

Elementos de arquitetura e engenharia no desenvolvimento Backend

Rodrigo Silva



Quem sou eu?

- Engenheiro de Software
- Java/Spring – AWS – Microserviços
- Fintechs e fluxos de pagamentos



Introdução

- **Arquitetura:** características não funcionais que auxiliam na segurança e conformidade da aplicação. Se relaciona com o design da aplicação.
- **Engenharia:** Boas práticas e organização das ferramentas que fazem parte do dia a dia da engenharia de software. Se relaciona com os processos utilizados para desenvolver, testar e manter o software.

O que iremos abordar?

Arquitetura

- Observabilidade
- Segurança
- Validações
- Documentação

Engenharia

- Tooling
- Boas práticas
- Testes
- Automatização

O que iremos abordar?

Arquitetura

- Observabilidade
- Segurança
- Validações
- Documentação

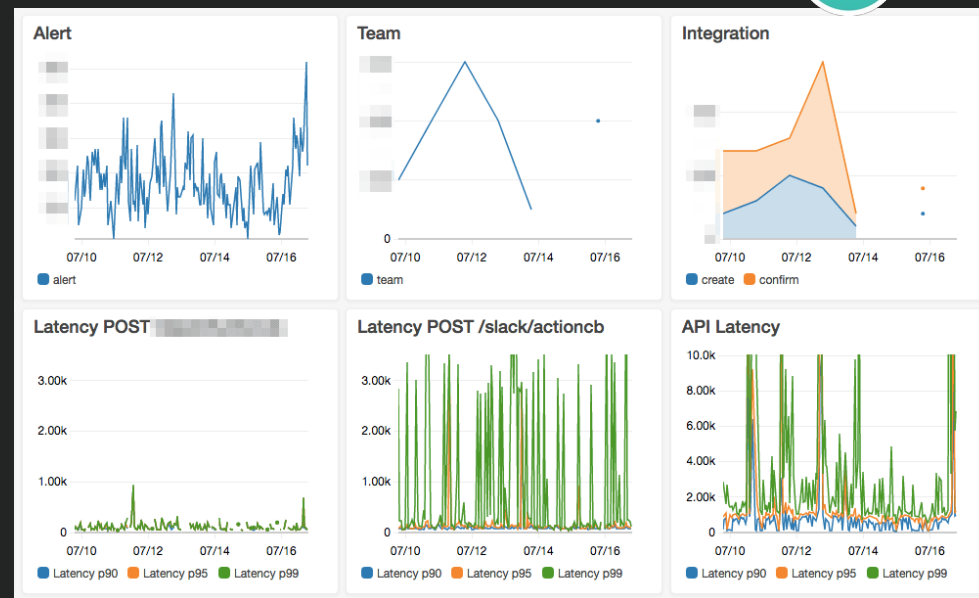
Engenharia

- Tooling
- Boas práticas
- Testes
- Automatização

Observabilidade

Precisamos de indícios seguros do funcionamento da aplicação

- Ferramentas de monitoramento.
- Logs expressivos.




Observabilidade

- Etapas do fluxo
- Dados (cuidado com dados pessoais/sensíveis)

```
INFO | Aplicação iniciou com sucesso!  
INFO | Iniciando criação de usuário...  
ERROR | Erro genérico.
```

Seu eu de amanhã irá agradecer!



```
INFO | Aplicação iniciou com sucesso!  
INFO | Iniciando criação de usuário. - request: [{ "nome": null, "idade": 23 }]  
INFO | Validando atributos do usuário...  
ERROR | Erro na criação de usuário. - message: atributos [nome] inválidos!
```

O que iremos abordar?

Arquitetura

- Observabilidade
- Segurança
- Validações
- Documentação

Engenharia

- Tooling
- Boas práticas
- Testes
- Automatização

Segurança

“Uma aplicação deve ser segura”

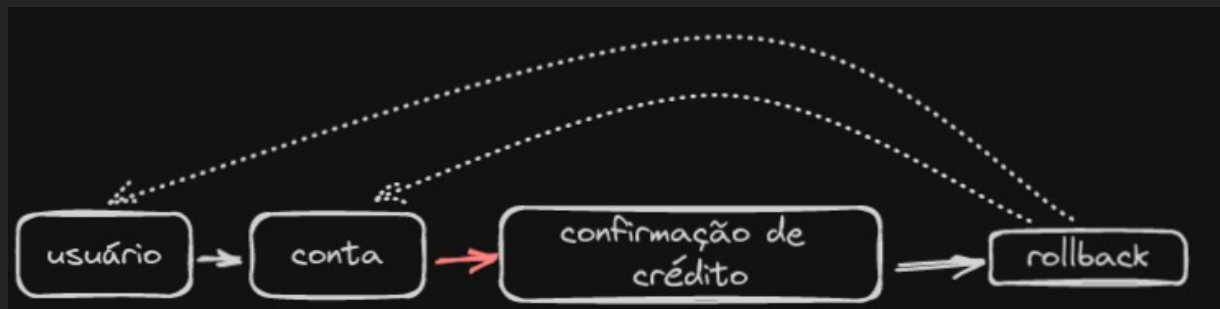
Mas o que isso significa?

- Disponível quando necessária
- Deve manter a integridade dos dados
- Princípio do menor privilégio
- (...)

Segurança

Mantendo a integridade dos dados:

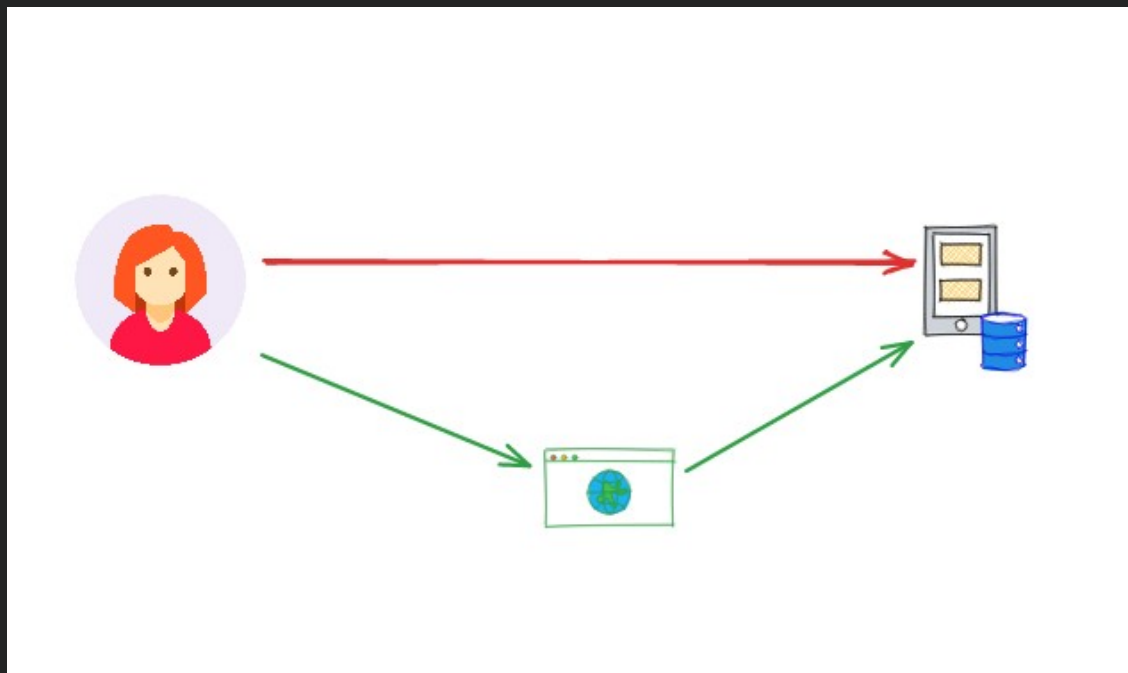
- **Transação:** conjunto de operações aplicadas de maneira atômica. Caso qualquer operação falhe, todo o conjunto é revertido.



Segurança

Princípio do menor privilégio

Recursos somente devem ser acessados por quem tem direito a eles.



O que iremos abordar?

Arquitetura

- Observabilidade
- Segurança
- Validações
- Documentação

Engenharia

- Tooling
- Boas práticas
- Testes
- Automatização

Validações

Uma aplicação deve garantir que está operando sobre dados consistentes:

- Todo o código deve ser escrito de maneira defensiva, descartando a hipótese de estar lidando com dados consistentes.

Ex: requisições duplicadas, dados incompletos.

Validações

Técnicas de validação:

- Validação por aspecto(@Valid) ou diretamente no código de atributos
- Aplicação de idempotência em cada etapa das transações de dados

Validações

```
public Produto(Long codigo, String descricao) {  
    List<String> invalid = new ArrayList<>();  
    if(Objects.isNull(codigo))  
        invalid.add("codigo");  
    if(Strings.isNullOrEmpty(descricao))  
        invalid.add("descrição");  
  
    if(!invalid.isEmpty())  
        throw new InvalidProductException(invalid);  
  
    this.codigo = codigo;  
    this.descricao = descricao;  
}
```

O que iremos abordar?

Arquitetura

- Observabilidade
- Segurança
- Validações
- Documentação

Engenharia

- Tooling
- Boas práticas
- Testes
- Automatização

Documentação

Uma aplicação deve possuir uma API bem documentada e padronizada, se for o caso:


- REST bem aplicado e de acordo com a especificação.

~~/users/create~~ → POST /users

HTTP 200 Response: [{"status": "error"}]


- OpenAPI Specification

Documentação


ReDoc

☒ CORS

20,384



[Introduction](#)

[OpenAPI Specification](#)

[Cross-Origin Resource Sharing](#)

[Authentication](#)

GENERAL

pet

New pet

Add a new pet to the store

Update an existing pet


Find pet by ID

Updates a pet in the store with form data

Deletes a pet

uploads an image

Finds Pets by status



Update an existing pet

AUTHORIZATIONS: > *petstore_auth*

COOKIE PARAMETERS

cookieParam	integer <int64>
required	Some cookie

HEADER PARAMETERS

Accept-Language	string
Default:	en-AU
Example:	en-US
The language you prefer for messages. Supported values are en-AU, en-CA, en-GB, en-US	

REQUEST BODY SCHEMA: application/json

Pet object that needs to be added to the store

category >	object
	Categories this pet belongs to
name	string
required	The name given to a pet
photoUrls	Array of strings <url>
required	<= 20 items
	The list of URL to a cute photos featuring pet
friend	object Recursive
tags >	Array of objects (Tag)
	non-empty
	Tags attached to the pet
status	string
	Enum: "available" "pending" "sold"
	Pet status in the store
petType	string
	Type of a pet
	cat

PUT /pet

Request samples

Content type

application/json

Example

cat

```

{
  "category": {
    "name": "string",
    "sub": { }
  },
  "name": "Guru",
  "photoUrls": [
    "string"
  ],
  "friend": { },
  "tags": [
    { }
  ],
  "status": "available",
  "petType": "cat",
  "huntingSkill": "adventurous"
}
    
```

O que iremos abordar?

Arquitetura

- Observabilidade
- Segurança
- Validações
- Documentação

Engenharia

- Tooling
- Boas práticas
- Testes
- Automatização

Tooling

O ambiente de trabalho deve estar bem configurado, e o(a) desenvolvedor(a) deve ter controle sobre qual versão de software está utilizando para gerar builds

Ferramentas: asdf, Docker, localstack

O que iremos abordar?

Arquitetura

- Observabilidade
- Segurança
- Validações
- Documentação

Engenharia

- Tooling
- Boas práticas
- Testes
- Automatização

Boas práticas

S.O.L.I.D.

Single Responsibility

Open-Closed

Liskov Substitution

Interface Segregation

Dependency Inversion

**Quem observar o nosso código
deve perceber logo o zelo e o
profissionalismo com o qual ele
foi escrito!**

O que iremos abordar?

Arquitetura

- Observabilidade
- Segurança
- Validações
- Documentação

Engenharia

- Tooling
- Boas práticas
- Testes
- Automatização

Testes

- Escrever e executar suites de testes automatizados não deve ser uma etapa separada, mas parte do dia a dia da engenharia de software.
- Devemos ser ambiciosos com a cobertura de teste, mas evitar escrever testes somente para fins de completa-la.

Testes

- *Tooling* bem configurado permite a execução célere e constante das suites de teste.
- Código bem modularizado facilita a escrita de testes. *Se está difícil testar, o problema é no código!*
- Testes bem escritos facilitam o *refactoring* e diminuem o débito técnico.

O que iremos abordar?

Arquitetura

- Observabilidade
- Segurança
- Validações
- Documentação

Engenharia

- Tooling
- Boas práticas
- Testes
- Automatização

Automatização

Como desenvolvedores(as), devemos ter ***horror*** a execuções repetidas de fluxos manuais.

- Se você percebe um fluxo que se repete ao longo do desenvolvimento, **automatize** e **documente**!
- Makefile, npm, gradle, shellscripts, etc

OBIGADO!