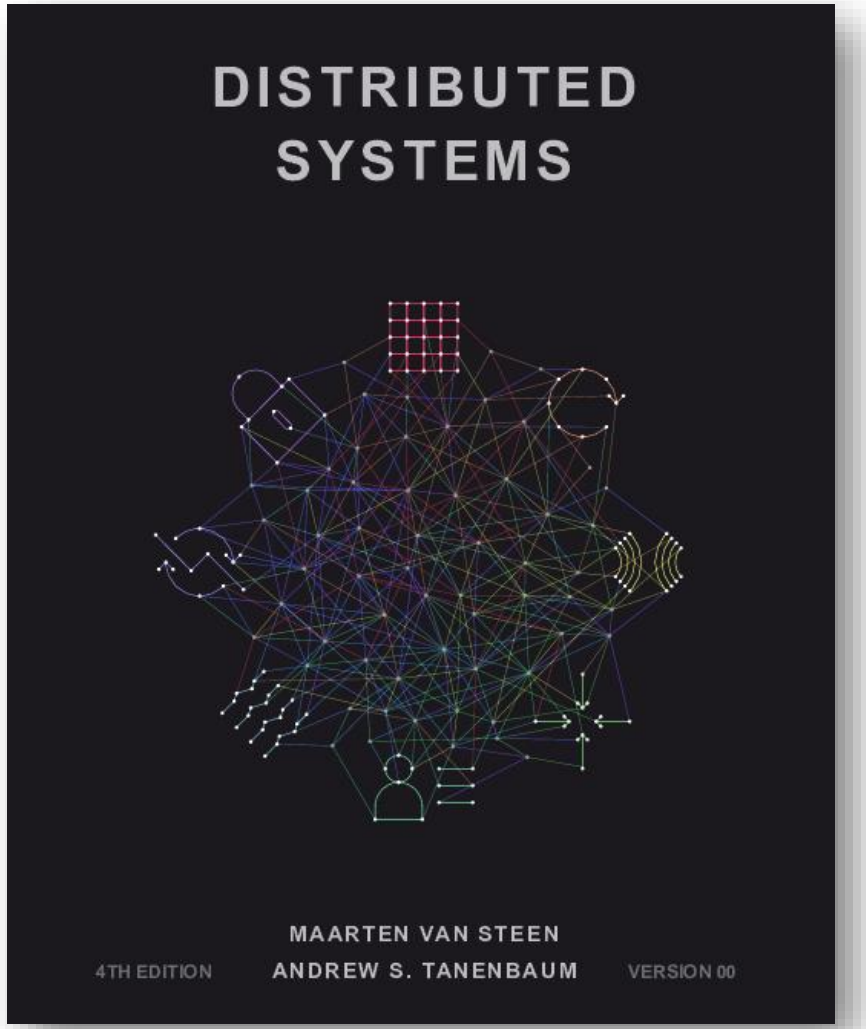
DCC 602 – Sistemas Distribuídos

2 – Arquiteturas

Sumário



Departamento de CIÊNCIA DA COMPUTAÇÃO

UFRR

2 Arquiteturas

- Sistemas distribuídos são frequentemente peças de software complexas, cujos componentes estão dispersos em múltiplas máquinas por definição
- Para dominar a complexidade, é crucial que esses sistemas sejam organizados adequadamente.
- Existem diferentes maneiras de visualizar a organização de um sistema distribuído, mas uma óbvia é fazer uma distinção entre:
  - A organização lógica da coleção de componentes de software
  - A realização física real (hardware e rede)

2 Arquiteturas

- A implementação real de um sistema distribuído requer que instanciemos e coloquemos os componentes de software em máquinas reais.
  - Existem muitas escolhas que podem ser feitas ao fazê-lo. A instância final de uma arquitetura de software também é chamada de **arquitetura do sistema**.
  - Vamos examinar arquiteturas centralizadas tradicionais, nas quais um único servidor implementa a maior parte dos componentes de software (e, portanto, da funcionalidade), enquanto os clientes remotos podem acessar esse servidor usando meios de comunicação simples
  - Vamos examinar considerar arquiteturas descentralizadas ponto-a-ponto, nas quais todos os nós desempenham mais ou menos papéis iguais.
  - Muitos sistemas distribuídos do mundo real são frequentemente organizados de forma **híbrida**, combinando elementos de **arquiteturas centralizadas e descentralizadas**

2.1 Estilos Arquitetônicos

- **Arquitetura de Software:** a organização lógica de um sistema distribuído em componentes de software
- **Estilo Arquitetônico:** é formulado em termos de componentes, da maneira como os componentes estão conectados entre si, dos dados trocados entre os componentes e, por fim, de como esses elementos são configurados em conjunto em um sistema
- **Componente:** é uma unidade modular com interfaces exigidas e fornecidas bem definidas, que é substituível dentro de seu ambiente
  - Componente pode ser substituído enquanto um sistema continua a operar

2.1 Estilos Arquitetônicos

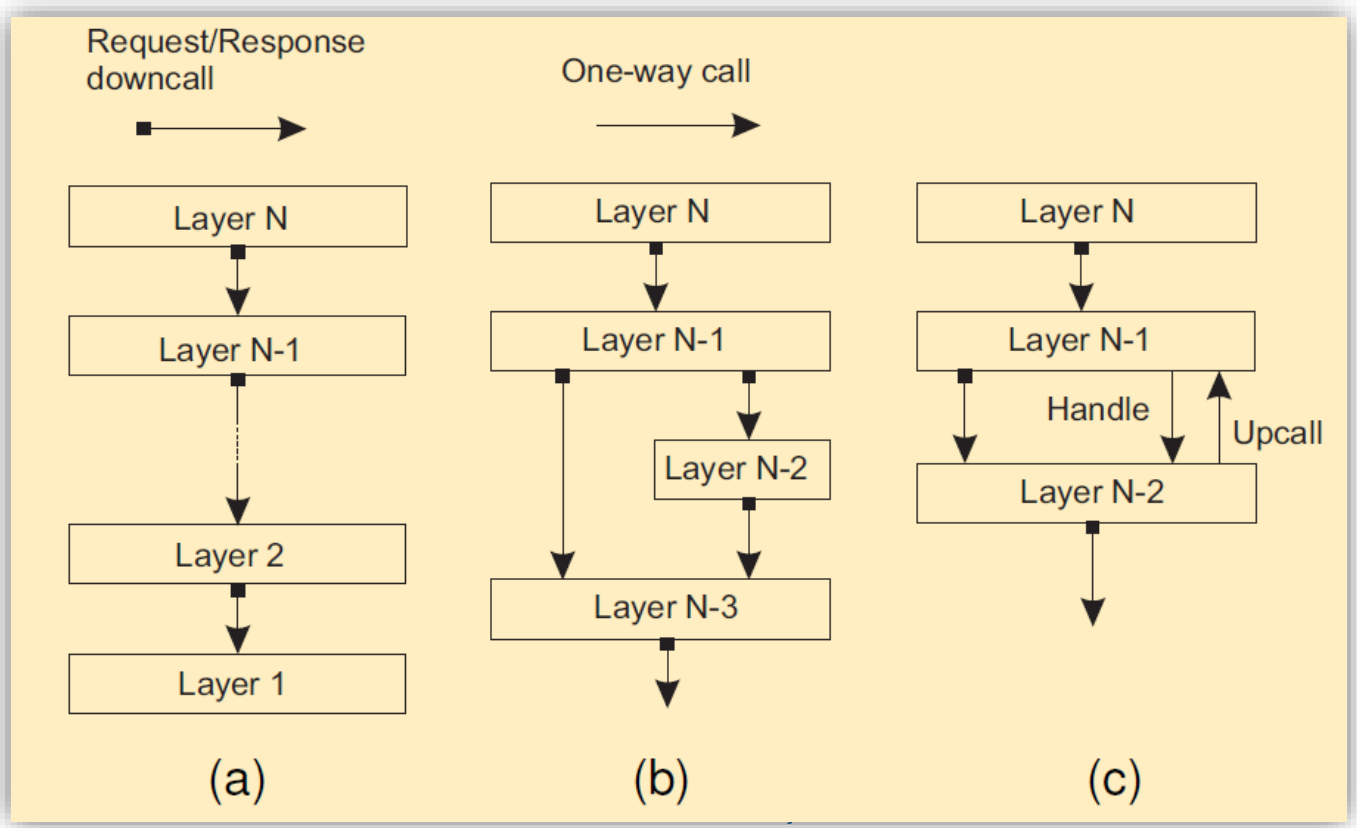
- **Um conceito um pouco mais difícil de compreender é o de um conector:**
  - Geralmente é descrito como um mecanismo que media a comunicação, coordenação ou cooperação entre componentes
    - Um conector pode ser formado pelas facilidades para chamadas de procedimento (remotas)
    - Passagem de mensagens ou transmissão de dados
    - Em outras palavras, um conector permite o fluxo de controle e dados entre componentes

2.1 Estilos Arquitetônicos

- Usando **componentes e conectores**, podemos chegar a várias configurações, que, por sua vez, foram classificadas em Estilos Arquitetônicos
- Vários estilos foram identificados até agora, dos quais os mais importantes para sistemas distribuídos são:
  - Arquiteturas em camadas
  - Arquiteturas orientadas a serviços
  - Arquiteturas publish-subscribe

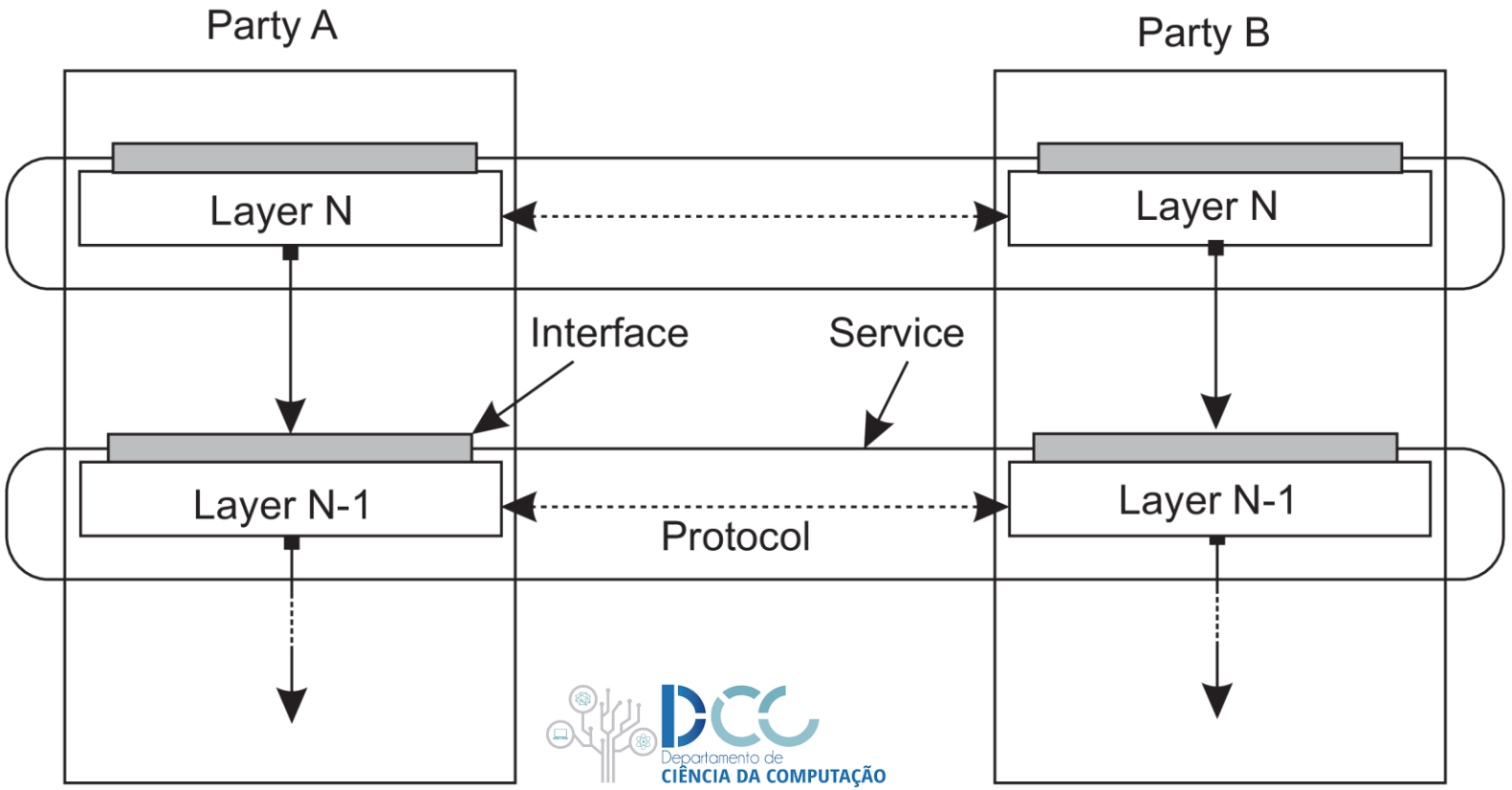
2.1 Estilos Arquitetônicos

- **Arquitetura em Camadas**



2.1 Estilos Arquitetônicos

- **Arquitetura em Camadas: Protocolos de comunicação em camadas**
  - Uma pilha de protocolos de comunicação em camadas, mostrando a diferença entre um serviço, sua interface e o protocolo que ele implementa



Departamento de CIÊNCIA DA COMPUTAÇÃO

UFRR

2.1 Estilos Arquitetônicos

- **Arquitetura em Camadas: Protocolos de comunicação em camadas**
  - Exemplo de implementação cliente/servidor em python usando sockets
    - https://colab.research.google.com/drive/17ceV2KQUA8EvOl6uj3a63nNuhSuE_NPZ?usp=sharing

2.1 Estilos Arquitetônicos

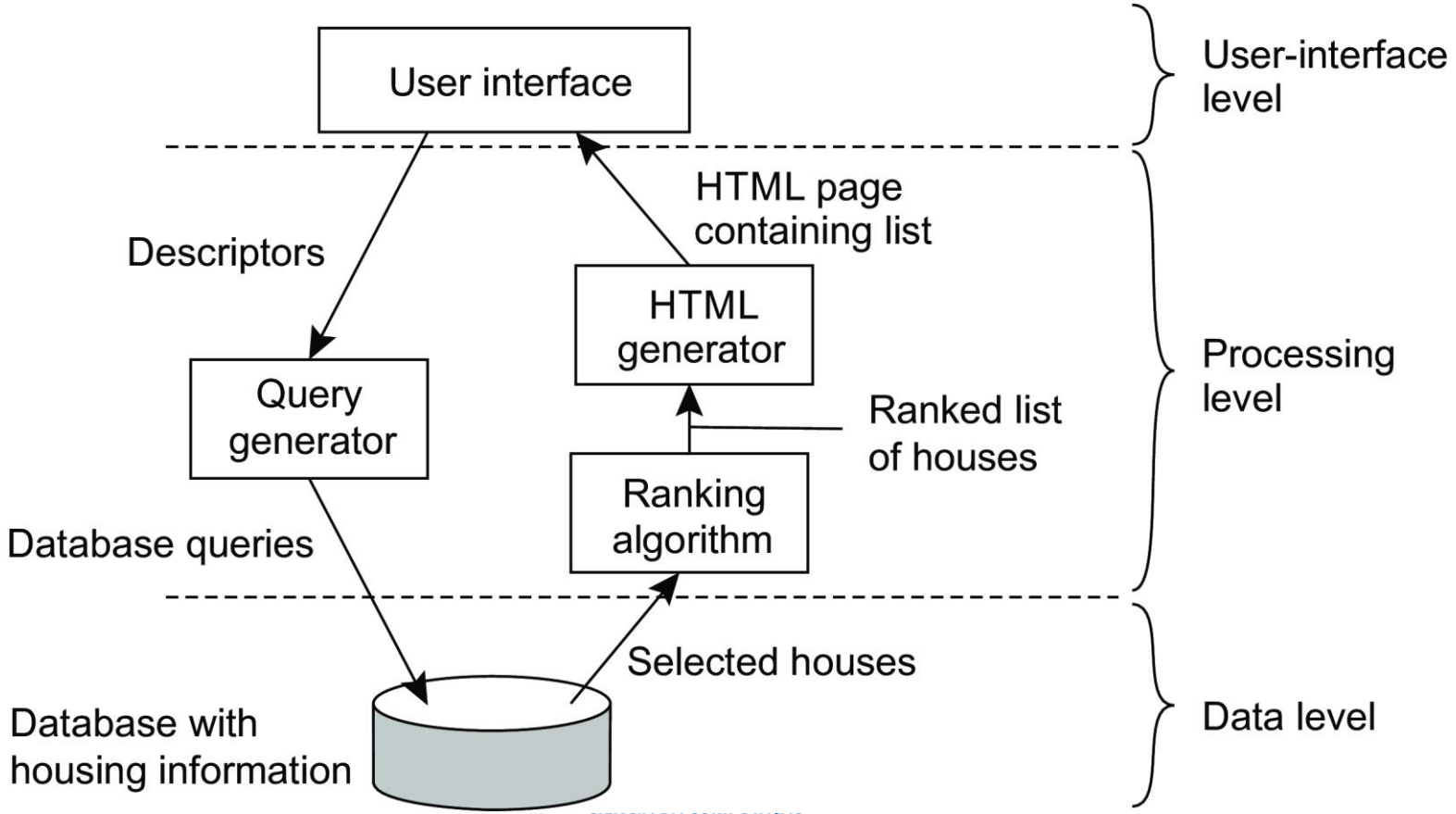
- **Arquitetura em Camadas: Protocolos de comunicação em camadas**
  - Atividades:
    1. Implementar um cliente e servidor TCP em Python usando o exemplo fornecido como base, mas com algumas modificações. Por exemplo, pode-se mudar a mensagem enviada pelo cliente ou a resposta do servidor. Os alunos também podem experimentar diferentes portas e endereços IP.
    2. Utilizar o tcpdump para capturar e analisar pacotes de outras comunicações na rede, por exemplo, acessar um site usando um navegador e capturar e analisar os pacotes gerados.
    3. Adicionar recursos de segurança, como criptografia, aos programas cliente e servidor para garantir a confidencialidade das informações trocadas.
    4. Implementar um programa de chat simples usando sockets TCP. Os alunos podem criar um servidor que aceita várias conexões de clientes e encaminha as mensagens de um cliente para todos os outros clientes conectados.

2.1 Estilos Arquitetônicos

- **Arquitetura em Camadas: Camadas de Aplicações**
  - Considerando que uma grande classe de aplicações distribuídas é direcionada para suportar usuários ou aplicações a acessarem bancos de dados, muitas pessoas têm defendido uma distinção entre três níveis lógicos, seguindo essencialmente um estilo arquitetural em camadas:
    - O nível de interface de aplicação
    - O nível de processamento
    - O nível de dados

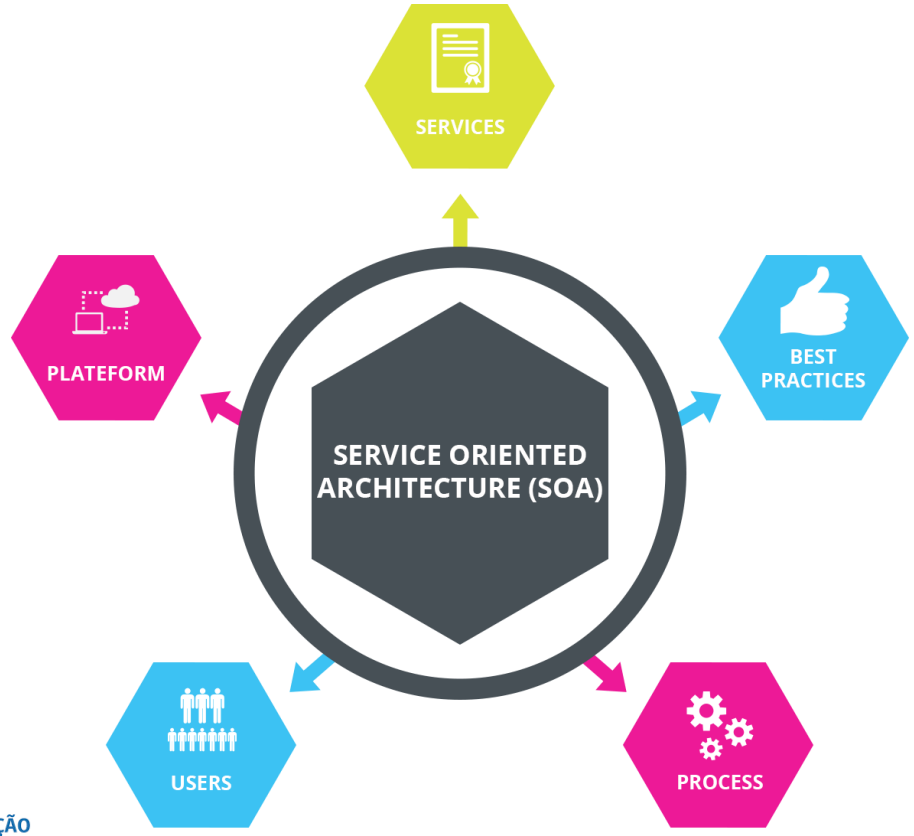
2.1 Estilos Arquitetônicos

- **Arquitetura em Camadas: Camadas de Aplicações**
  - Considere um motor de busca simples na internet, por exemplo, um dedicado à compra de casas



2.1 Estilos Arquitetônicos

- **Arquiteturas orientadas a serviços (SOA)**
  - É um estilo de arquitetura tem como foco o desenvolvimento de sistemas distribuídos, em que as funcionalidades são disponibilizadas por principalmente por meio de serviços web



2.1 Estilos Arquitetônicos

- **Arquiteturas orientadas a serviços (SOA)**
  - Os principais princípios de SOA incluem:
    - Separar as funcionalidades em serviços independentes e autônomos: Cada serviço é responsável por uma única tarefa específica e pode ser chamado por outras aplicações ou serviços.
    - Usar interfaces padronizadas: Os serviços são expostos por meio de interfaces padronizadas, como o protocolo HTTP e o formato XML ou JSON, para garantir que as aplicações possam interagir com diferentes serviços, independentemente da tecnologia usada.
    - Maximizar a reutilização: Os serviços são projetados para serem reutilizáveis, permitindo que várias aplicações possam utilizar as mesmas funcionalidades.
    - Focar na interoperabilidade: Os serviços devem ser projetados para trabalhar em conjunto com outras tecnologias, permitindo que as aplicações possam interagir com diferentes serviços.

2.1 Estilos Arquitetônicos

- **Arquiteturas orientadas a serviços (SOA)**
  - Algumas das principais arquiteturas orientadas a serviços em sistemas distribuídos incluem:
    - SOAP (Simple Object Access Protocol): é um protocolo baseado em XML que é usado para troca de informações entre sistemas distribuídos por meio de serviços web.
    - Exemplos:
      - Amazon Web Services (AWS) API: a AWS oferece diversas APIs SOAP para seus serviços em nuvem, como Amazon S3, Amazon EC2, Amazon SQS, entre outros.
      - Google Maps API: a Google oferece uma API SOAP para o serviço Google Maps, que permite que os desenvolvedores integrem mapas e dados geográficos em suas aplicações.
      - eBay API: a eBay oferece uma API SOAP para o desenvolvimento de aplicações que se integram à plataforma de vendas online da empresa.

2.1 Estilos Arquitetônicos

- **Arquiteturas orientadas a serviços (SOA)**
  - Algumas das principais arquiteturas orientadas a serviços em sistemas distribuídos incluem:
    - REST (Representational State Transfer): é um estilo de arquitetura que utiliza os métodos HTTP para acessar recursos na web, permitindo que as aplicações possam interagir com serviços web de uma maneira simples e flexível
    - Exemplos:
      - Twitter API: permite acessar dados do Twitter, como tweets e usuários.
      - Facebook Graph API: permite acessar dados do Facebook, como postagens e informações de perfil.
      - Google Maps API: permite integrar mapas e informações de localização em aplicativos.
      - GitHub API: permite acessar informações de repositórios e usuários do GitHub

2.1 Estilos Arquitetônicos

- **Arquiteturas orientadas a serviços (SOA)**
  - Algumas das principais arquiteturas orientadas a serviços em sistemas distribuídos incluem:
    - Microservices: é uma arquitetura de software que divide as aplicações em serviços menores e independentes que são implantados e escalados separadamente. Essa arquitetura permite uma maior flexibilidade e escalabilidade para as aplicações
    - Exemplos:
      - gRPC: é uma estrutura de alto desempenho desenvolvida pelo Google, que utiliza o protocolo RPC (Remote Procedure Call) para a comunicação entre serviços. Ele suporta diversas linguagens de programação e oferece recursos como streaming de dados e autenticação.
      - AWS Lambda: é um serviço de computação em nuvem da Amazon que permite a criação e execução de funções sem servidor. Ele é frequentemente usado para a criação de microservices em ambientes de nuvem.
      - Docker: é uma plataforma para criação e execução de contêineres que permite o isolamento de aplicativos e serviços. Ele é frequentemente usado para empacotar e distribuir microservices de forma eficiente.
      - Kubernetes: é um sistema de orquestração de contêineres que automatiza o gerenciamento de aplicativos e serviços distribuídos. Ele é frequentemente usado para implantar e escalar microservices em ambientes de nuvem

Continua!