# RESUMO –

## Variáveis – “palavra\_palavra”

Elas seguem o padrão de utilizar apenas letras minúsculas sem acento, e em caso de duas ou mais palavras, utilizar de “\_” para separar as palavras. Segue o exemplo:

“name\_user”.

“password\_user”

“name\_category”

“icon\_comunity”

## Funções – “palavraVerbo”

Elas seguem o padrão de utilizar letra minúsculas na primeira palavra e na segunda que deve ser um verbo Letra maiúscula na primeira letra, sem acento. Segue o exemplo:

“userRegister”

“userOpenProfile”

“categoryCreate”

“categoryClick”

## Classes – “PalavraPalavra”

Elas seguem o padrão de utilizar maiúscula na primeira letra de cada palavra. Segue o exemplo:

“UserModel”

“CategoryModel”

“ComunityController”

“PlaceModel”

**SUMÁRIO**

[RESUMO – 1](#_Toc174301262)

[Variáveis – “palavra\_palavra” 1](#_Toc174301263)

[Funções – “palavraVerbo” 1](#_Toc174301264)

[Classes – “PalavraPalavra” 1](#_Toc174301265)

[1. API – 2](#_Toc174301266)

[1.1. Variáveis – 2](#_Toc174301267)

[1.2. Funções – 3](#_Toc174301268)

[1.3. Classes – 4](#_Toc174301269)

# 

# 1. API –

## 1.1. Variáveis –

Aquelas que representem dados de tabelas do banco, terão o mesmo nome dos campos que representam, como por exemplo: “name\_user”; “description\_category”. Em seu nome já há “descrição” do que faz no código.

**context.** Relativo à conexão com o banco de dados, servindo como um array formado por todas as tabelas do banco.

**check\_email.** Váriavel booleana referente ao retorno de uma função que verifica se o email digitado já está inserido no banco de dados.

**check\_login.** Váriavel booleana referente ao retorno de uma função que verifica se o login, ou seja, o @ digitado já está inserido no banco de dados.

**get\_users.** Guarda todos os objetos “User” retornados na função para puxar os usuários do banco de dados.

**categories.** Integra a tabela categories do banco de dados com a classe CategoryModel.

**users.** Integra a tabela users do banco, com a classe UserModel.

**cities.** Integra a tabela cities do banco, com a classe CityModel.

**comunities.** Integra a tabela comunities do banco, com a classe ComunityModel.

## 1.2. Funções –

**userCreate.** Retorna um usuário novo para o banco de dados com base nos parâmetros inseridos.

**userGet.** Retorna todos os usuários existentes no banco de dados.

**userGetID.**  Retorna apenas um usuário, aquele com o id do parâmetro.

**userChange.** Altera todos os dados de um usuário especificado pelo id.

**userDelete.** Desativa o usuário especificado.

**categoryCreate.** Retorna uma categoria nova para o banco de dados com base nos parâmetros inseridos.

**categoryGet.** Retorna todos os usuários existentes no banco de dados.

**categoryGetID.** Retorna uma categoria, aquela com o id do parâmetro.

**categoryChange.** Altera todos os dados de uma categoria especificado pelo id.

**categoryDelete.** Deleta uma categoria do banco de dados.

**cityGet.** Retorna todos as cidades existentes no banco de dados.

**cityGetID.** Retorna apenas uma cidade, aquele com o id do parâmetro.

## 1.3. Classes –

**ConnectionContext.** Integra o banco de dados ao programa, neste caso, a API

**CategoryController.** Classe que gera as URLs para manipulação da tabela categories do banco.

**CityController.** Classe que gera as URLs para manipulação da tabela cities do banco.

**ComunityController.** Classe que gera as URLs para manipulação da tabela comunities do banco.

**UserController.** Classe que gera as URLs para manipulação da users categories do banco.

**CategoryModel.** Classe para instanciar uma linha da tabela categories do banco.

**CityModel.** Classe para instanciar uma linha da tabela cities do banco.

**ComunityModel.** Classe para instanciar uma linha da tabela comunities do banco.

**UserModel.** Classe para instanciar uma linha da tabela users do banco.

# 2. APLICAÇÃO

## **2.1. VÁRIAVEIS**

### 2.1.1 LoginActivity.kt

1. **binding:** Tipo: ActivityLoginBinding. Objeto da classe ActivityLoginBinding. Utilizado para agrupar e referenciar todos os componentes visuais dentro do layout da LoginActivity.
2. **common\_events:** Tipo: CommonEvents. Objeto da classe CommonEvents.
3. **btn\_login:** Tipo: AppCompatButton. Objeto da classe AppCompatButton. Utilizado para guardar o componente btn\_login do layout da LoginActivity.
4. **et\_username:** Tipo: EditText. Objeto da classe EditText. Utilizado para guardar o componente et\_username do layout da LoginActivity.
5. **et\_password:** Tipo: EditText. Objeto da classe EditText. Utilizado para guardar o componente et\_password do layout da LoginActivity.
6. **api\_user:** Tipo: Endpoint. Objeto da classe personalizada Endpoint. Utilizado para chamar os Endpoints(URL’s) relacionadas a parte de usuários da API.

## 2.2. CLASSES

### 2.2.1 LoginActivity.kt

Classe para a tela de login.

## 2.3. FUNÇÕES

### 2.3.1 LoginActivity.kt

1. **componentsInitialize():** retorna vazio. Inicializa as variáveis da classe com seus respectivos valores. E altera seus eventos, a partir do objeto common\_events.
2. **enterClick():** retorna vazio. Verifica se os campos et\_username e et\_password estão vazios. Caso não estejam, chama a função userLogar().
3. **userLogar().** Retorna vazio. Verifica.