Educação Profissional Paulista

Técnico em
Desenvolvimento de Sistemas

Secretaria da Educação

SÃO PAULO
GOVERNO DO ESTADO

Linguagens de programação back-end

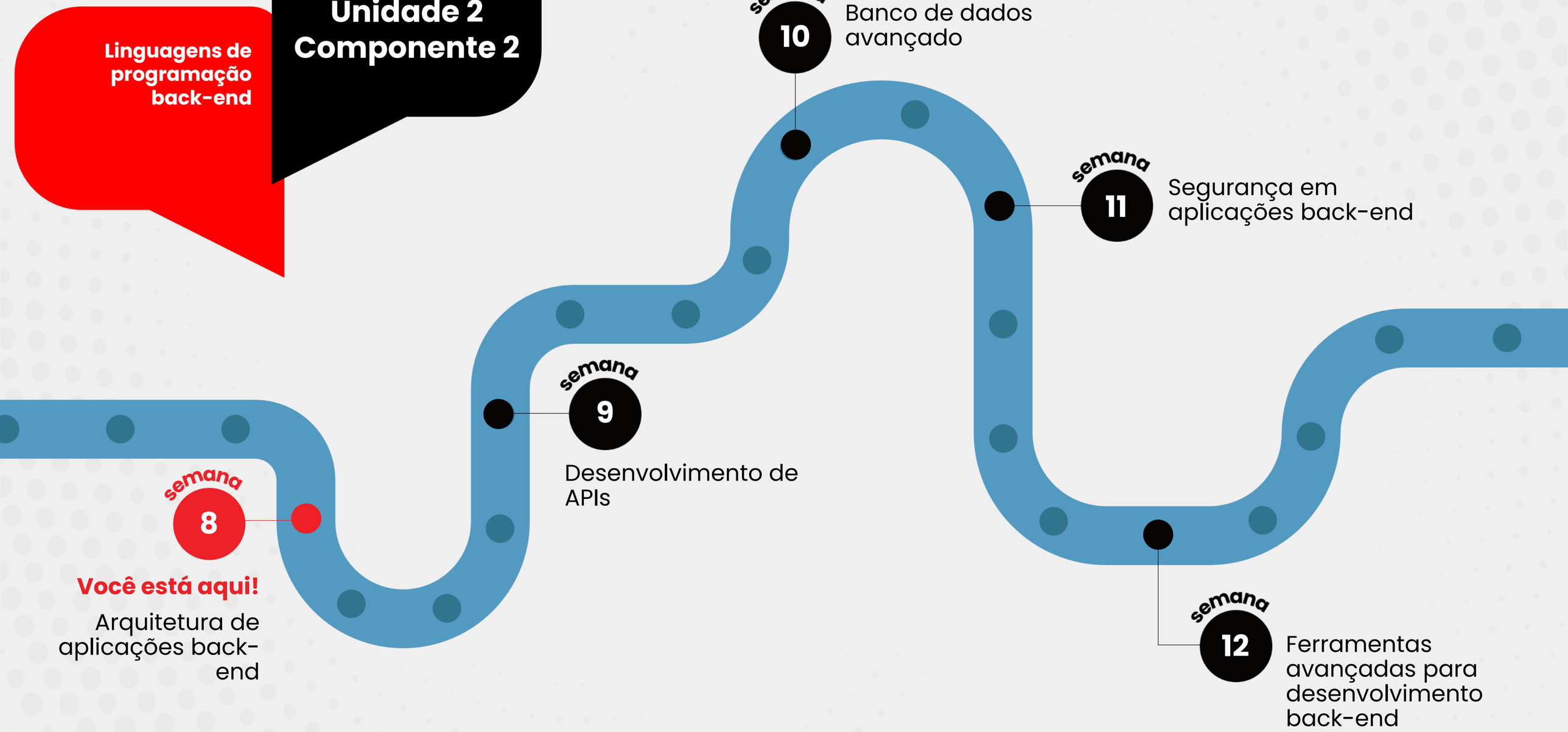
Arquitetura de aplicações back-end

Design patterns para back-end

Código da aula: [SIS]ANO2C2B2S8A2

Mapa da Unidade 2 Componente 2

Linguagens de programação back-end



Linguagens de programação back-end

Mapa da Unidade 2 Componente 2

Você está aqui!

8

Arquitetura de aplicações back-end

Aula 2: Design patterns para back-end

Código da aula: [SIS]ANO2C2B2S8A2

Objetivos da aula

- Aplicar design patterns no desenvolvimento de back-end.

Recursos didáticos

- Recurso audiovisual para exibição de vídeos e imagens;
- Lápis e caderno para anotações;
- Computador com acesso à internet.

Duração da aula

50 minutos.

Habilidades técnicas

- Saber diferenciar entre arquitetura monolítica e microsserviços.

Habilidades socioemocionais

- Desenvolver a curiosidade ao explorar diferentes arquiteturas de software.

Colocando em **prática**

Design patterns para back-end

Conceitos de design pattern para back-end

45 minutos

Em grupo de 4 pessoas

Roteiro da aula prática

Materiais necessários

- Computador com acesso à internet;
- Caderno para anotações;
- 1 caneta.

Passo a passo

1. Leia com atenção cada uma das ações propostas no documento.;
2. Antes de iniciar, realize esboços no papel para definir sua intenção e resultado;
3. Utilize as ferramentas de desenvolvimento disponíveis para sua atividade.

Baixe o roteiro dessa atividade

Registro

Pause e **responda**

Qual é a principal característica da arquitetura monolítica?

- Cada funcionalidade é um serviço independente.
- Escalabilidade fácil e independente.
- Tudo está integrado em um único bloco de código.
- Comunicação entre serviços via API.

Registro

Pause e **responda**

Qual é a principal característica da arquitetura monolítica?

- ✗ Cada funcionalidade é um serviço independente.
- ✗ Escalabilidade fácil e independente.
- ✓ Tudo está integrado em um único bloco de código.
- ✗ Comunicação entre serviços via API.

Pause e **responda**

Qual é um dos principais desafios ao migrar para uma arquitetura de microsserviços?

- Manter a aplicação em um único bloco de código.
- Garantir a comunicação eficiente entre serviços.
- Reduzir a complexidade do sistema.
- Eliminar a necessidade de escalar serviços individualmente.

Pause e **responda**

Qual é um dos principais desafios ao migrar para uma arquitetura de microsserviços?

- ✗ Manter a aplicação em um único bloco de código.
- ✓ Garantir a comunicação eficiente entre serviços.
- ✗ Reduzir a complexidade do sistema.
- ✗ Eliminar a necessidade de escalar serviços individualmente.

Registro

Pause e **responda**

Qual padrão de design é comumente usado para garantir a resiliência em uma arquitetura de microsserviços?

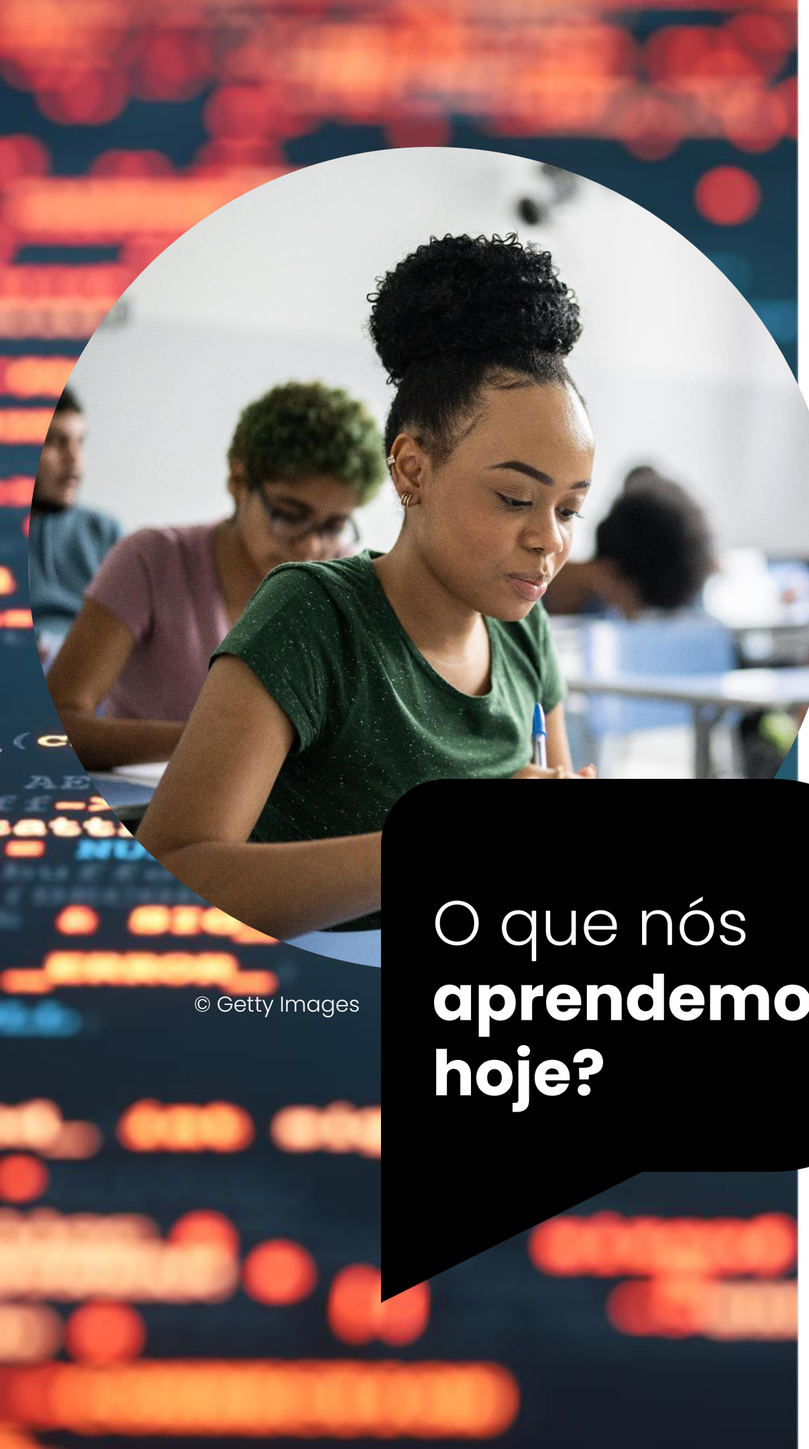
- Singleton
- Factory Method
- Circuit Breaker
- Observer

Registro

Pause e **responda**

Qual padrão de design é comumente usado para garantir a resiliência em uma arquitetura de microsserviços?

- ✗ Singleton
- ✗ Factory Method
- ✓ Circuit Breaker
- ✗ Observer



Então ficamos assim...

1. Compreendemos como a implementação de microsserviços muda a maneira como pensamos sobre a divisão do código;

2. Identificamos os desafios que surgem ao tentar refatorar o sistema para microsserviços;

3. Aprendemos como os padrões de design ajudaram a estruturar melhor o código em ambas as arquiteturas.

Referências da aula

CÓDIGO FONTE TV. **Aplicação monolítica** (a arquitetura de software mais “tradicional”) // Dicionário do Programador. Disponível em: https://www.youtube.com/watch?v=CsrHHHPHKwE. Acesso em: 7 nov. 2024.

HARRIS, C. **Microsserviços versus arquitetura monolítica**. Atlassian, [s.d.]. Disponível em: https://www.atlassian.com/br/microservices/microservices-architecture/microservices-vs-monolith. Acesso em: 7 nov. 2024.

KRÜGER, M. **Arquitetura de microsserviços x arquitetura monolítica**. Medium, 12 jul. 2022. Disponível em: Arquitetura de Microserviços x Arquitetura Monolítica | by Márcio Krüger | Medium. Acesso em: 27 dez. 2024.

VINCO. **Arquitetura de microsserviços X arquitetura monolítica**: 7 diferenças, 2019. Disponível em: https://blog.vinco.com.br/arquitetura-de-microsservicos-x-arquitetura-monolitica/. Acesso em: 7 nov. 2024.

Orientações ao professor

Slide 6

Orientações: professor, a seção **Colocando em prática** tem como objetivo aplicar os conhecimentos construídos durante a aula incentivando os estudantes a pensar criticamente e de forma prática.

Tempo: 45 minutos.

Expectativas de respostas:

1. Como a implementação de microsserviços mudou a maneira como você pensou sobre a divisão do código?
Resposta: A implementação de microsserviços exige que você pense em termos de responsabilidades distintas e isoladas para cada serviço. Em vez de agrupar todas as funcionalidades em uma única classe ou módulo, como na arquitetura monolítica, você precisa identificar claramente quais partes do sistema podem operar de forma independente e como elas irão se comunicar. Isso muda a maneira de estruturar o código, pois cada serviço deve ser coeso e autossuficiente, com uma interface bem definida para interagir com outros serviços. Essa abordagem promove um código mais modular e fácil de manter, pois as mudanças em um serviço não afetam diretamente os outros.

2. Quais desafios surgiram ao tentar refatorar o sistema para microsserviços?
Resposta: Um dos principais desafios ao refatorar o sistema para microsserviços foi gerenciar a comunicação entre os serviços. Diferente de um sistema monolítico, em que todos os componentes estão em um único espaço de memória, os microsserviços precisam se comunicar por meio de chamadas de rede, o que pode introduzir latência e falhas na comunicação. Outro desafio foi garantir a consistência dos dados entre os diferentes serviços, especialmente em operações que envolvem transações distribuídas. Além disso, a orquestração dos serviços e a necessidade de uma infraestrutura de suporte, como *service discovery* e balanceamento de carga, adicionaram complexidade ao sistema.

3. Como os padrões de design ajudaram a estruturar melhor o código em ambas as arquiteturas?
Resposta: Os padrões de design desempenharam um papel crucial em ajudar a organizar e estruturar o código de maneira mais eficiente em ambas as arquiteturas. No caso da arquitetura monolítica, padrões como o Singleton e Factory Method ajudaram a gerenciar a criação e a utilização de objetos de maneira centralizada e desacoplada, facilitando a manutenção e a evolução do código. Já na arquitetura de microsserviços, padrões como Service Registry e Circuit Breaker foram fundamentais para lidar com a complexidade da comunicação entre serviços, garantindo que o sistema permaneça resiliente e escalável. Esses padrões proporcionaram uma estrutura mais clara e modular, ajudando a reduzir a complexidade e a melhorar a reutilização e a manutenção do código.

Slide 8

Feedback AVA:

Tudo está integrado em um único bloco de código: Correta: Na arquitetura monolítica, todos os componentes e funcionalidades estão em um único bloco de código, facilitando o desenvolvimento inicial, mas limitando a escalabilidade.

Feedback geral:

Cada funcionalidade é um serviço independente: Incorreta. Na arquitetura monolítica, todas as funcionalidades estão integradas em um único bloco de código, não separadas em serviços independentes.

Escalabilidade fácil e independente: Incorreta: A escalabilidade na arquitetura monolítica não é independente, é preciso escalar todo o sistema como um único bloco.

Tudo está integrado em um único bloco de código: Correta: Na arquitetura monolítica, todos os componentes e funcionalidades estão em um único bloco de código, facilitando o desenvolvimento inicial, mas limitando a escalabilidade.

Comunicação entre serviços via API: Incorreta. A comunicação entre serviços via API é uma característica típica da arquitetura de microsserviços, não da monolítica.

.

Slide 10

Feedback AVA:

Garantir a comunicação eficiente entre serviços: Correta. Um dos principais desafios ao migrar para microsserviços é garantir a comunicação eficiente entre os diferentes serviços, devido à sua natureza distribuída.

Feedback geral:

Manter a aplicação em um único bloco de código: Incorreta. Manter a aplicação em um único bloco de código é um desafio da arquitetura monolítica, não dos microsserviços.

Garantir a comunicação eficiente entre serviços: Correta. Um dos principais desafios ao migrar para microsserviços é garantir a comunicação eficiente entre os diferentes serviços, devido à sua natureza distribuída.

Reduzir a complexidade do sistema: Incorreta. A arquitetura de microsserviços geralmente aumenta a complexidade do sistema, especialmente em termos de orquestração e gestão de serviços.

Eliminar a necessidade de escalar serviços individualmente: Incorreta. A arquitetura de microsserviços permite a escalabilidade independente dos serviços, que é uma vantagem, não um desafio.

Slide 12

Feedback AVA:

Circuit Breaker: Correta. O Circuit Breaker é um padrão de design utilizado em arquitetura de microsserviços para garantir a resiliência, interrompendo a comunicação com serviços que apresentam falhas, evitando que o sistema como um todo seja afetado.

Feedback geral:

Singleton: Incorreta. O padrão Singleton é mais comumente usados em arquitetura monolítica para garantir que uma classe tenha apenas uma instância.

Factory Method: Incorreta. O Factory Method é usado para criar objetos sem especificar a classe exata, mas não está diretamente relacionado à resiliência em microsserviços.

Circuit Breaker: Correta. O Circuit Breaker é um padrão de design utilizado em arquitetura de microsserviços para garantir a resiliência, interrompendo a comunicação com serviços que apresentam falhas, evitando que o sistema como um todo seja afetado.

Observer: Incorreta. O padrão Observer é mais relacionado a notificações e reatividade entre componentes, mas não especificamente à resiliência em microsserviços.

Tempo total para o quiz: 3 minutos.

Slide 13

Orientações: professor, a seção **O que nós aprendemos hoje?** tem o objetivo de reforçar e esclarecer os conceitos principais discutidos na aula. Essa revisão pode ser uma ferramenta de avaliação informal do aprendizado dos estudantes, identificando áreas que podem precisar de mais atenção em aulas futuras.

Tempo: 2 minutos.

Gestão de sala de aula:

- Mantenha um tom positivo e construtivo, reforçando o aprendizado em vez de focar as correções;
- Seja direto e objetivo nas explicações para manter a atividade dentro do tempo estipulado;
- Engaje os estudantes rapidamente, pedindo confirmações ou reações breves às definições apresentadas.

Condução da dinâmica:

- Explique que esta parte da seção, "Então ficamos assim...", é um momento de reflexão e esclarecimento sobre os conceitos abordados na aula;
- Informe que será uma rápida revisão para assegurar que os entendimentos dos estudantes estão alinhados com as definições corretas dos conceitos;
- Apresente o slide com a definição sintética de cada conceito principal discutido na aula, ampliando em forma de frases completas;
- Finalize resumindo os pontos principais e reiterando a importância de cada conceito e como ele se encaixa no contexto maior da aula;
- Reforce a ideia de que essa revisão ajuda a solidificar o entendimento dos estudantes e prepará-los para aplicar esses conceitos em situações práticas.

Expectativas de respostas:

Os estudantes devem sair da aula com um entendimento claro e preciso dos conceitos principais. A atividade serve como uma verificação rápida do entendimento dos estudantes e uma oportunidade para corrigir quaisquer mal-entendidos.

Educação Profissional Paulista

Técnico em
Desenvolvimento de Sistemas

Secretaria da Educação

SÃO PAULO
GOVERNO DO ESTADO