
SESSÃO
DA
TARDE
2019.1
Um CRUD
FullStack
12 a 23 de agosto
http://bit.ly/sdt-2019
Audio 1 (Bloco D)
INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
BAHIA
Campus Salvador
ADSIFBA


12.08
Docker
Sandro Andrade
14.08
GIT
Ícaro Jerry
16.08
Front-end
Hugo Deiró
19.08
Python
Marcos Sobral
21.08
Django
Manoel Neto
23.08
Android
Renato Novais
SESSÃO DA TARDE
http://bit.ly/sdt-2019
Audio 1 (Bloco D)
2019.1
ADSIFBA
Foto: unsplash.cc

DIA 1

Projeto, Ambiente, Docker

O Projeto

- Sistema de Controle de Livros - SCL (codigo, ISBN, titulo, autor, ano, editora)
  - Figuras gerais mostrando a solução?
- O que é full stack?
- O que são as tecnologias?
  - SCL em cada uma das tecnologias
  - Rest, json,
- Banco de dados, json, rest, postman, curl
- Testando as tecnologias
- Git/Git Hub
  - Git init, clone, branch, commit, push, pull, merge, fork, como colocar um projeto dele no github
- Clonar os repositório e rodar.

Cronograma

Dia	Atividade	Tutor
Dia 1	Projeto, Ambiente, Docker	Sandro Andrade
Dia 2	Git/GitHub	Icaro Jerry
Dia 3	Html, CSS, JavaScript, BootStrap, Templates	Hugo Deiró
Dia 4	Python + Django	Marcos Sobral
Dia 5	Django (models, views, serviços rest)	Manoel Neto
Dia 6	Android	Renato Novais

Ambiente de desenvolvimento

- Git
- Python
- Django
- Android Studio

Instalação Git

- Git
- Github
-

Instalação Python

-
-

Instalação DJango

-
-

Instalação Android Studio

Testando as tecnologias (Hello World)

DIA 2

Mais conceitos, Git, Github

Tópicos a serem abordados (além do Docker)

- web/internet
- request/response
- html/css/javascript
- Servidor web
- Json
  - JsonViewer -> http://jsonviewer.stack.hu/
- Python/Django
- Serviços rest
- Git (add, commit, push, pull, merge, tag, branch, lista de commits, voltar para um commit específico, gitignore, github)

DIA 3

Front-end

First things first

- Programação Web é programação para a internet?
- O que é o HyperText Transfer Protocol (HTTP)?
- Como é a comunicação via HTTP?
- O que é Ajax?
- Qual a diferença entre URL, URI e URN?

Programação Web é programação para a internet?

Sim. Não. Depende.

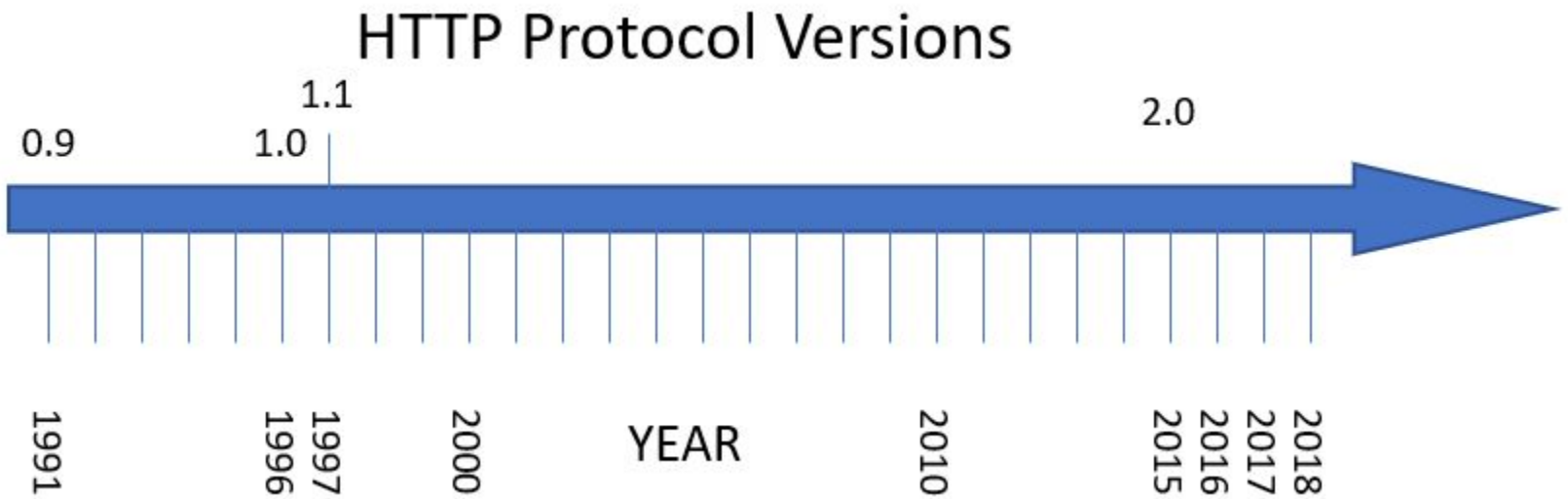
Muitas pessoas se confundem, mas a web e internet apesar de estarem relacionadas não são a mesma coisa.

O termo “internet” refere-se a “rede das redes” que interconecta computadores no mundo inteiro. Entretanto, existem outros tipos de redes como a intranet, por exemplo.

A programação web se refere ao desenvolvimento baseado nos serviços oferecidos pela camada de protocolos TCP/IP (SMTP, FTP, HTTP etc).

O que é HyperText Transfer Protocol (HTTP)?

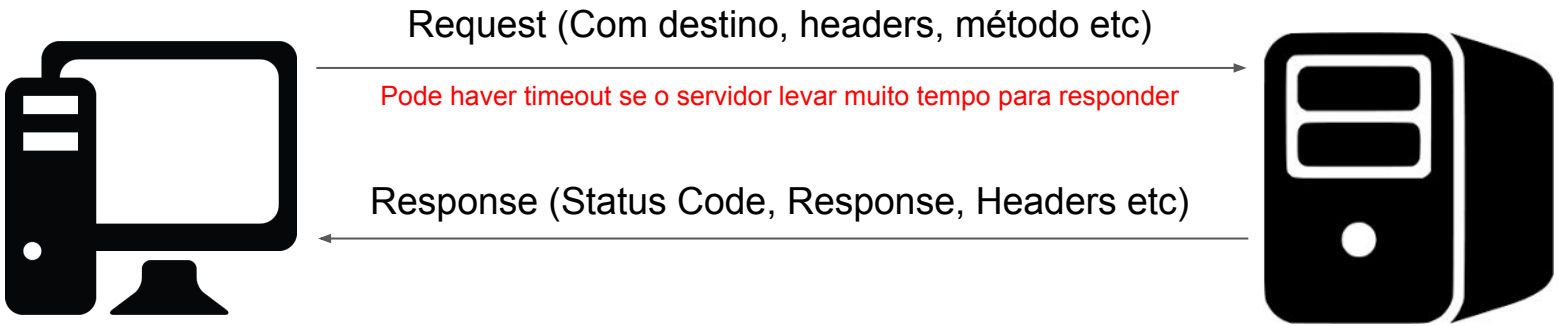
Protocolo de comunicação para transferência de **hipertexto** através de uma rede.



Fonte: JavaDeep

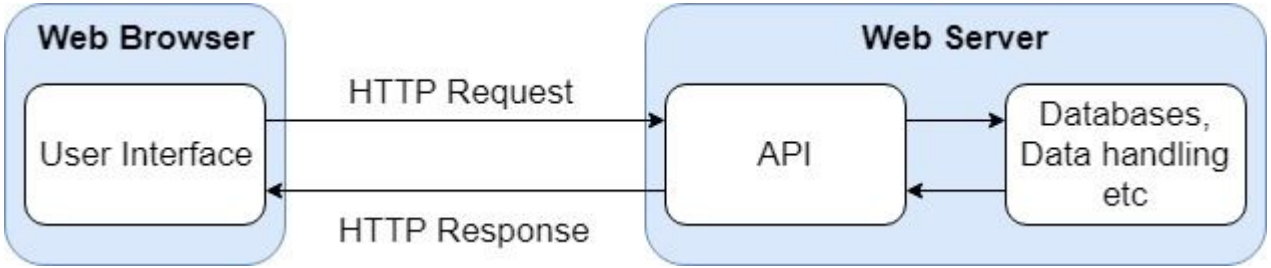
Como é a comunicação via HTTP?

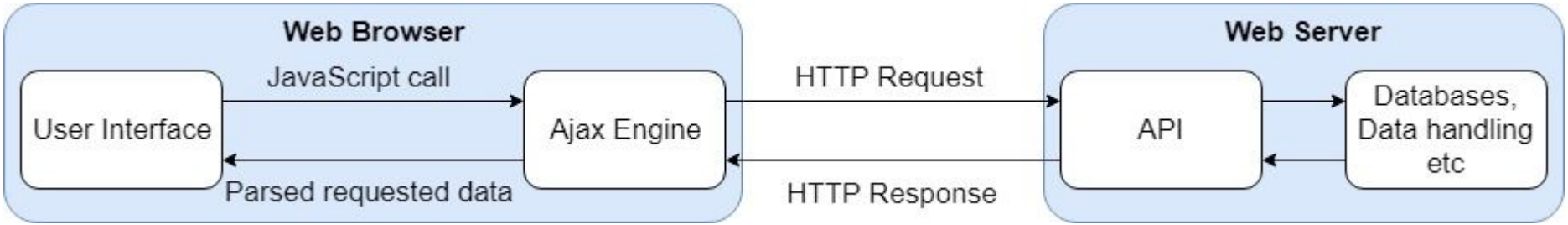
A comunicação é baseada num paradigma de **request** e **response** baseado em verbos/métodos que identificam a ação a ser executada (Principais: GET, POST, PATCH/PUT, DELETE).



O que é Ajax?

O Asynchronous JavaScript and XML é um meio de fazer comunicações assíncronas entre cliente e servidor.





Qual a diferença entre URL, URI e URN?

Todos os três são formas de identificação de recursos.

A URL e URN estão contidas na URI.

- URI = Uniform Resource Identifier (www.meusite.com/usuarios/lista)
- URL = Uniform Resource Locator (www.meusite.com)
- URN = Uniform Resource Name (/usuarios/lista)

HTML

HTML = HyperText Markup Language

Linguagem de marcação baseada em tags para construção de páginas Web para interpretação por navegadores.

É a estrutura de uma página web.

HTML

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title>Título da página</title>
    ... Metadados e parâmetros de configuração para o browser ...
</head>
<body>
    ... aqui vai todo o codigo HTML que faz seu site...
</body>
</html>
```

CSS

CSS = Cascading Style Sheets

Linguagem de estilo que permite adicionar espaçamentos, cores, animações e outros recursos a uma página web.

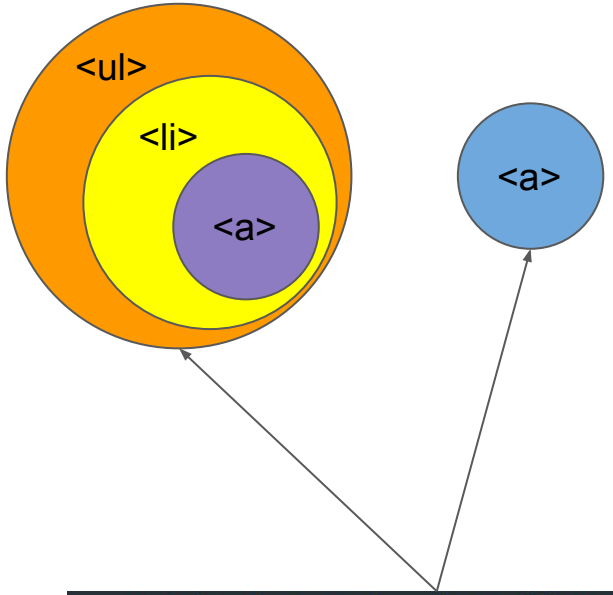
Sua função é única e exclusivamente propiciar uma experiência de uso mais agradável.

CSS

O código CSS usa uma ideia de conjuntos na sua interpretação.

Por conta disso, o CSS é interpretado de cima pra baixo da direita para a esquerda em termos de hierarquia de componentes.

Elementos com maior especificidade tem prioridade. No exemplo, a **ul li a** tem maior especificidade que **a**, por conta disso todo a dentro de li e ul é púrpura.




```
a { background-color: blue; }
ul { background-color: orange; }
li { background-color: yellow; }
ul li a { background-color: purple; }
```

CSS - ID, Classes e Elementos

No CSS é possível identificar elementos por **ID** (Ex: #meuId), **CLASSE** (Ex: .minhaClasse), **ELEMENTO** (Ex: button).

```html
<!-- EXEMPLO NO HTML -->
<button class="botao"
    id="botaoSalvar">
    SALVAR
</button>
```

```css
/*
    ID = Identifica (ao menos deveria)
    um único elemento
*/
#botaoSalvar {
    background-color: #00ff00; /* Cor do Botão */
}

/*
    CLASS = Identifica elementos de um mesmo tipo
    agrupados por um nome em comum
*/
.botao {
    color: white; /* Cor do texto */
}

/*
    ELEMENT = Atualiza TODOS elementos daquele tipo
*/
button {
    font-size: 12px; /* Tamanho do texto do botão */
}
```

CSS - Cores

Unidade	Descrição	Exemplo
Hexadecimal	Número hexadecimal que representa os canais RGB em alta e baixa frequência (RrGgBb)	#FF0000
Literals	Nomes de cores definidas convencionalmente entre browsers	red
RGB	Baseado em valores que identificam os canais de cores com intensidades entre 0 e 255	rgb(255, 0, 0)
RGBA	Igual ao RGB, adicionando o canal Alpha (transparência)	rgba(255, 0, 0, 1)
HSL	Cor definida por combinações de matiz (hue), saturação (saturation), claridade (lightness).	hsl(0, 100, 50)
HSLA	Igual ao HSL, adicionando canal Alpha (transparência)	hsla(0, 100, 50, 1)

CSS - Unidades Absolutas

Unidade	Descrição	Exemplo
Centímetros	Tamanho em centímetros	10cm
Milímetros	Tamanho em milímetros	10mm
Polegadas	Tamanho em polegadas	10in
Pixel	Tamanho em pixel. Apesar de absoluto, pode variar de acordo com a densidade de pixels (DPI) do dispositivo.	10px
Pontos	Tamanho em pontos	10pt
Picas (lá ele)	Tamanho em picas (lá ele)	10pc

CSS - Unidades Relativas

Unidade	Descrição	Exemplo
em	Tamanho relativo ao font-size de um elemento (2em = 2x o font-size daquele elemento)	2.73em
ch	Tamanho relativo ao Relativo ao width do caracter “0”	10ch
rem	Tamanho relativo ao font-size do elemento raiz	13rem
vw	Tamanho relativo a largura visível da tela (viewport width)	10vw
vh	Tamanho relativo a altura visível da tela (viewport height)	10vh
vmin	Tamanho relativo a menor dimensão visível para aquele elemento na tela	15vmin
vmax	Tamanho relativo a maior dimensão visível para aquele elemento na tela	12vmax
%	Tamanho relativo ao elemento pai	

JavaScript

Linguagem de programação que permitiu dinamizar o front-end das aplicações.

Originalmente criada para validações de formulários, hoje o JavaScript está presente em aplicações de todos os tipos (Front-end, Back-end, Mobile, IoT, entre outros).

JavaScript - Observações

- Linguagem **interpretada** que pode ser inserida em código HTML para ser aplicada através do browser.

```
<script type="text/javascript" src="meu-script.js"></script>
```

- A ordem dos scripts pode implicar na ordem com que eles são interpretados.

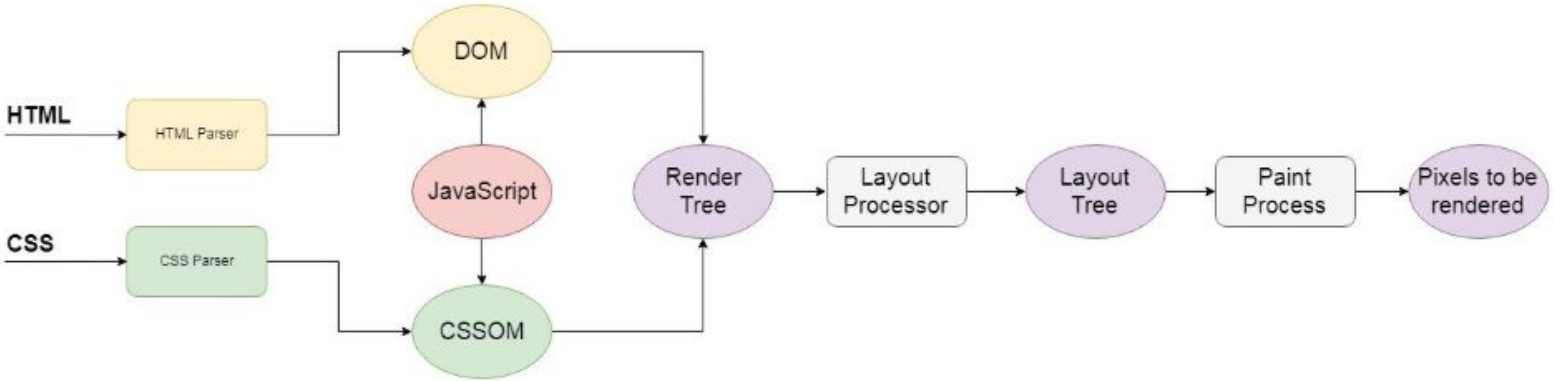
```
<script type="text/javascript" src="meu-script-1.js"></script>
<script type="text/javascript" src="meu-script-2.js"></script>
<script type="text/javascript" src="meu-script-3.js"></script>
<script type="text/javascript" src="meu-script-4.js"></script>
```

JavaScript - Observações

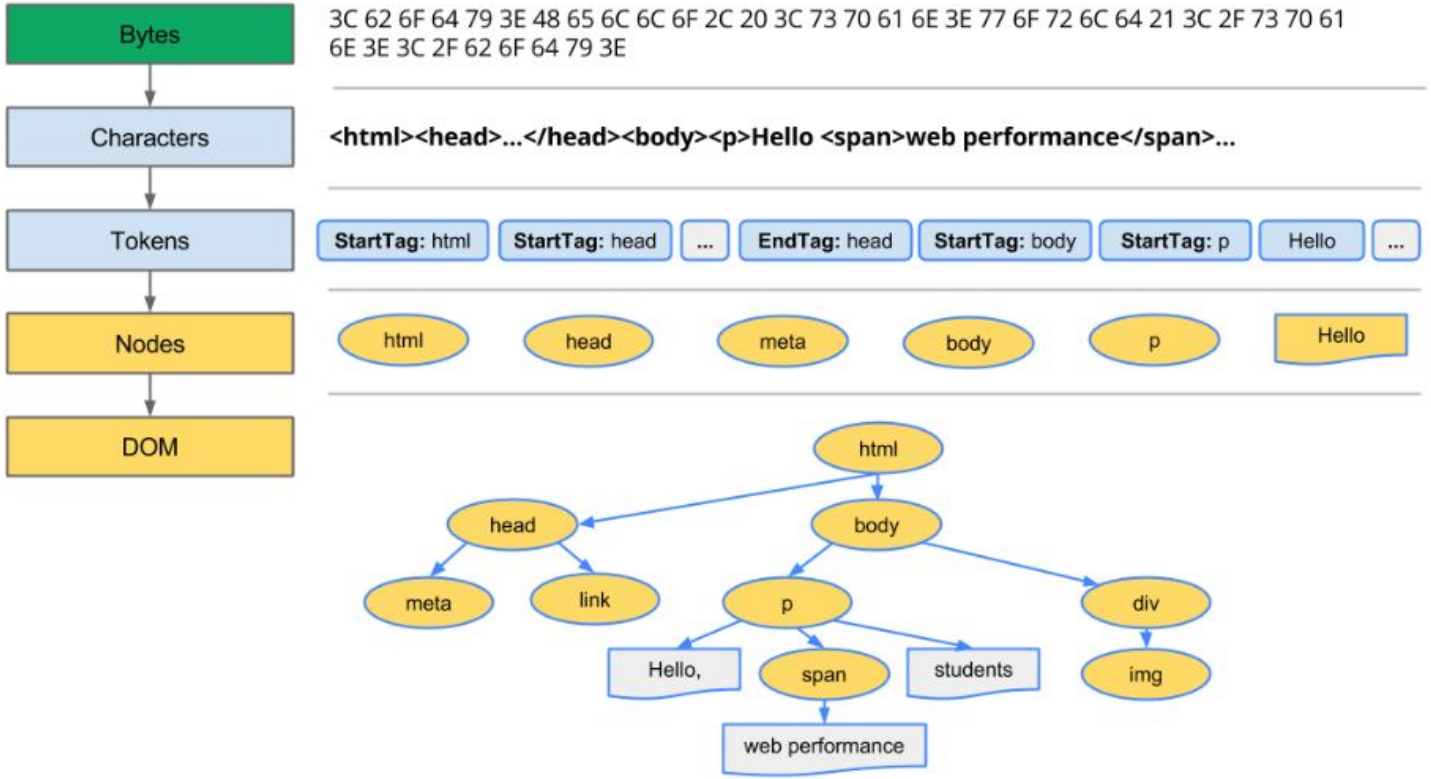
- A linguagem tem **tipagem dinâmica**. Isso significa que o tipo da variável será o tipo do valor que ela estiver armazenando.
- Tipos Primitivos: Number, String, Boolean, Null, Undefined e Symbol.
    - Os tipos primitivos contém valores **imutáveis**. Ou seja, 10 sempre será igual a 10.
- Tipos Não Primitivos: Objects, Functions, Arrays e Regex (Regular Expression)
    - Os tipos não primitivos **não são imutáveis**. Ou seja, um objeto não é igual ao outro, por exemplo.
- Null e Undefined **não são a mesma coisa**. O null representa uma referência nula, enquanto o undefined realmente representa “nada”.

Como meu site aparece na tela?

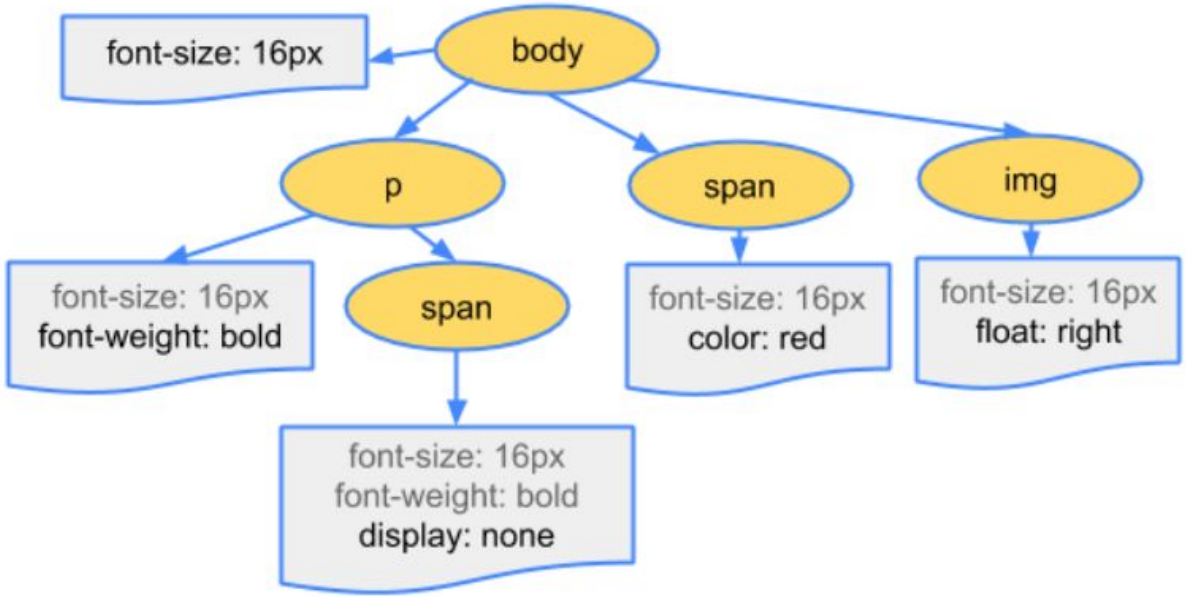
Todo código HTML e CSS é interpretado de forma a gerar um DOM e um CSSOM (que podem sofrer interferência do JavaScript). Após isso, a mescla de ambos gera uma Render Tree, que após ser processada pelo Layout Processor gera a Layout Tree e, por fim, passa pelo Paint Process e gera os pixels a serem renderizados na tela.



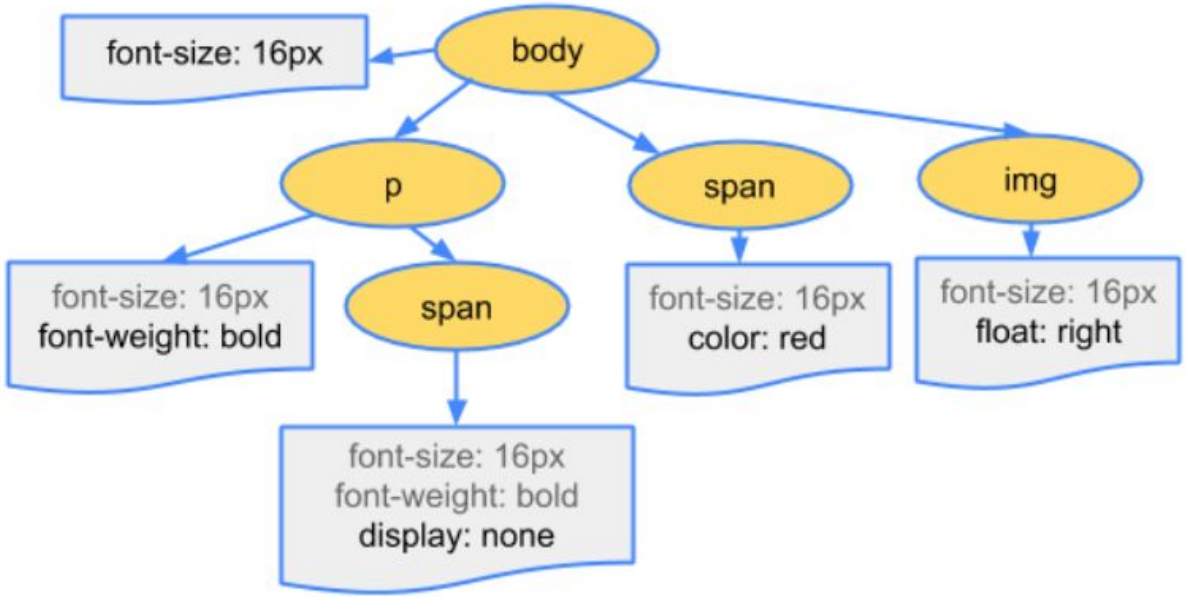
DOM (Document Object Model)



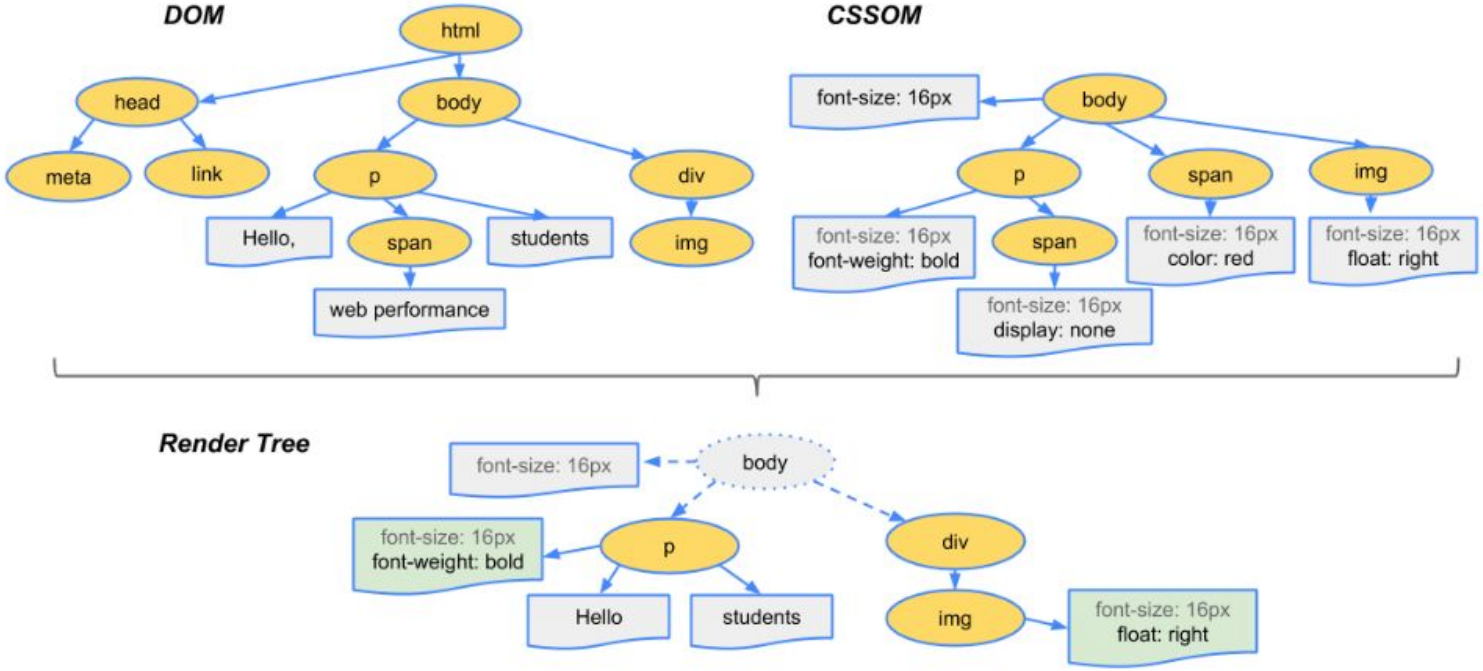
CSSOM (Cascade Style Sheet Object Model)



CSSOM (Cascade Style Sheet Object Model)



Render Tree


DOM
html
head
body
meta
link
p
div
Hello,
span
students
img
web performance
CSSOM
font-size: 16px
body
p
span
img
font-size: 16px
font-weight: bold
span
font-size: 16px
color: red
font-size: 16px
float: right
font-size: 16px
display: none
Render Tree
font-size: 16px
body
font-size: 16px
font-weight: bold
p
div
Hello
students
img
font-size: 16px
float: right

Layout Processor (Reflow Processing)

Após gerar a render tree, essa estrutura de dados passa por um processo chamado **Reflow Processing** no Layout Processor.

O Reflow Processing calcula o tamanho e posição de cada um dos elementos da render tree.

Todas as vezes que a tela é redimensionada o processo de reflow é refeito.

Painting Process

O painting process irá converter a Layout Tree em pixels que irão ser renderizados na tela.

https://github.com/HDeiro/sessao-da-tarde-2019-frontend

DIA 4

Python

O que vamos ver?

Python:

- A linguagem Python
- Sintaxe
  - Variáveis em Python
  - Tipos de dados
  - Operadores
  - Entradas e saídas
  - Comandos de decisão
  - Laços de repetição
  - Funções
  - Exceções e Erros
- Orientação a Objetos
- ORM + Django QuerySets

Sobre a Linguagem Python

Sobre a linguagem de programação Python

- Surgiu no ano de 1991, criada por Guido van Rossum.
- É uma linguagem de programação multiplataforma, de alto nível, interpretada, de script, imperativa, orientada a objetos, funcional, de tipagem dinâmica e forte.
- Permite desenvolver aplicações para games, desktops, web e dispositivos móveis.

Quem usa Python?

Sintaxe

Características importantes da programação em Python

- Python não usa ponto e vírgula (;) para delimitar o fim de uma instrução. Ex.:
  - `print(12 + 6)`
- Python usa indentação como delimitação de bloco, portanto devemos indentar corretamente o código fonte. Exemplo de código:
  - ```
    def spam():
        eggs = 12
        return eggs
    ```
    ```

Conceito de variável em Python

- Todas as variáveis são representadas por um objeto e todas elas possuem uma referência.
- Variáveis armazenam endereços de memória, não os seus valores.
- As variáveis não possuem um tipo fixo (tipagem dinâmica). Exemplos:
  - idade = 17, nome = “Marcos Sobral”, colecao = [1,2,3]
- O interpretador não faz conversões automáticas de valores entre tipos não compatíveis (tipagem forte). O seguinte trecho de código resultará na exception *TypeError:*

```
i, j = 10, "Rafael"
print("O resultado é: ", i + j)
```

Tipos de Dados - *Inteiros*

- Para a declaração de números inteiros, necessitamos que estes estejam entre -2147483648 a 2147483647;
- Para declarar um inteiro octal, deve ser utilizado o prefixo 0o;
- Para definir um número hexadecimal, o prefixo 0x deve ser utilizado;
- Para declarar um número binário, o prefixo 0b deve ser utilizado.
- Exemplos:

```
a = 42 #decimal
b = 0o10 #octal
c = 0xA #hexadecimal
d = 0b10 #binário
```

Tipos de Dados - *Long*

- Representa números inteiros longos e pode armazenar números tão grandes quanto a memória puder armazenar;
- Assim como o tipo int, o long também pode armazenar números tanto decimais, quanto octais e hexadecimais;
- Para declarar um valor long é necessário sufixar com a letra L (minúscula ou maiúscula);
- Exemplos:

```
a = 999998799941L #decimal
b = 0o10L #octal
c = 0xDDEFBDAEFBDAECBL #hexadecimal
d = 0b11111111111111L #binário
```

Tipos de Dados - *Float*

- Representa números reais e que possuem sinal de expoente (e ou E);
- Exemplos:

```
a = 0.0042
b = .005
c = 1.41965
d = 6.02e23
```

Tipos de Dados - *Bool*

- O tipo bool foi adicionado na versão 2.2 do Python como uma especialização do tipo int;
- Os valores do tipo bool podem representar dois valores completamente distintos: True (igual ao int 1) e False (igual ao int 0) ;
- Exemplos:

```
a = True #verdadeiro
b = False #falso
```

Tipos de Dados - *None*

- É uma constante embutida do Python que, assim como True e False, é frequentemente utilizada para representar a ausência de um valor, similar ao null na linguagem C e derivadas.
- Exemplos:

```
a = None # vazio
```

Tipos de Dados - *String*

- Para atribuirmos a uma variável uma referência do tipo string, basta que coloquemos entre aspas simples, duplas ou triplas.
- Exemplos:
  - `a = 'Isso é uma String com aspas duplas'`
  - `b = “Isso é uma String com aspas duplas”`
  - `c = “””Isso é uma String com aspas triplas”””`

Tipos de Dados - Lista

- Uma lista em Python é uma sequência ou coleção ordenada de valores, que podem possuir tipos diferentes (porém, lista são destinadas a serem sequências homogêneas).
  - **Declarando uma lista:**
    - `lista = ["Amarula", "Goiabada","Rosé"]`
  - **Adicionando um novo valor à lista:**
    - `lista.append("Salvador")`
  - **Removendo um valor da lista:**
    - `lista.remove("Amarula")`
  - **Obtendo um valor da lista:**
    - `lista[0]`
  - **Verificando se um valor existe na lista:**
    - `if "Salvador" in lista: print("existe")`
  - **Ordenando a lista:**
    - `lista.sort()`
  - **Invertendo posição dos itens:**
    - `lista.reverse()`
  - **Removendo o último valor da lista:**
    - `lista.pop()`

Tipos de Dados - Tupla

- Tupla é uma lista imutável, ou seja, depois de declarada não é permitido nenhum tipo de alteração (adição, remoção, ordenação…).
  - **Formas de declarar uma tupla:**
    - `tupla01 = (“Amarula”, “goiabada”, 123, 111)`
    - `tupla02 = “Amarula”, 123, 111`
  - **Obtendo valores de uma tupla:**
    - `tupla01[0]`

Tipos de Dados - Dicionários

- Dicionários são tipos de coleções um pouco diferente, pois se trata de uma coleção associativa desordenada. A associação dos valores é realizada através de uma **chave** imutável para um **valor** de qualquer tipo.
  - **Declarando um dicionário:**

```
dic = {
  "eu": "codo",
  "tu": "codas",
  "eles": "codam"
}
```

  - **Adicionando um novo ao dicionário:**

```
dic["nós"] = "codamos"
```

  - **Alterando um valor do dicionário:**

```
dic["nós"] = "codamos muito"
```

  - **Obtendo um valor do dicionário:**

```
dic["eu"]
dic.get("eu")
```

  - **Obtendo uma lista das chaves do dicionário:**

```
dic.key
```

  - **Obtendo uma lista dos valores do dicionário:**

```
dic.values
```

Operadores

Aritméticos	Comparação	Lógicos
+	==	and
-	!=	or
*	>	not
/ ou // (parte inteira)	<	
%	>=	
+= -= *= /=	<=	
**	in in not	
is		

Entradas e saídas de dados

- A função **print()** serve para imprimir argumentos, passados por parâmetro, no terminal.
  - `print("Hey, world!")`
- A função **input()** aguarda uma entrada do usuário no terminal.
  - `nome = input("Informe seu nome: ")`

Comandos de decisão

- Os comandos **if, elif** e **else** servem para determinar se uma parte do código será executada.
  - ```
    if nota > 7.0:
        print("Você está aprovado por média.")
    elif nota < 7.0:
        print("Você foi reprovado por média.")
    else:
        print("Foi por pouco, pai!")
    ```
    ```

Comandos de decisão com operadores lógicos

- Os operadores lógicos servem para combinar expressões lógicas.
  - ```
    if nota >= 5.0 and nota <= 7.0:
        print("Ketchup")
    elif nota > 7.0 and not nota > 9.8:
        print("Rabanete")
    elif nota == 9.9 or nota == 10.0:
        print("Abacaxi")
    ```
    ```

Comandos de repetição

- Os comandos **for** e **while** permitem executar um bloco de código repetidas vezes, enquanto uma dada condição é atendida.
  - ```
    vinhos = [“Tintos”, “Brancos”, “Rosés”]

    for tipo in vinhos:

        print(tipo)
    ```
  - ```
    count = 0

    while count < 5:

        print(tipo)

        Count = count + 1
    ```

Funções

- Funções são blocos de código que realizam alguma tarefa que normalmente serão executadas diversas vezes dentro de uma aplicação.
  - **Definindo uma função simples:**
    - ```
      def fala_meu_nome(nome):
          print(nome)
      ```
  - **Chamando uma função em python:**
    - ```
      fala_meu_nome("Sobral")
      ```
  - **Definindo uma função com retorno:**
    - ```
      def somar(a, b):
          return a + b
      ```
  - **Armazenando retorno:**
    - ```
      soma = somar(7,10)
      ```

Exceções e Erros

- Uma exceção é algo que não ocorre como planejado no decorrer da aplicação em execução.

  - **Exemplo de situação que causa uma exceção (TypeError):**

```
a = 10
b = "dois"
soma = a + b
```

  - **Tratando a exceção anterior:**

```
try:
    a = 10
    b = "dois"
    soma = a + b
except TypeError:
    print("Não é possível somar tipos de dados diferentes")
```

Orientação a Objetos

Orientação a Objetos

O que vamos ver?

- Classes e Objetos
- Atributos e Métodos
- Getters e Setters
- Herança e Polimorfismo

Objetos

Os objetos em Python apresentam os seguintes atributos:

- **Tipo**: O tipo de um objeto determina os valores que o objeto pode receber e as operações que podem ser executadas nesse objeto.
- **Valor**: O valor de um objeto é o índice de memória ocupada por essa variável. Como os índices das posições da memória são interpretados, isto é determinado pelo tipo da variável.
- **Tempo de vida**: A vida de um objeto é o intervalo de tempo de execução de um programa em Python, é durante este tempo que o objeto existe.

Classes

- Uma **classe** define uma estrutura de dados que contenha instância de atributos, instância de métodos e classes aninhadas.

- **Classe** é definida como um agrupamento de valores e operações. As classes facilitam a modularidade e abstração de complexidade.

Atributos e Métodos

- Um **classe** criada é chamada de classe objeto. Os nomes no namespace da classe objeto são chamados de **atributos** da classe. Funções definidas dentro de uma classe são chamadas de **métodos**.

Hora do código

- **Definindo uma classe com atributos e métodos:**

```
class CalculadoraSimples:
        # método construtor da classe
        def __init__(self, n1, n2):
                self.n1 = n1 # atributo n1
                self.n2 = n2 # atributo n2

        # método que realiza adição
        def somar(self):
                return self.n1 + self.n2

        # método que realiza subtração
        def subtrair(self):
                return self.n1 - self.n2
```

Getters e Setters

- Em **Python** não há o costume de se usar getters e setters. Porém, existe um **mecanismo** chamado de “propriedade” (*property*).

- O *property* é ideal quando, ao ler ou escrever um atributo, algum código customizado deverá ser executado.

Hora do código

- **Utilizando property:**

```
class CreditCard:
        # método construtor da classe
        def __init__(self, number):
                self.number = number

        @property
        def number(self):
                # implementar código customizado, por exemplo retornar os últimos 4 números do cartão
                return self.number

        @number.setter
        def number(self, value):
                # implementar código customizado, exemplo: validador de cartão
                return self.number
```

Herança

- Frequentemente classes diferentes possuem características comuns. As classes diferentes podem compartilhar valores comuns e podem executar as mesmas operações. Em Python tais relacionamentos são expressados usando derivação e herança.

- Na Programação Orientada a Objetos o conceito de herança é muito utilizado. Basicamente, dizemos que a herança ocorre quando uma classe (filha) herda características e métodos de uma outra classe (pai), mas não impede de que a classe filha possua seus próprios métodos e atributos.

Hora do código

- **Implementando herança:**

```
from calculadora import CalculadoraSimples

class CalculadoraPower(CalculadoraSimples):
        # método construtor da classe
        def __init__(self, n1, n2):
                super().__init__(n1,n2)

        # método que realiza multiplicação
        def multiplicar(self):
                return self.n1 * self.n2

        # método que realiza divisão
        def dividir(self):
                return self.n1 / self.n2
```

Polimorfismo

- **Polimorfismo** consiste na possibilidade de classes derivadas de uma mesma superclasse invocar métodos que têm a mesma identificação (assinatura) mas comportamentos distintos.

Hora do código

- **Exemplo de polimorfismo:**

```
from calculadora import CalculadoraSimples

class CalculadoraBugada(CalculadoraSimples):
        # método construtor da classe
        def __init__(self, n1, n2):
                super().__init__(n1,n2)

        # sobrescrevendo método da classe mãe
        def somar(self):
                return super().subtrair()

        # sobrescrevendo método da classe mãe
        def subtrair(self):
                return super().somar()
```

ORM e Querysets

Bancos de Dados Relacionais

- Um banco de dados relacional armazena dados em tabelas e possui uma Linguagem de Consulta Estruturada (**SQL**).

- Tabelas são organizadas em colunas, e cada coluna armazena um tipo de dados, que pode ser: inteiros, números reais, strings, dentre outros.

- Os dados de um registro (instância) são armazenados como uma linha.

- Em um banco de dados normalizado, cada registro possui uma ou mais chaves que identificam que sejam responsáveis pela identificação da mesma.

Mapeamento Objeto-Relacional (ORM)

- Um ORM é uma biblioteca de códigos que automatiza a transferência de dados armazenados em tabelas de bancos de dados relacionais para objetos.

- Os ORMs fornecem uma abstração de alto nível de um banco de dados relacional. Dessa forma, um desenvolvedor pode criar, ler, atualizar e excluir dados em seu banco de dados, utilizando sua linguagem de programação ao invés de escrever instruções SQL.

id	nome_completo	idade
1	Marcos Sobral	17
2	Roberto Carlos	32
3	Jaspion Brasileiro	17

Class Pessoa:
id = 1
nome_completo = “Marcos Sobral”
idade = 17

Class Pessoa:
id = 2
nome_completo = “Roberto Carlos”
idade = 32

Class Pessoa:
id = 3
nome_completo = “Jaspion Brasileiro”
idade = 24

QuerySets do Django

- Um QuerySet (conjunto de busca) é uma lista de objetos de um determinado modelo (classe Model).

- Com o QuerySet do Django é possível ler os dados a partir de uma base de dados, filtrá-los e ordená-los.

SQL x Django QuerySets

Adicionar uma pessoa ao banco de dados.

SQL

```
INSERT INTO pessoas (id, nome_completo, idade)
VALUES (1, “Marcos Sobral, 17);
```

Django ORM

```
Pessoas.objects.create(id=1, nome_completo=”Marcos Sobral”, idade=17)
```

SQL x Django QuerySets

Atualizar idade da pessoa de id = 1 para 25 anos.

SQL

```
UPDATE pessoas SET idade = 25 WHERE id = 1;
```

Django ORM

```
Pessoas.objects.get(id=1).update(idade=25)
```

SQL x Django QuerySets

Obter registros de pessoas com idade igual a 25 anos.

SQL

```
SELECT * FROM pessoas WHERE idade = 25;
```

Django ORM

```
pessoas = Pessoas.objects.filter(idade=25)
```

SQL x Django QuerySets

Obter dados da pessoa que de id = 1.

SQL

```
SELECT * FROM pessoas WHERE id = 1;
```

Django ORM

```
pessoa = Pessoas.objects.get(id=1)
```

SQL x Django QuerySets

Obter todos os registros de pessoas cadastradas no banco de dados.

SQL

```
SELECT * FROM pessoas;
```

Django ORM

```
pessoas = Pessoas.objects.all()
```

Django QuerySets - Extras

Fazer união de dois ou mais QuerySets

qs1.union(qs2, qs3)

Obter a interseção de dois QuerySets

qs1.intersection(qs2, qs3)

Obter a diferença entre dois ou mais QuerySets

qs1.difference(qs2, qs3)

Django QuerySets - Extras

Filtro com operador AND

```
pessoas = Pessoa.objects.filter(idade=25) &
Pessoa.objects.filter(nome=”Marcos”)
```

Filtro com operador OR

```
pessoas = Pessoa.objects.filter(idade=25) |
Pessoa.objects.filter(idade=17)
```

Referências

- https://www.devmedia.com.br/python-tutorial/33274
- https://wiki.python.org.br/ProgramacaoOrientadaObjetoPython
- https://tutorial.djangogirls.org/pt/django_orm/
- http://dirtsimple.org/2004/12/python-is-not-java.html
- https://www.fullstackpython.com/object-relational-mappers-orms.html
- https://www.devmedia.com.br/bancos-de-dados-relacionais/20401
- https://docs.djangoproject.com/en/2.2/ref/models/querysets/

DIA 5

DJango + Django Rest Framework

DJango

Sistema de Controle de livros
CRUD

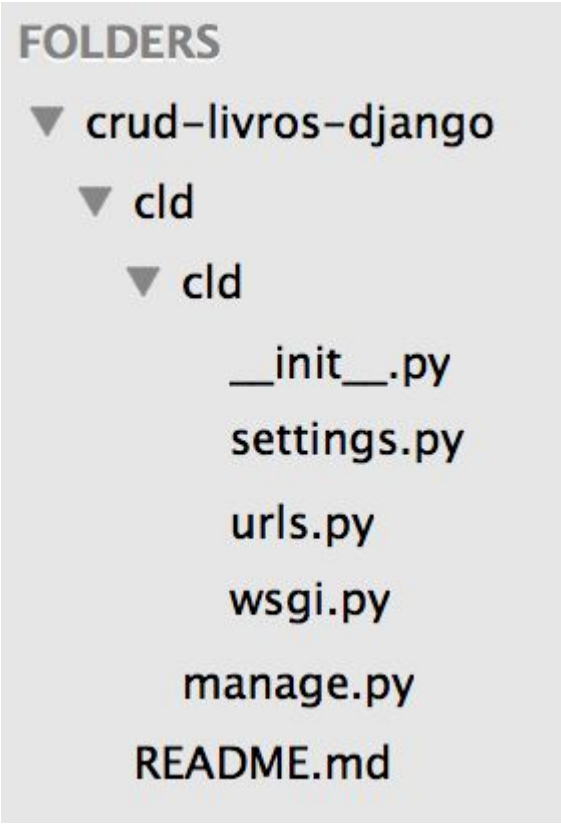
https://github.com/renatoln/crud-livros-django

Configurando/Inicializando o Ambiente

- Ambiente com python
- Criar uma virtual (duas opções abaixo)
  - virtualenv -p python3 crud-fullstack
  - python3 -m venv crud-fullstack
- Inicializar a virtual env (depende do sistema operacional que você utiliza)
  - source crud-fullstack/bin/activate
- Parar a virtual env
  - source deactivate
- Instalar o DJango dentro da virtual env
  - pip install django==2.1.2 whitenoise==2.0
- Usando o requirements.txt (documente as dependências dentro desse arquivo)
  - pip install -r requirements.txt

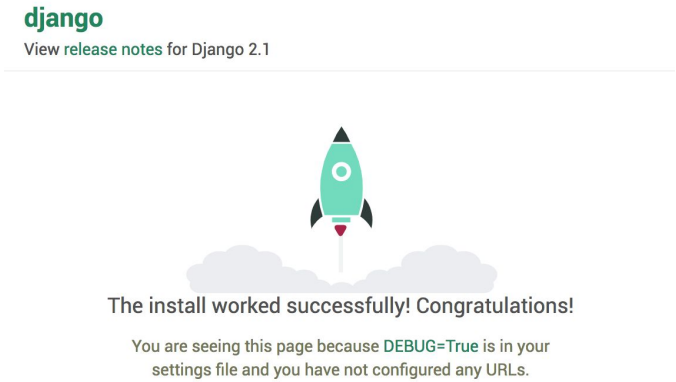
Criando a aplicação cld (crud-livros-django)

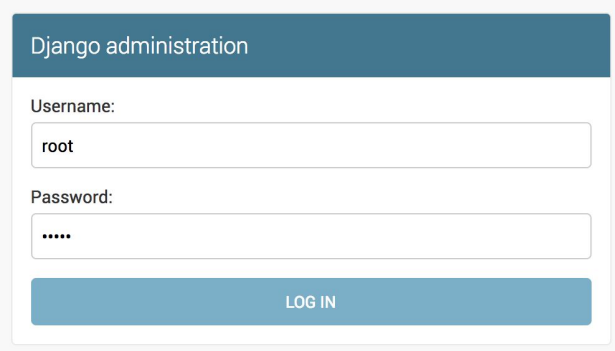
- Neste momento você deve navegar para dentro da pasta do projeto vazio (clone do github)
  - cd crud-livros-django
- Criar um novo projeto
  - django-admin startproject cld
- Os arquivos criados…
  - settings.py
    - contém várias configurações do projeto
  - Urls.py
    - Contém uma lista de padrões utilizadas pelo urlresolver
  - manage.py
    - Um script que ajuda na gestão do site



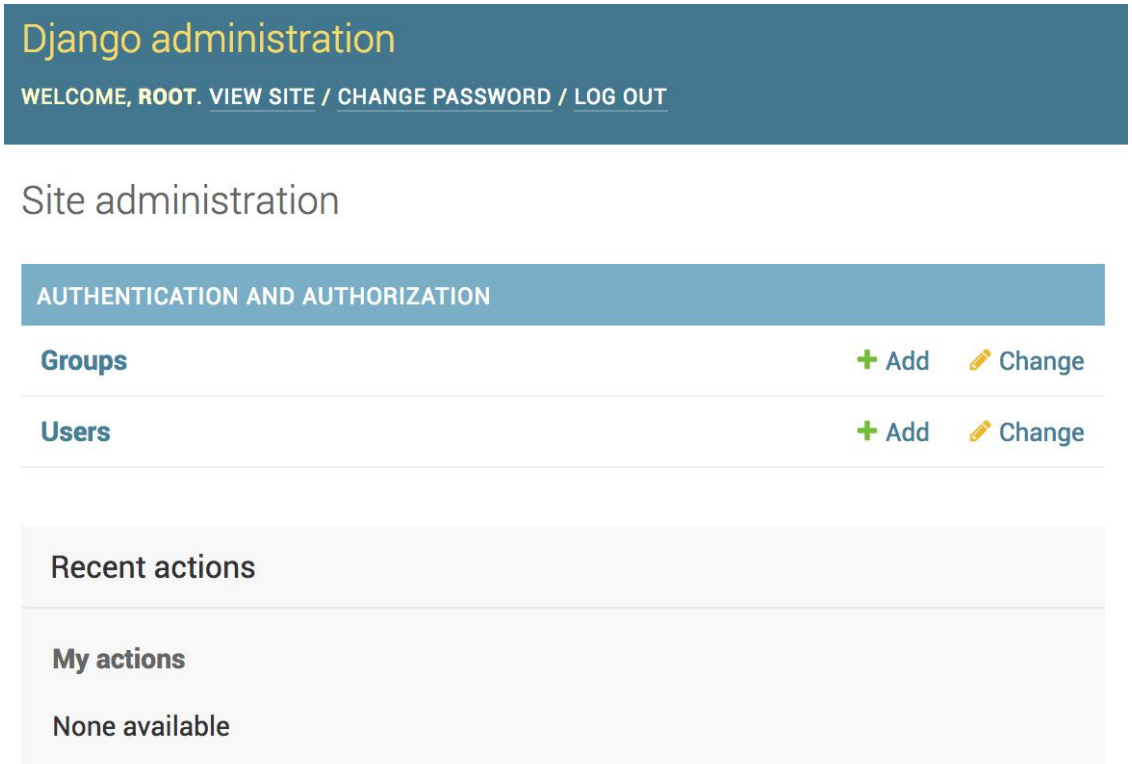
Criando a aplicação cld (crud-livros-django)

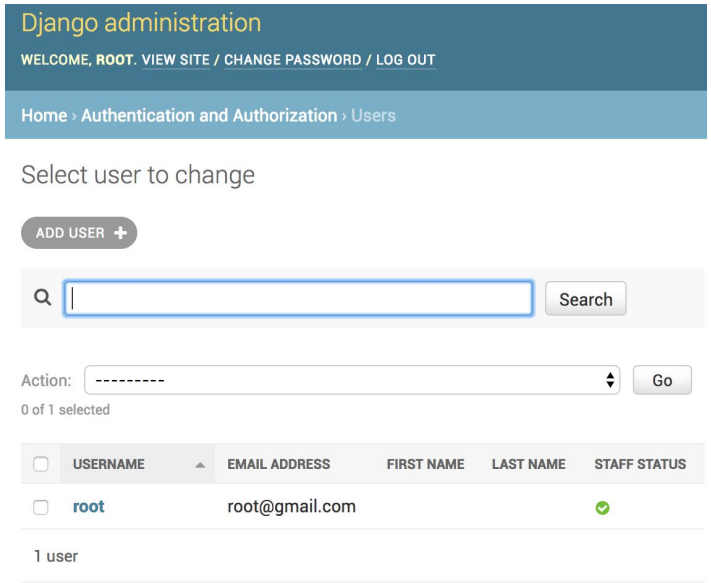
- Migrar as tabelas
  - cd cld
  - python manage.py migrate
- Criando super usuário
  - python manage.py createsuperuser
    - Usuário: root / email: qualquer@coisa.com / senha: admin
- Rodar o servidor
  - python manage.py runserver
- Acesse o site em http://127.0.0.1:8000/
- Acesse área admin em http://127.0.0.1:8000/admin
- Parar o servidor: ctrl + c





Área administrativa do cld



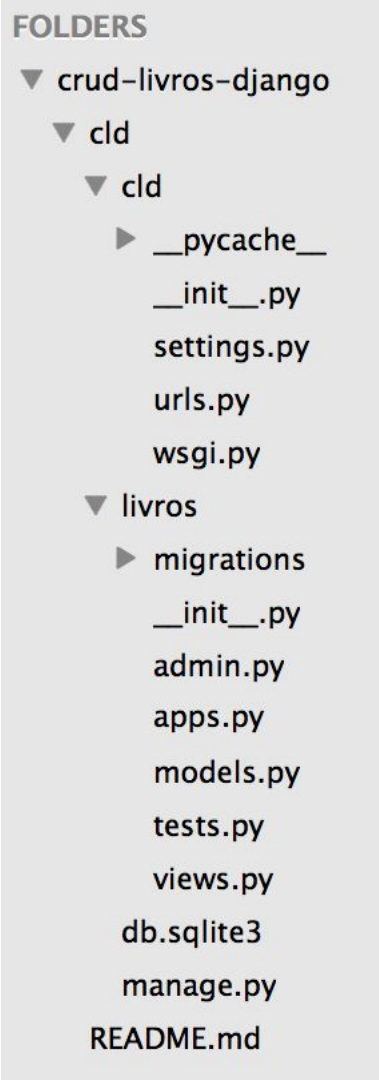


Configurações básicas no settings.py

- Time zone
  - TIME_ZONE = 'America/Sao_Paulo'
- Permitir outros hosts
  - ALLOWED_HOSTS = ['127.0.0.1', '.pythonanywhere.com']
- Mudar a língua padrão
  - LANGUAGE_CODE = 'pt-BR'
- Indicar onde ficam os arquivos estáticos (ao final do arquivo)
  - STATIC_ROOT = os.path.join(BASE_DIR, 'static')

Criar o novo módulo (app)

- Criar uma app livros
  - python manage.py startapp livros
- Os arquivos criados…
  - Admin.py
    - Admininstração da app
  - Models.py
    - Onde são adicionados os modelos da app
  - Views.py
    - A views da app



Registrar a nova app livros

- Dentro do arquivo settings.py

```
# Application definition

INSTALLED_APPS = [
        'livros',
        'django.contrib.admin',
        'django.contrib.auth',
        'django.contrib.contenttypes',
        'django.contrib.sessions',
        'django.contrib.messages',
        'django.contrib.staticfiles',
]
```

Definindo os modelos

- Em livros/models.py, adicione

```
from django.db import models
# Create your models here.

class Livro(models.Model):
    codigo = models.IntegerField()
    ISBN = models.CharField(max_length=50)
    titulo = models.CharField(max_length=100)
    autor = models.CharField(max_length=60)
    ano = models.IntegerField()
    editora = models.CharField(max_length=60)

    def __str__(self):
        return self.titulo
```

- Execute a migração
  - python manage.py makemigrations
- Crie a tabela
  - python manage.py migrate
- Registre o model em livros/admin.py

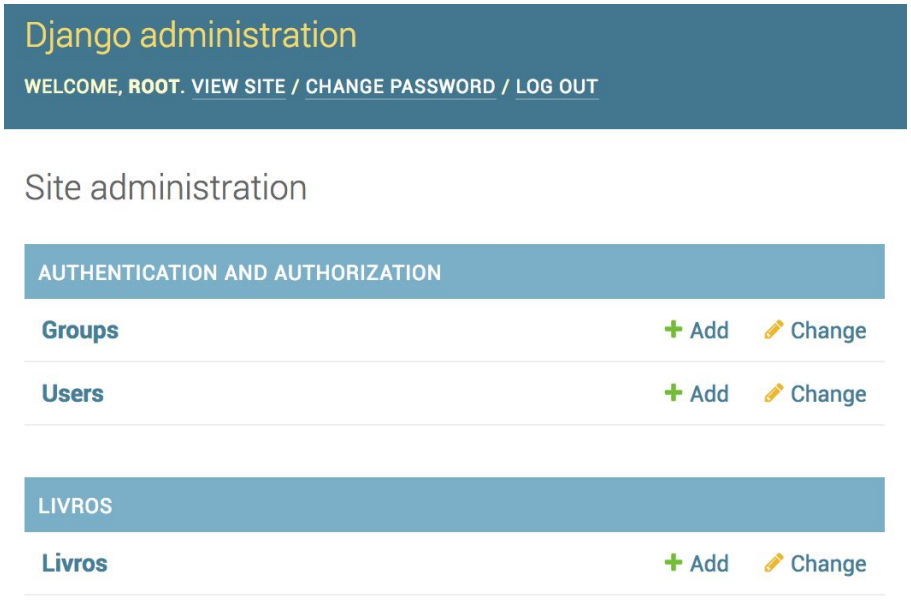
```
from django.contrib import admin

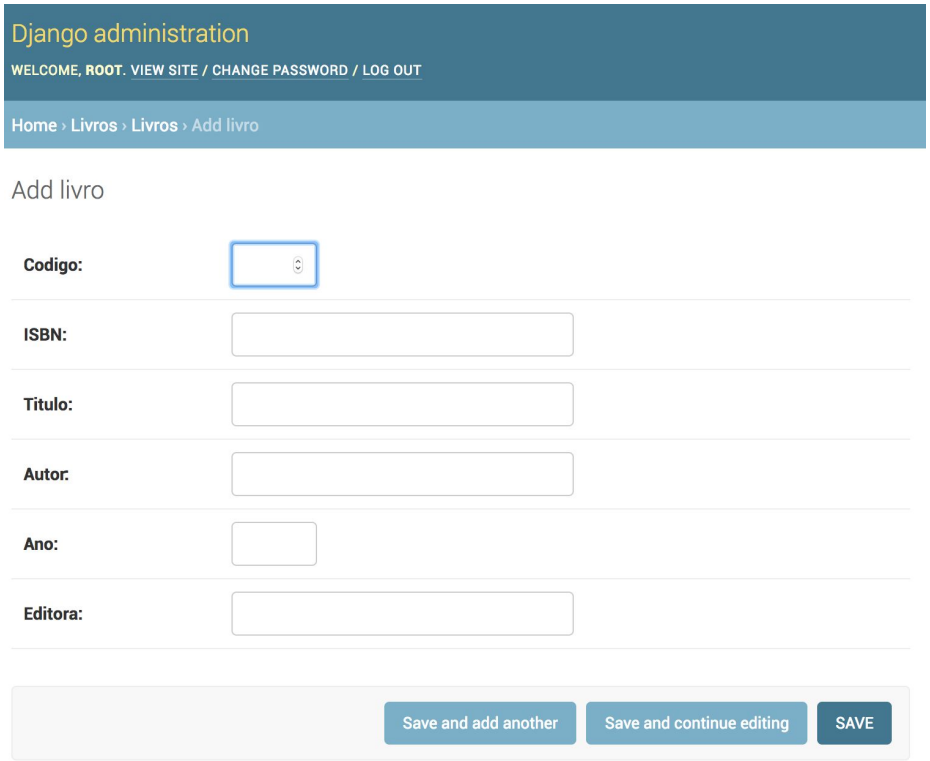
# Register your models here.

from .models import Livro

admin.site.register(Livro)
```

Definindo os modelos





Fazendo o deploy no pythonanywhere

- Esteja com seu código no github
- Logue no pythonanywhere.com
- Crie um api token. Accounts/Api token/create
- Vá em Dashboard/consoles/ crie um novo console do tipo bash
- Digite
  - pip3.6 install --user pythonanywhere
  - pa_autoconfigure_django.py https://github.com/renatoln/crud-livros-django.git
- Caso precise instalar algum pacote individualmente, faça
  - python -m pip install --user djangorestframework
  - Neste exemplo, estamos instalando djangorestframework, que será utilizado mais na frente

Criando a primeira urls

- Crie o arquivo **urls.py** dentro da pasta livros

```
// urls.py

from django.urls import path

from . import views

urlpatterns = [
        path('index', views.index, name='index'),
        path('', views.livro_list, name='livro_list'),
]
```

- Fazer referência a este novo **urls.py** dentro do urls.py principal que fica na pasta inicial do projeto

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
        path('admin/', admin.site.urls),
        path('livros/', include('livros.urls')),
]
```

Obs. se rodar agora dá um erro uma vez que ainda não existe a view livro_list

```
de-crud-fullstack/crud-livros-django/cld/livros/urls.py", line 7, in <
    path('', views.livro_list, name='livro_list'),
AttributeError: module 'livros.views' has no attribute 'livro_list'
```

Criando a view (1)

- Dentro de livros/views.py adicione o seguinte código para enviar a resposta HTTP

```
from django.shortcuts import render
from django.http import HttpResponse

from . import urls

# Create your views here.

def index(request):
    return HttpResponse("Olá, Mundo. Você está na página index de livros")
```

- Rode o servidor novamente e acesse a url seguinte
  - http://127.0.0.1:8000/index

Criando a view (2)

- Adicione também a view livro_list

```
from django.shortcuts import render
from django.http import HttpResponse

from . import urls

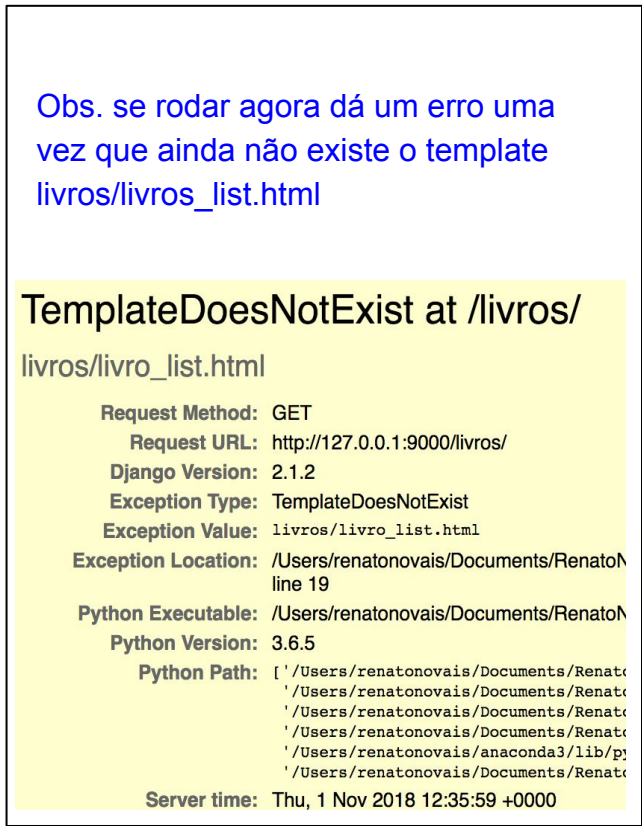
# Create your views here.

def index(request):
        return HttpResponse("Olá, Mundo. Você está na
página index de livros")

def livro_list(request):
    return render(request, 'livros/livro_list.html', {})
```

- Rode o servidor novamente e acesse a url seguinte
  - http://127.0.0.1:8000/livros

Obs. se rodar agora dá um erro uma vez que ainda não existe o template livros/livros_list.html



Criando os Templates

- Dentro da pasta livros, crie uma a seguinte estrutura
  - **templates/livros/livro_list.html**
- Código do arquivo index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Django CRUD Application</title>
</head>
<body>
    <h1>Lista de Livros</h1>
</body>
</html>
```

Avisar ao DJango onde estão os Templates

- No arquivo settings.py, edite o array TEMPLATES

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR,'templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

Isso realmente é necessário? No django girls isso não foi feito. Bastou criar a estrutura de pasta dentro do app/templates/livros/livro_list.html

???

Adicionar o caminho para os arquivos estáticos

- No arquivo settings.py, ao final adicione a linha em vermelho

```
STATIC_URL = '/static/'
STATIC_ROOT = os.path.join(BASE_DIR, 'static')
```

Fazendo o deploy no pythonanywhere

- Commit/push as alterações para o github
  - git add / commit / push
- No pythonanywhere, no bash
  - Entre no console e atualize o código a partir do github
    - cd ~/<your-pythonanywhere-username>.pythonanywhere.com
    - git pull

Dados dinâmicos em Templates

- Objetivo: Fazer com que os templates html mostrem os dados do banco
- Em livros/views.py adicione

```
from django.shortcuts import render
from django.http import HttpResponse
from .models import Livro
from . import urls


...


def livro_list(request):
    livros = Livro.objects.order_by('titulo')
    return render(request, 'livros/livro_list.html', {'livros': livros})
```

Tags de template do django

- Objetivo: colocar código python para gerar o html
- Em livros_list.html adicione `{{ livros }}` (variável definida na view)
- Coloque o código a seguir dentro da tag <body>

```
<div>
    <h1><a href="/">Lista de livros</a></h1>
</div>

{% for livro in livros %}
    <div>
        <h2><a href="">{{ livro.titulo }}</a></h2>
        <p>Código: {{ livro.codigo }}</p>
        <p>Autor(es): {{ livro.autor|linebreaksbr }}</p>
    </div>
{% endfor %}
```

Estendendo templates

- Reutilizar partes do template
- Crie um arquivo base.html na pasta de templates
- Mova todo conteúdo de livros_list.html para base.html
- No topo do arquivo adicione
  - `{% load static %}`
- Substitua o código do <body> de base.html para o código do body a seguir
- O código de base.html deve ficar como o código ao lado

```
{% load static %}
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible"
content="ie=edge">
    <title>Django CRUD Application</title>
</head>

<body>
        <div>
            <h1><a href="/livros">Lista de Livros</a></h1>
        </div>
            {% block content %}
            {% endblock %}
</body>

</html>
```

Estendendo templates

- Em livros_list.html, que está vazio, adicione

```
{% extends 'livros/base.html' %}

{% block content %}

{% for livro in livros %}
    <div>
        <h2><a href="">{{ livro.titulo }}</a></h2>
        <p>Código: {{ livro.codigo }}</p>
        <p>Autor(es): {{ livro.autor|linebreaksbr }}</p>
    </div>
{% endfor %}

{% endblock %}
```

Criando a página de detalhes de livros

- É preciso: criar a url, a view e o template
- Primeiro, em livro_list.html, adicione um link que vai passar a informação do post clicado.

```
{% extends 'livros/base.html' %}

{% block content %}

{% for livro in livros %}
    <div>
        <h2><a href="{% url 'livro_detail' pk=livro.pk %}">{{ livro.titulo }}</a></h2>
        <p>Código: {{ livro.codigo }}</p>
        <p>Autor(es): {{ livro.autor|linebreaksbr }}</p>
    </div>
{% endfor %}

{% endblock %}
```

{% url 'livro_detail' pk=livro.pk %}

- url 'livro_detail' => O Django espera que exista uma url de nome livro_detail especificada.
- pk = livro.pk, será criado uma variável pk, de valor livro.pk (chave primária de livro). pk será linkado do template e depois para a url para a view
- Em livros/ulrs.py adicione:

```
from django.urls import path

from . import views

urlpatterns = [
    path('index', views.index, name='index'),
    path('', views.livro_list, name='livro_list'),
    path('livro/<int:pk>/', views.livro_detail, name='livro_detail'),
]
```

Formulários do DJango

- Crie um arquivo chamado forms.py dentro de livros, com o seguinte conteúdo

```
from django import forms

from .models import Livro

class LivroForm(forms.ModelForm):

class Meta:
        model = Livro
        fields = ('codigo', 'titulo', 'ISBN', 'autor', 'editora', 'ano')
```

Adicione um link para o formulário

- Na página base.html adicione um link
- O if abaixo é um controle básico de segurança

```
<body>
    <div>
        <h1><a href="/livros">Lista de Livros</a></h1>
        </div>
        {% if user.is_authenticated %}
            <a href="{% url 'livro_new' %}" class="top-menu">[ + Novo ]</a>
        {% endif %}
        {% block content %}
    {% endblock %}
</body>
```

Adicione a url livro_new

```
path('livro/new', views.livro_new, name='livro_new'),
```

- Adicione a view correspondente

```
from .forms import LivroForm

...

def livro_new(request):
    form = LivroForm()
    return render(request, 'livros/livro_edit.html', {'form': form})
```

Adicione o template correspondente

```
{% extends 'livros/base.html' %}

{% block content %}
    <h1>Novo livro</h1>
    <form method="POST" class="post-form">{% csrf_token %}
        {{ form.as_p }}
        <button type="submit" class="save btn btn-default">Save</button>
    </form>
{% endblock %}
```

Fazer o formulário salvar

- Edite sua view

```
from django.shortcuts import redirect
...

def livro_new(request):
    if request.method == "POST":
        form = LivroForm(request.POST)
        if form.is_valid():
            livro = form.save(commit=False)
            #post.author = request.user
            #post.published_date = timezone.now()
            livro.save()
            return redirect('livro_detail', pk=livro.pk)
    else:
        form = LivroForm()
    return render(request, 'livros/livro_edit.html', {'form': form})
```

Permitir editar um livro

- Usaremos o mesmo form
- Em livro_detail adicione um link para editar

```
<div>
    {% if user.is_authenticated %}
    <a class="btn btn-default" href="{% url 'livro_edit' pk=livro.pk %}">[Editar]</a>
    {% endif %}

     <h2>{{ livro.titulo }}</h2>
```

Adicione a url e a view correspondente

- Em urls.py

```
path('livro/<int:pk>/edit/', views.livro_edit, name='livro_edit'),
```

- Em views.py

```
def livro_edit(request, pk):
    livro = get_object_or_404(Livro, pk=pk)
    if request.method == "POST":
        form = LivroForm(request.POST, instance=livro)
        if form.is_valid():
            livro = form.save(commit=False)
            livro.save()
            return redirect('livro_detail', pk=livro.pk)
    else:
        form = LivroForm(instance=livro)
    return render(request, 'livros/livro_edit.html', {'form': form})
```

Deletar um livro

- Novamente, é preciso criar a url, a view, e o template.
- No arquivo livro_detail.html, coloque um link para poder Apagar.

```
<div>
    {% if user.is_authenticated %}
    <a class="btn btn-default" href="{% url 'livro_edit' pk=livro.pk %}">[Editar]</a>
    <a class="btn btn-default" href="{% url 'livro_delete' pk=livro.pk %}">[Apagar]</a>
    {% endif %}
```

- Crie a url correspondente em urls.py

```
path('livro/<int:pk>/delete/', views.livro_delete, name='livro_delete'),
```

Deletar um livro: view

- No arquivo views.py, crie a view correspondente

```
def livro_delete(request, pk):
    livro = get_object_or_404(Livro, pk=pk)
    if request.method == "POST":
        livro.delete()
        return redirect('livro_list')
    else:
        form = LivroForm(instance=livro)
    return render(request, 'livros/livro_delete.html', {'livro': livro})
```

Deletar um livro: template

- Crie o arquivo livro_delete.html

```
{% extends 'livros/base.html' %}

{% block content %}
    <h1>Deletar livro</h1>
    <form method="POST" class="post-form">{% csrf_token %}
        Você tem certeza que deseja excluir esse livro?
        <p>Título: <strong>{{livro.titulo}}</strong></p>
        <button type="submit" class="save btn btn-default">Confirma Exclusão!</button>
    </form>
{% endblock %}
```

Fazendo o deploy no pythonanywhere

- Commit/push as alterações para o github
  - git add / commit / push
- No pythonanywhere, no bash
  - Entre no console e atualize o código a partir do github
    - cd ~/<your-pythonanywhere-username>.pythonanywhere.com
    - git pull
    - python manage.py collectstatic

DJango REST

Sistema de Controle de livros

https://github.com/renatoln/crud-livros-django

Definições

- API RESTful é uma interface de programação de aplicativo de transferência de estado representacional.
- Colocar dados em seu servidor da Web de uma forma acessível a outros servidores e clientes. Ele funciona por meio de requisições e respostas HTTP e rotas de URL cuidadosamente estruturadas para representar recursos específicos.

Configurando/Inicializando o Ambiente

- Instalar o DJango REST framework dentro da virtual env
  - pip install djangorestframework
- Usando o requirements.txt (documente as dependências dentro desse arquivo)
  - pip install -r requirements.txt
- Criar uma nova app (api_rest) para o serviço rest.
  - python manage.py startapp api_rest
- Adicione duas apps em cld/settings.py em

```
INSTALLED_APPS = [

    'api_rest',
    'rest_framework',
```

Adicionar a url da nova api

- Em cld/urls.py adicionar a referência para a urls.py de api_rest

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('livros/', include('livros.urls')),
    path('api_rest/', include('api_rest.urls')),
]
```

Implementar serviço para listar os livros

- Adicionar o arquivo urls.py em api_rest

```
from api_rest import views
from django.urls import path


urlpatterns = [
    path('livros/', views.LivroListServiceView.as_view(), name='livros'),
]
```

Implementar a view LivroList em api_rest/views.py

```
from livros import models
from api_rest import serializers
from rest_framework import generics

# Create your views here.

class LivroListServiceView(generics.ListCreateAPIView):
    queryset = models.Livro.objects.all()
    serializer_class = serializers.LivroSerializer
```

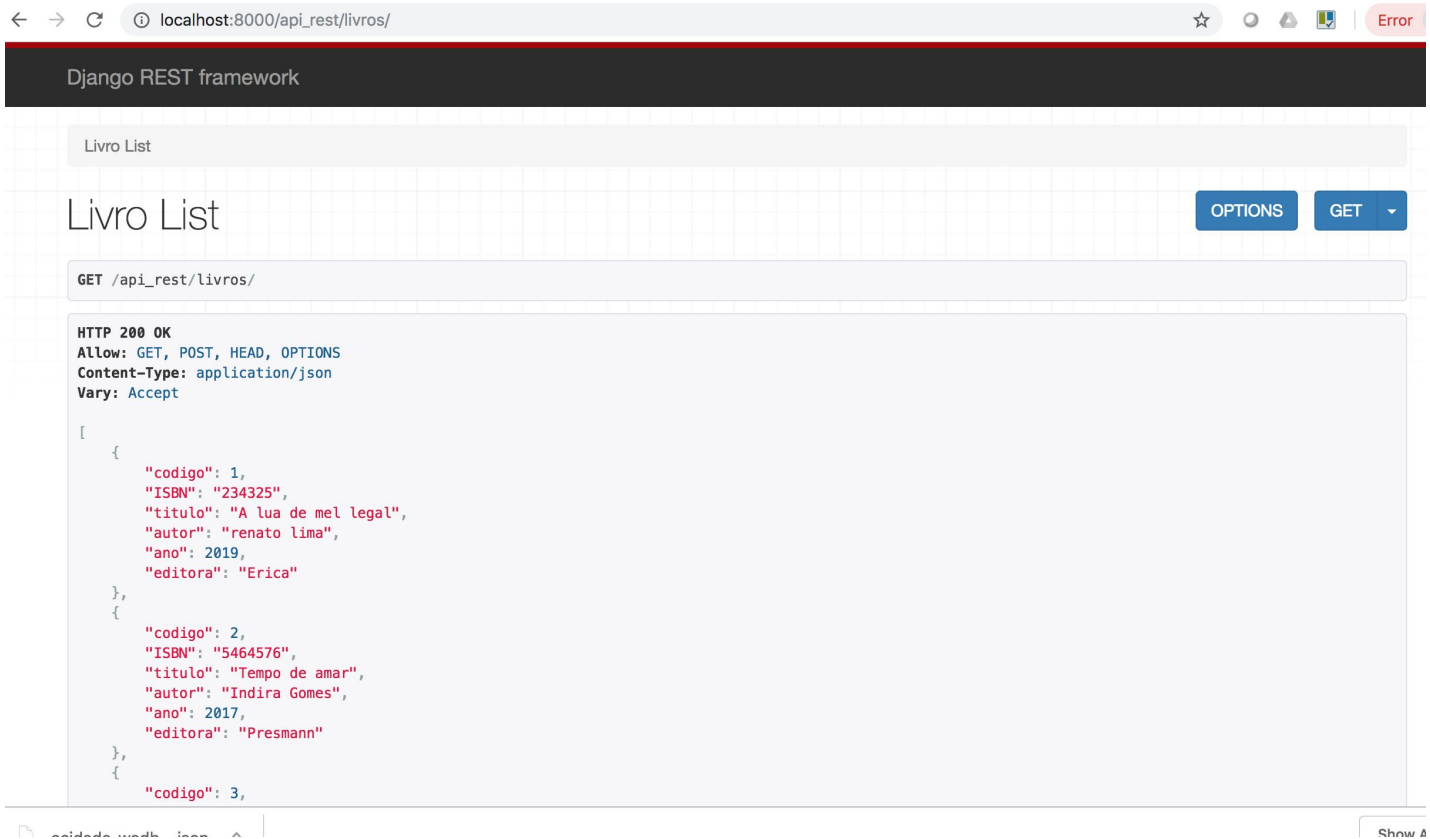
Criar o arquivo api_rest/serializers.py

- Vai ser responsável pela serialização dos dados (transforma os objetos em 'plain' textos

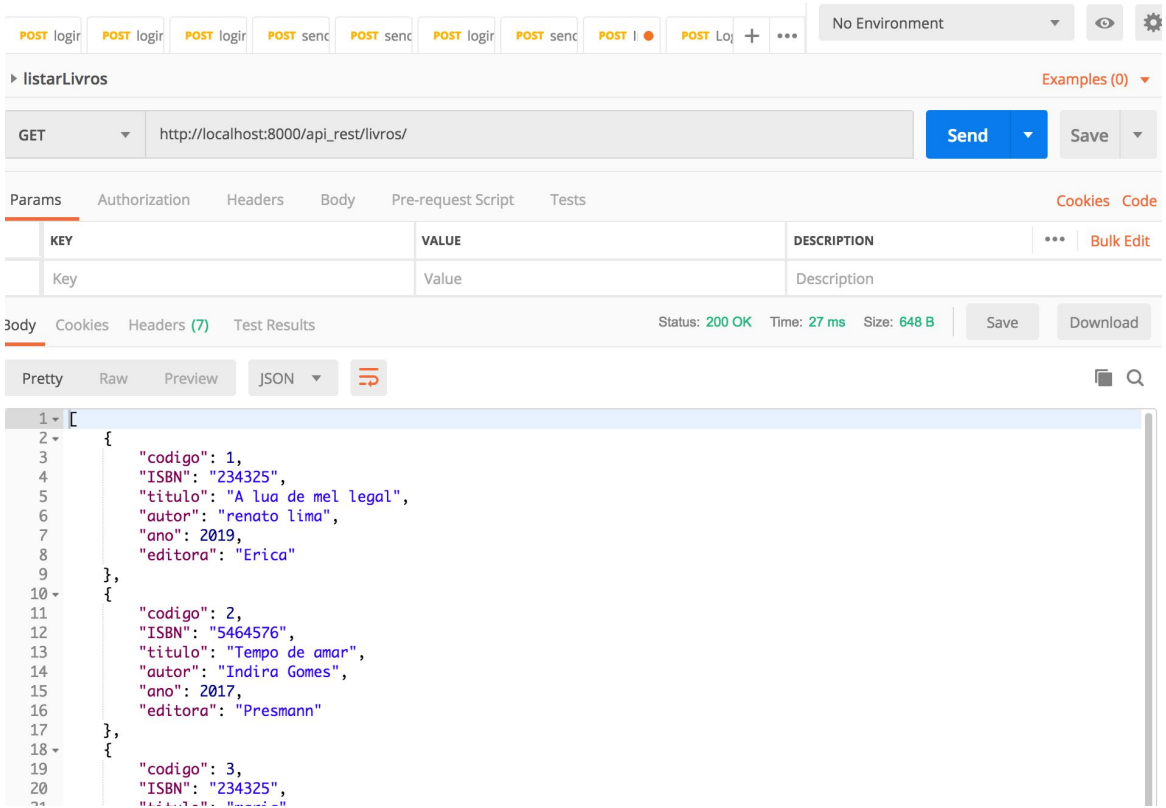
```
from rest_framework import serializers
from livros import models

class LivroSerializer(serializers.ModelSerializer):
    class Meta:
        model = models.Livro
        fields = ('codigo', 'ISBN', 'titulo', 'autor', 'ano', 'editora')
        depth = 1
```

Serviço para listar os livros



Usando o PostMan para testar a api rest



O arquivo para ser importado no postman está disponível no repositório na pasta dados/

Usando o curl para testar a api rest

```
curl -i -H "Accept: application/json" -H "Content-Type:
application/json" -X GET http://127.0.0.1:8000/api_rest/livros/
```

Implementar serviço para inserir um livro

- Criar nova url em api_rest/urls.py

```
from api_rest import views
from django.urls import path


urlpatterns = [
    path('livros/', views.LivroListServiceView.as_view(), name='livros'),
    path('cadastrarLivro/', views.CadastrarLivroServiceView.as_view(),
name='cadastrarLivro'),
]
```

Implementar a view CadastrarLivroServiceView (1/2)

```
from livros import models
from api_rest import serializers
from rest_framework import generics
from rest_framework.views import APIView
from rest_framework.response import Response

# Create your views here.

class LivroListServiceView(generics.ListCreateAPIView):
    queryset = models.Livro.objects.all()
    serializer_class = serializers.LivroSerializer
```

Implementar a view CadastrarLivroServiceView (2/2)

```
class CadastrarLivroServiceView(APIView):

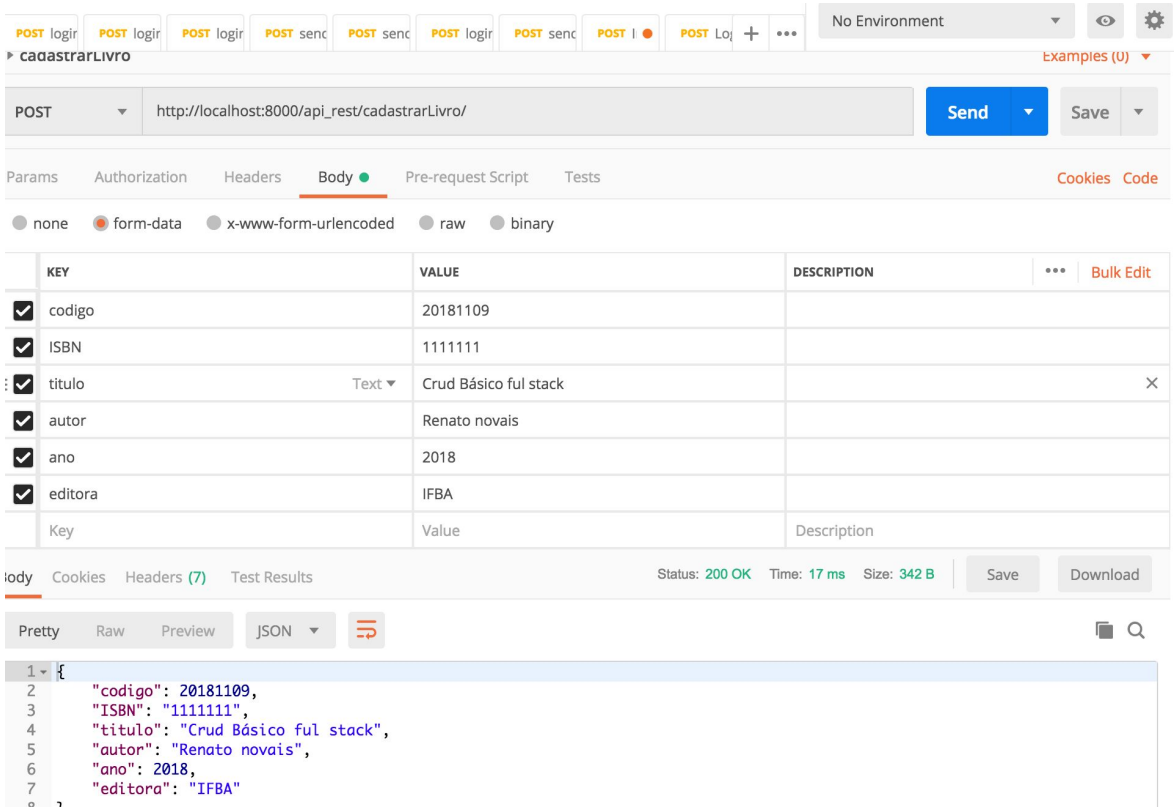
    def post(self, request, format=None):

        codigo = request.data.get('codigo')
        ISBN = request.data.get('ISBN')
        titulo = request.data.get('titulo')
        autor = request.data.get('autor')
        ano = request.data.get('ano')
        editora = request.data.get('editora')

        livro = models.Livro.objects.create(codigo = codigo, ISBN = ISBN, titulo = titulo,
                                            autor = autor, ano = ano, editora = editora)

        serializer = serializers.LivroSerializer(livro)
        return Response(serializer.data)
```

Usando o PostMan para testar a api rest



Usando o curl para testar a api rest

```
curl --header "Content-Type: application/json" \
  --request POST \
  --data '{"codigo": 19811981,"ISBN": "999999","titulo": "Using curl",
"autor": "Renato Lima","ano": 2019,"editora": "IFBA tech"}' \
  http://127.0.0.1:8000/api_rest/cadastrarLivro/
```

Implementar serviço para atualizar um livro

- Criar nova url em api_rest/urls.py

```
from api_rest import views
from django.urls import path

urlpatterns = [
    path('livros/', views.LivroListServiceView.as_view(), name='livros'),
    path('cadastrarLivro/', views.CadastrarLivroServiceView.as_view(),
name='cadastrarLivro'),
    path('atualizarLivro/', views.AtualizarLivroServiceView.as_view(),
name='atualizarLivro'),
]
```

Implementar a view AtualizarLivroServiceView

```
class AtualizarLivroServiceView(APIView):

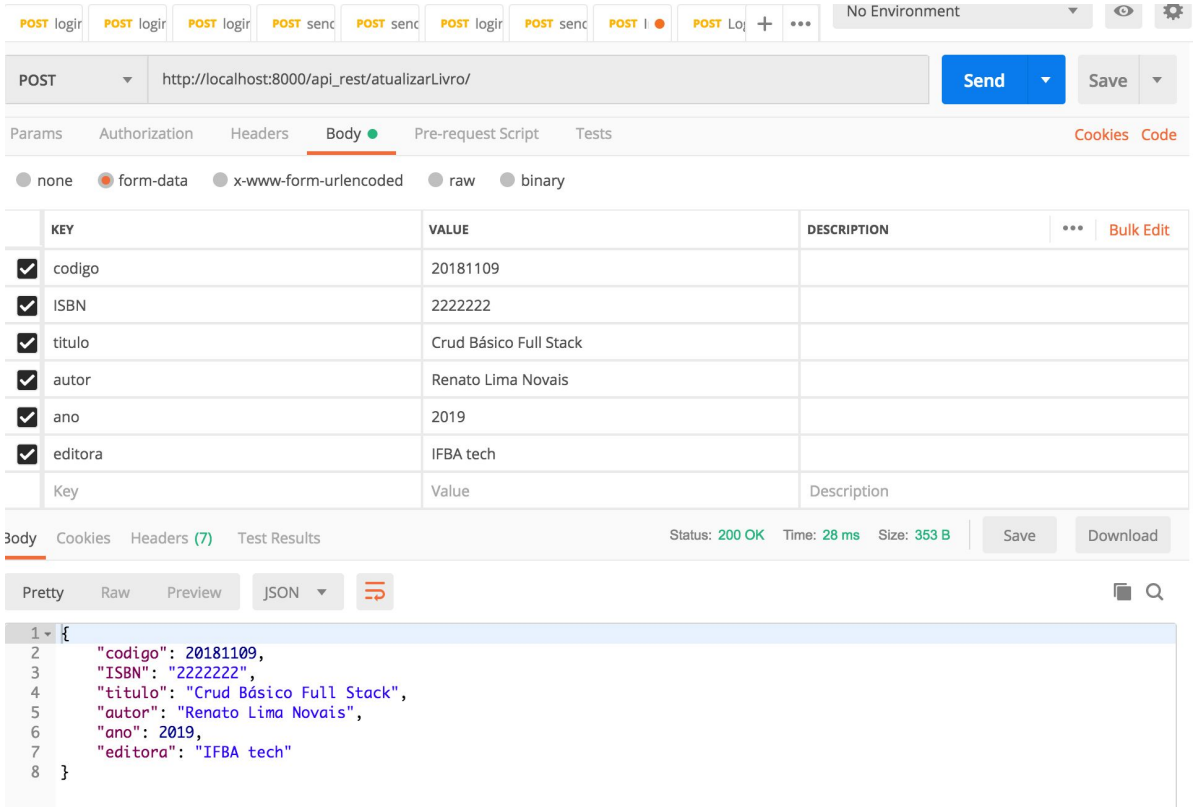
    def post(self, request, format=None):

        livro = models.Livro.objects.get(codigo=request.data.get('codigo'))
        livro.ISBN = request.data.get('ISBN')
        livro.titulo = request.data.get('titulo')
        livro.autor = request.data.get('autor')
        livro.ano = request.data.get('ano')
        livro.editora = request.data.get('editora')

        livro.save()

        serializer = serializers.LivroSerializer(livro)
        return Response(serializer.data)
```

Usando o PostMan para testar a api rest



Usando o curl para testar a api rest

```
curl --header "Content-Type: application/json" \
  --request POST \
  --data '{"codigo": 19811981,"ISBN": "88888","titulo": "Using Curl
para atualizar",    "autor": "Jener Novais","ano": 2018,"editora": "IFBA
Tech Enterprise Edition"}' \
  http://127.0.0.1:8000/api_rest/atualizarLivro/
```

Implementar serviço para excluir um livro

- Criar nova url em api_rest/urls.py

```
from api_rest import views
from django.urls import path

urlpatterns = [
    path('livros/', views.LivroListServiceView.as_view(), name='livros'),
    path('cadastrarLivro/', views.CadastrarLivroServiceView.as_view(),
name='cadastrarLivro'),
    path('atualizarLivro/', views.AtualizarLivroServiceView.as_view(),
name='atualizarLivro'),
]
```

Implementar a view ExcluirLivroServiceView

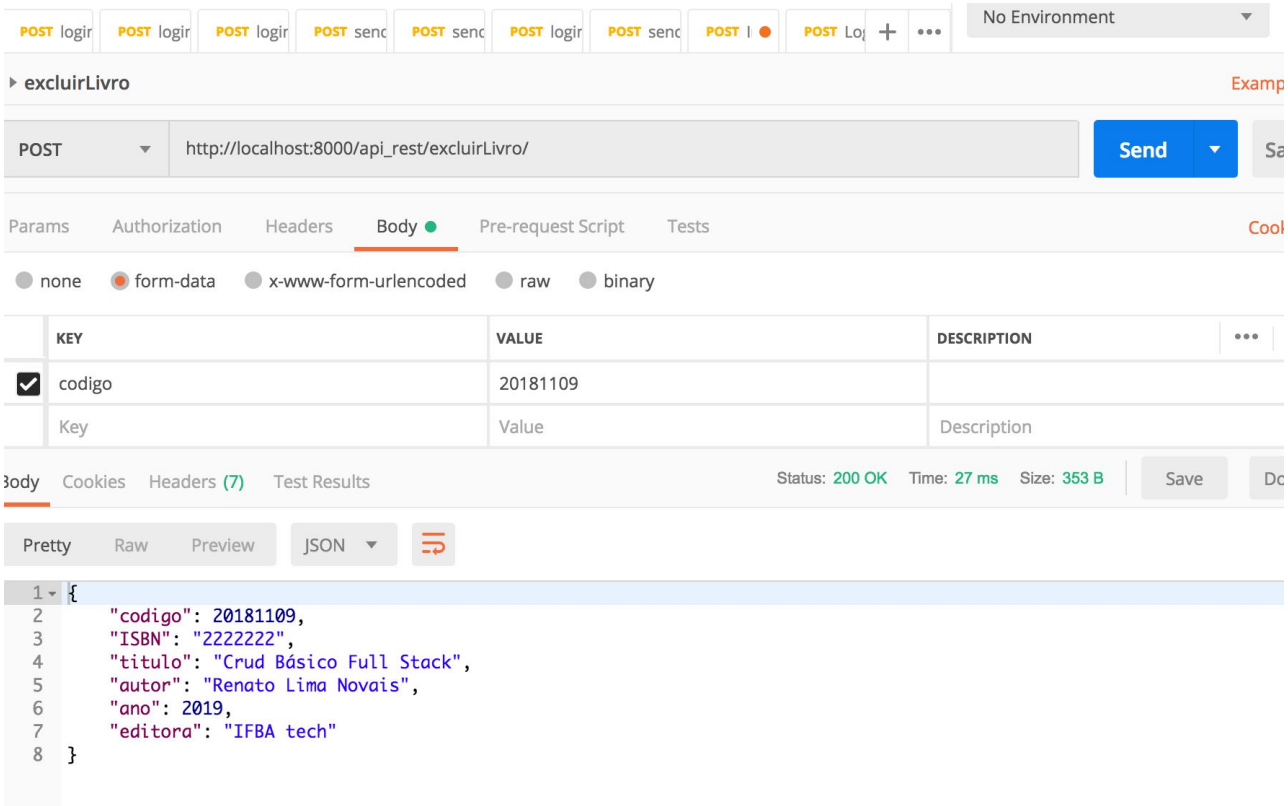
```
class ExcluirLivroServiceView(APIView):

    def post(self, request, format=None):

        livro = models.Livro.objects.get(codigo=request.data.get('codigo'))
        livro.delete()

        serializer = serializers.LivroSerializer(livro)
        return Response(serializer.data)
```

Usando o PostMan para testar a api rest



Usando o curl para testar a api rest

```
curl --header "Content-Type: application/json" \
  --request POST \
  --data '{"codigo": 19811981}' \
  http://127.0.0.1:8000/api_rest/excluirLivro/
```

Referências

- https://tutorial.djangogirls.org/pt/
- https://appdividend.com/2018/03/29/django-crud-tutorial-example/
- https://www.django-rest-framework.org
- https://codeburst.io/building-an-api-with-django-rest-framework-and-class-based-views-75b369b30396

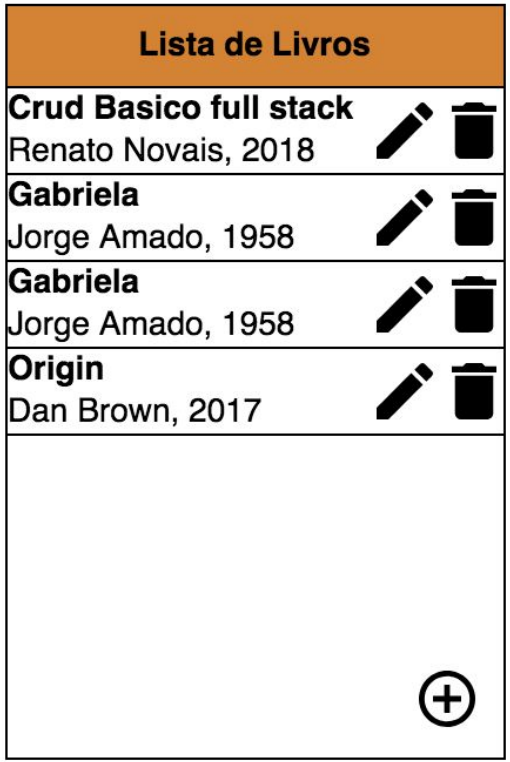
DIA 6

Android

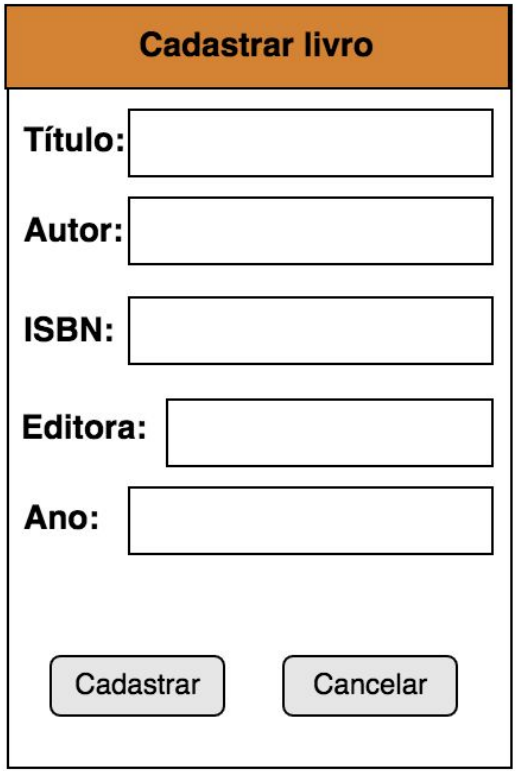
Sistema de Controle de livros
CRUD

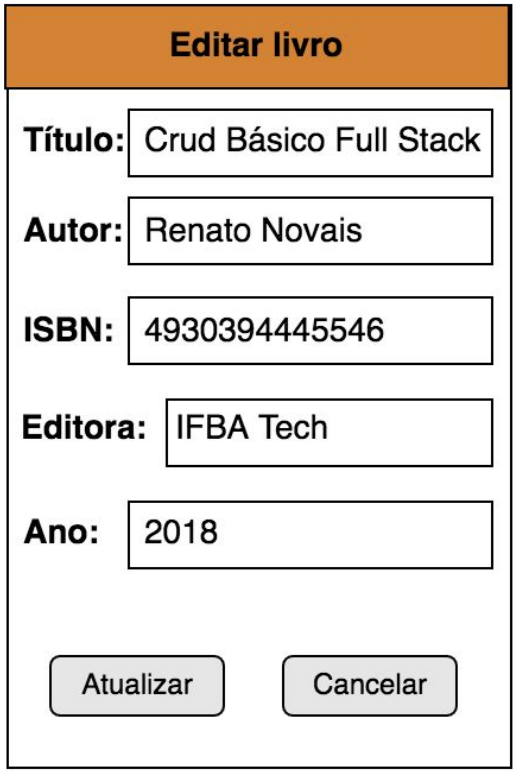
https://github.com/renatoln/crud-livros-mobile

Projeto de telas (1/2)



Projeto de telas (2/2)





Agenda

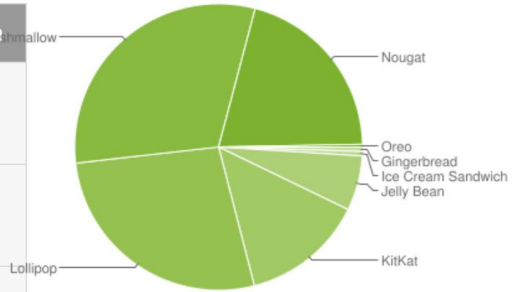
- Introdução
- Principais componentes (Arquitetura)
- Exemplos
  - Hello world
  - 2 activities
  - ListView
  - Restfull

Introdução

- Sistema operacional mobile, Kernel baseado em Linux
- Comprado pela Google (=~ $50 mi), da Android Inc, em 2005. Anunciado em 2007, lançado em 2008
- Rodava com Máquina virtual Java própria: Dalvik VM
  - Quase tudo compatível com Java
  - Descontinuada a partir de 2015, quando chega o Android Runtime (ART)

Versões

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.5%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.5%
4.1.x	Jelly Bean	16	2.2%
4.2.x		17	3.1%
4.3		18	0.9%
4.4	KitKat	19	13.8%
5.0	Lollipop	21	6.4%
5.1		22	20.8%
6.0	Marshmallow	23	30.9%
7.0	Nougat	24	17.6%
7.1		25	3.0%
8.0	Oreo	26	0.3%



Recomendado suportar por volta de 90% dos aparelhos

https://developer.android.com/about/dashboards/index.html

Ambiente de Desenvolvimento

Secure | https://developer.android.com/studio/index.html

Android Studio | FEATURES | USER GUIDE | PREVIEW | Search

← Back to Developers

- DOWNLOAD
- FEATURES
- USER GUIDE
- PREVIEW

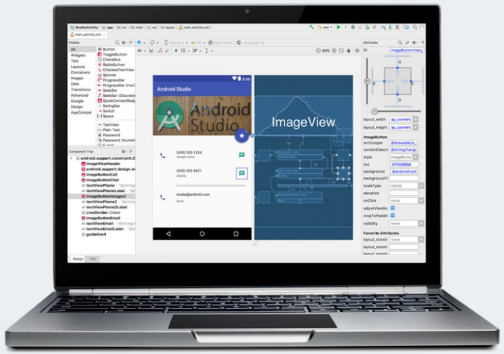
Android Studio

The Official IDE for Android

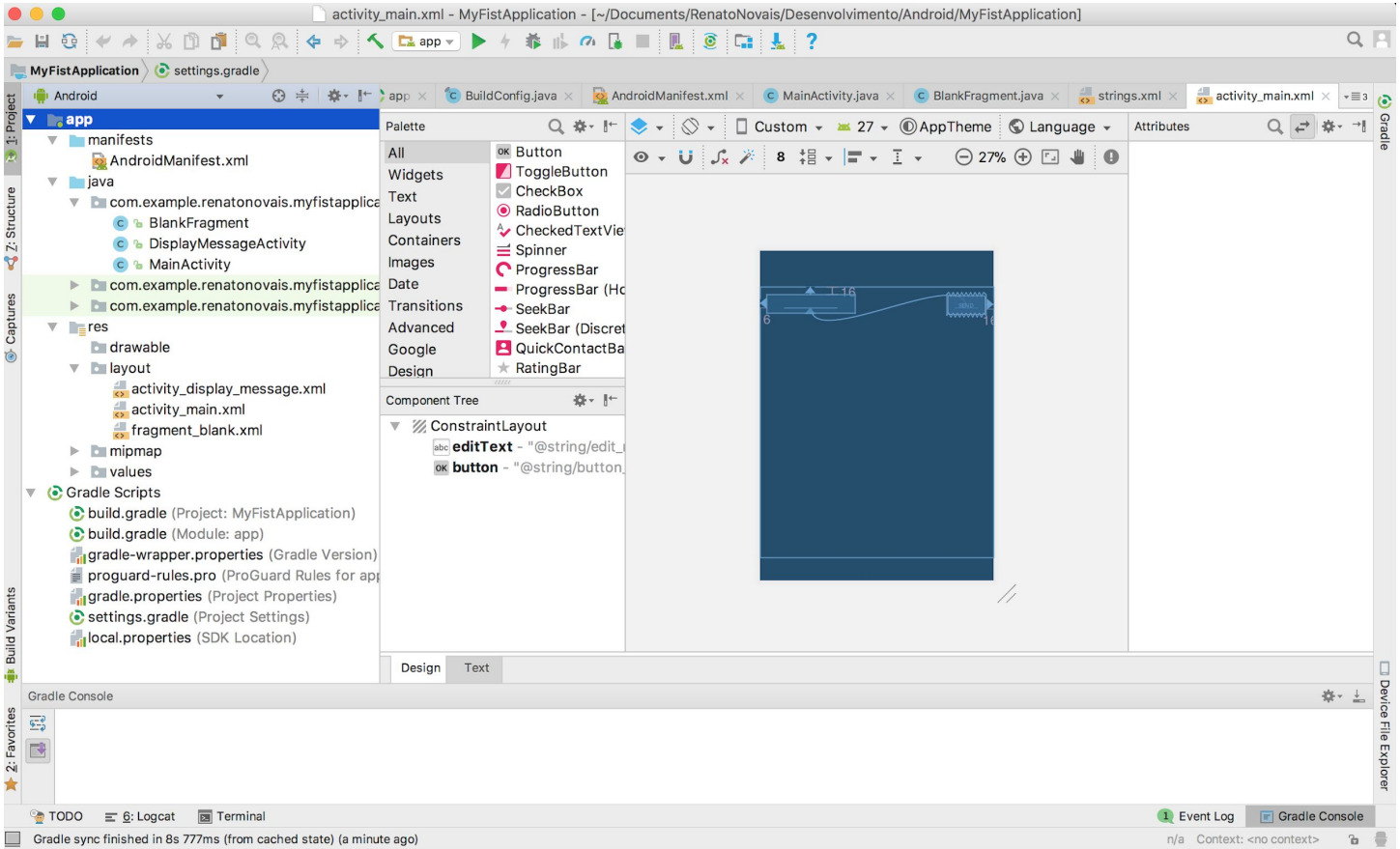
Android Studio provides the fastest tools for building apps on every type of Android device.

World-class code editing, debugging, performance tooling, a flexible build system, and an instant build/deploy system all allow you to focus on building unique and high quality apps.

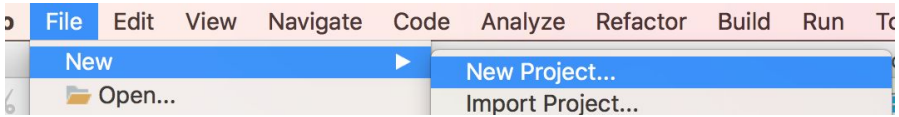
DOWNLOAD ANDROID STUDIO
3.0.1 FOR MAC (738 MB)

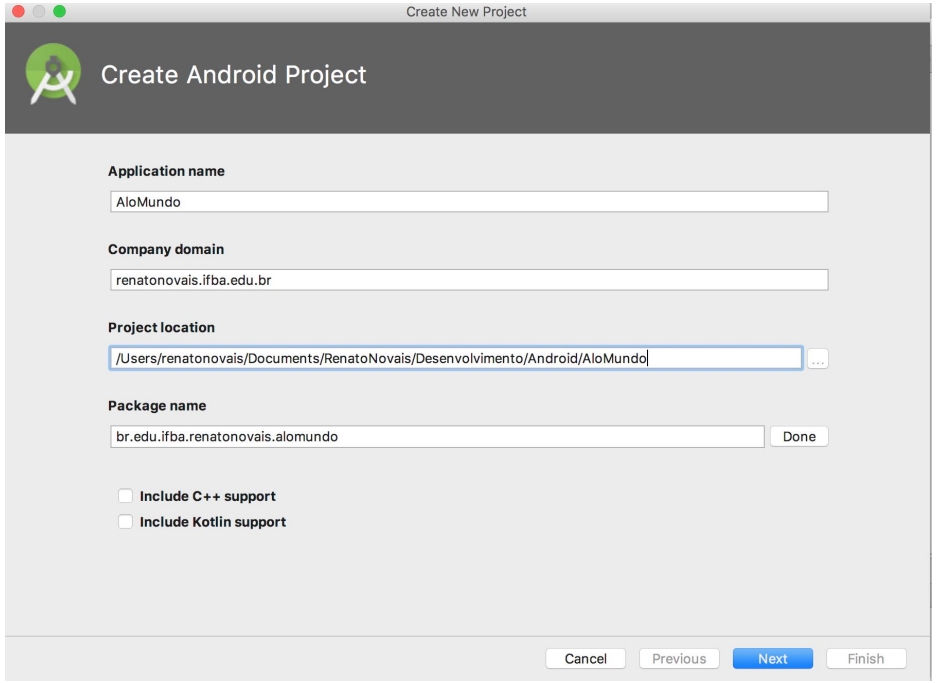


Ambiente de Desenvolvimento

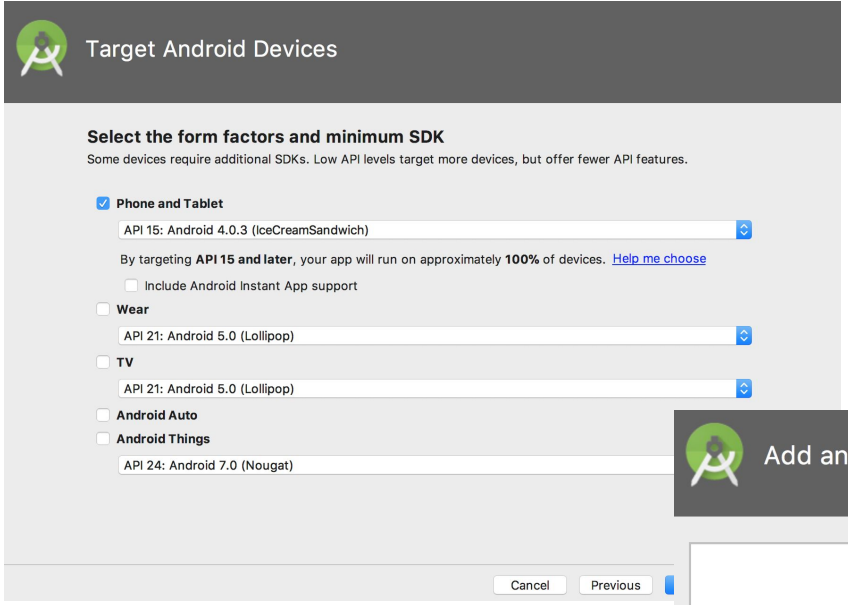


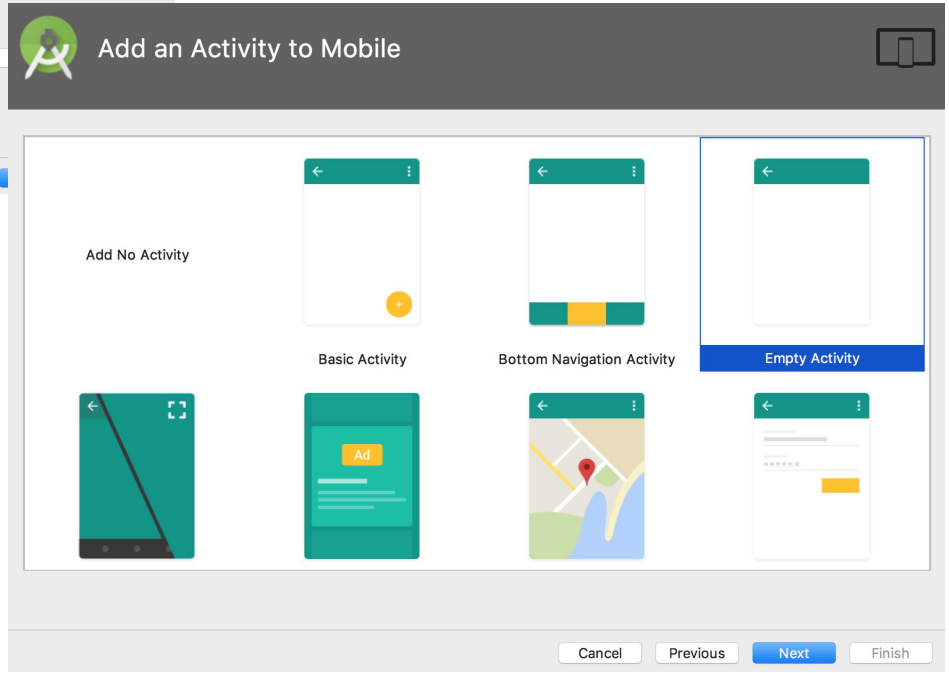
Prática: Alô Mundo!



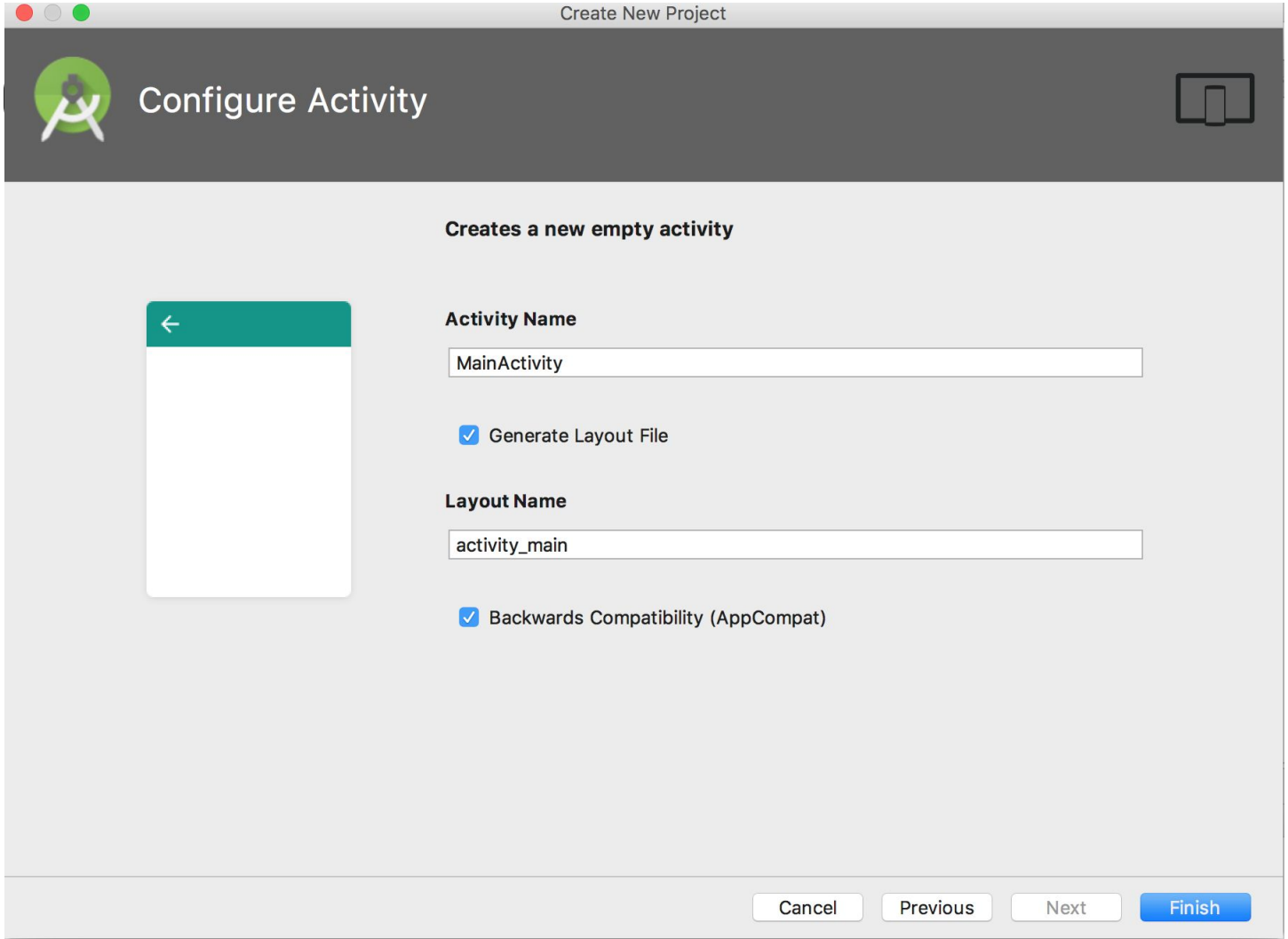


Pr

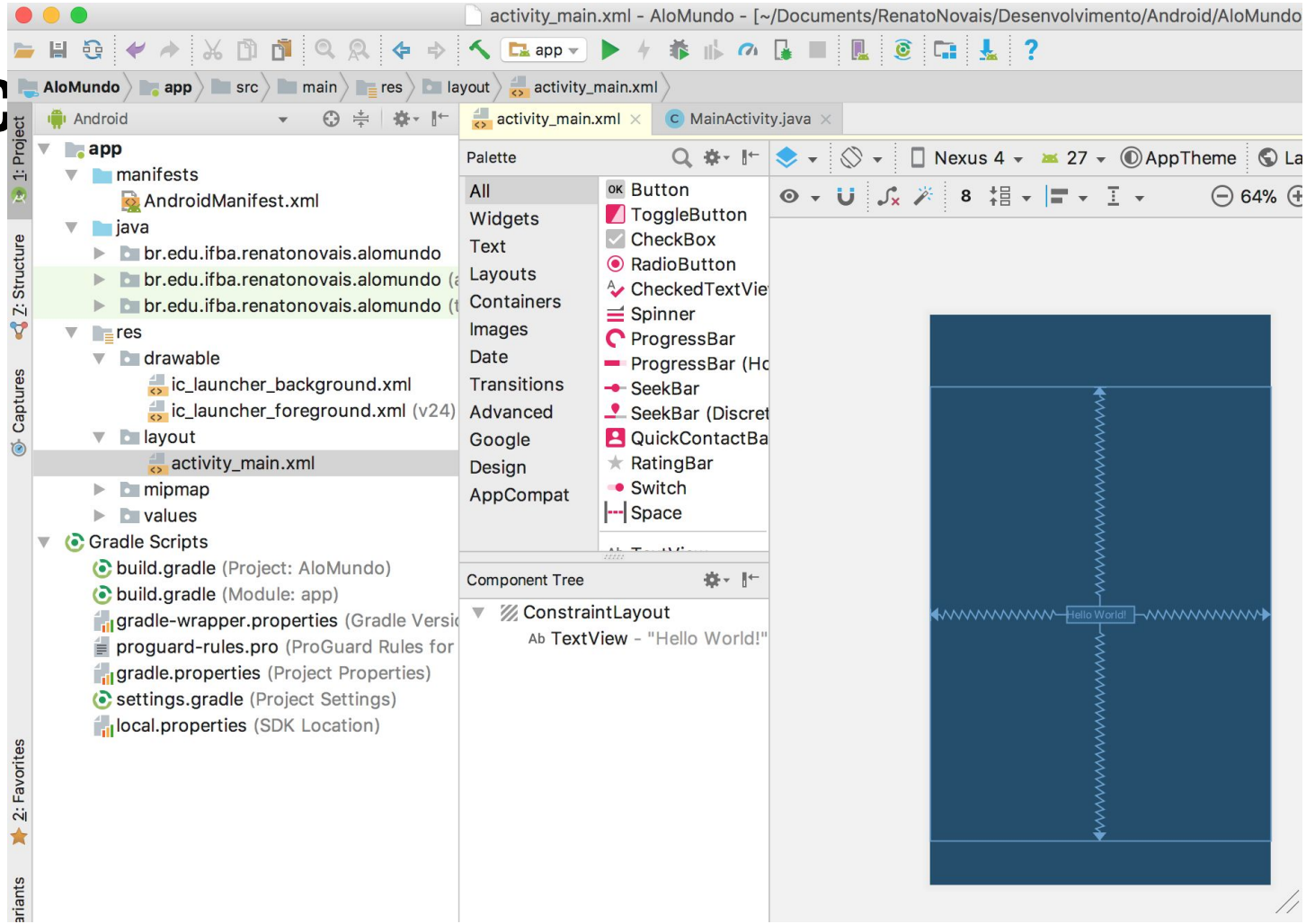

Target Android Devices
Select the form factors and minimum SDK
Some devices require additional SDKs. Low API levels target more devices, but offer fewer API features.
Phone and Tablet
API 15: Android 4.0.3 (IceCreamSandwich)
By targeting API 15 and later, your app will run on approximately 100% of devices. Help me choose
Include Android Instant App support
Wear
API 21: Android 5.0 (Lollipop)
TV
API 21: Android 5.0 (Lollipop)
Android Auto
Android Things
API 24: Android 7.0 (Nougat)
Cancel
Previous


Add an Activity to Mobile
Add No Activity
Basic Activity
Bottom Navigation Activity
Empty Activity
Ad
Cancel
Previous
Next
Finish

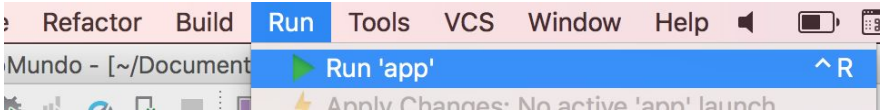
Prática: Al

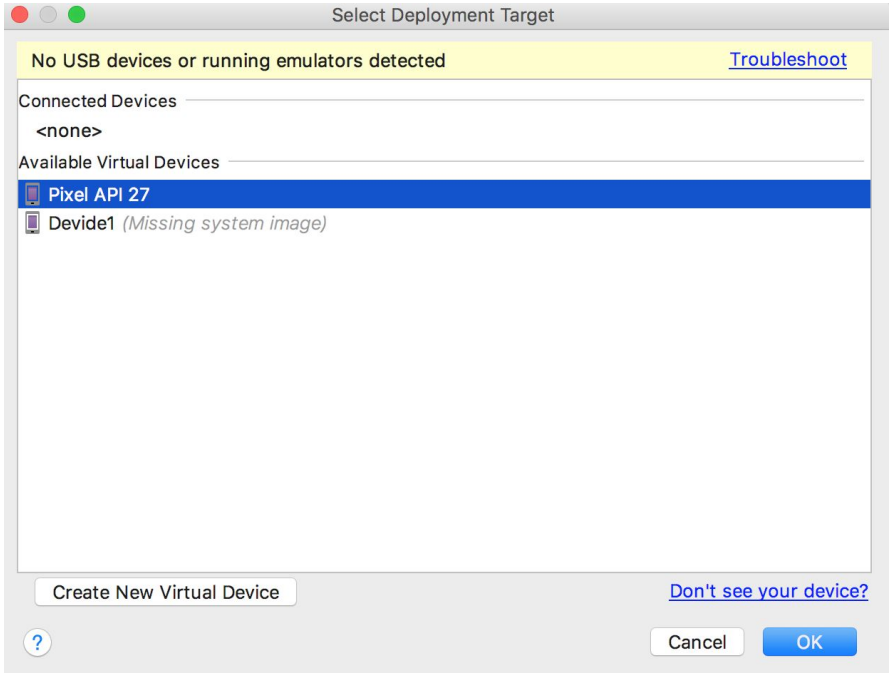

Create New Project
Configure Activity
Creates a new empty activity
Activity Name
MainActivity
Generate Layout File
Layout Name
activity_main
Backwards Compatibility (AppCompat)
Cancel
Previous
Next
Finish

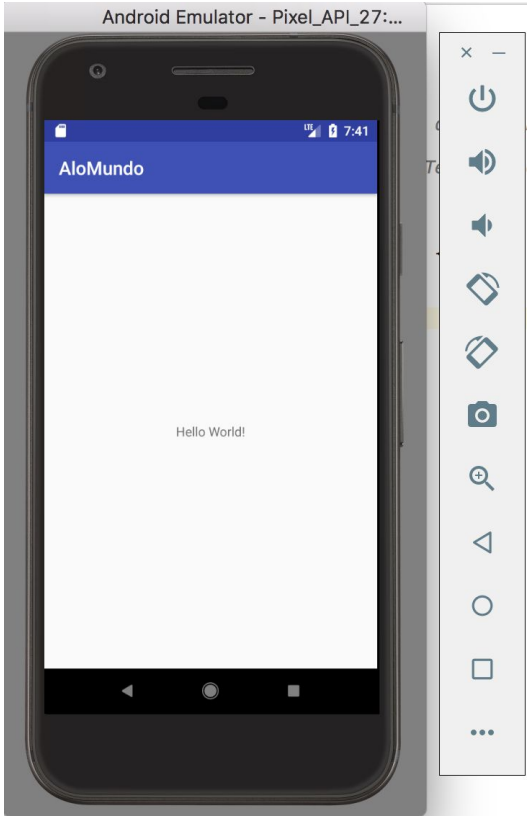
Prátic


activity_main.xml - AloMundo - [~/Documents/RenatoNovais/Desenvolvimento/Android/AloMundo
app
AloMundo
app
src
main
res
layout
activity_main.xml
Android
app
manifests
AndroidManifest.xml
java
br.edu.ifba.renatonovais.alomundo
br.edu.ifba.renatonovais.alomundo
br.edu.ifba.renatonovais.alomundo
res
drawable
ic_launcher_background.xml
ic_launcher_foreground.xml (v24)
layout
activity_main.xml
mipmap
values
Gradle Scripts
build.gradle (Project: AloMundo)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle Versi
proguard-rules.pro (ProGuard Rules for
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)
activity_main.xml
MainActivity.java
Palette
All
Widgets
Text
Layouts
Containers
Images
Date
Transitions
Advanced
Google
Design
AppCompat
Button
ToggleButton
CheckBox
RadioButton
CheckedTextVie
Spinner
ProgressBar
ProgressBar (Hc
SeekBar
SeekBar (Discret
QuickContactBa
RatingBar
Switch
Space
Component Tree
ConstraintLayout
TextView - "Hello World!"
Nexus 4
27
AppTheme
64%
Hello World!
1: Project
7: Structure
Captures
2: Favorites

Prática: Alô Mundo!

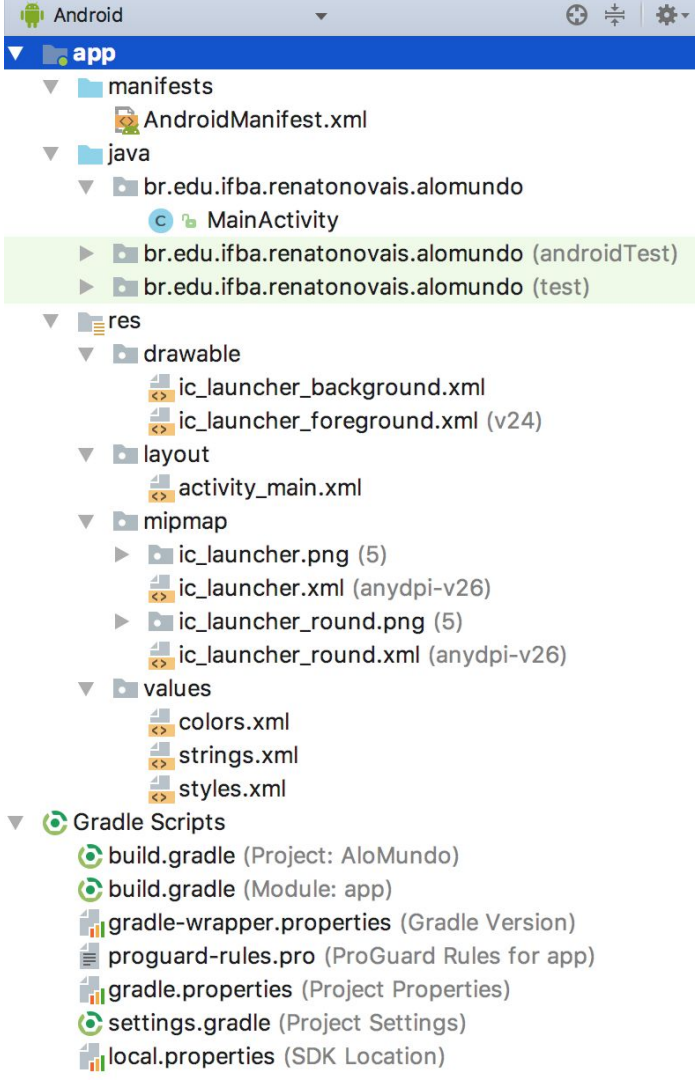






Estrutura de Arquivos

- **AndroidManifest.xml** – configurações gerais do app (nome, versão, permissões, etc.)
- **MainActivity.java** – classe inicial. Comportamento da Tela
- **res/** – pasta com recursos do app (telas, layouts, strings, ids, icones, etc.)
- **Gradle scripts** – scripts para build do app



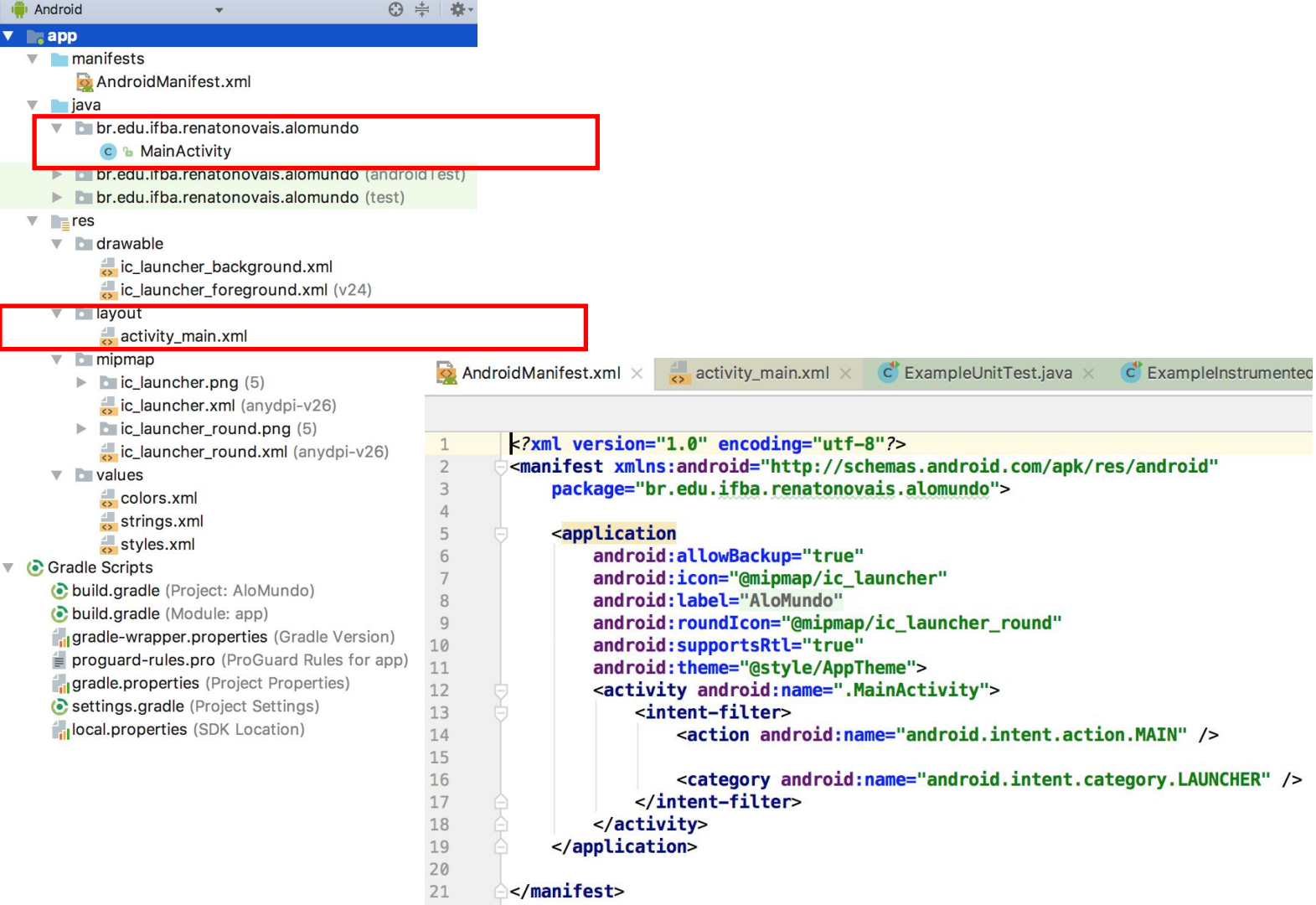
Principais Componentes

- Activity
- Services
- Broadcast receivers
- Content Providers

https://developer.android.com/guide/components/fundamentals.html

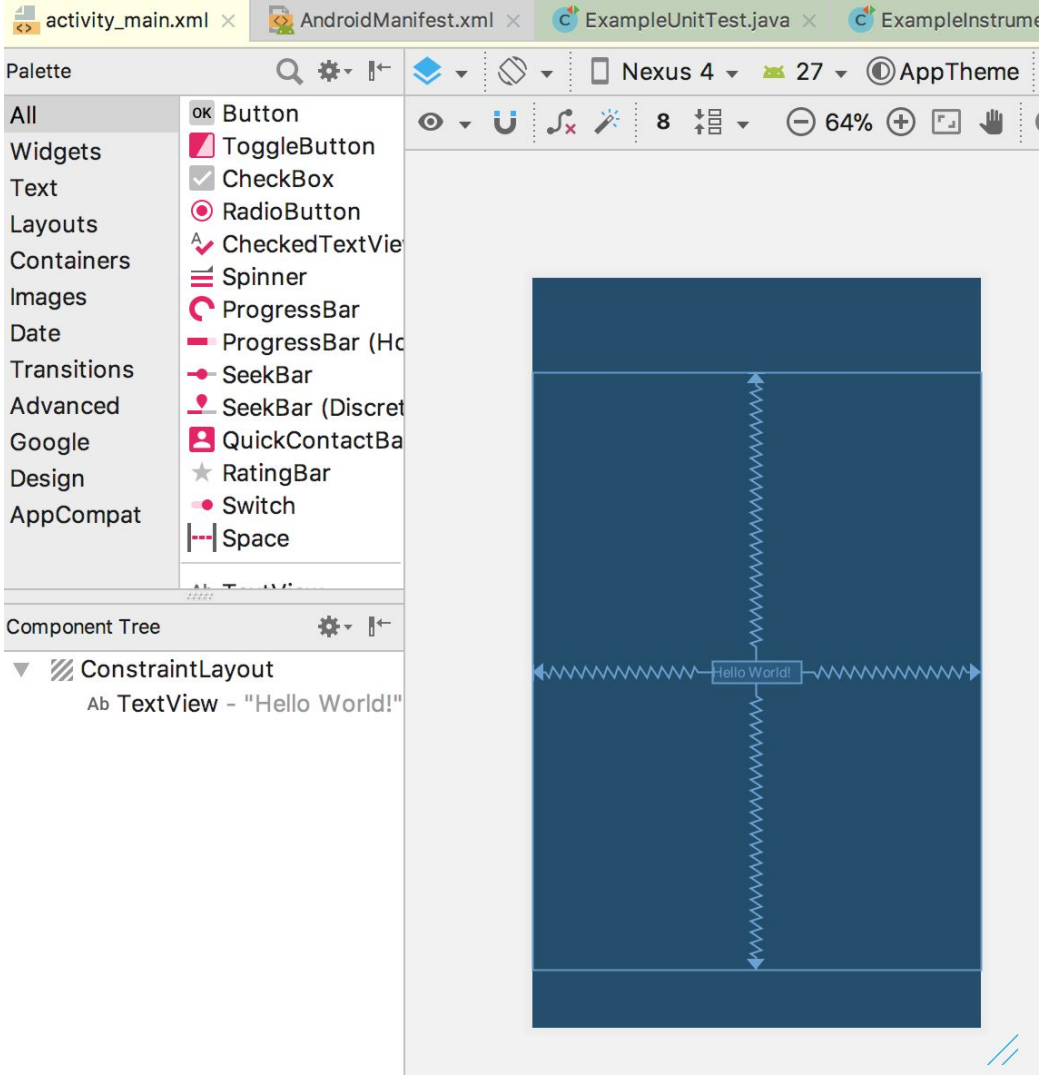
Activity

- Um ponto de entrada para interagir com o usuário
- Uma activity *main* é utilizada para iniciar a app
- Composta por dois arquivos principais
  - src/.../**NomeActivity**.java (o nome pode ser outro) -> define o comportamento
  - res/layout/**activity_nome.xml** -> define o layout
- Aparece listada no AndroidManifest.xml


Android
app
manifests
AndroidManifest.xml
java
br.edu.ifba.renatonovais.alomundo
MainActivity
br.edu.ifba.renatonovais.alomundo (androidTest)
br.edu.ifba.renatonovais.alomundo (test)
res
drawable
ic_launcher_background.xml
ic_launcher_foreground.xml (v24)
layout
activity_main.xml
mipmap
ic_launcher.png (5)
ic_launcher.xml (anydpi-v26)
ic_launcher_round.png (5)
ic_launcher_round.xml (anydpi-v26)
values
colors.xml
strings.xml
styles.xml
Gradle Scripts
build.gradle (Project: AloMundo)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle Version)
proguard-rules.pro (ProGuard Rules for app)
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)
AndroidManifest.xml
activity_main.xml
ExampleUnitTest.java
ExampleInstrumented
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3 package="br.edu.ifba.renatonovais.alomundo">
4
5 <application
6 android:allowBackup="true"
7 android:icon="@mipmap/ic_launcher"
8 android:label="AloMundo"
9 android:roundIcon="@mipmap/ic_launcher_round"
10 android:supportsRtl="true"
11 android:theme="@style/AppTheme">
12 <activity android:name=".MainActivity">
13 <intent-filter>
14 <action android:name="android.intent.action.MAIN" />
15
16 <category android:name="android.intent.category.LAUNCHER" />
17 </intent-filter>
18 </activity>
19 </application>
20
21 </manifest>

Layout

- Design



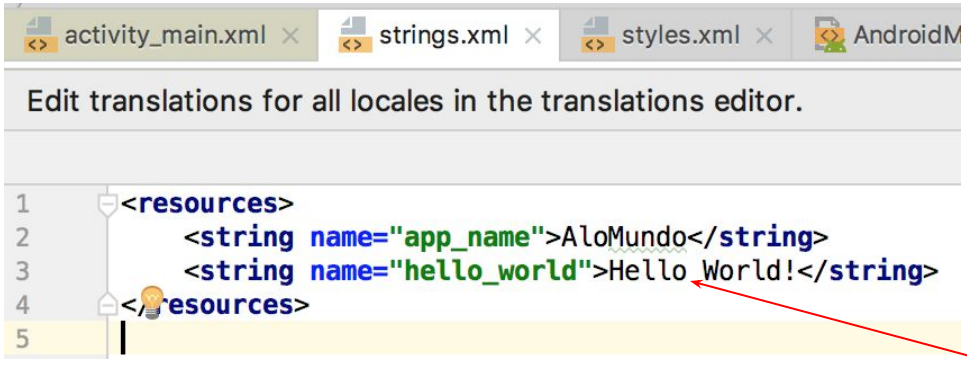
Layout

- Text (xml)

activity_main.xml | strings.xml | styles.xml | AndroidManifest.xml | ExampleUnitTest.java

```xml
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="br.edu.ifba.renatonovais.alomundo.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

string.xml



- Utilizado para traduzir o *app* para vários idiomas (um arquivo string.xml por idioma)

Código Java

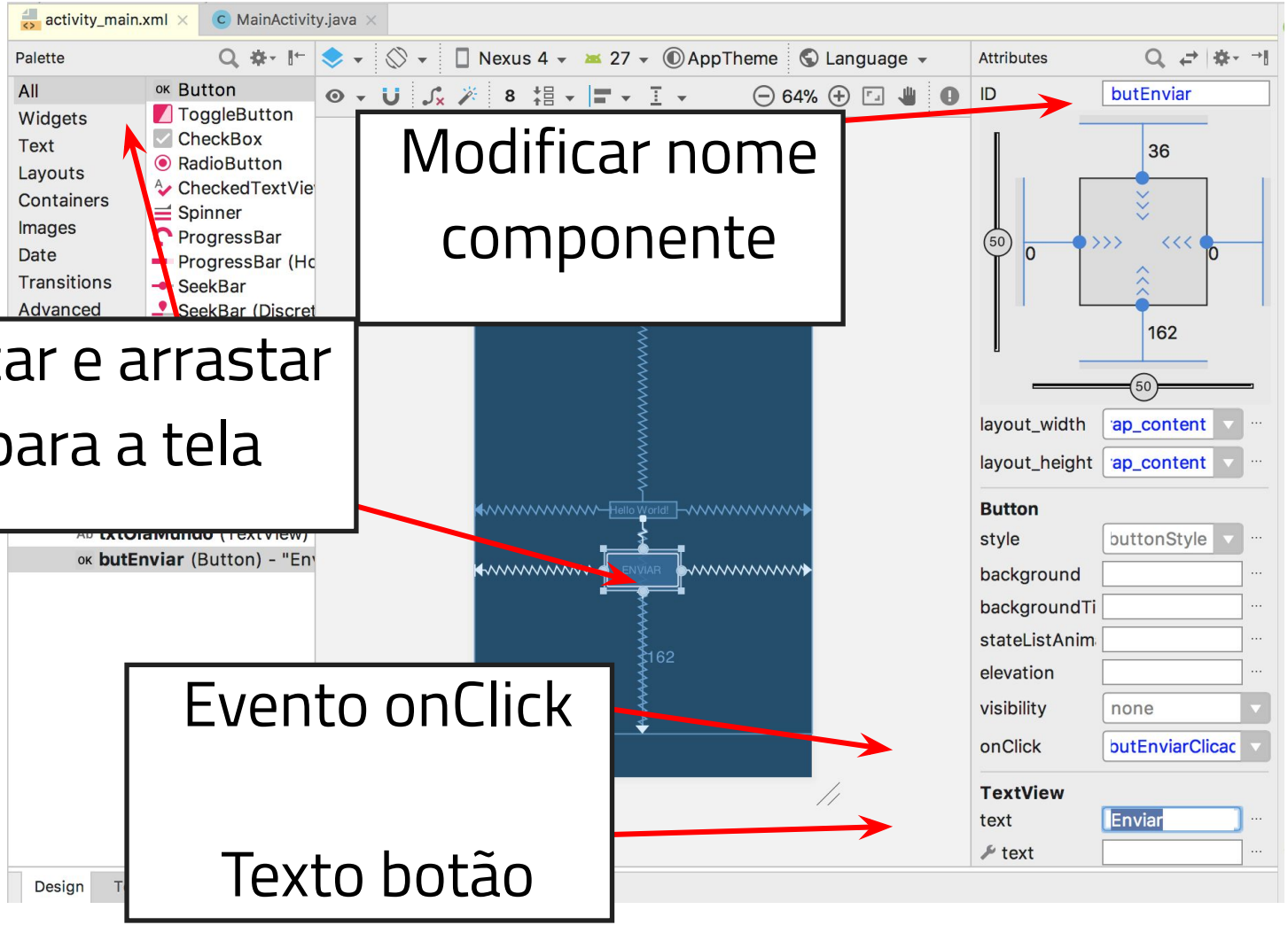
MainActivity.java | styles.xml | AndroidManifest.xml | Exar

```
1  package br.edu.ifba.renatonovais.alomundo;
2
3  import android.support.v7.app.AppCompatActivity;
4  import android.os.Bundle;
5
6  public class MainActivity extends AppCompatActivity {
7
8      @Override
9      protected void onCreate(Bundle savedInstanceState) {
10         super.onCreate(savedInstanceState);
11         setContentView(R.layout.activity_main);
12     }
13 }
14
```

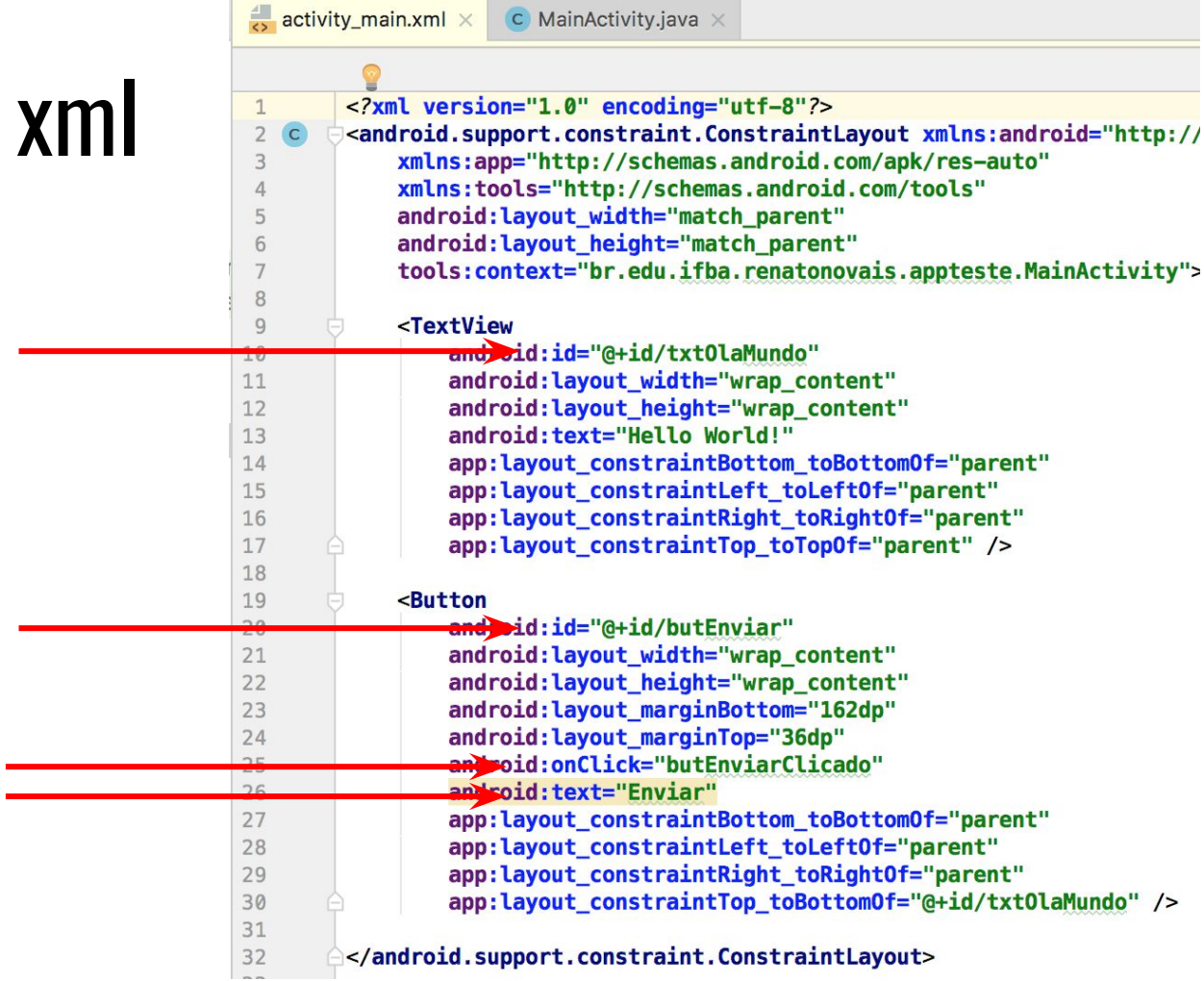
- Classe **R** utilizada para acessar os *resources* contidos em res/

Activity: interações, mais componentes

- Textview + button
  - Objetivo: trocar o texto do TextView ao clicar no botão
  - Passos
    - Adicionar um botão
    - Implementar o método que faz a troca do texto quando o botão é clicado


activity_main.xml
MainActivity.java
Palette
All
Widgets
Text
Layouts
Containers
Images
Date
Transitions
Advanced
Button
ToggleButton
CheckBox
RadioButton
Spinner
ProgressBar
SeekBar
Nexus 4
27
AppTheme
Language
64%
Attributes
ID
butEnviar
Modificar nome componente
Clicar e arrastar para a tela
Hello World!
ENVIAR
162
36
50
0
layout_width
layout_height
Button
style
background
visibility
none
onClick
butEnviarClicac
TextView
text
Enviar
Evento onClick
Texto botão
Design

Layout xml


activity_main.xml
MainActivity.java
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context="br.edu.ifba.renatonovais.appteste.MainActivity">
<TextView
android:id="@+id/txtOlaMundo"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Hello World!"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toTopOf="parent" />
<Button
android:id="@+id/butEnviar"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginBottom="162dp"
android:layout_marginTop="36dp"
android:onClick="butEnviarClicado"
android:text="Enviar"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toBottomOf="@+id/txtOlaMundo" />
</android.support.constraint.ConstraintLayout>

MainActivity: implementação evento

MainActivity

```
package br.edu.ifba.renatonovais.appteste;

import ...

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void butEnviarClicado(View view){
        TextView txtviewOla = (TextView)findViewById(R.id.txtOlaMundo);
        String txtOla = (String) txtviewOla.getText();
        String novoTexto = "Olá Mundo!";
        if (txtOla.equals(novoTexto)){
            novoTexto = "Hello World!";
        }
        txtviewOla.setText(novoTexto);

    }
}
```

App resultante

Ciclos de vida de um Activity

- ToDo

2 Activities: comunicação entre elas

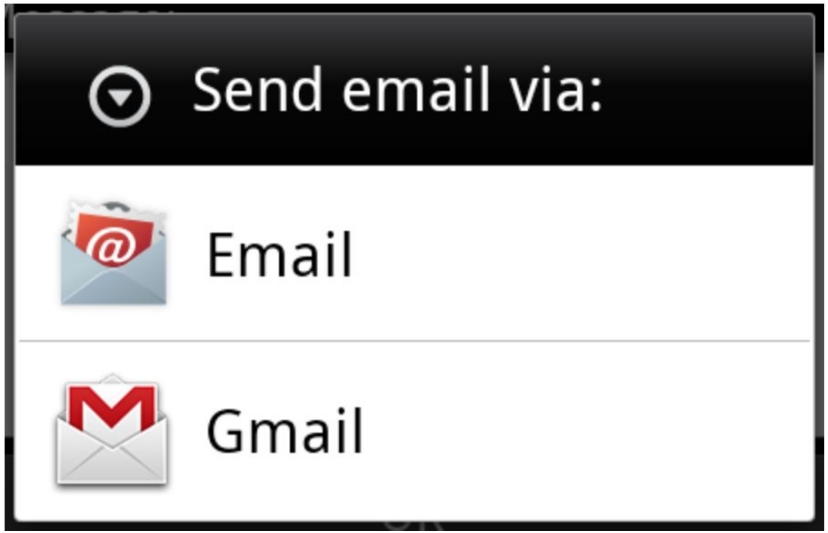
- Objetivo: passar a informação de uma tela para outra tela
  - Passos
    - Criar novo activitiy (2), com um TextView
    - Adicionar um TextField no activity 1, e ao clicar, enviar o texto para a activity 2
    - Implementar o método que faz tal operação

Intent

- é uma descrição abstrata de uma operação a ser executada.
  - Ele pode ser usado com startActivity para iniciar uma atividade,
  - broadcastIntent para enviá-lo para qualquer componente BroadcastReceiver interessado, e
  - startService (Intent) ou bindService (Intent, ServiceConnection, int) para se comunicar com um serviço de background.

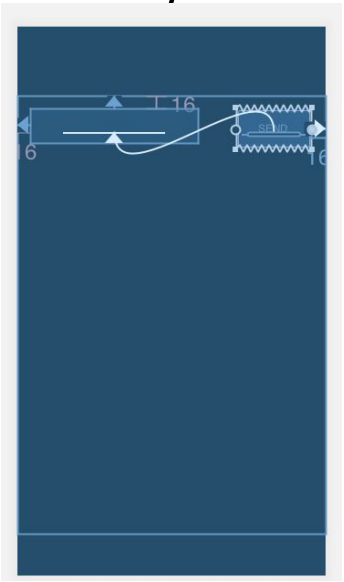
Intent: enviar email

```
Intent email = new Intent(Intent.ACTION_SEND, Uri.parse("mailto:"));
email.putExtra(Intent.EXTRA_EMAIL, recipients);
email.putExtra(Intent.EXTRA_SUBJECT, subject.getText().toString());
email.putExtra(Intent.EXTRA_TEXT, body.getText().toString());
startActivity(Intent.createChooser(email, "Choose an email client from..."))
```

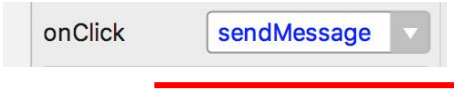


Intent: comunicando entre 2 activities

Activity_main.xml



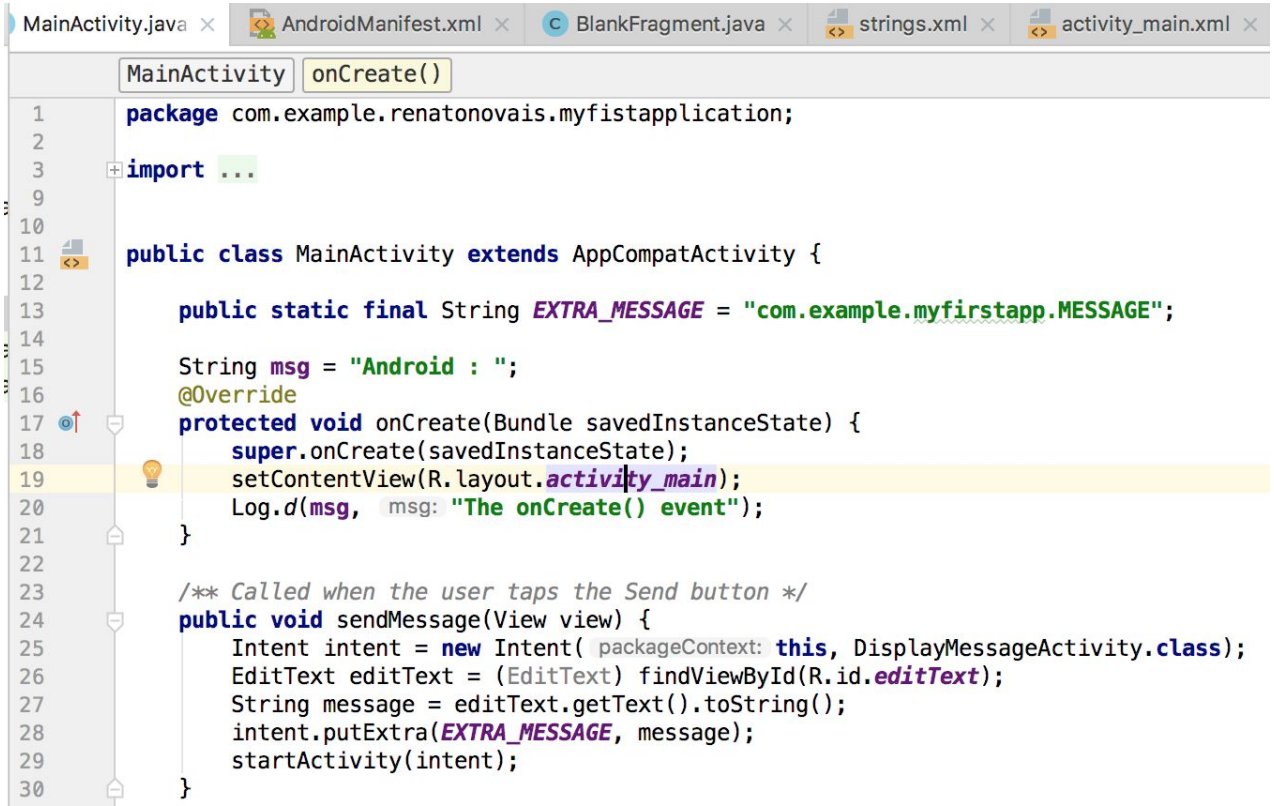
MainActivity.java



Activity_display_message.xml

DisplayMessageActivity.java

MainActivity.java

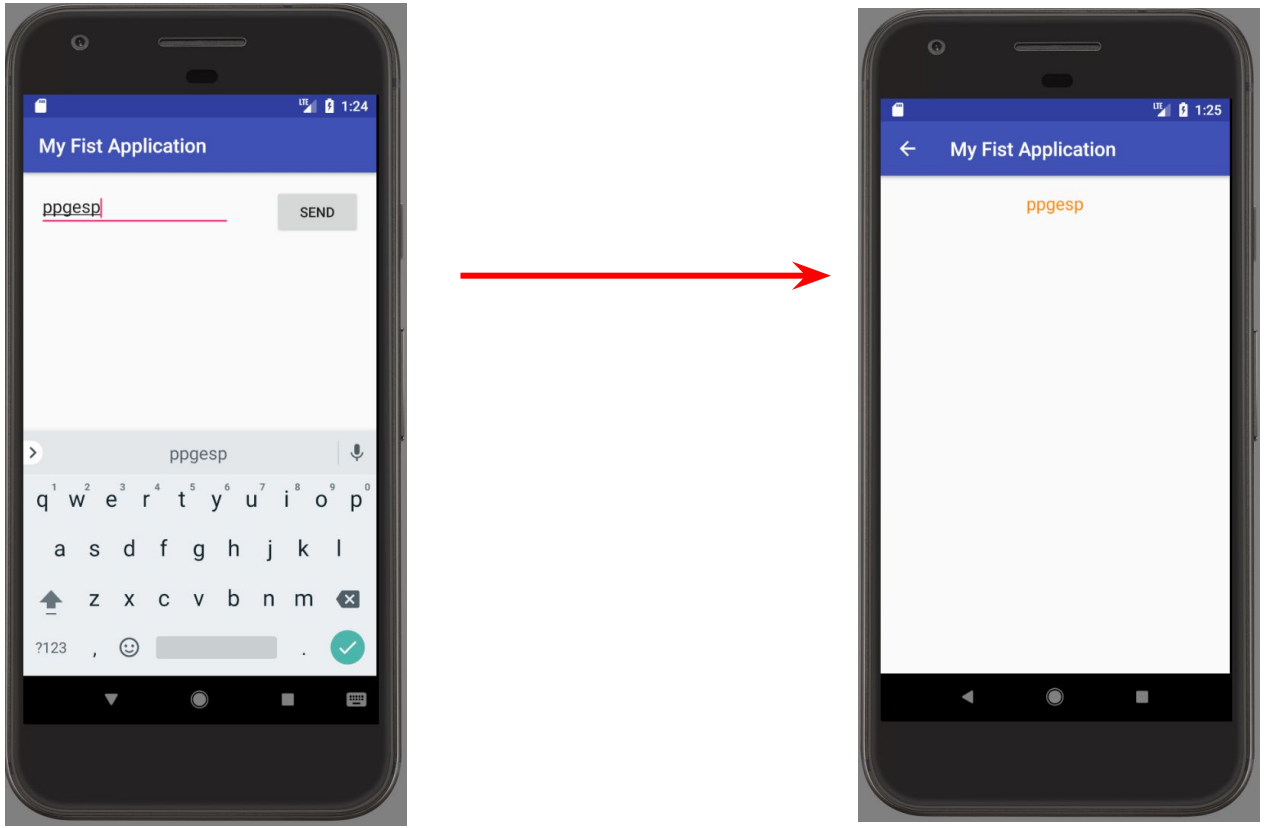


DisplayMessageMain.java

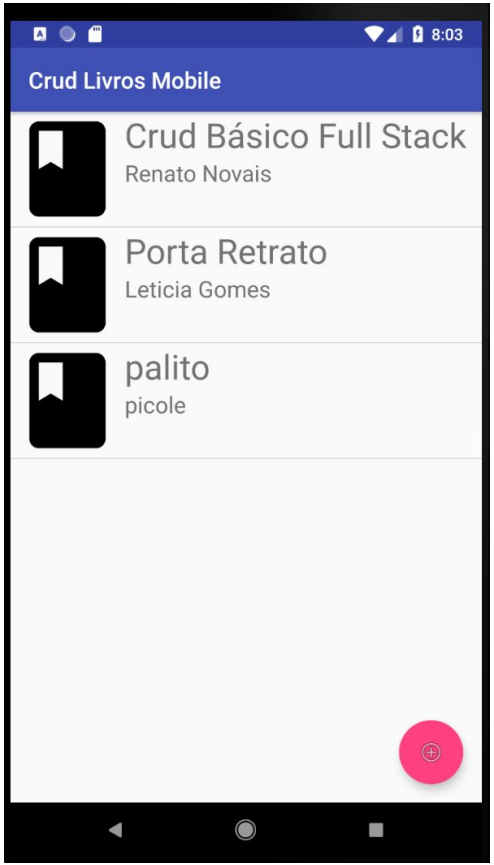
DisplayMessageActivity.java | MainActivity.java | AndroidManifest.xml | BlankFragment.jav

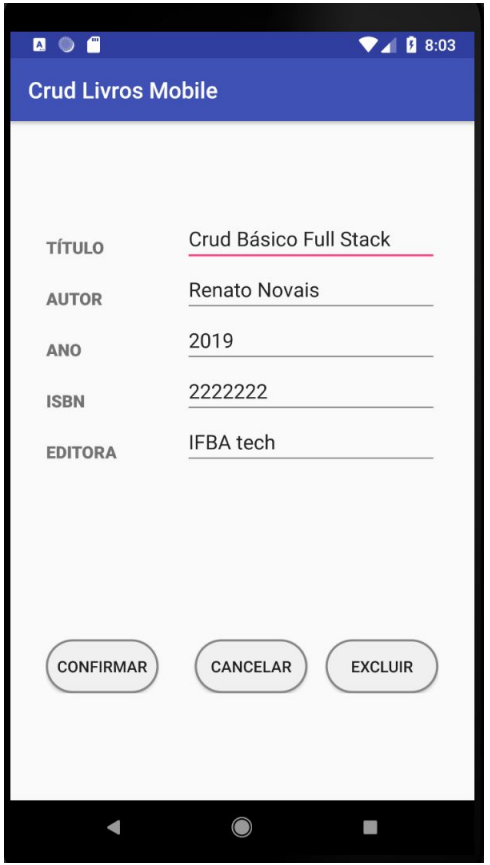
```
1   package com.example.renatonovais.myfistapplication;
2
3   import ...
7
8   public class DisplayMessageActivity extends AppCompatActivity {
9
10      @Override
11      protected void onCreate(Bundle savedInstanceState) {
12          super.onCreate(savedInstanceState);
13          setContentView(R.layout.activity_display_message);
14
15          // Get the Intent that started this activity and extract the string
16          Intent intent = getIntent();
17          String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
18
19          // Capture the layout's TextView and set the string as its text
20          TextView textView = (TextView) findViewById(R.id.textView);
21          textView.setText(message);
22      }
23  }
```

App resultante



A aplicação Crud-livros-Mobile





Activity com ListView

- Crie um projeto novo com um "Empty Activity" (sem suporte a Kotlin)
- Em resources/layout/activity_main.xml, substitua o código do botão por um de list view

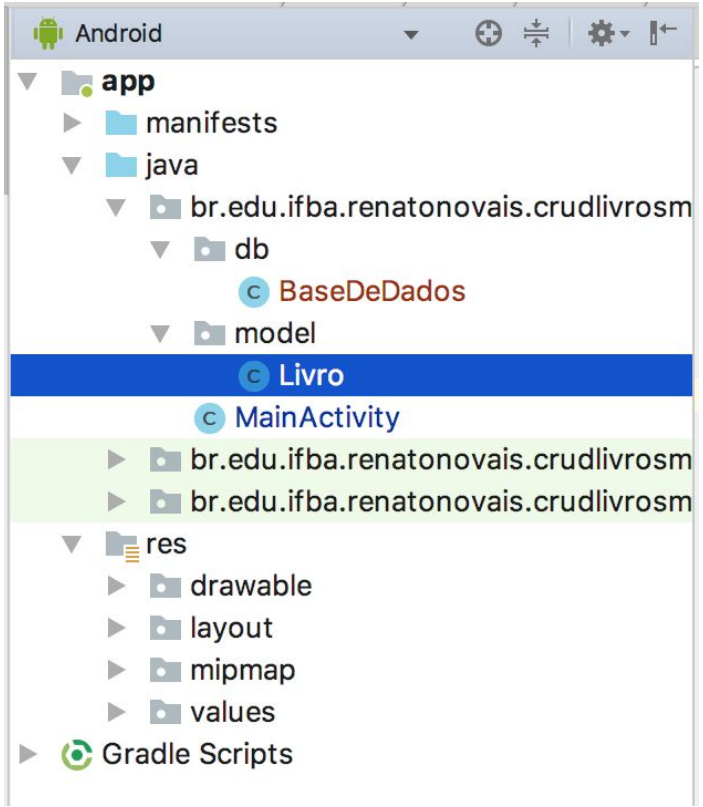
```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".MainActivity">

 <ListView
     android:id="@+id/lista"
     android:layout_width="match_parent"
     android:layout_height="match_parent">
</ListView>

</android.support.constraint.ConstraintLayout>
```

No MainActivity.onCreate

- i) acesse o list view, ii) crie a lista de livros, iii) crie um adapter com o layout e com a lista de livros, iv) sete o adapter no listview

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);

  ListView listViewDeLivros = (ListView) findViewById(R.id.lista);

  List<Livro> livros = BaseDeDados.getInstance().todosOsLivros();

  ArrayAdapter<Livro> adapter = new ArrayAdapter<Livro>(this,
        android.R.layout.simple_list_item_1, livros);

  listViewDeLivros.setAdapter(adapter);
}
```

Crie a classe de modelo que representa o Livro




```
package br.edu.ifba.renatonovais.crudlivrosmobile.model;

public class Livro {
        private int codigo;
        private String ISBN;
        private String titulo;
        private String autor;
        private int ano;
        private String editora;

        public Livro(int codigo, String ISBN, String titulo, String autor, int ano, String editora) {
                this.codigo = codigo;
                this.ISBN = ISBN;
                this.titulo = titulo;
                this.autor = autor;
                this.ano = ano;
                this.editora = editora;
        }

        public int getCodigo() {
             return codigo;
        }
        […]
```

Crie a classe que gerencia o acesso aos dados

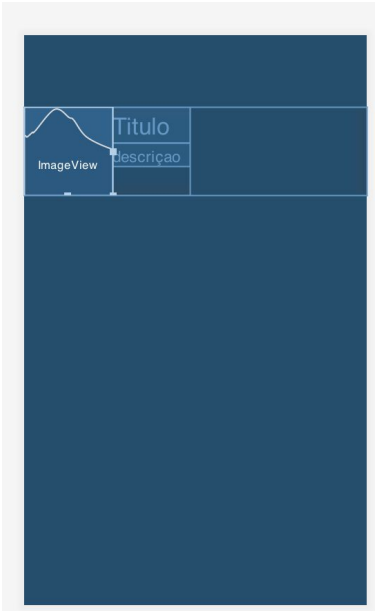
```
package br.edu.ifba.renatonovais.crudlivrosmobile.db;

import java.util.ArrayList;
import java.util.List;

import br.edu.ifba.renatonovais.crudlivrosmobile.model.Livro;

public class BaseDeDados {

   private static BaseDeDados bd;

   private BaseDeDados(){}
   public static BaseDeDados getInstance(){
      if (bd == null)
         bd = new BaseDeDados();
      return bd;
   }
```

```
public static List<Livro> todosOsLivros() {
      List<Livro> livros = new ArrayList<>();

      Livro livro1 = new Livro(1,
            "ISBN100", "Crud Básico full stack", "Renato Novais",2018,
"IFBA tech");
      Livro livro2 = new Livro(1,
            "ISBN101", "Django", "Renato Lima",2017, "GSort");
      Livro livro3 = new Livro(1,
            "ISBN5102", "Android", "Letícia Gomes",2014, "Favo");

      livros.add(livro1);
      livros.add(livro2);
      livros.add(livro3);

      return livros;
   }
}
```

Commit desta versão de código

Criando um ListView Customizado + SimpleAdapter

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="100dp"
  android:orientation="horizontal">
  <ImageView
    android:id="@+id/lista_livro_personalizada_imagem"
    android:layout_width="100dp"
    android:layout_height="match_parent"
    android:src="@drawable/ic_book_black_24dp" />

  <LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
      android:id="@+id/lista_livro_personalizada_nome"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="Titulo"
      android:textSize="30dp" />

    <TextView
      android:id="@+id/lista_livro_personalizada_descricao"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="descriçao"
      android:textSize="20dp" />

  </LinearLayout>

</LinearLayout>
```

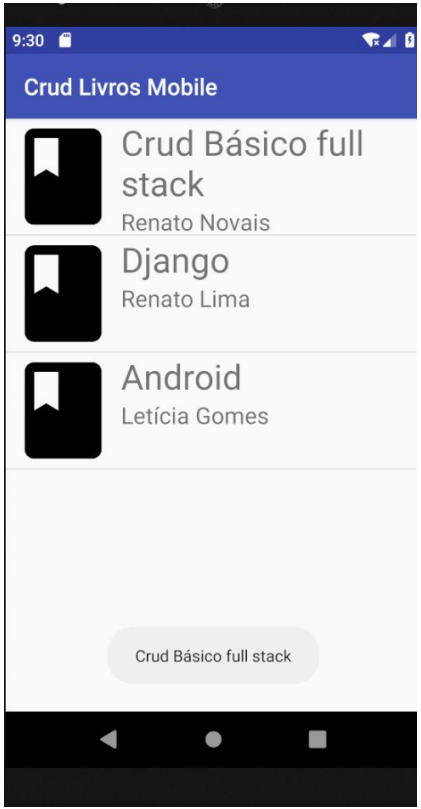


Criando o SimpleAdapter

```
// the placeholder to set content for each list item
String[] from = {"titulo", "autor"};
// the elements ids that will be set for each list item
int[] to = {R.id.lista_livro_personalizada_nome,
            R.id.lista_livro_personalizada_descricao};

ArrayList<HashMap<String,String>> arrayList=new ArrayList<>();

@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);

  ListView listViewDeLivros = (ListView) findViewById(R.id.lista);

  List<Livro> livros = BaseDeDados.getInstance().todosOsLivros();

  for (int i=0;i<livros.size();i++)
  {
     Livro livro = livros.get(i);
     HashMap<String,String> hashMap=new HashMap<>();
     hashMap.put("titulo",livro.getTitulo());
     hashMap.put("autor",livro.getAutor());
     hashMap.put("ano",livro.getAno()+"");
     arrayList.add(hashMap);//add the hashmap into arrayList
  }
```

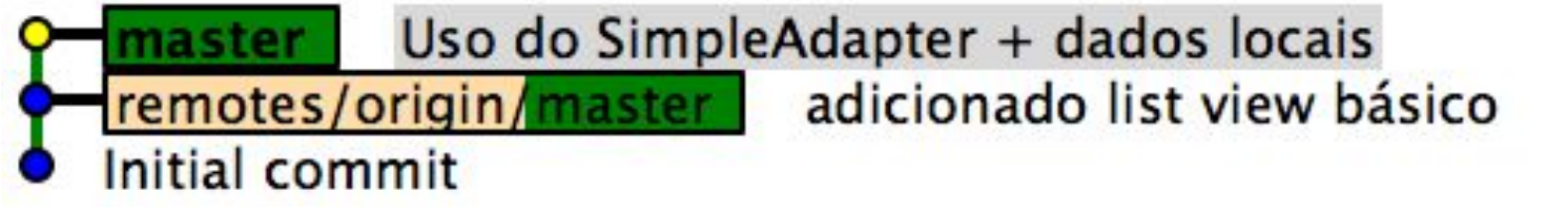
```
SimpleAdapter simpleAdapter=new
SimpleAdapter(this,arrayList,R.layout.lista_livro_personalizada,from,to);//C
reate object and set the parameters for simpleAdapter

listViewDeLivros.setAdapter(simpleAdapter);//sets the adapter for listView

  //perform listView item click event

listViewDeLivros.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
     @Override
     public void onItemClick(AdapterView<?> adapterView, View view, int i,
long l) {

Toast.makeText(getApplicationContext(),arrayList.get(i).get("titulo"),Toast.L
ENGTH_LONG).show();//show the selected image in toast according to
position
     }
  });

}
```

Como está o app?



- O icone foi adicionado a res/drawable através da opção: clique com botão direito sobre a pasta res, new -> Vector Asset
- O layout do ListView faz referência para ele
- Ao clicar no item da lista é mostrado uma mensagem com o Toast, na parte inferior

Commit desta versão de código



Próximo passo, configurar acesso ao Serviço rest

- Utilizaremos a api Retrofit
- Primeiramente, é necessário colocar no build.gradle as dependências relacionadas ao Retrofit
- É preciso definir um ServiceGenerator
  - Responsável pelas configurações iniciais do Retrofit
- Depois é preciso criar a classe responsável pelas requisições
  - Cada entidade pode ter sua classe própria.
  - No nosso caso temos o LivroService
- Os próximos três slides apresentam o builde.gradle, o ServiceGenerator.java e o LivroService.java

```
…

dependencies {
   implementation fileTree(dir: 'libs', include: ['*.jar'])
   implementation 'com.android.support:appcompat-v7:28.0.0'
   implementation 'com.android.support.constraint:constraint-layout:1.1.3'
   implementation 'com.android.support:design:28.0.0'
   testImplementation 'junit:junit:4.12'
   androidTestImplementation 'com.android.support.test:runner:1.0.2'
   androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'

   //retrofit

   implementation 'com.google.code.gson:gson:2.8.5'
   implementation 'com.squareup.retrofit2:retrofit:2.4.0'
   implementation 'com.squareup.retrofit2:converter-gson:2.4.0'
   implementation 'com.squareup.okhttp3:logging-interceptor:3.5.0'

}
```

```
public class ServiceGenerator {

  private static final String BASE_URL = "http://stadsifba.pythonanywhere.com/";

   private static Retrofit.Builder builder = new Retrofit.Builder()
           .baseUrl(BASE_URL)
         .addConverterFactory(GsonConverterFactory.create());

   private static Retrofit retrofit = builder.build();

   private static HttpLoggingInterceptor loggingInteceptor = new HttpLoggingInterceptor().setLevel(HttpLoggingInterceptor.Level.BODY);
   private static OkHttpClient.Builder httpClientBuilder = new OkHttpClient.Builder();

   public static <S> S createService(Class<S> serviceClass) {
      if (!httpClientBuilder.interceptors().contains(loggingInteceptor)) {
         httpClientBuilder.addInterceptor(loggingInteceptor);
         builder = builder.client(httpClientBuilder.build());
         retrofit = builder.build();
      }

      return retrofit.create(serviceClass);
   }

}
```

```java
public interface LivroService {

  @Headers({
        "Accept: application/json",
        "Content-type: application/json"
  })
  @GET("api_rest/livros/")
  Call<List<Livro>> getLivros();

  @POST("api_rest/cadastrarLivro/")
  Call<Livro> insereLivro(@Body Livro livro);

  @POST("api_rest/atualizarLivro/")
  Call<Livro> atualizaLivro(@Body Livro livro);

  @POST("api_rest/excluirLivro/")
  Call<Livro> excluirLivro(@Body Livro livro);

}
```

Próximo passo, pegar a lista do Serviço rest

- Utilizaremos a mesma listview. Ela será preenchida dentro do onStart.
- Nesse método, iremos:
  - Acessar a lista que está no layout
  - Instanciar o LivroService
  - Criar um mecanismo de progresso para mostrar processando enquanto os dados estão sendo carregados
  - Fazer a chamada ao serviço rest que retorna a lista de livros
  - Implementar o callback (retorno)
    - onResponse
    - onFailure

```
protected void onStart() {
  super.onStart();

  final ListView lista = (ListView) findViewById(R.id.lista);
  //registerForContextMenu(lista);
  LivroService livroService = ServiceGenerator.createService(LivroService.class);

  dialog = new ProgressDialog(MainActivity.this);
  dialog.setMessage("Carregando...");
  dialog.setCancelable(false);
  dialog.show();

  final Call<List<Livro>> call = livroService.getLivros();


…...
```

```
call.enqueue(new Callback<List<Livro>>() {
  @Override
  public void onResponse(@NonNull Call<List<Livro>> call, @NonNull Response<List<Livro>> response) {

    if (dialog.isShowing())
      dialog.dismiss();
    if (Objects.requireNonNull(response).isSuccessful()) {
      final List<Livro> listBooks = response.body();
      if (listBooks != null) {
        setData(listBooks);
        lista.setOnItemClickListener(new AdapterView.OnItemClickListener() {
          @Override
          public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
            HashMap<String,String> livroMap = (HashMap<String,String>)simpleAdapter.getItem(position);
            Intent intent = new Intent(MainActivity.this, CadastroLivroActivity.class);
            intent.putExtra("titulo", livroMap.get("titulo"));          intent.putExtra("autor", livroMap.get("autor"));
            intent.putExtra("ano", livroMap.get("ano"));          intent.putExtra("ISBN", livroMap.get("ISBN"));
            intent.putExtra("editora", livroMap.get("editora"));          intent.putExtra("codigo", livroMap.get("codigo"));
            startActivity(intent);
          }
        });
      }
    }
  }
```

Chamando outra activity

- Observe no código do slide anterior que no onClick do item da lista tem o uso do Intent para a chamada da nova activity e para passar valores

```
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        HashMap<String,String> livroMap = (HashMap<String,String>)simpleAdapter.getItem(position);
        Intent intent = new Intent(MainActivity.this, CadastroLivroActivity.class);
        intent.putExtra("titulo", livroMap.get("titulo"));
        intent.putExtra("autor", livroMap.get("autor"));
        intent.putExtra("ano", livroMap.get("ano"));
        intent.putExtra("ISBN", livroMap.get("ISBN"));
        intent.putExtra("editora", livroMap.get("editora"));
        intent.putExtra("codigo", livroMap.get("codigo"));
        startActivity(intent);
```

```
    @Override
    public void onFailure(@NonNull Call<List<Livro>> call, @NonNull Throwable t) {
        if (t instanceof IOException) {
            Toast.makeText(getBaseContext(),
                    "Problema ao conectar no servidor",
                    Toast.LENGTH_SHORT).show();
        }


    }
});

}
```

```
public void setData(List<Livro> livros){
  ListView listViewDeLivros = (ListView) findViewById(R.id.lista);

  for (int i=0;i<livros.size();i++)
  {
     Livro livro = livros.get(i);
     HashMap<String,String> hashMap=new HashMap<>();//create a hashmap to store the data in key value pair
     hashMap.put("codigo",livro.getCodigo());
     hashMap.put("titulo",livro.getTitulo());
     hashMap.put("autor",livro.getAutor());
     hashMap.put("ISBN",livro.getISBN());
     hashMap.put("editora",livro.getEditora());
     hashMap.put("ano",livro.getAno());

     arrayList.add(hashMap);//add the hashmap into arrayList
     if (Integer.parseInt(livro.getCodigo()) > maior_codigo)
        maior_codigo = Integer.parseInt(livro.getCodigo());
  }

  simpleAdapter=new SimpleAdapter(this,arrayList,R.layout.lista_livro_personalizada,from,to);//Create object and set the
parameters for simpleAdapter
  listViewDeLivros.setAdapter(simpleAdapter);//sets the adapter for listView
}
```

A mesma activity para Criar e Editar um livro

- Criar a activity e o layout
- No layout, adicionar os elementos visuais correspondentes
- Utilizaremos três botões
  - Confirmar: para inserir e atualizar
  - Cancelar: volta para a activity main
  - Excluir: para excluir

O Layout activity_cadastro_livro.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingTop="32dp"
    android:paddingRight="16dp"
    tools:context="br.edu.ifba.renatonovais.crudlivrosmobile.activities.CadastroLivroActivity">

    <LinearLayout...>

    <LinearLayout...>

    <LinearLayout...>

</RelativeLayout>
```

- Labels (TextView)
- Edits (EditText)
- Buttons (Button)

Labels

```
<LinearLayout
    android:id="@+id/labels_cadastro_livro"
    android:layout_width="76dp"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_marginStart="16dp"
    android:layout_marginTop="50dp"
    android:orientation="vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:fontFamily="sans-serif-medium"
        android:maxLines="1"
        android:text="Título"
        android:textAllCaps="true"
        android:textSize="15sp"
        android:textStyle="bold" />

    <TextView...>

    <TextView...>

    <TextView...>

    <TextView...>


</LinearLayout>
```

Edits

```
<LinearLayout
    android:layout_width="225dp"
    android:layout_height="wrap_content"
    android:layout_alignTop="@+id/labels_cadastro_livro"
    android:layout_alignParentEnd="true"
    android:layout_marginEnd="16dp"
    android:orientation="vertical">

    <EditText
        android:id="@+id/edtTitulo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="text" />

    <EditText...>


    <EditText...>

    <EditText...>


    <EditText...>


</LinearLayout>
```

Buttons

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true"
    android:layout_marginBottom="95dp"
    android:orientation="horizontal">

    <Button
        android:id="@+id/botao_enviar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="16dp"
        android:layout_weight="1"
        android:background="@drawable/button_shape"
        android:fontFamily="sans-serif-medium"
        android:text="Confirmar" />

    <Button...>

    <Button...>
</LinearLayout>
```

O Activity CadastroLivroActivity.java

- No onCreate é necessário verificar se veio dados da activity que chamou
  - se sim, preencher os campos
- No onCreate deve-se implementar o evento para ouvir o clicks dos botões

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_cadastro_livro);
    Intent intent = getIntent();

    final EditText edtTitulo = (EditText) findViewById(R.id.edtTitulo);
    final EditText edtAutor = (EditText) findViewById(R.id.edtAutor);
    final EditText edtAno = (EditText) findViewById(R.id.edtAno);
    final EditText edtIsbn = (EditText) findViewById(R.id.edtIsbn);
    final EditText edtEditora = (EditText) findViewById(R.id.edtEditora);

    final String codigo = (String) intent.getSerializableExtra("codigo");

    if (Integer.parseInt(codigo) != -1){ //atualizar
        edtTitulo.setText((String) intent.getSerializableExtra("titulo"));
        edtAutor.setText((String) intent.getSerializableExtra("autor"));
        edtAno.setText((String) intent.getSerializableExtra("ano"));
        edtIsbn.setText((String) intent.getSerializableExtra("ISBN"));
        edtEditora.setText((String) intent.getSerializableExtra("editora"));
    }
```

```
Button btnAtualizar = (Button) findViewById(R.id.botao_enviar);
  btnAtualizar.setOnClickListener(new View.OnClickListener() {
     @Override
     public void onClick(View v) {
        dialog = new ProgressDialog(CadastroLivroActivity.this);
        dialog.setMessage("Carregando...");
        dialog.setCancelable(false);
        dialog.show();

        Livro livro = new Livro(codigo,
              edtIsbn.getText().toString(),
              edtTitulo.getText().toString(),
              edtAutor.getText().toString(),
              edtAno.getText().toString(),
              edtEditora.getText().toString()
        );

        insere_atualiza_livro(livro);
     }
  });
```

```
Button btnCancelar = (Button)
findViewById(R.id.botao_cancelar);
  btnCancelar.setOnClickListener(new View.OnClickListener() {
     @Override
     public void onClick(View v) {
        Intent intent = new Intent(CadastroLivroActivity.this,
MainActivity.class);
        startActivity(intent);
        finish();
     }
  });
```

```
Button btnExcluir = (Button)findViewById(R.id.botao_excluir);
btnExcluir.setOnClickListener(newView.OnClickListener()
  @Override
  public void onClick(View v) {
     final Livro livro = new Livro(codigo,
           edtIsbn.getText().toString(),
           edtTitulo.getText().toString(),
           edtAutor.getText().toString(),
           edtAno.getText().toString(),
           edtEditora.getText().toString()
     );
     exibirConfirmacao(livro);
  }
});
```

```
private void insere_atualiza_livro(Livro livro) {
   LivroService livroService =ServiceGenerator.createService(LivroService.class);
   Call<Livro> call;
   final String msn;
   if (Integer.parseInt(livro.getCodigo()) == -1) {
      livro.setCodigo(Integer.toString(MainActivity.getNovoCodigo()));
       call = livroService.insereLivro(livro);
       msn = "inserido";
   }else{
       call = livroService.atualizaLivro(livro);
       msn = "atualizado";
   }
```

```
call.enqueue(new Callback<Livro>() {
      @Override
      public void onResponse(Call<Livro> call,Response<Livro> response) {
         if (dialog.isShowing()) {
            dialog.dismiss();
          if (response.isSuccessful()) {
               Toast.makeText(getBaseContext(), "Livro"+msn+" com sucesso", Toast.LENGTH_SHORT).show();
               startActivity(new Intent(CadastroLivroActivity.this, MainActivity.class));
               finish();
            } else {
               Toast.makeText(getBaseContext(), "Não foi possível realizar a operação",
Toast.LENGTH_SHORT).show();
            }
         }
      }
```

```
@Override
      public void onFailure(Call<Livro> call, Throwable t) {
         if (dialog.isShowing())
            dialog.dismiss();
         if (t instanceof IOException) {
            Toast.makeText(getBaseContext(),
                  "Problema ao conectar, verifique sua internet",
                  Toast.LENGTH_SHORT).show();
         }
      }
   });
}
```

```
private void delete_livro(Livro livro) {

    dialog = new ProgressDialog(CadastroLivroActivity.this);
    dialog.setMessage("Deletando...");
    dialog.setCancelable(false);
    dialog.show();

    LivroService livroService = ServiceGenerator.createService(LivroService.class);
    Call<Livro> call = livroService.excluirLivro(livro);
```

```
call.enqueue(new Callback<Livro>() {
  @Override
  public void onResponse(Call<Livro> call, Response<Livro> response) {

    if (dialog.isShowing()) {
      dialog.dismiss();
      if (response.isSuccessful()) {
        Toast.makeText(getBaseContext(), "Livro excluído com sucesso", Toast.LENGTH_SHORT).show();
        startActivity(new Intent(CadastroLivroActivity.this, MainActivity.class));
        finish();
      } else {
        Toast.makeText(getBaseContext(), "Não foi possível realizar a operação",
Toast.LENGTH_SHORT).show();
      }
    }
  }
```

```
@Override
    public void onFailure(Call<Livro> call, Throwable t) {
      if (dialog.isShowing())
        dialog.dismiss();
      if (t instanceof IOException) {
        Toast.makeText(getBaseContext(),
            "Problema ao conectar, verifique sua internet",
            Toast.LENGTH_SHORT).show();
      }
    }
  });
}
```

Referências

- https://speakerdeck.com/rodrigorgs/introducao-ao-android
- https://www.android.com/
- https://developer.android.com/training
- https://www.tutorialspoint.com/android