Programação Orientada a Serviços

Prof. Diego Cirilo

Aula 15: Serviços REST

Desenvolvendo Serviços

- Client/Server ou sistemas monolíticos?
- APIs/Serviços
- Por que criar APIs?
- SaaS - Software as a service
- Múltiplos clientes: web/mobile/terceiros.

RESTful APIs

- Uma API deve ser útil para programadores.
- O uso da API deve ser padronizado.
- Padrão REST/JSON.
- Documentação!

Boas práticas em RESTful APIs

- Endpoints claros: recursos.
- Métodos HTTP: ações.
- Serialização: JSON.
- Códigos de status HTTP: erros.
- Permita filtragem/paginação/ordenação.
- Backward Compatibility/Compatibilidade reversa.
- Versionamento

Django REST Framework

- Django vs. Flask?
- Django Rest Framework
- Suporte "nativo" a página de documentação, ORM, autenticação, etc.
- Pode ser adicionado a um projeto já existente.
- *Serializers*

Django REST Framework

- Instale o `djangorestframework` com o `pip`
- Adicione o `rest_framework` no `INSTALLED_APPS` do arquivo `settings.py`.
- Adicione os *serializers* em `serializers.py`.
- Adeque as *views*.
- Só?

Exemplo

Projeto Final

- Crie uma API JSON Placeholder da *Shopee*.
- Sua API deve permitir o CRUD de User, ToDo, Posts e Comments, seguindo o esquema da API original.
- Não é necessário Autenticação, e o modelo User pode ser simplificado.
- Crie também um cliente web (com Flask) que acesse sua API. O cliente deve ter front-end com estilos.
- O projeto é em dupla(no máximo!), e terá uma nota para o Cliente e outra para o Serviço.

Referências

- https://yalantis.com/blog/how-to-create-a-restful-api/
- https://stackoverflow.blog/2020/03/02/best-practices-for-rest-api-design/

Dúvidas?