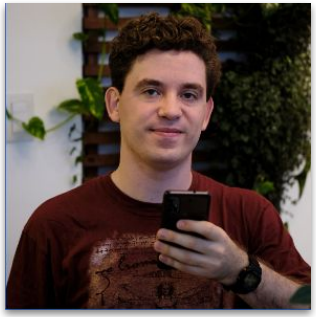
Package by Component + C4 + Structurizr

Lucas Rogério Masotti
contato@luroma.com.br

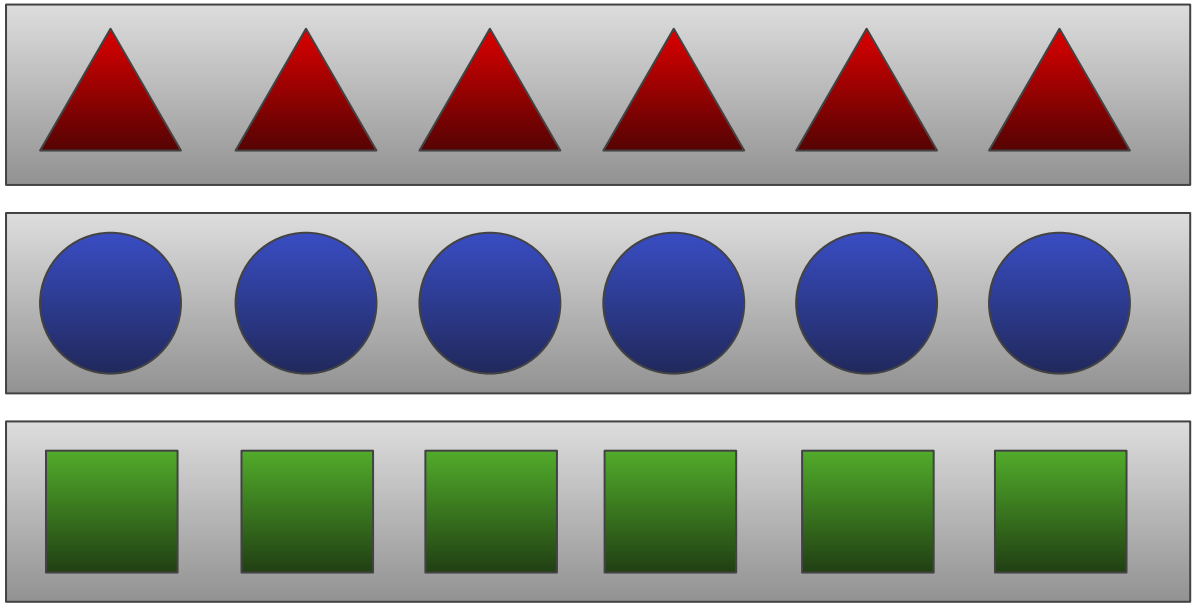


Lucas Rogério Masotti

- 11+ anos de experiência, principalmente backend e setor financeiro
- Staff Engineer Meios de Pagamentos - Sicredi
- Pós UX, MBA Gestão Marketing e pós Gestão de Negócios
- **Arquitetura, Design Organizacional e Diminuição de Complexidade Acidental**

Como organizar o código?

Package by layer

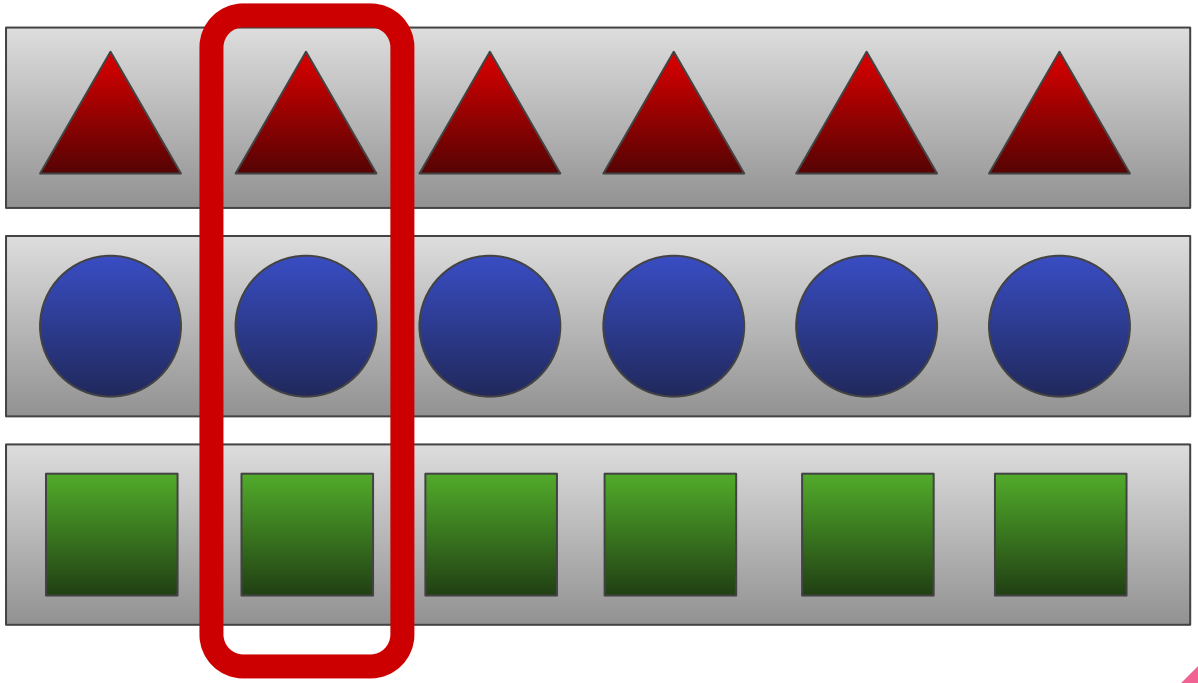


Controller
(web)

Service
(business)

Repository
(data)

Alterações

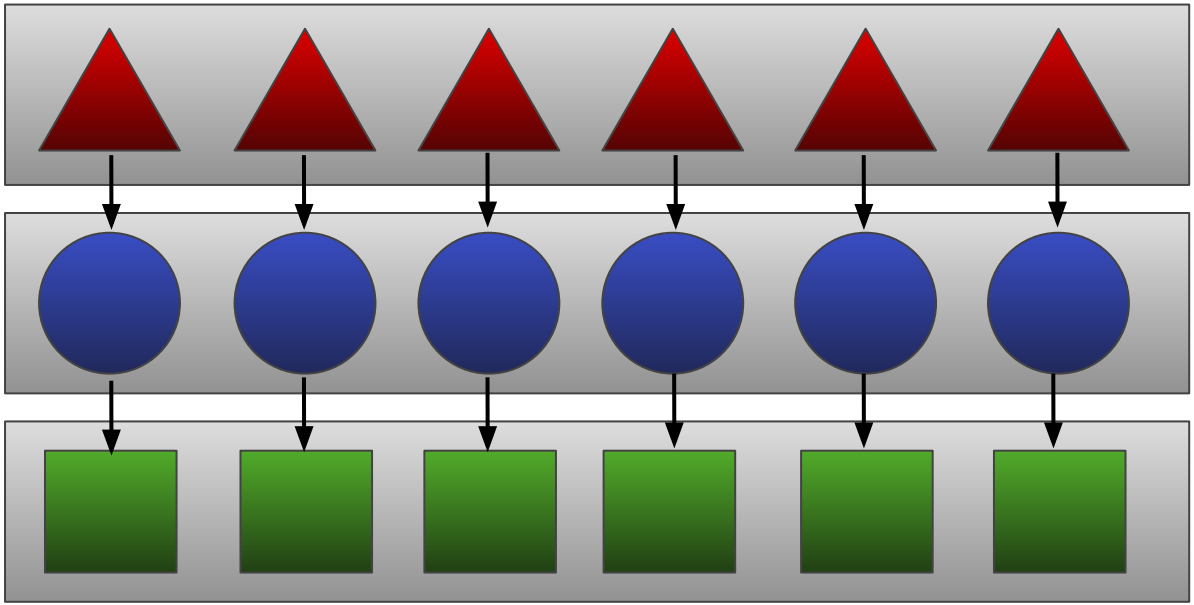


Controller
(web)

Service
(business)

Repository
(data)

Em teoria...

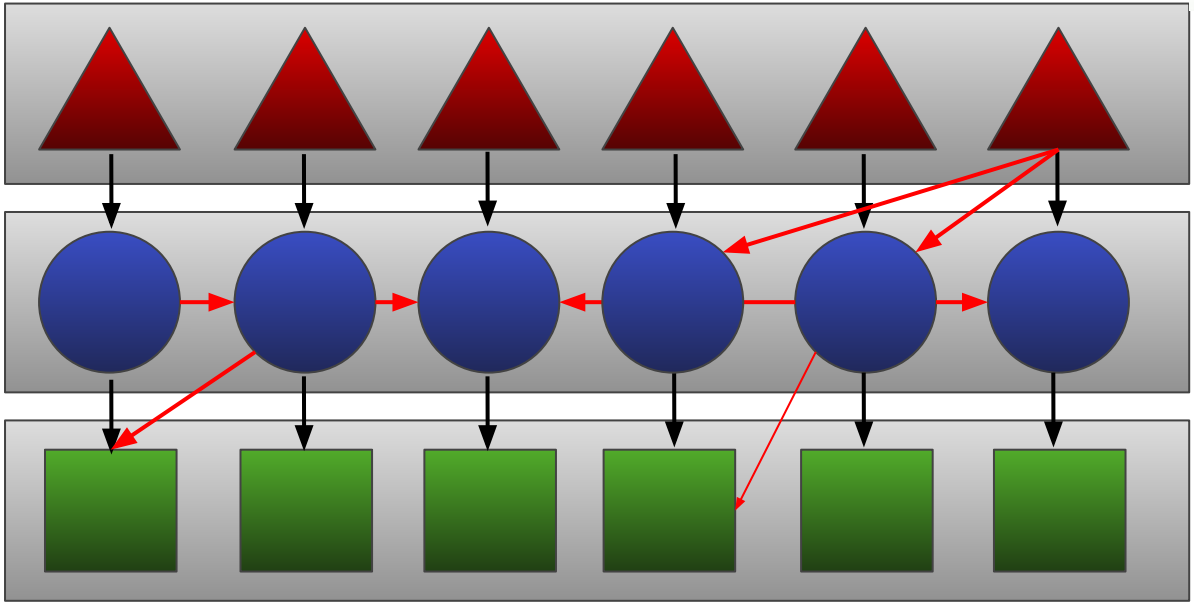


Controller
(web)

Service
(business)

Repository
(data)

Big ball of mud



Controller
(web)

Service
(business)

Repository
(data)

Big ball of mud

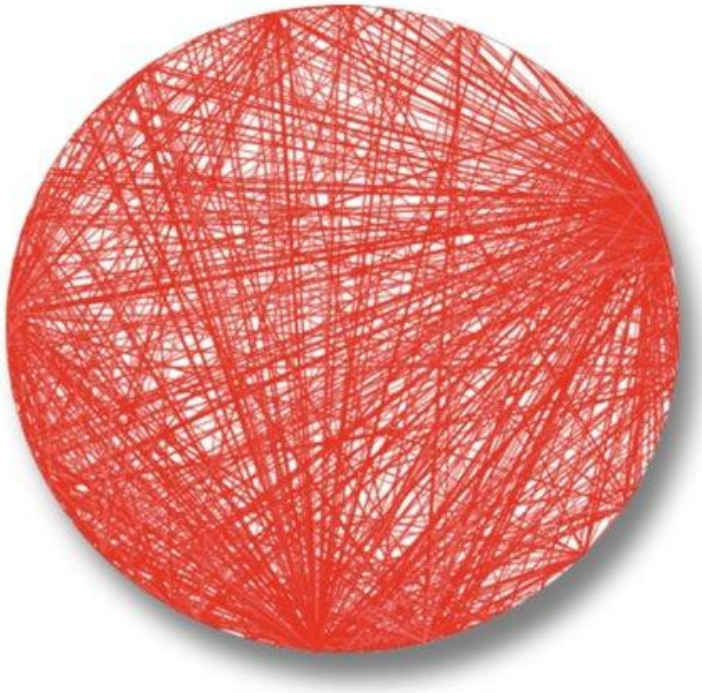
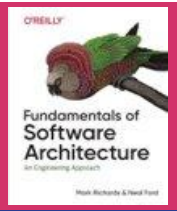


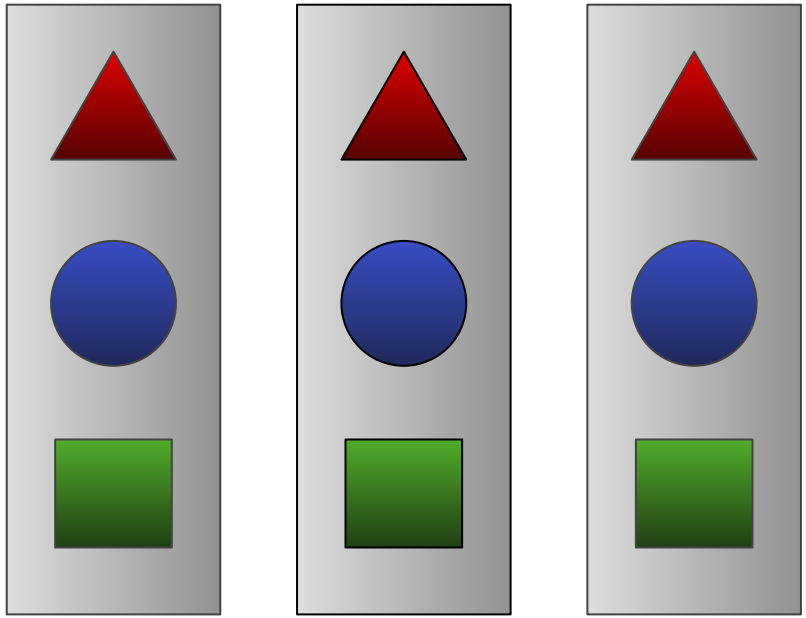
Figure 9-1. A Big Ball of Mud architecture visualized from a real code base

Cada ponto no perímetro do círculo representa um classe

Cada linha uma conexão entre classes



Package by feature



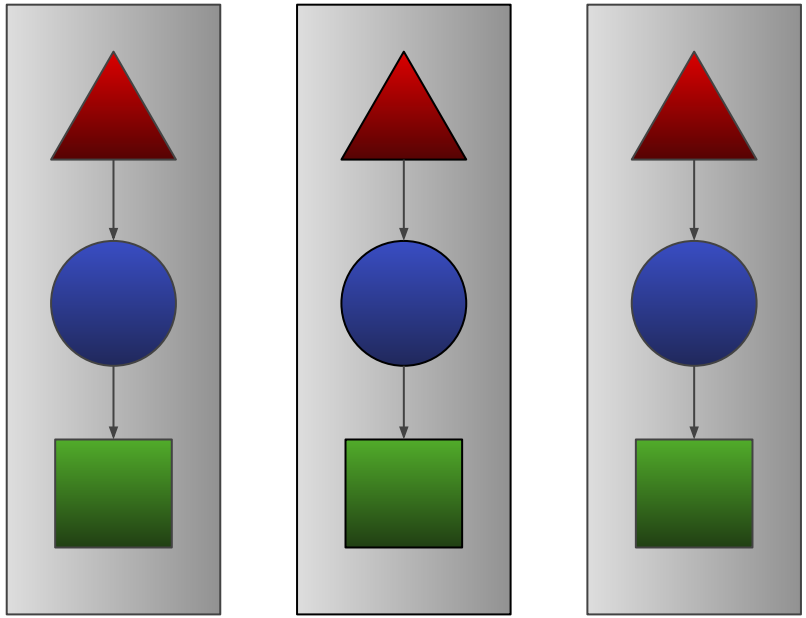
Funcionalidade A | Funcionalidade B | Funcionalidade C

Controller
(web)

Service
(business)

Repository
(data)

Em teoria...



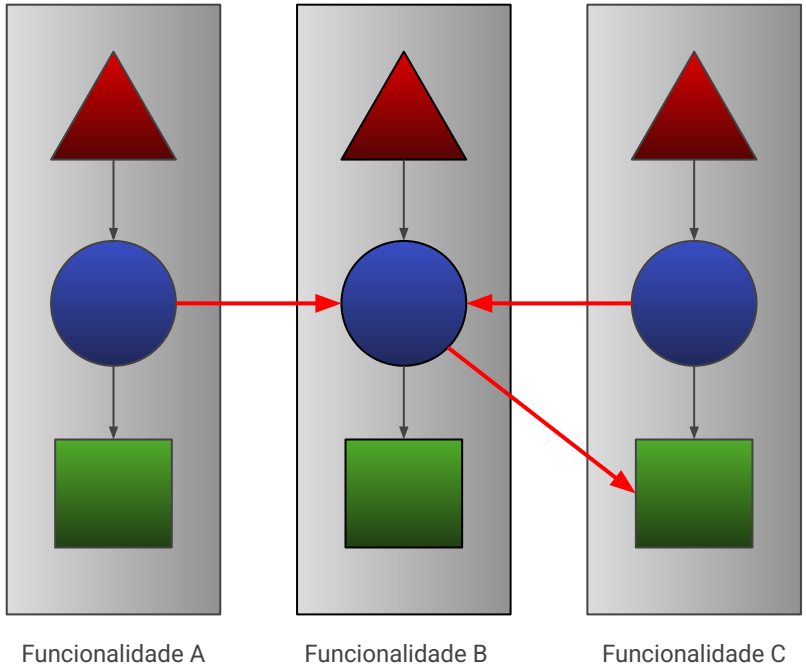
Funcionalidade A | Funcionalidade B | Funcionalidade C

Controller
(web)

Service
(business)

Repository
(data)

Package by feature

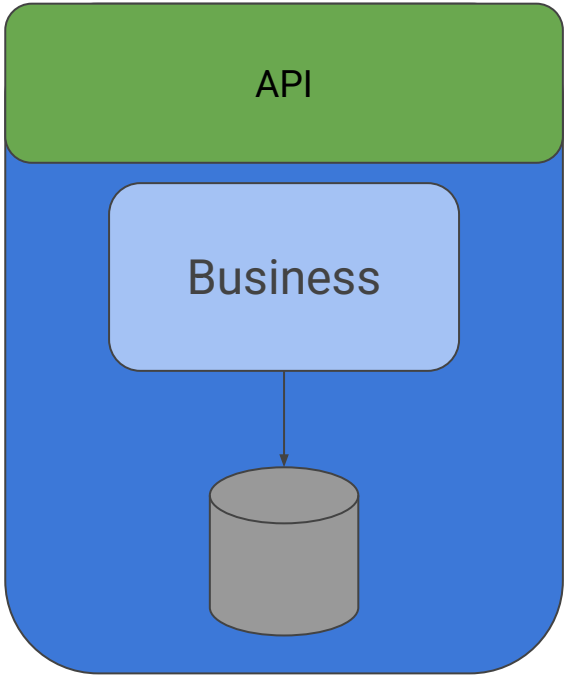


Controller
(web)

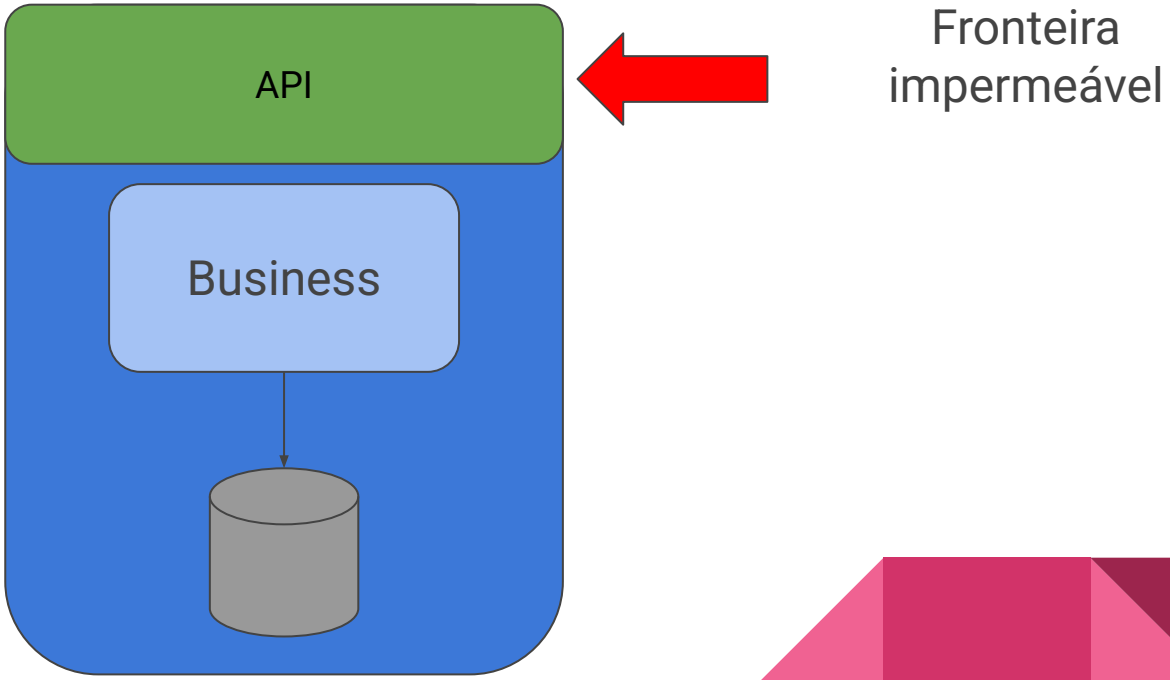
Service
(business)

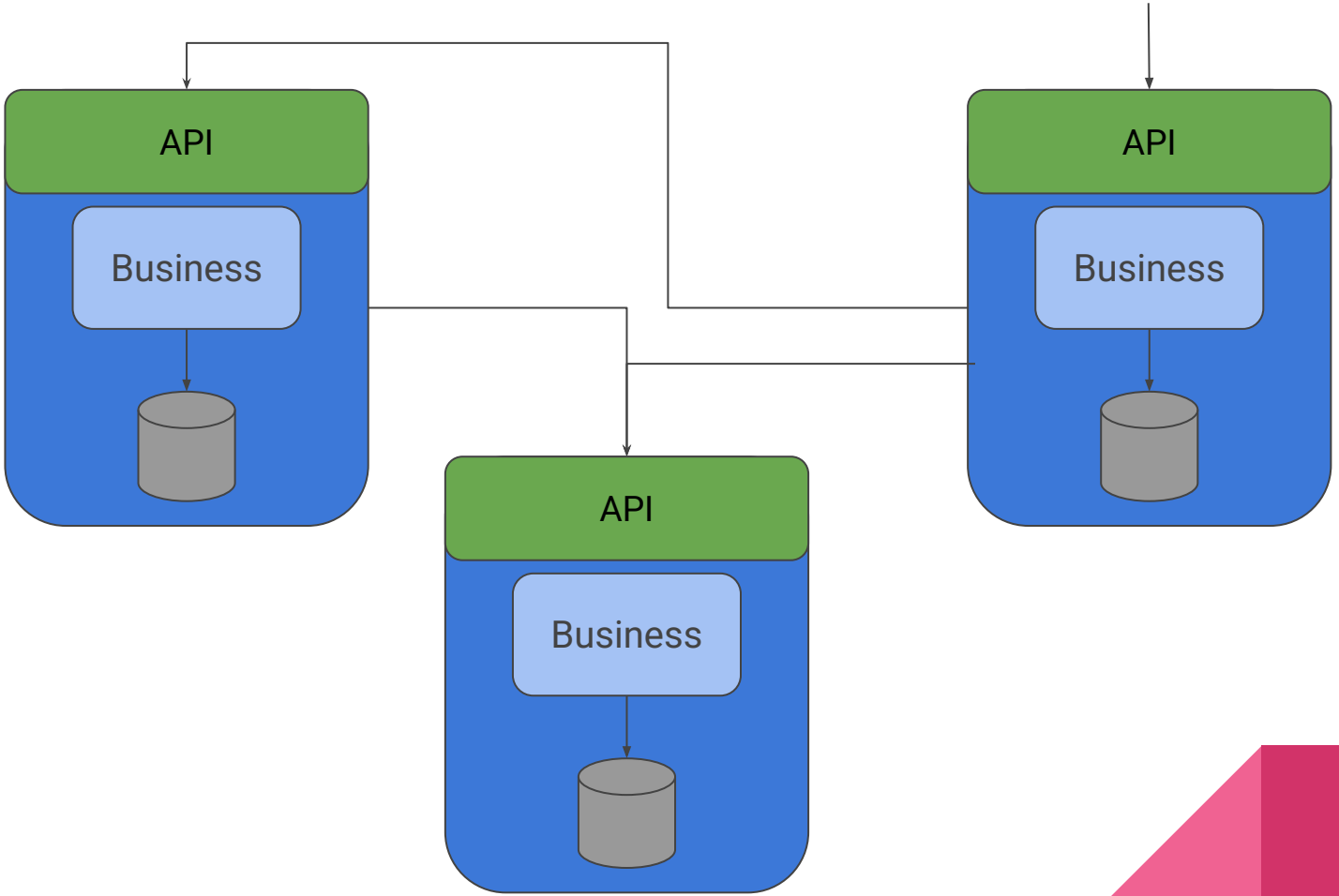
Repository
(data)

Microsserviços

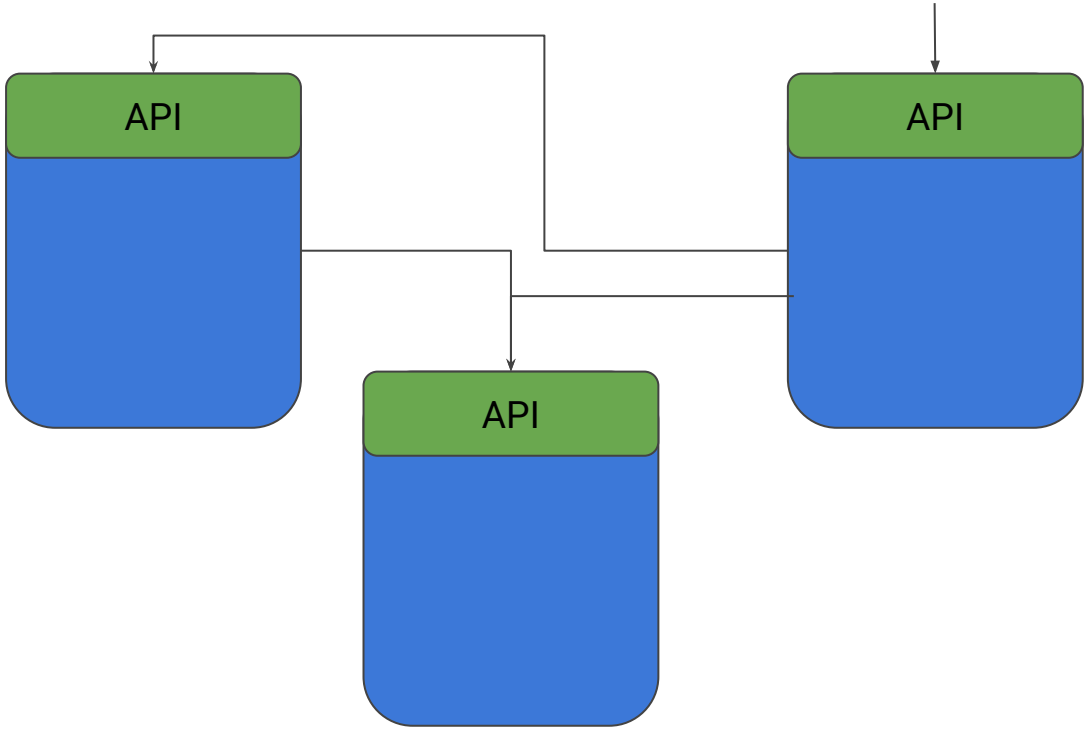


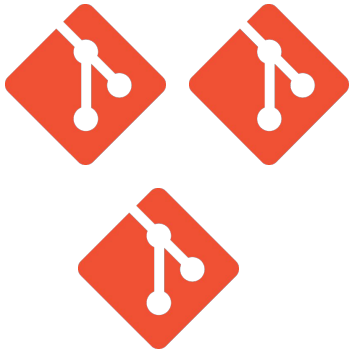
Microsserviços como forma de desacoplamento



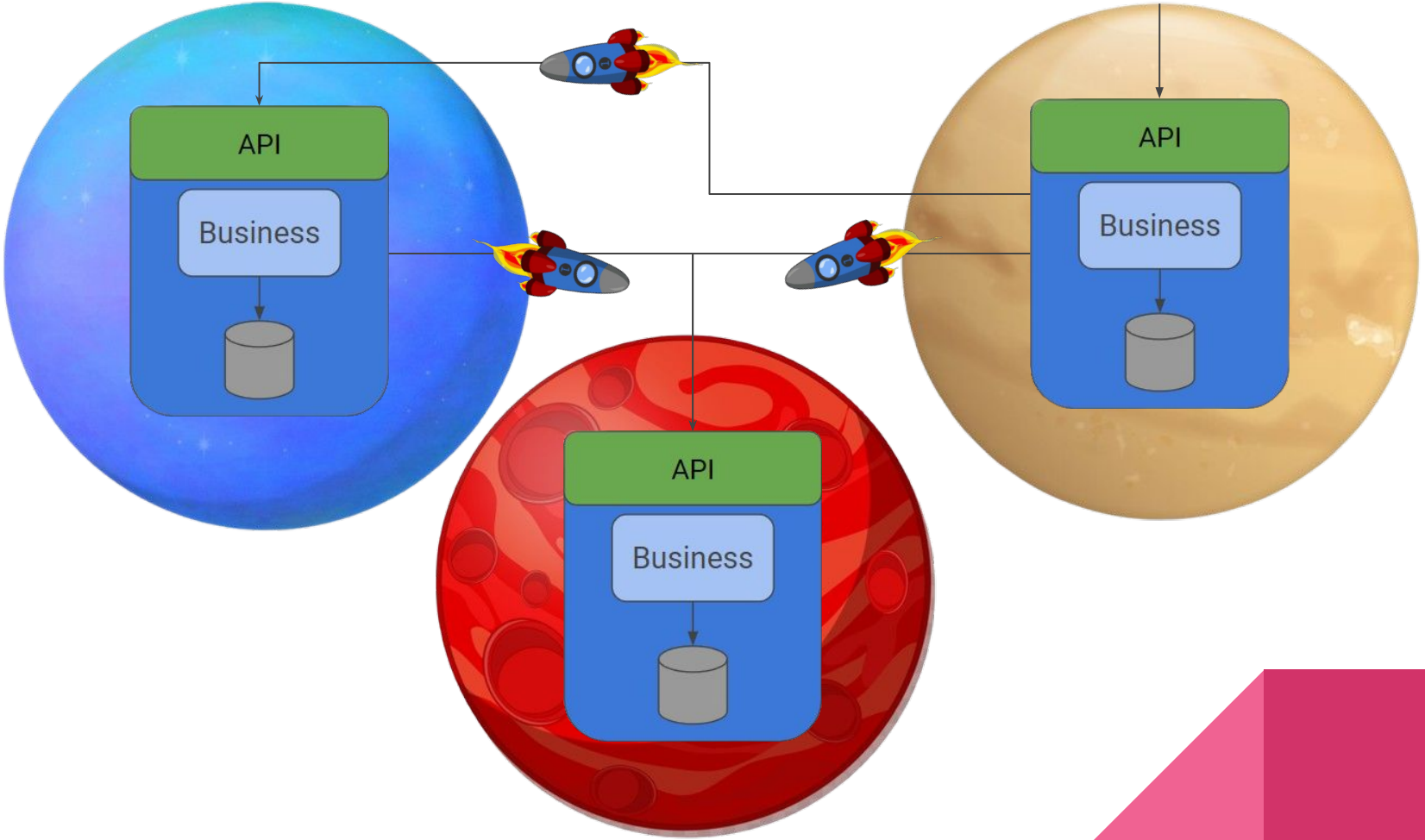

API
Business
API
Business
API
Business

Encapsulamento





É só usar microsserviços!
Tudo resolvido então?


API
Business
API
Business
API
Business

Falácias da computação distribuída

Fallacy #1: The Network Is Reliable

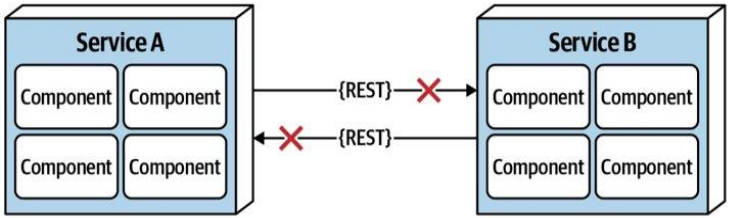


Figure 9-2. The network is not reliable

Fallacy #2: Latency Is Zero

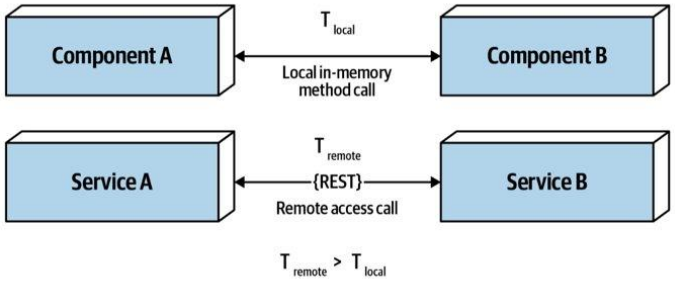


Figure 9-3. Latency is not zero

Fallacy #3: Bandwidth Is Infinite

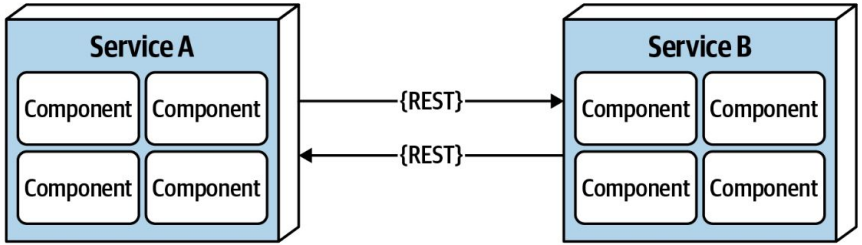


Figure 9-4. Bandwidth is not infinite

Fallacy #4: The Network Is Secure

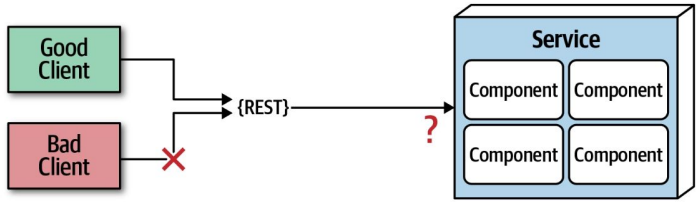
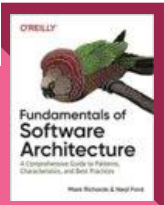


Figure 9-5. The network is not secure



Custo, performance e simplicidade

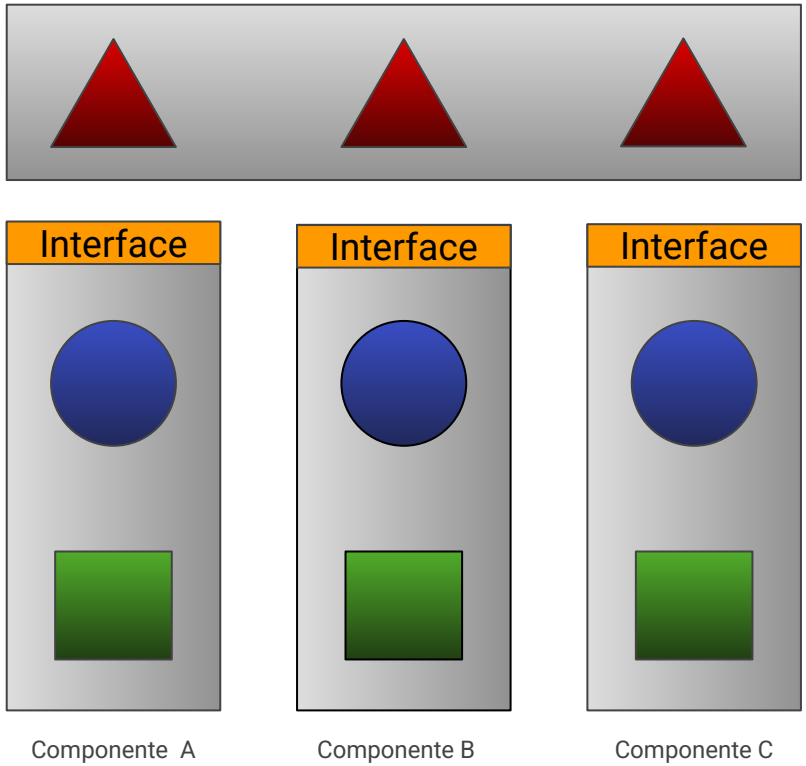
Architecture characteristic	Star rating
Partitioning type	Domain
Number of quanta	1 to many
Deployability	★★★★
Elasticity	★★★★★
Evolutionary	★★★★★
Fault tolerance	★★★★
Modularity	★★★★★
Overall cost	★
Performance	★★
Reliability	★★★★
Scalability	★★★★★
Simplicity	★
Testability	★★★★

Figure 17-13. Ratings for microservices

- Logs distribuídos
- Transações distribuídas (consistência eventual, sagas)
- Manutenção e versionamento de contratos

Package by component

Package by component



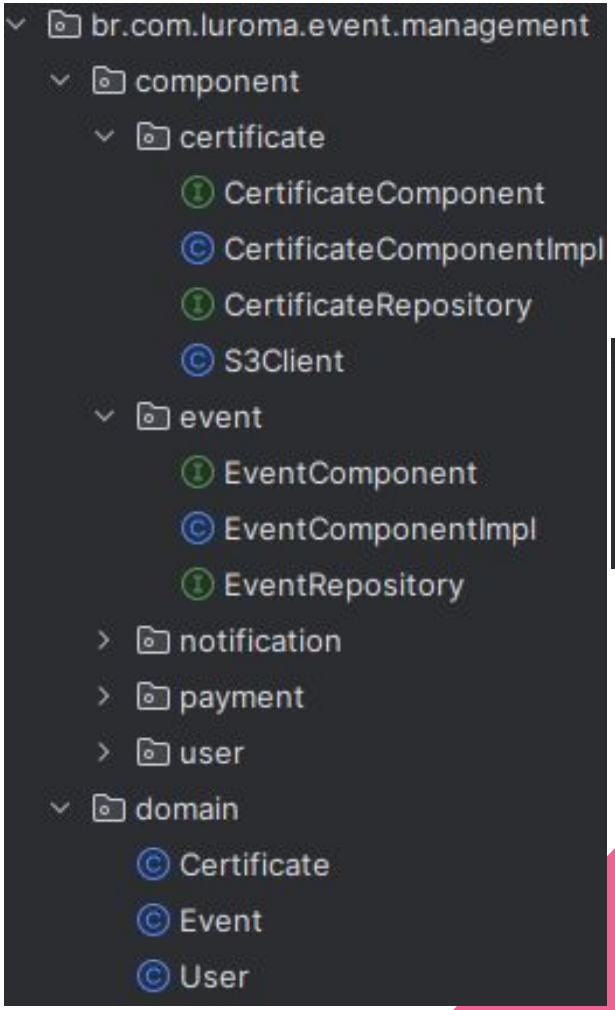
Controller
(web)

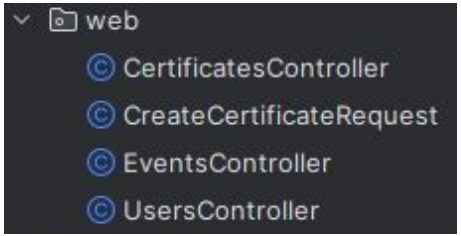
Service
(business)

Repository
(data)

Package by component

- Um pacote para as *controllers*
- Um pacote para cada *component*, com **uma** interface pública
- Um pacote *domain*
- O restante do código do pacote é *package-private*, estrutura flat



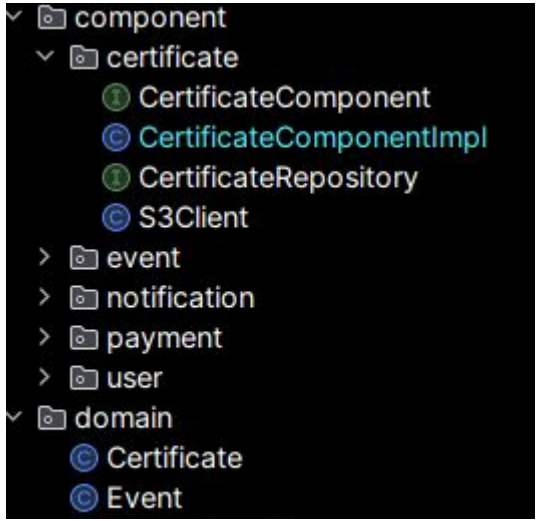


```java
public interface CertificateComponent {
    List<Certificate> getCertificates(String userId);
    Certificate createCertificate(String userId, String eventId);
}
```

```java
@Component
class CertificateComponentImpl implements CertificateComponent {
    private final CertificateRepository certificateRepository;

    public CertificateComponentImpl(CertificateRepository certificateRepository) {
        this.certificateRepository = certificateRepository;
    }

    @Override
    public List<Certificate> getCertificates(String userId) {
        return certificateRepository.findByUserId(userId);
    }

    @Override
    public Certificate createCertificate(String userId, String eventId) {
        return certificateRepository.save(new Certificate(userId, eventId));
    }
}
```

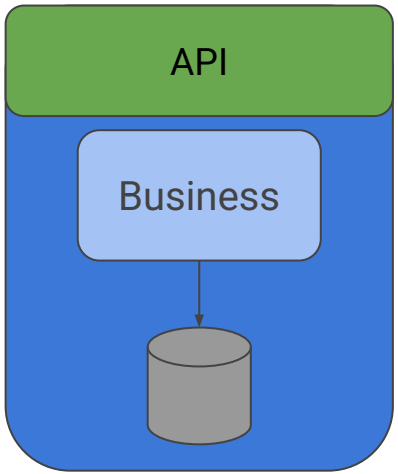
```java
interface CertificateRepository extends CrudRepository<Certificate, String> {
    List<Certificate> findByUserId(String userId);
}
```

```java
@Component
class CertificateComponentImpl implements CertificateComponent {
    private final CertificateRepository certificateRepository;

    public CertificateComponentImpl(CertificateRepository certificateRepository) {
        this.certificateRepository = certificateRepository;
    }

    @Override
    public List<Certificate> getCertificates(String userId) {

        return certificateRepository.findByUserId(userId);
    }

    @Override
    public Certificate createCertificate(String userId, String eventId) {
        return certificateRepository.save(new Certificate(userId, eventId));
    }
}
```



Na prática...

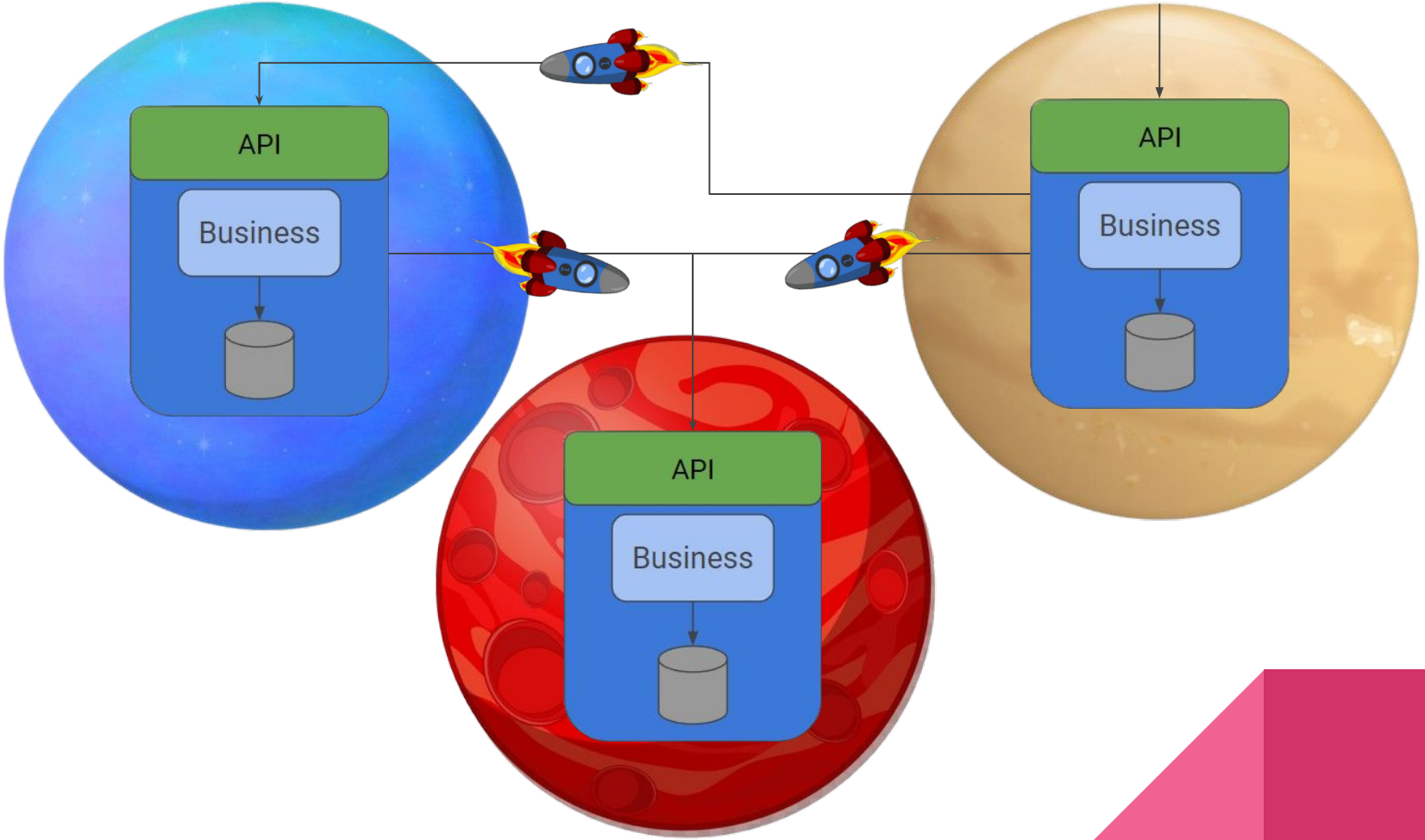
Fronteira impermeável

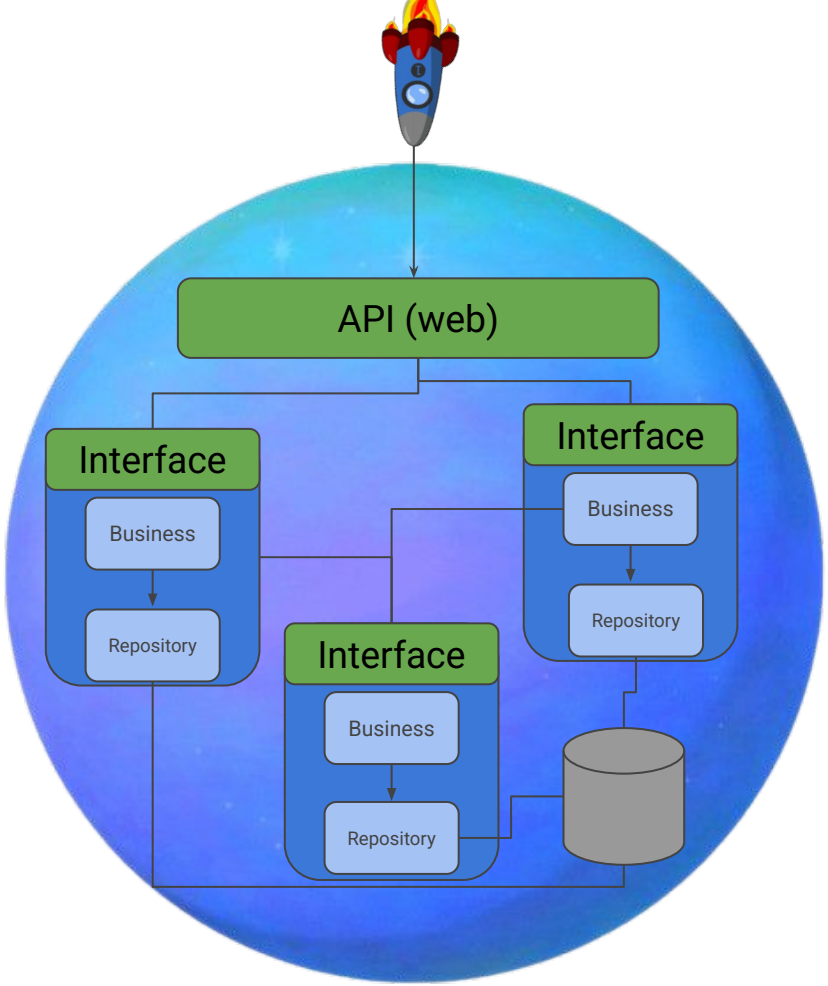


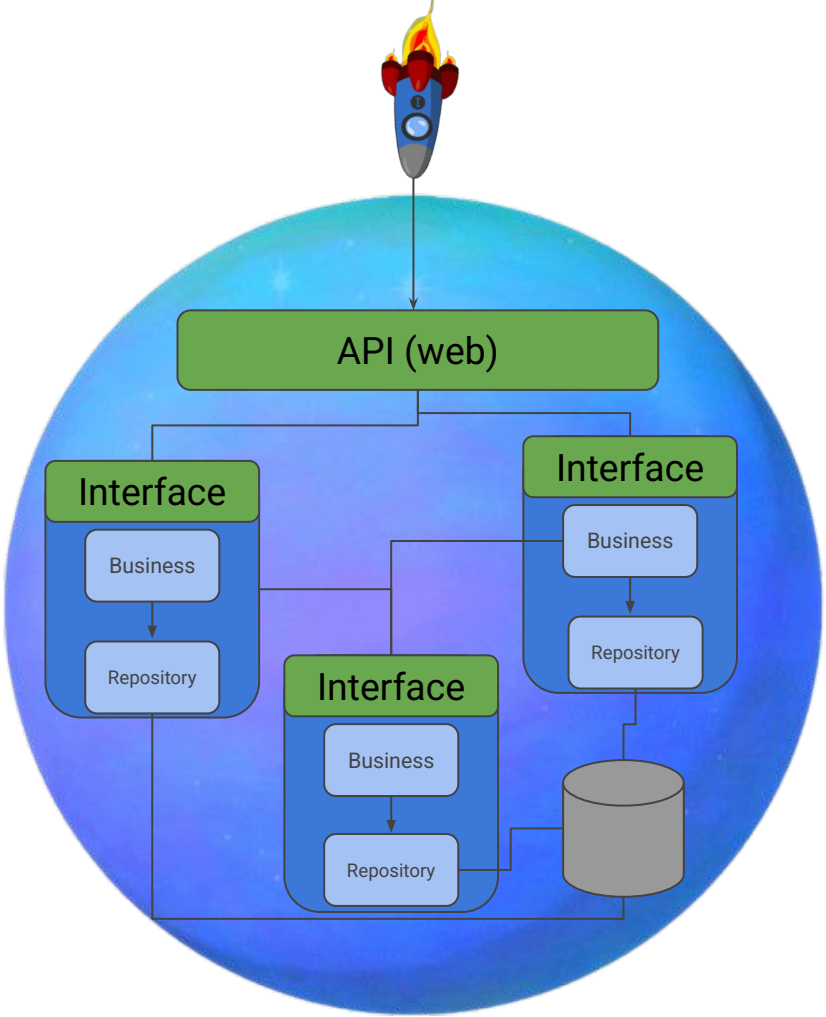

```java
public interface CertificateComponent {
    List<Certificate> getCertificates(String userId);
    Certificate createCertificate(String userId, String eventId);
}
```

```java
@Component
class CertificateComponentImpl implements CertificateComponent {
    private final CertificateRepository certificateRepository;

    public CertificateComponentImpl(CertificateRepository certificateRepository) {
        this.certificateRepository = certificateRepository;
    }

    @Override
    public List<Certificate> getCertificates(String userId) {
        return certificateRepository.findByUserId(userId);
    }

    @Override
    public Certificate createCertificate(String userId, String eventId) {
        return certificateRepository.save(new Certificate(userId, eventId));
    }
}
```

```java
interface CertificateRepository extends CrudRepository<Certificate, String> {
    List<Certificate> findByUserId(String userId);
}
```

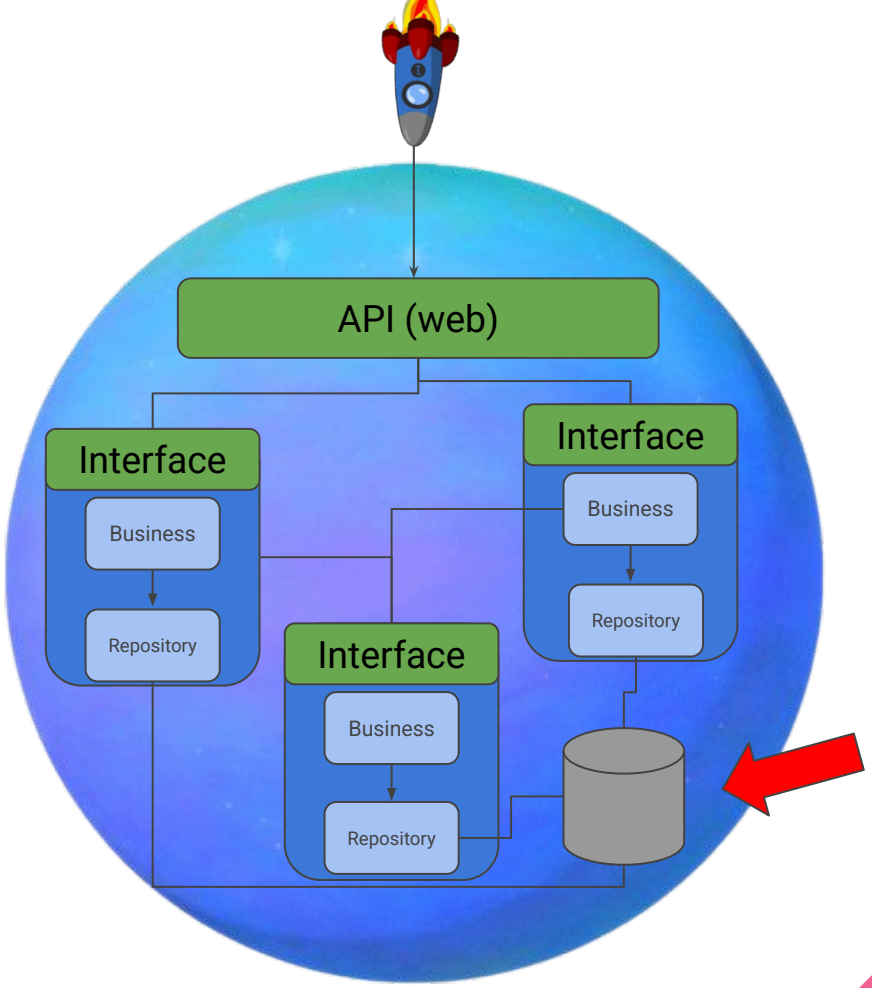

API
Business
API
Business
API
Business


API (web)
Interface
Business
Repository
Interface
Business
Repository
Interface
Business
Repository



- Facilidade de refatoração
- Chamadas em memória
- Desenvolvimento mais simples

Sem microsserviços?



Uma única base?

Como definir os components?

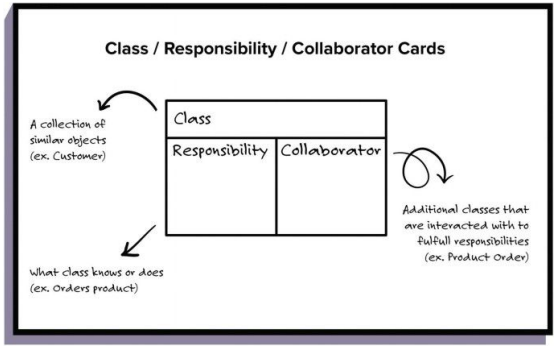
Programming Techniques — R. Morris, Editor

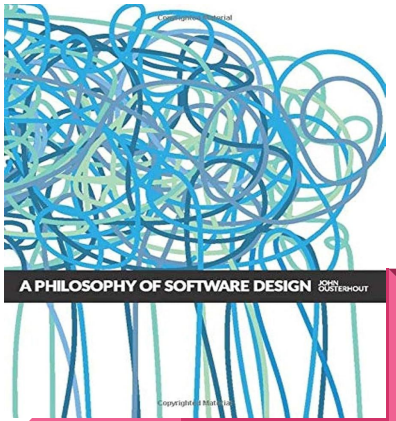
On the Criteria To Be Used in Decomposing Systems into Modules

Expected Benefits of Modular Programming

The benefits expected of modular programming are: (1) managerial—development time should be shortened because separate groups would work on each module with little need for communication: (2) product flexibility—it should be possible to make drastic changes to one module without a need to change others; (3) comprehensibility—it should be possible to study the system one module at a time. The whole system can therefore be better designed because it is better understood.

1972





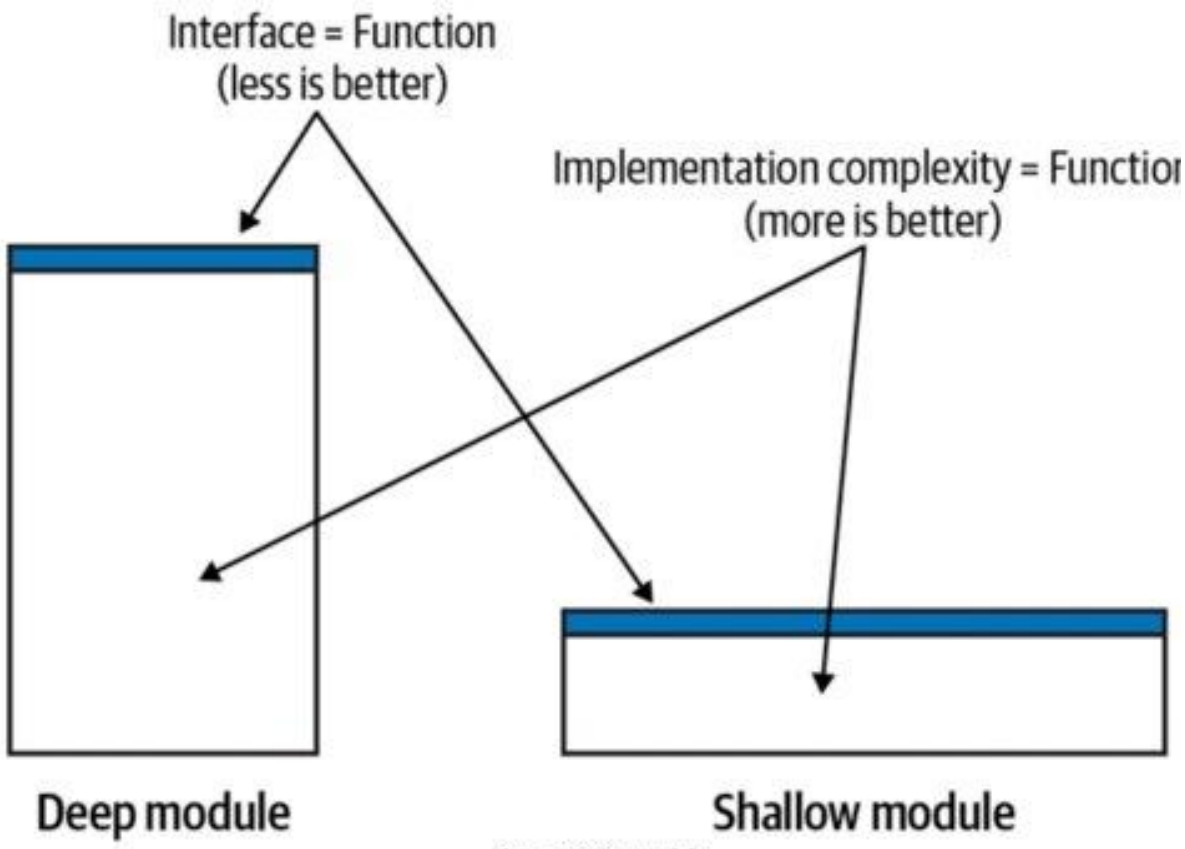
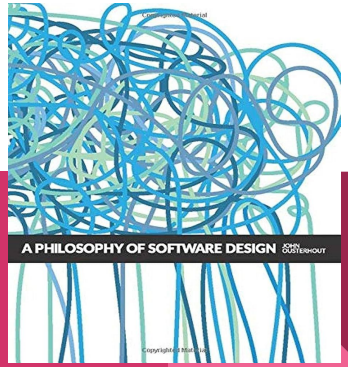


Figure 14-6. Deep modules



Como garantir certas regras?

ArchUnit

```java
@Test
public void controllersShouldBeOnWebPackage() {
    JavaClasses importedClasses = new ClassFileImporter().importPackages("br.com.luroma.event.management");

    classes().that().haveSimpleNameEndingWith("Controller")
            .should().resideInAPackage("br.com.luroma.event.management.web")
            .check(importedClasses);
}
```

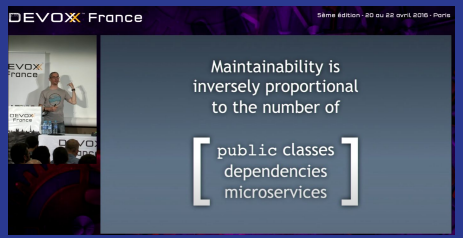
```
Architecture Violation [Priority: MEDIUM] - Rule 'classes that have simple name ending with 'Controller' should reside in a package 'br.com.luroma.event.management.web'' was violated (1 times):
Class <br.com.luroma.event.management.domain.TesteController> does not reside in a package 'br.com.luroma.event.management.web' in (TesteController.java:0)
java.lang.AssertionError Create breakpoint: Architecture Violation [Priority: MEDIUM] - Rule 'classes that have simple name ending with 'Controller' should reside in a package 'br.com.luroma.event
Class <br.com.luroma.event.management.domain.TesteController> does not reside in a package 'br.com.luroma.event.management.web' in (TesteController.java:0)
    at com.tngtech.archunit.lang.ArchRule$Assertions.assertNoViolation(ArchRule.java:94)
    at com.tngtech.archunit.lang.ArchRule$Assertions.check(ArchRule.java:86)
    at com.tngtech.archunit.lang.ArchRule$Factory$SimpleArchRule.check(ArchRule.java:165)
    at com.tngtech.archunit.lang.syntax.ObjectsShouldInternal.check(ObjectsShouldInternal.java:81)
    at br.com.luroma.event.management.EnforceControllersTest.controllersShouldBeOnWebPackage(EnforceControllersTest.java:17) <1 internal line>
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1597)
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1597)
```

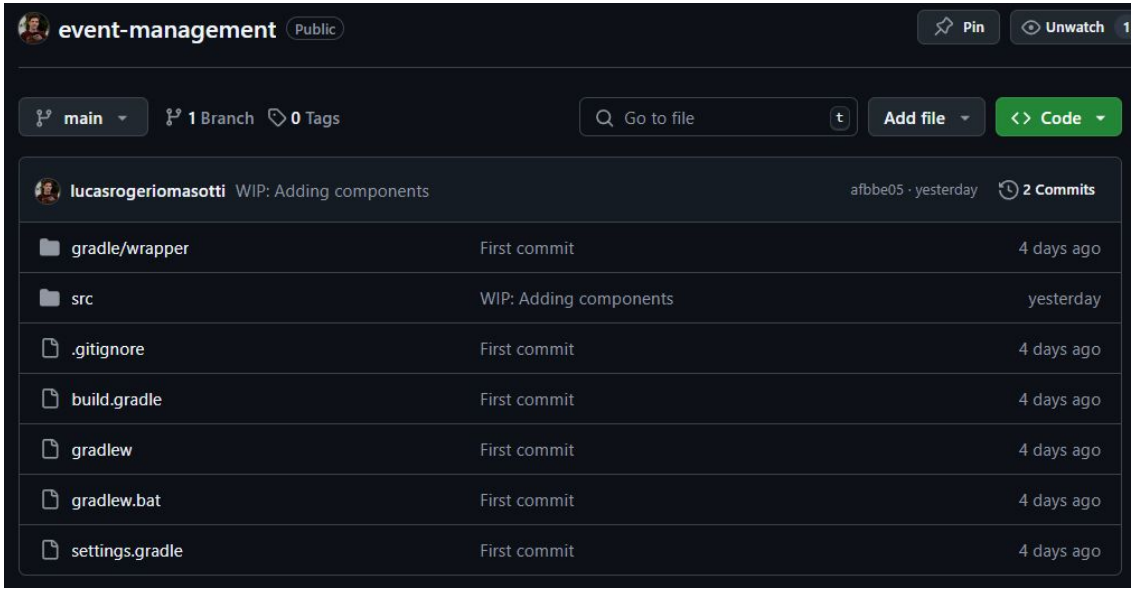
Public

```
@ArchTest
public static final ArchRule noPublicClassesTharAreNotInterfaces =  classes()
        .that().areNotInterfaces()
        .or().haveSimpleNameEndingWith("Repository")
        .and().resideInAPackage("..component..")
        .should().notBePublic();
```

“**Maintainability** is **inversely proportional** to the number of public classes, dependencies and microservices…”

Simon Brown

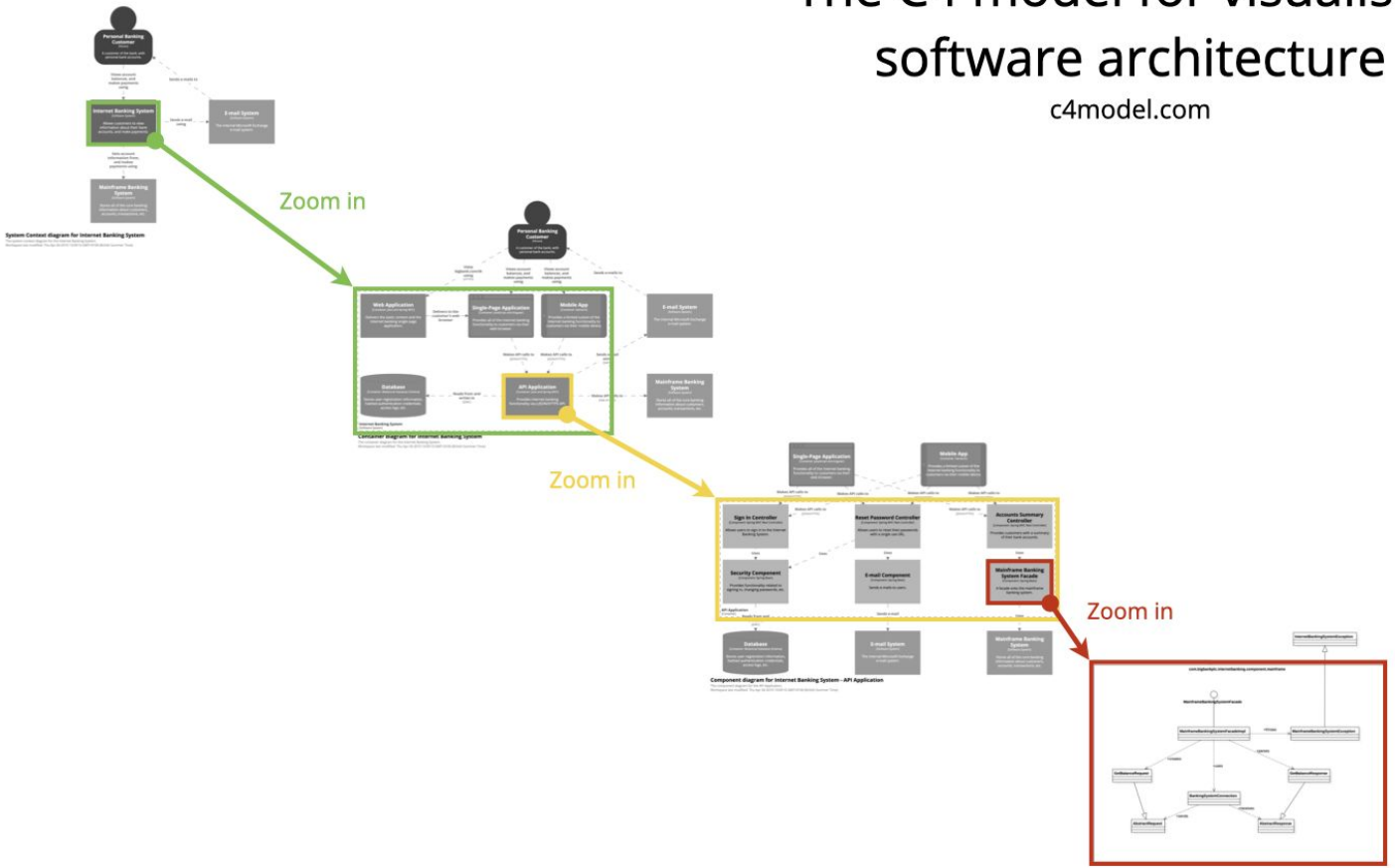



event-management
Public
Pin
Unwatch
main
1 Branch
0 Tags
Go to file
t
Add file
Code
lucasrogeriomasotti WIP: Adding components
afbbe05 · yesterday
2 Commits
gradle/wrapper First commit 4 days ago
src WIP: Adding components yesterday
.gitignore First commit 4 days ago
build.gradle First commit 4 days ago
gradlew First commit 4 days ago
gradlew.bat First commit 4 days ago
settings.gradle First commit 4 days ago

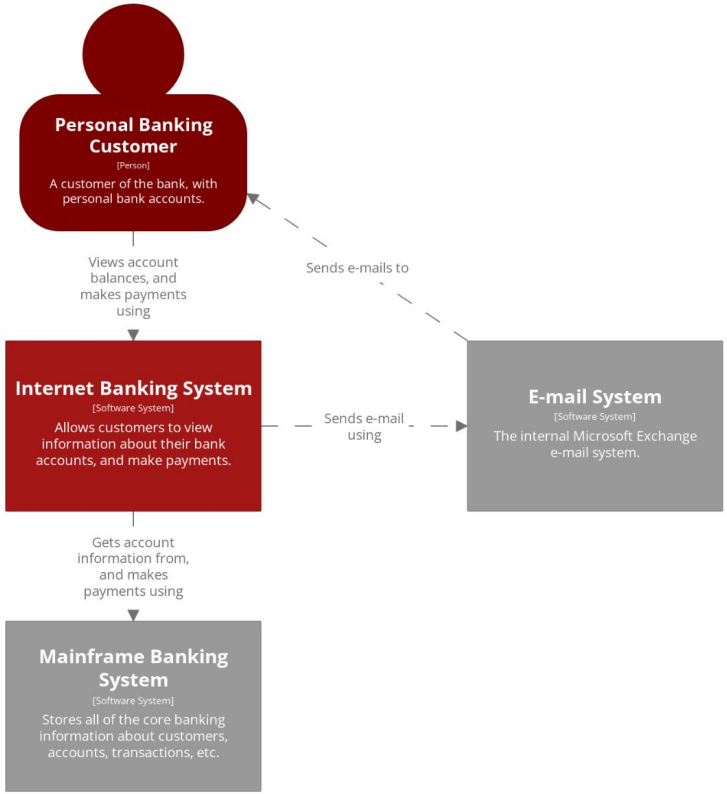
Como eu visualizo?

The C4 model for visualising software architecture

c4model.com

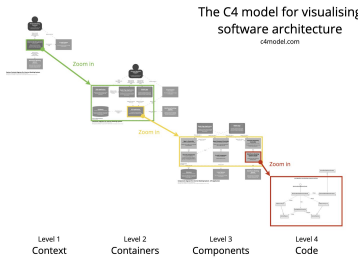


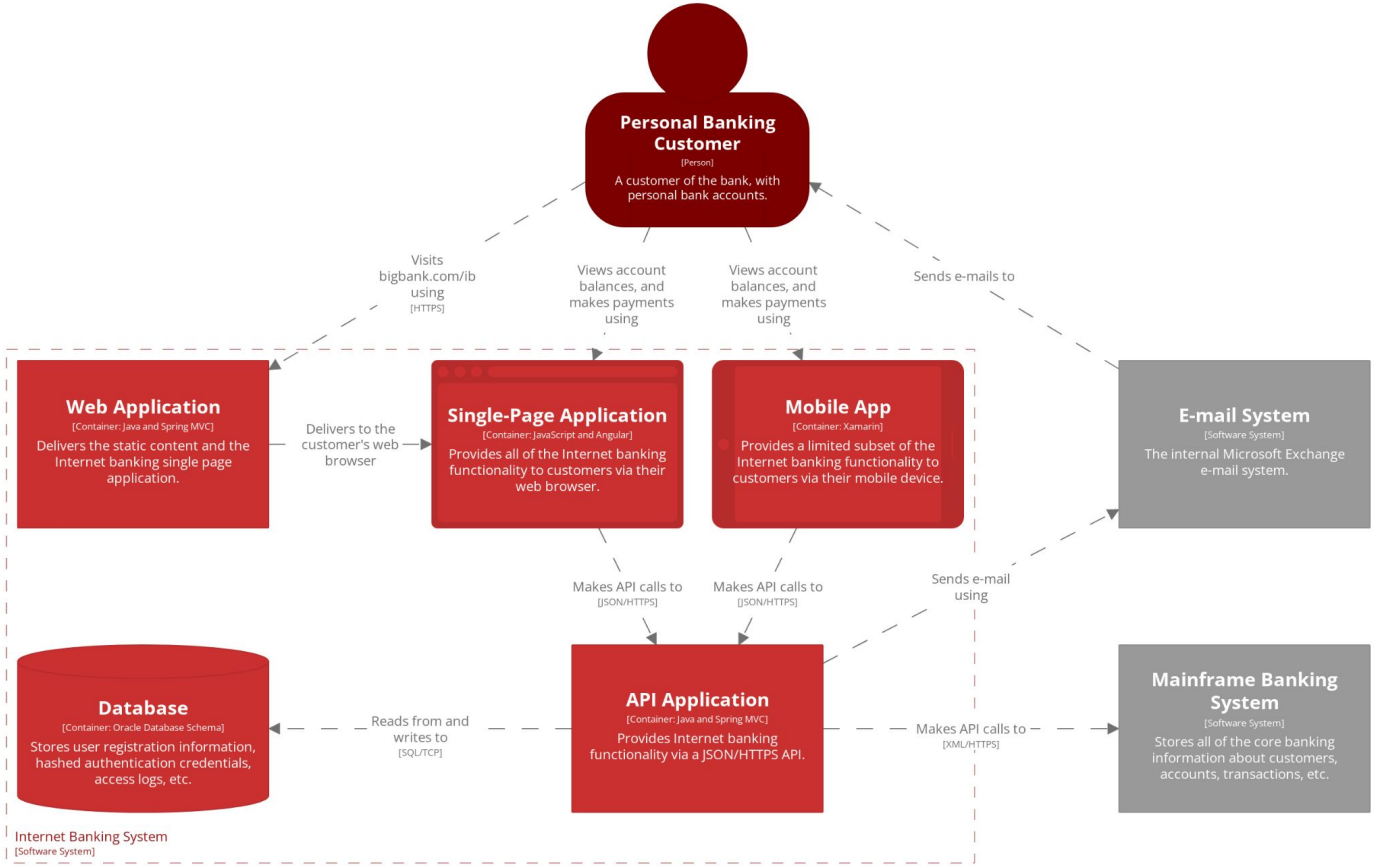
Level 1	Level 2	Level 3	Level 4
Context	Containers	Components	Code



[System Context] Internet Banking System

The system context diagram for the Internet Banking System - diagram created with Structurizr.
Wednesday, March 22, 2023 at 8:16 AM Coordinated Universal Time

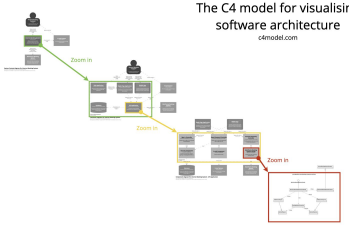


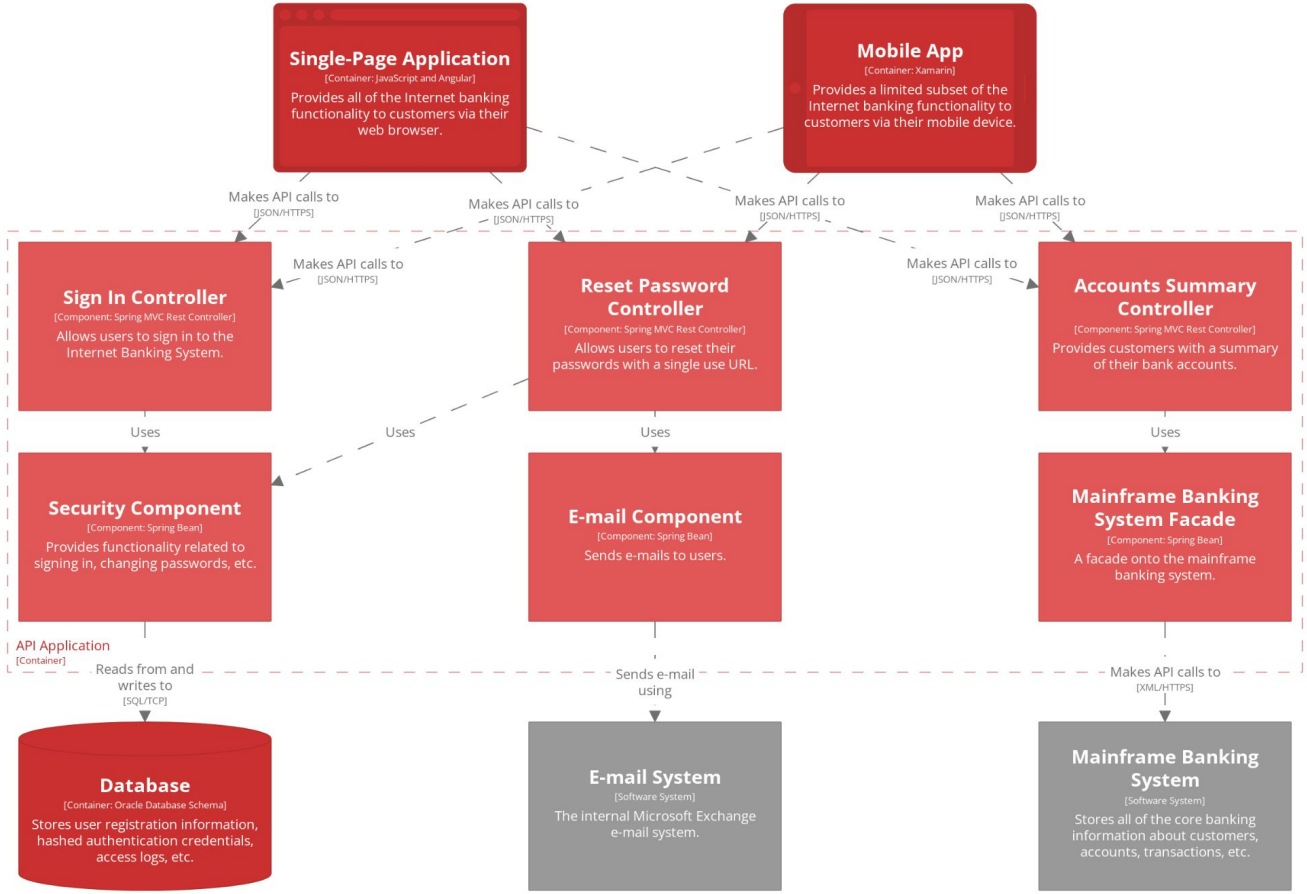


[Container] Internet Banking System

The container diagram for the Internet Banking System - diagram created with Structurizr.
Wednesday, March 22, 2023 at 8:16 AM Coordinated Universal Time

The C4 model for visualising software architecture
c4model.com

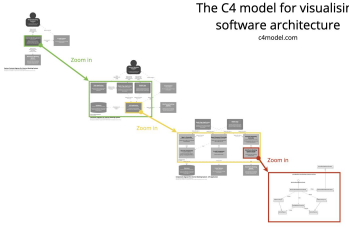




[Component] Internet Banking System - API Application

The component diagram for the API Application - diagram created with Structurizr.
Wednesday, March 22, 2023 at 8:16 AM Coordinated Universal Time

The C4 model for visualising software architecture
c4model.com



Components?

Diagramas Arquiteturais com C4 — Conceitos principais

Lucas Rogério Masotti

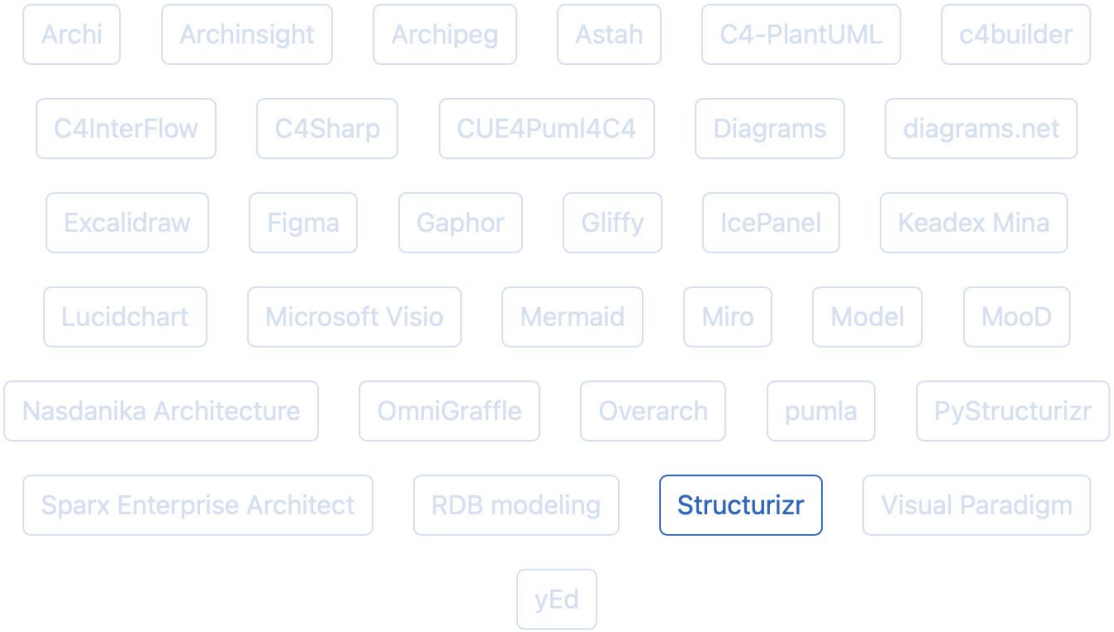
Published in Sicredi Tech · 6 min read · Mar 28, 2024

Imagine que você assumiu um sistema antigo, que ninguém mais na empresa conhece, e precisa alterar uma certa funcionalidade. O sistema é composto por vários componentes distintos e dezenas de milhares de linhas de código. Você procura por alguma documentação que possa dar uma luz de onde começar a olhar, mas não encontra nada.

Ou talvez você até encontre algo, mas é aquele desenho que mostra apenas um serviço chamando o outro, sem nenhuma contextualização do porquê, sem nenhum detalhe de tecnologias ou motivação.

Structurizr

- Diagram as code
  - Controle de versão
  - Integração com pipelines
  - Permite automatizar criação de diagramas
- Permite reúso de elementos
  - Separação model e view
- Open source



Na prática

```
implementation 'com.structurizr:structurizr-client:2.2.0'
```

```
public class StructurizrIntegration {
    public static void main(String[] args) throws Exception {
        Workspace workspace = new Workspace( name: "Event Management", description: "Modular Monolith example application
         using Simon Brown's proposal of package by component.");
        Model model = workspace.getModel();
        model.setImpliedRelationshipsStrategy(new CreateImpliedRelationshipsUnlessSameRelationshipExistsStrategy());

        Person user = model.addPerson( name: "Event Host", description: "The user that creates the event.");
        SoftwareSystem softwareSystem = model.addSoftwareSystem( name: "Event Management", description: "System responsible
         for managing events.");

        Container eventManagementFrontendContainer = softwareSystem.addContainer( name: "Web Application",
        description: "Frontend for event management", technology: "React");
        eventManagementFrontendContainer.addTags("Web");
```

* Automatização da criação de diagramas

DSL

```
workspace "Name" "Description" {

    !identifiers hierarchical

    model {
        u = person "User"
        ss = softwareSystem "Software System" {
            wa = container "Web Application"
            db = container "Database Schema" {
                tags "Database"
            }
        }

        u -> ss.wa "Uses"
        ss.wa -> ss.db "Reads from and writes to"
    }

    views {
        systemContext ss "Diagram1" {
            include *
            autolayout lr
        }

        container ss "Diagram2" {
            include *
            autolayout lr
        }

```

Structurizr

[System Context] Event Management
#SystemContext

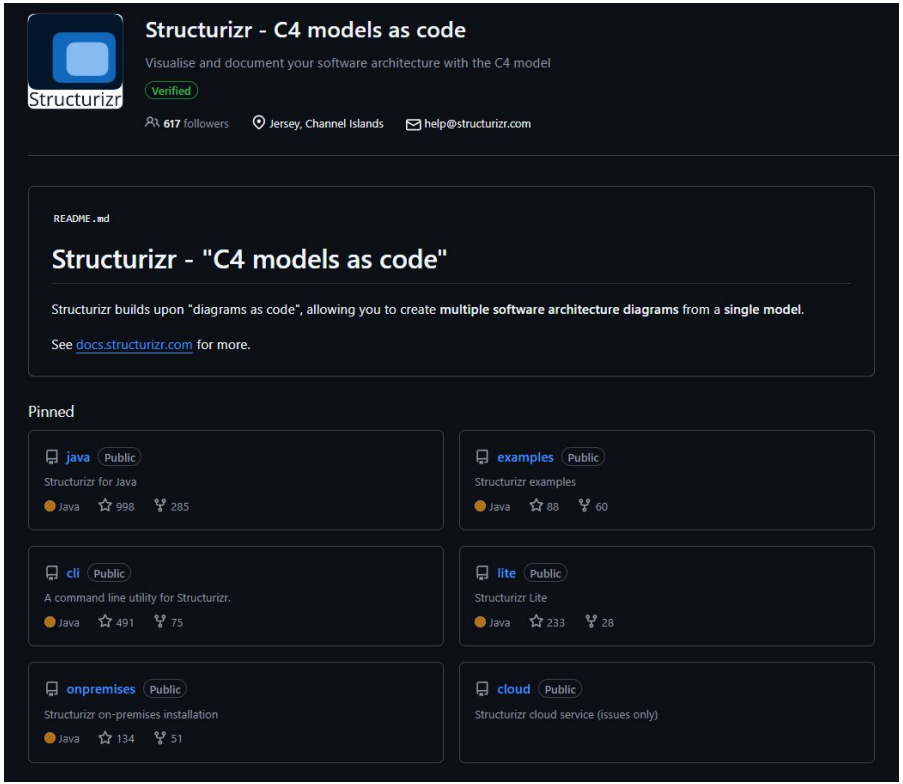
[Container] Event Management
#ContainerView

[Component] Event Management - event-management
#EventManagementComponentView

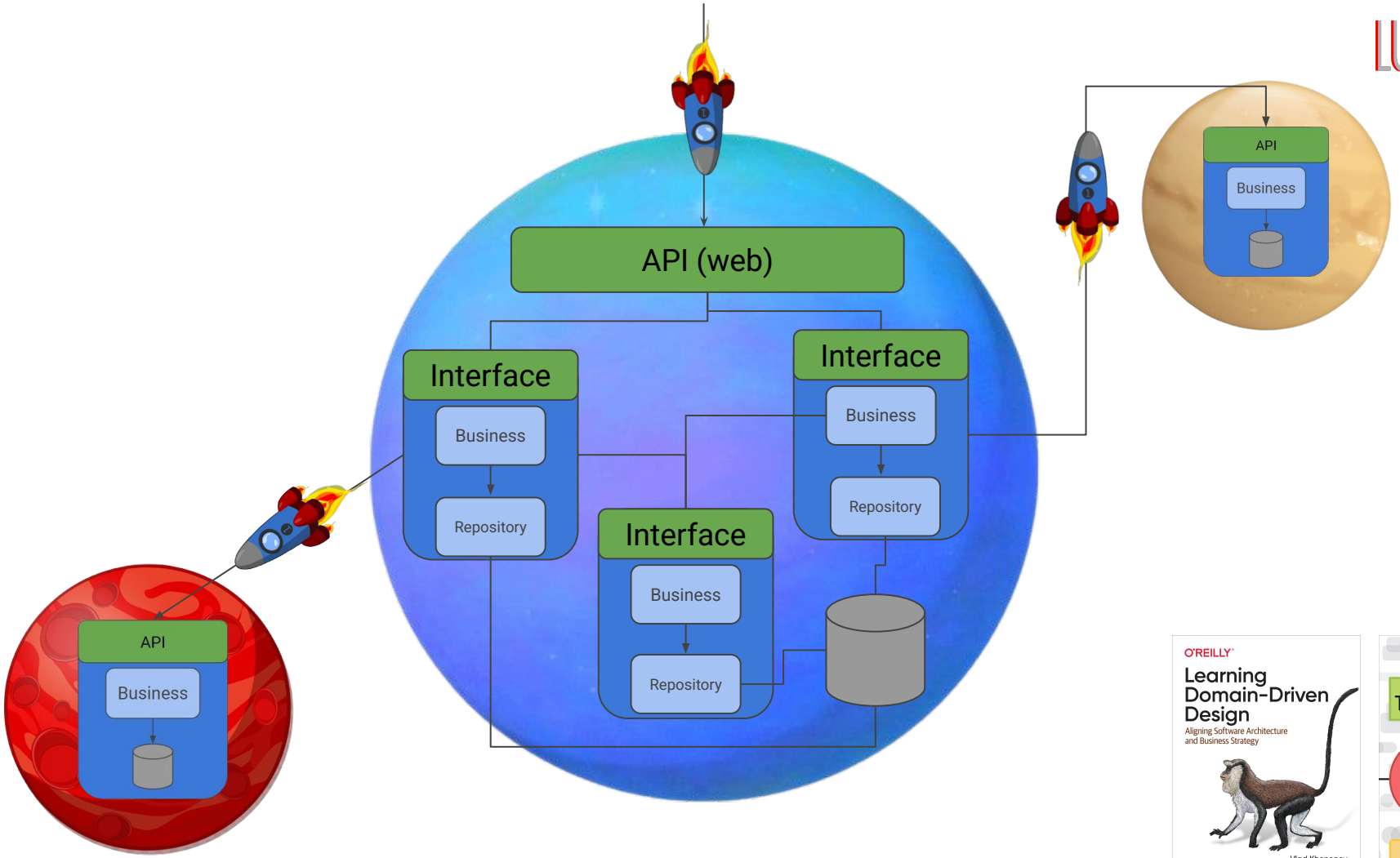
Light | Dark | System

[Container] Event Management
Container diagram
quinta-feira, 19 de setembro de 2024 às 15:42 Horário Padrão de Brasília

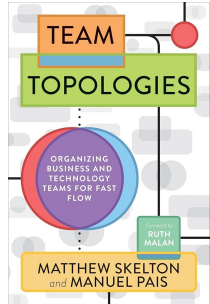
Structurizr



Big picture de arquitetura

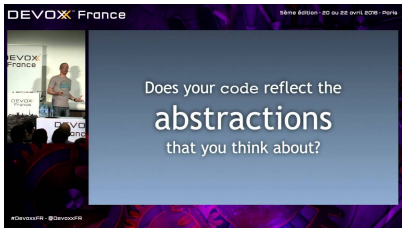

API (web)
Interface
Business
Repository
Interface
Business
Repository
Interface
Business
Repository
API
Business
API
Business

O'REILLY
Learning Domain-Driven Design
Aligning Software Architecture and Business Strategy
Vlad Khononov
Foreword by Julie Lerman

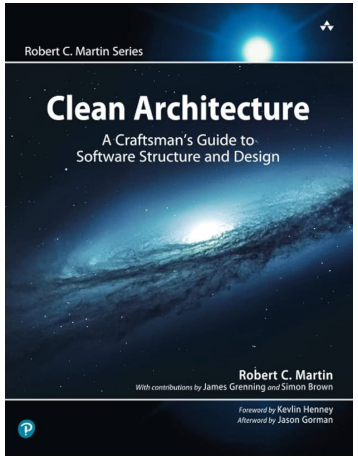

TEAM
TOPOLOGIES
ORGANIZING BUSINESS AND TECHNOLOGY TEAMS FOR FAST FLOW
RUTH MALAN
MATTHEW SKELTON
and MANUEL PAIS

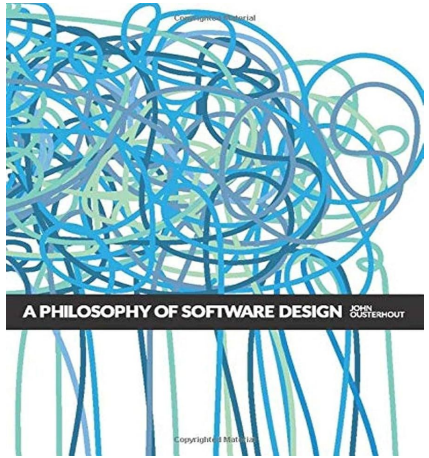
Referências e material complementar

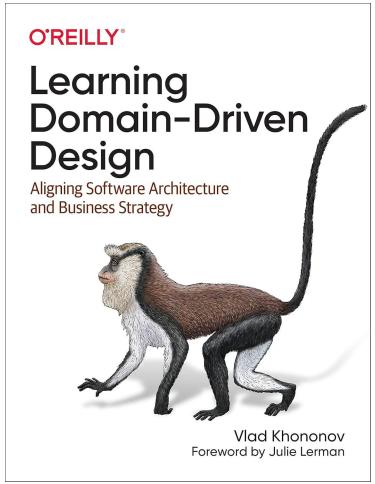
Modular Monoliths • Simon Brown • GOTO 2018



Modular monoliths (Simon Brown)



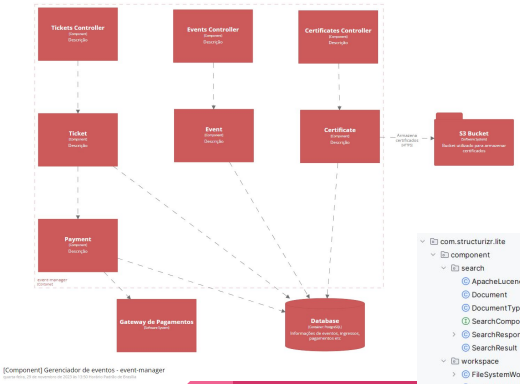


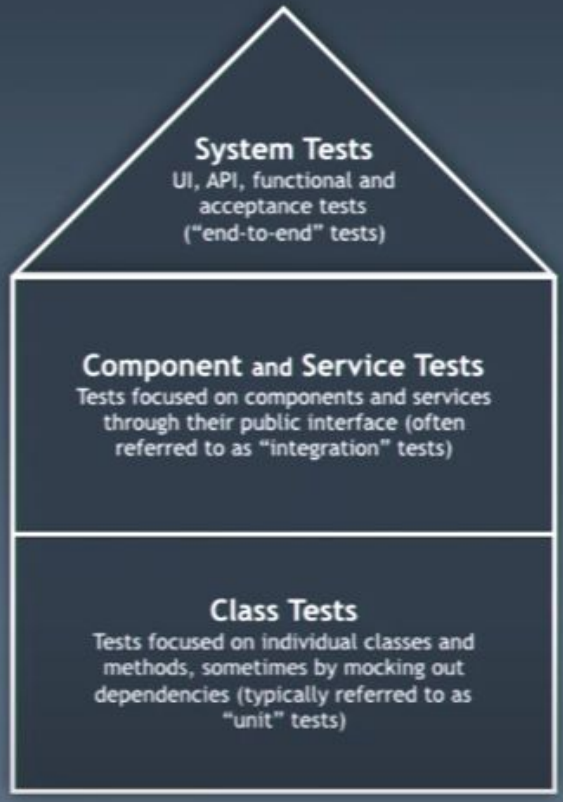


Concluindo

- Entenda os tradeoffs de uma arquitetura distribuída
- Considere Package by component
  - Alta coesão
  - Baixo acoplamento
  - Encapsula dados
  - Composable
  - **Simplicidade**
- Utilize C4 para visualizar sua arquitetura (Structurizr)
- Pare de usar tanto *public*
- **<u>Reduza a complexidade acidental!</u>**

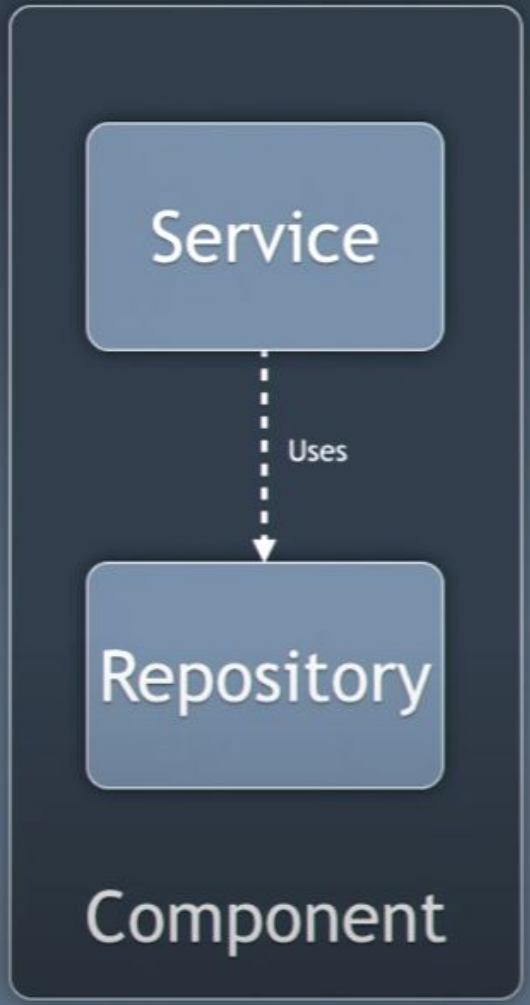
Linkedin





Architecturally-aligned testing

(and a reshaped testing pyramid)



Test the service in isolation?

("unit tests", with a mock repository)

Test the repository in isolation?

("integration tests", against the database)

Or test the component through the `public interface`?

("component tests")