**Olá,**

**Para darmos continuidade no processo seletivo, pedimos a gentileza que responda o teste abaixo.**

**1) Considerando a complexidade das operações logísticas, qual seria a arquitetura mais adequada para um sistema de microsserviços voltado para a gestão logística, levando em conta requisitos como escalabilidade, resiliência e integração de dados em tempo real?**

R. A arquitetura de microsserviços baseada em eventos tende a ser a mais indicada. Em conjunto com algum serviço de mensageria como Kafka.

Obs: Pelas características desse tipo de arquitetura é interessante implementar um sistema de monitoramento como grafana e(ou) Prometheus para a coleta de dados sobre o funcionamento da aplicação. Também é muito recomendado implementar um serviço de gerenciamento de Logs para localizar facilmente erros e ter um status mais detalhado do funcionamento da aplicação.

**2) Desenvolva uma api RESTful em PHP para criar, atualizar, deletar e listar todos os usuários. As informações devem ser salvas em um banco de dados MySQL.**

**O endpoint deve retornar os dados em formato JSON e permitir operações GET, POST, PUT e DELETE para manipular os registros de usuário.**

**Considere aspectos como segurança, validação de entrada e tratamento de erros. O exame deverá ser entregue através do link do projeto no Git.**

**Desejável que utilize Laravél ou CodeIgniter 3.**

R. Link para o repositório:

<https://github.com/skymarkos7/restful_users/tree/main>

**3) Suponha que você precise integrar uma API REST externa em um projeto PHP. A API fornece informações sobre o clima de uma determinada cidade.**

**Como você abordaria essa integração? Descreva os passos necessários para fazer uma solicitação à API, receber os dados de resposta e armazená-los em um banco de dados MySQL local para uso posterior.**

**Considere questões como autenticação, tratamento de respostas e cronograma de atualização dos dados.**

R. Primeiro é importante verificar se a api possui autenticação e se possui qual o método utilizado, para esses passo imaginarei que a api solicita um jwt para a requisição.

Utilizando o cURL para realizar as requisições eu faria da seguinte forma:

* Criaria uma variável para conter o JWT obtido por exemplo em uma etapa anterior de login.
* $token
* Criaria uma variável para conter possíveis parâmetros da requisição.
* $city
* Criaria uma variável com a url montada
* $url = “https:/[temponaminhacicade.com/weather?city=$city](http://temponaminhacicade.com/weather?city=$city)“
* Após isso já podemos iniciar a sessão cURL
* $ch = curl\_init();
* Como nessa API tem autenticação JWT nas requisições, precisarei setar opções adicionais nessa requisição.

curl\_setopt\_array($ch, [

CURLOPT\_URL => $url,

CURLOPT\_RETURNTRANSFER => true,

CURLOPT\_HTTPHEADER => [

'Authorization: Bearer ' . $jwtToken,

'Content-Type: application/json',

],

]);

* Pronto já é possível fazer a requisição.
* $response = curl\_exec($ch);
* É uma boa prática verificar se ocorreu algum erro durante a requisição.

if ($response === false) {

die('Erro ao fazer solicitação à API: ' . curl\_error($ch));

}

* Antes de trabalhar com a resposta da requisição é necessário converter o formato da resposta.
* $data = json\_decode($response, true);
* Eu também verificaria se ocorreu algum erro durante essa conversão.

if ($data === null) {

die('Erro ao decodificar resposta da API');

}

* Também pode ocorrer de vir uma resposta de erro como por EX: “ code:404 Essa cidade não existe”. Nesse caso também é interessante colocar uma verificação se a resposta possui qualquer código diferente de 200. (Também poderia verificar qualquer código no intervalo de 200).

if (isset($data['cod']) && $data['cod'] !== 200) {

die('Erro da API: ' . $data['message']);

}

* Agora é só extrair o dado que interessa.
* $temperatura = $data['main']['temp'];
* E inserir esse dado no banco de dados seja com eloquent do laravel ou com php vanila, usando PDO.

$pdo = new PDO('mysql:host=meu\_host;dbname=meu\_banco\_de\_dados', 'meu\_usuario', 'minha\_senha');

$stmt = $pdo->prepare("INSERT INTO dados\_climaticos (cidade, temperatura, umidade, data\_hora) VALUES (:cidade, :temperatura,, NOW())");

$stmt->execute([

'cidade' => $city,

'temperatura' => $temperatura]);

* Por fim é só fechar a sessão cURL.
* curl\_close($ch);
* Para a questão do cronograma de atualização de dados é possível pensar em uma lógica que execute todo o código periodicamente como.

function consultarClima() {

echo "Todo o bloco de consulta aqui";

}

// Defina o intervalo de tempo em segundos (3600 segundos = 1 hora)

$intervalo = 3600;

// Loop infinito para executar a consulta a cada intervalo de tempo

while (true) {

// Executa a função de consulta

consultarClima();

// Aguarda o intervalo de tempo antes da próxima execução

sleep($intervalo);

}

* Entretanto o passo 14 não é a melhor alternativa para esse tipo de tarefa que executa periodicamente uma ação assíncrona, é melhor utilizar um CRON JOB.
* Crie em seu terminal ou em seu host de hospedagem um CRON JOB que executará sua consulta periodicamente no tempo determinado.

**4) Quais são as melhores práticas para garantir um desenvolvimento de software seguro em PHP?**

R. Existem algumas práticas que costumo adotar e que colaboram para o desenvolvimento seguro e que ao mesmo tempo aumentam a confiabilidade no código como:

Testes de unidade, testes de integração e testes de feature com o PHPUnit. Legal notar que frameworks como laravel 11 já traz essa ferramenta integrada :)

Para entradas de dados vindos do usuário é legal sanitizar para prevenir ataques de injeção de código.

Preferir o uso de funções de hash seguras como bcrypt ao invés de hash’s mais vulneráveis como MD5. É legal notar que o framework laravel já traz o bcrypt integrado.

Procurar manter as bibliotecas do projeto, o php e o framework atualizados.

Em trabalho em equipe pode ser interessante implementar no processo de aprovação de pull request um check-list de auditoria de código, onde um desenvolvedor testa o código do outro.

Também acho importante ter nivelação de conhecimento de boas práticas periódicas entre a equipe. Onde pode ser compartilhado as melhores práticas para aplicar ao projeto (ao mesmo tempo pode ser definido essa boa prática como padrão obrigatório a ser seguido, e inserido com etapa no check-list para aceite no pull request).