

## Curso Técnico Superior Profissional em Tecnologias e Programação de Sistemas de Informação

Unidade Curricular: Desenvolvimento Web – Front-End

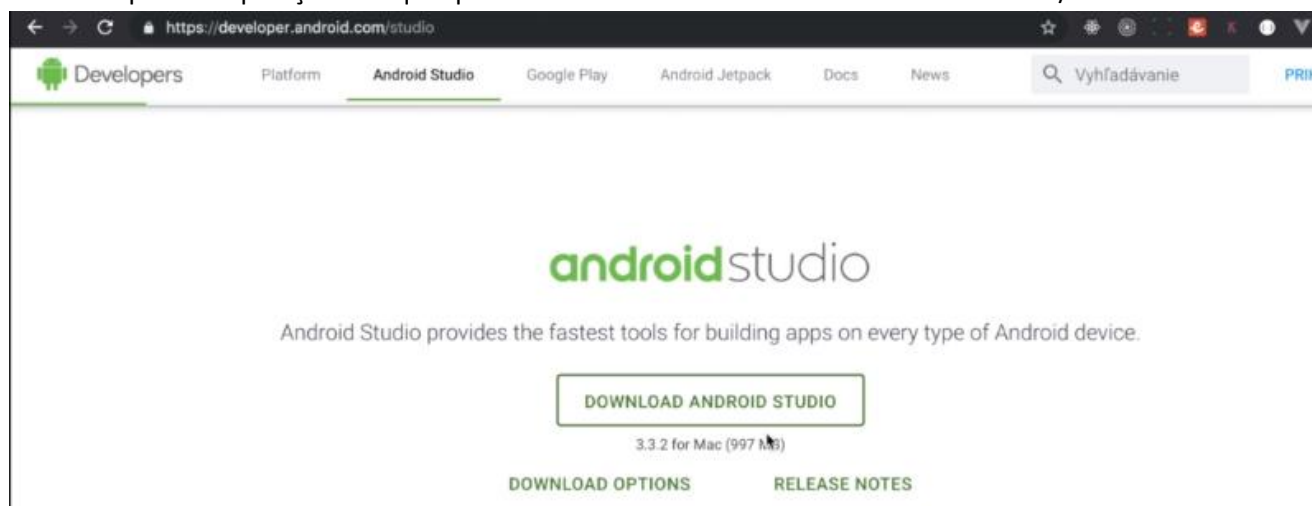
1.º Ano/2.º Semestre

Docente: Marco Miguel Olival Olim

Data 10/05/2019

### ESTE EXERCÍCIO INTRODUZ A ABORDAGEM NATIVA PARA A CRIAÇÃO DE APLICAÇÕES PARA DISPOSITIVOS MÓVEIS

As aplicações realizadas com componentes nativos não são aplicações web para dispositivos móveis uma vez que utilizam os mesmos blocos de Interface para o utilizador que uma aplicação Android/iOS. Neste caso, em vez de usar Java ou Swift estamos a usar Javascript. Por serem específicas para dispositivos móveis, estas aplicações não funcionam na web como nas estratégias anteriores. A implementação mais popular é o **React Native** e iremos abordar o seu port para VueJS. Outra abordagem distinta é com **NativeScript** que será analisada para comparação. Em qualquer um dos casos é necessário o SDK do Android e/ou XCode:



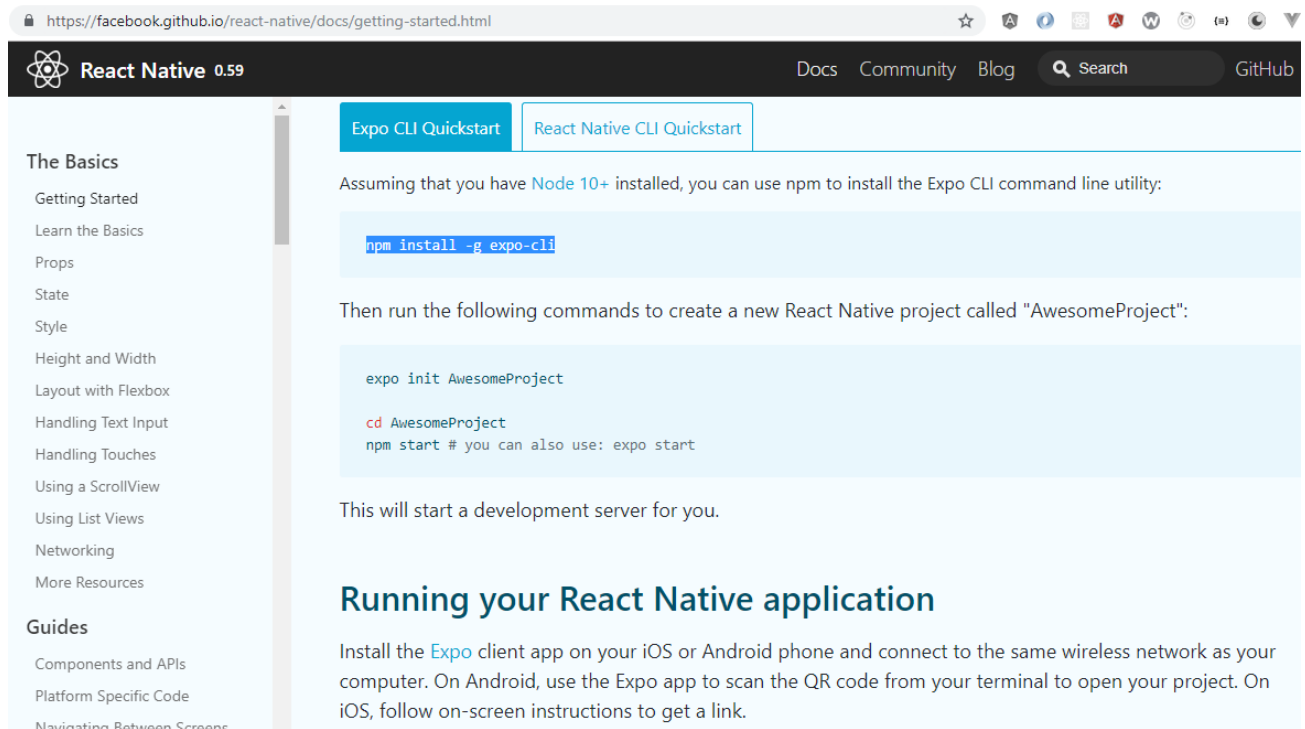
### ESTRATÉGIA 1: VUE NATIVE

instalamos o **React Native** com `npm install -g react-native-cli`



Cofinanciado por:

Para testar as aplicações sem ser no emulador, dispomos do EXPO CLI que gera um código QR para usar a app no telemóvel a partir de mesma rede wireless. Instale também globalmente com **npm install -g expo-cli**



The screenshot shows the React Native 0.59 documentation page. The left sidebar contains a navigation menu with sections: 'The Basics' (Getting Started, Learn the Basics, Props, State, Style, Height and Width, Layout with Flexbox, Handling Text Input, Handling Touches, Using a ScrollView, Using List Views, Networking, More Resources) and 'Guides' (Components and APIs, Platform Specific Code, Navigating Between Screens). The main content area is titled 'Expo CLI Quickstart' and includes the following text: 'Assuming that you have Node 10+ installed, you can use npm to install the Expo CLI command line utility:'. Below this is a code block: 

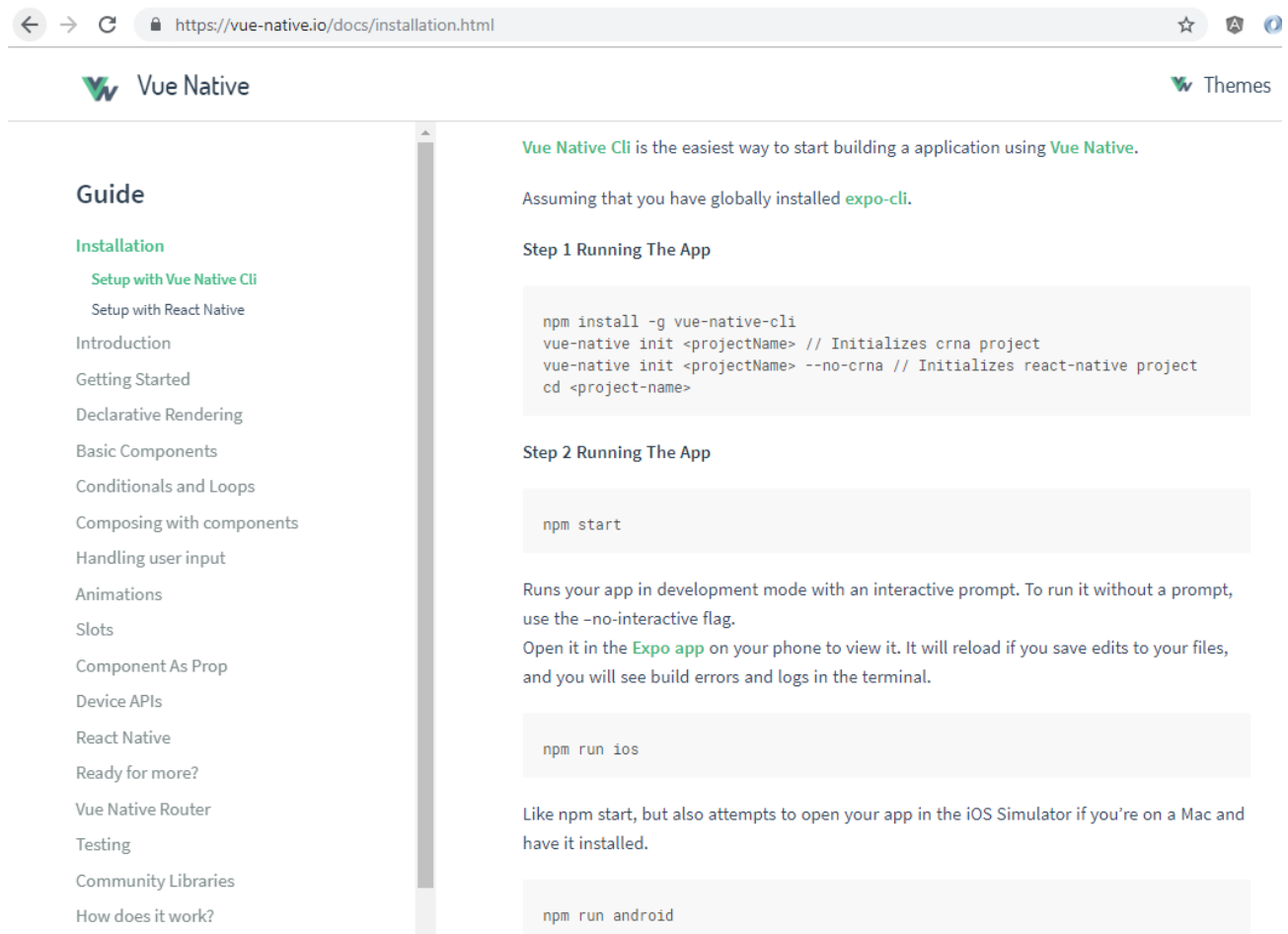
```
npm install -g expo-cli
```

. Then it says: 'Then run the following commands to create a new React Native project called "AwesomeProject":'. Below this is another code block: 

```
expo init AwesomeProject
cd AwesomeProject
npm start # you can also use: expo start
```

. It then states: 'This will start a development server for you.' The next section is 'Running your React Native application', which says: 'Install the Expo client app on your iOS or Android phone and connect to the same wireless network as your computer. On Android, use the Expo app to scan the QR code from your terminal to open your project. On iOS, follow on-screen instructions to get a link.'

Por último vamos utilizar o Vue em vez do React para programar. A aplicação, no entanto, continua a ser gerada pelo React Native. Instalamos este CLI com **npm install -g vue-native-cli**



The screenshot shows the Vue Native documentation page. The left sidebar contains a navigation menu with sections: 'Guide' (Installation, Setup with Vue Native Cli, Setup with React Native, Introduction, Getting Started, Declarative Rendering, Basic Components, Conditionals and Loops, Composing with components, Handling user input, Animations, Slots, Component As Prop, Device APIs, React Native, Ready for more?, Vue Native Router, Testing, Community Libraries, How does it work?). The main content area is titled 'Vue Native CLI is the easiest way to start building a application using Vue Native.' It then says: 'Assuming that you have globally installed expo-cli.' The next section is 'Step 1 Running The App', which includes a code block: 

```
npm install -g vue-native-cli
vue-native init <projectName> // Initializes crna project
vue-native init <projectName> --no-crna // Initializes react-native project
cd <project-name>
```

. The next section is 'Step 2 Running The App', which includes a code block: 

```
npm start
```

. It then says: 'Runs your app in development mode with an interactive prompt. To run it without a prompt, use the -no-interactive flag. Open it in the Expo app on your phone to view it. It will reload if you save edits to your files, and you will see build errors and logs in the terminal.' Below this is another code block: 

```
npm run ios
```

. It then says: 'Like npm start, but also attempts to open your app in the iOS Simulator if you're on a Mac and have it installed.' Below this is a final code block: 

```
npm run android
```

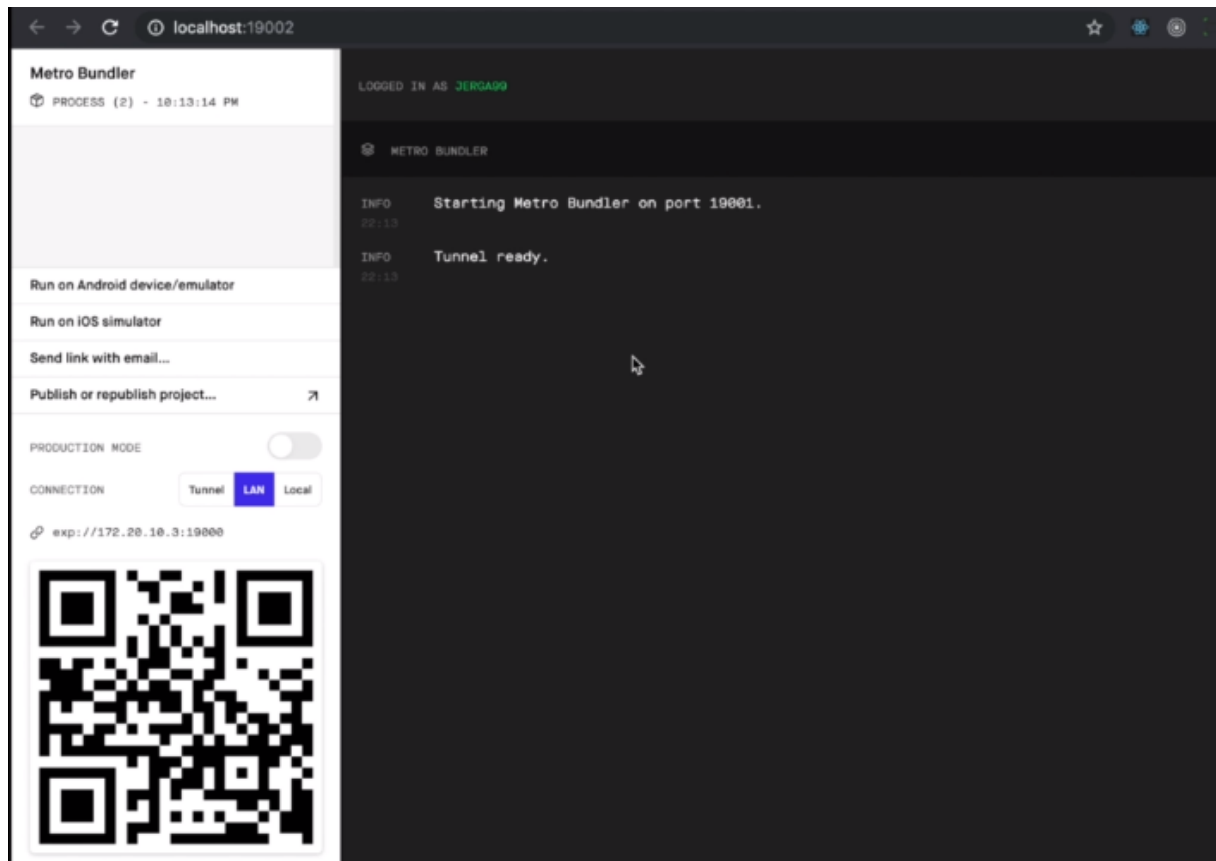
Cofinanciado por:



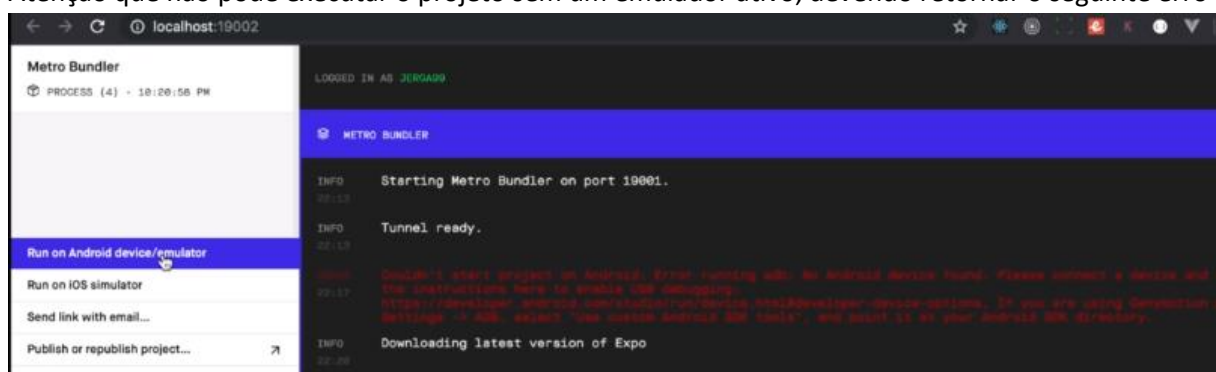
Para criar uma aplicação usamos **vue-native init nomeprojeto**

```
Creating Vue-Native vue-meetuper-mobile-cli App
Creating Crna vue-meetuper-mobile-cli project
? Please enter a few initial configuration values.
Read more: https://docs.expo.io/versions/latest/workflow/configuration/ 50% completed
{
  "expo": {
    "name": "<The name of your app visible on the home screen>",
    "slug": "vue-meetuper-mobile-cli"
  }
}
```

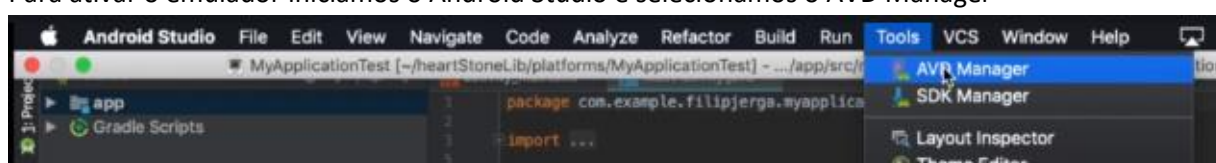
Para iniciarmos o projeto criado usamos de seguida **npm start** para abrir num browser



Atenção que não pode executar o projeto sem um emulador ativo, devendo retornar o seguinte erro



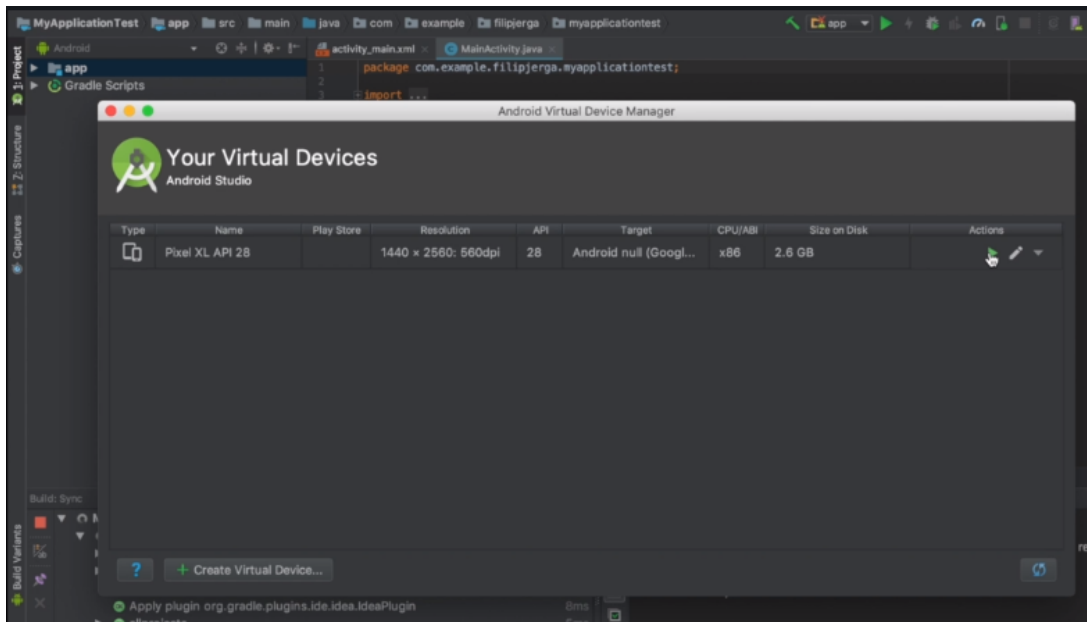
Para ativar o emulador iniciamos o Android Studio e seleccionamos o AVD Manager



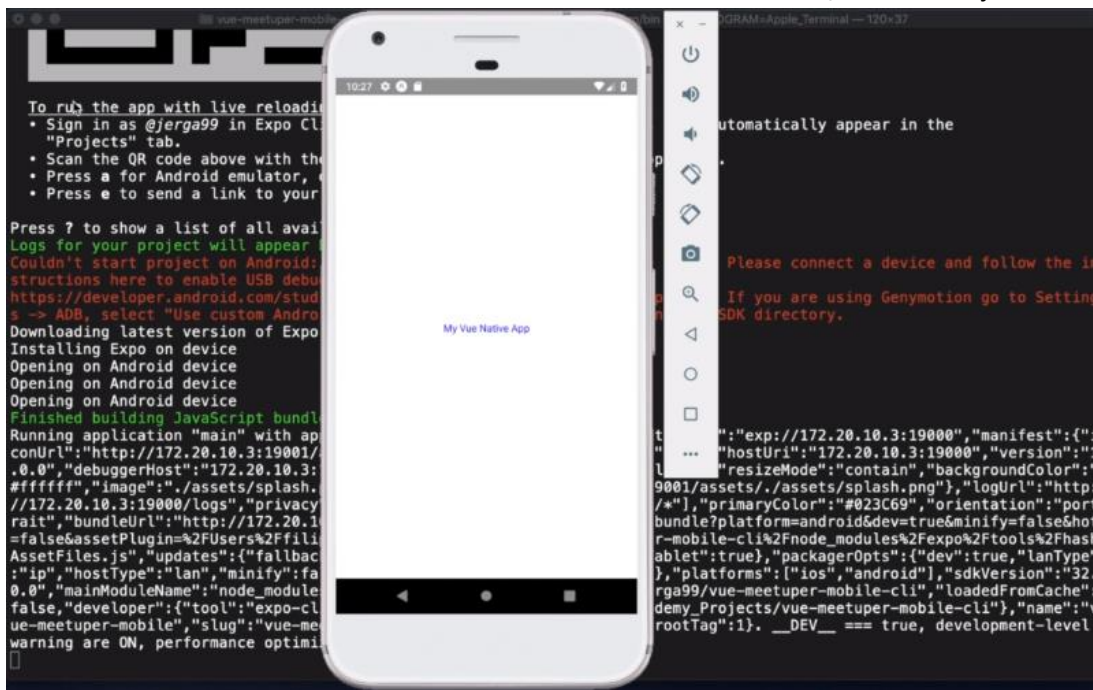
Cofinanciado por:



Execute agora um dos dispositivos que dispõe. No caso de não ter nenhum, crie um novo dispositivo



De volta ao vue-native efetuamos novamente **run on Android device/emulator** e já deverá ver a app



Vamos agora analisar o projeto. A estrutura das pastas é semelhante ao Vue CLI e também temos o App.vue



Cofinanciado por:



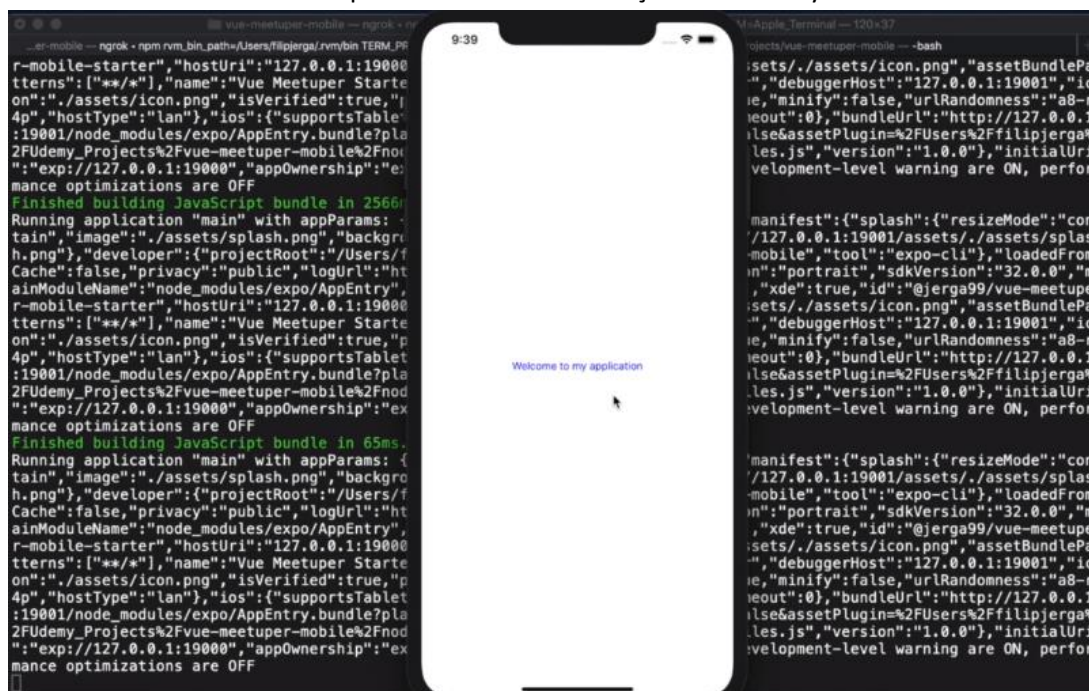
Pode agora modificar o *single file component* da mesma forma do que no Vue CLI.

```
<template>
  <view class="container">
    <text class="text-color-primary">{{message}}</text>
  </view>
</template>

<script>
  export default {
    data () {
      return {
        message: 'Welcome to my application'
      }
    }
  }
</script>


<style>
  .container {
    background-color: white;
    align-items: center;
    justify-content: center;
    flex: 1;
  }
  .text-color-primary {
    color: blue;
  }
</style>
```

Como o emulador tem HMR pode visualizar as alterações “on the fly” sem reiniciar o emulador



Note que os componentes de UI têm uma notação um pouco diferente, sendo por isso aconselhável consultar a documentação em <https://vue-native.io/docs/basic-components.html>

← → ↺ https://vue-native.io/docs/basic-components.html#Button ☆ A

 Vue Native

Themes

Basic Components

View

Text

Image

TextInput

ScrollView

Button

FlatList

ActivityIndicator

Alert

StatusBar

Switch

TouchableOpacity

WebView

Further Details

# Button

A basic button component that should render nicely on any platform. Supports a minimal level of customization.

We can do more than just bind to data and style views with Vue Directives.

This button has an event handler `on-press` that we are binding to. When that event gets fired, we run a method on our Vue Instance.

```
<template>
  <button
    :on-press="onPressLearnMore"
    title="Learn More"
    color="#841584"
    accessibility-label="Learn more about this purple button"
  />
</template>
```

Cofinanciado por:



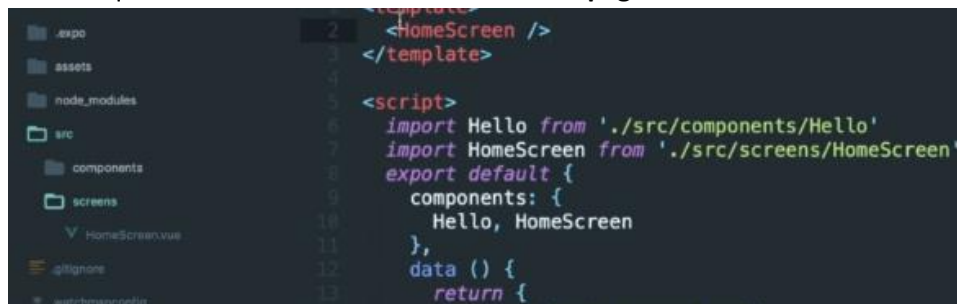
Para ilustrar a particularidade da notação destes componentes de UI vamos adaptar o exemplo de implementação de um botão que incrementa um contador

```
<template>
  <view>
    <text class="text-color-primary">{{message}}</text>
    <button :title="btnMessage" :on-press="handleClick" />
  </view>
</template>

<script>
  export default {
    data () {
      return {
        message: 'Welcome to my application',
        btnMessage: 'Click Me',
        clickCount: 0
      }
    },
    methods: {
      handleClick () {
        this.clickCount = this.clickCount + 1
        alert('I am Clicked! Click Count: ${this.clickCount}')
      }
    }
  }
</script>

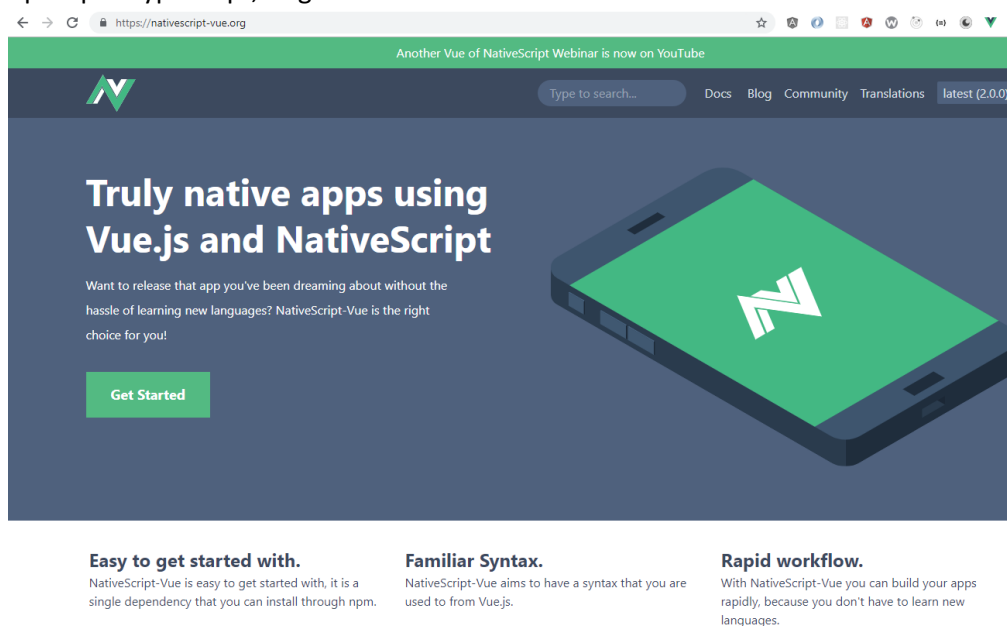
<style>
  .text-color-primary {
    color: blue;
  }
</style>
```

Numa última nota refira-se que não é possível implementar rotas como no Vue CLI, mas podemos designar **screens** que funcionam de forma semelhante às **pages** no **NUXT**



## ESTRATÉGIA 2: NATIVESCRIPT-VUE

O NativeScript é uma framework de criação de Aplicações Móveis com Javascript e que também permite optar por Typescript, Angular ou VueJS



Another Vue of NativeScript Webinar is now on YouTube

Type to search... Docs Blog Community Translations latest (2.0.0)

# Truly native apps using Vue.js and NativeScript

Want to release that app you've been dreaming about without the hassle of learning new languages? NativeScript-Vue is the right choice for you!

[Get Started](#)

**Easy to get started with.**  
NativeScript-Vue is easy to get started with, it is a single dependency that you can install through npm.

**Familiar Syntax.**  
NativeScript-Vue aims to have a syntax that you are used to from Vue.js.

**Rapid workflow.**  
With NativeScript-Vue you can build your apps rapidly, because you don't have to learn new languages.

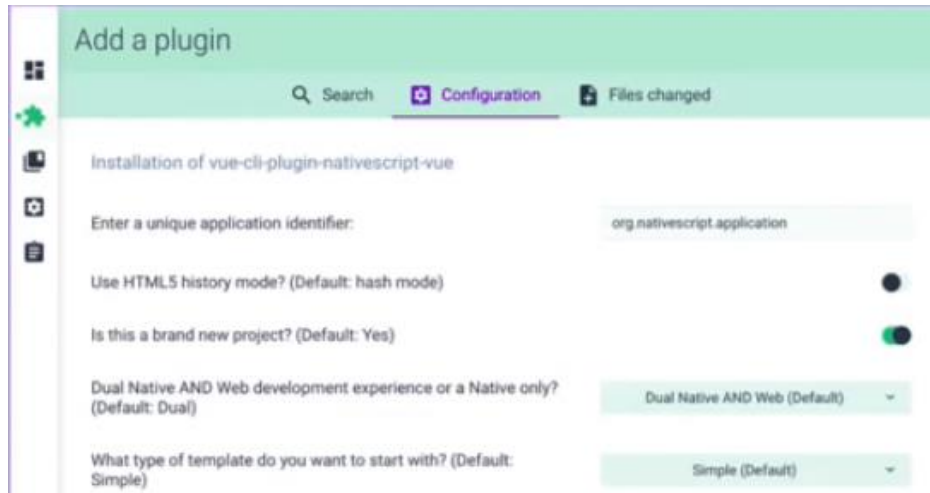
Cofinanciado por:



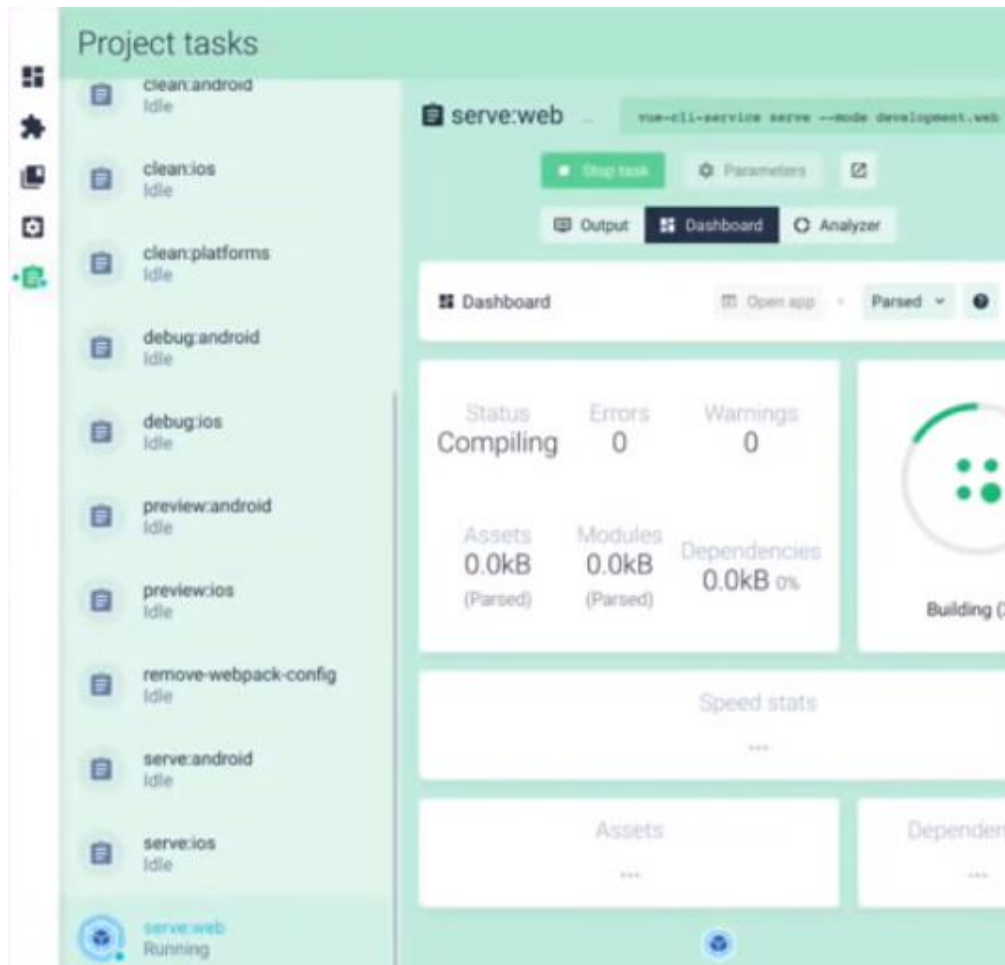
O NativeScript-Script usa o Vue CLI para gerir os projetos ao contrário do Vue Native, que usa o CLI do React Native. Sendo assim, crie um projeto com o Vue UI e depois adicione o plugin **vue-cli-plugin-nativescript-vue**



Efetue as alterações necessárias na configuração do plugin

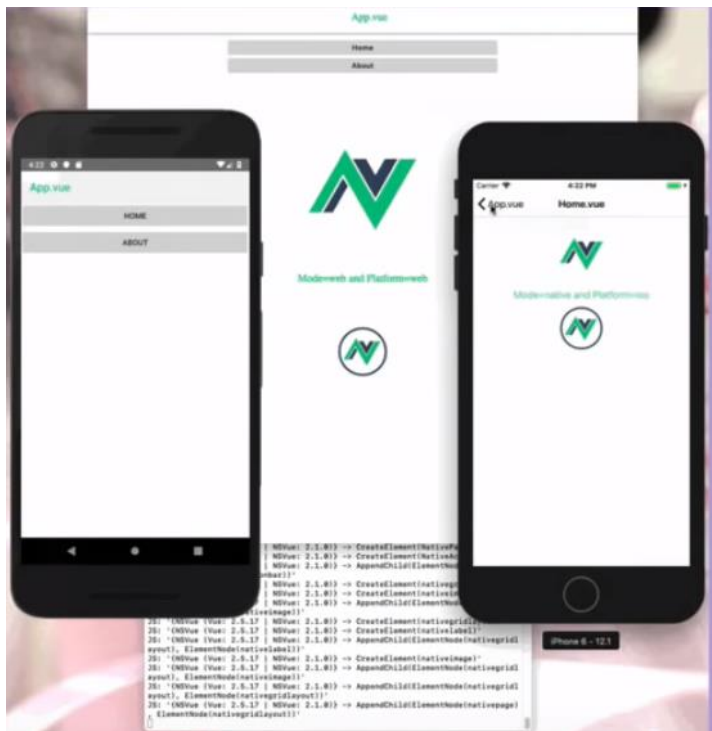


Pode agora iniciar o simulador para desenvolvimento em android, ios ou web



Cofinanciado por:

Se tiver os emuladores instalados (android studio e xcode), como já foi referido anteriormente a aplicação deve iniciar em cada um destes:



Caso não funcione, verifique primeiro se tem todas as dependências instaladas no **package.json**

```
"dependencies": {
  "nativescript-vue": "^2.2.0",
  "tns-core-modules": "^5.2.2",
  "vue": "^2.6.10",
  "vue-router": "^3.0.2",
  "vuex": "^3.1.0"
},
```

Note que o main.js tem de ser adaptado para usar aplicações nativas:

```
import Vue from 'vue';
import App from './App.vue';
import router from './router';
import store from './store';

Vue.config.productionTip = false;

new Vue({
  router,
  store,
  render: (h) => h(App)
}).$mount('#app');
```

```
import Vue from 'nativescript-vue';
import App from './App.vue';
import store from './store';

Vue.config.silent = false;

new Vue({
  store,
  render: (h) => h('frame', [h(App)])
}).$start();
```

A notação dos componentes nativos é mais semelhante ao XML, tal como acontece no android

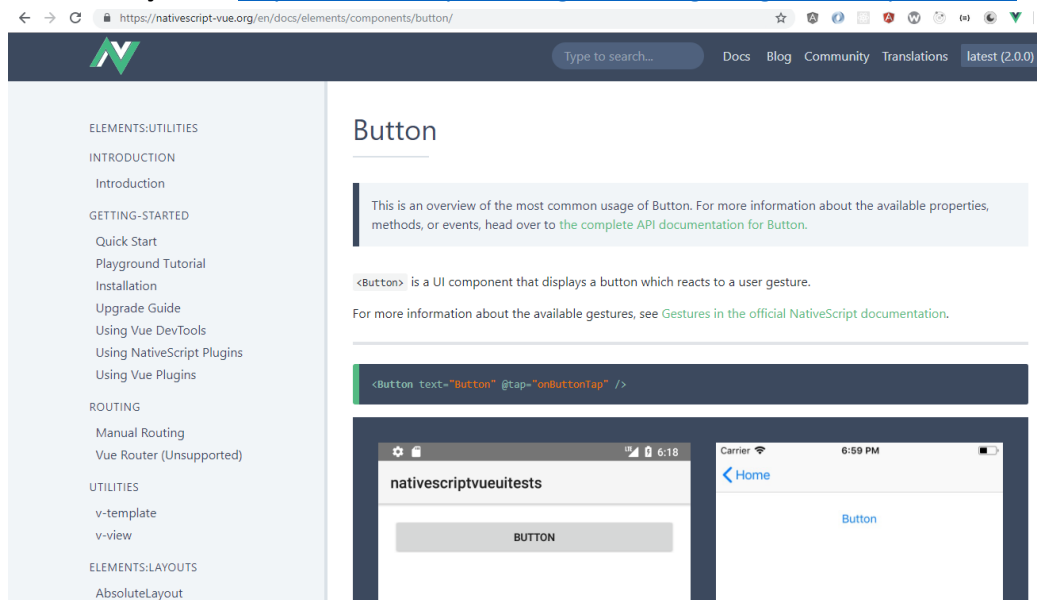
```
<template web>
  <div class="w-page">
    <nav>
      <ul class="w-navbar">
        <li class="w-title">{{navbarTitle}}</li>
      </ul>
    </nav>
    <div class="w-container">
      <router-link tag="button" class="w-button" id="homeButton" to="/">Home</router-link>
      <button class="w-button" id="aboutButton" v-on:click="goToAboutPage">About</button>
      <router-view/>
    </div>
  </div>
</template>

<template native>
  <Page>
    <ActionBar :title="navbarTitle"/>
    <GridLayout rows="auto, auto">
      <Button text="Home" @tap="goToHomePage" row="0"/>
      <Button text="About" @tap="goToAboutPage" row="1"/>
    </GridLayout>
  </Page>
</template>
```

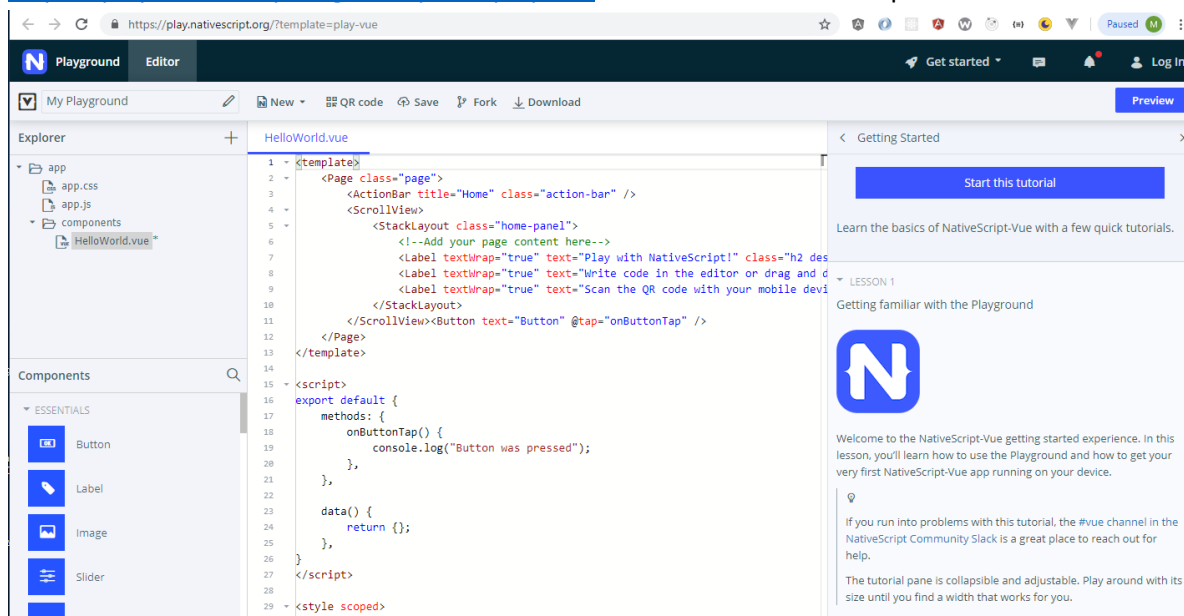
Cofinanciado por:



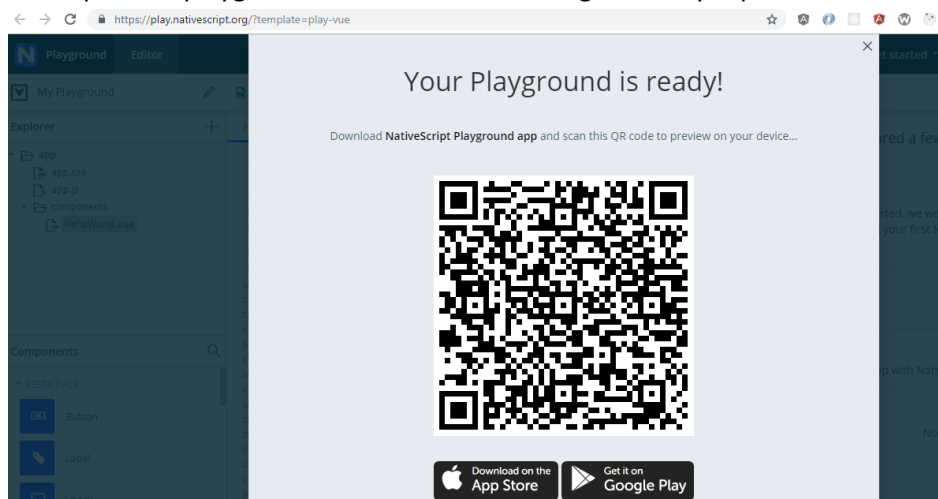
Como os componentes de UI têm essa notação um pouco diferente, é por isso aconselhável consultar a documentação em <https://nativescript-vue.org/en/docs/getting-started/quick-start/>



Está também disponível um playground para testar online os diferentes componentes em <https://play.nativescript.org/?template=play-vue> e é uma boa alternativa para testar em vez do CLI.



Note que este playground até funciona com drag-and-drop e permite simular no telemóvel com código QR



Cofinanciado por:

