

Capítulo

13

Desenvolvimento de Aplicativos Híbridos com o Ionic Framework

Allan Jheyson Ramos Gonçalves

Abstract

An application is called hybrid when, from the development of a single source code project, you can export this application to various platforms. This modality dispenses the knowledge necessary for the specific development in each platform, such as programming languages, development kits (SDKs), frameworks and IDEs. Ionic is an open source framework for the development of hybrid applications for mobile devices using HTML, CSS and JavaScript. The purpose of this mini-course is to provide the student with the necessary knowledge to develop hybrid applications running on Android, iOS and Windows Phone using the Ionic Framework in its current version.

Resumo

Uma aplicação é denominada híbrida quando, a partir do desenvolvimento de um único projeto de código-fonte, pode-se exportar esta aplicação para diversas plataformas. Esta modalidade dispensa o conhecimento necessário para o desenvolvimento específico em cada plataforma, como linguagens de programação, kits de desenvolvimento (SDKs), frameworks e IDEs. O Ionic é um framework open source para o desenvolvimento de aplicações híbridas para dispositivos móveis utilizando HTML, CSS e JavaScript. O objetivo deste minicurso é proporcionar ao aluno o conhecimento necessário para o desenvolvimento de aplicações híbridas que rodam em Android, iOS e Windows Phone utilizando o Ionic Framework em sua versão atual.

13.1. Introdução

Nos últimos anos houve um crescimento considerável do número de dispositivos móveis em uso em todo o mundo. As pessoas mudaram seus hábitos de comunicação e consumo, dentre outros aspectos. Com este cenário, visando suprir as necessidades dos clientes, as empresas estão investindo cada vez mais em soluções de software que estejam

rapidamente acessíveis para os consumidores. Desta forma, os aplicativos para dispositivos móveis tornaram-se essenciais para o mundo dos negócios.

Visando suprir a demanda, os profissionais da área de desenvolvimento de software devem estar cada vez mais capacitados para a produção de softwares funcionais, com boa interface gráfica de usuário e baixo consumo de recursos computacionais. Estes aspectos devem ser alcançados em paralelo ao baixo custo de produção, que em grande parte dos projetos podem ser acentuados de acordo com o tempo gasto durante o desenvolvimento.

A produtividade é uma das características mais valorizadas no processo de desenvolvimento de software, por esta razão, o desenvolvimento de aplicativos híbridos pode ser a melhor alternativa para o mercado atual, pois reduz os custos com o tempo de desenvolvimento e os recursos humanos necessários, visto que nesta modalidade não são necessários profissionais dedicados ao desenvolvimento específico para cada plataforma de software disponível no mercado.

Existem no mercado outras ferramentas para desenvolvimento híbrido como o jQuery Mobile e o Sencha Touch, porém a grande aceitação do Ionic se deve à constante preocupação de sua equipe com a otimização da performance de suas aplicações. Quanto mais fluida, mais responsiva e com desempenho mais próximo de uma aplicação nativa, melhor serão os feedbacks das aplicações híbridas e do próprio framework.

O Ionic Framework é uma alternativa em ascensão no mercado pela sua alta produtividade no desenvolvimento de aplicativos e a facilidade na manipulação de seus recursos, visto que muitos profissionais da área de desenvolvimento aprendem ainda na academia os conceitos utilizados pelo framework. Através da combinação de JavaScript, HTML e CSS, um mesmo projeto de código pode ser exportado como aplicativo para Android, iOS e Windows Phone com simples comandos disponibilizados pelo framework.

13.2. Visão geral do Ionic Framework

A escolha de um framework de desenvolvimento de software mobile é uma das etapas mais importantes no processo de criação de novos produtos e serviços. A complexidade e requisitos do projeto são aspectos fundamentais a serem analisados antes da escolha de um framework de desenvolvimento, pois estes poderão determinar completamente a escolha ou eliminação de algumas alternativas existentes no mercado.

A variedade de dispositivos móveis e de sistemas operacionais existentes impõe um grande desafio aos desenvolvedores, pois estes devem criar softwares capazes de funcionar perfeitamente em todos estes equipamentos. A indústria dos dispositivos móveis não segue um padrão na produção dos recursos disponíveis em seus produtos, assim, um mesmo smartphone, por exemplo, pode conter recursos como câmera, sensores e atuadores de diferentes fabricantes, e ainda, a mesma fábrica pode utilizar recursos de fabricantes diferentes em outros modelos de smartphones, aumentando ainda mais a diversidade de requisitos que deve ser atendida pelos novos aplicativos.

Uma aplicação híbrida é capaz de atender boa parte dos requisitos esperados para um aplicativo para dispositivos móveis. Diferente dos aplicativos nativos, que são desenvolvidos com as linguagens padrão para cada sistema operacional, os aplicativos

híbridos são desenvolvidos com os recursos de um sistema web, que podem ser empacotados e distribuídos nas lojas de aplicativos como se fossem nativos.

Todo o código produzido em uma aplicação híbrida irá ser executado dentro de um recurso chamado webview, um tipo especial de browser que é executado quando a aplicação híbrida é requisitada pelo usuário. Essa forma de execução sobre o webview não fica explícita para o usuário, deixando a impressão de que o aplicativo se comporta da mesma forma que um aplicativo nativo. O nome híbrido é dado justamente pela característica da junção de código nativo para empacotamento e distribuição do aplicativo com o código não nativo (HTML, CSS e JavaScript), responsável pelo visual e funcionalidades da aplicação.

Para que uma aplicação seja bem aceita pelo público-alvo, é necessário que esta atenda a alguns requisitos básicos, dentre eles: boa interface gráfica de usuário. É exatamente neste ponto que o Ionic Framework se destaca. Existe uma gama de recursos gráficos prontos que o desenvolvedor pode simplesmente declarar em seu código e rapidamente criar uma aplicação com interface gráfica amigável e funcional.

Durante o desenvolvimento o programador pode ainda definir estilos personalizados em cada tela, pois graças ao Angular, todos os componentes possuem um escopo próprio de configurações de aparência e funcionalidades. Para criar uma nova tela na aplicação, adicionar um serviço, exportar a aplicação, dentre outras funcionalidades, o Ionic disponibiliza uma série de comandos que geram e organizam adequadamente os arquivos de código.

O Ionic foi criado em 2013 pela Drifty Co. O foco do projeto foi a construção de uma estrutura que se concentrasse no desempenho com os padrões modernos da web. Desde então foram lançadas algumas versões com mudanças significativas entre a primeira versão e a versão atual. Em 2015 foram criados mais de 1,3 milhão de aplicativos com o framework.

Várias plataformas são suportadas pelo Ionic, com ênfase para o Android (a partir da versão 4.1) e iOS (versão 7 e superiores). É possível ainda criar aplicações para a plataforma Windows e Blackberry 10. O desempenho das aplicações híbridas desenvolvidas com o Ionic é melhorado devido à utilização do Angular (ao invés de jQuery), que alavanca as transições CSS transformando-as para animações afim de aproveitar melhor a GPU e otimizar o tempo de processador do dispositivo.

13.2.1. Cordova

As plataformas para dispositivos móveis existentes fornecem aos desenvolvedores as ferramentas necessárias para que possam utilizar recursos nativos dos dispositivos, como câmera, sensor biométrico, etc. No desenvolvimento híbrido, ao invés de dar manutenção em Java (Android), Swift (iOS) e C# (Windows Phone), o desenvolvedor irá se dedicar a apenas uma base de código. Para que todos os recursos nativos dos dispositivos possam ser acessados em diferentes plataformas, o Ionic Framework trabalha em conjunto com um outro framework, o Apache Cordova.

Cordova é uma plataforma de desenvolvimento móvel com APIs (Application Programming Interface) que permitem que o desenvolvedor acesse funções nativas do dispositivo, como câmera ou acelerômetro. Quando se cria uma aplicação com o Ionic, por padrão não são adicionados plugins do Cordova para que o aplicativo não fique

sobrecarregado com plugins que não serão utilizados. O desenvolvedor pode adicionar plugins em sua aplicação de acordo com a necessidade através de alguns comandos simples e uma conexão com a internet.

Em síntese, pode-se dizer que o Ionic Framework trabalha sobre o Cordova, ou seja, o Cordova é uma camada inferior, responsável pelo acesso aos recursos nativos dos dispositivos e pelo empacotamento da aplicação para as diferentes plataformas. O Ionic é a camada superior, focada na interface gráfica de usuário e regras de negócios da aplicação.

13.2.2. Angular

Visando abstrair a manipulação do DOM (Document Object Model), separando a lógica da view/template do aplicativo, o Ionic também conta com o Angular, que se trata de um framework open source mantido pela Google e comunidade. A programação pode ser realizada em Javascript/Typescript sob o padrão MVVM (model-view-view-model), com o objetivo de facilitar tanto o desenvolvimento quanto o teste unitário dos aplicativos.

No conteúdo HTML das páginas (views) do aplicativo são utilizadas tags especiais do framework que, ao serem localizadas dão início à execução do componente correspondente à tag. O vínculo entre o modelo e a view é representado por variáveis e funcionalidades Typescript, linguagem padrão do framework, apesar de ser possível também se trabalhar com Javascript.

A utilização do Angular deixa o projeto mais organizado e funcional. A manutenção do código e a realização de testes da aplicação tornam-se mais objetivas, pois há uma descentralização de recursos, graças ao padrão MVVM.

13.3. Preparando o ambiente

Para iniciar o desenvolvimento com o Ionic Framework é necessário preparar o ambiente instalando os recursos necessários no computador que será utilizado para a atividade. O procedimento é simples, bastando o usuário fazer o download de alguns programas necessários para que o Ionic possa ser instalado e configurado corretamente. Para a instalação é necessário que o computador tenha acesso à internet, pois durante o processo o framework irá fazer o download de todas as dependências necessárias. O tempo de instalação depende da velocidade instantânea da conexão.

O primeiro recurso a ser instalado é o NodeJS, uma plataforma construída sobre o motor JavaScript do Google Chrome que auxilia na criação rápida de aplicações escaláveis. O download da última versão pode ser realizado no site oficial (<https://nodejs.org>), de acordo com o sistema operacional utilizado pelo usuário. Para instalar basta executar o arquivo baixado e seguir os passos descritos pelo instalador. Este procedimento também irá realizar a instalação do NPM (Node Package Manager), um gerenciador de módulos JavaScript necessário para realizar a instalação do Ionic.

Prosseguindo com a preparação do ambiente, instala-se o Git, o famoso sistema de gerenciamento de código fonte e controle de versão distribuído. Assim como muitos projetos open source, o Ionic está hospedado no GitHub, um servidor de Git gratuito, por este motivo é necessária sua instalação na máquina de desenvolvimento. O download para o sistema operacional utilizado pode ser realizado através do link: “<https://git-scm.com/downloads>”.

Após a instalação destes recursos, o ambiente está apto para o próximo passo: a instalação do Cordova e do Ionic. O processo é simples e utiliza o NPM para instalar os módulos. Os comandos e exemplos abaixo são realizados em ambiente Windows, caso seja utilizada outra plataforma, basta utilizar os programas equivalentes, adicionando os comandos de permissão de administrador. Com o computador conectado à internet, basta abrir um terminal de comando (CMD, no Windows) com permissão de administrador e digitar o seguinte comando:

```
npm install -g cordova ionic
```

Figura 13.1 - Comando de instalação dos Frameworks Cordova e Ionic

A instalação pode demorar alguns minutos, a depender da velocidade da conexão com a internet no momento da instalação. Ao final do procedimento aparecerão algumas mensagens de confirmação no terminal. Para verificar se o procedimento foi realizado corretamente, basta executar o comando abaixo, deve ser exibido o número da versão atual do Ionic Framework:

```
ionic --version
```

Figura 13.2 - Comando de verificação da instalação do framework

Caso ocorra algum erro durante este procedimento, basta repetir a operação, assim os pacotes incompletos ou serão sobrepostos e tudo funcionará corretamente. Concluídos estes passos o ambiente de desenvolvimento está pronto para ser explorado. O tópico a seguir irá descrever os passos de criação e configuração de um novo aplicativo, apresentando sugestões de ferramentas de edição de código e testes da aplicação em desenvolvimento nas plataformas suportadas pelo Ionic. Um ponto importante a considerar é que, para realizar o build da aplicação para o Android é necessário possuir o Android SDK e o Java instalados na máquina e, para o build no iOS deve-se proceder a instalação do Xcode previamente.

13.4. Criando o primeiro aplicativo

A partir da instalação do Ionic, ficam disponíveis todos os comandos de manipulação de seus recursos através do CLI (Command Line Interface), que pode ser utilizado através de um terminal do SO. Desde a criação do projeto, criação de novas telas, inclusão de remoção de recursos, testes, todas as etapas do desenvolvimento podem ser realizadas por linha de comando. Recomenda-se o uso de um editor de texto para manipulação da interface gráfica e regras de negócios da aplicação, nesta seção será feita uma sugestão para esta finalidade.

Para facilitar a parte inicial do desenvolvimento de um aplicativo, o Ionic disponibiliza alguns templates funcionais, que podem ser editados e adaptados para atender aos requisitos desejados pelo desenvolvedor, o que facilita e agiliza bastante a fase inicial da produção. A sintaxe do comando de criação especifica o nome do aplicativo, o template a ser utilizado e pode especificar a versão do Ionic que será utilizada no projeto, pois houve grandes mudanças da versão inicial do framework em relação à versão atual e existem muitos aplicativos e desenvolvedores que ainda utilizam a primeira versão.

Os templates disponíveis são:

- **sidemenu**: adiciona um menu de navegação na lateral esquerda da aplicação (padrão no Android);
- **tabs**: navegação baseada em guias (padrão de organização do iOS);
- **blank**: projeto básico, sem nenhum template específico.
- **super**: projeto inicial completo com páginas pré-construídas, provedores e melhores práticas para o desenvolvimento.
- **conference**: projeto que demonstra uma aplicação do mundo real.
- **tutorial**: projeto tutorial baseado na documentação do Ionic.

No mês de abril de 2017 foi lançada a versão 3 do Ionic, assim, nos exemplos a seguir serão utilizados conceitos e recursos desta versão. O comando de criação de um projeto no Ionic possui algumas opções disponíveis ao usuário, possibilitando, inclusive, a criação de um projeto baseado na primeira versão do framework. É importante salientar que o projeto será criado no caminho atual do terminal em que o comando é executado. Assim sendo, o comando para criar o primeiro projeto de exemplo ficará conforme apresentado abaixo:

```
ionic start PrimeiroApp blank
```

Figura 13.3 - Comando de criação de um projeto Ionic

O comando é executado utilizando o gerenciador de pacotes do NodeJS para executar as tarefas. A palavra **start** indica que o objetivo é que seja criado um novo projeto Ionic. O parâmetro seguinte é o nome do projeto, neste caso será criada uma pasta de arquivos chamada **PrimeiroApp** contendo toda a estrutura do projeto. O Parâmetro **blank** refere-se ao template que será utilizado, neste caso, será criado um projeto em branco, somente com as configurações básicas.

A partir da versão 3 do Ionic, apesar dos criadores terem anunciado no blog oficial do framework a necessidade do parâmetro da versão ao final do comando, os testes mostraram que, usando esta versão, este comando não é mais válido, por padrão os projetos são criados na versão mais atual instalada. Ao executar o comando, será realizado o download dos recursos necessários e a tela apresentada ao final do procedimento deve ser semelhante a esta:

```
C:\WINDOWS\system32\cmd.exe
C:\Users\allan\Desktop\Minicurso Eripi 2017>ionic start PrimeiroApp blank
[INFO] Creating directory .\PrimeiroApp - done!
[INFO] Fetching app base (https://github.com/driftyco/ionic2-app-base/archive/master.tar.gz)
[INFO] Downloading - done!
[INFO] Fetching starter template blank (https://github.com/driftyco/ionic2-starter-blank/archive/master.tar.gz)
[INFO] Downloading - done!
[INFO] Updating package.json with app details - done!
[INFO] Creating configuration file ionic.config.json - done!
[INFO] Installing dependencies may take several minutes!
> npm install
[INFO] Running command - done!
> npm install --save-dev --save-exact @ionic/cli-plugin-ionic-angular@latest
[INFO] Running command - done!
> git init
[INFO] Running command - done!
> git add -A
[INFO] Running command - done!
> git commit -m Initial commit
[INFO] Running command - done!

🚀🚀🚀 Your Ionic app is ready to go! 🚀🚀🚀

Run your app in the browser (great for initial development):
  ionic serve

Run on a device or simulator:
  ionic cordova run ios

Test and share your app on a device with the Ionic View app:
  http://view.ionic.io

? Link this app to your Ionic Dashboard to use tools like Ionic View? No

Go to your newly created project: cd .\PrimeiroApp
C:\Users\allan\Desktop\Minicurso Eripi 2017>
```

Figura 13.4 – Tela após a criação do projeto

Para verificar o que já foi criado apenas com este comando, é possível iniciar um servidor local na máquina que irá abrir uma nova aba no navegador padrão, exibindo a aplicação. O LiveReload já é habilitado por padrão no Ionic, assim, quaisquer modificações realizadas no código fonte são automaticamente refletidas no navegador. Basta acessar a pasta raiz do projeto através de um terminal e executar o seguinte comando:

```
ionic serve
```

Figura 13.5 – Iniciando o servidor local para visualizar a aplicação no browser

A partir deste ponto, procede-se com a modificação do projeto inicial afim de criar o aplicativo desejado. Este projeto inicial gerado pelo Ionic é um dos fatores que o fazem ter uma grande aceitação por novos desenvolvedores, pois o processo de entendimento da relação entre os componentes na aplicação já é demonstrado de forma prática e objetiva.

Visando apresentar a aplicação da forma mais fiel possível na fase de desenvolvimento, algumas ferramentas de software podem ser utilizadas, dispensando o uso de um dispositivo físico. No navegador Google Chrome estão disponíveis alguns destes recursos que podem auxiliar durante o processo, como as ferramentas do desenvolvedor. Nesta opção é possível visualizar um console Javascript, dentre outras opções, além de testar a responsividade da aplicação em diferentes dispositivos, que podem ser selecionados em uma lista ativando a **device toolbar**. A figura abaixo apresenta a aplicação criada anteriormente sendo executada através do Chrome no modo desenvolvedor. O dispositivo selecionado para exibição é o Samsung Galaxy S5:

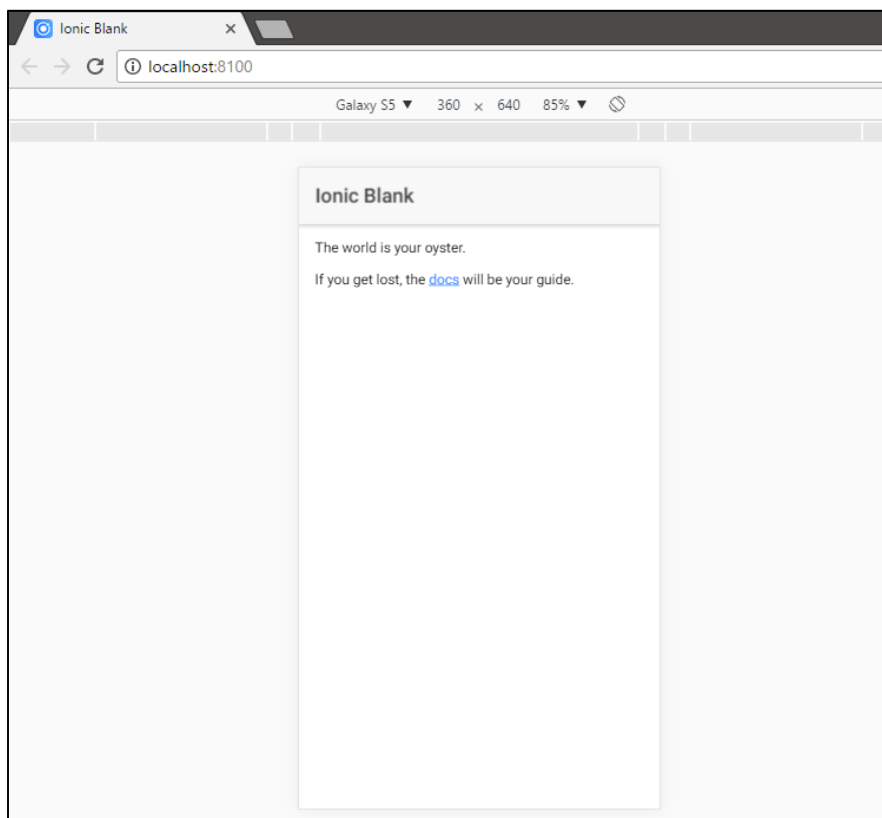


Figura 13.6 – Projeto inicial executando no servidor local

Para edição do código-fonte da aplicação, sugere-se o software Atom, um editor de texto open source leve e com uma gama de recursos disponíveis para o desenvolvimento de aplicativos Ionic. Através do próprio programa podem ser instalados plugins para auto-formatação de código, verificação de erros em tempo real, indentação, destaque de variáveis selecionadas, função auto-completar para o Typescript e fechamento automático de tags HTML, dentre muitos outros recursos.

Um plugin bastante interessante para o desenvolvimento com o Ionic disponível no Atom é o PlatformIO IDE Terminal, que ativa um terminal interno no Atom para que o desenvolvedor possa realizar todos os comandos Ionic CLI no projeto sem ter que sair do editor. O download pode ser realizado através do site : “<https://atom.io/>”.

13.5. Estrutura dos Projetos Ionic

Com a atualização do Ionic da versão inicial para sua versão 2, os projetos de aplicativos ganharam uma nova hierarquia e organização de pastas e arquivos. A mudança trouxe muitos benefícios para os programadores, principalmente aos adeptos da orientação a objetos, pois com esta mudança toda a lógica da aplicação é realizada utilizando a linguagem Typescript, onde as classes (models) podem se comunicar diretamente com as telas (views) da aplicação através da declaração de variáveis e métodos com nomes iguais em ambas as partes.

Cada view da aplicação possui um model correspondente, por padrão, em uma mesma pasta. Outro fator interessante no Ionic é a descentralização dos estilos das páginas. Para cada tela existe um arquivo de estilos de página em branco, que pode sobrescrever as configurações globais do projeto conforme a necessidade do desenvolvedor. Assim, cada pasta contém: o model, que é a classe Typescript contendo as ações da view, a view, que é uma página HTML composta pelas tags do Ionic e por último o arquivo com as configurações de estilos da tela (Sass para a escrita de CSS).

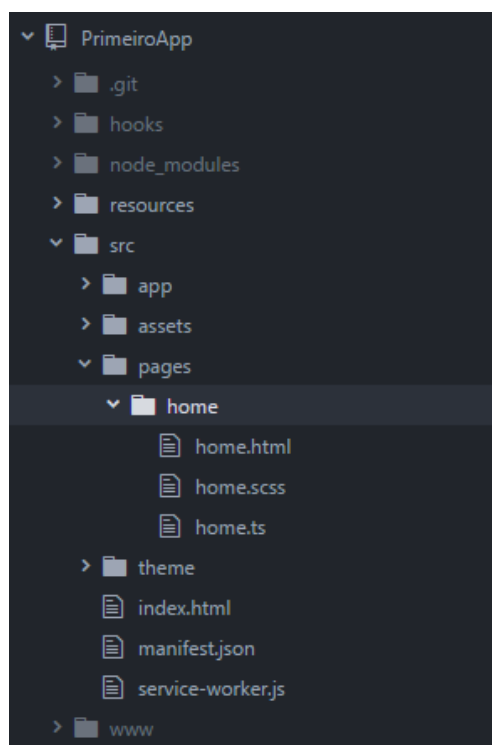


Figura 13.7 – Estrutura básica de um projeto Ionic

Todas estas configurações são padrões do Angular, que considera que os principais componentes de uma aplicação devem possuir escopos isolados. É possível verificar esta organização acessando a pasta do projeto e, em seguida o endereço: **src/pages**, pasta que contém as páginas da aplicação. Para um projeto criado com os templates **blank**, **sidemenu** e **tabs**, será criada uma página chamada **home**. Os demais componentes da estrutura dos projetos serão apresentados conforme sua necessidade, já que este artigo não tem o propósito de explorar todos estes componentes com detalhes.

Fazendo uma explanação sobre as principais pastas do projeto, podemos destacar as seguintes:

- **node_modules**: local onde são armazenados os plugins da aplicação, que foram instalados pelo npm.
- **platforms**: contém o código fonte de cada plataforma do aplicativo, para cada plataforma existe uma pasta que é criada ao comando do desenvolvedor, quando este deseja fazer o build da aplicação.
- **resources**: aqui ficam os arquivos de recursos relacionados a cada plataforma, como ícones do app, ícones de notificação, etc.

- **src:** pasta utilizada mais comumente durante o desenvolvimento. Nela ficam os arquivos de código fonte da aplicação que está sendo implementada.
- **www:** é para onde vai o código depois de transpilada a aplicação. O desenvolvedor não precisa se preocupar com esta parte.

Explorando os componentes básicos de uma página é possível entender o relacionamento entre eles. Dentro da pasta **home**, por exemplo, destacam-se o template (home.html) e o arquivo Typescript responsável por controlá-lo (home.ts). Através desta estrutura, o conteúdo de variáveis e ações de métodos da classe Typescript podem ser acessados pela view simplesmente utilizando o nome da variável ou do método desejado.

```

home.html
1  <ion-header>
2    <ion-navbar>
3      <ion-title>
4        PrimeiroApp
5      </ion-title>
6    </ion-navbar>
7  </ion-header>
8
9  <ion-content>
10
11 </ion-content>
12

```

Figura 13.8 – Estrutura básica de uma view

As views possuem uma estrutura básica simples e bem próxima a uma página HTML convencional. Todo o conteúdo do corpo da tela da aplicação deve ser declarado como conteúdo da tag **ion-content**. Não é necessário declarar na view qualquer relacionamento com folhas de estilo ou a classe controladora, pois a configuração do

```

home.ts
1  import { Component } from '@angular/core';
2  import { NavController } from 'ionic-angular';
3
4  @Component({
5    selector: 'page-home',
6    templateUrl: 'home.html'
7  })
8  export class HomePage {
9
10     constructor(public navCtrl: NavController) {
11
12     }
13
14 }

```

Figura 13.9 – Estrutura básica de uma classe

projeto já se encarrega de realizar estas operações. Na classe controladora podem ser observadas algumas declarações importantes para o desenvolvimento.

Cada classe controladora possui basicamente 3 blocos de código: Declaration, Decorator e Definition. No bloco Declaration são declarados os componentes externos ou bibliotecas que serão utilizados nesta implementação. O segundo bloco, composto por um Decorator, fornece metadados ou informações sobre a classe. Neste exemplo o decorador está definindo que as formatações no HTML serão feitas no componente **page-home** (útil para criação de regras CSS exclusivas) e o arquivo (template HTML) que será utilizado chama-se **home.html**. Por último, o bloco Definition, que é o escopo de classe, comum às linguagens do paradigma de orientação a objetos.

13.6. Componentes

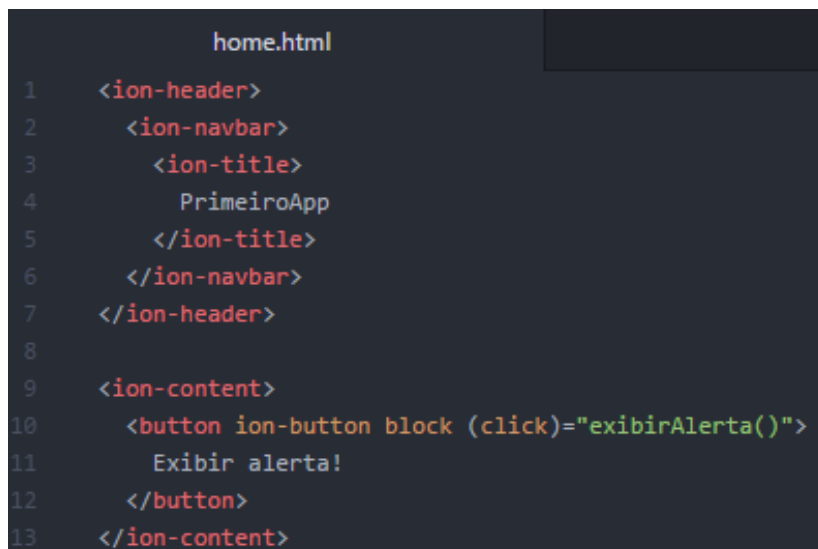
Além de possuir um bom desempenho, o grande diferencial de uma aplicação está na sua interface gráfica, que pode variar bastante em quantidade de cores e recursos em cada tela de acordo com o contexto da aplicação. Tudo deve ser bem pensado para proporcionar ao usuário uma experiência natural e satisfatória. O Ionic possui uma grande variedade de classes de estilos, ícones e outros componentes gráficos pré-configurados para que o desenvolvedor não necessite fazer todo o trabalho de design em suas aplicações.

Para demonstrar um destes recursos, iremos criar um botão personalizado que, ao ser acionado, dispara um evento na classe, exibindo uma mensagem de alerta para o usuário. Primeiro iremos implementar a classe **home.ts**, fazendo a importação do componente que iremos utilizar: o **AlertController**. Em seguida, é realizada a declaração de uma variável correspondente do construtor e, por fim, a implementação do método que irá ser invocado pelo botão.

```
home.ts
1  import { Component } from '@angular/core';
2  import { NavController, AlertController } from 'ionic-angular';
3
4  @Component({
5    selector: 'page-home',
6    templateUrl: 'home.html'
7  })
8  export class HomePage {
9
10     constructor(public navCtrl: NavController, private alertController: AlertController) { }
11
12     exibirAlerta(){
13         let alerta = this.alertController.create({
14             title: 'Olá!',
15             subTitle: 'Seja bem vindo ao mundo do desenvolvimento híbrido!',
16             buttons: ['OK']
17         });
18         alerta.present();
19     }
20 }
```

Figura 13.10 – Configurando a ação de um componente

Ao fim da implementação da classe, podemos iniciar a criação da estrutura da tela no arquivo **home.html**, que será bem simples. Um botão ocupando a parte superior da tela com um texto de boas-vindas que, ao ser tocado pelo usuário irá invocar o método **exibirAlerta()** implementado anteriormente.



```
home.html
1  <ion-header>
2    <ion-navbar>
3      <ion-title>
4        PrimeiroApp
5      </ion-title>
6    </ion-navbar>
7  </ion-header>
8
9  <ion-content>
10    <button ion-button block (click)="exibirAlerta()">
11      Exibir alerta!
12    </button>
13  </ion-content>
```

Figura 13.11 – Configurando um botão que dispara um evento da classe

Um detalhe importante a se ressaltar nesta parte é a presença dos parêntesis no atributo **click** do botão. Trata-se de um recurso chamado Event Binding que, em linhas gerais serve para enviar requisições no sentido view/model, assim, o evento que está declarado para o botão é enviado para a classe responsável (home.ts) e ela se encarregará de executar o procedimento requisitado.

Existem ainda outros dois conceitos neste mesmo contexto. O primeiro deles é o Property Binding, que trata o fluxo contrário ao Event Binding, ou seja, os dados trafegam no sentido model/view, assim, o valor de uma variável alterado na classe é automaticamente atualizado na página. Para declarar uma propriedade que irá receber os dados (Property Binding), basta declará-la entre colchetes.

O outro conceito é o Two Way Data Binding, que executa o fluxo em duplo sentido simultaneamente, sincronizando as propriedades da tela com os valores das variáveis, independente da origem da alteração. Nesta modalidade, a propriedade deve ser declarada entre colchetes e parêntesis internos aos colchetes. O assunto pode ser entendido mais a fundo através de um estudo das propriedades do Angular Framework.

13.7. Utilizando recursos nativos com o Cordova

A utilização dos recursos nativos dos dispositivos móveis pode ser realizada através do Cordova Framework, de acordo com a descrição feita no início deste capítulo. Para

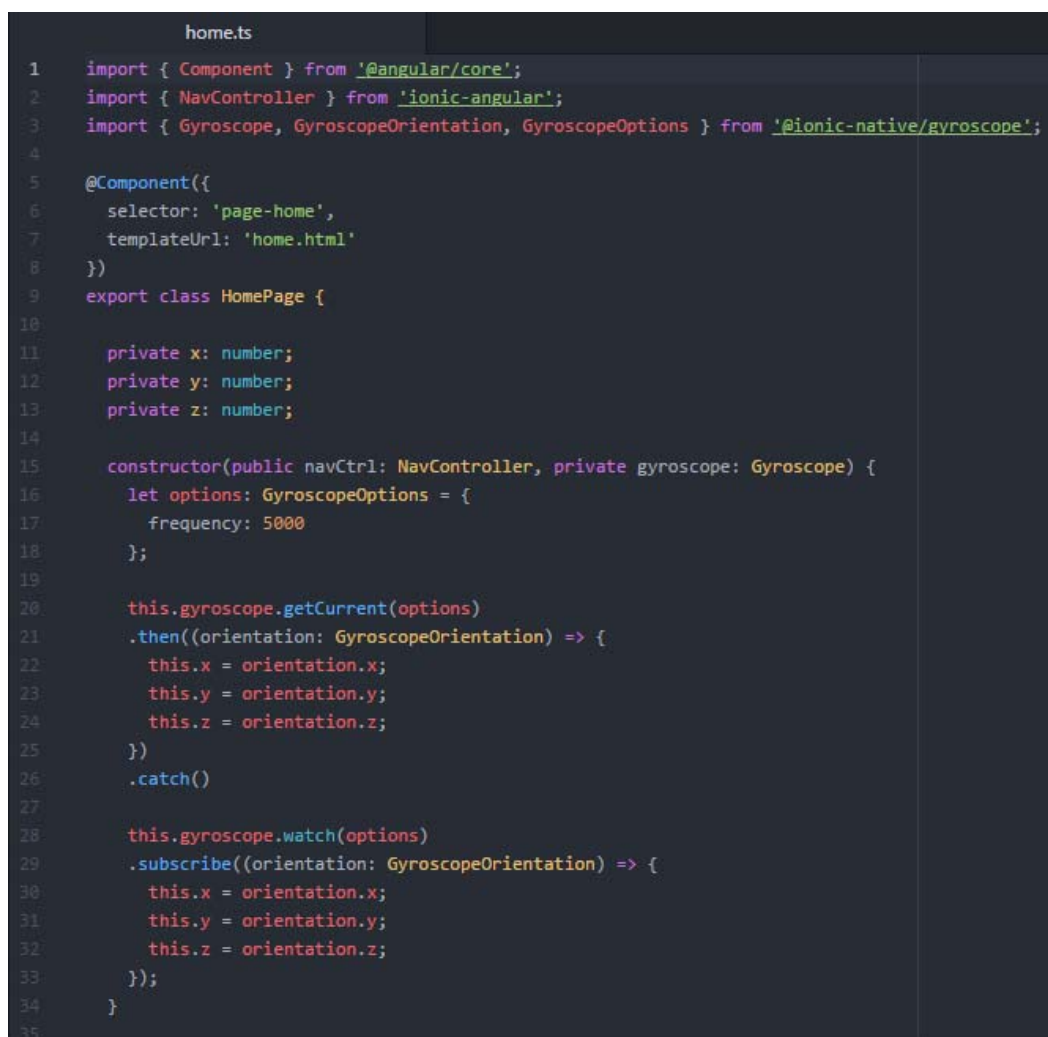
exemplificar o uso deste tipo de recurso, será realizado o acesso aos valores do sensor acelerômetro do dispositivo.

O primeiro passo é adicionar o plugin necessário ao projeto. O Cordova disponibiliza uma lista de plugins necessários para utilização dos diversos sensores e recursos dos dispositivos atuais. No caso do acelerômetro, é necessário adicionar o plugin Gyroscope, através dos seguintes comandos (executar um comando por vez):

```
ionic plugin add cordova-plugin-gyroscope
npm install --save @ionic-native/gyroscope
```

Figura 13.12 – Adicionando plugin do Cordova

A adição dos plugins também é dependente de uma conexão com a internet, pois os recursos serão baixados de repositórios no github. Após a execução e instalação com sucesso do plugin, devem aparecer algumas confirmações no próprio terminal, semelhante ao que aconteceu após a instalação do Ionic.



```
home.ts
1  import { Component } from '@angular/core';
2  import { NavController } from 'ionic-angular';
3  import { Gyroscope, GyroscopeOrientation, GyroscopeOptions } from '@ionic-native/gyroscope';
4
5  @Component({
6    selector: 'page-home',
7    templateUrl: 'home.html'
8  })
9  export class HomePage {
10
11    private x: number;
12    private y: number;
13    private z: number;
14
15    constructor(public navCtrl: NavController, private gyroscope: Gyroscope) {
16      let options: GyroscopeOptions = {
17        frequency: 5000
18      };
19
20      this.gyroscope.getCurrent(options)
21        .then((orientation: GyroscopeOrientation) => {
22          this.x = orientation.x;
23          this.y = orientation.y;
24          this.z = orientation.z;
25        })
26        .catch();
27
28      this.gyroscope.watch(options)
29        .subscribe((orientation: GyroscopeOrientation) => {
30          this.x = orientation.x;
31          this.y = orientation.y;
32          this.z = orientation.z;
33        });
34    }
35  }
```

Figura 13.13 – Código para capturar valores do sensor Acelerômetro

Instalado o plugin, é hora de adicionar o código necessário para que o aplicativo possa ter acesso ao acelerômetro. O próximo passo é importar as classes que possuem a implementação dos métodos de controle do sensor, que foram adicionadas ao projeto no passo anterior. Esta declaração deve estar na classe Typescript relacionada à tela em que o aplicativo irá necessitar dos valores do sensor, neste caso, a importação será declarada na página **home.ts** e os valores poderão ser visualizados na página inicial do aplicativo, a **home.html**.

É importante lembrar que, ao acionar recursos nativos dos dispositivos, não será possível visualizar os resultados no servidor local do Ionic, pois este não possui o suporte necessário para exibir os resultados no navegador. Uma alternativa que funciona para alguns recursos é utilizar um emulador do SO desejado, porém, para resultados mais fideis é recomendado que os testes de recursos nativos sejam realizados em dispositivos físicos.

```
home.html
1  <ion-header>
2    <ion-navbar>
3      <ion-title>
4        PrimeiroApp
5      </ion-title>
6    </ion-navbar>
7  </ion-header>
8
9  <ion-content>
10
11    <p>X: {{x}}</p>
12    <p>Y: {{y}}</p>
13    <p>Z: {{z}}</p>
14
15  </ion-content>
```

Figura 13.14 – Código para visualizar os valores capturados pelo sensor

Ao executar o projeto com o comando **ionic serve**, a execução irá incorrer em um erro indicando que não há um provider para um objeto Gyroscope. Para resolver este problema, temos que declarar este componentes na lista de providers do aplicativo, no endereço **src/app**, arquivo **app.module.ts**. Basta declarar a importação da classe no arquivo e em seguida, declarar a classe Gyroscope no vetor json de providers mais abaixo. Após esta declaração o erro será solucionado, porém, o resultado só poderá ser visualizado em um dispositivo físico, conforme explicado anteriormente.

Com a codificação da aplicação pronta, é possível dar o próximo passo e fazer o build da aplicação, podendo compilar o código para as diferentes distribuições de sistemas operacionais. Para realizar esta operação o procedimento é bem simples e será explicado na seção seguinte.

13.8. Adicionando plataformas ao projeto

O build de uma aplicação nada mais é do que uma versão compilada do software ou parte dele. No Ionic CLI podemos facilmente fazer o build da aplicação desenvolvida para a plataforma desejada. Para demonstrar esta operação, iremos adicionar a plataforma Android ao projeto atual, conforme o seguinte comando:

```
ionic cordova platform add android
```

Figura 13.15 – Adicionando uma plataforma ao projeto

A partir do momento em que o projeto conta com uma plataforma em sua estrutura, temos uma série de outras possibilidades para criar recursos diferenciados de acordo com o sistema operacional do dispositivo que está executando nossa aplicação. A personalização de ícones, tamanhos e cores são exemplos de recursos que podem ser configurados pelo programador para proporcionar uma melhor experiência ao usuário.

Uma ferramenta bastante útil nesta fase é o Ionic Lab. Com ele é possível visualizar a interface e testar as funcionalidades da aplicação em várias plataformas simultaneamente. Através de um menu de opções, é possível ativar a exibição para Android, iPhone e Windows. Para iniciar a execução deste modo, basta adicionar o parâmetro - **lab** logo após o comando **ionic serve**, ou simplesmente substituir a palavra **serve** por **lab**, ficando assim: **ionic lab**.

Há ainda uma outra ferramenta muito interessante criada pela equipe do Ionic: o Ionic Creator, uma ferramenta em que o desenvolvedor pode “montar” os componentes da aplicação, clicando e arrastando os componentes desejados para a tela do dispositivo na área central. O Ionic Creator funciona online, basta que o usuário faça um rápido cadastro para iniciar o uso. A conta é gratuita, mas possui limitações de recursos. Caso o usuário queira acesso a todos os recursos, precisa fazer um upgrade da conta, efetuando o pagamento do plano.

13.9. Executando a aplicação

Para testar a aplicação, existem várias alternativas disponíveis no ambiente do Ionic. A execução pode ser realizada diretamente em um dispositivo físico, em um emulador, de modo simples no navegador, multiplataforma no navegador, no dispositivo físico com depuração no navegador, enfim, são muitas as alternativas, vamos descrever algumas delas.

- **ionic cordova build [platform]**: prepara e compila a aplicação para uma plataforma. Disponível para Android e iOS.
- **ionic cordova emulate [platform]**: executa o aplicativo em um emulador da plataforma descrita no comando. Disponível para Android e iOS.
- **ionic cordova run [platform]**: executa o aplicativo em um dispositivo conectado. Disponível para Android e iOS.
- **ionic lab**: executa a aplicação no navegador, mostrando a interface em várias plataformas simultaneamente.

- **ionic serve –lab**: mesma ação do comando **ionic lab**.

No site oficial do Ionic são disponibilizados muitos exemplos sobre a utilização de seus recursos, tanto para interface gráfica de usuário quanto para aspectos funcionais das aplicações desenvolvidas. A comunidade do Ionic também é bastante ativa, tornando a resolução de quaisquer problemas muito mais fácil. O desenvolvimento com o Ionic Framework continua em ascensão em todo o mundo.

Referências

Angular. “Angular Docs”, <https://angular.io/docs/ts/latest/>, Maio.

Cordova. “Documentation – Apache Cordova”, <https://cordova.apache.org/docs/en/latest/>, Maio.

Ionic (2017a). “Build Amazing Native Apps and Progressive Web Apps with Ionic Framework and Angular”, <http://ionicframework.com/>, Abril.

Ionic (2017b). “Ionic Documentation”, <http://ionicframework.com/docs/>, Abril.

Ionic (2017c). “Ionic Creator – Rapid Mobile App Builder”, <https://ionic.io/>, Maio.