THIAGO ADRIANO E LEANDRO ANDRADE

PÓS TECH

FULL STACK DEVELOPMENT

LOW CODE DEVELOPMENT

AULA 03

SUMÁRIO

O QUE VEM POR AÍ?

Nesta aula, mergulharemos na criação de uma aplicação completa de backend utilizando a plataforma Outsystems. Nosso objetivo é não apenas construir funcionalidades robustas, mas também disponibilizar o Swagger da nossa aplicação. Exploraremos todos os recursos disponíveis na plataforma para desenvolver um backend sólido, e o destaque será a integração do Swagger.

HANDS ON

Nesta aula prática, mergulharemos no desenvolvimento do backend de uma aplicação com disponibilização de serviços para a camada de fronte e também desenvolvimento de uma API para disponibilizar os dados para outras aplicações. Os docentes demonstrarão como criar a sua aplicação backend, como construir a API e como documentá-la com o Swagger.

SAIBA MAIS

A OutSystems é uma plataforma fullstack, ou seja, com ela conseguimos criar aplicações por completo, não apenas frontend ou backend, e dentro do Service Studio está tudo junto, mas é uma boa prática de arquitetura fazer a separação de responsabilidade. Existe, inclusive, uma boa prática de arquitetura de aplicações OutSystems que se chama “Arquitetura Canvas”.

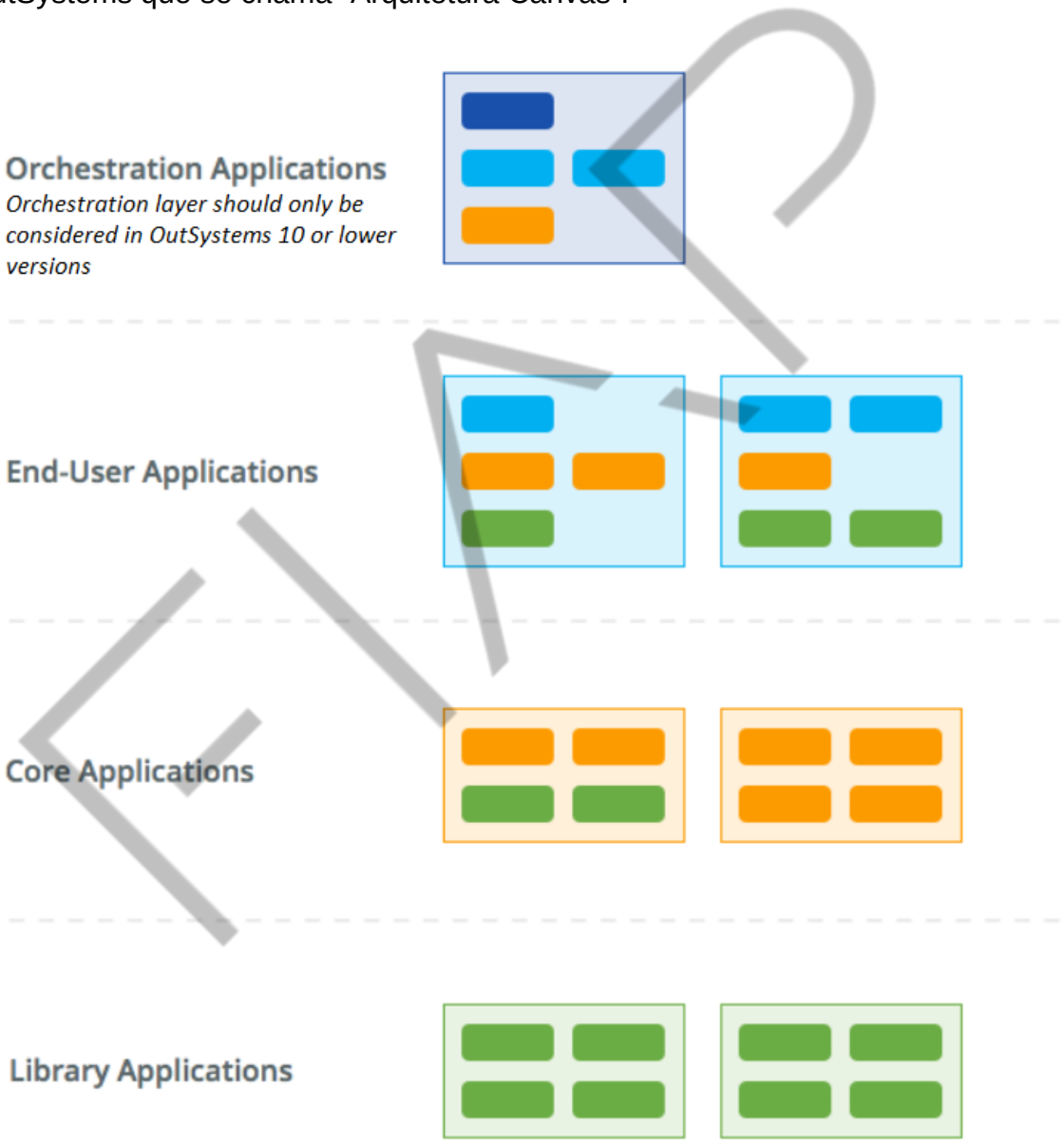

Figura 1 - Arquitetura Canvas
Fonte: Success Outsystems (2023)

O Canvas é um padrão de arquitetura da OutSystems projetada para simplificar o design de Arquiteturas Orientadas a Serviços (SOA). Ela promove a correta

abstração de serviços (micro)reutilizáveis e o isolamento adequado de módulos funcionais distintos, especialmente em situações em que você está desenvolvendo e mantendo várias aplicações que compartilham módulos comuns. Uma instalação típica de médio a grande porte da OutSystems pode dar suporte a mais de 20 aplicações críticas para a missão e mais de 200 módulos interdependentes.

Essas aplicações e módulos têm ciclos de vida de mudança diferentes e são mantidos e patrocinados por equipes diferentes. Novas aplicações tendem a evoluir rapidamente, enquanto serviços altamente reutilizados mudarão com menos frequência. O benefício mais significativo de uma arquitetura bem projetada é que as aplicações e os módulos que as compõem preservarão ciclos de vida independentes, reduzindo ao mínimo as dependências e o impacto geral das mudanças. O resultado é uma arquitetura OutSystems eficiente em termos de custos, mais fácil de manter e evoluir.

Você pode conhecer um pouco mais acessando o site da OutSystems.

Em nosso Hands On, visto anteriormente, nós simplificamos essa construção separando a aplicação em duas partes (backend e frontend).

Service Action

Na plataforma OutSystems, uma "Service Action" representa uma chamada remota baseada em REST para outro processo, contudo, sua aplicação assemelha-se de maneira marcante às "Server Actions" públicas. Ao expor uma "Service Action", cria-se uma dependência flexível da pessoa consumidora para o módulo produtor, estabelecendo uma relação de acoplamento solto.

Diferença entre Service Action e Server Action

Na plataforma OutSystems, tanto a "Service Action", quanto a "Server Action", são elementos utilizados para encapsular lógica de negócios ou funcionalidades específicas, mas existem algumas diferenças fundamentais entre elas:

Tipo de chamada

- **Service Action:** é utilizada para realizar chamadas remotas baseadas em REST para outros processos, geralmente em módulos diferentes. Ela facilita a comunicação entre módulos de forma desacoplada, permitindo uma integração mais flexível.
- **Server Action:** geralmente é utilizada para encapsular lógica de negócios ou operações no mesmo módulo onde está sendo desenvolvida. As "Server Actions" são frequentemente utilizadas para operações locais e são chamadas de forma síncrona.

Escopo e Localização

- **Service Action:** pode ser exposta e consumida por outros módulos, tornando-se uma maneira eficiente de criar serviços reutilizáveis e compartilháveis entre diferentes partes de uma aplicação.
- **Server Action:** geralmente é local para o módulo onde foi criada, sendo utilizada para encapsular e organizar a lógica de negócios específica desse módulo.

Integração e Reusabilidade

- **Service Action:** é mais voltada para a integração entre módulos, promovendo a reusabilidade de serviços em toda a aplicação.
- **Server Action:** é utilizada para encapsular lógica específica de um módulo, proporcionando organização e modularidade dentro desse contexto.

API

O termo API, sigla em inglês para 'Application Programming Interface', refere-se a uma interface de aplicações que possibilita a comunicação entre diferentes sistemas. Ao contrário das funções nos códigos, o objetivo das APIs é permitir essa comunicação sem exigir a compreensão detalhada ou a necessidade de modificar o sistema ao qual se conecta.

Imagine um sistema de vendas que realiza suas operações, como vendas, controle de estoque e emissão de notas. Suponha que, após um tempo de uso, seja necessário consultar as restrições de crédito de um comprador durante uma venda a prazo. Ao invés de desenvolver todo o processo de consulta no sistema de vendas, é possível simplesmente integrar uma API de um serviço especializado em consultas de crédito, sem precisar entender completamente como ele realiza essas consultas.

Portanto, uma API consiste em um conjunto de rotinas e padrões documentados por uma aplicação, permitindo que outras aplicações utilizem suas funcionalidades sem necessariamente conhecerem os detalhes de sua implementação. Essa interoperabilidade entre aplicações facilita a comunicação entre elas e entre os usuários e usuárias.

Protocolos de API referem-se aos procedimentos e convenções que estabelecem padrões para essa comunicação entre sistemas.

Dois dos protocolos mais comuns são o REST e SOAP: REST é a sigla para 'Representational State Transfer', e é um dos protocolos mais populares para o desenvolvimento de APIs. Ele adota uma arquitetura cliente/servidor sem estado, separando o frontend do backend da API, oferecendo agilidade e flexibilidade na implementação. Apesar de não armazenar dados entre requisições, os APIs RESTful conseguem armazenar dados em cache para otimização.

A seguir você tem uma imagem demonstrando este protocolo:

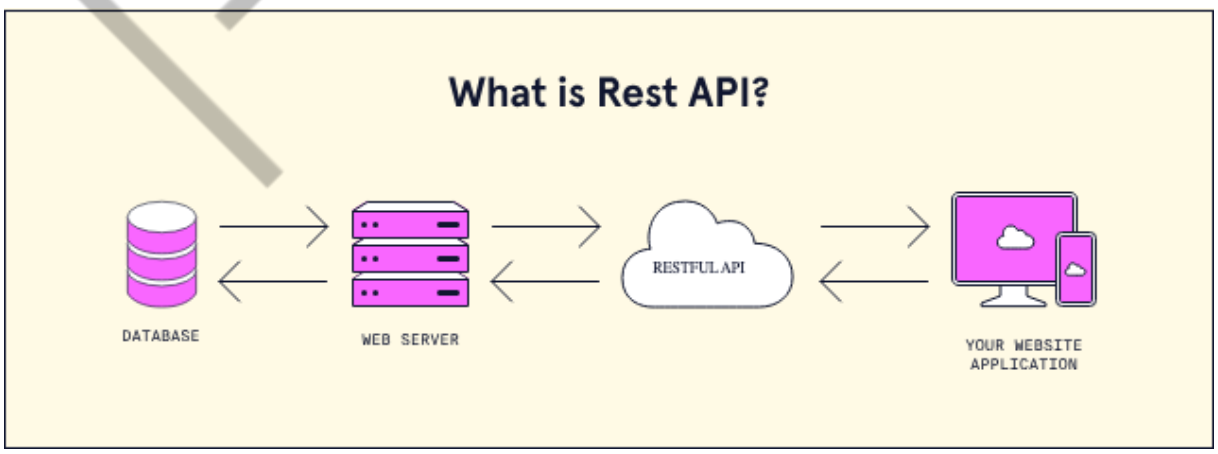

Figura 2 - Protocolo rest
Fonte: Code Academy (s.d.)

SOAP, 'Simple Object Access Protocol', é conhecido por sua extensibilidade e independência de estilos de programação. Este protocolo é neutro e pode operar por meio de diferentes protocolos de comunicação, facilitando a adição de funcionalidades ao código. A seguir você tem uma imagem demonstrando este protocolo:

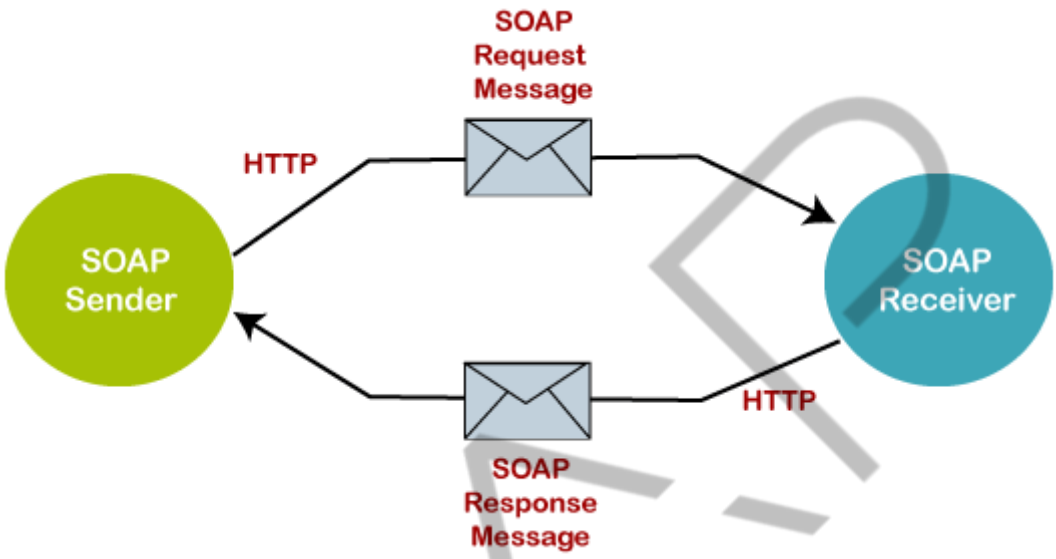

Figura 3 - Protocolo SOAP
Fonte: Java T Point (s.d.)

Métodos de uso de API incluem GET (consulta de dados), POST (envio de dados), PUT (atualização de dados) e DELETE (exclusão de dados), cada um utilizado para operações específicas na base de dados.

No contexto da plataforma OutSystems, a utilização de APIs REST e SOAP é simplificada, permitindo a exposição e consumo de funcionalidades sem a necessidade de codificação manual.

O processo de consumo e exposição de serviços segue padrões específicos para cada protocolo, permitindo a interação visual e facilitando a integração entre sistemas.

O QUE VOCÊ VIU NESTA AULA?

Nesta aula, você aprendeu a criar a camada de backend e uma API utilizando a plataforma Outsystems. Nosso objetivo foi demonstrar como separar a aplicação em camadas e como expor API da nossa aplicação para que os(as) nossos(as) parceiros(as) possam se integrar com a nossa aplicação.

REFERÊNCIAS

OutSystems. **The Architecture Canvas**. Disponível em: <https://success.outsystems.com/documentation/best_practices/architecture/designing_the_architecture_of_your_outsystems_applications/the_architecture_canvas/>. Acesso em: 06 fev. 2024.

OutSystems. **Applying the Architecture Canvas to applications**. Disponível em: <https://success.outsystems.com/documentation/best_practices/architecture/designing_the_architecture_of_your_outsystems_applications/application_composition/applying_the_architecture_canvas_to_applications/>. Acesso em: 06 fev. 2024.

Restfulapi. **What is REST**. Disponível em: <https://restfulapi.net/>. Acesso em: 06 fev. 2024.

Techtudo. **O que é API e para que serve? Cinco perguntas e respostas.** Disponível em: <https://www.techtudo.com.br/listas/2020/06/o-que-e-api-e-para-que-serve-cinco-perguntas-e-respostas.ghtml>. Acesso em: 06 fev. 2024.

Terralab. **Documentando sua API Rest com Swagger**. Disponível em: <https://www2.decom.ufop.br/terralab/documentando-sua-api-rest-com-swagger/>. Acesso em: 06 fev. 2024.

PALAVRAS-CHAVE

Arquitetura Canvas, API, Outsystems, LowCode, Swagger.