ENGENHARIA INFORMÁTICA

Arquiteturas Móveis – 2020/2021

Trabalho de Investigação - Ionic

António Santos – 2017014206

Bruno Matos – 2018018487

Filipe Silva – 2016020567

Turma: P2+PL

Coimbra, 2021

Índice

Introdução

O presente relatório foi elaborado no âmbito da disciplina de Arquiteturas Móveis e tem como finalidade estudar uma plataforma de desenvolvimento cross-platform, sendo que optamos pela Ionic.

Ionic, é um framework para desenvolvimento multiplataforma, isto é, tanto *web* como aplicações móveis. O objetivo da ferramenta é que com apenas um único código seja possível executar em várias plataformas, daí o termo: *cross-platform*. Outro objetivo é que esta seja de fácil entendimento e utilização. Para além do que foi referido, a Ionic, é uma biblioteca de componentes, possibilitando que com apenas a declaração de um componente seja feita uma estrutura, ou seja, é uma biblioteca de temas em que cada tema corresponde a um bloco de código em HYML, CSS e Javascript.

Ionic foi criada por Max Lynch, Ben Sperry e Adam Bradley da empresa *Drifty Co,* em 2013. Foi construída em javascript e originalmente usado com Angular e Apache Cordova. No entanto foi sendo atualizada ao longo dos anos e a partir de 2019 é possível, também, utilizar com React or Vue.js.

1. Conceitos essenciais

Ionic é uma framework de desenvolvimento móvel de aplicações em HTML5 com o objetivo de criar aplicações híbridas. Aplicações híbridas são essencialmente sites que são executados dentro de uma estrutura browser que tem acesso às funções do sistema.

Para facilitar a compreensão e utilização da aplicação seguem-se alguns conceitos essenciais:

- **Componentes UI:** Ionic é uma biblioteca de componentes user interface, isto é, blocos de código que serão usados para construir a *app*. Os componentes foram criados a partir de padrões *web* de html, css e javascript. Os componentes podem ser alterados pelo desenvolvedor.
- **Estilo adaptativo:** é uma ferramenta do Ionic que permite os desenvolvedores usem o mesmo código em várias plataformas. Cada componente é adaptada à plataforma onde está a ser executada. Assim é possível que para o mesmo código seja representado de forma diferente em cada plataforma, tendo assim, um aspeto mais familiar.
- **Navegação:** a navegação tradicional na *web* é continua, isto é, ao navegar no site não é salva a informação da página quando se muda para outra página. Já nas aplicações móveis a navegação é paralela, isto é, normalmente quando se muda para uma parte da app, ao voltar novamente atrás a informação permanece. Ionic utiliza a navegação em paralela guardando assim as informações ao longo da navegação, mesmo nas plataformas web.
- **Acesso nativo:** uma das vantagens de aplicações híbridas é que são executadas virtualmente em cada plataforma. Assim, o mesmo código irá utilizar funções nativas diferentes de cada plataforma.
- **Temas:** nuclearmente o ionic é construido usando css assim utilizando as propriedados do css. Assim, é fácil a utilização é redefenição do design da app. É dado os temas padrões que podem ser facilmente alterados pelos desenvolvedores.

2. Instalação de ferramentas

Passos de instalação:

1. Instalar Node.js
2. Abrir a consola CMD.
3. Introduzir o código: nmp -g install

```
C:\Users\Bruno Matos>nmp -g  install
```

3. Criação de um projeto

1 - Entrar na pasta onde se deseja criar o projeto;

```
C:\Users\Bruno Matos>cd Pasta_do_Projeto
```

2 - Inserir na consola:

```
C:\Users\Bruno Matos>ionic start
```

3 - Escolher a framework para o projeto.

```
? Framework: (Use arrow keys)
> Angular | https://angular.io
  React   | https://reactjs.org
  Vue     | https://vuejs.org
```

4 - Inserir o nome do projeto:

```
very great app needs a name!

lease enter the full name of your app. You can change this at any time. To bypass this prompt next time, supply name,
he first argument to ionic start.

 Project name:
```

5 - Escolher um template inicial para o projeto.

```
? Starter template: (Use arrow keys)
> tabs         | A starting project with a simple tabbed interface
  sidemenu     | A starting project with a side menu with navigation in the content area
  blank        | A blank starter project
  list         | A starting project with a list
  my-first-app | An example application that builds a camera with gallery
  conference   | A kitchen-sink application that shows off all Ionic has to offer
```

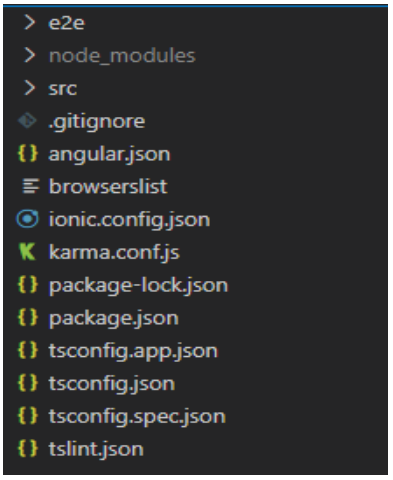
Para ver o programa em tempo real, abra a pasta onde se encontra o projeto, selecione o seguinte:

```
:\Users\Bruno Matos\Desktop\multiscreens>ionic serve
```

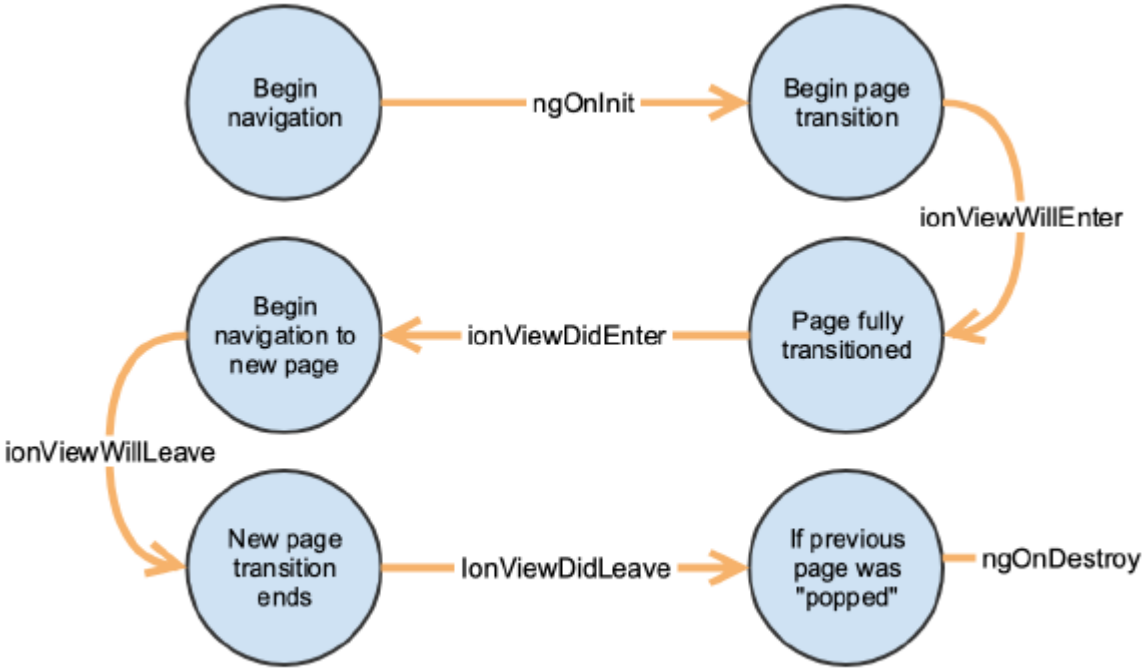
Abrir-se há uma página no browser caso este suporte este tipo de interação, essa pagina representa então o seu projeto em tempo real.

4. Ficheiros que constituem um novo projeto

- O exemplo apresentado abaixo utiliza a framework Angular.
- A diretoria ‘e2e’ serve para testar ponta a ponta.
- A diretoria ‘node_modules’, são as bibliotecas do javascript.
- A diretoria ‘src’ contém todos os ficheiros e recursos específicos da aplicação.
- Os restantes ficheiros são configurações javascript.

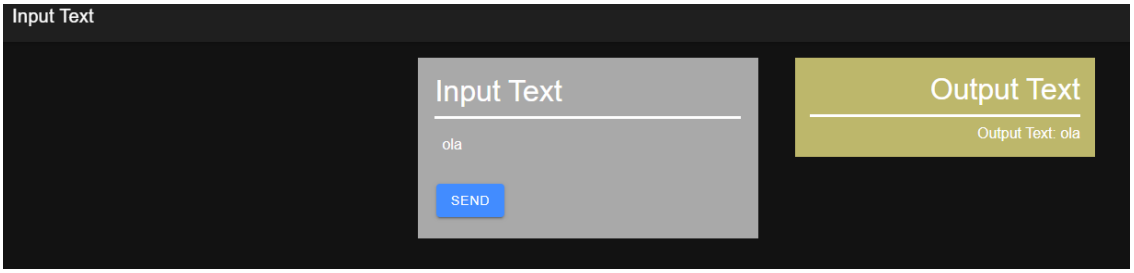


5. Ciclo de vida de uma página

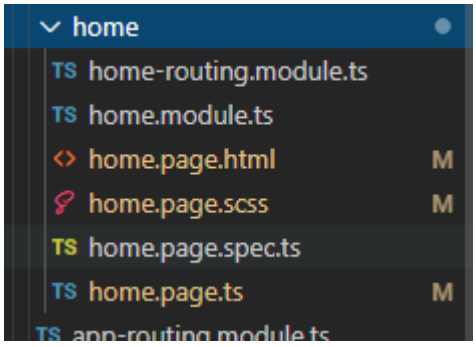


7. Projetos exemplificativos

Projeto de exemplo 1



- O objetivo do primeiro projeto foi criar um aplicação com uma caixa de texto onde se podia escrever e ao primir um botão o texto fosse exibido noutra caixa de texto.
- Foi utilizado um template disponiblizado “blank” devido a ser um projeto simples onde não seria necessário como por exemplo navegação entre páginas.



Apenas foi alterado do template disponiblizado os seguintes ficheiros (a amarelo na imagem):

- home.page.html
- home.page.scss
- home.page.ts

~

Home.page.html:

```
<ion-header [translucent]="true">
  <ion-toolbar>
    <ion-title>
      Input Text
    </ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <div class="container">
      <div id="block1">
        <div id="block11">
          <ion-label position="floating" style="font-size: 200%;">
            Input Text
          </ion-label>
          <hr style="border-top:3px solid white;">
          <ion-textarea clearOnEdit="true" [(ngModel)]="inputText" placeholder="Enter text here...">
          </ion-textarea>
          <ion-button color="primary" (click)="onBtnSubmit()">Send</ion-button>
        </div>
      </div>

      <div id="block1" style="padding: 20px;"></div>

      <div id="block2">
        <div id="block22">
          <ion-label position="floating" style="font-size: 200%;">
            Output Text
          </ion-label>
          <hr style="border-top:3px solid white;">
          <ion-label>
            Output Text: {{ outputText || 'Waiting for text'}}
          </ion-label>
        </div>
    </div>
  </div>
</ion-content>
```

- **Resumo dos componentes utilizados:**
  - Ion-header:
    - Introduzido um titulo da página.
      - ion-header: Declarando que é o header da app
      - ion-toolbar: Criada uma toolbar para o titulo
      - ion-title: Declarando o titulo da app.
  - Ion-content:
    - Utilizado 3 divs para o design da página (explicado melhor no .scss)
    - Ion-label: Utilizado para saida de texto.
    - Ion-textarea: Utilizado para introdução de texto.
    - Ion-button: Um botão para acionar a saida de texto resultante da "ion-textarea"
    - {{outputText || 'Waiting for text'}}: Será exibido "Waiting for text" até que a variável outputText tenha um valor.
- **Resumo do funcionamento:**
  - Será introduzido texto na "ion-textarea" ao primir o "ion-button" será enviado para a label à direira o texto introduzido.

Home.page.scss :

```scss
.container {
  display: flex;
  justify-content: center;
  padding: 1%;
}

#block1 {
  float:left;
}

#block2
{
  float:right;
}

#block11 {
  padding: 10%;
  background-color: darkgrey;
  float: right;;
  width: 200%;
}

#block22
{
  padding: 10%;
  text-align: right;
  background-color: darkkhaki;
  float: left;
  width: 200%;
}
```

- **Resumo dos componentes utilizados:**
  - .container:
    - Utilizado para encapsular todos os restantes elementos.
  - #block1:
    - Bloco dentro do container da esquerda.
  - #block2:
    - Bloco dentro do container da direita.
  - #block11:
    - Bloco dentro do do bloco 1 para alinar à direita.
  - #block22:
    - Bloco dentro do do bloco 2 para alinar à esquerda.

Home.page.ts :

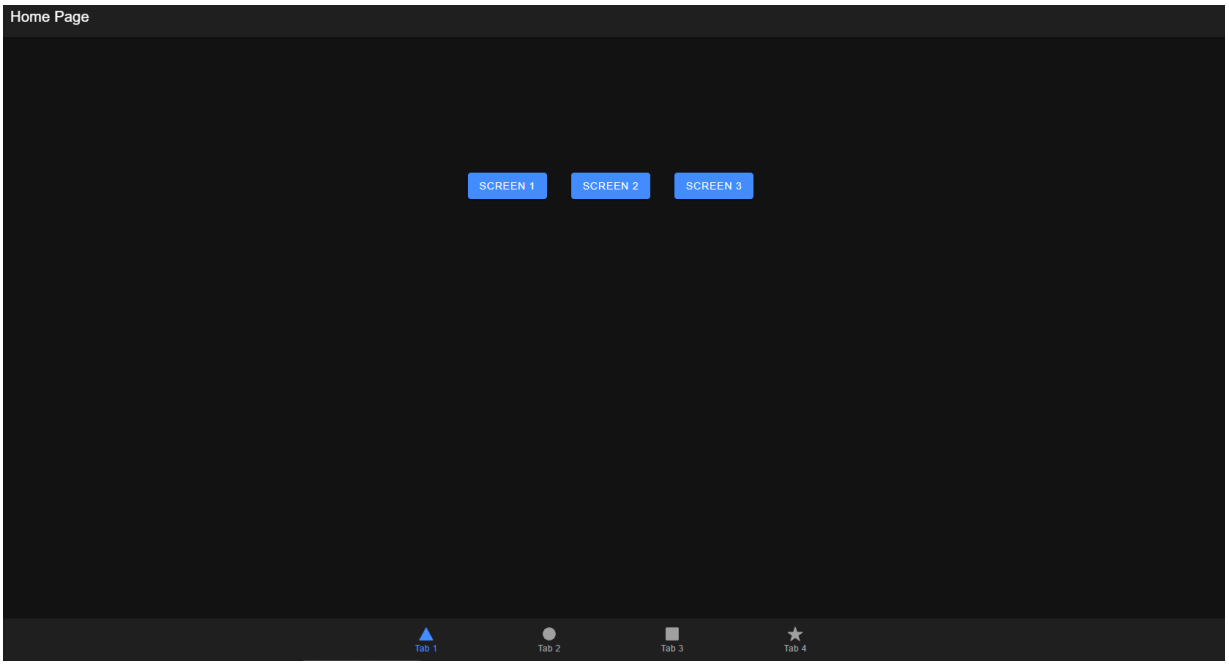
```
import { Component } from '@angular/core';

@Component({
  selector: 'app-home',
  templateUrl: 'home.page.html',
  styleUrls: ['home.page.scss'],
})
export class HomePage {

  inputText: string;
  outputText: string;


  constructor() {
  }

  onBtnSubmit(){
    this.outputText = this.inputText;
  }
}
```

- **Resumo dos componentes utilizados:**
  - onBtnSubmit(): Função será chamada quando o botão for acionado.
    - inputText: É uma string para onde será enviado o texto introduzido na "ion-textarea".

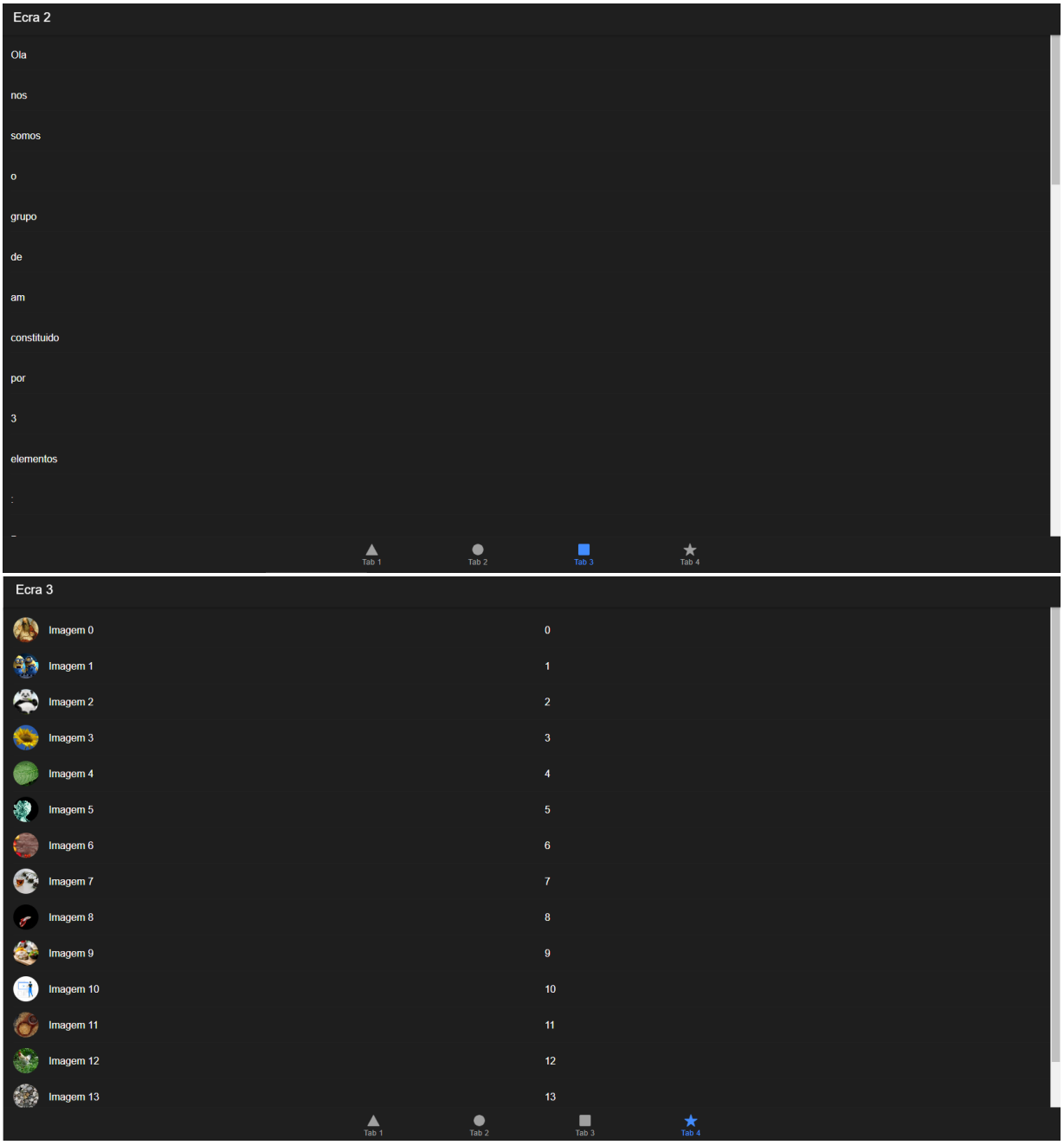
  outputText: É uma string que será enviada para a label.)

Projeto de exemplo nº2

Este projeto foi feito com a framework Angular. Esta aplicação é baseada na segunda proposta de aplicação colocada no enunciado do trabalho, em que consiste na criação de uma app com vários ecrãs e listas. No primeiro ecrã deverá haver 3 botões que nos deverão reencaminhar para outros 3 ecrãs. No primeiro ecrã é listada uma sequência de 1 a 1000. No segundo ecrã uma lista de palavras existentes num array gerido em memória com pelo menos 50 palavras. No terceiro ecrã lista com conteúdo, com mais de 15 elementos, sendo cada elemento constituído por vários campos independentes, cada elemento deverá incluis pelo menos uma imagem/icon, um texto e um número.


Home Page
SCREEN 1
SCREEN 2
SCREEN 3
Tab 1
Tab 2
Tab 3
Tab 4

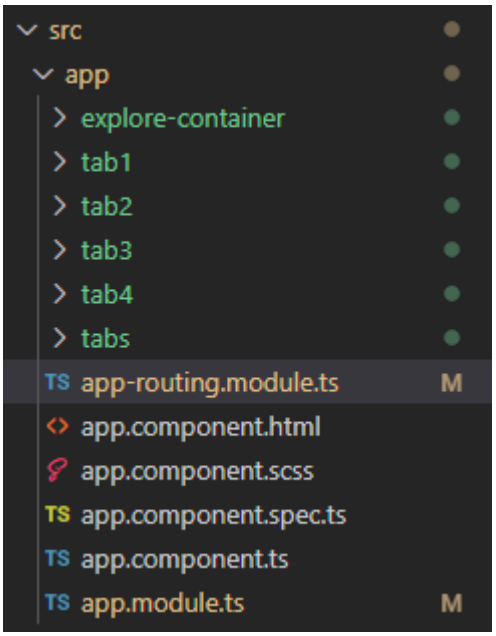

Ecra 1
1
2
3
4
5
6
7
8
9
10
11
12



- Este projeto foi criado através do template “tabs”, onde nos é fornecido uma barra de navegação com 3 páginas.

- Foi necessário adicionar mais uma página ao projeto. Para adicionar a página deve ser feito os seguintes passos:
  1. Abrir a consola na pasta do projeto, ou melhor ainda, usar a consola do VS Code;
  2. Inserir o comando: ionic g page <nome da página>

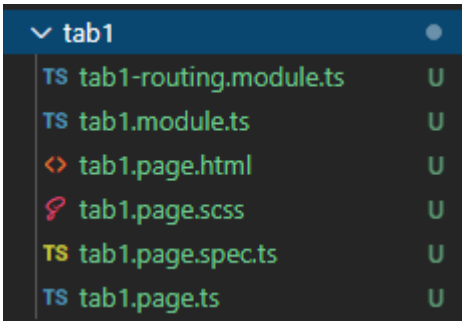
- É adicionada uma nova ao projeto uma pasta com o nome indicado e com vários ficheiros correspondentes.



Apenas foi alterado do template disponiblizado os seguintes pastas:

- Tab1
- Tab2
- Tab3
- Tab4
- Tabs (é carregada as paginas no rouiting aqui)
- App-routing.module.ts

Tab1



Tab1.page.html:

```
<ion-header [translucent]="true">
  <ion-toolbar>
    <ion-title>
      Home Page
    </ion-title>
  </ion-toolbar>
</ion-header>

<ion-content [fullscreen]="true">
  <div class="container">
    <ion-button color="primary" [routerLink]="['/tabs/tab2']" style="margin: 1%;">Screen 1</ion-button>
    <ion-button color="primary" [routerLink]="['/tabs/tab3']" style="margin: 1%;">Screen 2</ion-button>
    <ion-button color="primary" [routerLink]="['/tabs/tab4']" style="margin: 1%;">Screen 3</ion-button>
  </div>
</ion-content>
```

- **Resumo dos componentes utilizados:**
  - **Ion-header:**
    - **Introduzido um titulo da página.**
      - **ion-header: Declarando que é o header da app.**
      - **ion-toolbar: Criada uma toolbar para o titulo.**
      - **ion-title: Declarando o titulo da app.**
  - **Ion-content:**
    - **Utilizado 1 divs para o design da página.**
    - **Ion-button: Butao para redirecionar para outras paginas**
- **Resumo do funcionamento:**
  - **Escolhe um dos botões e será redirecionado para essas páginas.**

Tab1.page.scss :

```
.container {
    display: flex;
    justify-content: center;
    margin-top: 10%;
  }
```

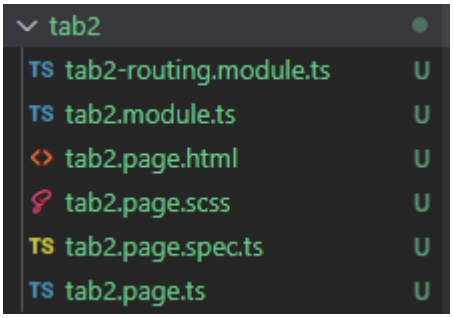
- O container serve para manter a informação no meio da mágina.

Tab1.page.ts :

```typescript
import { Component, OnInit } from '@angular/core';
import { Router, RouterLink } from '@angular/router'

@Component({
  selector: 'app-tab1',
  templateUrl: 'tab1.page.html',
  styleUrls: ['tab1.page.scss']
})
export class Tab1Page implements OnInit{

  constructor(private router: Router) { }

  ngOnInit() {
  }

  onBtnSubmit1() {
    this.router.navigate(['/tabs/tab2']);
  }
  onBtnSubmit2() {
    this.router.navigate(['/tab3']);
  }
  onBtnSubmit3() {
    this.router.navigate(['/tab4']);
  }
}

```

- Aqui é declarada as funções que irão ser chamadas ao carregar no botão e como podemos observar, quando estas são chamadas somos redirecionados para uma página.

Tab2



Tab2.page.html:

```
1
2   <ion-header [translucent]="true">
3     <ion-toolbar>
4       <ion-title>
5         Ecra 1
6       </ion-title>
7     </ion-toolbar>
8   </ion-header>
9
10  <ion-content [fullscreen]="true">
11    <ion-list *ngFor="let num of numeros">
12      <ion-item>
13        <ion-label>{{num}}</ion-label>
14      </ion-item>
15    </ion-list>
16  </ion-content>
17
```

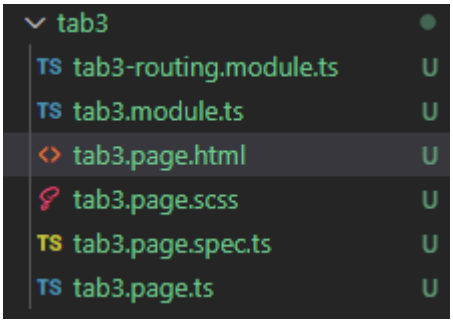
- **Resumo dos componentes utilizados:**
  - Ion-header:
    - Introduzido um titulo da página.
      - ion-header: Declarando que é o header da app.
      - ion-toolbar: Criada uma toolbar para o titulo.
      - ion-title: Declarando o titulo da app.
  - Ion-content:
    - Ion-list: Apresenta uma lista para cada num em numeros
    - Ion-item: É criado um item para cada num.
- **Resumo do funcionamento:**
  - É mostrado para cada num(numero) em numeros um item. Neste caso é apresentado um numero de 1 a 1000.

Tab2.page.ts :

```ts
import { Component } from '@angular/core';

@Component({
  selector: 'app-tab2',
  templateUrl: 'tab2.page.html',
  styleUrls: ['tab2.page.scss']
})
export class Tab2Page {

  numeros:Array<number>=new Array()
  constructor() {
    for(let i=0; i<1000;i++){
      this.numeros.push(i+1);
    }
  }
}
```

- É criado um array de números onde irão ser colocados números de 1 a 1000.

Tab3



Tab3.page.html:

```
<ion-header [translucent]="true">
  <ion-toolbar>
    <ion-title>
      Ecra 2
    </ion-title>
  </ion-toolbar>
</ion-header>

<ion-content [fullscreen]="true">
  <ion-list *ngFor="let palavra of palavras">
    <ion-item>
      <ion-label>{{palavra}}</ion-label>
    </ion-item>
  </ion-list>
</ion-content>
```

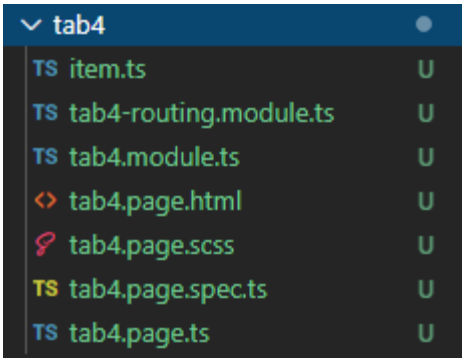
- **Resumo dos componentes utilizados:**
  - Ion-header:
    - Introduzido um titulo da página.
      - ion-header: Declarando que é o header da app.
      - ion-toolbar: Criada uma toolbar para o titulo.
      - ion-title: Declarando o titulo da app.
  - Ion-content:
    - Ion-list: Apresenta uma lista para cada palavra em palavras
    - Ion-item: É criado um item para cada palavra.
    - Ion-label: É mostrado uma label dessa palavra.
- **Resumo do funcionamento:**
  - É mostrado todas as palavras existentes no array palavras.

Tab2.page.ts :

```
import { Component} from '@angular/core';

@Component({
  selector: 'app-tab3',
  templateUrl: 'tab3.page.html',
  styleUrls: ['tab3.page.scss']
})
export class Tab3Page {

  palavras:Array<string>=new Array();

  constructor() {
    this.palavras.push('Ola','nos','somos','o','grupo','de','am','constituido','por','3','elementos',':','Bruno','Antonio','Filipe','e',
    'achamos','ionic','uma','ferramenta','bastante','util','para','o','desenvolvimento','de','uma','aplicação','damos','por','concluido','este','trabal
    'e','esperamos','ter','uma','boa','nota','obrigado','pela','vossa','disponiblidade');
  }
}
```

- É criado um array de palavras onde são adicionadas palavras dentro do construtor.

Tab4



Item.ts:

```
export class Item {
    numero:number;
    texto:string;
    img:string;

    constructor(numeros:number,textos:string,imgs:string){
        this.numero=numeros;
        this.texto=textos;
        this.img=imgs;
    }
}
```

- Foi criada uma class para ser adicionado um numero, uma imagem e um texto. Correspondendo assim a um elemento.

Tab4.page.html:

```
<ion-header [translucent]="true">
  <ion-toolbar>
    <ion-title>
      Ecra 3
    </ion-title>
  </ion-toolbar>
</ion-header>

<ion-content [fullscreen]="true">
  <ion-list>
    <ion-item *ngFor="let item of itens">
      <ion-avatar>
        <ion-img [src]="item.img"></ion-img>
      </ion-avatar>
        <ion-label style="margin-left: 1%;">{{item.texto}}</ion-label>
        <ion-label>{{item.numero}}</ion-label>
    </ion-item>

  </ion-list>
</ion-content>
```

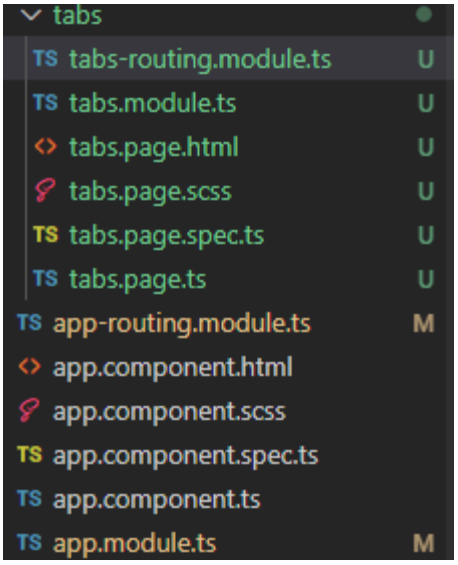
- **Resumo dos componentes utilizados:**
  - **Ion-header:**
    - **Introduzido um titulo da página.**
      - **ion-header: Declarando que é o header da app.**
      - **ion-toolbar: Criada uma toolbar para o titulo.**
      - **ion-title: Declarando o titulo da app.**
  - **Ion-content:**
    - **Ion-list: Apresenta uma lista para cada palavra em palavras**
    - **Ion-item: É criado um item para cada palavra.**
    - **Ion-avatar: Para todas as imagens ficarem com um tamanho e formato igual.**
    - **Ion-label: É mostrado uma label dessa palavra.**
- **Resumo do funcionamento:**
  - **É todos os item de itens, sendo que para cada um destes corresponde uma imagem um texto e um numero.**

Tab4.page.ts :

```typescript
import { Component } from '@angular/core';
import { Item } from './item';

@Component({
  selector: 'app-tab4',
  templateUrl: 'tab4.page.html',
  styleUrls: ['tab4.page.scss']
})

export class Tab4Page {
  itens: Array<Item> = new Array();
  palavra: string;
  numero: number = 0;
  imagem: string;

  constructor() {
    for (let i = 0; i < 16; i++) {
      this.palavra = "Imagem " + this.numero;
      this.imagem = "/assets/imagens/img" + this.numero + ".jpg";
      let item = new Item(this.numero, this.palavra, this.imagem)
      this.itens.push(item)
      this.numero++;
    }
  }
}
```

- Para cada item é adicionado uma imagem que se encontra na diretoria assets/imagens, um texto e um numero.

Tabs



Tabs.page.html:

```
<ion-tabs>

  <ion-tab-bar slot="bottom">
    <ion-tab-button tab="tab1">
      <ion-icon name="triangle"></ion-icon>
      <ion-label>Tab 1</ion-label>
    </ion-tab-button>

    <ion-tab-button tab="tab2">
      <ion-icon name="ellipse"></ion-icon>
      <ion-label>Tab 2</ion-label>
    </ion-tab-button>

    <ion-tab-button tab="tab3">
      <ion-icon name="square"></ion-icon>
      <ion-label>Tab 3</ion-label>
    </ion-tab-button>

    <ion-tab-button tab="tab4">
      <ion-icon name="star"></ion-icon>
      <ion-label>Tab 4</ion-label>
    </ion-tab-button>
  </ion-tab-bar>

</ion-tabs>
```

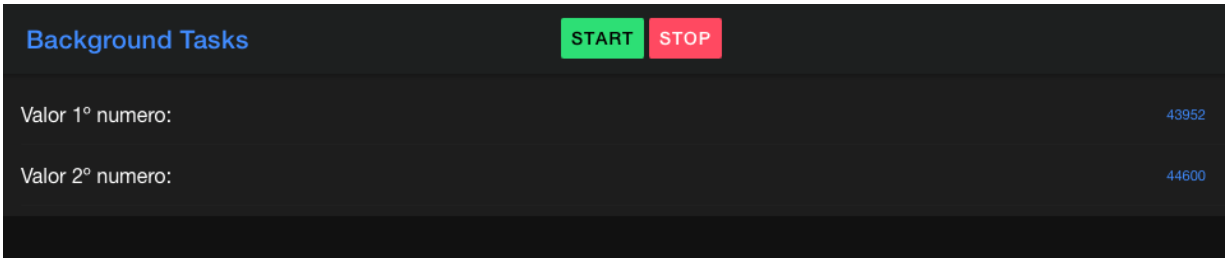
- **Resumo dos componentes utilizados:**
  - **Ion-tab-bar: Criada uma barra para a navegação.**
  - **Ion-tab-button: Criado um botão na barra de navegação.**
  - **Ion-icon: Criado um icon para o butão.**
  - **Ion-label: Criado uma label para o botão.**

tabs-routing.module.ts:

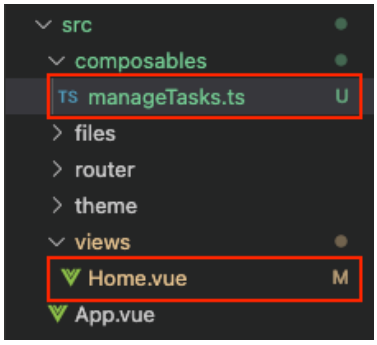
```ts
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { TabsPage } from './tabs.page';

const routes: Routes = [
  {
    path: 'tabs',
    component: TabsPage,
    children: [
      {
        path: 'tab1',
        loadChildren: () => import('../tab1/tab1.module').then(m => m.Tab1PageModule)
      },
      {
        path: 'tab2',
        loadChildren: () => import('../tab2/tab2.module').then(m => m.Tab2PageModule)
      },
      {
        path: 'tab3',
        loadChildren: () => import('../tab3/tab3.module').then(m => m.Tab3PageModule)
      },
      {
        path: 'tab4',
        loadChildren: () => import('../tab4/tab4.module').then(m => m.Tab4PageModule)
      },
      {
        path: '',
        redirectTo: '/tabs/tab1',
        pathMatch: 'full'
      }
    ]
  },
  {
    path: '',
    redirectTo: '/tabs/tab1',
    pathMatch: 'full'
  }
];

@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule]
})
export class TabsPageRoutingModule {}
```

- É carregado as paginas “tab1”,”tab2”,”tab3”,”tab4” para o routing.

Projeto de exemplo 3



Este projeto foi feito com a framework Vue. Esta aplicação é baseada na terceira e última proposta de aplicação colocada no enunciado do trabalho, em que consiste na criação de uma app com tarefas a correrem sobre 2 numeros no background, que estão armazenados num ficheiro. O objetivo é ler os números do ficheiro, incrementar o primeiro e decrementar o segundo, e por fim apresentá-los no ecrã, de 10 em 10 segundos até o utilizador escolher parar essas tarefas.



Para a sua implementação foi criado um novo projeto a partir do template "empty". Foi utilizada apenas a vista inicial "Home.vue", e para a gestão das tarefas foi criado um ficheiro **"manageTasks.ts"** que necessita de ser importado na vista para utilização dos seus métodos.

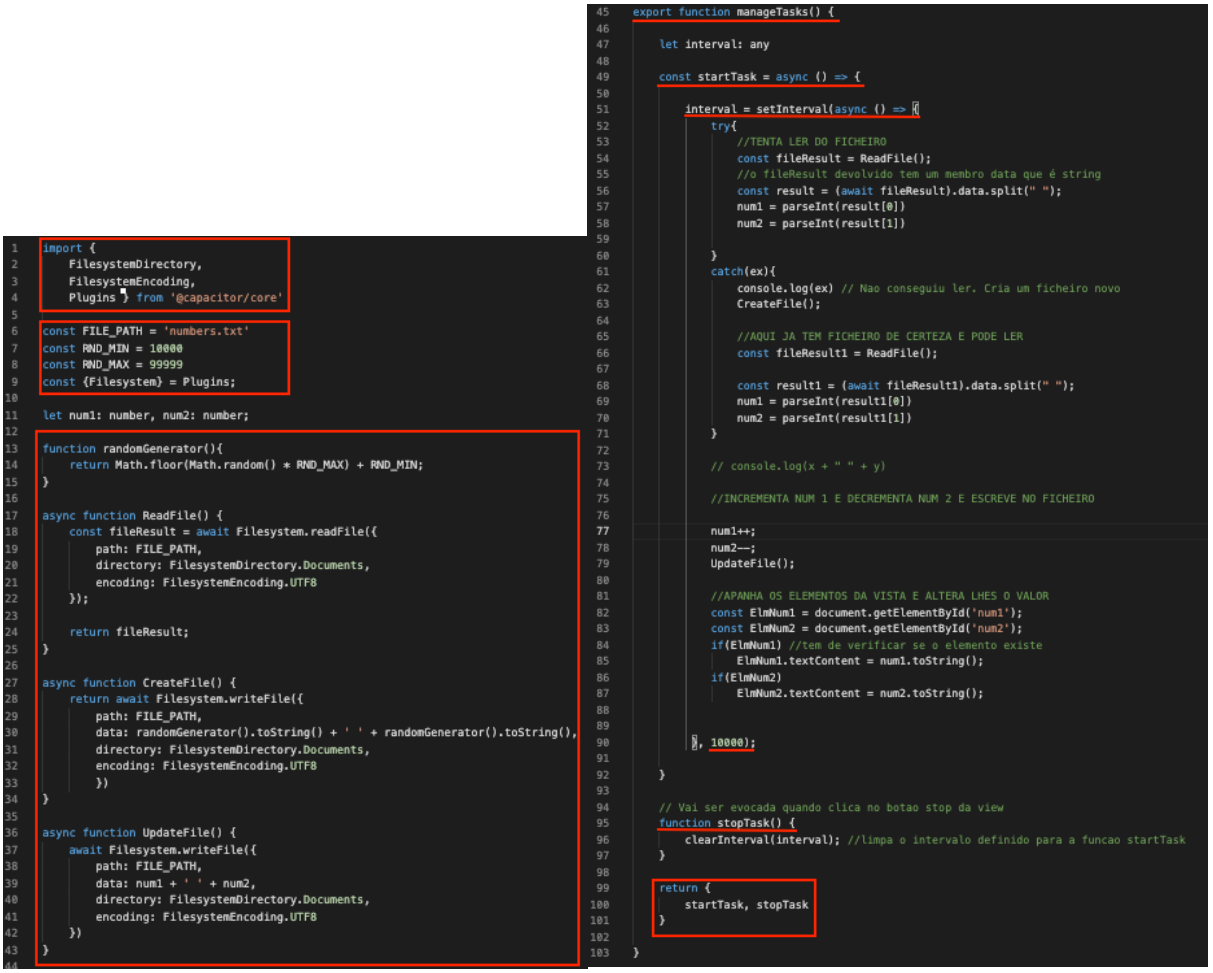
Home.vue:

```vue
<template>
  <ion-page>
    <ion-header :translucent="true">
      <ion-toolbar>
        <ion-title slot="start" color="primary" >Background Tasks</ion-title>
        <ion-buttons slot="start">
          <ion-button @click="startTask()" color="success" fill="solid">
            Start
          </ion-button>
          <ion-button @click="stopTask()" color="danger" fill="solid">
            Stop
          </ion-button>
        </ion-buttons>
      </ion-toolbar>
    </ion-header>

    <ion-content :fullscreen="true">
      <ion-list>
        <ion-item>
          <ion-label>Valor 1º numero:</ion-label>
          <ion-note slot="end" color="primary" id="num1">0</ion-note>
        </ion-item>
        <ion-item>
          <ion-label>Valor 2º numero:</ion-label>
          <ion-note slot="end" color="primary" id="num2">0</ion-note>
        </ion-item>
      </ion-list>
    </ion-content>
  </ion-page>
</template>

<script lang="ts">
import { IonButton, IonButtons,IonContent, IonHeader, IonPage, IonToolbar, IonIcon } from '@ionic/vue';
import { defineComponent } from 'vue';
import { manageTasks } from '@/composables/manageTasks';

export default defineComponent({
  name: 'Home',
  components: {
    IonButton,
    IonButtons,
    IonContent,
    IonHeader,
    IonPage,
    IonToolbar,
  },
  setup() {
    const {startTask, stopTask} = manageTasks();

    return {
      startTask,
      stopTask
    }
  }
});
</script>
```

- **Resumo dos componentes utilizados:**
    - **Ion-header:**
        - **Introduzido um titulo da página.**
        - **Dois botões para dar Start/Stop às Tarefas.**
            - **ion-header: Declarando que é o header da app**
            - **ion-toolbar: Criada uma toolbar para o titulo**
            - **ion-title: Declarando o titulo da app.**
    - **Ion-content:**
        - **Utilizada uma lista (ion-list) com 2 items para cada número a apresentar.**
            - **ion-label: Texto com o nome do numero que representa.**
            - **ion-note: Valor atual do numero.**

- **Resumo do funcionamento:**
    - Importação de todos os componentes a utilizar na UI (linha 33).
    - Importação do **manageTasks**, de modo a ter acesso aos seus métodos.
    - No **setup()** foi necesário declarar uma variável (tipo any) para cada função que vai ser utilizada do manageTasks() de modo a poderem ser chamadas na vista, fazerem a sua lógica e serem retornadas de volta para a vista.
    - Para cada botão Start/Stop foi adicionado um evento **onClick()** que vai chamar a sua respetiva função do **manageTasks().**

manageTasks.ts:



```
import {
    FilesystemDirectory,
    FilesystemEncoding,
    Plugins } from '@capacitor/core'

const FILE_PATH = 'numbers.txt'
const RND_MIN = 10000
const RND_MAX = 99999
const {Filesystem} = Plugins;

let num1: number, num2: number;

function randomGenerator(){
    return Math.floor(Math.random() * RND_MAX) + RND_MIN;
}

async function ReadFile() {
    const fileResult = await Filesystem.readFile({
        path: FILE_PATH,
        directory: FilesystemDirectory.Documents,
        encoding: FilesystemEncoding.UTF8
    });

    return fileResult;
}

async function CreateFile() {
    return await Filesystem.writeFile({
        path: FILE_PATH,
        data: randomGenerator().toString() + ' ' + randomGenerator().toString(),
        directory: FilesystemDirectory.Documents,
        encoding: FilesystemEncoding.UTF8
        })
}

async function UpdateFile() {
    await Filesystem.writeFile({
        path: FILE_PATH,
        data: num1 + ' ' + num2,
        directory: FilesystemDirectory.Documents,
        encoding: FilesystemEncoding.UTF8
    })
}
```

```
export function manageTasks() {

    let interval: any

    const startTask = async () => {

        interval = setInterval(async () => {
            try{
                //TENTA LER DO FICHEIRO
                const fileResult = ReadFile();
                //o fileResult devolvido tem um membro data que é string
                const result = (await fileResult).data.split(" ");
                num1 = parseInt(result[0])
                num2 = parseInt(result[1])

            }
            catch(ex){
                console.log(ex) // Nao conseguiu ler. Cria um ficheiro novo
                CreateFile();

                //AQUI JA TEM FICHEIRO DE CERTEZA E PODE LER
                const fileResult1 = ReadFile();

                const result1 = (await fileResult1).data.split(" ");
                num1 = parseInt(result1[0])
                num2 = parseInt(result1[1])
            }

            // console.log(x + " " + y)

            //INCREMENTA NUM 1 E DECREMENTA NUM 2 E ESCREVE NO FICHEIRO

            num1++;
            num2--;
            UpdateFile();

            //APANHA OS ELEMENTOS DA VISTA E ALTERA LHES O VALOR
            const ElmNum1 = document.getElementById('num1');
            const ElmNum2 = document.getElementById('num2');
            if(ElmNum1) //tem de verificar se o elemento existe
                ElmNum1.textContent = num1.toString();
            if(ElmNum2)
                ElmNum2.textContent = num2.toString();


        }, 10000);

    }

    // Vai ser evocada quando clica no botao stop da view
    function stopTask() {
        clearInterval(interval); //limpa o intervalo definido para a funcao startTask
    }

    return {
        startTask, stopTask
    }
}
```

- **Resumo do funcionamento:**
    - **Importação dos plugins do capacitor para manipulação de ficheiros.**
    - **Definição de variáveis constantes a serem utilizadas nos métodos.**
    - **Implementação dos seguintes métodos auxiliares:**
        - **randomGenerator():** Retorna um numero aleatório entre um valor minimo e máximo.
        - **ReadFile():** Le um ficheiro **"FILE_PATH"** que é especificado nas constantes, armazenado no diretório "Documentos" do dispositivo e retorna o seu resultado (**FileReadResult**).
        - **CreateFile():** Cria um ficheiro com dois valores aleatórios entre 10000 e 99999, separador por 1 caractêr espaço, no diretório "Documentos" do dispositivo com o nome especificado na constante **"FILE_PATH".**
        - **UpdateFile():** Atualiza o ficheiro **"FILE_PATH"** onde guarda o valor atual do num1 e do num2, separados por um caractêr espaço.

    - **Criação do método principal manageTasks()** que vai ter a caracteristica **"export"** de modo a que os seus métodos possam ser exportados para a view. **Para isso é necessário retornar os seus métodos.**
    - **Implementação de um intervalo de 10000ms (10 segundos)** onde o método **startTask()** vai estar sempre a ser executado nessas condições até que o utilizador clique no botão de

Stop() e seja executado o método **stopTasks()** que vai limpar o intervalo e parar a execução do **startTask().**

Bibliografia

https://ionicframework.com/docs/intro/cli

https://ionic.zone/capacitor/overview

https://ionicframework.com/docs/angular/lifecycle

https://www.youtube.com/watch?v=OVIrtvciCDE&t

https://www.youtube.com/watch?v=8JGa44vdQ8Y&t

https://www.youtube.com/watch?v=5QqvO_9LPzQ