Aqui está uma avaliação de diferentes arquiteturas (Camadas, Monolítica, Microsserviços, Serverless, Orientada a Eventos e SOA) com base em requisitos não funcionais: *Desempenho, **Escalabilidade, **Usabilidade* e *Segurança*.

1. *Arquitetura em Camadas (Layered Architecture)*

- *Desempenho: **Parcial*
  - A arquitetura em camadas pode adicionar overhead ao sistema, já que cada solicitação passa por múltiplas camadas, o que pode impactar a latência.

- *Escalabilidade: **Parcial*
  - As camadas podem ser escaladas, mas isso muitas vezes requer escalabilidade vertical (aumento de capacidade do servidor) em vez de horizontal (distribuição de carga), o que é menos eficiente.

- *Usabilidade: **Adequada*
  - A separação clara de responsabilidades nas camadas melhora a manutenção e a modularidade do sistema, o que ajuda na usabilidade e organização do código.

- *Segurança: **Adequada*
  - A arquitetura em camadas facilita a implementação de mecanismos de segurança em pontos específicos, como controle de acesso na camada de aplicação ou validação de dados na camada de apresentação.

---

2. *Arquitetura Monolítica*

- *Desempenho: **Adequada*
  - Em uma arquitetura monolítica, a comunicação entre componentes é interna, o que minimiza a latência. Porém, pode perder eficiência à medida que o sistema cresce.

- *Escalabilidade: **Não Atende*
  - Monólitos são mais difíceis de escalar horizontalmente. Geralmente requerem escalabilidade vertical, o que é menos flexível e mais caro.

- *Usabilidade: **Parcial*
  - No início, um monólito pode ser simples de implementar, mas conforme cresce, torna-se mais difícil de manter, resultando em código rígido e menos usável.

- *Segurança: **Adequada*
  - Por ser um sistema único, as políticas de segurança podem ser implementadas centralmente. No entanto, se uma parte for comprometida, todo o sistema pode estar em risco.

---

3. *Arquitetura de Microsserviços (Microservices Architecture)*

- *Desempenho: **Parcial*
  - Microsserviços podem melhorar o desempenho ao permitir a otimização de serviços específicos, mas a comunicação entre serviços pode aumentar a latência e complexidade.

- *Escalabilidade: **Adequada*
  - Cada serviço pode ser escalado independentemente com base em sua carga, oferecendo grande flexibilidade para gerenciar o crescimento do sistema.

- *Usabilidade: **Adequada*
  - Microsserviços permitem que equipes desenvolvam, testem e implementem de forma independente, facilitando a manutenção e a evolução contínua.

- *Segurança: **Adequada*
  - Microsserviços podem ser mais seguros devido ao isolamento de cada serviço, mas exigem uma gestão mais complexa de autenticação e autorização entre serviços.

---

4. *Arquitetura Serverless*

- *Desempenho: **Parcial*
  - Serverless é eficiente para cargas variáveis, mas pode sofrer com "cold starts", que adicionam latência nas primeiras invocações.

- *Escalabilidade: **Adequada*
  - Escalabilidade automática é um dos principais pontos fortes do serverless, ajustando a capacidade de acordo com a demanda sem intervenção manual.

- *Usabilidade: **Adequada*
  - Como a infraestrutura é abstraída, desenvolvedores podem se concentrar na lógica de negócios, o que melhora a usabilidade em termos de desenvolvimento.

- *Segurança: **Adequada*
  - A segurança é reforçada pelo provedor de nuvem, mas ainda é necessário configurar permissões corretamente para cada função, além de garantir a segurança na comunicação entre funções.

---

5. *Arquitetura Orientada a Eventos (Event-Driven Architecture)*

- *Desempenho: **Adequada*
  - A arquitetura orientada a eventos pode fornecer alta responsividade e desempenho para sistemas com grandes volumes de dados e alta demanda de I/O, pois processa eventos de forma assíncrona.

- *Escalabilidade: **Adequada*

  - Esta arquitetura é altamente escalável, pois serviços e processos podem ser escalados com base no volume de eventos gerados.

- *Usabilidade: **Parcial*
  - Pode ser complexa para desenvolver e manter, pois envolve um modelo de processamento assíncrono e distribuído, o que pode complicar a depuração e o fluxo lógico do sistema.

- *Segurança: **Parcial*
  - A segurança pode ser complexa, já que envolve o rastreamento e controle de eventos que podem disparar ações em diferentes partes do sistema, exigindo rigoroso controle de acesso.

---

6. *Arquitetura SOA (Service-Oriented Architecture)*

- *Desempenho: **Parcial*
  - SOA pode ser eficiente se bem implementada, mas a comunicação entre serviços através de redes (geralmente via SOAP ou REST) pode introduzir latência.

- *Escalabilidade: **Adequada*
  - SOA permite escalabilidade de serviços individuais, mas pode não ser tão flexível ou granular como uma arquitetura de microsserviços.

- *Usabilidade: **Adequada*
  - SOA promove a reutilização de serviços e a modularidade, o que facilita o desenvolvimento e manutenção do sistema.

- *Segurança: **Adequada*
  - A arquitetura SOA oferece boa segurança com a implementação de políticas de autenticação, autorização e criptografia entre serviços, mas a complexidade pode ser maior devido à comunicação distribuída.

Resumo:

Arquitetura	Desempenho	Escalabilidade	Usabilidade	Segurança
Camadas	Parcial	Parcial	Adequada	Adequada
Monolítica	Adequada	Não Atende	Parcial	Adequada
Microsserviços	Parcial	Adequada	Adequada	Adequada
Serverless	Parcial	Adequada	Adequada	Adequada
Event-Driven	Adequada	Adequada	Parcial	Parcial
SOA	Parcial	Adequada	Adequada	Adequada

Cada arquitetura tem seus pontos fortes e fracos dependendo do contexto e dos requisitos específicos do sistema. Para garantir o melhor desempenho, escalabilidade, usabilidade e segurança, é importante analisar o caso de uso e adaptar a arquitetura conforme necessário.