trainitek

How Many Sides Does Your Architecture Have?

Vadzim Prudnikau

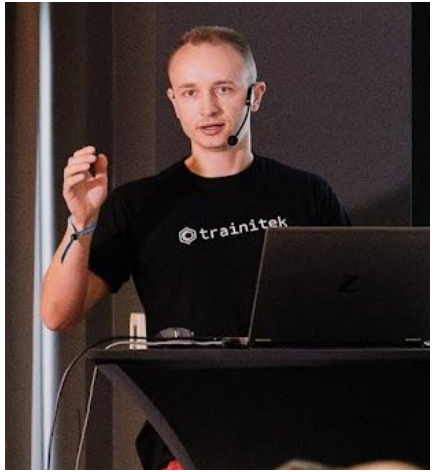
trainitek

THANK YOU!!!

trainitek

Shortly about me

I run training in Architecture, DDD, EventStorming, and TDD



Co-owner and instructor at trainitek

Hands-on architect at APOTEK 1

Vadzim Prudnikau

vadim-prudnikov

trainitek

What will we do today?

- Understand what Hexagonal Architecture is
- Look at some code
- Understand what Hexagonal Architecture is not
- Reveal the most important: why it has 6 sides :)

trainitek

Original idea

https://alistair.cockburn.us/hexagonal-architecture/

“Allow an application to equally be driven by users, programs, automated test or batch scripts, and to be developed and tested in isolation from its eventual run-time devices and databases.”

trainitek

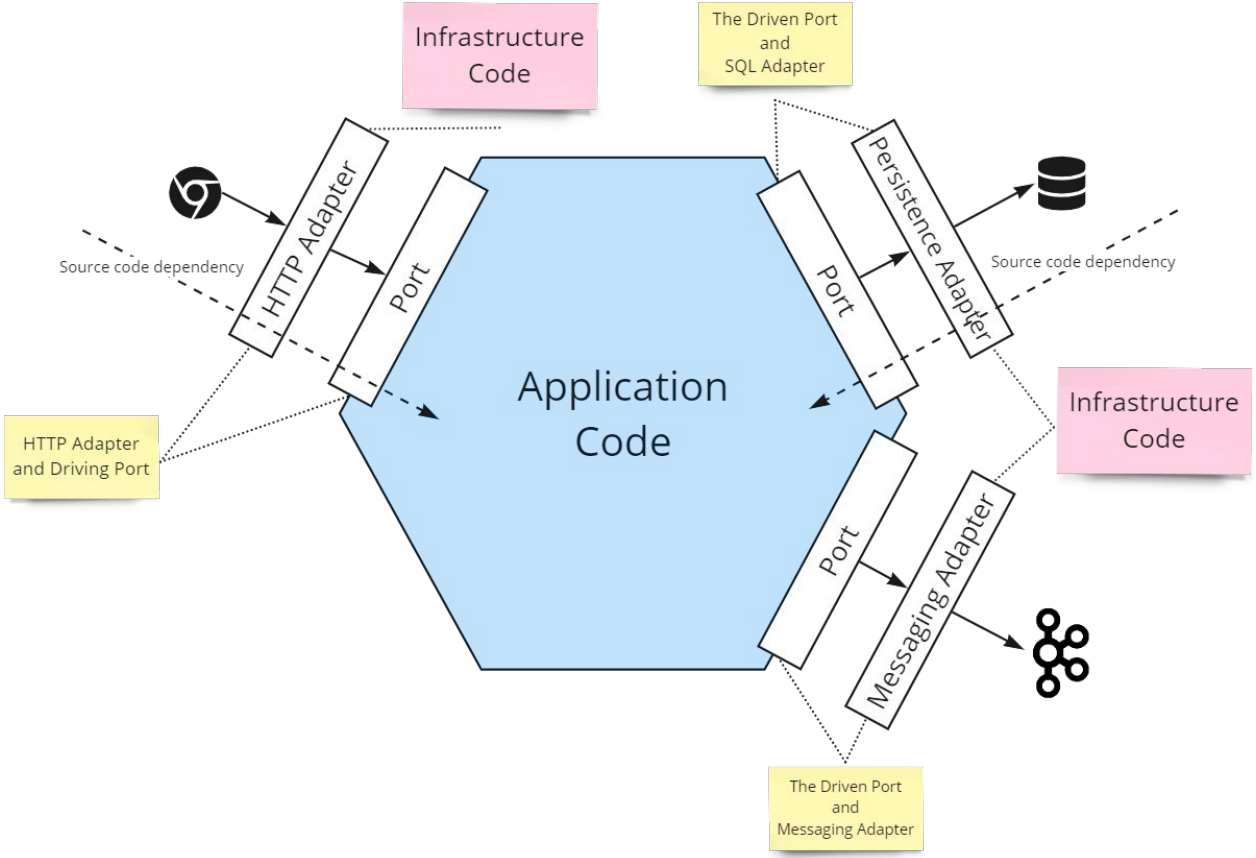
Original idea - unpacked

Main idea: Split application logic and infrastructure code

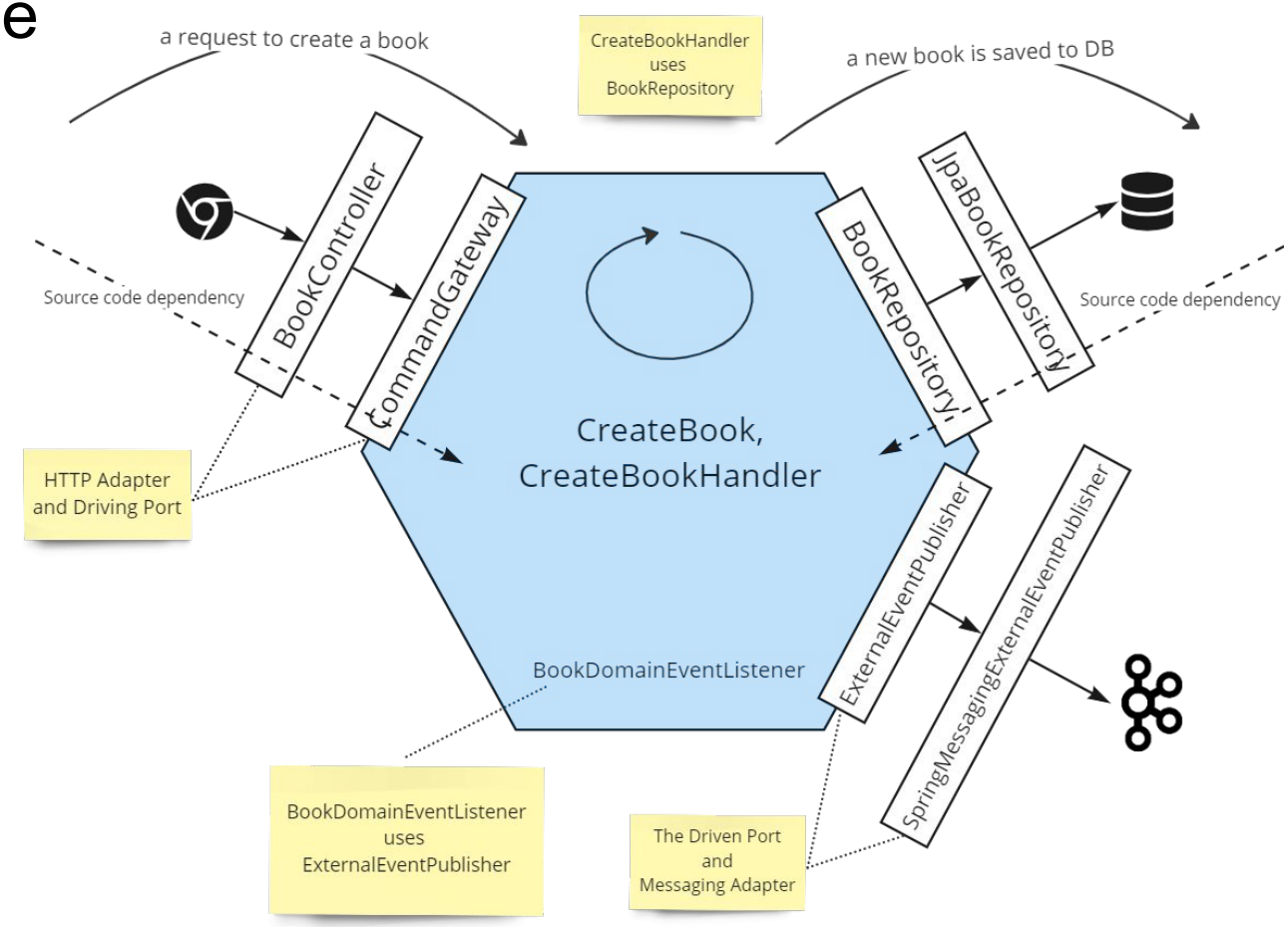
Why:

- It leads to stronger separation of concerns
- It’s easier to replace the "infrastructure"
- It’s easier to test (which also leads to a better design)
- It’s easier to tune

Conceptual picture



Example


a request to create a book
CreateBookHandler uses BookRepository
a new book is saved to DB
BookController
CommandGateway
BookRepository
JpaBookRepository
Source code dependency
Source code dependency
CreateBook,
CreateBookHandler
HTTP Adapter and Driving Port
ExternalEventPublisher
SpringMessagingExternalEventPublisher
BookDomainEventListener
BookDomainEventListener uses ExternalEventPublisher
The Driven Port and Messaging Adapter

What does the code look like?

trainitek

BookController - an HTTP adapter

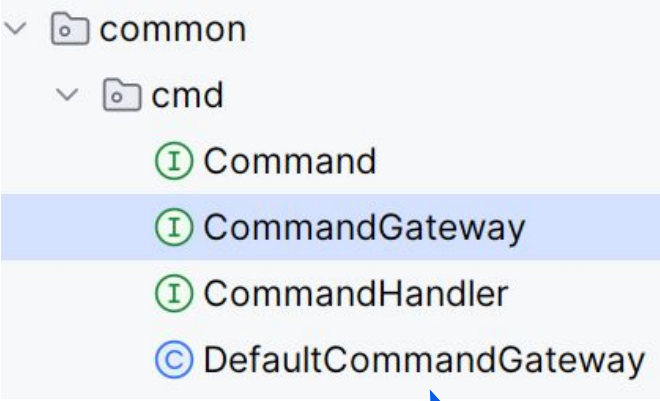
- book
  - application
  - domain
  - infrastructure
    - persistence
    - web
      - BookController
      - CreateBookRequest

```
public class BookController {

    private final CommandGateway commandGateway;

    @PostMapping    Vadzim Prudnikau
    public ResponseEntity<String> create(@RequestBody CreateBookRequest request) {
        var command = new CreateBook(request.title(), request.author());

        var id = commandGateway.dispatch(command);

        return ResponseEntity.created(URI.create("/api/books/%s".formatted(id))).build();
    }
}
```

Port

trainitek

CommandGateway - a driving port




```
public interface CommandGateway {

    <R> R dispatch(Command<R> cmd);

}
```

Finds a corresponding command handler

trainitek

CreateBookHandler - an implementation

- book
  - application
    - BookDomainEventListener
    - CreateBook
    - CreateBookHandler

```java
public class CreateBookHandler implements CommandHandler<CreateBook, UUID> {

    private final BookRepository repository;

    private final ApplicationEventPublisher eventPublisher;
    private final Clock clock;

    public UUID handle(CreateBook command) {
        var book = repository.save(new Book(command.title(), command.author()));

        eventPublisher.publishEvent(new BookCreated(book.getId(), clock));

        return book.getId();
    }
}
```

Port

trainitek

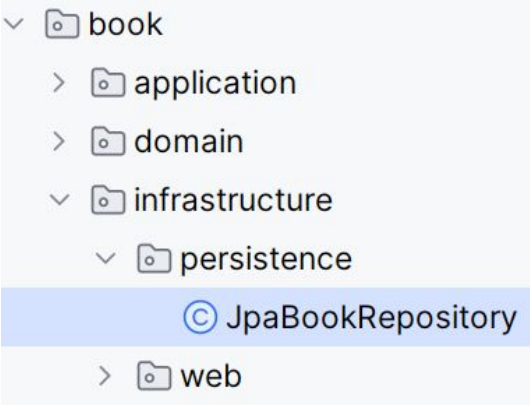
BookRepository - a driven port

- book
  - application
  - domain
    - Book
    - BookCreated
    - BookRepository

```java
public interface BookRepository {

    Book save(Book book);

    Optional<Book> find(UUID id);

    default Book load(UUID id) {
        return find(id).orElseThrow(() ->
                new EntityNotFoundException(Book.class, id));
    }
}
```

trainitek

JpaBookRepository - a driven adapter




```java
public class JpaBookRepository implements BookRepository {

    private final EntityManager entityManager;

    @Override
    public Book save(Book book) {
        entityManager.persist(book);
        return book;
    }

    @Override
    public Optional<Book> find(UUID id) {
        return Optional.ofNullable(entityManager.find(Book.class, id));
    }
}
```

trainitek

BookDomainEventListener - publishes external events

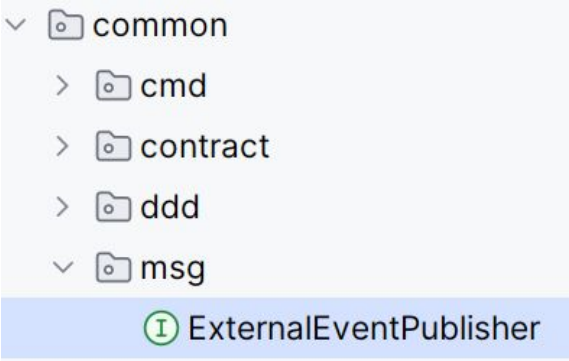
- book
  - application
    - BookDomainEventListener
    - CreateBook
    - CreateBookHandler
  - domain
  - infrastructure
- common
- infrastructure.msg
- integration
  - BookCreatedEvent

```
public class BookDomainEventListener {

    private final BookRepository bookRepository;
    private final ExternalEventPublisher eventPublisher;

    @EventListener
    public void on(BookCreated event) {
        var book = bookRepository.load(event.bookId());
        eventPublisher.publish(
                new BookCreatedEvent(
                        randomUUID(),
                        book.getId(),
                        book.getTitle(),
                        book.getAuthor()));
    }
}
```

Port

trainitek

ExternalEventPublisher - a driven port




```
public interface ExternalEventPublisher {

    <T> void publish(@NonNull T event);
}
```

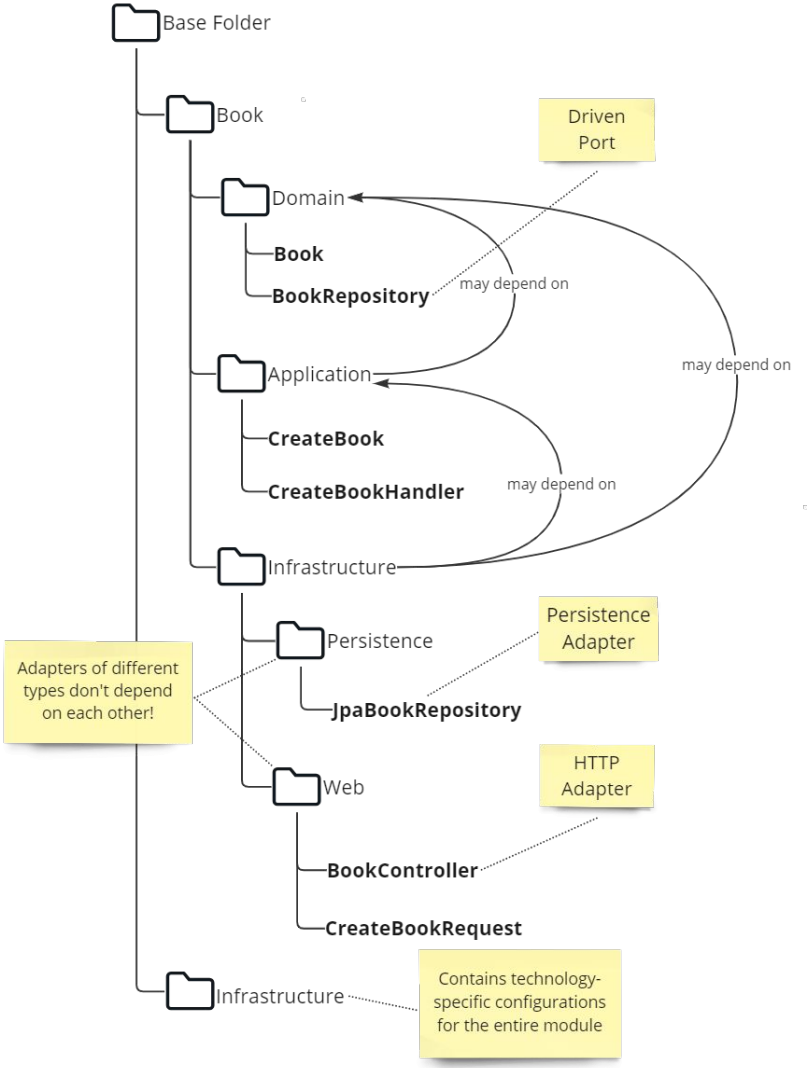
trainitek

SpringMes…EventPublisher - a driven adapter

- book
- common
- infrastructure.msg
  - SpringMessagingExternalEventPublisher

```
public class SpringMessagingExternalEventPublisher implements ExternalEventPublisher {

    private final StreamBridge streamBridge;

    @Override
    public <T> void publish(@NonNull T event) {
        Message<?> message = toMessage(event);
        streamBridge.send( bindingName: "output", message);
        log.info(">>> External event {} published", message);
    }
```

Recap: folder structure

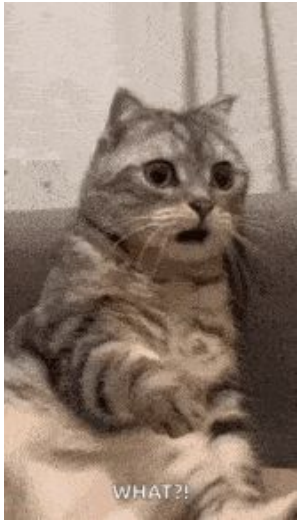


trainitek

trainitek

How to preserve the folder structure?

A. Double-check before committing! Pay attention while doing code review!



B. Use tools like ArchUnit, ArchUnitNet, ts-arch, pytestarch.

trainitek

Question: Isn’t it common sense?

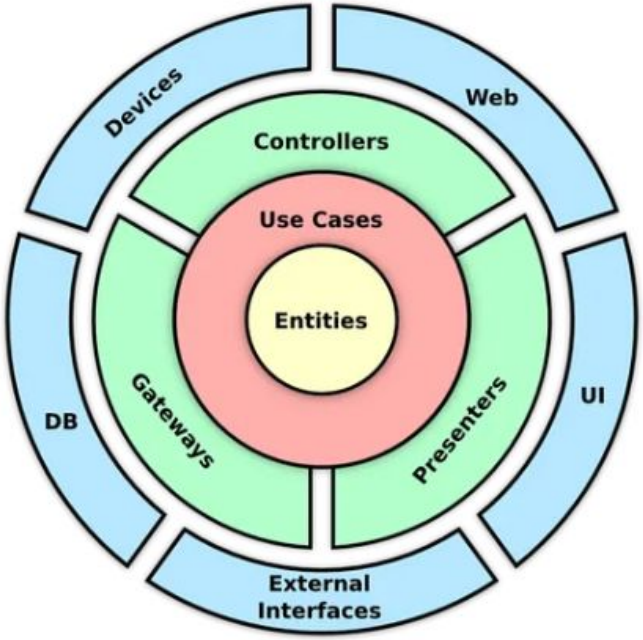
Yes, it is.

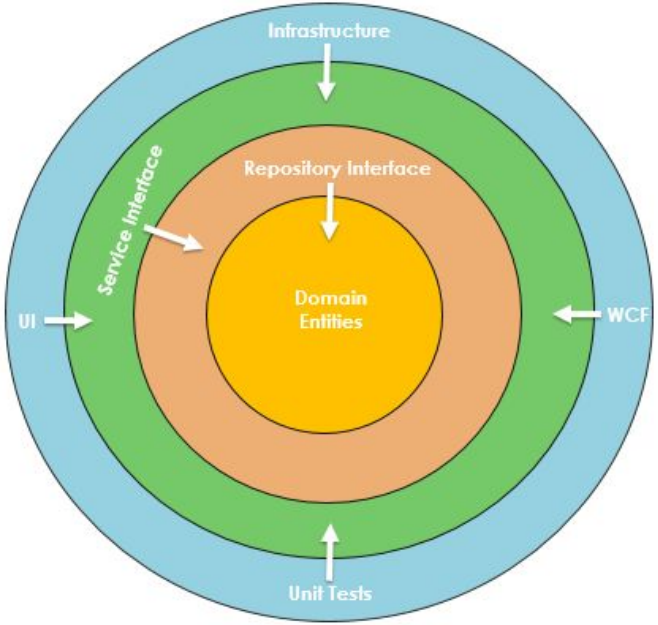
- Inheritance and polymorphism are fundamental principles of Object-Oriented Programming (OOP)
- Abstractions should not depend on their implementations

However, how often do you see that being ignored in real code? :)

trainitek

Wait! But we have Clean and Onion architectures…





trainitek

Wait! But we have Clean and Onion architectures…

- **Clean Architecture**: separation of business logic from frameworks
- **Onion Architecture**: dependency inversion with domain at the core
- **Hexagonal Architecture**: isolating application logic through ports and adapters

Aren’t they talking about the same thing? :)

trainitek

Why do I prefer discussing Hexagonal Architecture?

- It was the first one (2005) :)
- It has a very specific focus and memorizable concept (ports and adapters)
- It’s easier to explain it by showing the code
- It’s easier to see whether people understand it

trainitek

Do we always need Hexagonal Architecture?

No! It’s not always the recommended choice, for example:

- Simple CRUD
- Strict performance requirements
- Tight integration with third-party APIs
- …

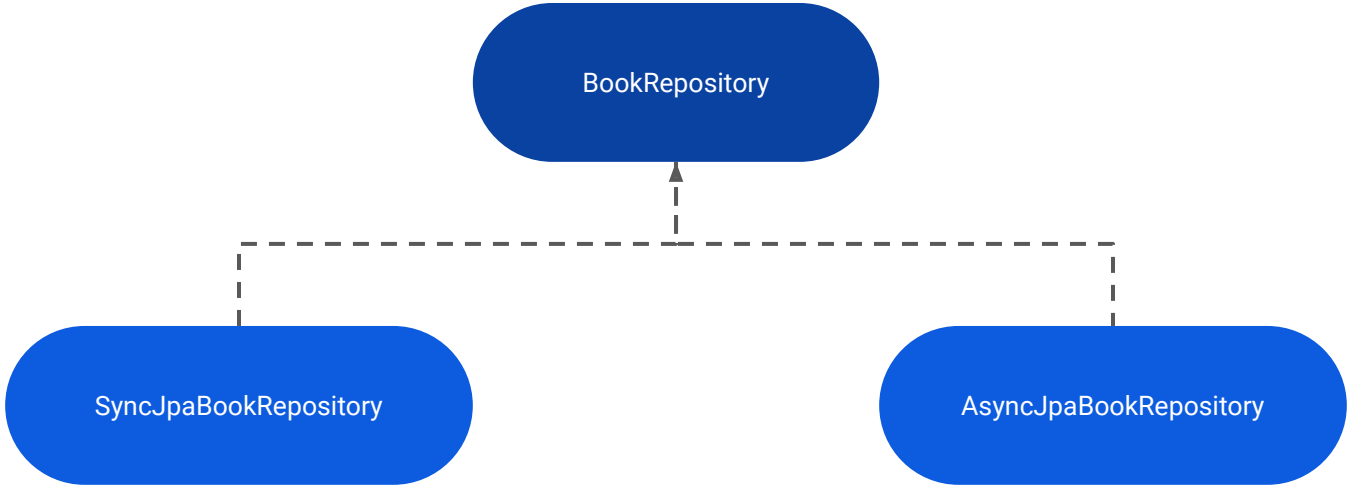
trainitek

Illusion #1

"I can easily switch between sync/async ways of communication with my infrastructure"

trainitek

Illusion #1: Sync/async adapters



What if you want to access the book right after saving it?

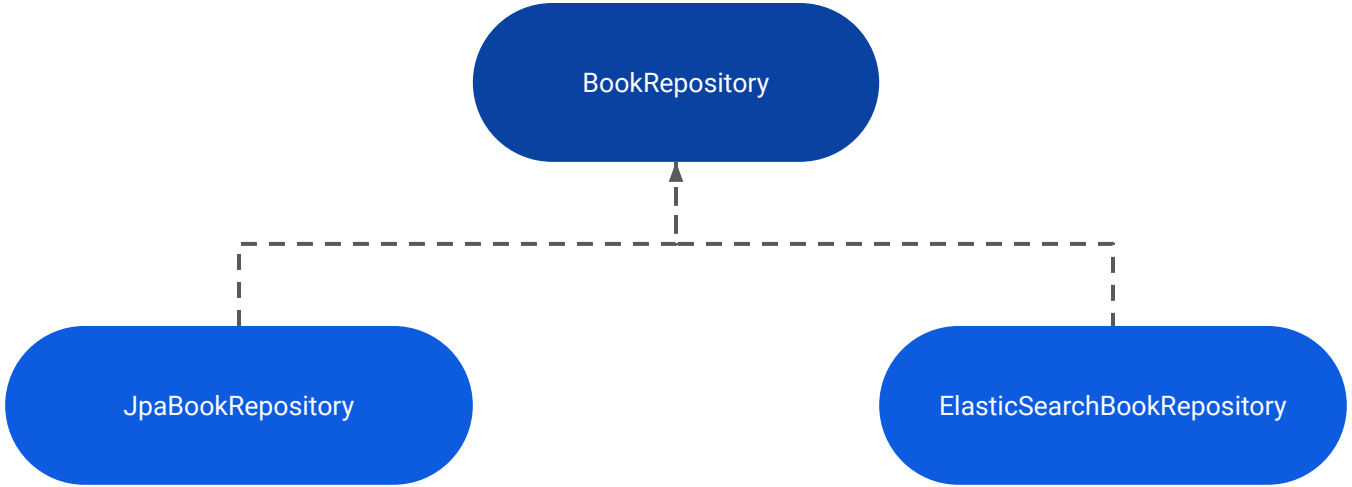
trainitek

Illusion #2

"I can easily switch to infrastructure with different ACID properties"

trainitek

Illusion #2: Adapters with different ACID properties



What if saving a book fails?

trainitek

Important question: Why does it have 6 sides?

I don’t know, however, I have some assumptions :)

- Boxes were already occupied
- Triangles and pentagons don’t have enough number of sides
- Heptagons and shapes with more sides is harder to draw

trainitek

Last Note

Remember: the main idea of Hexagonal Architecture is to separate application logic from infrastructure code, not to find a silver bullet for a universal architecture.

trainitek

Thank You!

https://trainitek.com/

vadim-prudnikov