Estilos e padrões de arquitetura de SW

Leonardo Rodrigues Marques - 178610

1. O projeto é uma Agenda Médica Online 2.0 destinado para os médicos, secretários dos médicos e para os pacientes, portanto uma distribuição de diferentes interfaces seria adequado a essa situção. A arquitetura que oferece essa abstração é a **Orientada a Serviços.** Outra arquitetura adequada ao projeto da agenda é o Estilo **Cliente-Servidor.** Ela é justificada devido ao seu comportamento bidirecional(servidor -><- cliente) entre clientes e servidor através de uma rede. O cliente(médico, paciente, secretário) tem a função de escrever dados (marcar uma consulta) ou puxar dados(consultar disponibilidade) no servidor.Não há interação computacional entre clientes e clientes. Uma arquitetura complementar que poderia ser empregada é a **Layered**(Camadas), bastante útil no desenvolvimento de front-end de Web Apps.
2. Para o projeto em questão foram escolhidos quatro padrões:
   1. Padrão Criativo(Creational Pattern): nesse projeto, têm-se vários objetos a serem declarados, portanto seria interessante um padrão que lidasse com os mecanismo de criação desses elementos de forma mais adequada.
   2. Padrão de Lembrança(Memento Pattern): não está claro esse padrão no projeto, mas ao meu ver restaurar um estado anterior seria uma menção a um breve histórico do paciente? Uma nova receita(objeto) indica aumento na dose de um remédio para o paciente. Caso o paciente sofra efeitos colaterais perversos, é possível recuperar o estado da receita anterior utilizando esse padrão.
   3. Padrão de Estado(State Pattern): esse padrão é bastante aplicável a essa situação. Nesse ambiente de trabalho, uma emergência pode surgir inesperadamente, consequentemente as medidas também mudarão. Exatamente isso que o padrão trata, ou seja, caso o estado de um objeto muda, seu comportamento também muda.
   4. Padrão de Observação(Observer Pattern): uma mudança(cancelamento ou alteração) gera notificação sobre o acontecimento. Esse padrão retrata essa condição. Um objeto mantém uma lista de dependências, chamados de observadores, que são notificados para chamar seus métodos quando uma mudança de estado ocorre.