# **1)**

# **A)** On-Premise (Local):

Hospedar aplicativos localmente, ou On-Premise, significa que a empresa mantém seus servidores e infraestrutura de TI em suas próprias instalações físicas, em vez de utilizar serviços de nuvem ou data centers externos.

Vantagens:

- Controle total sobre a infraestrutura.

- Possibilidade de personalização extensiva.

- Maior controle sobre a segurança dos dados.

Desvantagens:

- Custos iniciais elevados.

- Necessidade de manutenção e atualizações constantes.

- Menos flexibilidade e escalabilidade comparado a soluções em nuvem.

**B)** PaaS (Plataforma como Serviço):

PaaS é um modelo de computação em nuvem que fornece uma plataforma para desenvolvedores criarem, executar e gerenciar aplicativos sem se preocupar com a complexidade da infraestrutura.

Exemplos de benefícios:

- Desenvolvimento e implementação mais rápidos.

- Escalabilidade automática.

- Menos preocupação com questões de infraestrutura.

Exemplos populares de PaaS:

- Google App Engine: Fornece ambiente para desenvolver e hospedar aplicativos.

- Microsoft Azure App Services: Suporta várias linguagens e frameworks de desenvolvimento.

- Heroku: Plataforma fácil de usar para construir, executar e escalar aplicativos.

**C)** IaaS (Infraestrutura como Serviço):

IaaS é um modelo de computação em nuvem que fornece infraestrutura virtualizada pela internet, como máquinas virtuais e armazenamento.

Diferenças com PaaS:

- Em IaaS, a empresa tem mais controle sobre a infraestrutura, enquanto em PaaS, o foco é no desenvolvimento de aplicativos.

- Em IaaS, a empresa é responsável pelo gerenciamento da infraestrutura, enquanto em PaaS, o provedor gerencia a infraestrutura.

Exemplos de recursos em IaaS:

- Máquinas Virtuais: Recursos computacionais virtuais.

- Armazenamento em Nuvem: Espaço de armazenamento para dados e aplicativos.

- Redes Virtuais: Isolamento e segurança na nuvem.

**D)** SaaS (Software como Serviço):

SaaS é um modelo de entrega de software onde os aplicativos são hospedados por um provedor de serviços e disponibilizados aos clientes pela internet.

Exemplos comuns:

- Google Workspace (anteriormente G Suite): Conjunto de aplicativos de produtividade baseados na web.

- Salesforce: Plataforma de gerenciamento de relacionamento com o cliente (CRM) baseada em nuvem.

- Microsoft 365: Suite de aplicativos de produtividade baseados em nuvem.

Vantagens do SaaS:

- Sem necessidade de instalação ou manutenção.

- Acessível de qualquer dispositivo com conexão à internet.

- Atualizações automáticas.

**E)** Comparação e Decisão:

A recomendação dependerá das necessidades específicas da empresa:

- Escalabilidade: Para crescimento rápido e demandas flutuantes, PaaS e IaaS são ideais devido à sua escalabilidade automática.

- Custo: SaaS geralmente têm custos menores devido à ausência de investimentos em hardware e manutenção.

- Manutenção: SaaS requer menos manutenção, enquanto On-Premise exige mais atenção às atualizações e segurança.

- Segurança: IaaS e On-Premise oferecem maior controle sobre a segurança, enquanto SaaS e PaaS contam com as medidas de segurança do provedor.

Para empresas com orçamento limitado e necessidade de rápida implementação, SaaS é uma escolha sólida. Para organizações que precisam de controle total e têm recursos para manter a infraestrutura, o On-Premise é apropriado. Para flexibilidade e desenvolvimento eficiente, PaaS é vantajoso, enquanto IaaS é ideal para empresas que precisam de controle sobre a infraestrutura, mas não querem lidar com a complexidade física.

## **2)**

## **1)** API RESTful:

Uma API RESTful (Representational State Transfer) é uma arquitetura de software que define um conjunto de restrições para criar serviços web. Ela segue os seguintes princípios-chave:

- Stateless (Sem estado): Cada requisição de cliente para o servidor deve conter toda a informação necessária para compreender e processar a requisição. O servidor não deve manter o estado da sessão do cliente entre requisições.

- Recursos (Resources): As informações são tratadas como recursos e são acessadas via URI (Uniform Resource Identifier). Cada recurso tem um URI único associado a ele.

- Representações: Os recursos podem ter diferentes representações, como JSON ou XML. Um cliente pode requisitar uma representação específica (por exemplo, JSON) ao acessar o recurso.

- Métodos HTTP: Utiliza os métodos padrão HTTP (GET, POST, PUT, DELETE) para realizar operações em recursos.

REST é amplamente usada no desenvolvimento de sistemas distribuídos devido à sua simplicidade, escalabilidade e flexibilidade. A separação clara entre cliente e servidor permite que eles evoluam independentemente.

2) Verbos HTTP REST Comuns:

- GET: Usado para recuperar dados de um recurso específico ou uma lista de recursos.

- POST: Utilizado para criar um novo recurso.

- PUT: Usado para atualizar um recurso existente.

- DELETE: Utilizado para excluir um recurso específico.

Esses verbos se mapeiam às operações CRUD da seguinte forma:

- GET: Equivale à operação de leitura (Read).

- POST: Equivale à operação de criação (Create).

- PUT: Equivale à operação de atualização (Update).

- DELETE: Equivale à operação de exclusão (Delete).

**3)** Verbos para o CRUD:

- Create (Criação):

- Verbo HTTP: POST

- Exemplo de URL: `/tasks`

- Payload JSON:

json

{

"title": "Nova Tarefa",

"status": "Pendente"

}

- Read (Leitura):

- Verbo HTTP: GET

- Listar todas as tarefas:

- Exemplo de URL: `/tasks`

- Recuperar detalhes de uma tarefa específica:

- Exemplo de URL: `/tasks/{id}` (onde `{id}` é o identificador único da tarefa)

- Update (Atualização):

- Verbo HTTP: PUT

- Exemplo de URL: `/tasks/{id}` (onde `{id}` é o identificador único da tarefa a ser atualizada)

- Payload JSON (para atualizar o título da tarefa):

json

{

"title": "Novo Título da Tarefa"

}

- Payload JSON (para atualizar o status da tarefa):

json

{

"status": "Concluída"

}

- Delete (Exclusão):

- Verbo HTTP: DELETE

- Exemplo de URL: `/tasks/{id}` (onde `{id}` é o identificador único da tarefa a ser excluída)