Caracterização de SD

1. Cite cinco tipos de recursos de hardware e cinco tipos de recursos de software ou dados que podem ser compartilhados. Exemplifique como esse compartilhamento ocorre em SDs.

Recursos de Hardware:

1. Processadores
2. Memória RAM
3. Armazenamento de dados (disco rígido, SSD)
4. Dispositivos de entrada/saída (teclado, mouse, monitor)
5. Dispositivos de rede (roteadores, switches, cabos)

Recursos de Software ou Dados:

1. Sistemas Operacionais
2. Bancos de dados
3. Aplicativos de software (softwares de edição de texto, planilhas eletrônicas)
4. Bibliotecas de código (como uma biblioteca de funções matemáticas)
5. Dados (como arquivos de texto ou imagens)

O compartilhamento desses recursos ocorre em SDs através de vários mecanismos, incluindo:

1. Virtualização: a criação de ambientes virtuais que compartilham os recursos subjacentes do hardware, como processadores, memória e armazenamento. Um exemplo é a virtualização de servidores, onde várias máquinas virtuais são executadas em um único servidor físico.

2. Clusterização: a combinação de vários servidores para formar um único sistema distribuído. Neste caso, cada nó no cluster compartilha recursos como armazenamento e processamento.

3. Compartilhamento de arquivos: sistemas de arquivos distribuídos permitem que os usuários acessem arquivos armazenados em diferentes dispositivos de armazenamento conectados em rede.

4. Sistemas de banco de dados distribuídos: permitem que múltiplos usuários acessem e atualizem dados armazenados em diferentes locais.

5. Computação em grade: permite que vários computadores compartilhem recursos de processamento e memória para realizar tarefas complexas que exigem muitos recursos.

Esses são apenas alguns exemplos de como o compartilhamento de recursos ocorre em sistemas distribuídos.

2. Como sincronizar o relógio de dois computadores ligados por uma rede local sem referência a uma fonte externa de tempo? Quais fatores limitam a precisão do procedimento descrito por você? Como poderia um grande número de computadores conectados à Internet serem sincronizados? Discuta a precisão do procedimento.

Uma forma de sincronizar os relógios de dois computadores ligados por uma rede local sem referência a uma fonte externa de tempo é utilizando o protocolo NTP (Network Time Protocol). O NTP utiliza a sincronização de relógio distribuída para sincronizar os relógios de vários computadores em uma rede. Isso é feito através do envio de pacotes NTP que contêm informações de tempo entre os computadores. Cada computador ajusta seu relógio com base nas informações de tempo recebidas de outros computadores na rede.

Os fatores que limitam a precisão do procedimento descrito incluem:

1. Atrasos de rede: a sincronização do relógio depende da transferência de pacotes NTP entre os computadores na rede. A latência de rede pode afetar a precisão da sincronização do relógio, especialmente em redes com alto tráfego ou com conexões instáveis.

2. Diferenças de precisão do relógio: cada computador tem seu próprio relógio interno, que pode ter precisão diferente. Isso pode resultar em diferenças de tempo entre os computadores, mesmo após a sincronização.

3. Erros de software ou hardware: problemas de software ou hardware em um ou mais computadores podem afetar a precisão da sincronização do relógio.

Para sincronizar um grande número de computadores conectados à Internet, geralmente é utilizado um servidor NTP público, que fornece uma fonte precisa e confiável de tempo. Os computadores na rede se sincronizam com o servidor NTP, que utiliza várias fontes de tempo, como GPS ou relógios atômicos, para fornecer tempo preciso.

A precisão do procedimento depende da precisão do servidor NTP utilizado e da latência de rede entre os computadores na rede e o servidor NTP. Em geral, a precisão da sincronização do relógio é alta o suficiente para a maioria das aplicações. No entanto, em aplicações que exigem altíssima precisão de tempo, como sistemas de negociação financeira, pode ser necessário utilizar fontes de tempo ainda mais precisas, como relógios atômicos locais.

3. Considere as estratégias de implementação de MMOG (massively multiplayer online games) discutidas na Seção 1.2.2. Em particular, quais vantagens você vê em adotar a estratégia de servidor único para representar o estado do jogo para vários jogadores? Quais problemas você consegue identificar e como eles poderiam ser resolvidos?

A estratégia de servidor único para representar o estado do jogo para vários jogadores tem algumas vantagens em comparação com outras estratégias de implementação de MMOG:

1. Consistência: o servidor único é responsável por manter o estado do jogo, o que significa que todos os jogadores veem a mesma versão do jogo. Isso garante que as ações dos jogadores sejam consistentes entre si.

2. Segurança: o servidor único pode implementar medidas de segurança, como autenticação e criptografia de dados, para garantir que apenas jogadores autorizados tenham acesso ao jogo.

3. Escalabilidade: a implementação de servidor único pode ser escalonada para lidar com um grande número de jogadores, tornando-a adequada para MMOGs.

No entanto, a estratégia de servidor único também tem alguns problemas que precisam ser resolvidos:

1. Latência: como o servidor único é responsável por atualizar o estado do jogo para todos os jogadores, a latência pode ser um problema. Se o servidor estiver sobrecarregado, pode haver atrasos na atualização do estado do jogo.

2. Desempenho: um servidor único pode ter dificuldade em lidar com grandes quantidades de dados, especialmente se houver muitos jogadores jogando simultaneamente. Isso pode resultar em atrasos no processamento do jogo e na atualização do estado do jogo.

3. Falha do servidor: se o servidor único falhar, o jogo inteiro pode ser interrompido, o que pode ser frustrante para os jogadores. Para mitigar esse problema, é necessário implementar medidas de backup e redundância para garantir que o jogo continue funcionando mesmo se o servidor principal falhar.

4. Custo: a implementação de um servidor único pode ser cara, especialmente se houver muitos jogadores jogando simultaneamente. É necessário investir em hardware e software adequados para garantir o bom funcionamento do servidor.

Para resolver esses problemas, algumas soluções podem ser adotadas, como o uso de servidores distribuídos para lidar com a carga de jogadores, a implementação de técnicas de compressão de dados para reduzir a quantidade de dados transmitidos entre o servidor e os jogadores, e a implementação de mecanismos de tolerância a falhas para garantir que o jogo continue funcionando mesmo se o servidor principal falhar. Além disso, a utilização de servidores de jogos de nuvem pode ser uma opção viável para reduzir o custo de implementação.

4. Considere um viajante em uma estação de trem que nunca visitou antes, portando seu PDA com dispositivos para conexão sem fio. Sugira como o usuário poderia descobrir os serviços oferecidos pela estação e enumere os desafios técnicos para alcançar tal objetivo.

Para descobrir os serviços oferecidos pela estação de trem, o usuário poderia realizar as seguintes ações:

1. Verificar a presença de redes Wi-Fi disponíveis em sua área: a maioria das estações de trem oferece acesso Wi-Fi gratuito. O usuário poderia verificar se há redes Wi-Fi disponíveis em sua área usando seu PDA.

2. Conectar-se à rede Wi-Fi: uma vez que o usuário encontre uma rede Wi-Fi disponível, ele poderia se conectar a ela usando seu PDA.

3. Pesquisar por informações online: uma vez que o usuário esteja conectado à rede Wi-Fi, ele poderia usar seu PDA para pesquisar informações online sobre a estação, como horários de chegada e partida dos trens, serviços oferecidos pela estação, opções de transporte público, etc.

4. Usar aplicativos móveis: o usuário também poderia usar aplicativos móveis projetados especificamente para fornecer informações sobre a estação de trem, como o aplicativo oficial da estação ou aplicativos de terceiros.

No entanto, há alguns desafios técnicos que o usuário pode enfrentar ao tentar descobrir os serviços oferecidos pela estação de trem:

1. Disponibilidade de conexão sem fio: se a estação de trem não oferecer uma rede Wi-Fi ou se a conexão estiver fraca, o usuário pode não ser capaz de se conectar à rede e acessar informações online.

2. Confiabilidade das informações online: as informações online podem não ser precisas ou atualizadas, o que pode levar a problemas como a perda do trem ou a escolha de um serviço incorreto.

3. Usabilidade do aplicativo: se o aplicativo for difícil de usar ou não estiver disponível em um formato compatível com o PDA do usuário, pode ser difícil para o usuário obter as informações necessárias.

4. Segurança da conexão sem fio: se a conexão sem fio não estiver protegida, o usuário pode estar em risco de ter seus dados pessoais comprometidos.

5. Compatibilidade do dispositivo: o dispositivo do usuário pode não ser compatível com as redes sem fio ou aplicativos disponíveis na estação de trem, o que pode limitar sua capacidade de acessar informações online.

5. Compare e contraste a computação em nuvem com a computação cliente servidor mais tradicional. O que há de novo em relação à computação em nuvem como conceito?

A computação em nuvem é um modelo de computação que permite o acesso sob demanda a recursos de computação, como servidores, armazenamento e aplicativos, por meio da

internet. Os usuários podem acessar esses recursos a partir de qualquer lugar e a qualquer momento, pagando apenas pelos recursos que usam.

Por outro lado, a computação cliente-servidor tradicional é um modelo em que os recursos de computação são distribuídos entre um ou mais servidores centralizados e clientes que solicitam serviços desses servidores. Nesse modelo, a maioria dos recursos de computação são gerenciados pelo servidor, enquanto o cliente geralmente fornece apenas uma interface de usuário.

A principal diferença entre esses dois modelos é a localização dos recursos de computação e o grau de controle que o usuário tem sobre esses recursos. Na computação cliente-servidor, os recursos são geralmente mantidos localmente ou em servidores centralizados, enquanto na computação em nuvem, os recursos são fornecidos por meio da internet, a partir de um pool de recursos compartilhados.

O que há de novo na computação em nuvem é que ela permite que as empresas reduzam seus custos com infraestrutura e aumentem sua eficiência e flexibilidade, pagando apenas pelos recursos que realmente usam. A computação em nuvem também oferece escalabilidade e redundância, permitindo que as empresas aumentem ou diminuam seus recursos de acordo com as necessidades do negócio, sem precisar investir em infraestrutura adicional.

Além disso, a computação em nuvem também oferece serviços adicionais, como software como serviço (SaaS), plataforma como serviço (PaaS) e infraestrutura como serviço (IaaS), que podem ser personalizados para atender às necessidades de diferentes tipos de empresas.

Em resumo, a computação em nuvem é um modelo mais flexível e escalável do que a computação cliente-servidor tradicional, permitindo que as empresas reduzam seus custos com infraestrutura, aumentem sua eficiência e melhorem sua capacidade de se adaptar às mudanças no ambiente de negócios.

6. Utilize a WWW como um exemplo para ilustrar os conceitos de compartilhamento de recurso, cliente e servidor. Quais são as vantagens e desvantagens de HTML, URL e HTTP como as tecnologias base para navegação em informação? Alguma dessas tecnologias é conveniente como para a computação cliente/servidor em geral?

A World Wide Web (WWW) é um exemplo de sistema distribuído que utiliza o modelo cliente-servidor para compartilhar recursos, como páginas da web, entre usuários em todo o mundo. O usuário acessa as informações (cliente) por meio de um navegador da web, que faz a solicitação para o servidor, que então envia a resposta (página da web) de volta ao usuário.

As vantagens do HTML, URL e HTTP são que eles são tecnologias amplamente adotadas e padronizadas para navegação na web, o que significa que é fácil para os desenvolvedores criar e compartilhar páginas da web, e para os usuários acessá-las por meio de diferentes dispositivos e navegadores. O HTML é a linguagem de marcação padrão para a criação de páginas da web, o HTTP é o protocolo de comunicação padrão para a transferência de

dados na web e o URL é o endereço que permite aos usuários acessar páginas da web específicas.

No entanto, existem algumas desvantagens dessas tecnologias. Uma desvantagem é que o HTML não é uma linguagem de programação completa, o que significa que é limitado em sua capacidade de criar aplicativos interativos ou sofisticados. Outra desvantagem é que o HTTP é um protocolo de comunicação não seguro, o que significa que as informações enviadas por meio dele podem ser vulneráveis a ataques cibernéticos.

Quanto à conveniência dessas tecnologias para a computação cliente/servidor em geral, o HTTP é amplamente utilizado como protocolo de comunicação em muitas aplicações cliente/servidor, pois é fácil de implementar e é amplamente suportado por muitas linguagens de programação. No entanto, para aplicações mais complexas ou sensíveis à segurança, outros protocolos de comunicação, como o HTTPS, podem ser mais apropriados.

Em resumo, o HTML, URL e HTTP são tecnologias amplamente adotadas para navegação na web, mas têm limitações em termos de funcionalidade e segurança. O HTTP é amplamente utilizado como protocolo de comunicação para aplicações cliente/servidor, mas pode ser necessário usar outros protocolos de comunicação para aplicações mais complexas ou sensíveis à segurança.

7. Um programa servidor escrito em uma linguagem (por exemplo, C++) fornece a implementação de um objeto BLOB destinado a ser acessado por clientes que podem estar escritos em outra linguagem (por exemplo, Java). Os computadores cliente e servidor podem ter hardware diferente, mas todos eles estão ligados em uma rede. Descreva os problemas devidos a cada um dos cinco aspectos da heterogeneidade que precisam ser resolvidos para que seja possível um objeto cliente invocar um método no objeto servidor.

Existem cinco aspectos de heterogeneidade que precisam ser considerados para que um objeto cliente possa invocar um método no objeto servidor em um ambiente de rede distribuído:

1. Heterogeneidade de linguagem: o cliente pode estar escrito em uma linguagem diferente daquela usada pelo servidor. Isso pode levar a problemas de compatibilidade entre as linguagens, como diferenças na representação de dados ou chamadas de métodos.

2. Heterogeneidade de plataforma: o cliente e o servidor podem estar em plataformas diferentes, o que pode levar a problemas de compatibilidade de hardware e software. Isso pode afetar a capacidade de comunicação entre os sistemas, a disponibilidade de bibliotecas de software necessárias e a execução do código em diferentes arquiteturas de hardware.

3. Heterogeneidade de protocolo: o cliente e o servidor podem estar usando diferentes protocolos de comunicação, o que pode levar a problemas de interoperabilidade.

Isso pode afetar a capacidade de comunicação entre os sistemas, a forma como os dados são transmitidos e interpretados, e a segurança da comunicação.

4. Heterogeneidade de formato de dados: o cliente e o servidor podem usar diferentes formatos de dados para representar informações, o que pode levar a problemas de compatibilidade. Isso pode afetar a forma como os dados são interpretados e processados pelo servidor e pelo cliente, levando a erros ou perda de informações.

5. Heterogeneidade de tempo: o cliente e o servidor podem estar em diferentes fusos horários, o que pode levar a problemas de sincronização. Isso pode afetar a forma como as informações são interpretadas e processadas pelo servidor e pelo cliente, levando a erros ou inconsistências nos dados.

Para resolver esses problemas de heterogeneidade, é necessário implementar mecanismos de adaptação e interoperabilidade que permitam que o cliente e o servidor se comuniquem de forma eficaz. Isso pode envolver o uso de ferramentas de middleware que traduzem entre diferentes linguagens e protocolos, a padronização de formatos de dados e protocolos de comunicação, e a sincronização de relógios em diferentes fusos horários.

8. Um sistema distribuído aberto permite que novos serviços de compartilhamento de recursos (como o objeto BLOB do Exercício anterior) sejam adicionados e acessados por diversos programas clientes. Discuta, no contexto desse exemplo, até que ponto as necessidades de abertura do sistema diferem das necessidades da heterogeneidade.

No contexto desse exemplo, as necessidades de abertura do sistema e da heterogeneidade estão relacionadas, mas são diferentes. A abertura do sistema se refere à capacidade de permitir que novos serviços sejam adicionados ao sistema distribuído sem exigir modificações significativas no sistema existente ou nos clientes que utilizam esses serviços. Por outro lado, a heterogeneidade refere-se às diferenças entre os clientes e os servidores, incluindo diferenças de hardware, sistemas operacionais, linguagens de programação, protocolos de rede, etc.

No caso específico do objeto BLOB, a heterogeneidade pode se manifestar em vários aspectos, como a representação interna de dados, a codificação de caracteres, a forma de passar parâmetros e valores de retorno, e assim por diante. Para permitir que um cliente escrito em uma linguagem específica possa invocar métodos em um objeto BLOB fornecido por um servidor escrito em outra linguagem, é necessário lidar com essas diferenças.

No entanto, a abertura do sistema é igualmente importante, porque permite que novos serviços de compartilhamento de recursos sejam adicionados ao sistema sem que seja necessário modificar ou atualizar todos os clientes existentes. Isso é particularmente importante em sistemas distribuídos abertos, nos quais diferentes organizações podem contribuir com serviços e recursos que são usados por outros participantes do sistema. A abertura do sistema também pode permitir que novos serviços sejam adicionados rapidamente em resposta a mudanças nas necessidades dos usuários ou no ambiente operacional.

Em resumo, a heterogeneidade e a abertura do sistema são duas questões importantes em sistemas distribuídos, e ambas precisam ser abordadas para garantir a interoperabilidade e a flexibilidade do sistema.

9. Suponha que as operações do objeto BLOB sejam separadas em duas categorias – operações públicas que estão disponíveis para todos os usuários e operações protegidas, que estão disponíveis somente para certos usuários nomeados. Indique todos os problemas envolvidos para se garantir que somente os usuários nomeados possam usar uma operação protegida. Supondo que o acesso a uma operação protegida forneça informações que não devem ser reveladas para todos os usuários, quais outros problemas surgem?

Para garantir que somente os usuários nomeados possam usar uma operação protegida, são necessários mecanismos de autenticação e autorização. O mecanismo de autenticação deve verificar se o usuário é quem ele diz ser, e o mecanismo de autorização deve verificar se o usuário tem permissão para acessar a operação protegida.

Alguns problemas que podem surgir incluem:

1. Garantir que apenas usuários autorizados tenham acesso à operação protegida.
2. Proteger as informações fornecidas pela operação protegida, para que não sejam acessadas por usuários não autorizados.
3. Gerenciar a lista de usuários autorizados e suas permissões, garantindo que seja atualizada e precisa.
4. Verificar a integridade das informações fornecidas pelos usuários, para garantir que elas sejam confiáveis e autênticas.
5. Garantir que os usuários não autorizados não possam acessar a operação protegida por meio de ataques de força bruta ou outras técnicas de hacking.

Para proteger as informações fornecidas pela operação protegida, pode ser necessário utilizar técnicas de criptografia para proteger as informações em trânsito e em repouso. Além disso, é importante garantir que os usuários não possam acessar as informações de outros usuários, por meio de técnicas de segurança, como autenticação de usuário e isolamento de dados.

10. O serviço INFO gerencia uma grande quantidade de recursos, cada um dos quais podem ser acessados por usuários através da Internet por meio de uma chave (uma string, por exemplo). Discuta uma abordagem para projetar os nomes dos recursos que resultem na mínima perda de desempenho quando o número de recursos no serviço aumenta. Sugira como o serviço INFO pode ser implementado para evitar gargalos de desempenho quando o número de usuários torna-se muito grande.

Para projetar os nomes dos recursos de forma a minimizar a perda de desempenho, podemos adotar uma abordagem hierárquica de nomes, na qual os recursos são agrupados em subgrupos e sub-subgrupos, seguindo uma estrutura de árvore. Cada nó da árvore seria um grupo de recursos e cada chave seria uma concatenação de nomes de nós pais e filhos, por exemplo: "grupo1.subgrupo1.subsubgrupo1.recursoid".

Para evitar gargalos de desempenho quando o número de usuários aumenta, o serviço INFO pode ser implementado utilizando técnicas de escalabilidade horizontal, como a replicação do serviço em múltiplos servidores. Nesse caso, os usuários poderiam ser distribuídos de forma balanceada entre os diferentes servidores, de acordo com a carga de processamento de cada um. Além disso, o serviço poderia utilizar técnicas de cache para reduzir o número de acessos ao banco de dados, armazenando em memória as informações mais frequentemente acessadas. Outra técnica seria o uso de balanceadores de carga para distribuir os acessos dos usuários entre os diferentes servidores disponíveis.

11. Liste os três principais componentes de software que podem falhar quando um processo cliente chama um método em um objeto servidor, dando um exemplo de falha em cada caso. Sugira como os componentes podem ser feitos de modo a tolerar as falhas uns dos outros.

Os três principais componentes de software que podem falhar em um processo cliente que chama um método em um objeto servidor são:

1. Rede: Pode haver uma falha de comunicação entre o cliente e o servidor, como um cabo desconectado ou uma rota de rede inacessível. Nesse caso, uma estratégia para tolerar essa falha é utilizar redundância de rede, como a duplicação de links e o roteamento de backup.

2. Servidor: Pode ocorrer uma falha no servidor que pode ser causada por um erro de software, falta de energia ou outros problemas de hardware. Nesse caso, uma estratégia para tolerar essa falha é utilizar múltiplos servidores para fornecer redundância e replicação de dados para garantir a disponibilidade contínua do serviço.

3. Aplicação: Pode ocorrer uma falha na aplicação do cliente ou servidor que pode ser causada por um erro de programação, um problema de configuração ou outros fatores. Nesse caso, uma estratégia para tolerar essa falha é utilizar técnicas de depuração para localizar e corrigir o problema, e também monitorar a aplicação para alertar os administradores de sistema sobre possíveis falhas.

Para aumentar a tolerância a falhas, os componentes podem ser projetados para detectar falhas e se recuperar delas de forma autônoma. Por exemplo, o servidor pode ser projetado para ser capaz de detectar falhas de hardware e comutar para um servidor de backup automaticamente. O cliente pode ser projetado para reconectar automaticamente em caso de perda de conexão. Além disso, as aplicações podem ser projetadas para lidar com falhas, por exemplo, restaurando o estado do sistema a partir de um backup ou reconstruindo dados perdidos.

12. Um processo servidor mantém um objeto de informação compartilhada, como o objeto BLOB do Exercício 7. Dê argumentos contra permitir que os pedidos do cliente sejam executados de forma concorrente pelo servidor e a favor disso. No caso de serem executados de forma concorrente, dê um exemplo de uma possível

“interferência” que pode ocorrer entre as operações de diferentes clientes. Sugira como essa interferência pode ser evitada.

Permitir que os pedidos do cliente sejam executados de forma concorrente pelo servidor pode aumentar o desempenho e a capacidade de resposta do sistema, permitindo que vários clientes usem o objeto compartilhado simultaneamente. No entanto, isso pode levar a condições de corrida, onde duas ou mais operações concorrentes acessam e modificam o mesmo objeto de forma intercalada, levando a resultados imprevisíveis e possíveis falhas no sistema.

Um exemplo de possível interferência que pode ocorrer entre as operações de diferentes clientes é quando dois clientes tentam acessar e modificar o mesmo objeto de BLOB ao mesmo tempo, levando a uma inconsistência nos dados armazenados.

Para evitar a interferência, o servidor pode utilizar mecanismos de sincronização, como locks, semáforos ou monitores, para garantir que apenas um cliente possa acessar o objeto compartilhado de cada vez. Além disso, o servidor pode implementar técnicas de transação para garantir a atomicidade, consistência, isolamento e durabilidade das operações realizadas pelos clientes no objeto compartilhado.

13. Um serviço é implementado por vários servidores. Explique por que recursos poderiam ser transferidos entre eles. Seria satisfatório para os clientes fazer multicast (difusão seletiva) de todos os pedidos para o grupo de servidores, como uma maneira de proporcionar transparência de mobilidade para os clientes?

Recursos poderiam ser transferidos entre servidores por diversas razões, tais como balanceamento de carga, falhas de servidor, necessidade de escalabilidade ou para permitir a manutenção em um servidor sem interrupção do serviço.

No entanto, fazer multicast de todos os pedidos para o grupo de servidores não seria uma solução satisfatória para proporcionar transparência de mobilidade para os clientes. Isso porque o multicast envolve enviar mensagens para um grupo de receptores simultaneamente e, como resultado, todos os servidores no grupo receberiam todos os pedidos de todos os clientes. Isso poderia causar um aumento na latência e na sobrecarga do sistema, uma vez que os servidores precisariam processar e responder a todos os pedidos, mesmo aqueles que não são destinados a eles.

Uma abordagem melhor seria implementar um mecanismo de gerenciamento de serviço que permita que os clientes possam ser redirecionados para outros servidores de forma transparente, sem afetar o desempenho e a qualidade do serviço. Isso pode ser feito, por exemplo, por meio de um servidor de balanceamento de carga, que encaminha os pedidos do cliente para o servidor mais apropriado com base em critérios como carga do servidor, localização geográfica, entre outros. Assim, os recursos podem ser transferidos entre servidores de forma eficiente, sem que isso afete a experiência do usuário ou a qualidade do serviço prestado.

14. Os recursos na World Wide Web e outros serviços são nomeados por URLs. O que denotam as iniciais URL? Dê exemplos de três diferentes tipos de recursos da Web que podem ser nomeados por URLs.

As iniciais URL significam "Uniform Resource Locator", em português, "Localizador Uniforme de Recursos". Um URL é um endereço que identifica de forma exclusiva um recurso na Internet.

Aqui estão três exemplos de recursos da Web que podem ser nomeados por URLs:

1. Página da Web: https://www.google.com
2. Arquivo de imagem: https://www.example.com/image.jpg
3. Vídeo no YouTube: https://www.youtube.com/watch?v=dQw4w9WgXcQ

No primeiro exemplo, o URL identifica a página inicial do site Google. No segundo exemplo, o URL identifica um arquivo de imagem em um site chamado "example.com". No terceiro exemplo, o URL identifica um vídeo específico no YouTube pelo seu ID exclusivo.

15. Cite um exemplo de URL HTTP. Liste os principais componentes de um URL HTTP, dizendo como seus limites são denotados e ilustrando cada um, a partir de seu exemplo. Até que ponto um URL HTTP tem transparência de localização?

Um exemplo de URL HTTP é: http://www.exemplo.com.br/pagina1.html

Os principais componentes de um URL HTTP são:

- Protocolo: no exemplo acima, o protocolo é "http".
- Domínio: neste caso, o domínio é "www.exemplo.com.br". Ele identifica o servidor onde a página está hospedada.
- Caminho: o caminho é "/pagina1.html", que indica a localização do recurso dentro do servidor.
- Porta: se necessário, um número de porta pode ser especificado na URL para acessar um serviço específico no servidor.
- Parâmetros: podem ser adicionados à URL para passar informações extras para o servidor.

O limite entre os componentes da URL é denotado por símbolos específicos, por exemplo:

- O limite entre o protocolo e o domínio é denotado por "://".
- O limite entre o domínio e o caminho é denotado por "/".
- O limite entre o domínio e o número de porta é denotado por ":".

Um URL HTTP tem transparência de localização em certa medida, pois o domínio identifica o servidor onde o recurso está hospedado, mas não necessariamente sua localização física. No entanto, pode haver casos em que a localização precisa ser especificada na URL, como quando se acessa um recurso em um servidor que está atrás de um balanceador de carga.