CQRS meets modern Java

Simon Martinelli
@simas_ch
martinelli.ch

About Me

Web

https://martinelli.ch

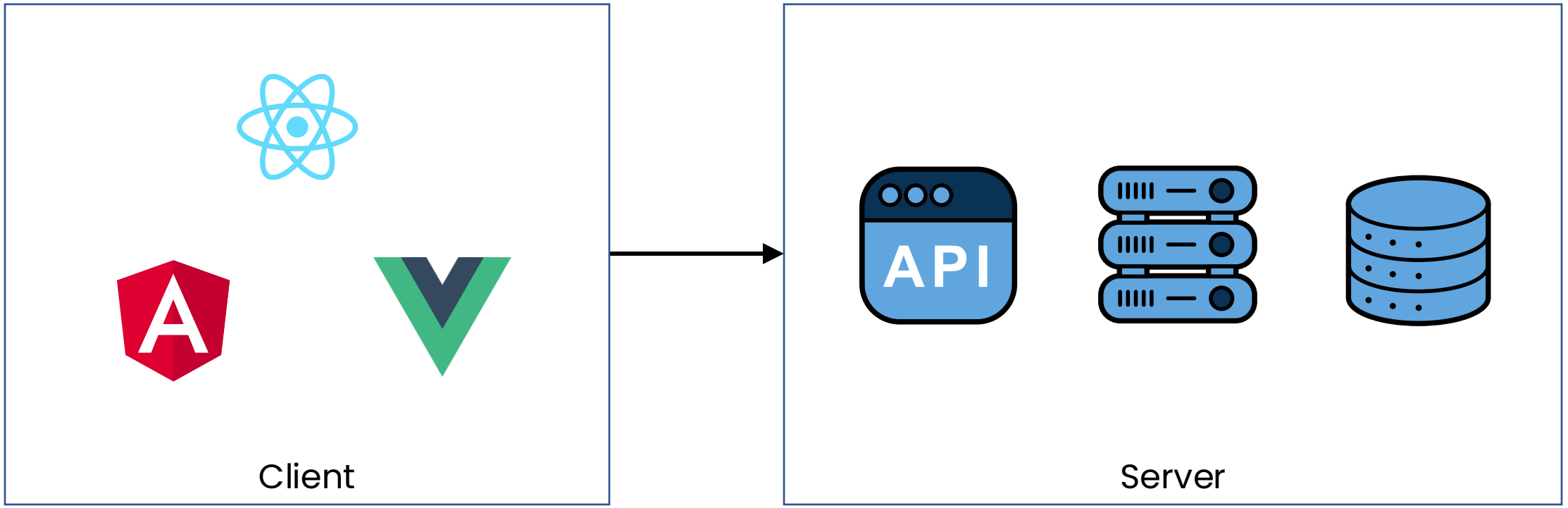
E-Mail

simon@martinelli.ch

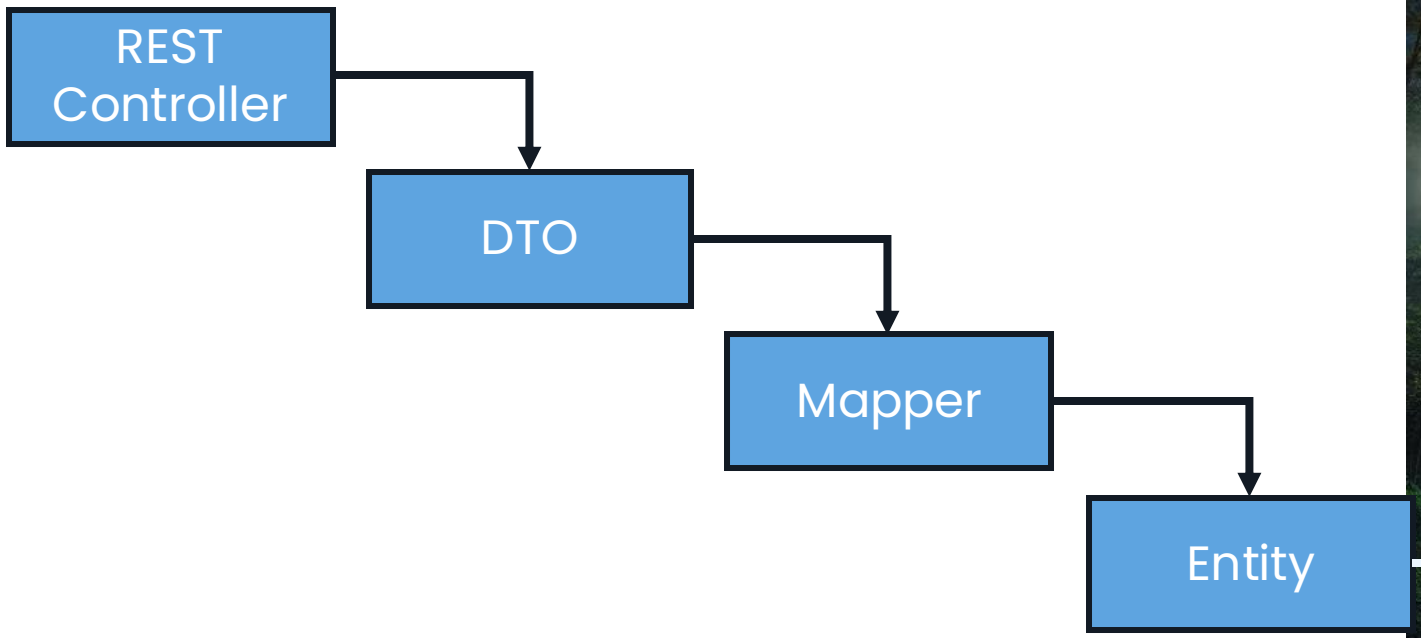
X/Twitter

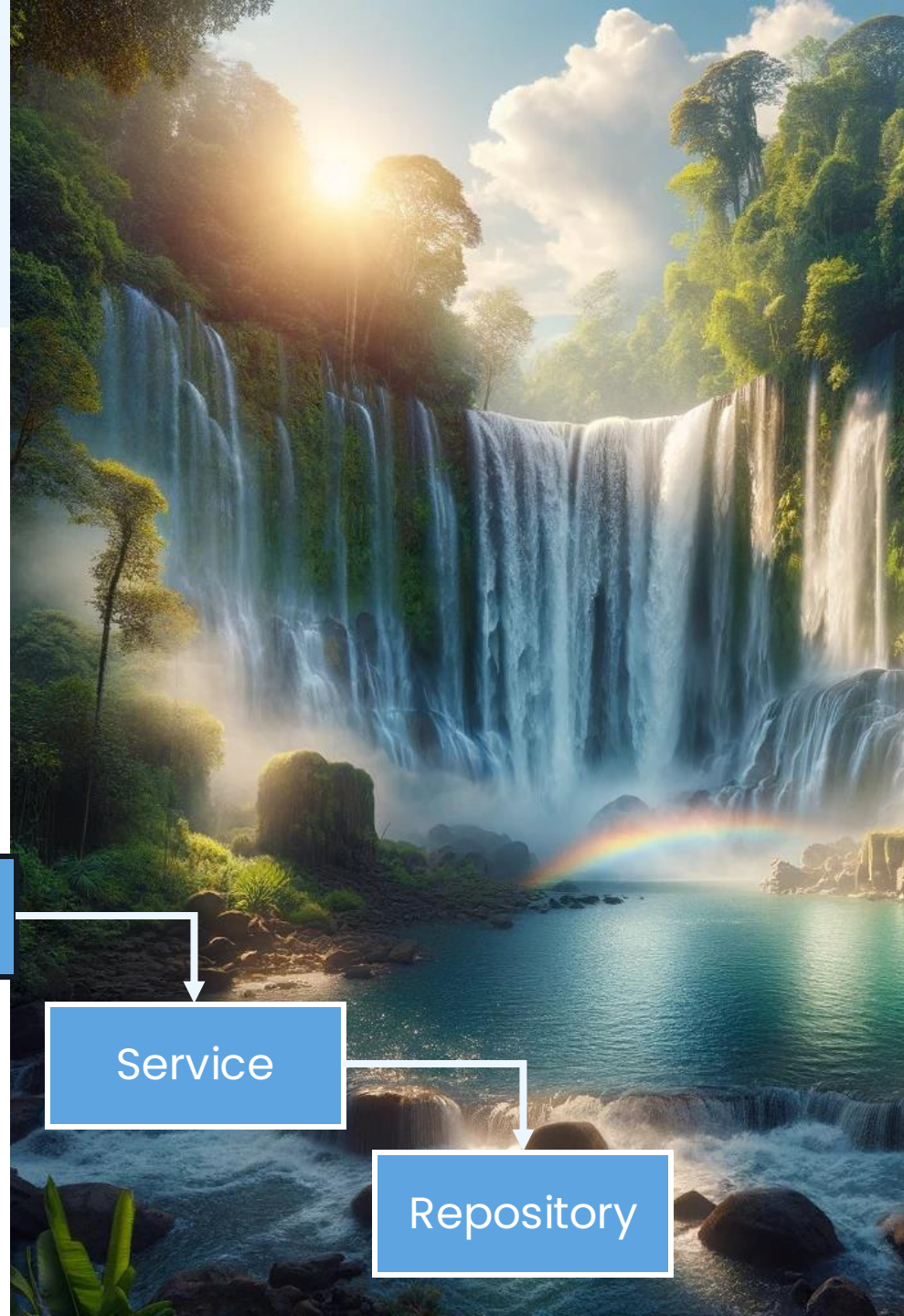
@simas_ch

Client/Server Communication



Mapping Waterfall





Example Project

- https://github.com/simasch/cqrs-meets-modern-java

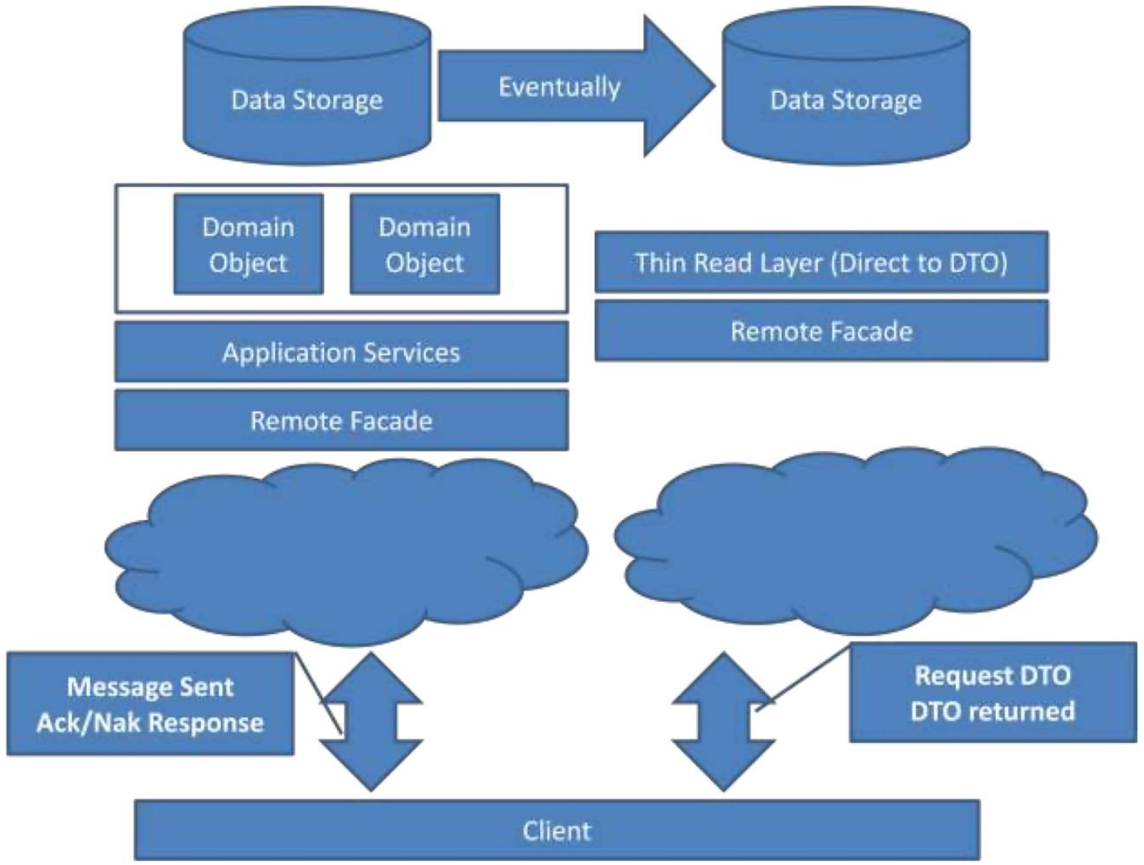
Issue 1: Too Much

- Over fetching
  - Loading data that is not needed
- Not all data can be changed
  - Too much data is sent in the update request
  - Potential bugs if blindly mapped

Issue 2: Too Little

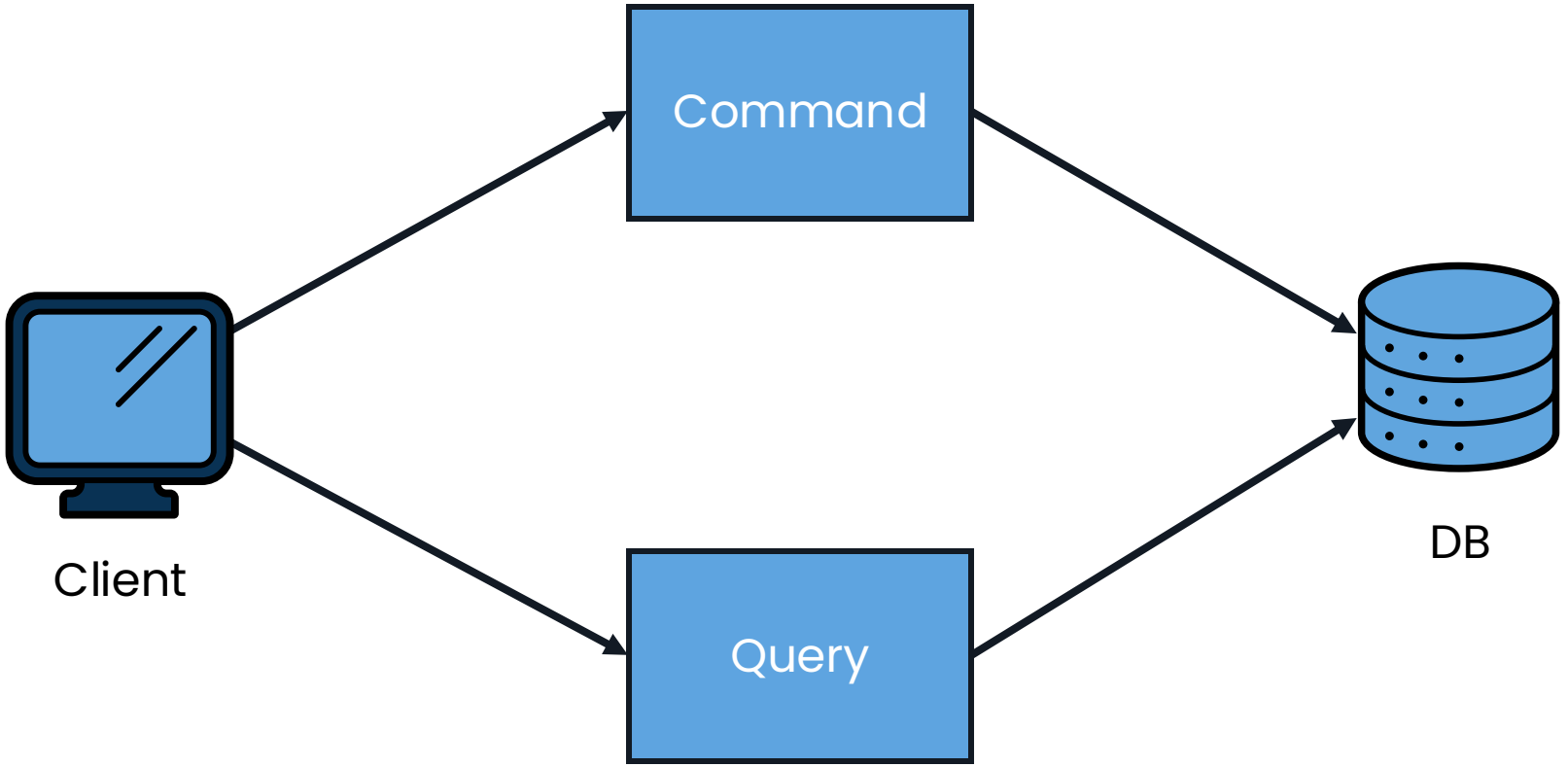
- Under fetching
  - Results in multiple requests
  - Produces the N+1 select problem on the client-side
- Under storing
  - Updates/inserts not within the same transaction

CQRS 😱

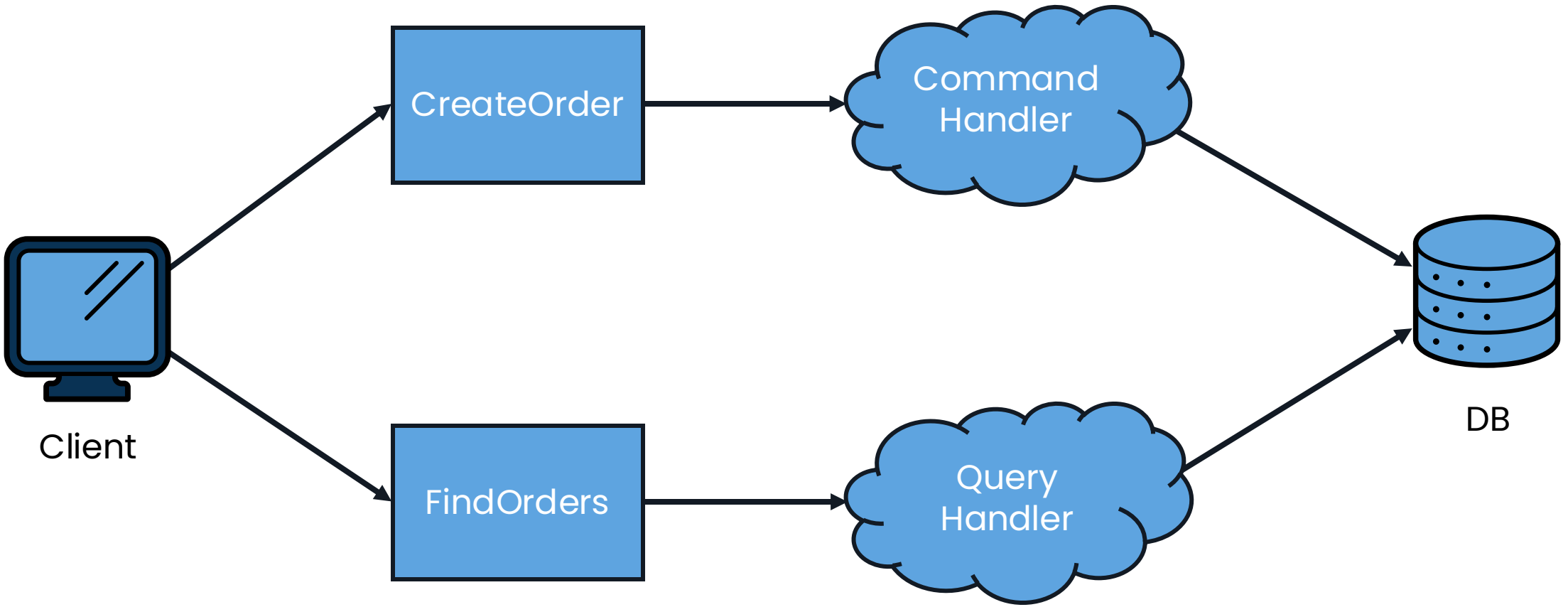


Source: https://cqrs.wordpress.com/wp-content/uploads/2010/11/cqrs_documents.pdf

Start Simple



Example


CreateOrder
Command Handler
Client
FindOrders
Query Handler
DB

Modern Java: Records

Preview: 14
Final: 16

- Provide a compact syntax for declaring classes as transparent holders for **immutable data**

```
record AddItemCommand(long orderId,
                      long productId,
                      int  quantity) {
}
```

Modern Java: Sealed Classes

Preview: 15
Final: 17

- Sealed classes and interfaces restrict which other classes or interfaces may extend or implement them

```
abstract sealed class Shape
        permits Circle, Rectangle, Square {
}
```

Modern Java: Switch Expression

Preview: 12
Final: 14

- Exhaustive: the switch expression must cover all cases
- Preview in Java 12 and finalized in Java 14

```
String result = switch (color) {
    case RED -> "Red";
    case GREEN -> "Green";
    case BLUE -> "Blue";
};
```

Modern Java: Pattern Matching

Preview: 14/17
Final: 17/21

- `instanceof` operator
- `switch` expression and statement

```
double area = switch (shape) {
    case Circle c -> Math.PI * c.radius * c.radius;
    case Rectangle r -> r.length * r.width;
    default -> throw new IllegalArgumentException();
};
```

Modern Java: Record Patterns

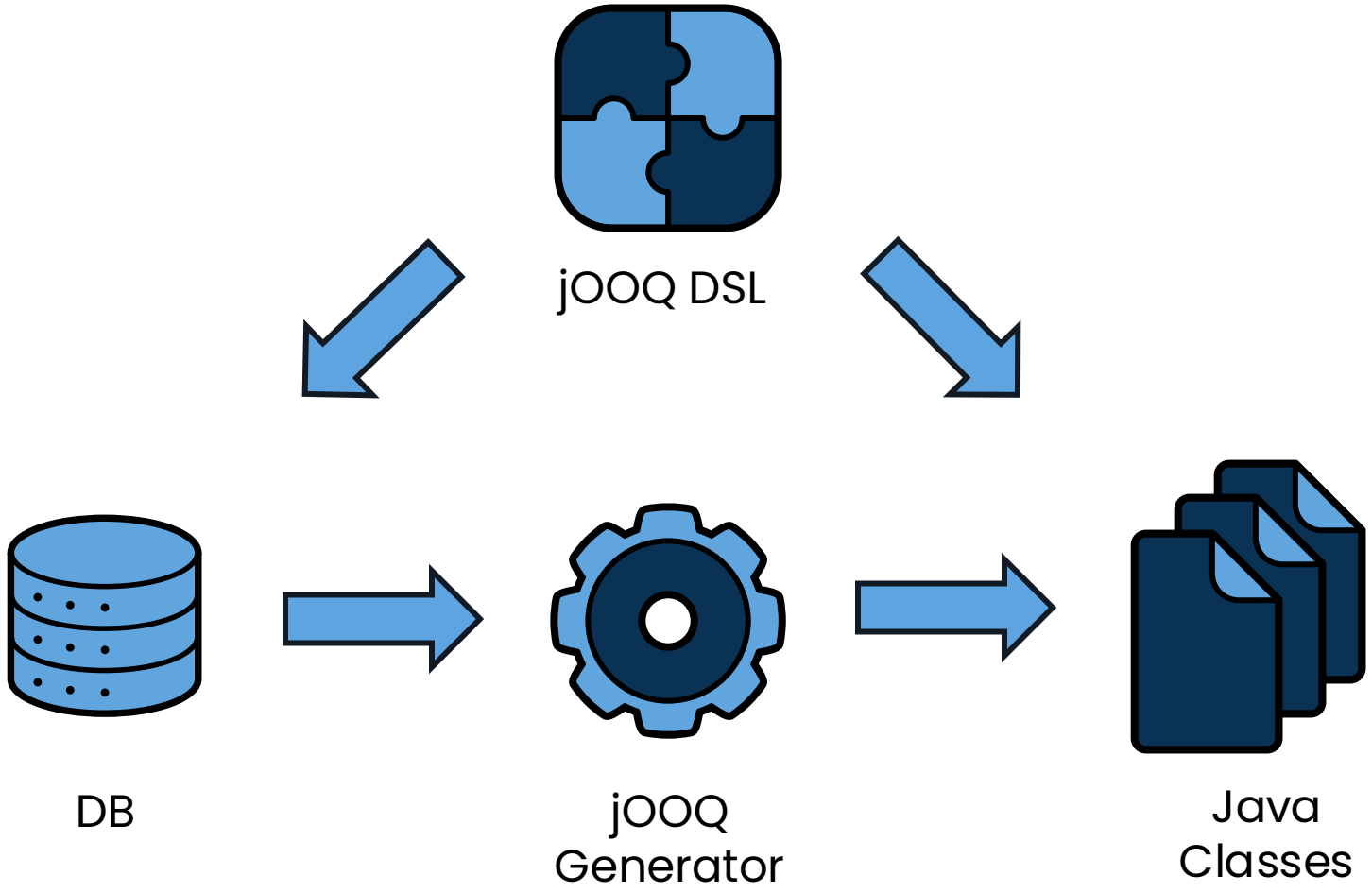
Preview: 20
Final: 21

- Use to test whether a value is an instance of a record class type and, if it is, recursively perform pattern matching on its component values

```
record Pair(int left, int right) {}

if (p instanceof Pair(int left, int right)) {
   ...
}
```

What is jOOQ?



Why jOOQ?

Commands

- Usually, insert, update, or delete data in the database
- Use jOOQ to generate minimal updates

Queries

- Often return nested structures like Order → Item
- jOOQ provides the MULTISET value constructor


Demo time
what could go wrong

Conclusion

The separation of commands and queries

- Improves the understandability
- Helps to avoid over- and under-fetching
- Can make mapping Entity <-> DTO obsolete
- Enables an event-driven approach

Thank you!

Web

https://martinelli.ch

E-Mail

simon@martinelli.ch

X/Twitter

@simas_ch