Atividade de avaliação individual 3

Professor Eng. Dr. Gerson Penha

Introdução

Uma aplicação web pode ser entendida como um software que funciona dentro de um navegador e que depende de um servidor para executar suas funcionalidades. Diferente de um site tradicional, cujo principal objetivo é apresentar informações de forma estática ou semi-estática, a aplicação web vai além da exibição de conteúdo: ela permite a interação do usuário, o processamento de dados, a autenticação de acessos, o armazenamento de informações em bancos de dados e a execução de tarefas específicas. Exemplos comuns de aplicações web são o Aplicativos como: Gmail, Google Docs, Facebook e sistemas de internet banking.

O conceito de aplicação web não nasceu de um único criador, mas sim de uma evolução natural da própria internet. Em 1991, Tim Berners-Lee criou a World Wide Web, que inicialmente se limitava a páginas em HTML, totalmente estáticas. Porém, com a popularização de linguagens de programação como JavaScript e PHP, a partir de meados da década de 1990, tornou-se possível desenvolver páginas dinâmicas, capazes de responder às ações do usuário. Frameworks como o ASP da Microsoft, lançado em 1996, também contribuíram para transformar sites simples em sistemas interativos. Pouco depois, empresas como a Google passaram a apostar em serviços online cada vez mais sofisticados, consolidando a ideia de que a web poderia ser não apenas uma vitrine de conteúdo, mas também uma plataforma para rodar softwares completos.

A diferença entre um website e uma aplicação web está essencialmente no propósito e na forma de interação com o usuário. Um website tem como foco principal disponibilizar informações: notícias, artigos, páginas institucionais ou conteúdos de blogs. Ele pode até ter elementos dinâmicos, mas a essência continua sendo informativa. Já uma aplicação web tem como prioridade oferecer funcionalidades. O usuário não está apenas consumindo informação, mas também interagindo com o sistema, preenchendo formulários, enviando e recebendo dados em tempo real, colaborando em documentos, realizando transações financeiras ou gerenciando tarefas.

Pode-se dizer que toda aplicação web é, em certo sentido, um site, pois também é acessada por meio de um navegador. No entanto, nem todo site é uma aplicação web. A linha que os separa está no fato de que, enquanto o site é pensado para comunicar informações, a aplicação web é projetada para resolver problemas e executar tarefas. Essa distinção explica por que hoje a maioria das plataformas digitais modernas são classificadas como aplicações web, já que o usuário contemporâneo não busca apenas conteúdo, mas também serviços e interatividade diretamente no ambiente online.

Contextualização

A qualidade de uma aplicação web é um aspecto essencial para garantir sua eficiência, confiabilidade, segurança e boa experiência do usuário. Avaliar a qualidade de um sistema não se limita apenas a analisar seu código ou infraestrutura, mas também envolve medir como ele se comporta em situações reais de uso, como responde às

requisições dos usuários e como mantém a integridade e a segurança das informações. Para isso, diversas métricas podem ser extraídas e monitoradas, cada uma oferecendo insights específicos sobre diferentes dimensões da qualidade.

Uma das dimensões mais importantes é o desempenho da aplicação. Métricas como tempo de resposta, latência, throughput e tempo de carregamento de páginas são fundamentais para compreender a rapidez com que o sistema atende às solicitações dos usuários. O tempo de resposta indica quanto tempo a aplicação leva para processar uma requisição, enquanto a latência mede o atraso entre o momento em que o usuário envia uma solicitação e o início da resposta. O throughput, por sua vez, demonstra a capacidade da aplicação em lidar com múltiplas requisições simultâneas, sendo crucial em sistemas de grande porte. Além disso, o monitoramento do uso de recursos, como CPU, memória e largura de banda, ajuda a identificar gargalos e otimizar a performance da aplicação.

Outra dimensão relevante é a confiabilidade, que mede a capacidade do sistema de operar de forma contínua e sem falhas. Métricas como taxa de falhas, tempo médio entre falhas (MTBF), tempo médio para recuperação (MTTR) e disponibilidade (uptime) são essenciais para garantir que o sistema esteja sempre acessível e funcional. Uma aplicação confiável oferece maior segurança aos dados dos usuários e reduz o impacto de possíveis interrupções no serviço, aumentando a confiança do público e a reputação da organização.

A usabilidade também é um fator crítico na avaliação de qualidade. Métricas de usabilidade incluem a taxa de sucesso em tarefas, tempo para concluir ações, número de cliques ou etapas necessárias para realizar uma tarefa e taxa de abandono de usuários. Além disso, indicadores de satisfação, como pesquisas de NPS (Net Promoter Score) ou SUS (System Usability Scale), ajudam a medir a percepção dos usuários sobre a facilidade de uso da aplicação. Uma boa usabilidade não apenas melhora a experiência do usuário, mas também aumenta a eficiência e a produtividade ao interagir com o sistema.

A segurança é outra dimensão essencial, especialmente em aplicações que lidam com dados sensíveis. Métricas de segurança incluem o número de vulnerabilidades identificadas, o tempo médio para corrigir essas falhas, tentativas de ataques bloqueadas e o percentual de requisições seguras via HTTPS. A segurança garante que informações confidenciais sejam protegidas, reduzindo riscos de vazamentos e ataques cibernéticos, e garantindo a conformidade com regulamentações e boas práticas do setor.

A manutenibilidade refere-se à facilidade com que o sistema pode ser modificado, corrigido ou ampliado. Métricas como complexidade ciclomática, cobertura de testes automatizados, densidade de defeitos, tempo médio para corrigir bugs, acoplamento e coesão do código são indicativos de quão bem estruturado está o sistema. Uma boa manutenibilidade facilita atualizações e melhorias futuras, reduzindo custos e tempo de desenvolvimento, além de garantir maior robustez do sistema.

Por fim, a experiência do usuário (UX/UI) é uma dimensão cada vez mais valorizada. Métricas como os Core Web Vitals, propostos pelo Google, ajudam a avaliar a performance percebida pelos usuários, incluindo LCP (Largest Contentful Paint), FID

(First Input Delay) e CLS (Cumulative Layout Shift). Esses indicadores medem o tempo de carregamento do maior elemento visível, a rapidez da resposta às interações e a estabilidade visual da página. Ferramentas como heatmaps e monitoramento de cliques também permitem analisar o comportamento dos usuários dentro da aplicação, enquanto métricas de engajamento, como a taxa de retorno, indicam fidelidade e satisfação do público.

A qualidade de uma aplicação web é um conceito multifacetado, que envolve desempenho, confiabilidade, segurança, manutenibilidade, usabilidade e experiência do usuário. A extração e o monitoramento de métricas específicas permitem às equipes de desenvolvimento identificar pontos de melhoria, otimizar o sistema e oferecer uma experiência segura, eficiente e agradável para os usuários, garantindo o sucesso da aplicação no ambiente digital.

Atividade:

Enfim você terminou o desenvolvimento da GUI do sistema do sistema de gestão da produção de aeronaves. Após muito tempo e esforço aplicado, o projeto da GUI ficou excelente. Por isso, um dos seus melhores funcionários e responsável pelo gerenciamento dos departamentos de marketing e vendas, resolveu apresentar o protótipo navegável em uma feira de soluções na Europa, que reuniu os principais possíveis clientes do setor, onde a Aerocode pretende vender seu sistema. O resultado da apresentação foi um sucesso enorme.

O sucesso da nova GUI foi tamanho que cinco novos contratos foram fechados, com as principais empresas que produzem aeronaves comerciais e militares no mundo. Seus novos clientes são Boeing, Airbus, Embraer, Comac e a Bombardier. Sua empresa, a Aerocode, irá fornecer software para as cinco maiores fabricantes de aeronaves do mundo.

Claramente que o resultado obtido por causa da apresentação na feira lhe deixou extremamente satisfeito, afinal se tornar fornecedor de sistema para as maiores fabricantes de aeronaves do planeta é um feito notável. Contudo, isso trouxe um desafio novo e talvez o maior que você já enfrentou. Agora você precisa finalizar a aplicação, terminar tudo que é necessário para construir uma aplicação web e poder colocar a aplicação disponível para seus clientes.

Dado a importância deste desenvolvimento e sua sabedoria em sempre buscar bons conselhos, você decidiu contratar uma consultoria de uma empresa especializada em desenvolvimentos de aplicações web e sistemas críticos.

Um sistema crítico é um tipo de sistema de computação ou controle cuja falha ou mau funcionamento pode resultar em consequências severas, como prejuízos financeiros significativos, danos à reputação de uma organização, riscos à segurança das pessoas ou impactos ambientais graves. Esses sistemas são essenciais para a operação contínua e segura de atividades estratégicas em diversos setores, incluindo saúde, transporte, energia, finanças e defesa. Por sua natureza, eles exigem altos níveis de confiabilidade, disponibilidade e segurança, e são projetados para minimizar ao máximo a ocorrência de falhas. Dado que o sistema da Aerocode irá controlar o gerenciamento da produção de aeronaves, ele é considerado crítico.

A consultoria lhe rendeu ótimos conselhos, descritos em um relatório que você se apressou em analisar com esmero. No relatório o primeira sugestão entregue pela empresa de consultoria foi utilizar tecnologias consideras software livre, gratuitos ou de código aberto, mas que fossem robustas, amplamente utilizadas, bem documentadas e seguras. Por exemplo, SGBDs que se encaixem nessa classificação.

Os sistemas gerenciadores de banco de dados (SGBDs) surgiram a partir da necessidade de organizar, armazenar e recuperar grandes volumes de informações de maneira estruturada e eficiente. Antes de sua criação, os dados eram armazenados de forma manual ou em arquivos simples, o que tornava o acesso lento, sujeito a erros e com pouca padronização.

No mercado, alguns dos principais sistemas gerenciadores de banco de dados são o Oracle Database, reconhecido por sua robustez e segurança, muito usado em grandes corporações; o Microsoft SQL Server, que se integra bem com soluções da Microsoft e é bastante utilizado em ambientes corporativos; o MySQL, uma solução de código aberto muito popular para aplicações web; o PostgreSQL, também open source, que se destaca por sua conformidade com o padrão SQL e por oferecer recursos avançados; e o MongoDB, um dos principais representantes dos bancos NoSQL, focado em documentos e bastante aplicado em sistemas modernos de grande escala.

Depois de analisar com cuidado, o SGBD escolhido para armazenar os dados da aplicação web foi o MySQL. O MySQL é um dos sistemas gerenciadores de banco de dados mais populares e amplamente utilizados no mundo, sendo conhecido por sua robustez, flexibilidade e facilidade de uso. Ele é um SGBD relacional, o que significa que organiza as informações em tabelas relacionadas entre si por meio de chaves primárias e estrangeiras, seguindo o modelo relacional proposto por Edgar F. Codd. O MySQL é amplamente empregado em aplicações web, sistemas corporativos, e-commerce e soluções de análise de dados.

O MySQL foi desenvolvido inicialmente por Michael Widenius e David Axmark, em 1995, na empresa sueca MySQL AB. O objetivo era criar um banco de dados de código aberto que fosse rápido, confiável e capaz de suportar aplicações de alta demanda sem os custos elevados associados aos SGBDs proprietários. Desde então, o MySQL passou por diversas evoluções, incorporando novos recursos, otimizações de performance e suporte a transações, o que permitiu que se tornasse uma escolha confiável para pequenas, médias e grandes empresas. Em 2008, a MySQL AB foi adquirida pela Sun Microsystems, que posteriormente foi comprada pela Oracle Corporation, atual mantenedora do MySQL.

Dado que você já conhece o TypeScript e que foi essa linguagem utilizada para o desenvolvimento da primeira versão do sistema, a versão CLI. Você foi orientado pela consultoria a manter o desenvolvimento de toda aplicação nesta plataforma. Assim, para que a comunicação com o MySQL torne-se robusta, escalável e flexível a consultoria sugeriu utilizar um framework de mapeamento objeto-relacional.

O Mapeamento Objeto-Relacional, comumente conhecido pela sigla ORM (Object-Relational Mapping), é uma técnica de desenvolvimento de software que permite a interação entre sistemas orientados a objetos e bancos de dados relacionais. Em termos práticos, o ORM cria uma ponte entre os objetos utilizados no código de uma aplicação

e as tabelas de um banco de dados relacional, facilitando operações de inserção, consulta, atualização e exclusão de dados. Ao utilizar ORM, os desenvolvedores podem trabalhar diretamente com objetos da linguagem de programação, sem precisar escrever instruções SQL complexas para cada operação no banco de dados, tornando o desenvolvimento mais produtivo e reduzindo a ocorrência de erros.

O conceito de Mapeamento Objeto-Relacional surgiu no contexto do crescimento da programação orientada a objetos, na década de 1980, quando se percebeu a dificuldade de integrar sistemas orientados a objetos com bancos de dados tradicionais, que utilizam o modelo relacional. Embora não haja uma única pessoa atribuída à invenção do ORM, o conceito foi formalizado e popularizado através de frameworks e pesquisas na área de engenharia de software, destacando-se projetos pioneiros em linguagens como Smalltalk e Java. Desde então, o ORM tornou-se um padrão de desenvolvimento amplamente adotado em diversas linguagens, permitindo que sistemas complexos possam manipular dados de forma mais intuitiva e consistente com a lógica orientada a objetos.

No contexto do TypeScript, que é uma linguagem moderna voltada para o desenvolvimento de aplicações web e back-end, existem diversos frameworks ORM que permitem integrar objetos da aplicação com bancos de dados relacionais de maneira eficiente. Entre os mais conhecidos está o TypeORM, um dos mais populares para TypeScript, que oferece suporte a múltiplos bancos de dados, incluindo MySQL, PostgreSQL, SQLite e SQL Server. Outro framework relevante é o Prisma, que combina ORM com um gerador de clientes para consultas tipadas, proporcionando segurança e autocompletar durante o desenvolvimento. Também existem opções como Sequelize, que, embora inicialmente desenvolvido para JavaScript, oferece suporte completo a TypeScript e é bastante utilizado em projetos de Node.js.

As vantagens de utilizar ORM com TypeScript incluem a tipagem forte, que ajuda a prevenir erros comuns relacionados a tipos de dados e estruturas do banco de dados, a produtividade aumentada, pois operações de banco de dados podem ser feitas diretamente através de métodos de objetos, e a manutenção facilitada, já que mudanças na estrutura dos dados podem ser refletidas no código de forma consistente. Além disso, frameworks ORM oferecem recursos avançados, como migrations, que permitem versionar alterações no esquema do banco de dados, e relacionamentos automáticos entre entidades, simplificando a modelagem de dados complexos.

Após uma pesquisa sobre vantagens, desvantagens, flexibilidade e escalabilidade você optou por utilizar o framework Prima ORM. O Prisma ORM é um moderno framework de mapeamento objeto-relacional voltado para TypeScript e JavaScript, que se destaca por oferecer tipagem segura, autocompletar inteligente e integração simplificada com diversos bancos de dados relacionais, como PostgreSQL, MySQL, SQLite e SQL Server. Ele funciona gerando um cliente Prisma a partir do esquema de dados definido pelo desenvolvedor, permitindo realizar consultas, inserções, atualizações e exclusões de forma intuitiva, sem a necessidade de escrever SQL manualmente. Além disso, o Prisma oferece suporte a migrations, facilitando a evolução do banco de dados de maneira organizada, e a modelagem de relacionamentos complexos entre tabelas, tornando-o uma ferramenta poderosa para o desenvolvimento de aplicações escaláveis, seguras e de fácil manutenção.

Dado que a GUI utilizou um framework onde pode-se aplicar JavaScript e TypeScript, que o framework ORM é voltado para TypeScript torna-se natural que toda a estrutura da aplicação web seja feita sobre a plataforma Node.js. Assim, todo back-end da aplicação web deve ser implementado para que funcione nesta plataforma. Também, sua aplicação web deve contemplar todos os requisitos que fora disponibilizados desde a versão CLI do sistema.

A empresa de consultoria, que foi contratada por você, também evidenciou algo muito importante em seu relatório. O sistema a ser desenvolvido deve ser classificado como crítico. Desde quando você decidiu se tornar empreendedor e criar uma empresa para construir e vender sistemas para o setor de fabricação de aeronaves já se sabia que há muita preocupação com a qualidade que suas sistemas deveriam ter, afinal, o setor de fabricação de aeronaves é altamente regulamento porque a própria fabricação deste tipo de transporte é considerada crítica – se algo falhar durante um voo as consequências podem ser catastróficas. Por isso, para você não basta apenas desenvolver a aplicação e mostrar sua execução. Também é necessário garantir o mínimo de qualidade.

Além do desenvolvimento da aplicação você decidiu que irá produzir um relatório de qualidade contendo informações e medições que comprovem a qualidade do seu sistema e refutem qualquer tentativa de cancelamento de contrato ou ataques de concorrentes contra a reputação da Aerocode.

No relatório você deve apresentar gráficos com as seguintes métricas: latência, tempo de resposta e tempo de processamento. Esses termos estão relacionados ao modo como o sistema lida com requisições dos usuários e como isso impacta na experiência de navegação. Embora estejam conectados, cada um deles representa um aspecto diferente da comunicação entre cliente e servidor. A Figura 1 mostra uma representação de onde deve-se medir cada uma das três métricas que você decidiu obter.

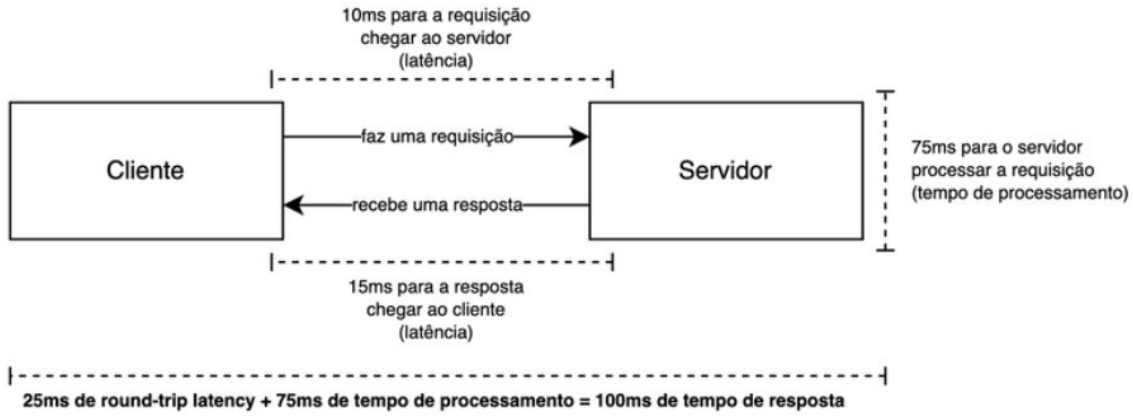

Figura 1. Métricas: latência, tempo de resposta e tempo de processamento.

A latência refere-se ao tempo que um dado leva para percorrer o caminho entre o cliente (por exemplo, o navegador do usuário) e o servidor onde a aplicação está hospedada. Em outras palavras, é o atraso natural que ocorre durante a transmissão da informação na rede. Esse tempo pode variar de acordo com a distância física entre o usuário e o servidor, a qualidade da conexão de internet, a quantidade de roteadores e

intermediários no percurso, entre outros fatores. Quanto maior a latência, mais perceptível será o atraso inicial para que o servidor comece a responder.

Já o tempo de processamento diz respeito ao período em que o servidor efetivamente trabalha para atender à requisição. Quando uma aplicação recebe um pedido — como carregar uma página, buscar informações no banco de dados ou executar uma regra de negócio — o servidor precisa processar instruções, realizar cálculos, acessar recursos e preparar a resposta. Esse tempo depende da eficiência do código, da capacidade dos servidores, da estrutura do banco de dados e até mesmo do volume de requisições simultâneas que estão sendo atendidas.

Por fim, o tempo de resposta é a soma de todo o percurso que envolve a requisição e a entrega da resposta ao cliente. Ele abrange tanto a latência de rede quanto o tempo de processamento do servidor. Em termos práticos, é o tempo total que o usuário percebe entre realizar uma ação (como clicar em um botão ou enviar um formulário) e visualizar o resultado no navegador. Assim, o tempo de resposta é o indicador mais próximo da experiência real do usuário, pois reflete o quanto a aplicação parece rápida ou lenta do ponto de vista de quem a utiliza.

No relatório que você irá produzir deve-se ter um gráfico para cada uma das três métricas e suas medições devem ocorre para conexões ou requisições que serão escaladas da seguinte forma: quando apenas um usuário estiver utilizando o sistema, quando cinco usuários estiverem utilizando o sistema e quando dez usuários estiverem utilizando o sistema. A unidade de medida para medições deve ser, sempre, milissegundo. Também, não esqueça de descrever como você decidiu obter as métricas, qual técnicas utilizou ou ainda como programou sua aplicação para fornecer as informações que estarão nos gráficos.

Por fim, nesta terceira e última etapa, você também deve garantir que a aplicação funcione em servidores das seguintes plataformas: Windows 10 ou superior, Linux Ubuntu 24.04.03 ou superior e distribuições Linux derivadas do Ubuntu.