CENTRO UNIVERSITÁRIO AUGUSTO MOTTA

UNISUAM

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

PEDRO DOS SANTOS BITTENCOURT CYRNE

RESUMO DAS SEMANAS 9 A 12 DE APRENDIZADO

RIO DE JANEIRO

2024

SEMANA 9:

Durante a semana 9 o principal tópico foi a Linguagem de Modelagem Unificada (UML) que nada mais é que uma linguagem padronizada de modelagem de software, onde é amplamente utilizada na engenharia de software para especificar, visualizar, construir e documentar sistemas complexos. A UML foi desenvolvida na década de 90 como uma forma de unificar diversas metodologias de modelagem existentes, resultando em um conjunto consistente e abrangente de notações.

A necessidade de modelagem de software surge da complexidade inerente aos sistemas modernos. Modelar um sistema permite simplificar a complexidade, facilitando a compreensão e análise. A UML oferece diferentes tipos de diagramas para capturar diversos aspectos do sistema. Por exemplo, diagramas de casos de uso ajudam a representar os requisitos funcionais e a interação entre os atores e o sistema, enquanto diagramas de classes detalham a estrutura estática, mostrando classes, atributos, métodos e os relacionamentos entre eles.

A especificação de requisitos funcionais é essencial para garantir que o sistema atenda às necessidades. Casos de uso são uma técnica eficaz para capturar requisitos, descrevendo as interações entre os atores (usuários ou outros sistemas) e o sistema em si. Um caso de uso é composto por elementos como atores, cenários, pré-condições, pós-condições e fluxos de eventos.

Para que um fluxo principal de um cenário de caso de uso seja criado, é necessário identificar a sequência típica de interações entre o ator e o sistema que levam ao objetivo desejado. Esse fluxo principal descreve o caminho mais comum e ideal, enquanto fluxos alternativos e de exceção são projetados para capturar variações e erros que podem ocorrer durante a interação. Esses fluxos adicionais são cruciais para garantir que o sistema seja robusto e capaz de lidar com diferentes situações de maneira eficaz.

Os diagramas de classe são um dos tipos mais importantes de diagramas na UML, representando a estrutura estática de um sistema. Eles mostram as classes, seus atributos e métodos, e os relacionamentos entre elas, como associações, generalizações e dependências. Conceituar diagramas de classe envolve entender como modelar entidades do domínio do problema e suas interações de forma organizada e compreensível.

Construir diagramas de classe requer a identificação das principais classes que compõem o sistema, bem como a definição de seus atributos e métodos. Uma vez construídos, esses diagramas servem como um blueprint para a programação orientada a objetos, facilitando a tradução do modelo para código executável. Aplicar diagramas de classes na programação orientada a objetos ajuda a manter a consistência entre o modelo de design e a implementação real, promovendo a reutilização de código e a manutenção do sistema.

SEMANA 10:

Ainda na semana 10 o principal tópico abordado foi o JavaScript, a qual, é uma linguagem de programação essencial para o desenvolvimento web, conhecida pela sua sintaxe simples e flexível. A sintaxe do JavaScript inclui variáveis, operadores, estruturas de controle (como loops e condicionais) e funções, todas essas ferramentas são necessárias para a construção de lógica complexa nos aplicativos web.

Desenvolver códigos de interação em JavaScript envolve a manipulação do DOM (Document Object Model) para responder a eventos do usuário, como cliques, movimentos do mouse e entrada de dados. Além disso, incluir código JavaScript em páginas HTML é fundamental e pode ser feito de várias maneiras: diretamente dentro de tags <script>, em arquivos externos vinculados via a tag <script src="arquivo.js">, ou até mesmo embutido em atributos de eventos HTML. Essa inclusão permite que as páginas HTML sejam não apenas estáticas, mas também interativas, melhorando significativamente a experiência do usuário.

Objetos nativos em JavaScript são componentes fundamentais fornecidos que facilitam uma ampla variedade de operações. Definir objetos nativos envolve entender que eles são tipos predefinidos que fornecem funcionalidades essenciais, como manipulação de strings, números, arrays, e objetos de data. Esses objetos nativos são intrínsecos à linguagem e não requerem bibliotecas externas para serem usados.

Reconhecer os diferentes tipos de objetos nativos pode ser crucial para escrever códigos eficientes. Por exemplo, o objeto String permite manipulações detalhadas de textos, enquanto o objeto Array facilita o armazenamento e a manipulação de listas de dados. Incluir objetos nativos em códigos JavaScript não apenas simplifica o desenvolvimento, mas também garante que as operações sejam realizadas de maneira otimizada e segura, aproveitando ao máximo as capacidades da linguagem.

O DOM é uma interface de programação que permite que scripts acessem e manipulem a estrutura de documentos HTML e XML. Definir o DOM envolve entender que ele é uma representação do documento, onde cada elemento, atributo e texto é um objeto que pode ser acessado e modificado via JavaScript. A estrutura de árvore permite que os desenvolvedores naveguem pelos nós do documento de forma sistemática e intuitiva.

Descrever a estrutura do DOM implica reconhecer que cada documento é organizado em uma hierarquia de nós, onde o nó raiz é o objeto document, seguido por nós filhos que representam os elementos e conteúdos do documento. Reconhecer as interfaces do DOM, como Core, HTML e CSS, é vital para eficiência. A interface Core fornece métodos e propriedades básicas para navegar e modificar a árvore DOM, a interface HTML é específica para manipulações de elementos HTML, e a interface CSS permite a alteração de estilos diretamente via JavaScript. Juntas, essas interfaces permitem uma interação dinâmica e poderosa com os documentos web.

SEMANA 11:

Na semana 11, o javascript ainda foi um tópico abordado em grande parte, sendo no início dela falando sobre F# a qual é uma linguagem de programação funcional, mas também suporta outros paradigmas, como o imperativo e o orientado a objetos, o que a torna uma linguagem multiparadigma. Os principais conceitos incluem imutabilidade, funções de primeira classe e tipos discriminados. A linguagem é especialmente adequada para tarefas de análise de dados e processamento de algoritmos complexos devido à sua capacidade de expressar operações de maneira concisa e eficiente.

Os fundamentos da linguagem JavaScript incluem a manipulação do DOM, eventos, além de sua sintaxe baseada em protótipos e seu modelo assíncrono baseado em eventos. Python, por outro lado, é uma linguagem de programação de alto nível conhecida por sua simplicidade, frequentemente usada em desenvolvimento web, automação, análise de dados e aprendizado de máquina.

O BOM refere-se à estrutura de objetos que o navegador disponibiliza para a manipulação de elementos fora do escopo do DOM. Ele inclui objetos como window, navigator, screen, location, e history, permitindo aos desenvolvedores interagir com o navegador em si, além do conteúdo da página. Reconhecer os principais objetos do BOM é essencial para tarefas como redirecionamento de páginas, manipulação do histórico do navegador, e obtenção de informações sobre a tela do dispositivo.

Implementar cookies e temporizadores em JavaScript envolve o uso de funcionalidades específicas do BOM. Cookies são pequenos pedaços de dados armazenados pelo navegador para manter informações persistentes entre visitas à página. Eles são úteis para autenticação de usuário e preferências de configuração. Temporizadores, implementados com setTimeout e setInterval, permitem a execução de código após um intervalo de tempo especificado, ou repetidamente, respectivamente. Estes são fundamentais para a criação de funcionalidades como slideshows automáticos, verificação de tempo ocioso do usuário, e atualizações periódicas de conteúdo.

JQuery é uma biblioteca JavaScript rápida e concisa que simplifica a manipulação do DOM, tratamento de eventos e animações. Ela permite que seja escrito menos código para realizar tarefas comuns de JavaScript, aumentando a produtividade e a eficiência.

SEMANA 12:

Durante a semana 12, o primeiro tópico foi implementar funcionalidades utilizando a biblioteca jQuery, permitindo manipular o DOM de maneira simplificada. O jQuery facilita a seleção de elementos, a adição de efeitos e o tratamento de eventos, eliminando muitas das complexidades inerentes ao JavaScript puro. Utilizar jQuery pode reduzir significativamente a quantidade de código necessário para realizar operações comuns, acelerando o desenvolvimento e tornando o código mais legível e fácil de manter.

Os ganhos obtidos a partir do uso da biblioteca jQuery incluem maior produtividade e consistência no desenvolvimento web. A biblioteca abstrai as diferenças entre os navegadores, permitindo que os desenvolvedores escrevam código que funcione de forma uniforme em diferentes ambientes. Analisar a aplicação do jQuery em cenários reais demonstra como ele pode ser utilizado para criar interfaces de usuário interativas e dinâmicas, melhorando a experiência do usuário e tornando as aplicações web mais robustas e responsivas.

Requisições assíncronas são essenciais para criação de aplicações web modernas que interajam com servidores de forma eficiente, sem a necessidade de recarregar a página. A função .ajax() da biblioteca jQuery facilita a realização dessas requisições, permitindo o envio e recebimento de dados entre o cliente e o servidor. Com a utilização de métodos como .get() e .post(), é possível buscar e enviar informações, respectivamente, de forma assíncrona, melhorando a interatividade e a performance da aplicação.

Desenvolver uma aplicação utilizando .get() e .post(), por exemplo, envolve criar funções que façam requisições ao servidor para obter dados ou enviar formulários, por exemplo. Essas operações podem ser integradas diretamente com métodos de controladores no backend, permitindo uma comunicação fluida entre o front end e o back end. Ao reconhecer a importância de requisições assíncronas, os desenvolvedores podem criar

Citações:

Referências: