**Autenticações**

**Single Sign-On:**(SSO) Acesso a vários sistemas diferentes, de modo transparente e unificado, por meio de uma única autenticação. O paradigma de (SSO) pode ser classificado em três tipos: *single sign-on web, single sign-on* federalizado e *single sign-on* desktop.

**Características:**

● Combinação de nome de usuário e senha únicos;

● Único método de administração, centralizado ou descentralizado, no qual as mudanças são propagadas por meio dos diversos sistemas da organização;

● Sólida segurança nas sessões de logon e armazenamento de senhas;

● Integração das regras de autorização de múltiplas aplicações.

**Pontos negativos:**

● A descoberta da senha implica no acesso a todos os serviços;

● Repositório central de dados do usuário constitui um único ponto de invasão e o acesso de todos os usuários ficará comprometido.

**Pontos positivos:**

● Aumento da produtividade;

● Padronização da política de nomes de acesso/senhas;

● Aplicação consistente na política de segurança.

Muito utilizado hoje em dia, também, dentro dos próprios serviços do Google

**JWT** - O JSON Web Token é um padrão da Internet para a criação de tokens de acesso baseados em JSON que afirmam um certo número de declarações. Por exemplo, um servidor pode gerar um token com a declaração "logado como administrador" e fornecê-lo a um cliente.

**SAML (Security Assertion Markup Language)** é constituído por componentes modulares que permitem a transferência de identidade, autenticação e informações de autorização entre organizações. Dentre esses componentes, os protocolos são utilizados para realizar as requisições definidas pelo SAML e retornar as respostas apropriadas.

O protocolo que fornece um mecanismo pelo qual as mensagens do protocolo SAML podem ser passadas por referência, utilizando, para isso, um valor pequeno e de tamanho fixo, é o Artifact Resolution Protocol.

**Autenticação de múltiplos fatores (ou verificação em duas etapas) -** utiliza varias tecnologias para autenticação e é dividida em 3 fatores:

1º - o que você é? Ex: biometria, íris, escaneamento facial...

2º - o que você sabe? Ex: Senha

3º - o que você possui? Ex: cartão, token...

Não existe ordem. Para ser forte necessita de ao menos 2 fatores.

O **múltiplo fator de autenticação (MFA)** é uma técnica para aumentar o nível de segurança de acessos em sistemas, que consiste em 2 (duplo fator) ou mais métodos de autenticação em conjunto para autenticação e autorização em ambientes computacionais.

Como colegas disseram anteriormente, esses métodos consistem basicamente em 3 tipos: o que você sabe (ex.: senha), o que você tem (ex.: cartão de crédito), quem você é (ex.: biometria).

**PAM**: autenticação de usuários nos ambientes Linux/Unix. Através do PAM e de suas bibliotecas, é possível configurar um esquema de autenticação segura para qualquer aplicação de forma transparente.

**SIEM**: coleta e agrega dados de log gerados em toda a infraestrutura de tecnologia da organização, desde sistemas host e aplicativos até dispositivos de rede e segurança, como firewalls e filtros antivírus.

**OAuth** é um padrão aberto para autorização, comumente utilizado para permitir que os usuários da Internet possam fazer logon em sites de terceiros usando suas contas do Google, Facebook, Microsoft, Twitter, etc.—mas, sem expor sua senha.[1] Geralmente, o OAuth fornece aos clientes um "acesso seguro delegado" aos recursos do servidor em nome do proprietário do recurso. Ele especifica um processo para proprietários de recursos para autorizar o acesso de terceiros aos seus recursos de servidor sem compartilhar suas credenciais. Projetado especificamente para trabalhar com o Protocolo de Transferência de Hipertexto (HTTP), o OAuth permite essencialmente tokens de acesso a ser emitidos para clientes de terceiros, mediante autorização do servidor, com a aprovação do proprietário do recurso. O terceiro, em seguida, usa o token de acesso para recursos protegidos hospedados pelo servidor. **OAuth permite que sites de terceiros acessem sua conta sem você ter de dar-lhes a sua senha.** Os aplicativos são capazes de acessar a sua conta, se e somente se você autorizar a fazê-lo, e se você revogar esse direito, em seguida, o aplicativo será imediatamente incapaz de tomar ações em seu nome.

Em segundo lugar, cada site de terceiros que você autoriza pode apenas fazer ações específicas, autorizadas por você. Isso significa que, por exemplo, se você é da administração e autoriza um aplicativo que solicita apenas “Direitos básicos”, se o mesmo tentar apagar uma página (que requer direitos administrativos), a wiki rejeitará a solicitação. Anteriormente, se um aplicativo tivesse a sua senha, você estaria dependente das garantias do autor do aplicativo que ele não usaria seus direitos avançados.

**JWT**: padrão da Internet para a criação de dados com assinatura opcional e/ou criptografia cuja sua payload contém o JSON que afirma algum número de declarações

**SSH**: protocolo de rede criptográfico para operação de serviços de rede de forma segura sobre uma rede insegura.

**DDD – Domain Driven Design**

A principal ideia do DDD é a de que o mais importante em um software não é o seu código, nem sua arquitetura, nem a tecnologia sobre a qual foi desenvolvido, mas sim o problema que o mesmo se propõe a resolver, ou em outras palavras, a regra de negócio.

* Aplicável à área de negócio específica e complexa  
  Não tem como foco principal a tecnologia, mas o entendimento das regras de negócio e de como elas devem estar refletidas no código e no modelo de domínio.

 Arquitetura   
        Camada de Apresentação (User Interface)  
        Camada de Aplicação (Application)  
        Camada de Modelo (Domain)  
        Camada de Infraestrutura (Infrastructure)

Características  
        Alinhamento do código como o negócio  
        Favorecer reutilização  
        Mínimo de acoplamento  
        Independência da Tecnologia

Padrão  
        Contexto  
        Um problema  
        Uma solução  
  
    Não é arquitetura em camadas  
    Úbiqua  
    Anticorrupção  
    Escrito por Eric Evans - Projeto Orientado a Domínio

**DevOps**

**DevOps**: as equipes de desenvolvimento e operações não ficam mais separadas. Às vezes, essas duas equipes são combinadas em uma só. Os engenheiros trabalham durante o ciclo de vida inteiro do aplicativo, da fase de desenvolvimento e testes à fase de implantação e operações, e desenvolvem várias qualificações não limitadas a uma única função. **É um termo criado para descrever um conjunto de práticas para integração entre as equipes de desenvolvimento de softwares, operações (infraestrutura ou sysadmin) e de apoio envolvidas (como controle de qualidade) e a adoção de processos automatizados para produção rápida e segura de aplicações e serviços.**

* Uso de processos e metodologias de desenvolvimento ágil.
* Demanda crescente na taxa de produção de aplicativos e unidades de negócios.
* Ampla disponibilidade de infraestrutura na Cloud e virtualizada por provedores internos e externos.
* Uso crescente de automação de data center e ferramentas de gerenciamento de configuração.
* **Infraestrutura como um código:** as configurações e scripts de execução para instalação de serviços devem ser versionados no mesmo repositório e da mesma forma que o código da aplicação para que possam ser disponibilizados, auditados e evoluídos juntos.

**Cadeira de ferramentas Devops:**Codificação > Compilação > Teste > Pacote> Liberação > Configuração > Monitoramento.

**Benefícios DevOps:** Confiabilidade; **Segurança;**Velocidade; Entrega rápida; Escala; Colaboração melhorada; Economia de recursos;  Processos otimizados e simplificados; Melhora na qualidade da comunicação; Produções mais ágeis; Aumento na motivação da equipe; Ambiente colaborativo entre equipes; Automação de tarefas.

A cultura DevOps sustenta-se nos seguintes pilares:

* **Integração Contínua**: fácil transferência de conhecimento e experiências entre as áreas de Desenvolvimento, Operações e Apoio.
* **Implantação Contínua**: liberação rápida e continua de novas versões de software ou serviços.
* **Feedback contínuo:** feedbacks frequentes das equipes envolvidas em todas as fases do ciclo de vida do software ou serviço.

**Containers seguem o paradigma da infraestrutura imutável, ou seja, instâncias de containers criadas com base em um mesmo arquivo de imagem te dão uma garantia de que elas possuirão o mesmo software instalado.**

Containers são úteis pois eles são construídos para se adaptar às necessidades impostas pelo desenvolvedor. Dessa forma, quando um desenvolvedor cria um container com as configurações desejadas por ele, este já vem com todos os binários, bibliotecas e sistemas de arquivos necessários para o início da produção de aplicativos.