

# **Escola Superior de Tecnologia e Gestão**

## **Licenciatura em Engenharia Informática**

### **Sistemas Distribuídos**

**Ano Letivo 2021/22**

### **Trabalho 3**

**Elaborado em: 2022/06/27**

**Daniel Santos | N°2019133865**

## **Índice**

<b>List of Figures .....</b>	<b>ii</b>
<b>1. Introdução .....</b>	<b>1</b>
<b>2. App .....</b>	<b>2</b>
2.1. Páginas da app .....	3
2.2. Gestão de dados .....	11
2.3. Diagrama ER .....	14
2.4. API REST .....	15
<b>3. Conclusão .....</b>	<b>17</b>
<b>4. Referências .....</b>	<b>18</b>

## List of Figures

FIGURA 1 - ELIMINAR LISTA/TAREFA.....	2
FIGURA 2 - EDITAR LISTA/TAREFA.....	2
FIGURA 3 - ADICIONAR LISTA/TAREFA.....	2
FIGURA 4 - LISTAR TODAS AS TAREFAS.....	2
FIGURA 5 - RETORNAR À PÁGINA ANTERIOR.....	2
FIGURA 6 - PÁGINA ESTÁTICA ABOUT IT.....	2
FIGURA 7 - MainActivity PORTRAIT.....	3
FIGURA 8 - MainActivity LANDSCAPE.....	3
FIGURA 9 - ABOUT IT.....	4
FIGURA 10 - NOME DE UTILIZADOR (DANIEL).....	4
FIGURA 11 - MainActivity COM NOME.....	5
FIGURA 12 - LISTAGEM DE TODAS AS TAREFAS.....	5
FIGURA 13 - ADICIONAR ToDoList.....	6
FIGURA 14 - EDITAR ToDoList.....	6
FIGURA 15 - ELIMINAR ToDoList.....	7
FIGURA 16 - ToDoListActivity.....	7
FIGURA 17 - ADICIONAR TASK.....	8
FIGURA 18 - TaskInfoActivity.....	8
FIGURA 19 - TASK CONCLUÍDA.....	9
FIGURA 20 - MUDAR TASK.....	9
FIGURA 21 - ELIMINAR TASK1.....	10
FIGURA 22 - FUNCTION ONActivityRESULT.....	11
FIGURA 23 - GETTASK PARTE1.....	11
FIGURA 24 - GETTASK PARTE2.....	12
FIGURA 25 - ADDLIST PORTRAIT.....	13
FIGURA 26 - ADDLIST LANDSCAPE.....	13
FIGURA 27 - DIAGRAMA ER.....	14
FIGURA 28 - MÉTODO POST.....	15
FIGURA 29 - MÉTODO GET.....	15
FIGURA 30 - MÉTODO PUT.....	16

## **1. Introdução**

Este trabalho foi realizado no âmbito da disciplina de Tecnologias de Aplicações Móveis por Daniel Santos a pedido de trabalho dos professores Francisco Afonso e Gonçalo Marques.

Neste relatório irei demonstrar e explicar o desenvolvimento da app desenvolvida para o trabalho, com as screenshots devidas.

## 2. App

A app contém vários botões que apesar das imagens serem intuitivas irei apresentar:



Figura 1 - Eliminar lista/tarefa



Figura 2 - Editar lista/tarefa



Figura 3 - Adicionar lista/tarefa



Figura 4 - Listar todas as tarefas



Figura 5 - Retornar à página anterior



Figura 6 - Página estática About It

## 2.1. Páginas da app

Ao começar a app iremos observar uma destas activities:



Figura 7 - MainActivity portrait

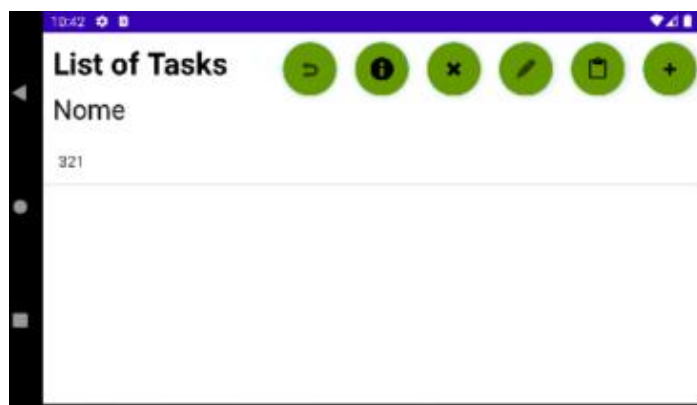


Figura 8 - MainActivity landscape

Ao carregar no botão About It irá dar para uma página de informação:

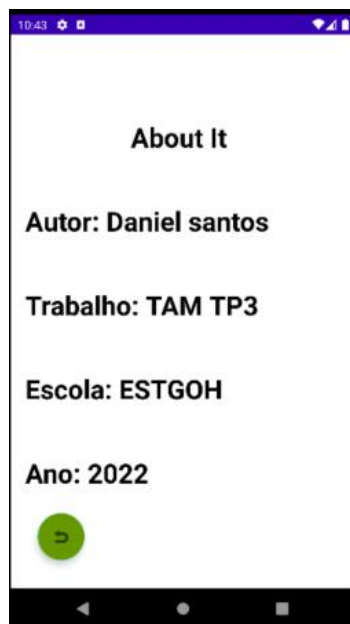


Figura 9 – About It

Para retornar pode simplesmente carregar no botão retornar.

Ao carregar no botão mudar o utilizador (ferramenta) irá para uma página para mudar o nome do utilizador:

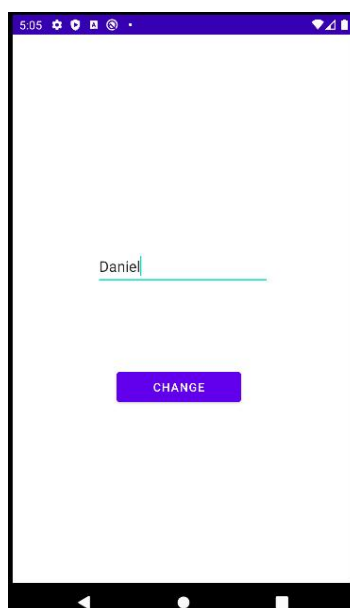


Figura 10 – Nome de Utilizador (Daniel)

Ao voltar aparecerá assim:

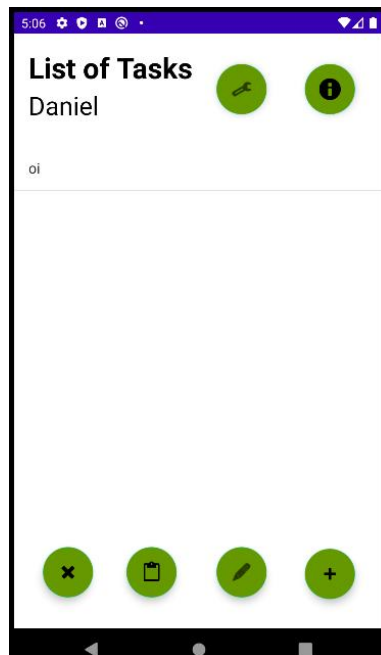


Figura 11 – MainActivity com nome

Ao carregar no botão listar tudo irá ser levado para a seguinte página:

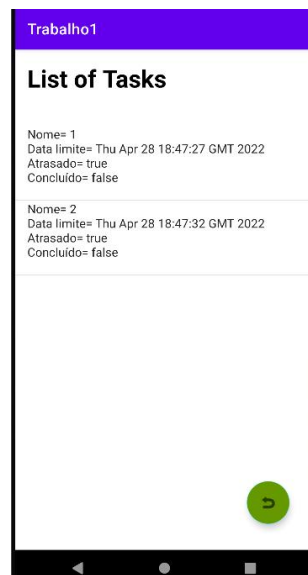


Figura 12 - Listagem de todas as tarefas

Para retornar pode simplesmente carregar no botão retornar.



Ao voltar para a MainActivity pode adicionar uma ToDoList através do botão adicionar:

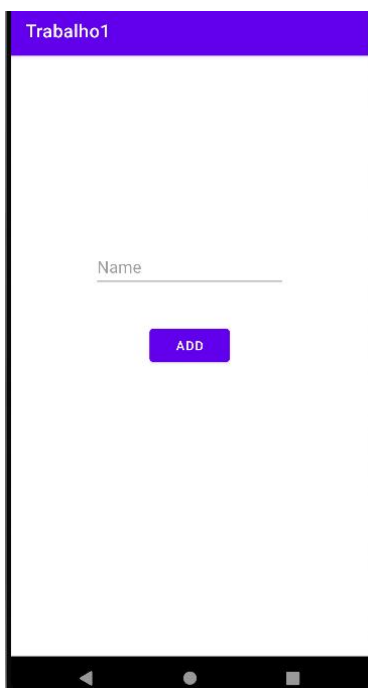


Figura 13 - Adicionar ToDoList

Depois de adicionar pode também modificar o nome da lista através do botão editar:

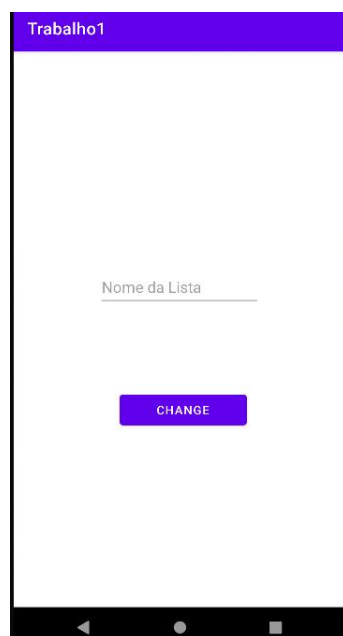


Figura 14 – Editar ToDoList

Só falta o botão eliminar da MainActivity que serve para eliminar uma ToDoList mas a app só permite fazê-lo quando não existem tarefas associadas à ToDoList:



Figura 15 - Eliminar ToDoList

Ao carregar em cima da ToDoList que deseja irá ser redirecionado para outra a ToDoListActivity:

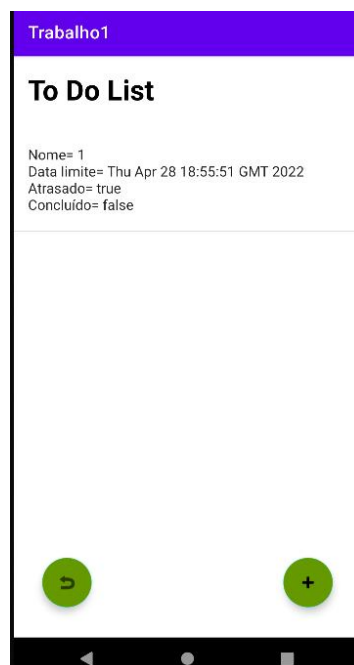


Figura 16 – ToDoListActivity

Existe duas opções:

Retornar irá voltar à MainActivity.

Adicionar irá adicionar uma nova Task:

Ao adicionar uma Task:

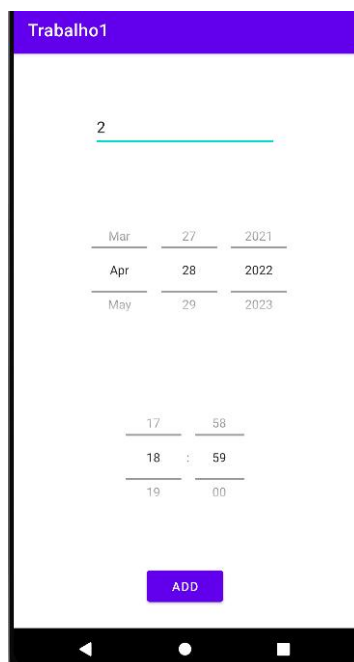


Figura 17 - Adicionar Task

Adicionar uma Task é constituída por uma descrição, um datePicker e um timePicker.

Ao carregar em cima de uma tarefa será levado para uma TaskInfoActivity:



Figura 18 - TaskInfoActivity

Aqui poderá visualizar todas as informações acerca da Task e poderá retornar, editar, eliminar e definir a task como concluída.

Concluir a Task irá levá-lo para a activity anterior e o concluído passa para True:

```
Nome= 1  
Data limite= Thu Apr 28 18:55:51 GMT 2022  
Atrasado= true  
Concluído= true
```

Figura 19 - Task Concluída

Neste caso esta Task também se encontra atrasada pois o prazo da task já passou.

Ao editar a Task irá aparecer uma interface similar à de adicionar:

Trabalho1

1

Mar 27 2021  
Apr 28 2022  
May 29 2023

18 06  
19 07  
20 08

CHANGE

Figura 20 - Mudar Task

Por fim poderá eliminar a Task com apenas um toque na cruz:

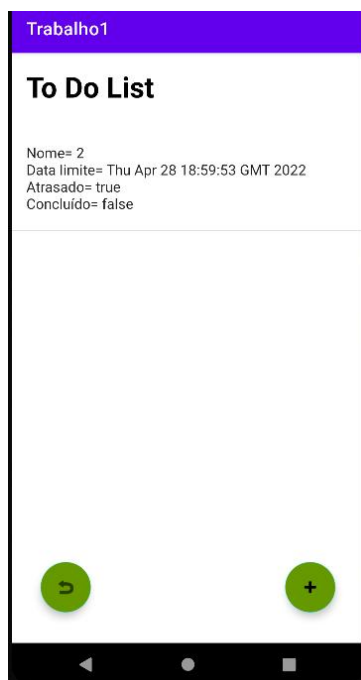


Figura 21 - Eliminar Task1

Antes existia a Task1 e Task2, agora só existe a Task2 pois a Task1 foi eliminada.

## 2.2. Gestão de dados

A gestão de dados é feita através de uma base de dados. Os dados são inseridos, editados, eliminados e obtidos através da base de dados e usa-se um arrayList para processar esses dados.

Para resolver tal problema usei o método onActivityResult que atualiza o arrayAdapter sempre que algo é modificado nas arraylist's e dou check se as datas estão atrasadas e atualiza na base de dados:

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    getTasks();
    if(listTask!=null) {
        for (int i = 0; i < listTask.size(); i++) {
            if (listTask.get(i).getData_limite().before(Calendar.getInstance())) {
                bd.open();
                bd.updateTaskLate(listTask.get(i).getId(), aAtrasado: 1);
                bd.close();
            }
        }
    }
    getTasks();
}
```

Figura 22 - Function onActivityResult

A função getTasks() funciona como um sincronizador da minha base de dados com o meu arrayList:

```
public void getTasks() {
    listTask.clear();

    bd.open();

    Cursor cur = bd.getAllTasks();

    if (cur != null) {
        if (cur.moveToFirst()) {
            while (!cur.isAfterLast()) {
                @SuppressWarnings("Range") int id = cur.getInt(cur.getColumnIndex(s: "_id"));
                @SuppressWarnings("Range") String descricao = cur.getString(cur.getColumnIndex(s: "descricao"));
                @SuppressWarnings("Range") String aDataLimite = cur.getString(cur.getColumnIndex(s: "dataLimite"));
                @SuppressWarnings("Range") String listName = cur.getString(cur.getColumnIndex(s: "listName"));
                @SuppressWarnings("Range") int aPassouLimite = cur.getInt(cur.getColumnIndex(s: "atrasado"));
                @SuppressWarnings("Range") int aConcluido = cur.getInt(cur.getColumnIndex(s: "concluido"));
            }
        }
    }
}
```

Figura 23 - getTask parte1

```
Calendar dataLimite = Calendar.getInstance();
String str[] = aDataLimite.split(" ");
dataLimite.set(Integer.parseInt(str[0]), Integer.parseInt(str[1]), Integer.parseInt(str[2]), Integer.parseInt(str[3]),
boolean passouLimite=false;
boolean concluido=false;

if(aPassouLimite==1){
    passouLimite=true;
}
if(aConcluido==1){
    concluido=true;
}

Task task = new Task(id, descricao, dataLimite, listName, passouLimite, concluido);
listTask.add(task);
cur.moveToNext();
}
}
}
bd.close();
}
```

Figura 24 - getTask parte2

Na parte 1 da função getTask() o programa corre o cursor e obtêm os dados.

Na parte 2 da função getTask() o programa processa os dados e adiciona no arrayList.

Os programas também mantêm dados o utilizador muda para landscape ou vise-versa:

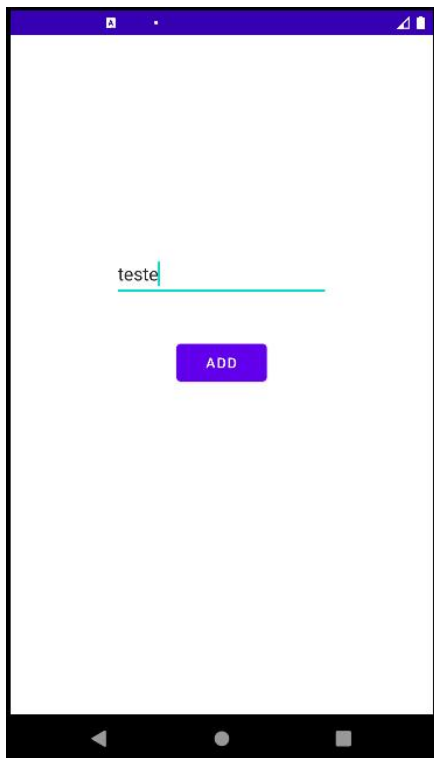


Figura 25 - AddList portrait

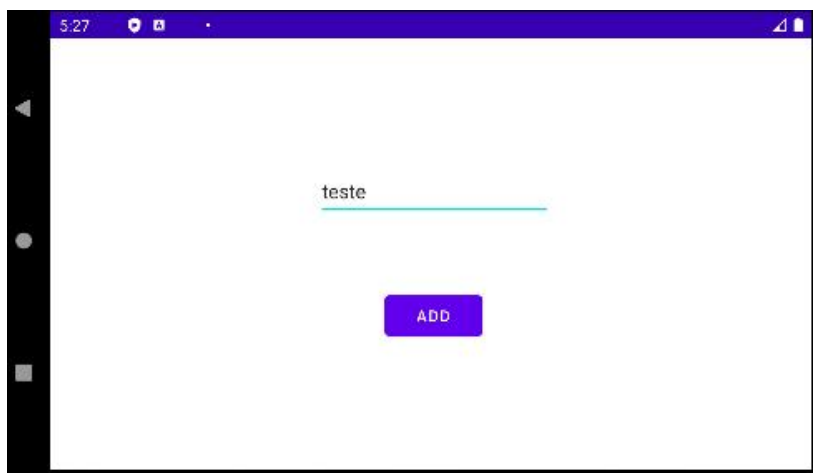


Figura 26 - AddList Landscape



## 2.3. Diagrama ER

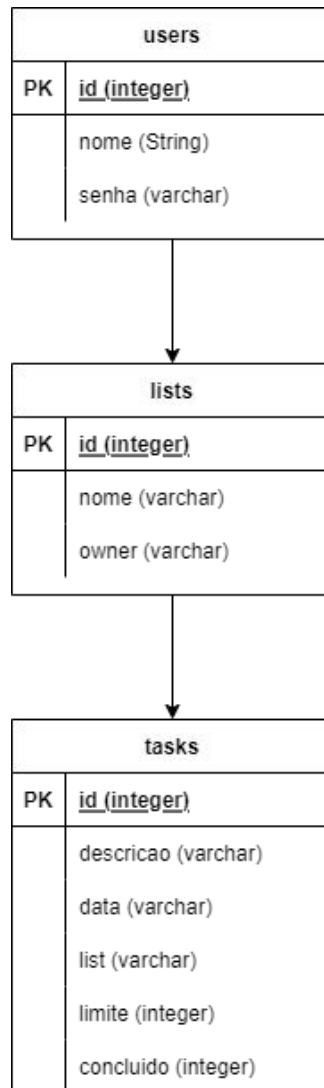


Figura 27 - Diagrama ER

## 2.4. API REST

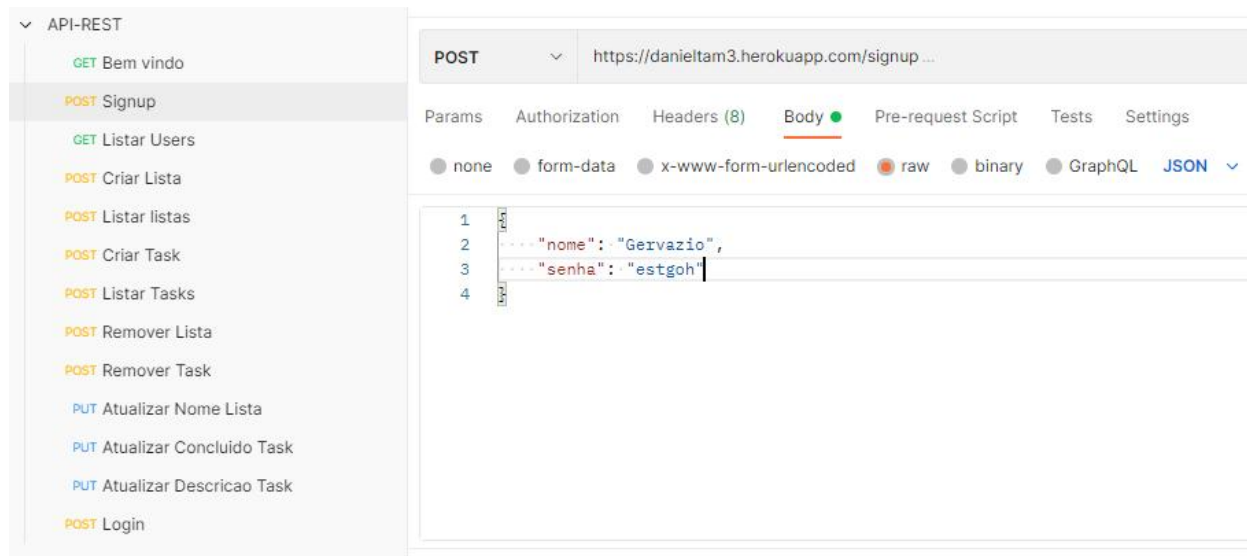


Figura 28 - Método POST

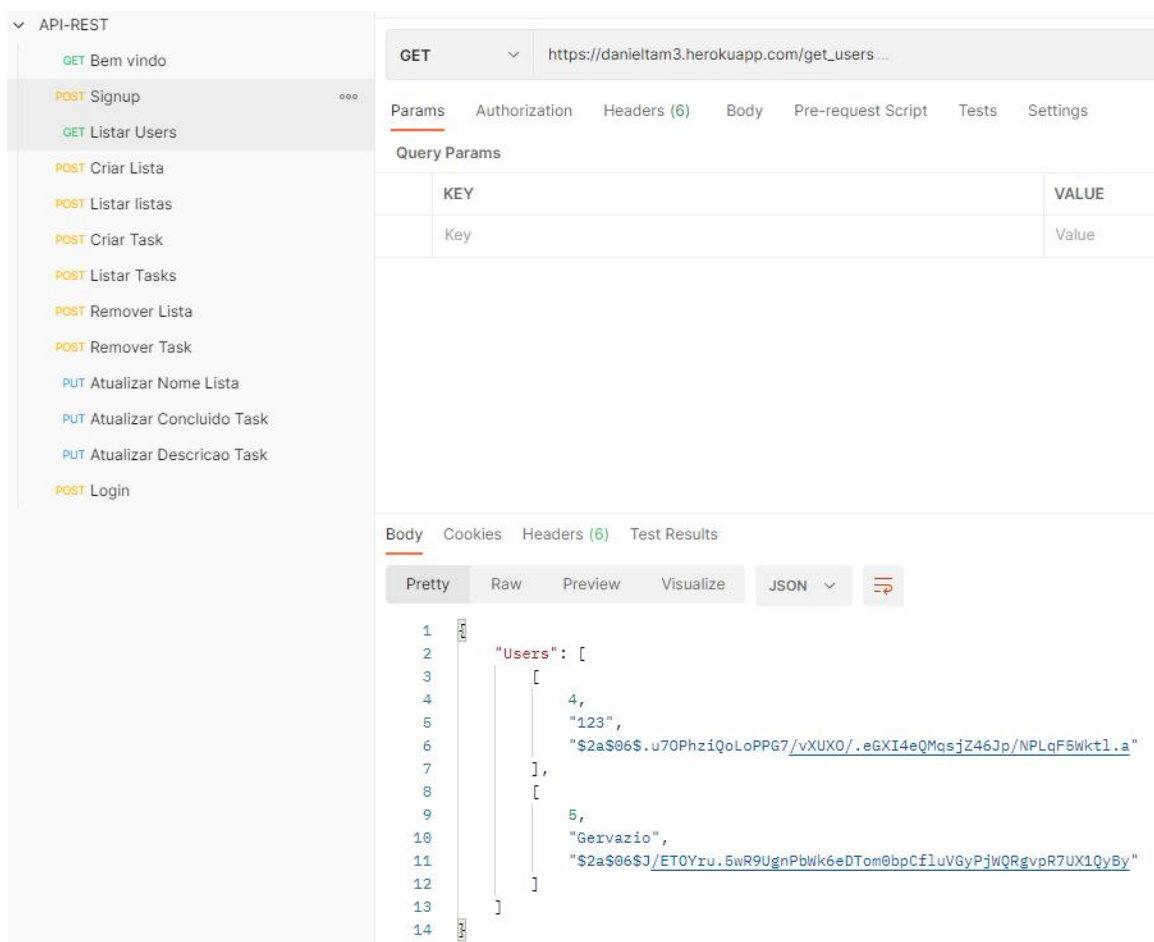


Figura 29 - Método GET

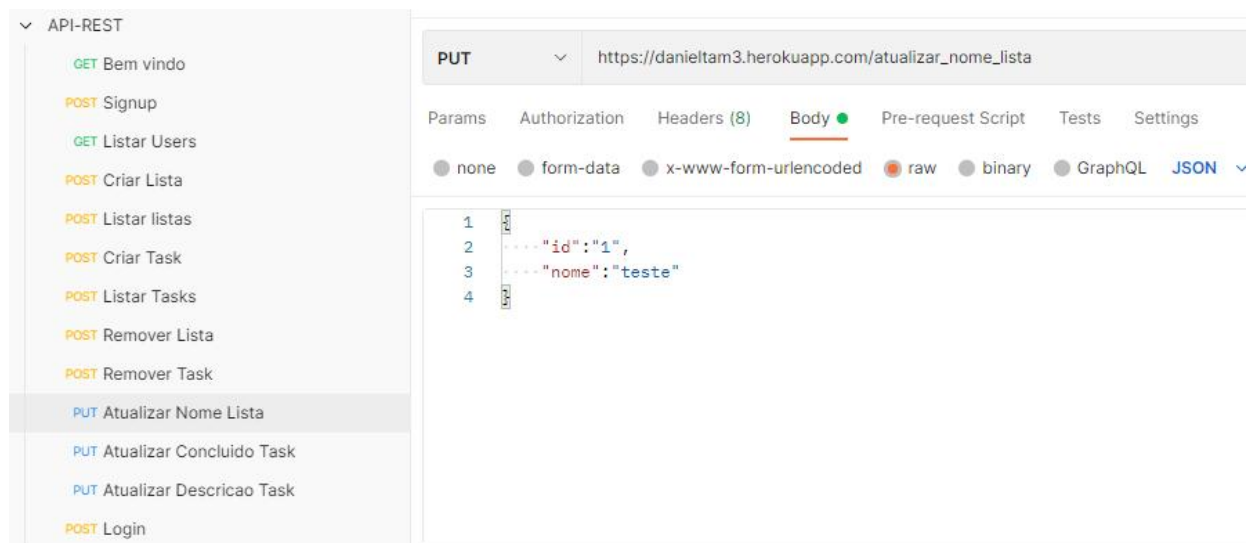


Figura 30 - Método PUT

Os requests são todos baseados nesta forma:

Os métodos POST's são feitos através de formato JSON, e os parâmetros dependem das variáveis introduzidas (ex: signup pede nome e pass, criar listas pede nome e dono).

Os métodos GET's são feitos através de formato JSON para obter bem-vindo e listar utilizadores.

Os métodos PUT's são feitos através de formato JSON, e os parâmetros pedem sempre um ID, mais uma parâmetro que muda as variáveis.

A minha API comunica com a base de dados para adicionar, listar, editar e remover ficheiros.

Tomei em particular atenção à encriptação de passwords na Base de Dados.

### **3. Conclusão**

Gostei de realizar o trabalho, pois ensinou-me a trabalhar com o Android Studio e API REST que são tecnologias que para o mercado de trabalho podem vir a ajudar.

Felizmente, já tinha um conhecimento sólido em java o que permitiu um “flow” de trabalho e aprendizagem consistente.

## **4. Referências**

- [1] Tyrone, «android: makes activity A wait for activity B to finish and returns some values», *Stack Overflow*, 16 de junho de 2012. <https://stackoverflow.com/q/11046810> (acedido 28 de abril de 2022).
- [2] «Seletores | Desenvolvedores Android», *Android Developers*. <https://developer.android.com/guide/topics/ui/controls/pickers?hl=pt-br> (acedido 28 de abril de 2022).