3 / 2 / 22
D S T Q Q S S

API → Aplication Programing Interface

• Conjunto de rotinas e padrões estabelecidos por um software para a utilização das suas funcionalidades por aplicativos que não pretendem envolver-se em detalhes da implementação do software.

• Exemplos:

• Api do banco central que eu consulto e ela apenas me devolve os dados solicitados, sem que eu precise me preocupar com a regra de negócio e a pesquisa no banco deles.

• Apis de disparo de corrida feitos pelo Mob, eu não precisei me preocupar com o back-end de disparar a corrida, apenas disparei e peguei o id.

• Apis são usadas também para dissassociar back-end de front-end, pois como foi convencionado que apis sempre devem retornar Json, isso possibilita que o mesmo back-end sirva aplicações de diversas plataformas (web, mobile etc), ficando a cargo do dev front end exibir esses dados da maneira como ele quiser.

Continua

• Normalização dos dados:

Como explicado no último exemplo, convencionou-se que as apis deveriam retornar por padrão a response em json, para que aplicações de diversas tecnologias possam interagir com a mesma api. Todas as apis funcionam com HTTP

Api que aplica Rest

• Rest (Representational state transfer) e Restful

Conjunto de regras, convenções e boas práticas para criação de Apis. Ao todo são 6 constraints:

- Client-Server: O cliente e o servidor precisam estar separados
- Stateless: Cada requisição feita para o servidor deve conter todas as informações necessárias para o servidor responder a request, sem que esses dados da requisição sejam gravados em algum lugar, ficando apenas na Request.
- Cacheable: As respostas para uma requisição deverão ser explícitas ao dizer se aquela requisição pode ou não ser cacheada pelo client
- Layered System: O cliente acessa um endpoint sem precisar saber da complexidade, passos ou camadas do servidor pelo qual a request está passando, para que a request seja respondida
- Code on demand (Opcional): Dá a possibilidade da nossa aplicação pegar códigos, como Js e executar no client

3/2/22

D S T Q Q S S

- Boas práticas
  - Utilizar verbos HTTP para nossas requisições
  - Não deixar "/" no final do endpoint
  - Sempre devolver um HTTP status code

- Verbos HTTP
  - Get: Recebe dados de um resource
  - Post: Envia dados a serem processados em um res.
  - Put: Atualiza dados de um resource.
  - Delete: Deleta um resource.