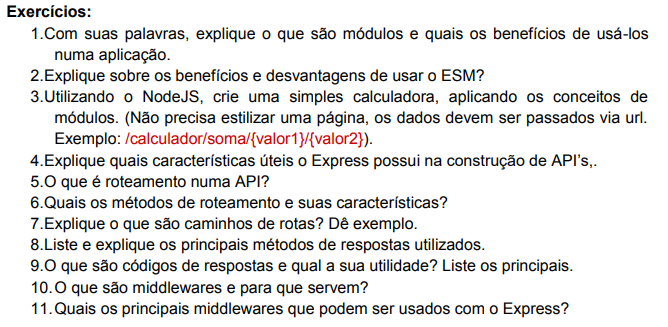
**ATIVIDADE 08 – MÓDULO 09**



Q5)

Roteamento é como uma API conecta uma URI (Uniform Resource Identifier) a uma ação. Ele determina como um aplicativo responde a uma solicitação do cliente por um endpoint específico, que é uma URI (ou caminho) e um método de solicitação HTTP específico (GET, POST, e assim por diante). Cada rota pode ter uma ou mais funções de manipulação, que são executadas quando a rota é correspondida.

Q6) Uma Web API suporta dois tipos de roteamento:

Roteamento baseado em convenção - usa modelos de rota para determinar qual controlador e método de ação executar. Pelo menos um modelo de rota deve ser adicionado à tabela de rotas para lidar com várias solicitações HTTP.

Roteamento de atributos - Como o nome indica, o roteamento de atributo usa o atributo [Route()] para definir rotas. O atributo Route pode ser aplicado em qualquer controlador ou método de ação.

Q7) O roteamento refere-se a como os endpoints (URIs) de um aplicativo respondem às solicitações do cliente. Os caminhos de rota (route paths), em combinação com um método de solicitação, definem os pontos de extremidade (endpoints) nos quais as solicitações podem ser feitas. Os caminhos de rota podem ser strings, padrões de string ou expressões regulares. Por exemplo, “/carList” poderia representar o caminho de rota para um cliente conseguir acessar a listas de carros disponíveis em um site de venda de carro. Outro exemplo, o usuário poderia usar o verbo DELETE junto com o caminho “/carList/3” para apagar o carro que possui identificador 3. Então o caminho de rota seria esse citado anteriormente. Um mesmo caminho de rota pode servir a um ou mais verbos do HTTP.

Q8)

Uma resposta HTTP é feita por um servidor para um cliente. O objetivo da resposta é fornecer ao cliente o recurso que ele solicitou, ou informar ao cliente que a ação solicitada foi realizada; ou então informar ao cliente que ocorreu um erro no processamento de sua solicitação.

Uma resposta HTTP contém:

* Uma linha de estado.
* Uma série de cabeçalhos HTTP ou campos de cabeçalho.
* Um corpo de mensagem, que geralmente é necessário.

A **linha de status** é a primeira linha na mensagem de resposta. É composto por três itens:

* O número da versão HTTP, mostrando a especificação HTTP com a qual o servidor tentou fazer com que a mensagem fosse compatível.
* Um código de status (status code), que é um número de três dígitos que indica o resultado da solicitação.
* Uma frase de razão, também conhecida como texto de status, que é um texto legível por humanos que resume o significado do código de status.

Um exemplo de response line é: HTTP/1.1 200 OK

Q9)

Os códigos de status das respostas HTTP indicam se uma requisição HTTP foi corretamente concluída. As respostas são agrupadas em cinco classes:

* Respostas de informação (100-199),
* Respostas de sucesso (200-299),
* Redirecionamentos (300-399)
* Erros do cliente (400-499)
* Erros do servidor (500-599).

Esses status são definidos pela seção 10 da RFC 2616

Comentando sobre algumas Respostas de informação:

100 – Continue - Essa resposta provisória indica que tudo ocorreu bem até agora e que o cliente deve continuar com a requisição ou ignorar se já concluiu o que gostaria.

101 – Switching Protocol - Esse código é enviado em resposta a um cabeçalho de solicitação Upgrade pelo cliente, e indica o protocolo a que o servidor está alternando.

102 – Processing - Este código indica que o servidor recebeu e está processando a requisição, mas nenhuma resposta está disponível ainda.

Comentando sobre algumas Respostas de Sucesso:

GET: O recurso foi buscado e transmitido no corpo da mensagem.

HEAD: Os cabeçalhos da entidade estão no corpo da mensagem.

PUT ou POST: O recurso descrevendo o resultado da ação é transmitido no corpo da mensagem.

TRACE: O corpo da mensagem contém a mensagem de requisição recebida pelo servidor.

200 – OK - Estas requisição foi bem sucedida. O significado do sucesso varia de acordo com o método HTTP:

201 – Created - A requisição foi bem sucedida e um novo recurso foi criado como resultado. Esta é uma tipica resposta enviada após uma requisição POST.

202 – Accepted - A requisição foi recebida mas nenhuma ação foi tomada sobre ela. Isto é uma requisição não-comprometedora, o que significa que não há nenhuma maneira no HTTP para enviar uma resposta assíncrona indicando o resultado do processamento da solicitação. Isto é indicado para casos onde outro processo ou servidor lida com a requisição, ou para processamento em lote.

203 – Non-Authoritative Information - Esse código de resposta significa que o conjunto de meta-informações retornadas não é o conjunto exato disponível no servidor de origem, mas coletado de uma cópia local ou de terceiros. Exceto essa condição, a resposta de 200 OK deve ser preferida em vez dessa resposta.

204 – No Content - Não há conteúdo para enviar para esta solicitação, mas os cabeçalhos podem ser úteis. O user-agent pode atualizar seus cabeçalhos em cache para este recurso com os novos.

Mensagens de redirecionamento:

300 – Multiple Choice - A requisição tem mais de uma resposta possível. User-agent ou o user deve escolher uma delas. Não há maneira padrão para escolher uma das respostas.

301 – Moved Permanently - Esse código de resposta significa que a URI do recurso requerido mudou. Provavelmente, a nova URI será especificada na resposta.

Resposta de erro do cliente:

400 – Bad Request - Essa resposta significa que o servidor não entendeu a requisição pois está com uma sintaxe inválida.

401 – Unauthorized - Embora o padrão HTTP especifique "unauthorized", semanticamente, essa resposta significa "unauthenticated". Ou seja, o cliente deve se autenticar para obter a resposta solicitada.

402 – Pa yment Required Experimental - Este código de resposta está reservado para uso futuro. O objetivo inicial da criação deste código era usá-lo para sistemas digitais de pagamento porém ele não está sendo usado atualmente.

403 – Forbidden - O cliente não tem direitos de acesso ao conteúdo portanto o servidor está rejeitando dar a resposta. Diferente do código 401, aqui a identidade do cliente é conhecida.

404 – Not Found - O servidor não pode encontrar o recurso solicitado. Este código de resposta talvez seja o mais famoso devido à frequência com que acontece na web.

Respostas de erro do servidor:

500 – Internal Server Error - O servidor encontrou uma situação com a qual não sabe lidar.

501 – Not Implemented - O método da requisição não é suportado pelo servidor e não pode ser manipulado. Os únicos métodos exigidos que servidores suportem (e portanto não devem retornar este código) são GET e HEAD.

502 – Bad Gateway - Esta resposta de erro significa que o servidor, ao trabalhar como um gateway a fim de obter uma resposta necessária para manipular a requisição, obteve uma resposta inválida.

Q10)

O middleware são serviços de software e nuvem que oferecem serviços comuns e recursos a aplicações, além de ajudar os desenvolvedores e operadores a criar e implantar aplicações com mais eficiência. O middleware tem o papel de conectar aplicações, dados e usuários. O middleware pode incluir ambientes de execução da aplicação, integração de aplicações empresarial e vários tipos de serviços em nuvem. Gerenciamento de dados, serviços de aplicações, sistema de mensageria, autenticação e gerenciamento de interface de programação de aplicações (API) são recursos comumente operados por um software de middleware. Hoje, o middleware é a base tecnológica de arquiteturas nativas em nuvem modernas. Para as organizações com ambientes multicloud e em containers, o uso de middleware torna o desenvolvimento e a execução de aplicações em escala algo econômico.

Q11) Middleware no Express são funções que entram em ação depois que o servidor recebe a requisição e antes que a resposta seja enviada ao cliente. Eles estão dispostos em uma cadeia e são chamados em sequência.

Podemos usar funções de middleware para diferentes tipos de tarefas de processamento necessárias para atender à solicitação, como consulta ao banco de dados, fazer chamadas de API, preparar a resposta etc. e, finalmente, chamar a próxima função de middleware na cadeia.

As funções de middleware recebem três argumentos: o objeto de solicitação (request), o objeto de resposta (response) e, opcionalmente, a função de middleware next().

As funções de middleware no Express são dos seguintes tipos:

* Middleware de nível de aplicativo que é executado para todas as rotas em um objeto de aplicativo;
* Middleware de nível de roteador que é executado para todas as rotas em um objeto de roteador;
* Middleware integrado fornecido pelo Express como express.static, express.json, express.urlencoded;
* Middleware de tratamento de erros para lidar com erros;
* Middleware de terceiros mantido pela comunidade.

-----

Falando sobre Vantagens de usar middleware:

* O middleware pode processar objetos de solicitação várias vezes antes que o servidor trabalhe para essa solicitação.
* Middleware pode ser usado para adicionar funcionalidade de log e autenticação.
* O middleware melhora o desempenho de renderização do lado do cliente.
* Middleware é usado para definir alguns cabeçalhos HTTP específicos.
* Middleware ajuda na otimização e melhor desempenho.

Gostei dessas referências:

<https://reflectoring.io/getting-started-with-express/>

<https://www.geeksforgeeks.org/middleware-in-express-js/>