Expliquem, com suas palavras, o que é uma API no contexto web e como PHP a consome ou expõe;

APIs no contexto de web especificamente, são ferramentas que fazem uma ponte de comunicação entre página e página ou entre página e servidor web, com essa ponte de comunicação eles podem expor dados do navegador e do ambiente ao redor do computador, além de claro conseguir trabalhar com esses dados
e o php consegue consumir ou expor a API, utilizando as funções de POST e GET no qual ele faz o contato com o usuário para coletar essas informações e guardar em um pacote que é enviado a API que dependendo da query faz o que precisa com esses dados.

---

diferencie biblioteca de framework em termos de inversão de controle e escopo;

a principal diferença é a inversão de controle e no escopo: uma biblioteca é um conjunto de funções ou classes que você chama quando quiser, ou seja, você controla o fluxo do código (sem inversão de controle); já um framework define a estrutura geral da aplicação e chama seu código em momentos específicos, ou seja, ele controla o fluxo (há inversão de controle). Além disso, bibliotecas costumam ser menores e focadas em tarefas específicas (como manipular PDFs ou enviar e-mails), enquanto frameworks, como Laravel ou Symfony, oferecem uma base completa para desenvolver aplicações inteiras

---

descreva o papel do Composer e do Packagist na gestão de dependências e versões (SEMVER);

Composer: ele gerencia as bibliotecas que o seu projeto depende, cuidando para as versões serem compatíveis, e faz um auto carregamento das classes das bibliotecas para que você não precise usar o include.
Packagist: ele é onde está a maioria das bibliotecas necessárias, as pessoas podem publicarem as suas bibliotecas lá.

---

exemplifique três bibliotecas populares (por exemplo: Guzzle para HTTP, PHPMailer para e-mail, Monolog para logging) dizendo quando fazem sentido e quando não;

1- O PHPMailer é uma biblioteca que vai possibilitar a simplificação dos envios de emails com php, usando SMTP que é um padrão técnico para enviar e receber emails pela internet, faz sentido usar quando você precisa mexer com emails reais, com anexos e etc, quando quer evitar problemas de segurança e quando vai integrar o sistema com outros serviços de email externos tipo gmail, outlook e etc, não faz sentido usar se você não envia emails ou só vai simular envios

2- Guzzle é um cliente http para fazer requisições tipo o GET e o POST para APIS faz sentido usar quando você precisa consumir APIS externas tipo buscar dados do clima, pagamentos ou redes sociais, quando você precisa de maior controle das requisições e nao faz sentido se você só vai fazer uma chamada HTTP simples e quer evitar dependências externas ou se o ambiente é muito limitado onde não daria nem pra instalar o composer por exemplo

3- O monolog é uma biblioteca para registrar logs (erros, avisos, ações do sistema) em arquivos, banco de dados ou serviços externos, faz sentido usar em aplicações médias ou grandes, onde vai ser importante monitorar erros e eventos e não faz sentido usar em scripts pequenos ou projetos de aprendizado, onde um simples error_log() já vai resolver ou se o servidor ou ambiente não precisa armazenar logs complexos.

---

compare dois frameworks (por exemplo: Laravel e Slim) destacando curva de aprendizado, organização, performance típica, comunidade e ecossistema;

Framework	Laravel	Symfony
Curva de Aprendizado	Mais suave	Mais desafiador
Organização	Convenção sobre configuração	Configuração sobre convenção
Performance Típica	Performance boa para maioria das aplicações	Performance ligeiramente superior em alta escala

Comunidade	Comunidade muito ativa e acolhedora	Comunidade técnica sólida
Ecossistema	Ecossistema rico e integrado	Componentes modulares de alta qualidades

---

mapeie riscos comuns ao depender de terceiros (licenças, descontinuação, vulnerabilidades, acoplamento) e estratégias de mitigação (pin de versão, changelog, substituibilidade, auditoria de dependências);

Licenças: risco de incompatibilidade ou mudança de licença.
Mitigação: auditoria de licenças, uso de licenças permissivas, version pinning.

Descontinuação: projeto ou API pode ser encerrado.
Mitigação: monitorar atualizações, abstrair dependências, ter alternativas mapeadas.

Vulnerabilidades: falhas de segurança em bibliotecas externas.
Mitigação: auditoria de dependências, atualização regular, uso de ferramentas de segurança (ex: Dependabot, Snyk).

Acoplamento: dificuldade de trocar tecnologia ou fornecedor.
Mitigação: aplicar baixo acoplamento, criar camadas de abstração e registrar changelogs.

---

relacione requisitos não funcionais a escolhas tecnológicas — por exemplo, quando escolher um microframework em vez de um full-stack;

A escolha entre um microframework (como Flask, Sinatra ou Lumen) e um framework full-stack (como Django, Rails, Spring ou Laravel) é uma decisão técnica crucial que deve ser guiada por requisitos não funcionais específicos do projeto. Estes requisitos determinam as qualidades do sistema, como desempenho, escalabilidade e manutenibilidade.

---

por fim, descreva um cenário fictício simples (ex.: “cadastro de pedidos com envio de e-mail e consumo de CEP”) e indique:
a) API(s) que seriam usadas,
b) bibliotecas candidatas
c) um framework apropriado,
justificando cada escolha em uma frase.

Um sistema de cadastro de pedidos online que, ao registrar uma nova compra, envia um e-mail de confirmação ao cliente e consulta o CEP para preencher automaticamente o endereço.

A) **ViaCEP API** → para buscar dados de endereço a partir do CEP informado.
SendGrid API (ou similar, como AWS SES) → para envio automático de e-mails de confirmação ao cliente.

B) **Requests** → para consumir as APIs externas (ViaCEP e SendGrid).
Pydantic → para validação dos dados do pedido e do cliente.
Smtplib (alternativa nativa ao SendGrid) → se for preferido envio de e-mails via SMTP direto.

C) **FastAPI** → porque oferece alta performance, integração fácil com Pydantic e Requests, e suporte nativo a APIs RESTful de forma simples e moderna.

Fontes:
https://aws.amazon.com/pt/what-is/api/
https://developer.mozilla.org/pt-BR/docs/Learn_web_development/Extensions/Client-side_APIs/Introduction
https://www.bairesdev.com/blog/phplibrary/