API

Uma API, ou Interface de Programação de Aplicações, é um conjunto de regras e definições que permite que diferentes softwares se comuniquem entre si. Ela funciona como uma ponte, permitindo que um aplicativo acesse funcionalidades ou dados de outro, sem precisar entender o código interno desse segundo aplicativo.
Exemplo simples

- Imagine que você tem um aplicativo ou site de previsão do tempo.
- Em vez de desenvolver seu próprio sistema para obter informações meteorológicas.
- Uma API de um provedor de serviços meteorológicos fornece seus dados.

O que é uma API REST?

As APIs REST (Representational State Transfer) são um estilo arquitetural que define um conjunto de restrições para criar serviços web. Elas utilizam os métodos HTTP padrão (GET, POST, PUT, DELETE) para realizar operações em recursos, proporcionando uma comunicação simples, escalável e padronizada.

Características de uma API REST:

Stateless:

Cada requisição do cliente para o servidor contém todas as informações necessárias para compreender e processar a requisição. O servidor não armazena nenhum estado sobre o cliente entre as requisições. Em resumo, o "estado" seria qualquer dado que o servidor normalmente armazenaria para "lembrar-se" de uma interação anterior com o cliente. No contexto de um servidor **stateless**, esse dado não é mantido entre as requisições, o que aumenta a escalabilidade do sistema.

Recursos:

Os dados ou serviços oferecidos pela API são representados como recursos, que podem ser acessados ou manipulados pelos métodos HTTP.

HTTP:

O HTTP (**Hypertext Transfer Protocol**) é um protocolo de comunicação que permite a transferência de informações na internet. Ele é a base para a comunicação entre navegadores e servidores Web, sendo utilizado para a transmissão de conteúdo como textos, imagens e vídeos.

GET: busca dados, **POST**: envia dados, **PUT**: atualiza dados, **DELETE**: remove dados.

URL x URI x URN:

URL – Uniform Resource Locator | Localizador de Recursos Universal
- Exemplo de URL
  - fatecpg.edu.br

URN – Uniform Resource Name | Nome de Recursos Universal
- Exemplo de URN
  - home.html

URI – Uniform Resource Identifier | Identificador de Recursos Universal
- Exemplos de URI
  - https://fatecpg.edu.br/cursos/dsm

O que é Serialização de Objetos?

Os computadores utilizam os arquivos como estruturas de dados para armazenar (ou persistir) na memória secundária de um sistema, usualmente unidades de disco rígido, grandes volumes de dados.

O mecanismo de serialização de objetos permite converter a representação interna de um objeto para uma sequência de bytes. Uma vez serializado, um objeto pode ser salvo em arquivo e recuperado a partir do arquivo e desserializado para recriar o objeto na memória.

O processo de serialização de objetos com o propósito de persisti-los em arquivos de dados passa por três etapas:

1) estruturar as informações do objeto em uma classe que implementa a interface Serializable;

2) persistir os objetos serializados em um arquivo binário;

3) recuperar os objetos serializados do arquivo binário.

https://www.devmedia.com.br/serializacao-de-objetos-em-java/23413

Diferenças Entre Interfaces e Classes:

Classes são estruturas que combinam dados (variáveis) e comportamentos (métodos), incluindo sua implementação, e podem ser instanciadas para criar objetos. Também suportam herança.

Interfaces são contratos que definem apenas as assinaturas dos métodos, sem implementação, e não contêm dados. Elas garantem que as classes que as implementam sigam o contrato, mas não podem ser instanciadas diretamente.

A **assinatura de um método** refere-se ao seu nome, tipos de parâmetros e tipo de retorno, sem incluir a implementação (o código que define o que o método faz).

https://pt.stackoverflow.com/questions/136404/diferen%C3%A7a-entre-classe-e-interface

Diferenças Entre Classes e Records:

Os records são estruturas fixas onde o foco é o armazenamento de dados para sua manipulação. As classes são mais dinâmicas onde o foco é o comportamento.

https://www.treinaweb.com.br/blog/conheca-o-recurso-de-records-no-java

Gerenciadores de dependências:

Maven e Gradle são ferramentas de automação de

compilação e gerenciamento de dependências.

Diferenças:

- Maven usa arquivos XML para gerenciar as dependências e descreve a lógica de construção usando plugins.
- Gradle usa um formato de script e descreve a lógica de construção como código.

Bibliotecas:

Matcher e Pattern:

Regex, ou expressões regulares, é uma sequência de caracteres que define um padrão de busca. Elas são muito úteis para realizar buscas e manipulações de strings de forma mais flexível e poderosa.

No Java, as classes Matcher e Pattern são utilizadas para trabalhar com expressões regulares. A classe Pattern é responsável por compilar o padrão de busca em um objeto Pattern, enquanto a classe Matcher é utilizada para aplicar esse padrão em uma string e realizar as operações desejadas.

https://cursos.alura.com.br/forum/topico-duvida-regex-matcher-e-pattern-268271

FileReader e FileWriter:

FileReader é usado para ler dados de um arquivo de texto, um caractere por vez. Para criar um objeto FileReader, você deve passar o caminho para o arquivo que deseja ler como argumento do construtor.

FileWriter é usado para gravar dados em um arquivo de texto, um caractere por vez. Ele pode ser usado para escrever qualquer tipo de arquivo de texto, incluindo documentos de texto, arquivos HTML e arquivos XML. Para criar um objeto FileWriter, você deve passar o caminho para o arquivo no qual deseja gravar como argumento do construtor.
https://codespindle.com/Java/Java_fileReader_fileWriter.html

https://www.devmedia.com.br/classes-de-entrada-e-saida-de-dados-em-java/26029