trainitek

How Many Sides Does Your Architecture Have?

Vadzim Prudnikau


ProductDock


HTEC


trainitek


RT-RK

smart office


BAY
42


AMBSTUDIO


ElevateBits


LUKAS
NAKIĆ


supernova


BLC
20 godina inovacija


LEŽIBEG


avenga


Nektar Light

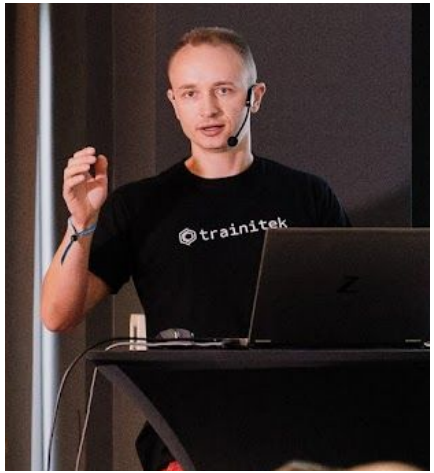

AVI Design


kuhinjica

trainitek

Shortly about me

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

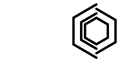


Co-owner and instructor at trainitek

Hands-on architect at APOTEK 1

Vadzim Prudnikau

vadim-prudnikov



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** :)

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.”

Original idea - unpacked

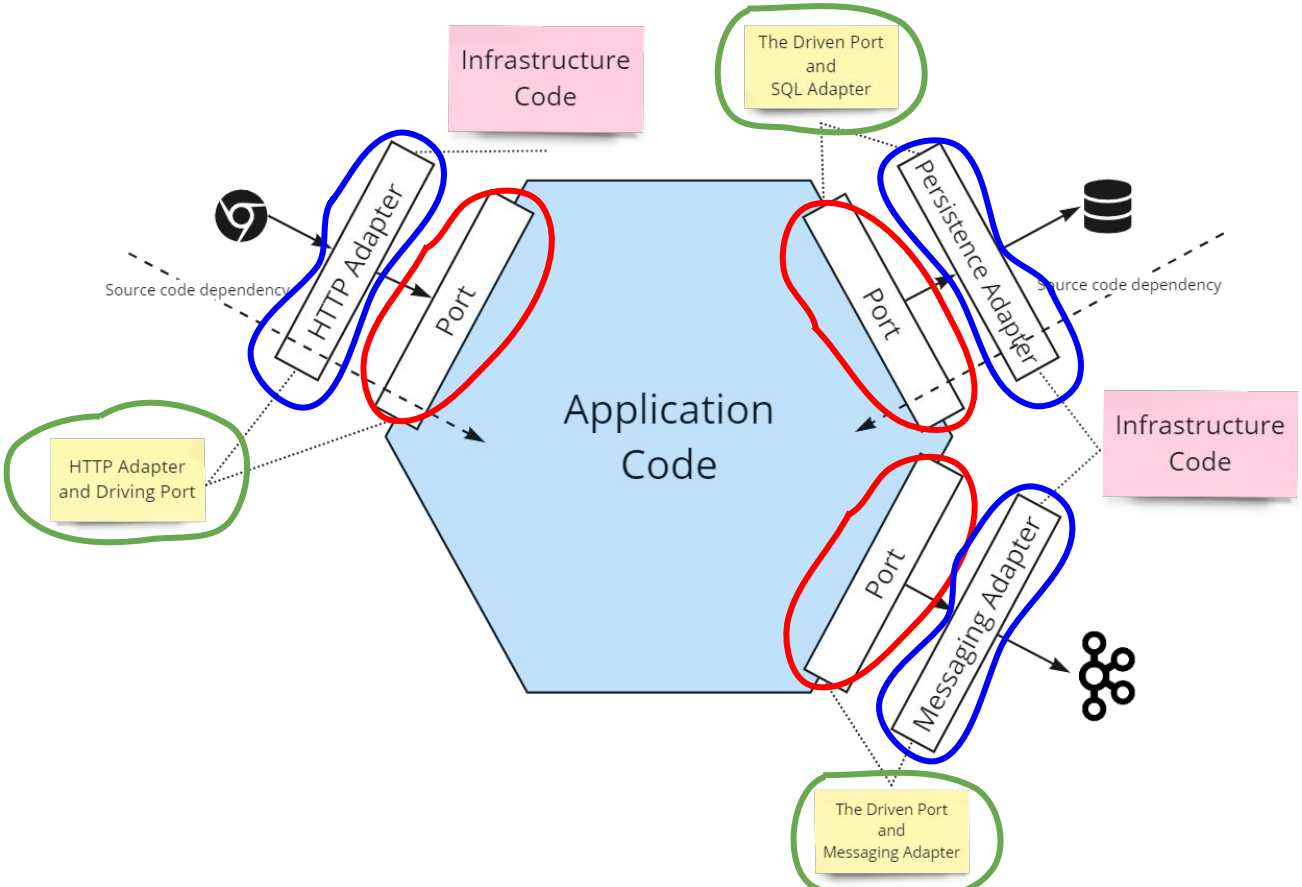
Main idea: **Split application logic and infrastructure code**

Why:

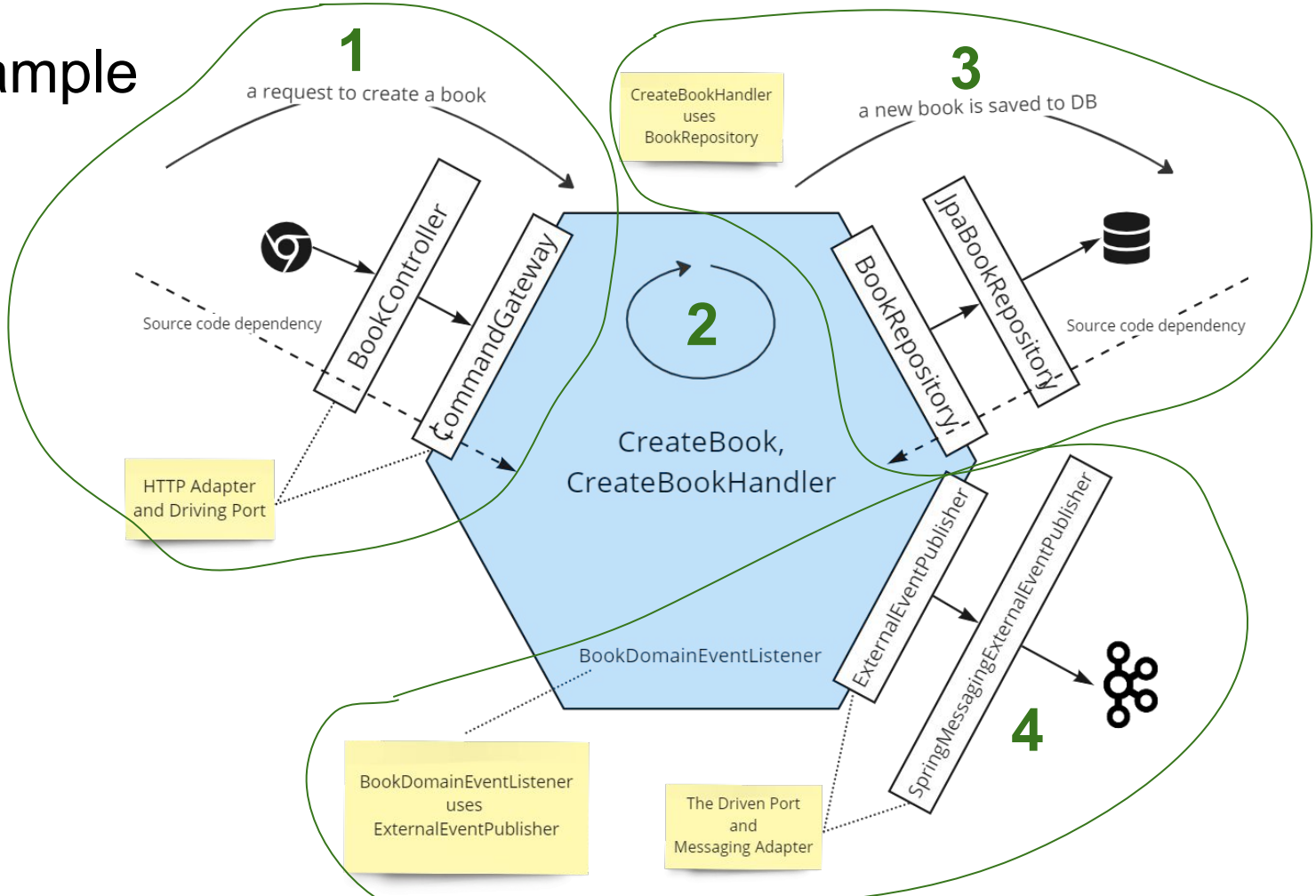
- It leads to stronger separation of concerns
- It’s easier to replace the "infrastructure"
- It’s easier to test (which also leads to a better design)
- It’s easier to tune

trainitek

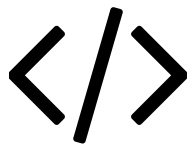
Conceptual picture



Example


1
a request to create a book
CreateBookHandler uses BookRepository
3
a new book is saved to DB
BookController
CommandGateway
2
BookRepository
JpaBookRepository
Source code dependency
Source code dependency
CreateBook,
CreateBookHandler
HTTP Adapter and Driving Port
BookDomainEventListener
ExternalEventPublisher
SpringMessagingExternalEventPublisher
4
BookDomainEventListener uses ExternalEventPublisher
The Driven Port and Messaging Adapter

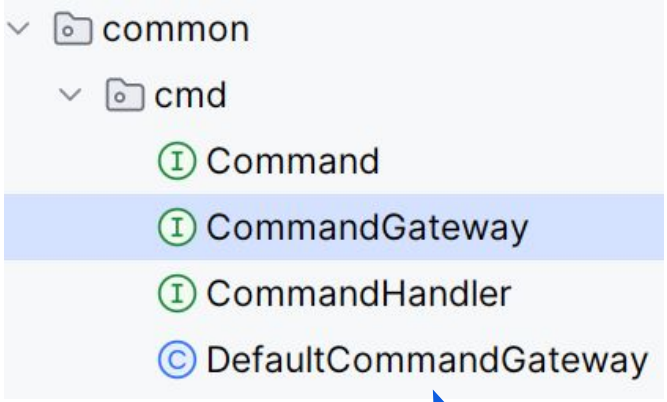
What does the code look like?



trainitek

BookController - an HTTP adapter

CommandGateway - a driving port

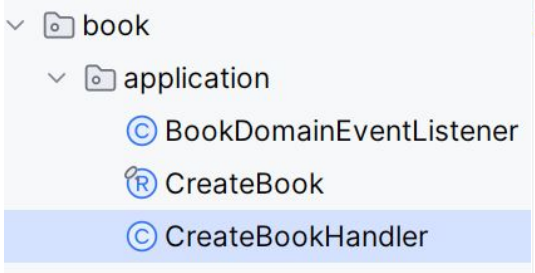



```
public interface CommandGateway {

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

}
```

Finds a corresponding command handler

trainitek

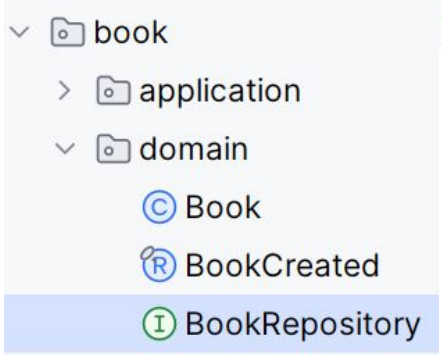
CreateBookHandler - an implementation




```
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

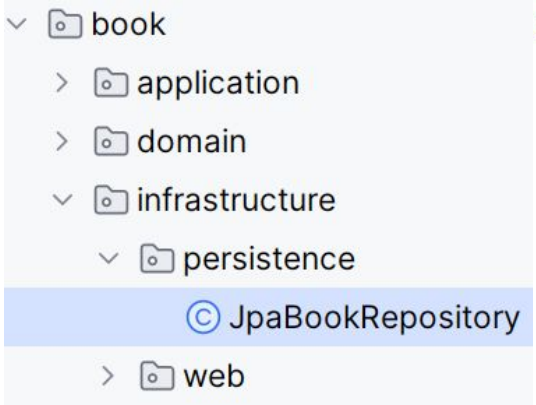
BookRepository - a driven port




```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

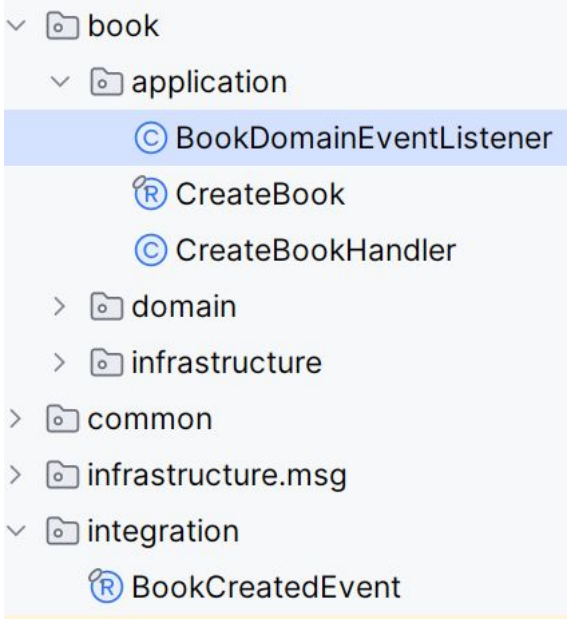
JpaBookRepository - a driven adapter




```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

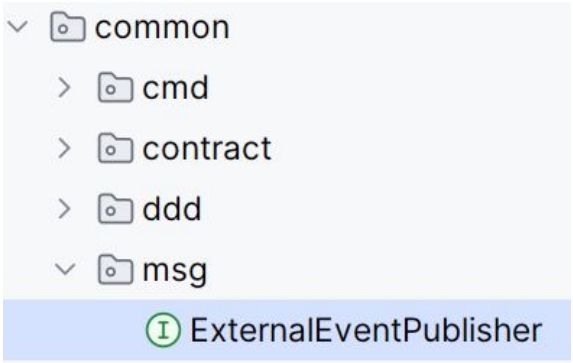



```java
public class BookDomainEventListener {

    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

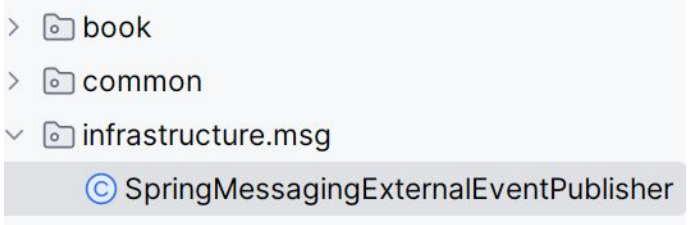
ExternalEventPublisher - a driven port




```java
public interface ExternalEventPublisher {

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

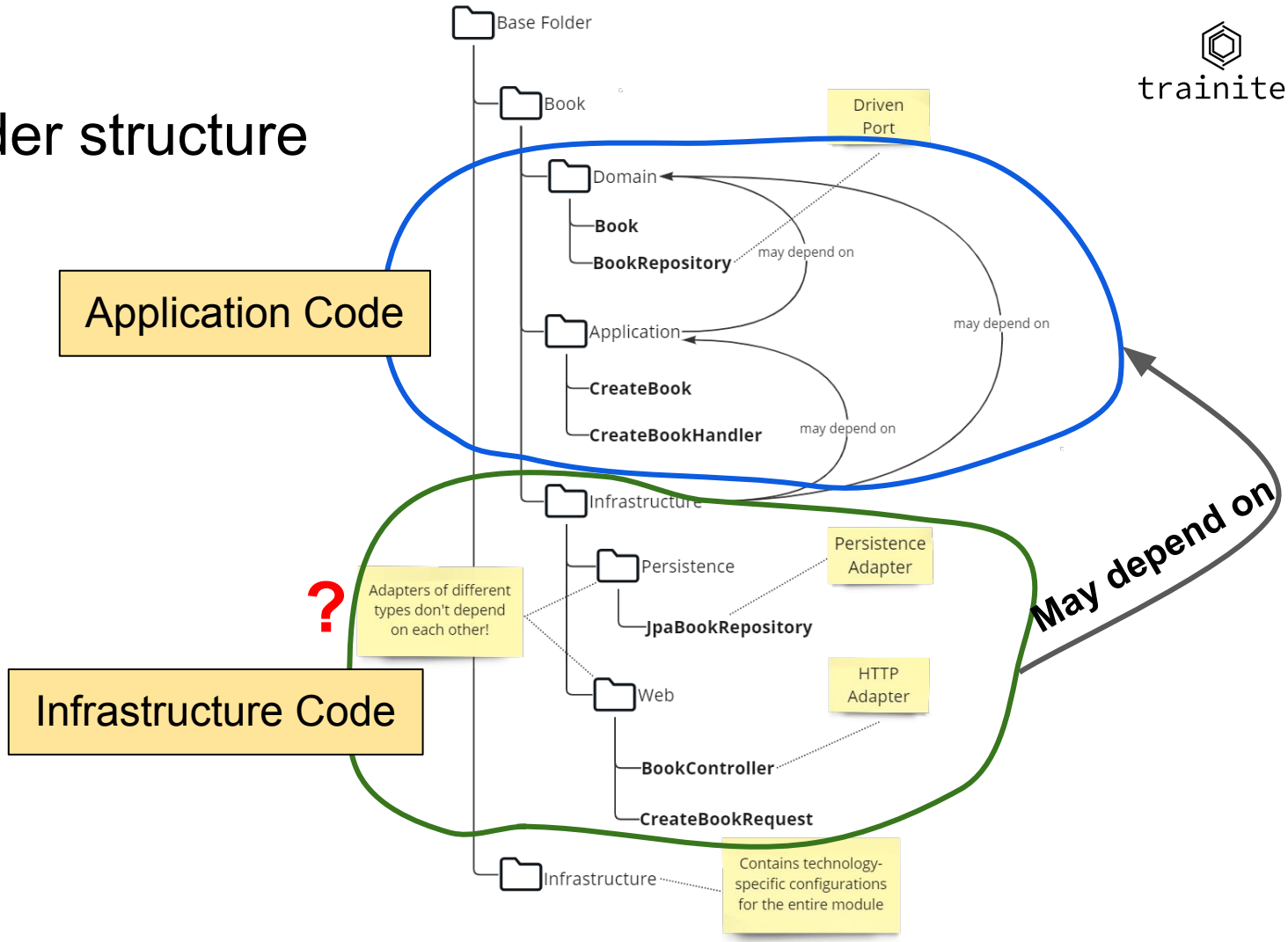
trainitek

SpringMes…EventPublisher - a driven adapter




```java
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);
    }
```

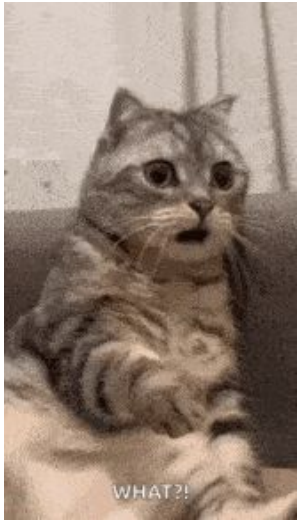
Recap: folder structure



trainitek

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

ArchUnit example: layers

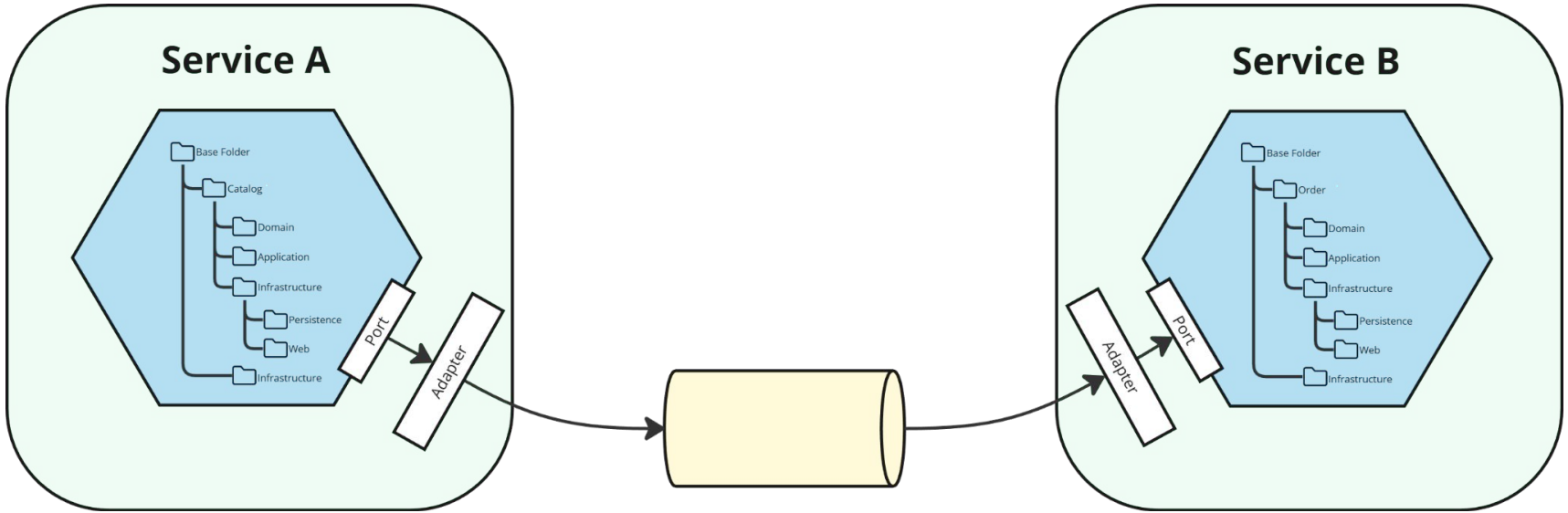
```
private static final String PKG_DOMAIN = "..domain..";  2 usages
private static final String PKG_APPLICATION = "..application..";  1 usage
private static final String PKG_INFRASTRUCTURE = "..infrastructure..";  1 usage
private static final String LAYER_DOMAIN = "Domain";  2 usages
private static final String LAYER_APPLICATION = "Application";  3 usages
private static final String LAYER_INFRASTRUCTURE = "Infrastructure";  4 usages

@ArchTest  no usages
static final ArchRule layers = layeredArchitecture() DependencySettings
        .consideringAllDependencies().withOptionalLayers(true) LayeredArchitecture
        //define layers
        .layer(LAYER_DOMAIN).definedBy(PKG_DOMAIN)
        .layer(LAYER_APPLICATION).definedBy(PKG_APPLICATION)
        .layer(LAYER_INFRASTRUCTURE).definedBy(PKG_INFRASTRUCTURE)
        //define dependencies between layers
        .whereLayer(LAYER_DOMAIN).mayOnlyBeAccessedByLayers(LAYER_APPLICATION, LAYER_INFRASTRUCTURE)
        .whereLayer(LAYER_APPLICATION).mayOnlyBeAccessedByLayers(LAYER_INFRASTRUCTURE)
        .whereLayer(LAYER_INFRASTRUCTURE).mayNotBeAccessedByAnyLayer();
```

ArchUnit example: adapters

```
@ArchTest  no usages
static final ArchRule adaptersDoNotDependOnEachOther =
    SlicesRuleDefinition.slices().matching( packageIdentifier: "..infrastructure.(*)..") GivenSlices
        .should() SlicesShould
        .notDependOnEachOther() SliceRule
        .as( newDescription: "Adapters do not depend on each other")
        .because( reason: "in Hexagonal Architecture, adapters of different types should be easily replaceable");
```

trainitek

Hexagonal Architecture in the Microservices World



Can we really call it “Architecture”?

Question: Isn’t it common sense?

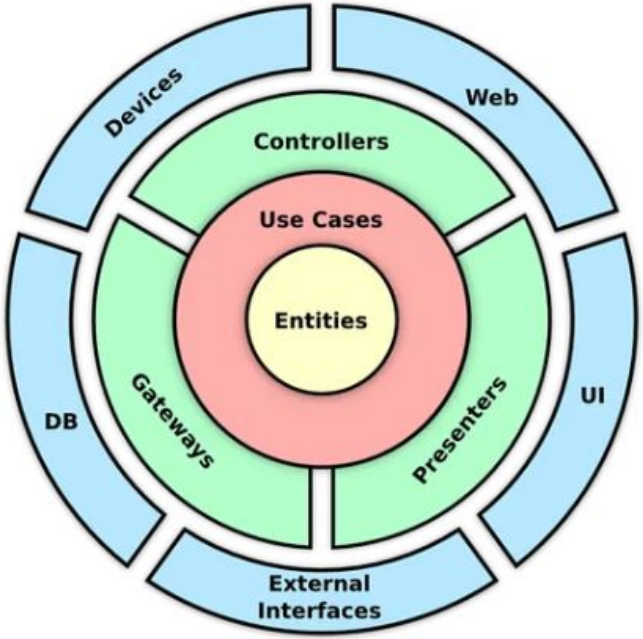
Yes, it is.

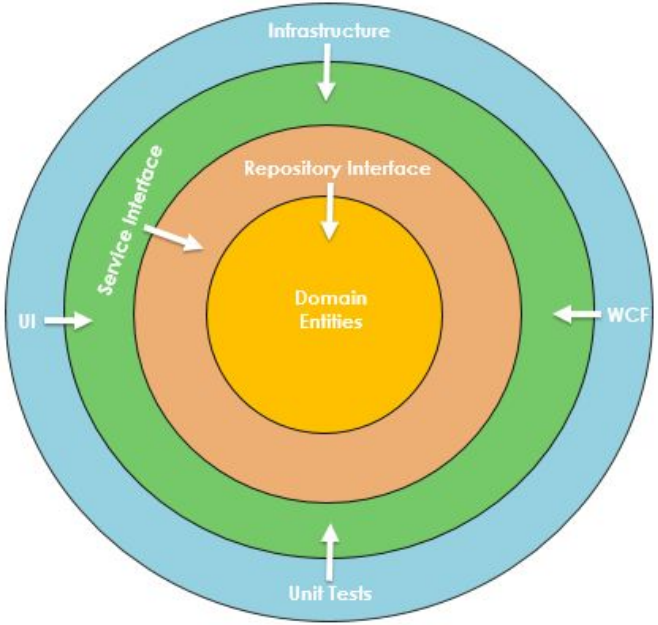
- Inheritance and polymorphism are fundamental principles of Object-Oriented Programming (OOP)
- Abstractions should not depend on their implementations

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

trainitek

Wait! But we have Clean and Onion architectures…





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? :)

Why do I prefer discussing Hexagonal Architecture?

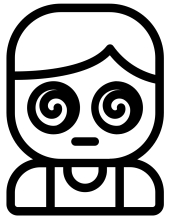
- It was the first one (2005) :)
- It has a very specific focus and memorizable concept (ports and adapters)
- It’s easier to explain it by showing the code
- It’s easier to see whether people understand it

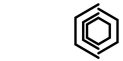
Do we always need Hexagonal Architecture?

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

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

Illusions surrounding Hexagonal Architecture

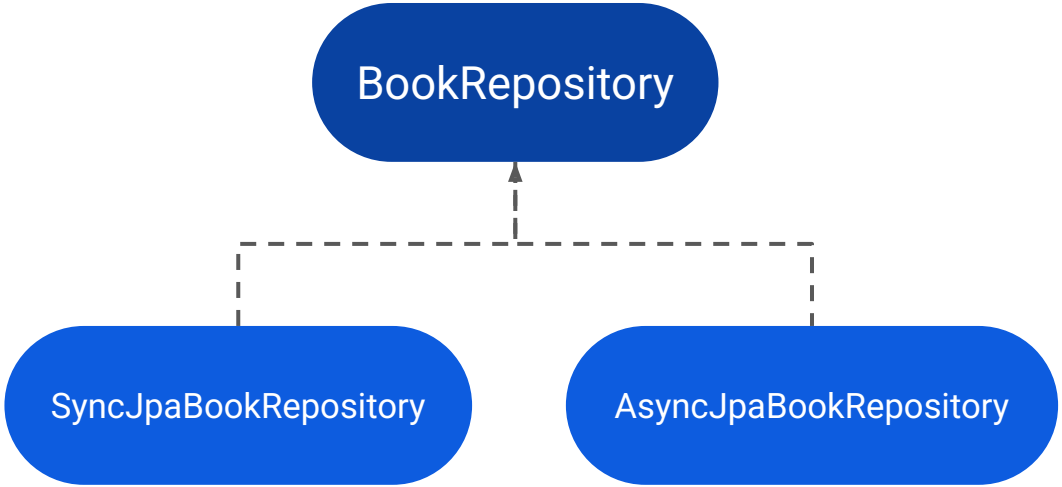




Illusion #1

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

Illusion #1: Sync/async adapters




```
1    ...
2    bookRepository.save(book)
3    ...
4    bookRepository.find(bookId)
```

Will this line work in the same way?

trainitek

Illusion #1: Sync/async adapters

Conclusion: prefer exposing async support on the port level

```
1   interface BookRepository {
2       Book save(Book book);
3   }
```

```
1   interface BookRepository {
2       CompletableFuture<Book> save(Book book);
3   }
```

In Java, Virtual Threads may help, instead of using Futures

```
1   interface IBookRepository
2   {
3       Book Save(Book book);
4   }
```

```
1   interface IBookRepository
2   {
3       Task<Book> SaveAsync(Book book);
4   }
```

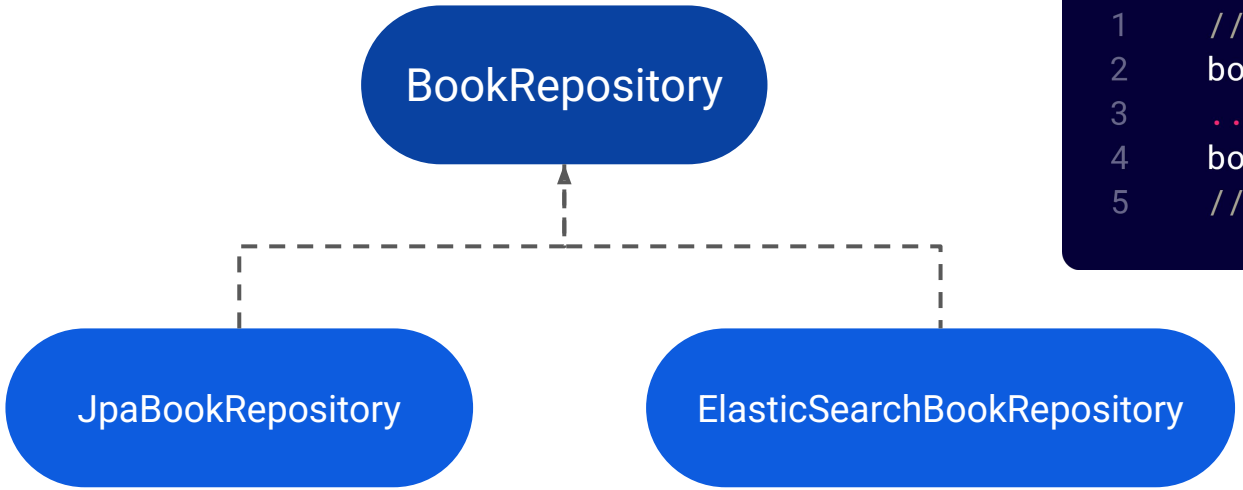
In C#, we do this in most cases

Illusion #2

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

trainitek

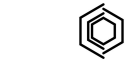
Illusion #2: Adapters with different ACID properties




```
1    //tx start
2    bookRepository.save(book1)
3    ...
4    bookRepository.save(book2)
5    //tx commit
```

Will the transaction be rolled back if this line fails?

Conclusion: the main idea is to separate application logic from infrastructure code



Illusion #3

"I can’t have library-specific annotations/attributes on the domain classes"

Illusion #3: Metadata on domain classes

```
1    @Entity
2    public class UserAccount extends UuidAggregateRoot {
```

Illusion #3: Metadata on domain classes

The rule of thumb: **be pragmatic**.

- How often do you change a chosen persistence framework?
- Do annotations/attributes help you become faster or less effective?
- Can you test your code without running infrastructure?
- Do you clearly see the value of being a purist?

Conclusion: Hexagonal Architecture is a tool, and it’s your responsibility to apply it in the most effective way to meet your needs.

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

Last Note

Remember: the **main idea** of Hexagonal Architecture is to **separate application logic from infrastructure code**, rather than to seek a silver bullet for universal architecture.

trainitek

Thank You!

Presentation Slides

vadim-prudnikov