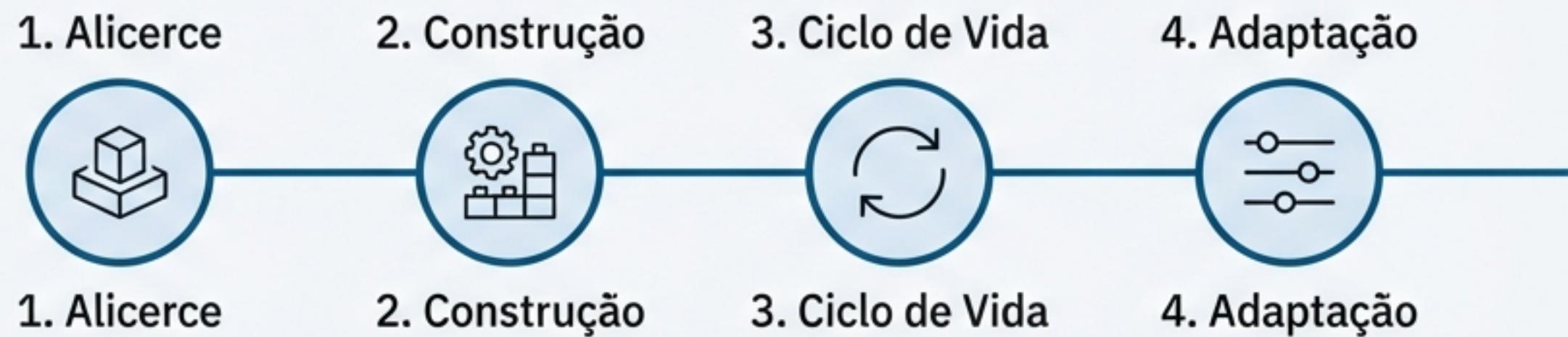


A Jornada do Desenvolvedor com CAP Java

Um guia completo do ciclo de vida:
da arquitetura à nuvem.



Esta apresentação mapeia o processo de criação de uma aplicação com o **Cloud Application Programming Model (CAP)** para Java. Vamos explorar cada etapa, desde os princípios arquitetônicos até as melhores práticas de configuração, capacitando você a construir, testar e implantar aplicações de forma eficiente e robusta.

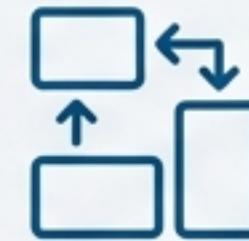
O Alicerce: Flexibilidade Através da Modularidade

"Opinionated but open"

O framework oferece um caminho claro com tecnologias de ponta, mas mantém a porta aberta para escolhas personalizadas.

Platform and service agnostic"

A funcionalidade é separada em componentes independentes e desacoplados.



Componentes Intercambiáveis

A troca de um serviço de persistência ou a adaptação a uma nova plataforma cloud torna-se uma questão de configuração, não de reescrita de código.



Redução de Custos

A adaptação a novos requisitos de plataforma ou serviços é simplificada, diminuindo o risco de alterações custosas no código customizado.



Agnosticismo de Código

“O código customizado não precisa ser escrito contra o tipo escolhido de serviço de persistência, mas pode usar o serviço de persistência genérico baseado em CQL.”

Anatomia de uma Aplicação CAP

1. Application Framework (Spring Boot, Java Nativo):

Fornece a base da sua aplicação web, como um contêiner de tempo de execução para o seu código de negócio. Lida com tarefas comuns como processamento de endpoints HTTP, configuração e logging.

2. Protocol Adapters (OData V4, Custom):

Mapeiam eventos web específicos de protocolos em eventos CQN para processamento posterior. O adaptador OData V4 é o mais proeminente.

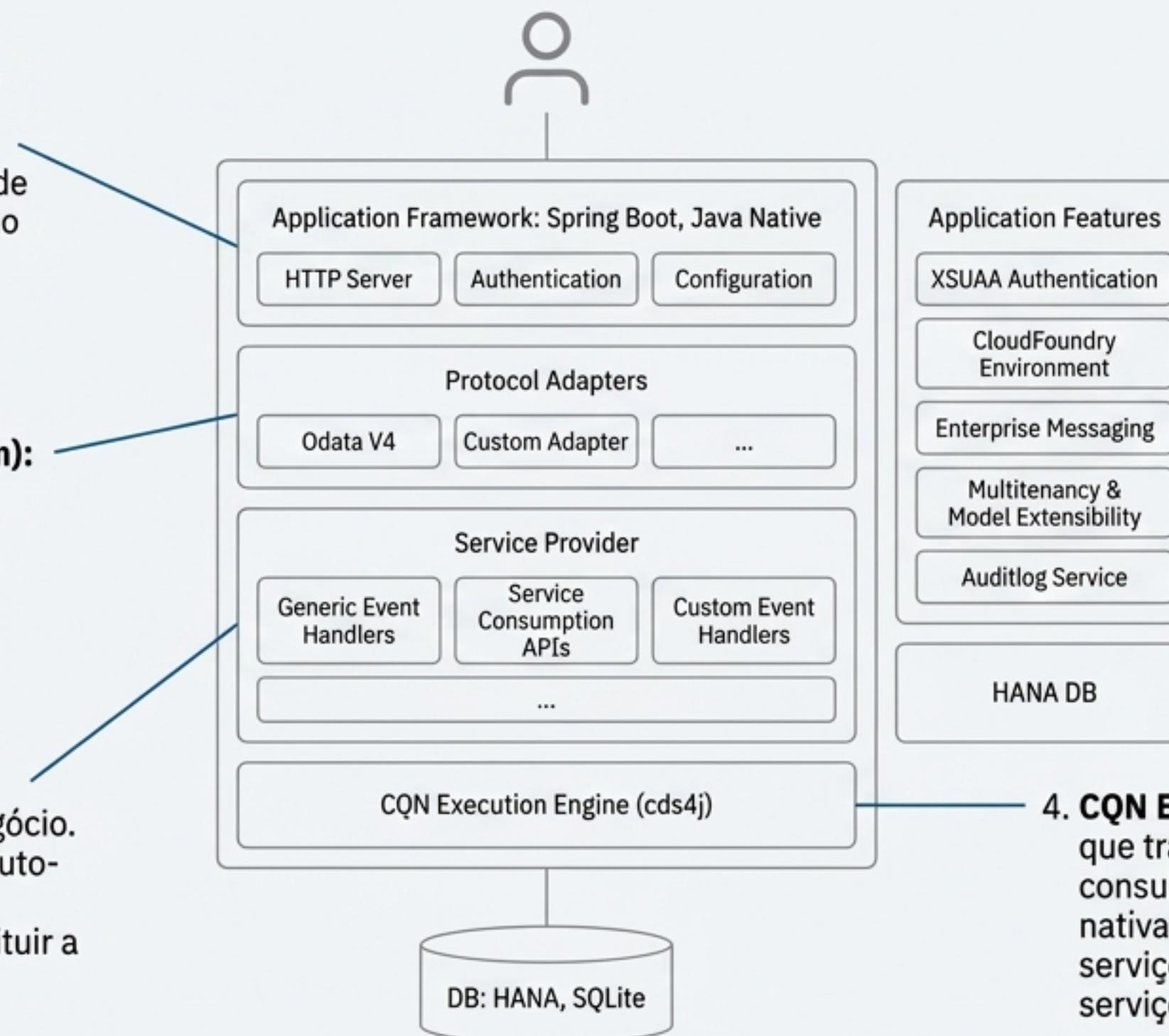
3. Service Provider (Generic & Custom Handlers):

O coração da lógica de negócio. Contém todos os serviços genéricos auto-expostos pelo CAP, além de handlers customizados para estender ou substituir a lógica padrão.

5. Application Features (Multitenancy, Auditlog):

Extensões opcionais (plugins) que adicionam capacidades como multitenancy, integração com serviços de plataforma (XSUAA, Enterprise Messaging) ou auditoria.

4. CQN Execution Engine (cds4j): O motor que traduz declarações CQN (a linguagem de consulta nativa do CAP) em declarações nativas de um destino de dados, como um serviço de persistência (HANA, H2) ou um serviço remoto.



A Construção: Estruturando seu Projeto com Maven

Sincronizando Versões com o "Bill of Materials" (BOM)

O artefato `cds-services-bom` ajuda a controlar as versões de todos os módulos CAP, garantindo que estejam em sincronia. É altamente recomendado declará-lo na seção `dependencyManagement` do seu `pom.xml` pai.

```
<properties>
    <cds.services.version>2.6.0</cds.services.version>
</properties>

<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>com.sap.cds</groupId>
            <artifactId>cds-services-bom</artifactId>
            <version>${cds.services.version}</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
```

As Dependências Essenciais

Seu `pom.xml` precisa declarar os módulos necessários para executar a aplicação web. As dependências mínimas cobrem o framework da aplicação, um adaptador de protocolo e o runtime do CAP.

```
<dependencies>
    <!-- 1. Application Framework (Recomendado) -->
    <dependency>
        <groupId>com.sap.cds</groupId>
        <artifactId>cds-framework-spring-boot</artifactId>
        <scope>runtime</scope>
    </dependency>

    <!-- 2. Protocol Adapter -->
    <dependency>
        <groupId>com.sap.cds</groupId>
        <artifactId>cds-adapter-odata-v4</artifactId>
        <scope>runtime</scope>
    </dependency>

    <!-- 3. CAP Java SDK (Core Runtime) -->
    <dependency>
        <groupId>com.sap.cds</groupId>
        <artifactId>cds-services-api</artifactId>
    </dependency>
    <dependency>
        <groupId>com.sap.cds</groupId>
        <artifactId>cds-services-impl</artifactId>
        <scope>runtime</scope>
    </dependency>
</dependencies>
```

-
1. **Application Framework (Recomendado):** Integração total com Spring, injeção de dependência e auto-configuração.
2. **Protocol Adapter:** Expõe automaticamente seus serviços como endpoints OData V4.
3. **CAP Java SDK (Core Runtime):** A API e a implementação principal do runtime do CAP Java.

Acelerando o Desenvolvimento com o CDS Maven Plugin

Geração de Código para Acesso Tipado
(Type-Safe Access)

O goal `generate` do `CDS Maven Plugin` cria interfaces Java a partir dos seus modelos CDS. Isso permite acesso aos dados com segurança de tipos, reduzindo erros e melhorando a legibilidade do código.

Configuração (pom.xml)

```
<execution>
  <id>cds.generate</id>
  <goals>
    <goal>generate</goal>
  </goals>
  <configuration>
    <basePackage>cds.gen</basePackage>
  </configuration>
</execution>
```

Estilo `BEAN` (Padrão)

```
Authors author = Authors.create();
author.setName("Emily Brontë");
Books book = Books.create();
book.setAuthor(author);
book.setTitle("Wuthering Heights");
```

Estilo `FLUENT` (Configurável)

```
Authors author = Authors.create().name("Emily Brontë");
Books.create().author(author).title("Wuthering Heights");
```

O Ciclo de Vida: Execução e Depuração Ágil

Ferramentas para otimizar o ‘inner loop’ de desenvolvimento



Para o Terminal (`mvn cds:watch`)

Executa o build do CDS e inicia sua aplicação.

Em conjunto com o **Spring Boot Devtools**, detecta alterações no modelo CDS ou em classes Java e reinicia a aplicação automaticamente.

No Windows, o `watch` só funciona se o Spring Boot Devtools estiver habilitado.



Para a IDE (`mvn cds:auto-build`)

Execute este comando em um terminal separado. Ele reage a qualquer mudança em arquivos CDS e realiza o rebuild do modelo.

Você pode então reiniciar a aplicação diretamente na sua IDE (Run/Debug) para que as alterações do modelo sejam aplicadas.

Dica: Adicione o **Spring Boot Devtools** ao seu projeto. Qualquer alteração em classes Java ou recursos no classpath (como `application.yaml`) acionará uma recarga automática do contexto da aplicação, muito mais rápido que um reinício completo.

Construindo Confiança: Uma Estratégia de Testes em Camadas



A arquitetura desacoplada do CAP permite definir com precisão o escopo dos seus testes. Ao focar em diferentes camadas, você pode construir uma suíte de testes robusta e eficiente.

Ferramentas Recomendadas

Ferramentas: Aproveite a conveniência do framework Spring para testes, que oferece injeção de dependências, [MockMvc](#) e usuários mockados, além de bibliotecas como [JUnit](#) e [Mockito](#).

Testando a Lógica de Negócio: Camadas de Handler e Serviço

1. Teste de Unidade (EventHandler Layer)

Objetivo: Verificar a lógica pura do handler, isolando-a de dependências externas como o banco de dados.

```
@ExtendWith(MockitoExtension.class)
public class CatalogServiceHandlerTest {
    @Mock
    private PersistenceService db; // Mock da dependência

    @Test
    public void discountBooks() {
        Books book = Books.create();
        book.setTitle("Book 2");
        book.setStock(200);

        CatalogServiceHandler handler = new CatalogServiceHandler(db);
        handler.discountBooks(Stream.of(book)); // Invocação direta

        assertEquals("Book 2 -- 11% discount", book.getTitle());
    }
}
```



2. Teste de Serviço (Service Layer)

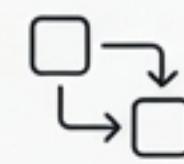
Objetivo: Interagir diretamente com a API do serviço para validar o comportamento de um evento, sem passar pela camada de protocolo.

```
@SpringBootTest
public class CatalogServiceTest {
    @Autowired
    private CqnService catalogService;

    @Test
    public void submitOrder() {
        SubmitOrderContext context = SubmitOrderContext.create();
        context.setBook("4a519e61-3c3a-4bd9-ab12-d7e8c5329933");
        context.setQuantity(2);

        catalogService.emit(context); // Dispara o evento no serviço

        assertEquals(20, context.getResult().stoStock());
    }
}
```



Validando o Fluxo Completo: Teste de Integração

Objetivo: Verificar o comportamento de um 'roundtrip' completo, começando na camada de adaptador de protocolo (OData), passando por toda a arquitetura CAP, e retornando a resposta.

Ferramenta Chave:
MockMvc do Spring Test



Exemplo de Código (GET /Books)

```
@SpringBootTest
@AutoConfigureMockMvc
public class CatalogServiceITest {
    @Autowired
    private MockMvc mockMvc;

    @Test
    public void discountApplied() throws Exception {
        // Simula uma requisição GET ao endpoint OData dos Livros
        mockMvc.perform(get("/api/browse/Books?$filter=stock gt 200&top=1"))
            .andExpect(status().isOk())
        // Valida que o título na resposta JSON contém o desconto
        .andExpect(jsonPath("$.value[0].title").value(containsString("-- 11% discount")));
    }
}
```

Banco de Dados

Para desenvolvimento e testes locais, o H2 é configurado automaticamente ao criar um projeto com o Maven Archetype, oferecendo um ambiente rápido e em memória.



A Adaptação: Configurando para Múltiplos Cenários

Inter SemiBold, Inter Sening Boot

O Poder dos Perfis do Spring Boot

O CAP Java utiliza plenamente os recursos de configuração do Spring Boot. Isso permite que você defina múltiplos perfis para diferentes cenários, como desenvolvimento local e implantação na nuvem, no mesmo arquivo `application.yaml`.

Para desenvolvimento local

IBM Plex Sans Regular

```
spring:  
  config.activate.on-profile: default  
  sql.init.platform: h2 # Usa o banco de  
 dados em memória H2  
  # ...  
  cds.security.mock.enabled: true # Habilita  
 usuários mockados
```

Para produção

IBM Plex Sans Regular

```
---
```

```
spring:  
  config.activate.on-profile: cloud  
  # ... (configurações para HANA, XSUAA, etc.  
  #       seriam injetadas pelo ambiente)  
  # ... O perfil de produção desativa  
  #       funcionalidades de desenvolvimento
```

Protegendo sua Aplicação: O Perfil de Produção

Ao ativar um perfil de produção (o padrão é `cloud`), o CAP ajusta automaticamente um conjunto de propriedades para configurações recomendadas em implantações produtivas.

Funcionalidades Desativadas por Padrão no Perfil `cloud`:

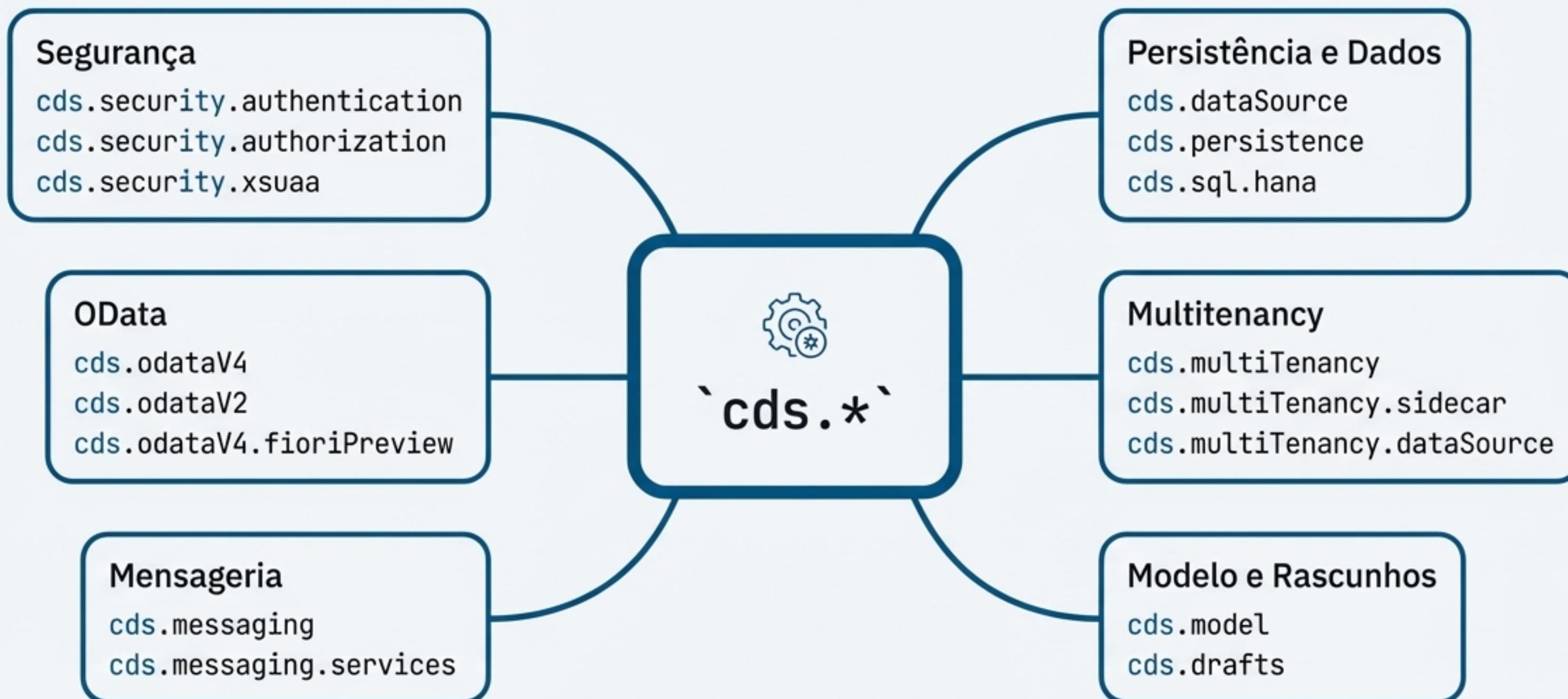
- cds.index-page.enabled: **false**
 - **Por quê?** A página de índice expõe informações sobre os serviços e não deve estar acessível publicamente.
- cds.security.mock.enabled: **false**
 - **Por quê?** Os usuários mockados são uma ferramenta de desenvolvimento e representam uma vulnerabilidade de segurança grave em produção.
- cds.security.authentication.internalUserAccess.enabled: **false**
 - **Por quê?** O acesso para testes internos deve ser estritamente desabilitado.



Não habilite manualmente em produção funcionalidades que são desativadas pelo perfil de produção, pois isso pode introduzir sérias vulnerabilidades de segurança.

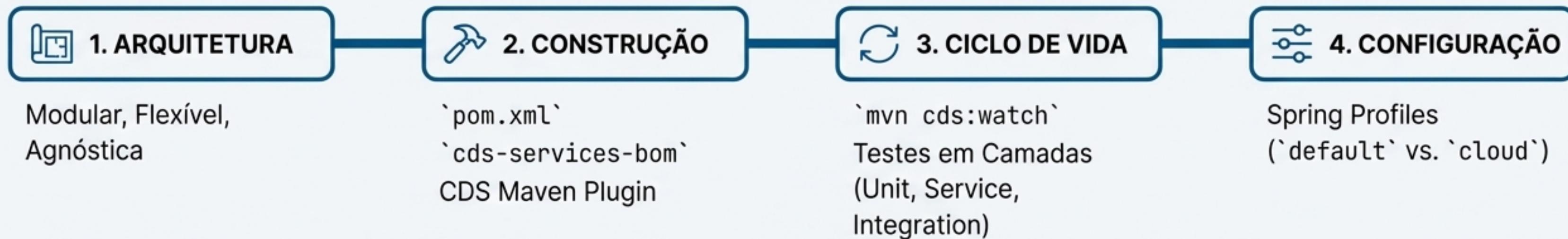
Um Universo de Possibilidades de Configuração

O CAP Java oferece um rico conjunto de propriedades de configuração para ajustar cada aspecto da sua aplicação. Elas são definidas no seu `application.yml`.



Esta é apenas uma amostra. Consulte a documentação oficial para a lista completa de propriedades e suas descrições.

A Jornada do Desenvolvedor: Um Resumo Visual



Caixa de Ferramentas Essencial

➔ Conceitos-Chave

- cds-services-bom: Para sincronia de versões.
- Spring Profiles: Para configuração de ambientes.
- H2 Database: Para desenvolvimento e testes locais.
- Typed Access: Para código seguro e legível.

➔ Comandos Essenciais

- mvn archetype:generate: Para criar um novo projeto.
- mvn cds:watch: Para o "inner loop" de desenvolvimento.
- mvn spring-boot:run: Para executar a aplicação.

Próximos Passos: Da Aplicação ao Ambiente Produtivo

Para acelerar a configuração de aplicações prontas para produção, o CAP Java oferece “Starter Bundles”.



cds-starter-cloudfoundry

Agrupa funcionalidades para o ambiente SAP BTP, Cloud Foundry.

- ✓ Autenticação XSUAA
- ✓ Persistência SAP HANA
- ✓ Multitenancy

O que são? São dependências Maven que agrupam um conjunto de funcionalidades para cenários comuns, como implantação na SAP BTP.



cds-starter-k8s

Agrupa funcionalidades para o ambiente SAP BTP, Kyma/K8s.

- ✓ Autenticação XSUAA
- ✓ Persistência SAP HANA
- ✓ Multitenancy

Use os ‘Starter Bundles’ como ponto de partida para seus projetos. Eles fornecem uma base robusta e configurada com as melhores práticas para ambientes SAP BTP, permitindo que você foque na lógica de negócio.