**Aplicações Móveis**

Exercícios de Revisão

**Coordenação de Engenharia Informática**

Departamento de Engenharias e Tecnologias

Instituto Superior Politécnico de Tecnologias e Ciências

Nome : Marcelo Rocha - 20210032

# Adaptabilidade

1. **Considere a noção de adaptabilidade. Apresente a sua definição e dê alguns exemplos.**

**R:** A adaptabilidade é a capacidade de um sistema ou processo se ajustar automaticamente a condições de trabalho variáveis, mantendo a melhor qualidade de serviço possível tornando o sistema capaz de se adaptar às influências de condições externas, proporcionando uma experiência aceitável para o usuário. Exemplos:

* Assistir video no YouTube durante a viagem de taxi
* Acesso de ficheiros de um laptop com disponibilidade GSM/3G/4G
* Lendo e-mail em uma conexão de rede lenta

1. **É bem-sabido que o particionamento estático (*static partitioning*) falha em levar em conta a variabilidade inerente aos ambientes das aplicações móveis. Apresente cinco casos que justifiquem a frase anterior.**

**R:**

* **Conexão de Rede Variável:** Em ambientes móveis, a qualidade e a disponibilidade da rede podem mudar durante a execução do aplicativo. O particionamento estático não consegue se adaptar a essas mudanças, o que pode levar a falhas ou a um desempenho inferior, já que ele não ajusta a alocação de recursos de acordo com a conectividade disponível.
* **Mudança na Disponibilidade de Memória**: Em dispositivos móveis, a quantidade de memória disponível pode variar dependendo do uso de outros aplicativos ou do sistema. O particionamento estático não pode responder a essas variações, causando possível sobrecarga ou falhas quando o aplicativo excede a memória alocada.
* **Variação na Largura de Banda:** Em redes móveis, a largura de banda pode ser reduzida ou alterada conforme o ambiente. O particionamento estático não pode ajustar a distribuição de dados para acomodar essas mudanças, resultando em uma experiência de usuário degradada ou falhas na execução de tarefas.
* **Variação de Localização:** Aplicações móveis frequentemente dependem de informações de localização, como GPS. O particionamento estático não leva em conta mudanças na localização do dispositivo, podendo resultar em ações incorretas ou na não otimização do desempenho em áreas de cobertura variável.
* **Consumo de Energia:** A bateria dos dispositivos móveis é limitada e seu consumo varia conforme a execução do aplicativo e o uso de recursos. O particionamento estático não considera a variabilidade no consumo de energia, o que pode levar ao esgotamento prematuro da bateria e interrupção do serviço, sem uma adaptação dinâmica às necessidades energéticas.

1. **Considere a necessidade de uma aplicação móvel se adaptar automaticamente ao contexto. Apresente um exemplo em que a alta-velocidade em que um sistema é capaz de se adaptar é boa e outro exemplo em que essa alta-velocidade não é boa.**

**R:** A alta velocidade de adaptação em aplicações móveis pode ser tanto benéfica quanto prejudicial, dependendo do contexto.

Um exemplo onde alta velocidade é boa seria a navegação por GPS. Em um sistema de navegação, a alta velocidade de adaptação é crucial para recalcular rotas rapidamente ao detectar mudanças de direção ou imprevistos, como trânsito ou desvios. Isso permite que o usuário tenha direções atualizadas em tempo real, o que é essencial para uma navegação eficaz.

Um exemplo onde alta velocidade não é boa seria a mudança automática de brilho da tela. Se um sistema ajustar o brilho da tela rapidamente de acordo com mudanças mínimas de luz ambiente, isso pode resultar em uma experiência incômoda, com mudanças constantes e abruptas de brilho que distrai o usuário. Um tempo de resposta mais lento pode proporcionar uma experiência mais confortável e fluida.

1. **Considere as noções de aplicação *mobile-transparent* e *mobile-aware* e a seguinte frase: “Esta abordagem visa permitir que as aplicações se adaptem activamente às condições do ambiente móvel”. A que noção (das duas anteriores) esta frase se aplica? Como a transparência é alcançada se a outra noção for seguida?**

**R:** A frase "Esta abordagem visa permitir que as aplicações se adaptem ativamente às condições do ambiente móvel" se aplica à noção de mobile-aware.

**Mobile-aware** refere-se à capacidade da aplicação de estar ciente e se adaptar ativamente às mudanças nas condições do ambiente móvel, como variações de rede, recursos disponíveis, localização e outros fatores contextuais. Isso implica que a aplicação ajusta seu comportamento de forma dinâmica e inteligente com base nas condições externas, como a necessidade de otimizar o uso da rede ou ajustar o consumo de energia dependendo do contexto.

Já a noção de **mobile-transparent** implica que as mudanças no ambiente móvel (como a variação de rede ou de recursos) são tratadas de forma automática e transparente para o usuário, ou seja, a aplicação não precisa se adaptar conscientemente, e o usuário não percebe essas adaptações. A transparência é alcançada, então, quando o sistema ou a plataforma de execução da aplicação gerencia automaticamente essas condições sem que a aplicação precise fazer ajustes ativos ou o usuário precise se preocupar com isso. Ou seja, a aplicação funciona sem a necessidade de tomar decisões conscientes sobre essas variações.

1. **Considere um sistema que, para fins de adaptabilidade, suporte a computação *offloading* de um dispositivo móvel para um servidor na nuvem em tempo de execução, ou seja, enquanto uma aplicação está em execução. Supondo que a aplicação seja escrita em Java, isso pode ser feito sem modificar a JVM?**

**R:** É possível realizar computação offloading de um dispositivo móvel para um servidor na nuvem enquanto uma aplicação está em execução, sem modificar a JVM, utilizando abordagens como o uso de APIs de nuvem e tecnologias de comunicação remota (como HTTP ou WebSockets), que permitem a transferência de tarefas e dados para a nuvem. Além disso, a aplicação pode empregar frameworks ou bibliotecas de terceiros que facilitam a integração com a nuvem, sem alterações diretas na JVM. Embora não seja necessário modificar a JVM, a implementação do offloading exige ajustes na lógica da aplicação para gerenciar o envio de dados e a execução de tarefas remotamente de forma eficiente.

1. **Para descoberta de serviço, o SLP (Service Location Protocol) é uma solução possível. Qual é o mecanismo usado para lidar com agentes de serviço malcomportados que não cancelam o registo de serviço quando não é mais oferecido?**

**R:** No contexto do Service Location Protocol (SLP), o mecanismo utilizado para lidar com agentes de serviço malcomportados (que não cancelam o registro de serviço quando não estão mais oferecendo o serviço) é o registro com expiração (registration expiration).

SLP usa um sistema baseado em tempo de vida (TTL - Time To Live) para gerenciar os registros de serviços. Quando um agente de serviço registra um serviço, ele define um tempo de expiração para esse registro. Se o agente de serviço não renovar ou cancelar o registro antes do tempo de expiração, o registro é automaticamente removido do diretório de serviços (Service Directory). Isso ajuda a garantir que o serviço não fique registrado indefinidamente se o agente não estiver mais oferecendo o serviço.

Esse mecanismo de expiração permite que os clientes descubram serviços de forma confiável, mesmo que os agentes de serviço falhem em cancelar corretamente seus registros.

1. **Considere a seguinte afirmação: “As aplicações podem ser adaptadas automaticamente às condições variadas dos ambientes móveis por meio de ajuste fino exclusivo do sistema operativo; portanto, nenhuma alteração é necessária no código-fonte das aplicações.”. Concorda com esta afirmação? Ilustre seu ponto de vista com um exemplo.**

**R:** Não concordo plenamente com a afirmação de que as aplicações podem ser adaptadas automaticamente às condições dos ambientes móveis apenas por ajustes feitos pelo sistema operacional, sem que haja necessidade de alterações no código-fonte da aplicação. Embora sistemas operacionais modernos, como Android e iOS, ofereçam mecanismos que permitem ao dispositivo ajustar automaticamente certos parâmetros — como o uso de CPU, memória e rede — para se adaptar a condições como conectividade fraca ou consumo de bateria, a adaptação completa e eficiente de uma aplicação frequentemente exige ajustes no próprio código. Por exemplo, uma aplicação de streaming de vídeo pode se beneficiar do ajuste automático da qualidade do vídeo de acordo com a largura de banda disponível, mas para isso, a aplicação precisa ser programada para detectar e reagir a essas mudanças. Sem essas implementações no código, a aplicação não seria capaz de responder adequadamente a condições de rede instáveis ou de baixo desempenho, prejudicando a experiência do usuário. Em resumo, embora o sistema operacional ofereça suporte, muitas vezes a aplicação precisa ser projetada para interagir de maneira inteligente com essas condições variáveis, garantindo um desempenho ideal em ambientes móveis.

1. **Uma possível solução para lidar com a falta de rede e permitir que o utilizador continue a trabalhar com o seu dispositivo móvel é fornecida pelo sistema Rover.**

* 1. **Quais são as duas técnicas mais relevantes propostas pela Rover? Como funcionam?**
  2. **A Rover lida com conflitos? Descreva um cenário em que tal situação de conflito possa surgir e explique como a Rover lida com isso.**

**R:** As duas técnicas mais relevantes propostas pelo Rover para lidar com a falta de rede e permitir que o usuário continue a trabalhar com seu dispositivo móvel são:

* **Replicação de Dados:** O Rover utiliza a replicação de dados para garantir que os dados necessários estejam disponíveis localmente no dispositivo móvel, mesmo quando a conexão com a rede não está disponível. Isso permite que os usuários acessem e modifiquem os dados localmente. Quando a conexão é restabelecida, as alterações podem ser sincronizadas com o servidor.
* **Tentativas de Atualizações (Tentative Updates):** Rover permite que os usuários façam atualizações em dados que estão armazenados localmente, mesmo quando estão desconectados. Essas atualizações são consideradas "tentativas" até que a conexão com a rede seja restabelecida e as alterações possam ser enviadas ao servidor. Isso minimiza a dependência de uma conexão constante e permite que os usuários continuem a trabalhar sem interrupções.

O Rover lida com conflitos que podem surgir durante a sincronização de dados. Um cenário em que um conflito pode ocorrer é quando dois usuários, um em um dispositivo móvel e outro em um servidor, fazem alterações no mesmo dado enquanto estão desconectados. Por exemplo, se um usuário modifica um documento em seu dispositivo móvel e, ao mesmo tempo, outro usuário altera o mesmo documento no servidor, quando a conexão for restabelecida, haverá um conflito entre as duas versões do documento.

Para resolver esse conflito, o Rover implementa um mecanismo de resolução de conflitos que pode incluir:

* **Resolução Automática:** O sistema pode aplicar regras predefinidas para determinar qual versão do dado deve prevalecer. Por exemplo, pode-se optar pela versão mais recente com base em timestamps.
* **Interação do Usuário:** Em casos onde a resolução automática não é possível, o Rover pode notificar o usuário sobre o conflito e solicitar que ele escolha qual versão do dado deve ser mantida ou como as alterações devem ser mescladas.

Essas abordagens ajudam a garantir a integridade dos dados e a continuidade do trabalho do usuário, mesmo em situações de desconexão e conflito.

1. **Que tipo de variações nos recursos devem ser levados em consideração?**

**R:** Nos ambientes de computação móvel e ubíqua, as variações nos recursos podem ser tanto qualitativas quanto quantitativas. As variações qualitativas incluem a conexão ou desconexão de rede, a presença de dispositivos específicos, como impressoras ou periféricos próximos, e mudanças nas configurações de segurança ou inconsistências nos dados. Já as variações quantitativas envolvem fatores como a largura de banda utilizável, que pode variar de acordo com a qualidade da rede, e a quantidade de memória disponível, que pode ser insuficiente em determinadas situações. Essas mudanças constantes exigem que as aplicações sejam projetadas para serem adaptáveis, permitindo que lidem com diferentes cenários sem que os programadores precisem prever e codificar manualmente todas as possibilidades, o que seria ineficiente e propenso a erros.

1. **Para quais recursos um sistema deve ser adaptável?**

**R:** Um sistema deve ser adaptável a recursos tanto do sistema quanto do ambiente. Entre os recursos do sistema, destacam-se a rede, a energia, a CPU e a memória, pois esses podem variar conforme o uso e as condições do dispositivo. Do lado ambiental, o sistema também precisa se adaptar a variáveis como luminosidade, temperatura e localização, pois esses fatores podem influenciar o desempenho e a funcionalidade da aplicação. Em resumo, um sistema adaptável deve considerar todos os recursos cuja qualidade ou disponibilidade pode mudar ao longo do tempo ou em diferentes locais, enquanto a aplicação está em execução, garantindo uma experiência contínua e eficiente para o usuário.

1. **Como podemos medir a eficácia dos mecanismos de adaptabilidade? Explique o que é cada um.**

**R:** A eficácia dos mecanismos de adaptabilidade pode ser medida por várias métricas, cada uma focada em um aspecto diferente do desempenho do sistema em situações dinâmicas. As principais métricas incluem:

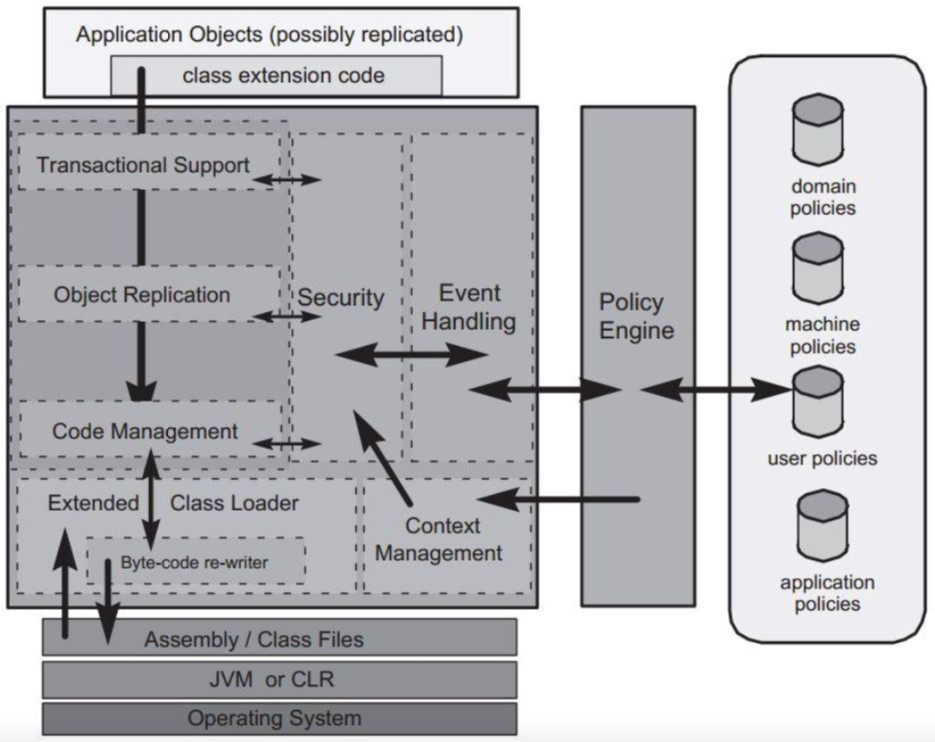
* **Satisfação do usuário:** Avalia a experiência do usuário ao interagir com a aplicação. Mede o grau de aceitação e conforto com as mudanças adaptativas realizadas pelo sistema, refletindo o impacto da adaptabilidade na usabilidade e eficiência percebida.
* **Alterações "Pico" na disponibilidade de recursos:** Refere-se a quão bem o sistema lida com picos ou quedas repentinas na disponibilidade de recursos (como largura de banda, memória ou energia). Essa métrica verifica a capacidade do sistema de continuar operando eficazmente durante variações inesperadas nos recursos.
* **Velocidade para reagir:** Mede a rapidez com que o sistema se adapta a mudanças nas condições de recursos ou no ambiente. Quanto mais rápido for o sistema em se ajustar a novas circunstâncias, melhor será sua performance em ambientes móveis e dinâmicos.
* **Estabilidade do sistema:** Reflete a capacidade do sistema de manter sua funcionalidade sem falhas ou instabilidade, mesmo diante de mudanças nos recursos ou no ambiente. Sistemas com boa estabilidade podem se adaptar sem comprometer o funcionamento geral.
* **Portabilidade das soluções:** Avalia a capacidade da solução de ser executada em diferentes plataformas, dispositivos ou ambientes. Uma boa portabilidade significa que o sistema pode adaptar-se não apenas às variações de recursos, mas também a diferentes configurações de hardware ou software.
* **Uso de recursos das soluções de adaptabilidade:** Refere-se à eficiência no uso dos recursos ao aplicar mecanismos de adaptabilidade. Um sistema eficaz deve ser capaz de adaptar-se sem consumir excessivamente recursos como CPU, memória ou energia, garantindo um equilíbrio entre desempenho e eficiência.

1. **Em que nível a solução de adaptabilidade deve ser implementada? Quais são os pros e contra de cada uma?**

**R:** A solução de adaptabilidade pode ser implementada em três níveis: sistema operativo, middleware e aplicação. Cada um desses níveis tem suas vantagens e desvantagens:

* **Sistema Operativo:**
* **Vantagens:**
* **Alheio à semântica da aplicação:** O sistema operativo lida com a adaptação sem precisar entender o conteúdo da aplicação, o que facilita sua implementação em uma variedade de cenários.
* **Soluções globais e iguais para todos:** As adaptações aplicadas ao sistema operativo podem ser uniformes, garantindo que todos os programas rodem de maneira otimizada em diferentes condições de recursos.
* **Portabilidade:** Soluções baseadas no sistema operativo podem ser mais portáveis entre diferentes plataformas, uma vez que o sistema operativo é uma camada comum entre as aplicações.
* **Desvantagens:**
* **Sem reconhecimento da aplicação:** O sistema operativo não tem uma compreensão profunda dos requisitos específicos de cada aplicação, o que pode levar a soluções menos precisas ou eficientes em alguns casos.
* **Limitação na personalização:** A solução oferecida pelo sistema operativo pode ser muito genérica e não considerar as necessidades específicas de cada aplicação.
* **Middleware:**
* **Vantagens:**
* **Mecanismos comuns reutilizáveis:** O middleware permite o uso de mecanismos de adaptabilidade que podem ser reutilizados em várias aplicações, o que reduz o esforço de desenvolvimento.
* **Interação com as aplicações:** O middleware pode ser projetado para ser semanticamente consciente, ou seja, ele pode entender as necessidades das aplicações e ajustar-se conforme necessário.
* **Portabilidade:** Como uma camada intermediária, o middleware pode ser portável entre diferentes sistemas, mantendo a flexibilidade em múltiplos ambientes.
* **Desvantagens:**
* **Complexidade adicional:** A adição de middleware pode aumentar a complexidade do sistema, exigindo mais tempo e recursos para o desenvolvimento e manutenção.
* **Necessidade de integração com as aplicações:** O middleware precisa interagir com as aplicações para entender suas necessidades, o que pode ser um desafio, especialmente em sistemas muito dinâmicos ou heterogêneos.
* **Aplicação:**
* **Vantagens:**
* **Semanticamente consciente:** A aplicação pode adaptar-se de maneira muito mais precisa, levando em conta suas necessidades específicas e otimizando os recursos de acordo com o contexto.
* **Personalização total:** Como a adaptação ocorre no nível da aplicação, é possível criar soluções altamente customizadas que atendem exatamente às exigências da aplicação em termos de performance e uso de recursos.
* **Desvantagens:**
* **Não portável:** Soluções adaptativas implementadas diretamente na aplicação são específicas para aquela aplicação e podem não ser reutilizáveis em outras, o que dificulta a portabilidade entre diferentes sistemas.
* **Dependência de programação detalhada:** Implementar adaptabilidade diretamente na aplicação exige que os desenvolvedores considerem muitos cenários possíveis, o que pode ser ineficiente e propenso a erros.

1. **Descreva a figura a seguir (possível arquitectura genérica para adaptabilidade) e explique seus componentes.**

****

**R:** É possível lidar com a variabilidade através de adaptação automática de aplicações. Para tal, o mediador deve fornecer flexibilidade para o desenvolvimento de aplicação e adaptabilidade em runtime. Assim, as aplicações podem lidar com os vários requisitos e a diversidade de uso encontrados nas configurações móveis.

* **Motor de Política (Policy Engine):** O motor de política é um componente de inferência responsável por desencadear ou mediar respostas a eventos que ocorrem no sistema. Ele opera com base em um conjunto variável de políticas aplicadas ao sistema, funcionando da seguinte maneira:
* Recebe eventos gerados por módulos e aplicações de middleware.
* Avalia regras de política previamente definidas.
* Dispara eventos e executa ações com base nos resultados da avaliação.

Por exemplo, a replicação de um objeto pode ser realizada de acordo com as políticas especificadas no sistema.

* **Gestão de Contexto (Context Management) :** realiza a abstração de recursos e gerencia propriedades cujos valores podem variar durante a execução. Situações como a detecção de dispositivos que aparecem ou desaparecem e a descoberta de recursos remotos ou contrapartes de aplicações permitem que middleware e aplicações tomem decisões, como replicar dados de diferentes fontes ou trocar informações entre dispositivos. Em termos gerais, qualquer alteração nas propriedades gerenciadas pelo gestor de contexto, seja de recursos, estado do middleware ou definições do utilizador, pode acionar eventos associados, definidos pelas políticas carregadas no sistema.

1. **Que tipo de políticas podem ser previstas? Explique cada uma.**

**R:**

* **Consistência:** Define se versões desatualizadas de objetos podem ser usadas quando o nó remoto correspondente não está disponível. Essa política pode ser aplicada a objetos individualmente ou a um conjunto de objetos, garantindo maior flexibilidade na gestão de dados em cenários com falhas de conectividade.
* **Fetching (Busca):** Estabelece se e quantos objetos devem ser pré-buscados antes da execução de uma transação ou se os objetos devem ser buscados sob demanda, durante a execução. Isso permite ajustar o comportamento do sistema para otimizar o desempenho e a disponibilidade dos dados.
* **Delegação:** Permite transferir a responsabilidade de confirmar uma transação/sessão para outro nó, caso seja necessário. Isso é especialmente útil em situações onde o nó original está indisponível ou com recursos limitados.
* **Atomicidade:** Define se uma transação/sessão pode ser confirmada mesmo que nem todos os nós envolvidos sejam alcançáveis. Por exemplo, é possível descartar algumas alterações e ainda assim concluir a transação, garantindo flexibilidade e continuidade em ambientes móveis dinâmicos.
* **Cache:** Possibilita o armazenamento de objetos buscados ou confirmados por transacções/sessões localmente no cache de um nó. Esse recurso é essencial para permitir acesso aos dados mesmo durante desconexões, promovendo resiliência e eficiência no acesso às informações.
* **Gestão de Falhas:** Determina como o sistema deve reagir quando as condições especificadas de consistência, busca e atomicidade não são atendidas. Essa política ajuda a minimizar impactos causados por falhas ou inconsistências, promovendo uma operação mais estável.

1. **Porquê o termo “sessões” é usado em vez de “transações”?**

**R:** O termo "sessões" é usado em vez de "transações" para refletir melhor a flexibilidade e adaptabilidade necessárias em ambientes móveis dinâmicos, onde conectividade intermitente, falhas de comunicação e entrada ou saída frequente de nós tornam inviável o uso de transações tradicionais, que possuem regras rígidas como as propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade) . Sessões permitem maior controle e personalização, como definir políticas específicas para consistência, atomicidade, cache e delegação, além de evitar abortos desnecessários ao oferecer alternativas para tratar desconexões ou falhas. Diferentemente das transações, que tendem a abortar integralmente diante de qualquer problema, as sessões podem ajustar seu comportamento dinamicamente em tempo de execução, adaptando-se às mudanças no ambiente ou aos requisitos da aplicação, garantindo resiliência e eficiência em operações críticas.