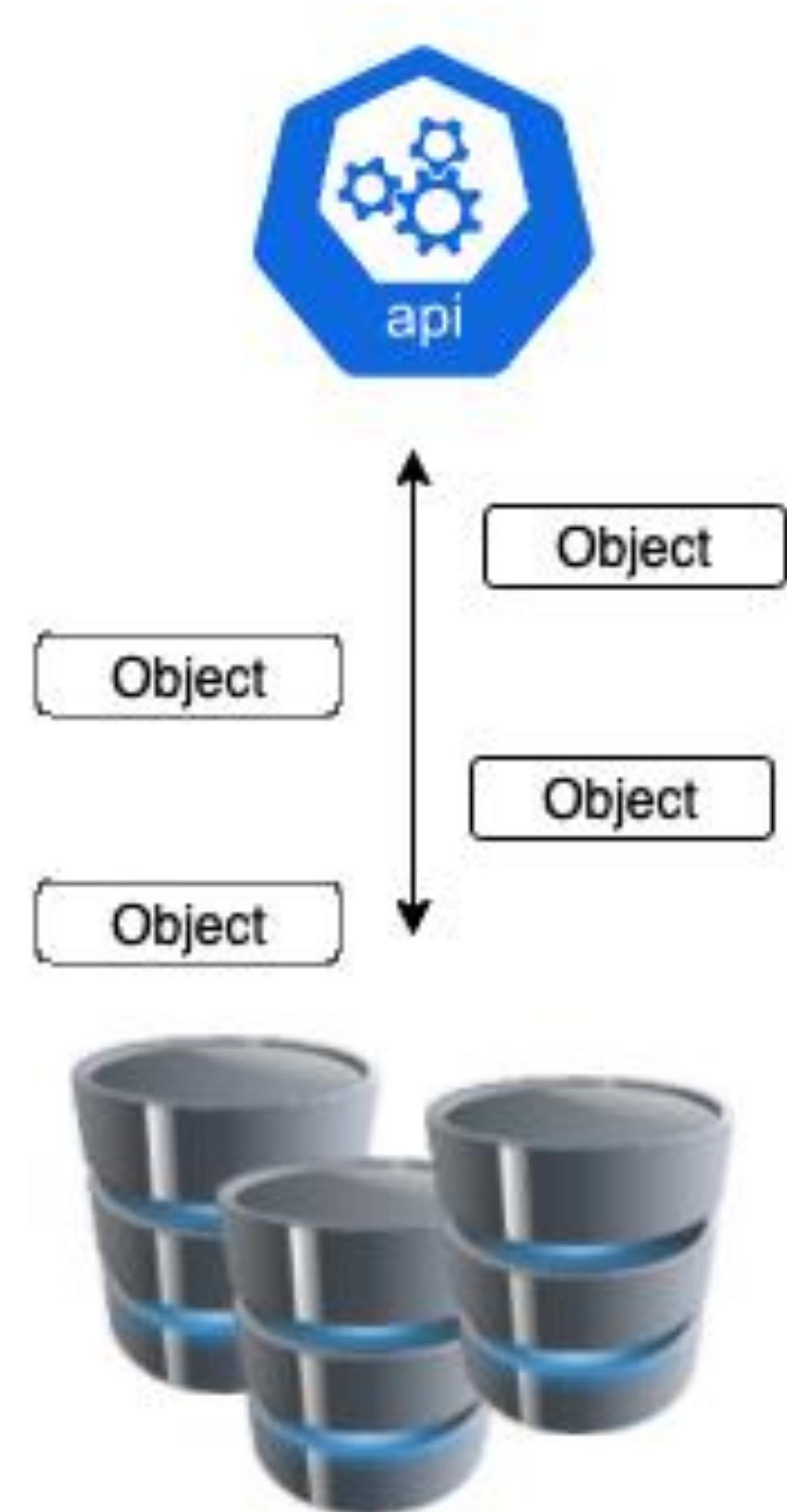
◆ TRI NAJČEŠĆA SCENARIJA

- Direktna komunikacija klijent <-> server kroz poziv servisa
- Serverska aplikacija <-> Baza podataka
- Asinhrona komunikacija razmenom poruka preko reda poruka (message queue)



◆ ŠTA JE CILJ?

- Aplikacije treba trajno da skladište podatke
- Ti podaci mogu biti pohranjeni u DMBS, tekstualne fajlove, itd.
- Ako razvijamo aplikacije u OOP maniru, hoćemo da čuvamo i dobavljamo te podatke kao objekte



◆ ŠTA JE PROBLEM SA RELACIONIM BAZAMA PODATAKA?

- DBMS su (pretežno) orijentisane ka čuvanju podataka u redovima u tabelama (a to ne liči na objekte)
- Asocijacije i kolekcije možemo da simuliramo korišćenjem veza između redova u tabelama
- Postoje konceptualne razlike u paradigmama poznate pod nazivom Object-Relational Impedance Mismatch¹

¹ Object-relational impedance mismatch https://en.wikipedia.org/wiki/Object-relational_impedance_mismatch

OOP VS. SQL

Aspekt	Objektni model (OOP)	Relacioni model (SQL)	Mismatch / Problem
Osnovna jedinica	Objekat (stanje + ponašanje)	Red u tabeli	Objekat ima metode, red nema
Struktura podataka	Ugnježđeni objekti, kolekcije	Tabele, kolone, normalizacija	Ugnježđeni objekat → više tabela + JOIN
Identitet	Referenca u memoriji	Primarni ključ (ID)	Mora se mapirati referenca → ID; tracking problemi
Odnosi	Reference i kolekcije (obj.Prop)	Strani ključevi, JOIN	N+1 problem, veliki JOIN-ovi, lazy/eager problemi
Nasleđivanje	Hijerarhije klasa	Nema nasleđivanja	Strategije mapiranja su sve kompromisi
Polimorfizam	virtual / override	Ne postoji	Teško mapiranje polimorfnih tipova na tabele

¹ Object-relational impedance mismatch https://en.wikipedia.org/wiki/Object%E2%80%93relational_impedance_mismatch

OOP VS. SQL

Aspekt	Objektni model (OOP)	Relacioni model (SQL)	Mismatch / Problem
Ponašanje	Logika u objektima (metode)	Baza čuva samo podatke	Teško čuvati domenske objekte bogate logikom
Mutabilnost	Menjaju se polja objekta pojedinačno	SQL radi nad skupovima (UPDATE SET...)	OOP „po jedan objekat“ vs SQL „set-based“
Graf objekata	Duboke reference	Više tabela, JOIN	Kada i koliko duboko učitavati? Lazy/eager
Transakcije	Nisu prirodni deo OOP-a	BEGIN/COMMIT	ORM mora da kontroliše kada se upisuje u bazu
Integritet	Validacija u kodu	Ograničenja (FK, CHECK)	Pravila se dupliraju ili razilaze
Lifecycle	Objekat živi u memoriji	Red postoji trajno	Attached/detached, concurrency, state tracking
Migracije	Refaktorisanje klasa	ALTER TABLE, migracije	Svaka promena modela traži migraciju baze

¹ Object-relational impedance mismatch https://en.wikipedia.org/wiki/Object%E2%80%93relational_impedance_mismatch

◆ **KAKO NAM ORM MOŽE POMOĆI U OSTVARIVANJU CILJA?**

- Radi konverziju objekata u format koji je u skladu sa API-em za slanje na čuvanje DBMS-u
- Pravi upite ka DBMS-u i kreira od rezultata objekte
- Čuva i rekreira veze između objekata

◆ PREDNOSTI

- Omogućena je konverzija iz bogatog skupa tipova podataka podržanih u OOP jezicima u skromniji skup tipova podataka podržanih od strane DBMS
- Pisanje ispravnih i optimizovanih upita je jednostavnije i brže
- Kod je lakši za ažuriranje, održavanje i ponovno korišćenje jer se sve vreme u aplikaciji koriste objekti
- Konvertuju upite u preporučeni format za specifični DBMS smanjujući prostor za grešku ili napad (SQL Injection)
- nude koncept "apstrakcije baze" kojim se omogućava laka zamena DBMS-a koji aplikacija koristi

◆ MANE

- Manje kontrole nad kodom jer “neko drugi radi posao za nas”
- Upiti u koje se konvertuje kod mogu biti spori ako se ne poštuju preporuke i principi koje ORM nameće
- Kompleksniji upiti se moraju pisati u nativnom SQL
- Mogu se smatrati primerom apstrakcija koje imaju manjkavosti (*leaky abstractions*¹)

¹ Leaky abstractions <https://www.joelonsoftware.com/2002/11/11/the-law-of-leaky-abstractions/>

◆ KLASA

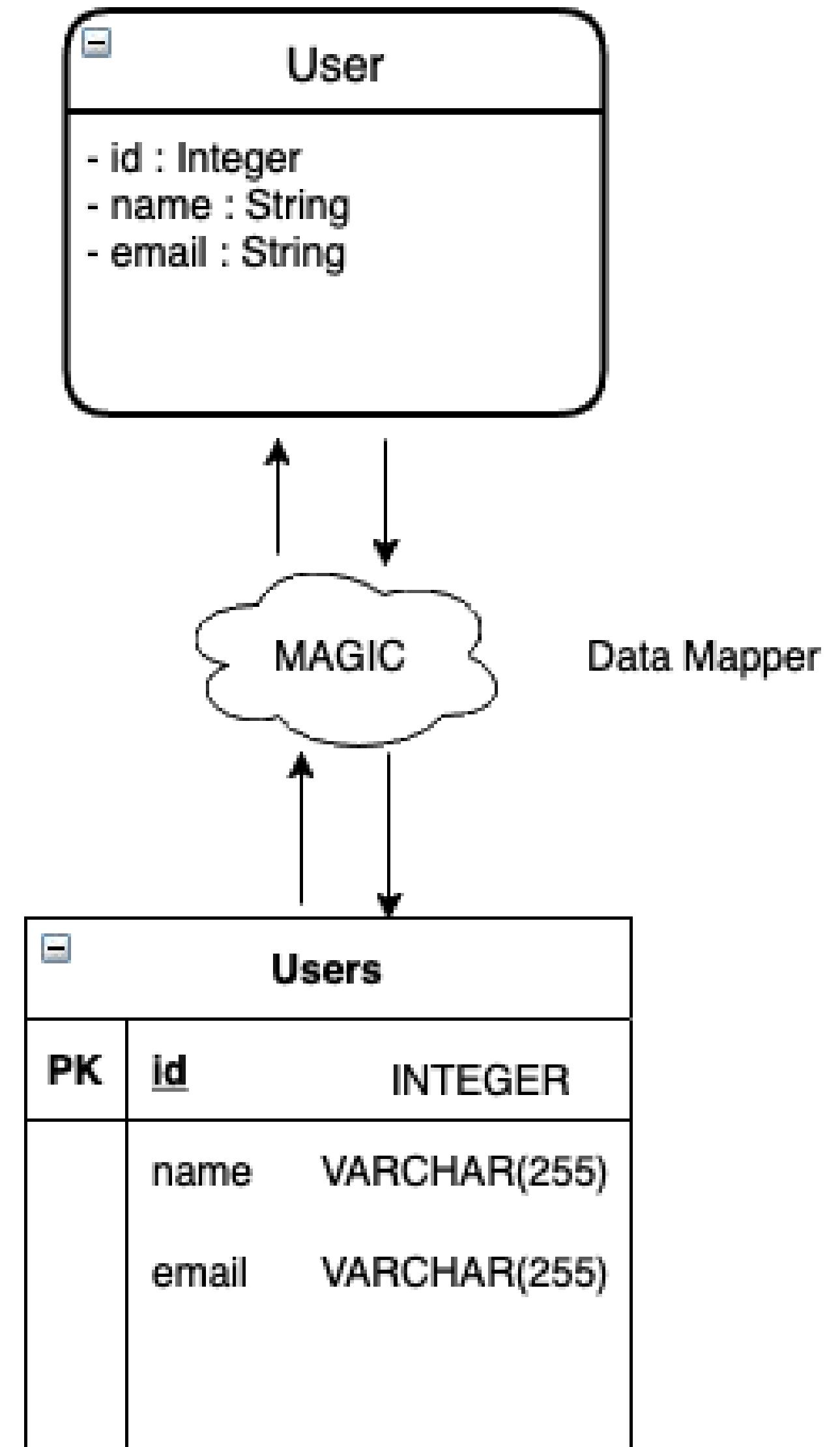
- Trebala bi da ima jedinstveni identifikator kao atribut

◆ DATA MAPPER

- Konvertuje objekat u red tabele, konvertuje tipove podataka, instancira objekte

◆ TABELA

- Ima jedinstveni identifikator koji je obično primarni ključ



◆ TABLE DATA GATEWAY¹ JE ŠABLON KOJI:

- Predstavlja jednu tabelu u bazi i sadrži CRUD operacije za tu tabelu
- Karakteristike:
 - Jedna klasa = jedna tabela
 - Metode: Find, Insert, Update, Delete
 - Nema poslovne logike – samo pristup podacima
 - Radi sa skupovima podataka (DataTable, DTO, record)

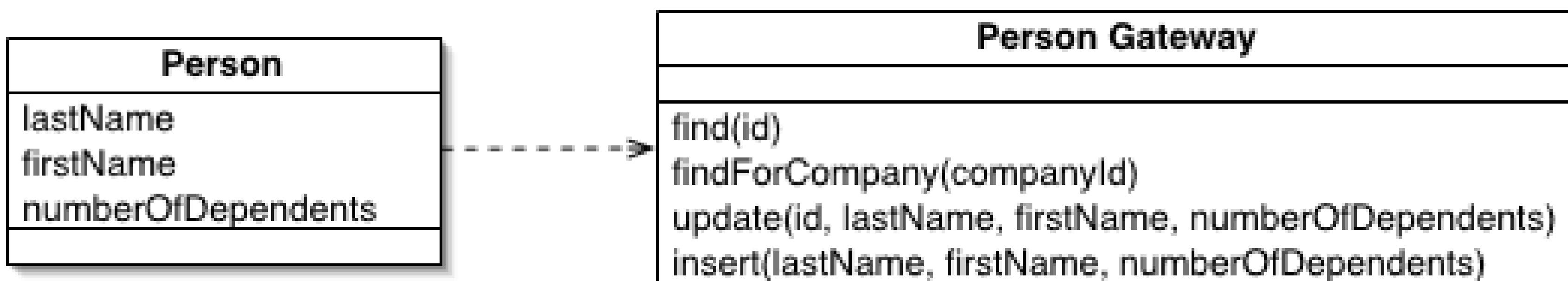


TABLE DATA GATEWAY

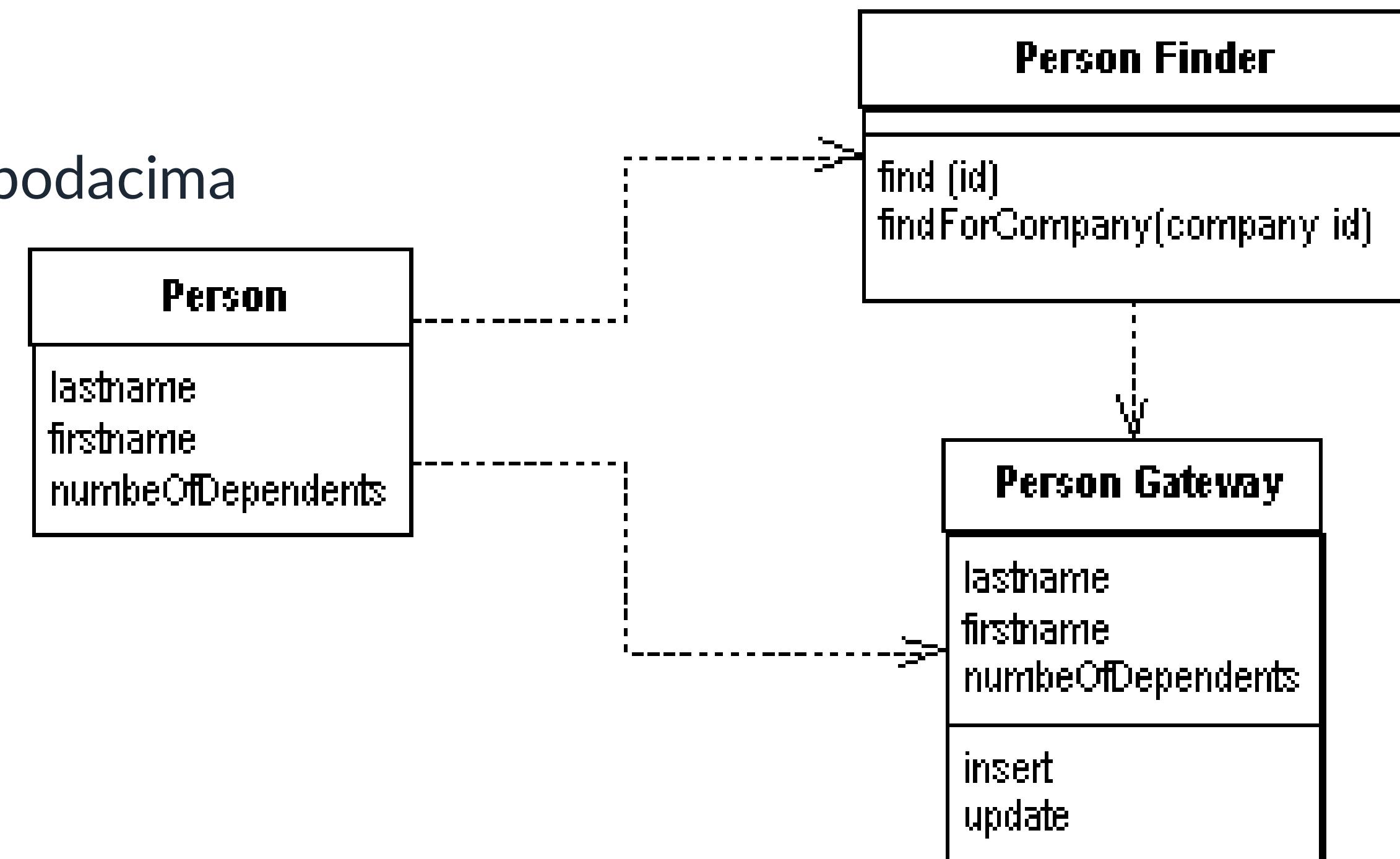
◆ TABLE DATA GATEWAY¹ JE ŠABLON KOJI:

- Prednosti:
 - Jednostavno
 - Centrizovan SQL
 - Jasna separacija pristupa bazi
- Mane:
 - Nema objekata domena
 - Logika pristupa podacima nije OO
 - Kompleksni upiti postaju teški za održavanje

```
class PersonGateway {  
    Person find(int id) { ... }  
    void insert(int id,...) { ... }  
    void update(int id,...) { ... }  
}
```

◆ ROW DATA GATEWAY¹ JE ŠABLON GDE:

- Svaki objekat predstavlja jedan red iz tabele, i sadrži operacije za rad nad sopstvenim podacima
- Karakteristike:
 - Instanca = jedan red u tabeli
 - Metode za: Load, Update, Delete
 - Još uvek nije domenski model – fokus je na podacima



¹ Data Source Architectural Patterns <https://martinfowler.com/eaaCatalog/>

◆ ROW DATA GATEWAY¹ JE ŠABLON KOJI:

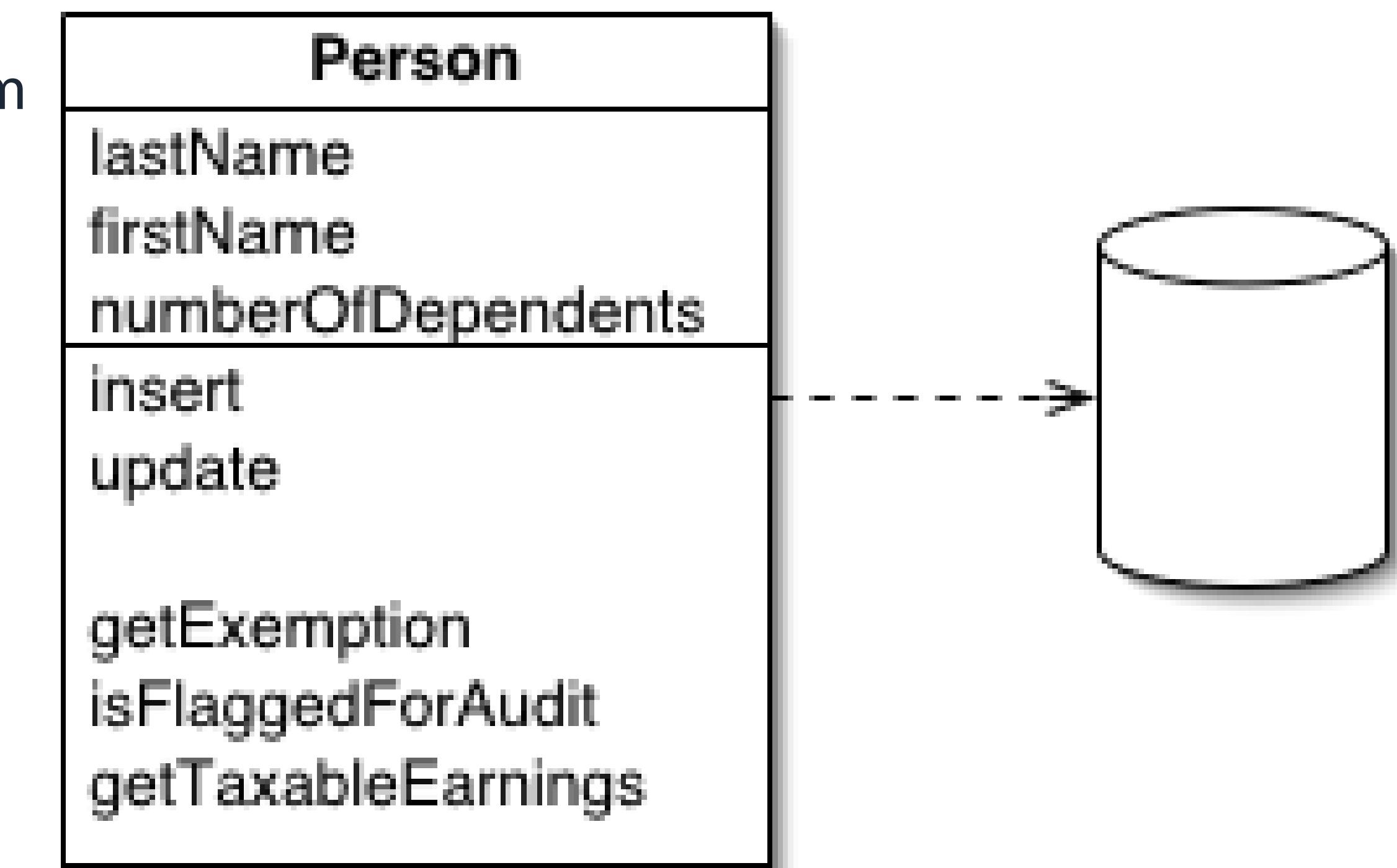
- Prednosti:
 - Objektno „pakovanje“ jednog reda
 - Jednostavan za razumeti
 - Dobar za jednostavne CRUD aplikacije
- Mane:
 - Poslovna logika ostaje van objekta
 - Teško radi sa kompleksnim relacijama
 - Previše gateway objekata → šum u arhitekturi

```
class PersonRow {  
    int id;  
    string name;  
    void insert() { ... }  
    void update() { ... }  
    void delete() { ... }  
}
```

ACTIVE RECORD

◆ ACTIVE RECORD¹ JE ŠABLON GDE:

- Objekat domena = red u bazi + CRUD metode
- Sadrži i poslovnu logiku i SQL logiku (malo podseća na MVC stil)
- Karakteristike:
 - Klasa ima polja domena i operacije nad bazom
 - Save(), Delete(), Find() su u samoj klasi
 - Koriste ga Ruby on Rails, Laravel Eloquent



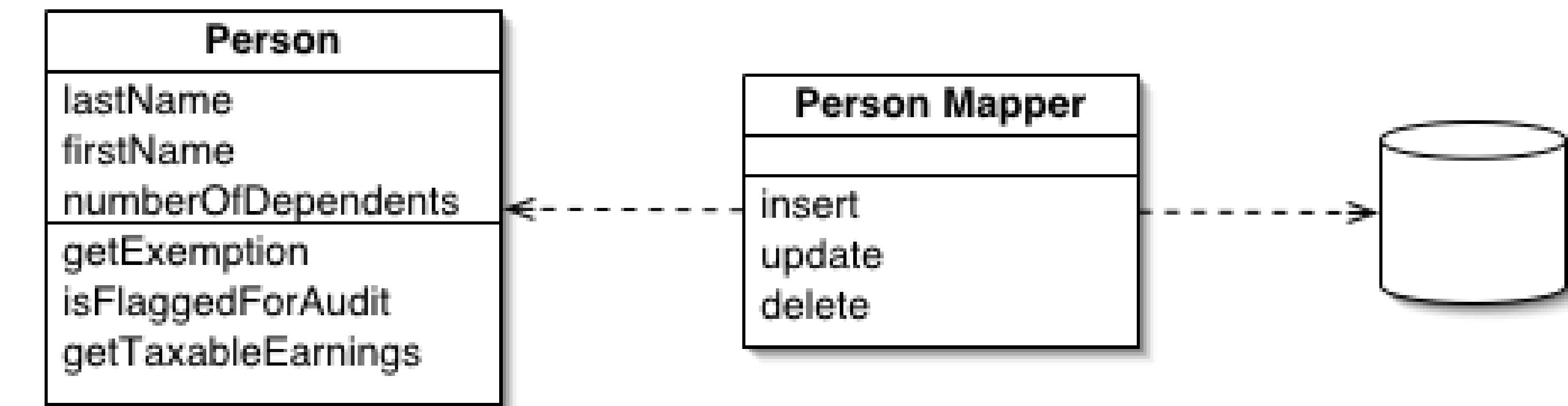
◆ ACTIVE RECORD¹ JE ŠABLON GDE:

- Prednosti:
 - Veoma jednostavan za male sisteme
 - Brz razvoj
 - Malo koda i direktna veza model-tabela
- Mane:
 - Meša domenski sloj i perzistenciju
 - Teško testiranje
 - Ne skalira dobro u kompleksnim domenima

```
class Person {  
    int id;  
    string name;  
    void save() { ... }  
    static Person find(int id) { ... }  
}
```

◆ DATA MAPPER¹ JE ŠABLON GDE:

- Najčistiji pristup – domenski objekti znaju samo za domensku logiku, a mapper ih prevodi u tabele i nazad
- Karakteristike:
 - Domenski model je potpuno čist (POJO)
 - Klase nemaju SQL metode
 - Mapper ili ORM radi prevodenje objekat ↔ tabela
 - Koriste ga EF Core, Hibernate, TypeORM



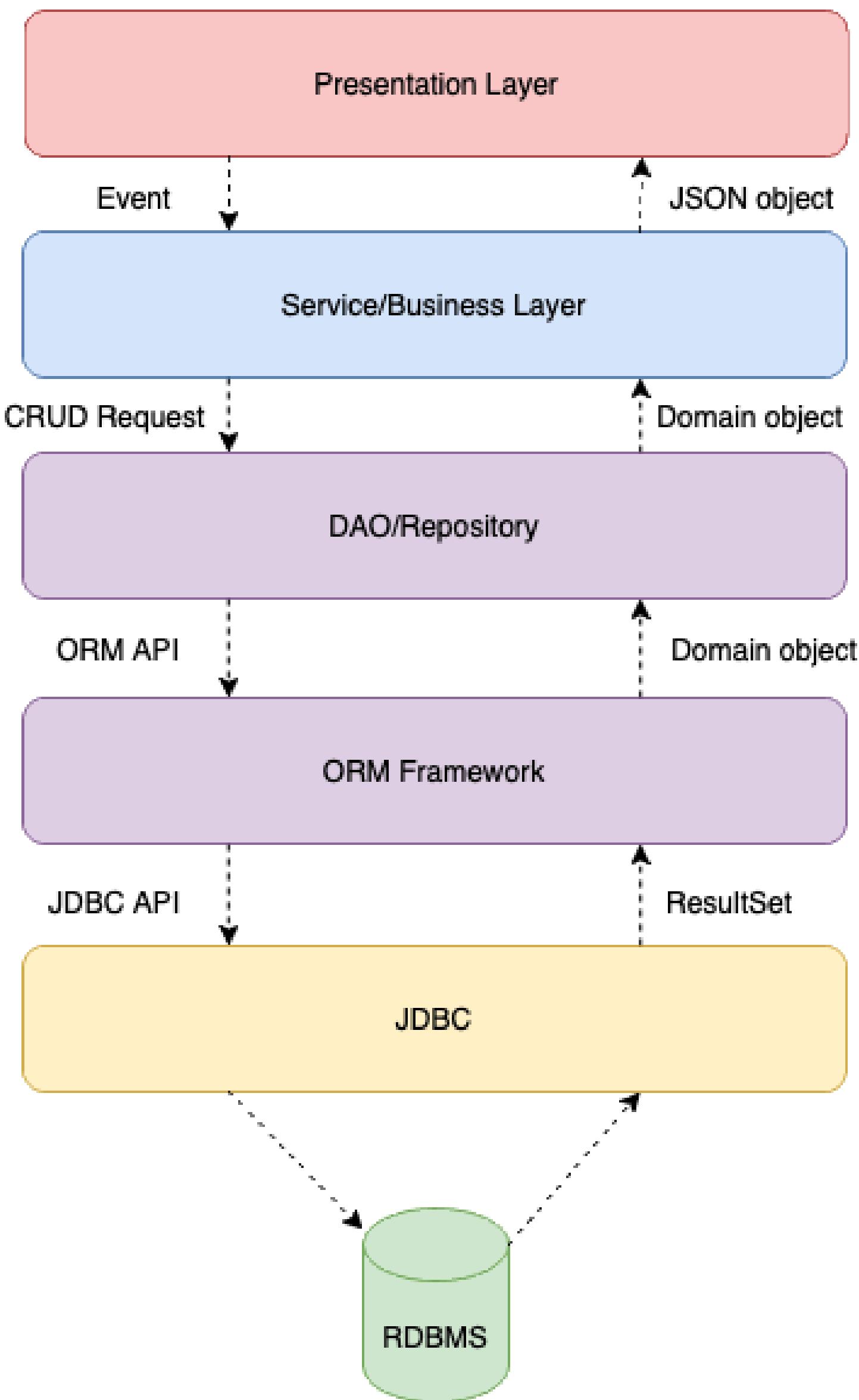
◆ DATA MAPPER¹ JE ŠABLON GDE:

- Prednosti:
 - Čist domen (DDD)
 - Jasna separacija slojeva
 - Najbolje rešenje za kompleksne sisteme
- Mane:
 - Složenije za implementirati
 - ORM može biti magičan i težak za kontrolisati
 - Potrebno razumevanje mapiranja i transakcija

```
class Person { int id; string name; }

class PersonMapper {
    Person load(int id) { ... }
    void save(Person p) { ... }
}
```

ORM I ARHITEKTURA APLIKACIJE





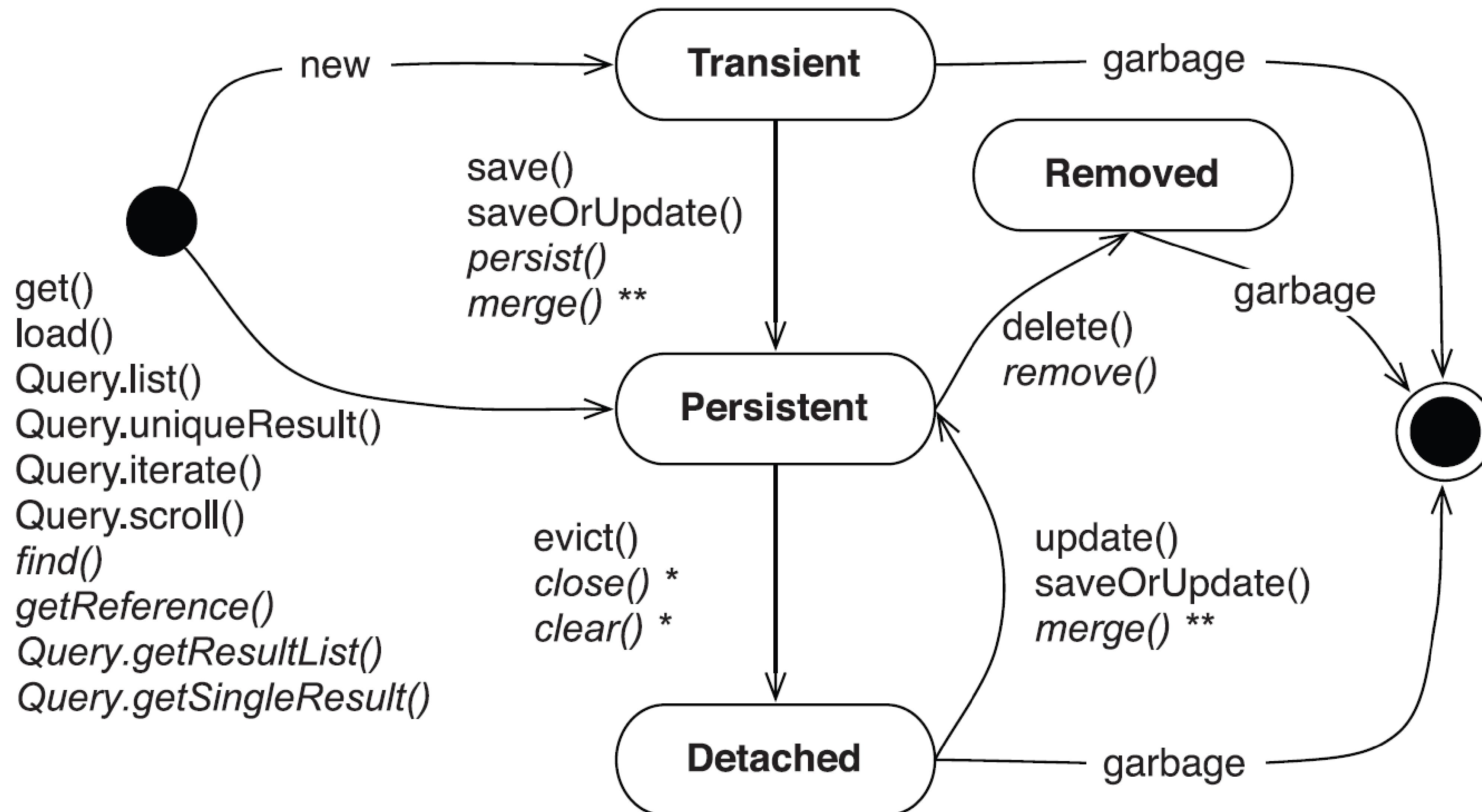
JAKARTA PERSISTENCE API (JPA)

23

◆ ŠTA JE JPA?

- Java standardna specifikacija za upravljanje relacionim podacima u enterprise Java aplikacijama (praktično specifikacija za ORM)
- Ranije poznatija kao Java Persistence API
- Entitet je objekat koji se čuva u bazi
- Koriste se anotacije da se označe različite komponente koda kako bi se odradilo mapiranje na DMBS
- Postoje različite implementacije JPA
 - Hibernate (najpopularniji, JBoss)
 - EclipseLink (referentna implementacija, Oracle)
 - ObjectDB
 - ...

JPA BEANS: ŽIVOTNI CIKLUS



REFERENCE

- ◆ PRIMERI PO UZORU NA <https://github.com/mbranko/isa19/tree/master/07-orm>
- ◆ FOWLER M. ORMHATE <https://martinfowler.com/bliki/OrmHate.html>
- ◆ DREWSKY, HIGHSCALABILITY.COM. THE CASE AGAINST ORM FRAMEWORKS IN HIGH SCALABILITY ARCHITECTURES
<http://highscalability.com/blog/2008/2/2/the-case-against-orm-frameworks-in-high-scalability-architec.html>
- ◆ IRELAND C. ET AL. A CLASSIFICATION OF OBJECT-RELATIONAL IMPEDANCE MISMATCH. FIRST INTERNATIONAL CONFERENCE ON ADVANCES IN DATABASES, KNOWLEDGE, AND DATA APPLICATIONS. <https://ieeexplore.ieee.org/document/5071809>
- ◆ TORRES A. ET AL. TWENTY YEARS OF OBJECT-RELATIONAL MAPPING: A SURVEY ON PATTERNS, SOLUTIONS, AND THEIR IMPLICATIONS ON APPLICATION DESIGN
<https://www.sciencedirect.com/science/article/abs/pii/S0950584916301859>
- ◆ CHEN T. P. ET AL. DETECTING PERFORMANCE ANTI-PATTERNS FOR APPLICATIONS DEVELOPED USING OBJECT-RELATIONAL MAPPING
<https://bit.ly/2Pbh6hx>
- ◆ FOWLER M. PATTERNS OF ENTERPRISE APPLICATION ARCHITECTURE
<https://martinfowler.com/eaaCatalog/>

**KOJA SU VAŠA
PITANJA?**