UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
ESCOLA DE INFORMÁTICA APLICADA

Desenvolvimento do aplicativo UNIRIO móvel

Rodrigo Gomes Cezar

Orientador
Márcio de Oliveira Barros

RIO DE JANEIRO, RJ – BRASIL
DEZEMBRO DE 2018

Sistema de Agenda do Estudante UNIRIO

Rodrigo Gomes Cezar

Projeto de Graduação apresentado à Escola de Informática Aplicada da Universidade Federal do Estado do Rio de Janeiro (UNIRIO) para obtenção do título de Bacharel em Sistemas de Informação.

Aprovada por:

_____

Prof. Márcio de Oliveira Barros, D.Sc. (UNIRIO)

_____

Prof. Morganna Carmem Diniz, D.Sc. (UNIRIO)

_____

Prof. Ronaldo da Silva Busse, D.Sc. (UNIRIO)

RIO DE JANEIRO, RJ – BRASIL.

DEZEMBRO DE 2018

RESUMO

Este trabalho apresenta o desenvolvimento de um aplicativo que realiza a leitura e tratamento de dados do sistema PLONE da UNIRIO e organiza as informações obtidas para exibição em um dispositivo móvel. O aplicativo foi desenvolvido com o *framework* IONIC, onde a principal linguagem de programação é o *Typescript.* Por utilizar um *framework* de desenvolvimento de aplicativos híbridos, ou seja, capazes de funcionar em múltiplas plataformas, este trabalho apresenta um estudo sobre a arquitetura e ferramentas envolvidas na construção desse tipo de aplicativo.

Palavras-chave: Aplicativos híbridos para dispositivos móveis, IONIC, *Typescript*.

ABSTRACT

This work presents the development of an application that retrieves and processes data from the PLONE system installed at UNIRIO, preparing the retrieved information to be displayed on a mobile device. The application was developed using the IONIC framework, whose main programming language is *Typescript*. By using a hybrid application development framework, i.e., one that allows developing applications that run on multiple platforms, this work also presents a study about the architecture and tools involved in the development of such kind of application.

Keywords: Hybrid mobile applications, IONIC, Typescript.

Índice

Índice de Figuras

Índice de Tabelas

1 Introdução

1.1 Motivação

São notórios os benefícios trazidos por dispositivos móveis à nossa sociedade no que diz respeito ao acesso à informação. Graças a esses dispositivos, e também à Internet, o acesso a uma quantidade praticamente ilimitada de informação está ao alcance das mãos em praticamente qualquer lugar. Tão notáveis quanto essa facilidade de acesso são as formas de se interagir com essas informações.

O uso dessas tecnologias também tem se difundido nas universidades, inclusive através da criação de aplicativos para dispositivos móveis. Esses aplicativos têm o objetivo de facilitar o acesso às informações importantes do universo acadêmico e também da interação do aluno com os sistemas da instituição. A motivação deste trabalho é a necessidade de criar um aplicativo dessa natureza para a UNIRIO.

1.2 Objetivos

O objetivo deste trabalho é criar um aplicativo que apresente informações importantes sobre a UNIRIO em um dispositivo móvel. Para tanto, parte das informações que estão no sistema PLONE institucional da UNIRIO serão formatadas de acordo com um padrão previamente estabelecido, de modo que possam ser lidas pelo aplicativo e apresentadas de forma adequada aos usuários em seus dispositivos móveis. Um importante requisito deste aplicativo é a persistência dos dados, ou seja, uma vez que o aplicativo tenha acessado os dados do sistema PLONE, eles devem ser armazenados no aparelho para possibilitar o acesso aos mesmos, posteriormente, ainda que sem conexão com a Internet.

1.3 Estrutura do Texto

O presente trabalho está estruturado em capítulos e, além desta introdução, será desenvolvido da seguinte forma:

- Capítulo II: apresenta uma pesquisa sobre a difusão da tecnologia móvel na sociedade e nas universidades do Brasil e do mundo;
- Capítulo III: apresenta a especificação funcional do aplicativo que foi desenvolvido para a UNIRIO e as tecnologias utilizadas em seu desenvolvimento;
- Capítulo IV: apresenta a forma e funcionalidade do aplicativo, bem como detalhes técnicos relacionados com a sua construção;
- Capítulo V: apresenta conclusões e expectativas sobre novas versões do aplicativo, a ser desenvolvidas no futuro.

2 Tecnologia Móvel em Universidades

2.1 Introdução

A difusão dos dispositivos móveis entre aqueles que usam tecnologia no seu dia a dia alcançou marcos notáveis na sociedade em todo mundo. Em março de 2015, pela primeira vez, o número de usuários de dispositivos móveis, entre adultos, superou o de computadores desktop [1]. O mesmo também foi observado em relação ao tráfego de dados na Internet, onde o volume de dados foi superior nos acessos por dispositivos móveis.

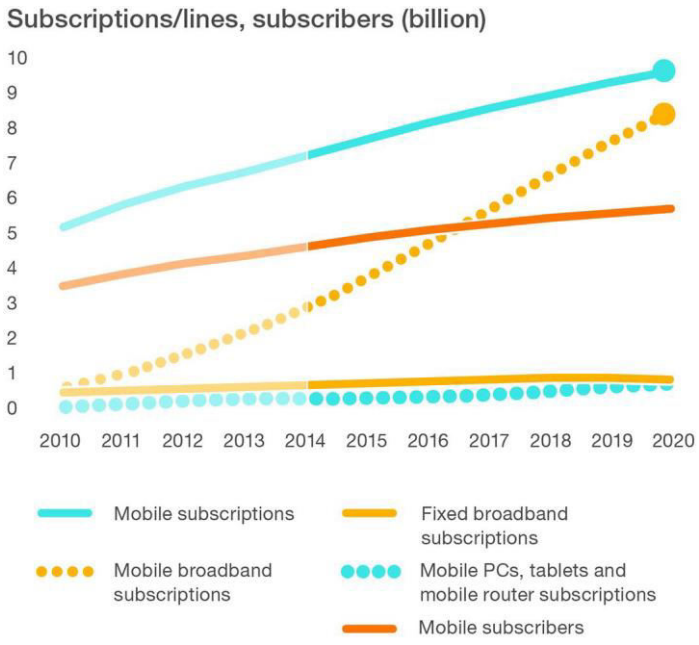

Figura 1 - Projeção de número de assinaturas de Internet fixas e móveis [2]

A Figura 1 mostra uma comparação de diferentes tipos de assinaturas de Internet. Até 2020, estima-se haver 7,7 bilhões de assinaturas de banda larga móvel. A banda larga fixa ainda se mostrará presente devido a uma predominância no meio empresarial.

A empresa sueca Ericsson disponibiliza bimestralmente um relatório sobre mobilidade, contendo dados e estimativas sobre tráfego móvel, assinaturas de planos de dados e tendências do mercado na adoção de tecnologia em todo o mundo [2]. Nesses relatórios, estima-se que haverá 6,1 bilhões de usuários de dispositivos móveis em todo o mundo até 2020, além dos seguintes fatos:

- Cerca de 345 milhões de *smartphones* foram vendidos no segundo trimestre de 2018, o que equivale a 85% de todos os telefones celulares vendidos no trimestre (note que em 2018 ainda existem celulares que não são *smartphones*). Assinaturas associadas a *smartphones* agora são responsáveis por 60% de todas as assinaturas de celular;
- O número de assinaturas de banda larga móvel cresceu cerca de 170 milhões no segundo trimestre de 2018, atingindo cerca de 5,5 bilhões de assinaturas. Isso reflete um aumento ano a ano de cerca de 15%;
- O número de assinaturas distintas de planos de dados móveis é de cerca de 5,4 bilhões no mundo. A diferença entre o número de assinaturas e o número de assinantes é devido a assinaturas inativas, a posse de vários dispositivos por um mesmo indivíduo e / ou otimização de assinaturas para diferentes tipos de chamadas;

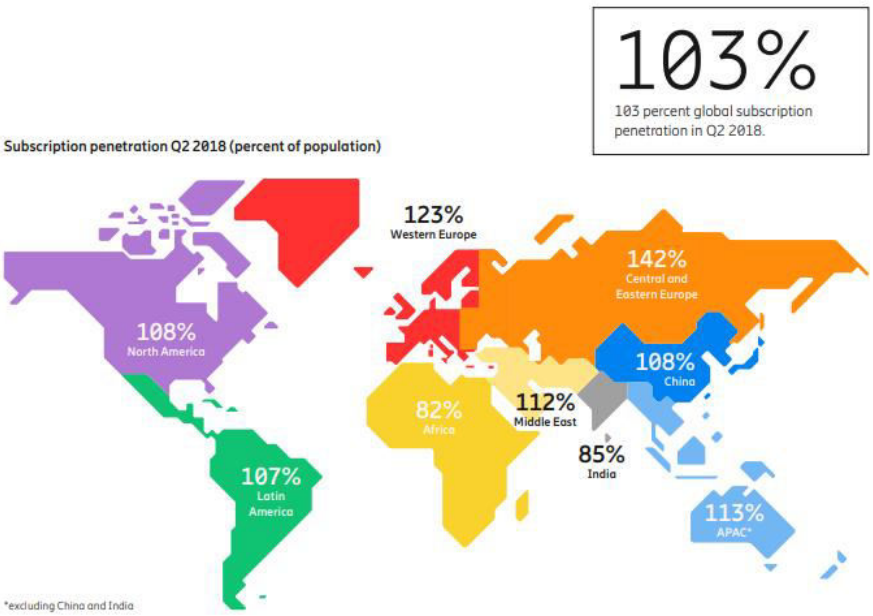

Figura 2 - Difusão do uso da Internet móvel no mundo em 2018 [2]

- O tráfego de dados móveis cresceu 52% entre o segundo trimestre de 2017 e o segundo trimestre de 2018.

A Figura 2 ilustra a difusão das assinaturas de planos de Internet móvel em relação à população nas regiões destacadas. Nota-se que em diversas regiões do mundo, o número de assinaturas de Internet móvel supera o número de habitantes.

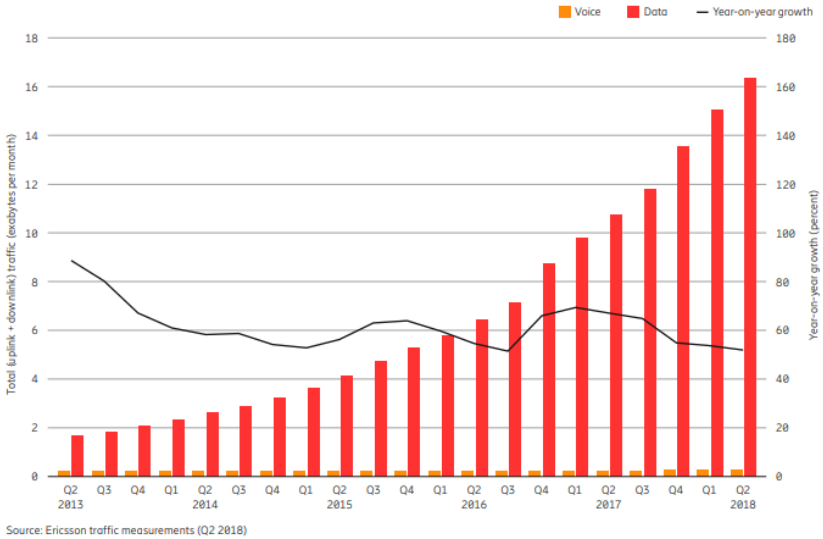

Figura 3 - Tráfego de dados de Internet móvel [2]

O crescimento do tráfego de Internet móvel, ilustrado na Figura 3, é impulsionado pelo aumento do número de usuários de *smartphones* e também pela expansão da banda disponibilizada nas assinaturas de pacote de dados, notavelmente atribuído a uma maior visualização de conteúdo de vídeo, em resoluções cada vez maiores.

2.2 Aplicativos móveis

Aplicações digitais que ajudam nas tarefas do dia a dia podem ser usadas através de sites da Internet ou em aplicativos desenvolvidos especialmente para a plataforma móvel. Um site pode ser desenvolvido especificamente para visualização em dispositivos móveis e sua construção é semelhante à de qualquer outro, ou seja, criado em páginas para navegadores compatíveis com HTML e o padrão CSS. Os sites para dispositivos móveis são projetados para telas de tamanho pequeno e também podem fazer uso de funcionalidades específicas dos dispositivos móveis, como clique para chamada de voz ou localização através do GPS. Mas, mesmo com toda a otimização disponível para um dispositivo móvel, ainda é necessário abrir um navegador, digitar

uma URL e esperar que os dados sejam renderizados. O desenvolvimento de um aplicativo pode tornar o uso dessas aplicações digitais muito mais rápido e eficiente.

Aplicativo móvel, conhecido normalmente por seu nome abreviado *app*, é um software desenvolvido para ser instalado em um dispositivo eletrônico móvel, como um PDA, telefone celular, *smartphone* ou um leitor de MP3 [3]. Além da opção de se desenvolver um site otimizado para dispositivos móveis, também existem as opções de se desenvolver essa aplicação de forma nativa ou híbrida.

Os aplicativos nativos são construídos para o sistema operacional específico do dispositivo móvel, o que significa que seu funcionamento é restrito aos dispositivos que possuem esse sistema operacional. Nesse caso, os desenvolvedores precisam ter domínio de linguagens de programação específicas, SDK e outras ferramentas para a plataforma de destino. Por exemplo, a linguagem de programação para o desenvolvimento de um aplicativo para iOS é o *Swift* (anteriormente *Objective C*) , para Android é Java e para *Windows phone*, C#.

Os aplicativos híbridos são construídos com base em tecnologias de desenvolvimento da Web, como HTML5, CSS e JavaScript. Esses aplicativos são então executados em um *wrapper,* como o Apache Cordova [4], que é o responsável pela tradução do código para o sistema destino e também pelo acesso a funcionalidades do hardware. Depois que o código é criado, ele pode ser utilizado em diversas plataformas móveis, necessitando de pouca ou nenhuma alteração.

Os aplicativos híbridos buscam oferecer aos usuários uma experiência indistinta dos aplicativos nativos. Entretanto, demonstram limitação quando se trata de aplicações onde é necessário processamento rápido, como jogos 3D. Os aplicativos híbridos também podem não ter acesso a todas as funcionalidades de hardware do dispositivo, como, por exemplo, o *Multi-touch*, que atualmente pode ser acessado apenas por linguagens nativas. Uma IDE (*Integrated development environment*) com suporte a *debugging* e outras ferramentas importantes para os desenvolvedores, também são exclusividade do desenvolvimento nativo.

Uma visão geral para a comparação entre diferentes tipos de aplicativos móveis é mostrada na Tabela 1, colocando o nível de funcionalidade de um aplicativo híbrido entre o nativo e o Web.

		Nativo	Web	Híbrido
Recursos	Gráficos	APIs nativas	HTML, canvas, SVG	HTML, canvas, SVG
	Performance	Rápido	Lento	Lento
	Distribuição	Appstore	Internet	Appstore
	Conectividade	Online e offline	Online	Online e Offline
Acesso ao hardware	Câmera	Sim	Não	Sim
	Push notification	Sim	Não	Sim
	Sistema de arquivos	Sim	Não	Sim
	Geolocalização	Sim	Sim	Sim
Gestos	*Swipe*	Sim	Sim	Sim
	Pinch, spread	Sim	Não	Sim

Tabela 1 - Comparação entre aplicações nativas, web e híbridas, para dispositivos móveis [4].

Embora o desenvolvimento de aplicativos nativos pareça ser sempre a melhor opção em termos de qualidade, é necessário avaliar o custo do desenvolvimento na hora da escolha entre nativo e híbrido. Dependendo do tipo de aplicação, um aplicativo híbrido pode ter a mesma qualidade de um aplicativo nativo a um custo de desenvolvimento muito mais baixo e sem a necessidade de desenvolvimento extra para múltiplas plataformas.

2.3 Android e IOS

Nos últimos anos, o mercado de *smartphones* tem testemunhado uma batalha entre dois gigantes da tecnologia, a saber, Apple e Google, empresas responsáveis pelos sistemas operacionais iOS e Android, respectivamente. Juntas, essas empresas criaram um Duopólio e respondem por quase 95% da difusão de sistemas operacionais de dispositivos móveis.

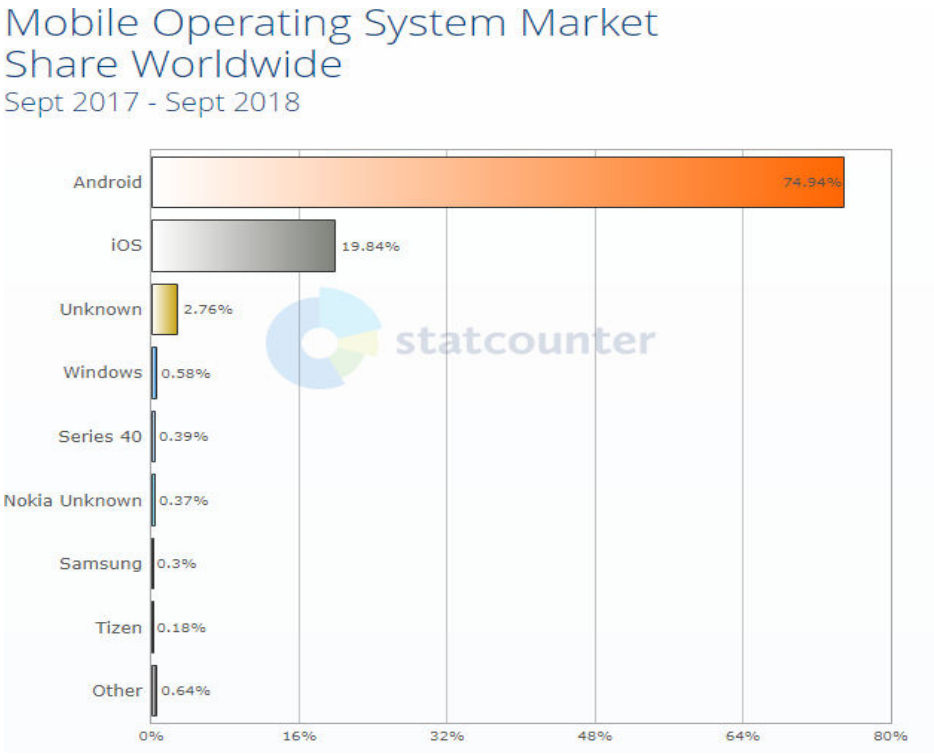

Figura 4 - Participação dos sistemas operacionais no mercado móvel mundial [5].

Na Figura 4 podemos ver um gráfico da distribuição do mercado de sistemas operacionais móveis em setembro de 2018.

2.3.1 Android

O Android é um sistema operacional para dispositivos móveis desenvolvido pelo Google e baseado em uma versão modificada do kernel do Linux e de outros softwares de código aberto, projetado principalmente para dispositivos móveis com tela sensível ao toque, como *smartphones* e *tablets*. Além disso, o Google também desenvolveu versões do Android para televisores, automóveis e relógios de pulso, cada um com uma interface com usuário especializada. Variantes do Android também são usadas em consoles de jogos, câmeras digitais, PCs e outros eletrônicos.

Inicialmente desenvolvido pela Android Inc., empresa comprada pelo Google em 2005, o Android foi revelado em 2007 e o primeiro dispositivo Android foi lançado em setembro de 2008. O sistema operacional já passou por vários lançamentos importantes, sendo a versão atual 9.0 "Pie", lançada em agosto de 2018. O código-fonte do Android é conhecido como "*Android Open Source Project*" (AOSP) e é licenciado sob a licença Apache.

Desde 2011, o Android é o sistema operacional de *smartphones* mais difundido em todo o mundo e, em *tablets*, desde 2013. Em maio de 2017, haviam mais de dois bilhões de usuários ativos mensais, a maior base instalada de qualquer sistema operacional. Em junho de 2018, existem mais de 3,3 milhões de aplicativos na Google Play Store [6].

2.3.2 iOS

O iOS (antigo iPhone OS) é um sistema operacional para dispositivos móveis criado e desenvolvido pela Apple Inc. exclusivamente para o seu hardware. É o sistema operacional usado em muitos dos dispositivos da empresa, como o iPhone, iPad e iPod Touch. É o segundo sistema operacional para dispositivos móveis mais popular no mundo, depois do Android.

Revelado em 2007, para o iPhone, o iOS foi estendido para suportar outros dispositivos da Apple, como o iPod Touch (setembro de 2007) e o iPad (janeiro de 2010). Em janeiro de 2017, a App Store da Apple registrou mais de 2,2 milhões de aplicativos iOS, dos quais 1 milhão são nativos para iPads. Esses aplicativos foram coletivamente baixados mais de 130 bilhões de vezes [7].

A interface com o usuário do iOS é baseada em manipulação direta, usando gestos *multi-touch* com definições específicas dentro do contexto do sistema operacional. Os acelerômetros internos são usados por alguns aplicativos para responder ao tremor do dispositivo (um resultado comum é o comando desfazer) ou girá-lo em três dimensões (um resultado comum é alternar entre o modo retrato e paisagem). A Apple tem demonstrado um esforço notável por incorporar funções completas de acessibilidade ao iOS, permitindo que usuários com deficiências visuais e auditivas usem adequadamente seus produtos.

Versões principais do iOS são lançadas anualmente. A versão atual, iOS 12, foi lançada em 17 de setembro de 2018. Está disponível para todos os dispositivos iOS com processadores de 64 bits.

2.4 Aplicativos móveis em universidades

Assim como é observado na sociedade em geral, aplicativos para dispositivos móveis também podem trazer benefícios ao dia a dia de estudantes universitários. Aplicativos que venham mostrar detalhes não apenas do funcionamento do centro acadêmico, mas também de informações individualizadas da situação de cada aluno

podem ajudar o corpo discente e despertar interesse em potenciais candidatos a uma vaga no centro acadêmico. Nas próximas subseções serão mostrados alguns aplicativos construídos para melhorar a experiência na universidade.

2.4.1 Aplicativo "Portal do Aluno UFRJ"

O aplicativo "Portal do Aluno UFRJ", disponível para plataforma Android, na loja de aplicativos do Google[1], disponibiliza aos seus alunos o acesso à sua matrícula, histórico, localização de salas de aula e diversos outros dados que ajudam a tornar mais eficiente o dia a dia dos estudantes.

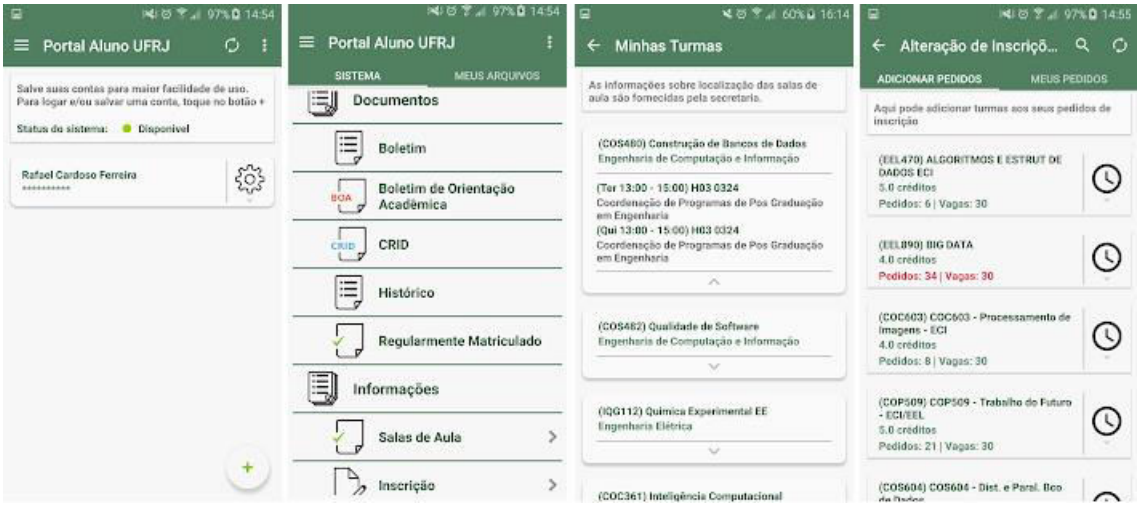

Figura 5 - Telas do aplicativo "Portal do Aluno UFRJ".

Na Figura 5 podemos ver alguns exemplos de telas do aplicativo "Portal do aluno UFRJ". Uma vez que é uma aplicação exclusiva para os alunos, é necessária a autenticação no sistema da universidade. Em seguida, o aluno tem acesso a informações sobre a sua matrícula, horários, localização das salas de aula, entre outras. O aplicativo também possibilita ao aluno a solicitação, alteração ou cancelamento de inscrição em disciplinas, uma funcionalidade em que, em geral, demanda a ida a secretaria do curso.

2.4.2 Aplicativo "MIT mobile"

O aplicativo "MIT mobile", do Instituto de Tecnologia de Massachusetts (em inglês: *Massachusetts Institute of Technology* - MIT), disponível para plataforma Android, na loja de aplicativos do Google[2].

[1] https://play.google.com/store/apps/details?id=br.ufrj.gnosys.mobile&hl=en
[2] https://play.google.com/store/apps/details?id=edu.mit.mitmobile2&hl=en

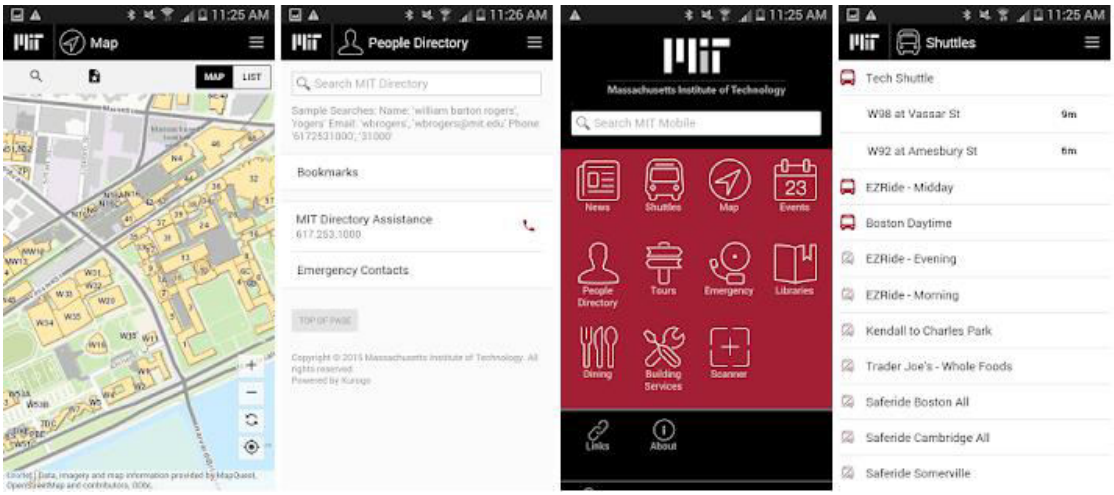

Figura 6 - Telas do aplicativo "MIT mobile".

Na figura 6 podemos ver alguns exemplos de telas do aplicativo "MIT mobile", tais como:

- Um canal oficial de notícias do campus;
- Rastreamento, em tempo real, dos ônibus do campus;
- Mapa do campus onde é possível buscar localidades;
- Calendário de eventos, exposições, feriados e calendário acadêmico;
- Diretório de professores, funcionários e alunos da instituição;
- Informações sobre canais de atendimento de Emergências;
- Gerenciamento de contas de bibliotecas MIT e pesquisa de catálogo;
- Cardápios e horários de refeições;
- *Push notifications* para informativos emergenciais.

2.4.3 Aplicativo "APP Aluno | PUC-Campinas"

O aplicativo "APP Aluno | PUC-Campinas", da Pontifícia Universidade Católica de Campinas (PUC-Campinas), disponível para a plataforma Android, na loja de aplicativos do Google[3].

[3] https://play.google.com/store/apps/details?id=br.edu.puccampinas.appdoaluno&hl=pt_BR

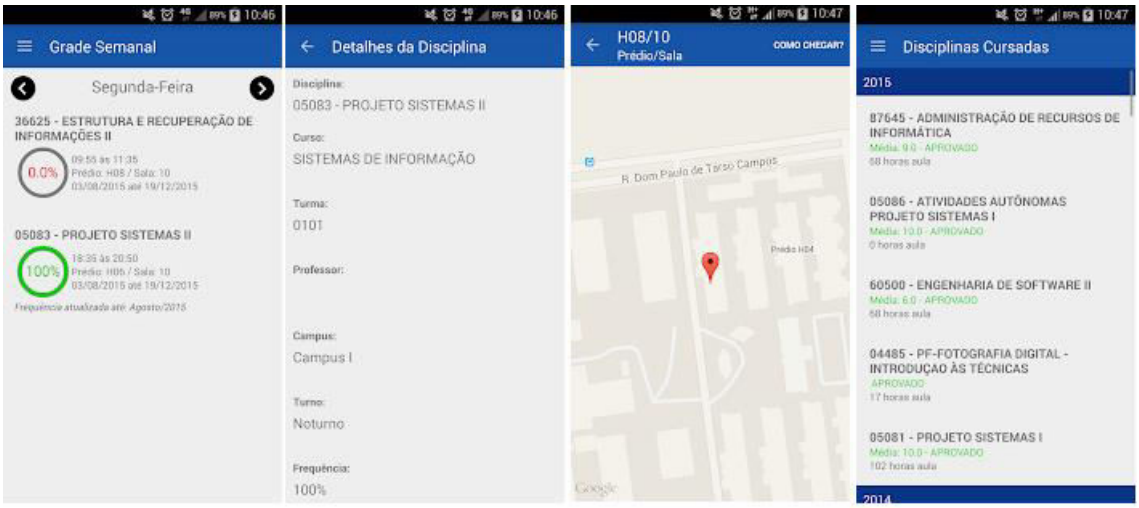

Figura 7 - Telas do aplicativo "APP Aluno | PUC-Campinas"

Na Figura 7, podemos ver alguns exemplos de telas do aplicativo "APP Aluno | PUC-Campinas", onde é possível observar alguns recursos, tais como horário e Local das aulas, notas, frequências e dados cadastrais.

2.4.4 Aplicativo "Estácio"

O aplicativo "Estácio", da Universidade Estácio de Sá (UNESA), disponível para a plataforma iOS, na loja de aplicativos do Apple[4], disponibiliza aos seus alunos o acesso a informações relativas ao rendimento acadêmico e quitação de despesas. O aplicativo possui uma interface personalizável, onde é possível criar uma tela inicial com atalhos para as funcionalidades favoritas.

Figura 8 - Telas do aplicativo "Estácio"

[4] https://itunes.apple.com/br/app/est%C3%A1cio/id1062656922?mt=8

Na Figura 8, podemos ver alguns exemplos de telas do aplicativo “Estácio”, onde é possível observar alguns recursos, tais como:

- Carteirinha Digital de Estudante;
- Visualização de notas e frequência;
- Quadro de horários;
- Segunda via de boletos;
- Agendamento de prova online;
- Consulta ao Histórico Escolar;
- Datas de provas presenciais e online;
- Manual do aluno;
- Biblioteca Virtual.

2.4.5 Aplicativo “UCL Go !”

O aplicativo “UCL Go !”, da universidade britânica “*University College London* (UCL)”, disponível para a plataforma iOS[5] permite que os alunos acessem informações completas sobre sua universidade. A UCL foi uma das primeiras universidades do Reino Unido a disponibilizar um aplicativo capaz de fornecer integração em tempo real entre os alunos e os principais sistemas universitários.

Figura 9 - Telas do aplicativo “UCL Go!”.

[5] https://itunes.apple.com/us/app/ucl-go-student-edition/id357476297#?platform=iphone

Na Figura 9, podemos ver alguns exemplos de telas do aplicativo “UCL Go!”, onde é possível observar alguns recursos, tais como:

- Pesquisar mapas do campus para edifícios e locais;
- Verificar empréstimos e disponibilidade de recursos da biblioteca;
- Acesso ao calendário do curso;
- Procurar amigos, colegas ou palestrantes usando a lista de contatos;
- Ligar ou enviar e-mails para contatos;
- Receber as últimas notícias e eventos da Universidade.

2.4.6 Aplicativo “Harvard Mobile”

O aplicativo “Harvard Mobile”, da universidade norte americana “Harvard University”, disponível para a plataforma Android, na loja de aplicativos do Google[6], é uma iniciativa da universidade para melhorar a experiência móvel de alunos, professores, funcionários, visitantes e vizinhos que interagem com o campus e a comunidade de Harvard.

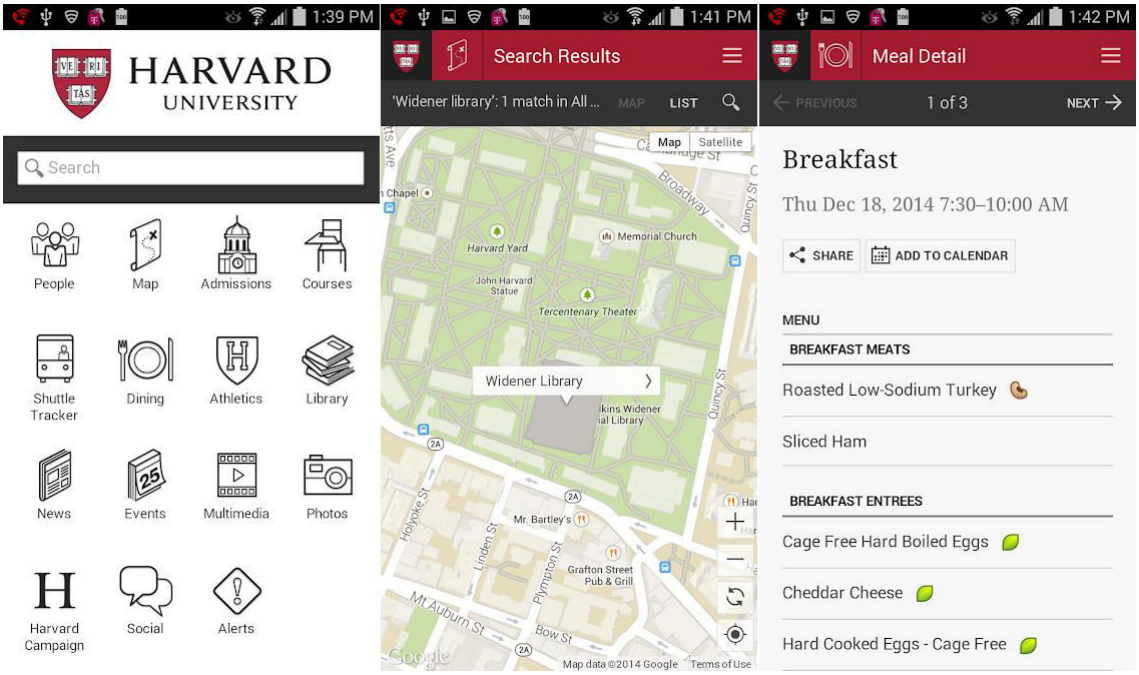

Figura 10 - Telas do aplicativo “Harvard Mobile”.

Na Figura 10, podemos ver alguns exemplos de telas do aplicativo “Harvard Mobile”, onde é possível observar uma série de recursos, tais como:

[6] https://play.google.com/store/apps/details?id=edu.harvard.harvardmobile&hl=en

- Catálogo onde é possível pesquisar o corpo docente e discente;
- Um mapa da universidade;
- Cardápio e horários dos refeitórios;
- Verificar disponibilidade de recursos na biblioteca;
- Receber as últimas notícias e eventos da Universidade;
- Acesso ao histórico escolar;
- Verificação, em tempo real, da disponibilidade e localização dos ônibus do campus.

2.5 Considerações Finais

A partir da análise feita sobre a difusão da tecnologia móvel é razoável concluir que o uso dessa tecnologia não é mais apenas um luxo, e sim um requisito para se acompanhar o ritmo atual da sociedade. O mesmo também é verdade para várias universidades no mundo, onde a interação com o sistema universitário não mais necessita da presença do aluno na instituição e várias funcionalidades sequer seriam viáveis sem o uso dessa tecnologia. Com isso, existe a necessidade da criação de um aplicativo para aprimorar a interação do aluno da UNIRIO com o sistema da universidade, assim como nas atividades do dia a dia no campus. O próximo capítulo irá abordar a criação desse aplicativo.

3 Aplicativo UNIRIO Mobile

3.1 Visão Geral do Aplicativo

O objetivo deste projeto é criar um aplicativo que apresente informações importantes sobre a UNIRIO em um dispositivo móvel. Para tanto, parte das informações que estão no sistema PLONE da UNIRIO serão formatadas de acordo com um padrão previamente estabelecido, de modo que possam ser lidas pelo aplicativo e apresentadas de forma adequada aos usuários.

A coleta de dados começará na página inicial do aplicativo no PLONE[7], que conterá títulos e links para as páginas de informação. Ao entrar no aplicativo, o usuário verá uma página com o logo da UNIRIO e, clicando nesta página, será direcionado para o menu principal com todas as opções contidas na página inicial do aplicativo. Ele poderá selecionar uma das opções deste menu.

Figura 11 - Exemplo de página para o aplicativo

[7] http://www.unirio.br/prograd/aplicativo/aplicativo

A Figura 11 mostra a área de interesse de uma página de informação (circundada em vermelho). É possível perceber que esta página tem cabeçalhos (circundados em verde), que dividem o conteúdo. Como a página pode ter muita informação para as telas pequenas dos dispositivos móveis, ela será dividida em partes usando os cabeçalhos como referência.

Figura 12 - Exemplo de página para o aplicativo

A Figura 12 apresenta uma página com diversos cabeçalhos e a divisão das informações que serão apresentadas pelo aplicativo. Cada parte em laranja será apresentada em uma tela separada no aplicativo, de modo a não "inundar" o usuário com muitas informações em uma tela.

Ao selecionar uma opção do menu principal, o aplicativo recuperará as informações do link associado à esta opção e apresentará ao usuário da seguinte forma:

- Se a página tiver cabeçalhos, o aplicativo apresentará todo o texto e as imagens até o primeiro cabeçalho seguidos de um menu com uma opção para cada cabeçalho da página;

    - Esta página terá uma opção para que o usuário retorne para o menu principal. Ela também deverá apresentar a data em que as informações foram colhidas para que o usuário tenha uma ideia sobre quão atualizadas estão estas informações;
    - Se o usuário selecionar uma das opções do menu de cabeçalhos, ele será levado a uma nova tela com as informações específicas daquele cabeçalho (em laranja na figura acima);
    - A página com informações detalhadas do cabeçalho conterá uma opção para que o usuário retorne para a página de informações.
- Se a página não tiver cabeçalhos, o aplicativo apresentará todo o texto e as imagens contidos na página de forma corrida.
    - Esta página terá uma opção para que o usuário retorne para o menu principal. Ela também deverá apresentar a data em que as informações foram colhidas para que o usuário tenha uma ideia sobre quão atualizadas estão estas informações.

3.2 Tecnologias Utilizadas

Para a construção do aplicativo foi escolhido o framework IONIC [8], que permite a criação de aplicativos móveis híbridos, ou seja, que funcionam em múltiplas plataformas. Para uma melhor compreensão da arquitetura de um aplicativo híbrido, serão apresentadas nessa seção as diferentes tecnologias que compõem a arquitetura da construção de um aplicativo IONIC.

3.2.1 Node.js

Node.js [9] é uma plataforma para o desenvolvimento de aplicações web escaláveis, de alto desempenho, construída sobre a *engine* V8[10]. A *V8* foi criada pelo Google e é usada como interpretador de JavaScript em seu navegador, o Chrome. Historicamente, o JavaScript é usado para execução de *scripts* do lado do cliente, os quais são incorporados no HTML de uma página da Web e executados pelo navegador. O Node.js permite que desenvolvedores usem JavaScript no lado do servidor, possibilitando a produção de conteúdo *web* antes que a página seja enviada ao navegador do usuário.

O Node.js usa uma arquitetura baseada em um loop de eventos, onde todas as requisições são interpretadas de forma assíncrona em uma única *thread*. Isso o torna notoriamente rápido ao lidar com um número alto de requisições [9].

3.2.2 Apache Cordova

Cordova[11] é um *framework* de código aberto para o desenvolvimento de aplicativos para plataformas móveis. Ele permite o uso de tecnologias e linguagens de programação padrões da *web*, como HTML5, CSS3 e JavaScript, para o desenvolvimento de aplicativos compatíveis com múltiplas plataformas, evitando a necessidade de desenvolvimento usando a linguagem nativa de cada uma das plataformas. Esses aplicativos são executados em *wrappers* configurados para a plataforma desejada, que cuidam do acesso a recursos de hardware, como câmera, armazenamento de dados e sensores.

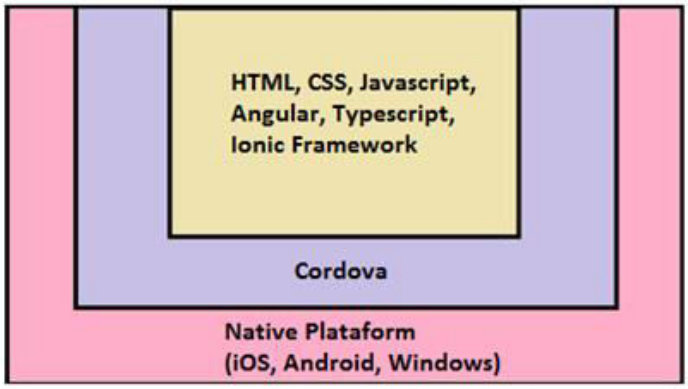

Figura 13 - Arquitetura macro de um app híbrido

Na Figura 13 temos uma visão da camada preenchida pelo Cordova na arquitetura de um aplicativo híbrido. O código, escrito nas diferentes linguagens citadas na figura, é traduzido e direcionado pelo Cordova para a plataforma escolhida.

Na Figura 14 podemos ver destacados os componentes principais da arquitetura do Cordova. O componente *WebView* fornece ao aplicativo toda a sua interface com o usuário. Em algumas plataformas, ele pode ser um componente dentro de um aplicativo híbrido maior, que mistura o *WebView* com componentes nativos.

O componente *WebApp* é onde o código do aplicativo reside. O aplicativo em si é implementado como uma página da Web, que, por padrão, possui um arquivo local chamado *index.html*, que faz referência a *CSS, JavaScript*, imagens, arquivos de mídia ou outros recursos necessários para a sua execução. O aplicativo é executado em um *WebView* dentro do *wrapper* de aplicativo nativo. Nesse componente há um arquivo chamado *config.xml*, que contém parâmetros que definem o comportamento do aplicativo no dispositivo móvel (e.g., a forma com que deve responder a mudanças de orientação posicional do dispositivo) e o ícone que será exibido na lista de aplicativos.

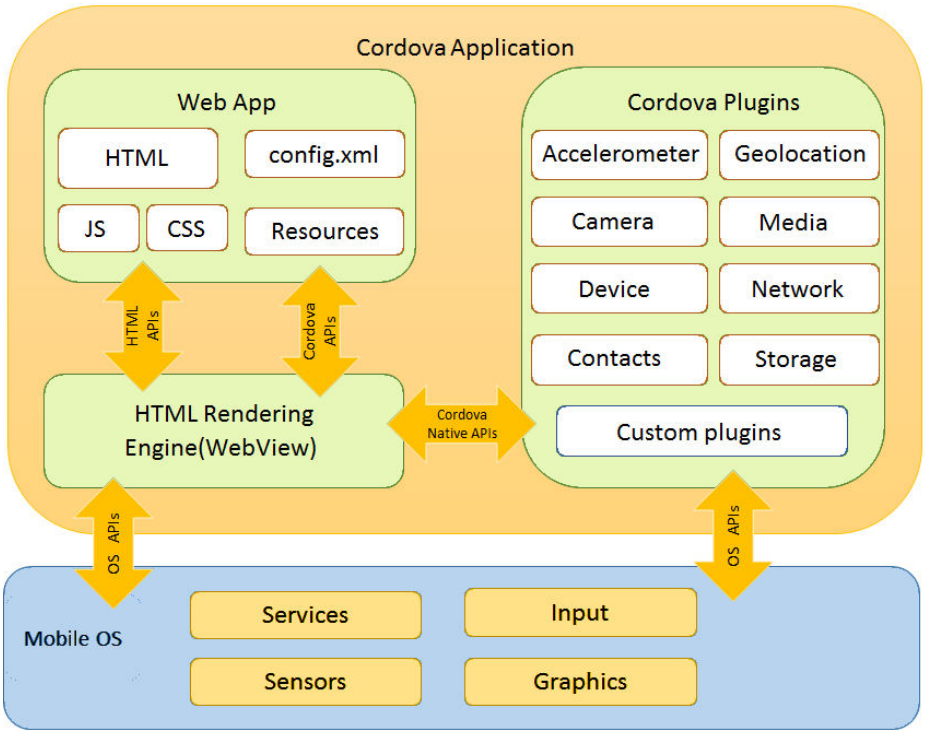

Figura 14 - Visão de alto nível da arquitetura do Apache Cordova [11]

Os *plugins* fornecem um conjunto de interfaces para que o Cordova se comunique com os componentes nativos do sistema operacional do dispositivo móvel. O projeto Apache Cordova mantém um conjunto de *plugins* chamado de "*Core plugins*". Esses *plugins* fornecem acesso a recursos do dispositivo, como bateria, câmera, contatos, etc. Além desses "*Core plugins*", há *plugins* disponíveis na Internet, desenvolvidos por terceiros, que fornecem acesso a recursos não necessariamente disponíveis em todas as plataformas. Plugins como estes podem ser desenvolvidos por qualquer um, conforme descrito no Guia de desenvolvimento de plug-ins [12].

3.2.3 Typescript

O TypeScript [13] é uma linguagem de programação de código aberto desenvolvida e mantida pela Microsoft. A linguagem originou-se das deficiências do *JavaScript* para o desenvolvimento de aplicativos de grande porte, tanto na Microsoft quanto entre seus clientes. Para a solução desse problema, os desenvolvedores do TypeScript procuraram não apenas uma solução para a criação de aplicações de grande porte, mas que também cuidasse da manutenção de compatibilidade com os padrões multiplataforma. Sendo assim, é possível utilizar o Typescript para o desenvolvimento de aplicações que sejam executadas no lado cliente (navegador de Internet) ou no lado servidor (Node.js).

O TypeScript suporta arquivos de definição (*headers*) que podem conter informações de bibliotecas JavaScript existentes, assim como os arquivos de cabeçalho de C/C++. Isso permite que outros programas usem os valores definidos nos arquivos como se fossem entidades TypeScript digitadas estaticamente. Existem arquivos de cabeçalho de terceiros para bibliotecas populares, como jQuery, MongoDB e D3.js. Cabeçalhos TypeScript para os módulos básicos do Node.js também estão disponíveis, permitindo o desenvolvimento de programas Node.js usando TypeScript.

3.2.4 Angular

Angular é uma plataforma e *framework* de código aberto para criação de aplicações em HTML e *TypeScript*. O Angular disponibiliza diversas funcionalidades na forma de um conjunto de bibliotecas que podem ser importadas e usadas como componentes em módulos *Typescript.* Embora o Angular tenha surgido como uma segunda versão do *Angular.js*, vale notar que não é uma atualização deste e sim um novo *framework* completamente reescrito. O Angular, assim como seu predecessor, o Angular.js, foram criados e são mantidos pelo Google.

Os elementos básicos de construção de uma aplicação Angular são chamados de *NgModules*, que fornecem um contexto para a interpretação dos *componentes*. Os *NgModules* fazem um agrupamento de códigos em conjuntos funcionais, de forma análoga a um *Package* em uma aplicação Java. Uma aplicação Angular deve possuir ao menos um *NgModule*.

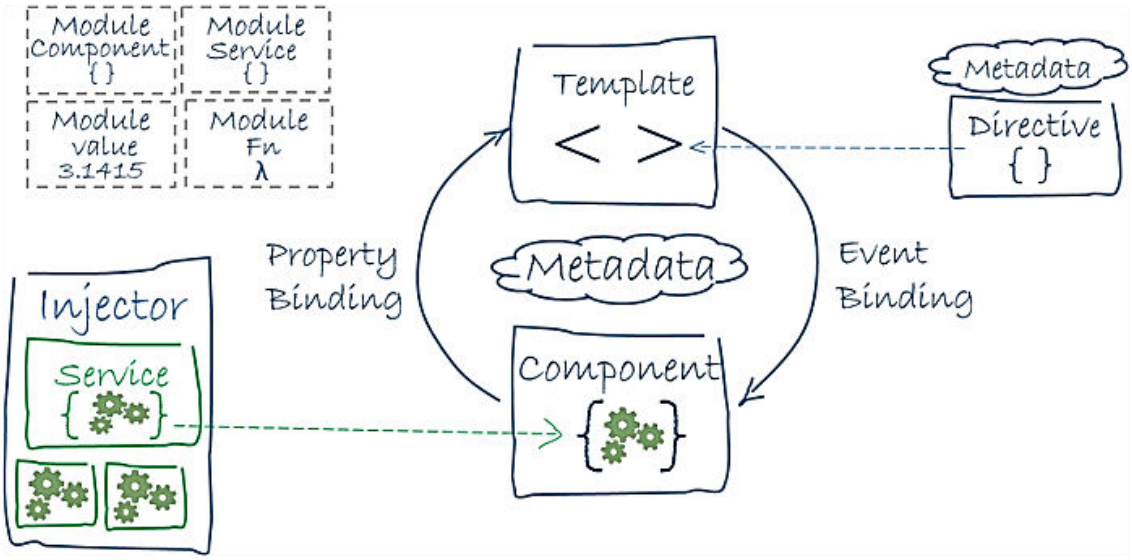

Figura 15 - Visão geral da arquitetura do Angular [14].

Com a ajuda da Figura 15, podemos identificar as sete unidades principais de construção de uma aplicação Angular.

- **Component**: implementa a lógica que trata a exibição de dados em uma *view*. Em outras palavras, é o *controller* de uma *view;*
- **Metadata***: o*s metadados de um componente indicam ao Angular onde obter os principais blocos de construção necessários para criar e apresentar o componente e sua *view*;
- **Template**: define a *view* do componente. De forma resumida, os *Templates* são o HTML molde da página;
- **Data Binding:** realiza a conexão dos dados da aplicação com o DOM;
- **Directives:** permite alterar o comportamento de elementos DOM. De forma resumida, permite acrescentar lógica de programação ao HTML;
- **Services and Dependency Injection:** para dados ou lógica que não estão associados a uma *view* específica e que devem ser compartilhados entre diferentes componentes, cria-se uma *Service Class*. A definição de uma *Service class* é precedida por *@Injectable (),* que fornece os metadados que permitem que seu serviço seja injetado nos componentes do cliente como uma dependência *(dependency injection)*.

3.2.5 IONIC Framework

O IONIC é um framework de código aberto para desenvolvimento de aplicativos para dispositivos móveis utilizando HTML5. Ele suporta versões iOS7, ou superior, em dispositivos Apple, e Android 4.1, ou superior, em dispositivos Android. Os aplicativos IONIC são híbridos, ou seja, funcionam em múltiplas plataformas e, para tal, fazem uso do *framework* Cordova. Ele usa componentes JavaScript e CSS, *plugins* do Apache Cordova e a arquitetura MVC do Angular. A instalação do IONIC disponibiliza vários aplicativos-modelo que servem como ponto de partida para o desenvolvimento.

3.3 Preparação do Ambiente

Para o desenvolvimento de qualquer aplicativo, é necessário definir alguns requisitos e configurações. O framework IONIC, versão 4.2.1, usado neste projeto é

direcionado para iPhone e Android. Essa versão do IONIC suporta dispositivos Android versão 4.1 ou superior, e versão iOS7 ou superior. Após o desenvolvimento, o Cordova deve ser implantado na plataforma desejada, para que então o aplicativo IONIC possa ser implantado.

Os aplicativos IONIC são criados e desenvolvidos principalmente por meio de uma interface de linha de comando do IONIC (o "CLI") e usam o *Cordova* para criar e implantar como um aplicativo nativo. Isso significa que é necessária a instalação de alguns utilitários para iniciar o desenvolvimento.

3.3.1 Node e NPM

A maioria das ferramentas no CLI é baseada no Node.js e gerenciada por meio do *npm*, portanto, o primeiro passo é a instalação do Node.js[8]. Ao término da instalação, será possível chamar o *npm* pela linha de comando do sistema operacional.

3.3.2 IONIC e Cordova CLI

A instalação do Cordova, e do IONIC propriamente dito, são feitas em conjunto a partir do utilitário npm do Node.js[9]. Uma única linha de comando realiza a instalação das duas ferramentas (*npm install -g ionic cordova*).

3.3.3 Angular e TypeScript

O Angular[10] e TypeScript[11] também devem ser instalados pelo *npm.*

3.3.4 IDE Visual Studio Code

O Visual Studio Code foi a IDE escolhida para a edição do código desse projeto, por ser uma ferramenta gratuita e de código aberto, desenvolvida pela Microsoft para Windows, Linux e MacOs [15]. Essa ferramenta também oferece suporte nativo às linguagens utilizadas no projeto.

3.3.5 Android SDK

Um kit de desenvolvimento de software (SDK[12]) é um conjunto de ferramentas de desenvolvimento de software que permite a criação de aplicativos para uma

[8] https://nodejs.org/en/download/
[9] https://ionicframework.com/docs/cli/
[10] https://www.npmjs.com/package/angular
[11] https://www.npmjs.com/package/typescript

determinada plataforma. No caso do Android, é o Android SDK. O Android SDK é fornecido pelo Google em conjunto com sua IDE oficial, o Android Studio [17].

3.3.6 Criação de um Projeto IONIC

Com a instalação das ferramentas descritas nas subseções anteriores, é possível criar um novo projeto a partir da interface de linha de comando do IONIC (CLI). O CLI disponibiliza diversos modelos de novos projetos. Para o desenvolvimento do aplicativo UNIRIO mobile foi usado um projeto novo vazio, que pode ser criado pelo comando "*ionic start helloWorld blank*".

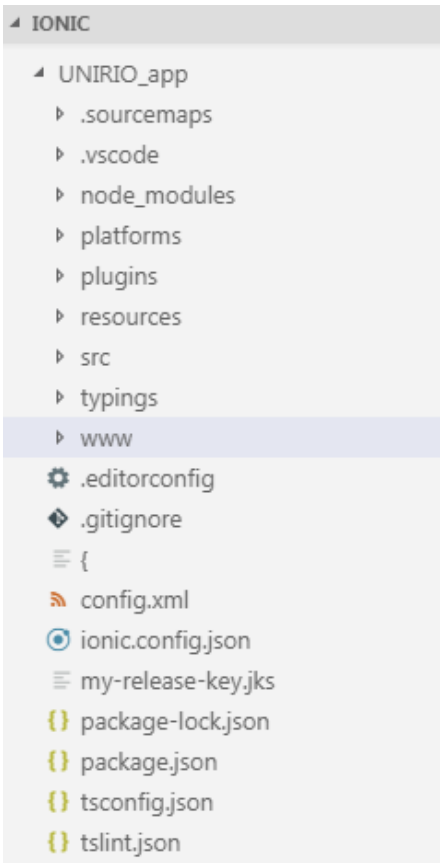

Figura 16 - Visão geral do sistema de arquivos de um projeto IONIC.

A Figura 16 mostra a estrutura de arquivos de um projeto IONIC no *Visual Studio Code*. Este projeto modelo já possui a estrutura necessária para gerar um aplicativo *helloworld*, que pode ser implantado em um dispositivo móvel Android ou iOS.

[12] https://en.wikipedia.org/wiki/Software_development_kit

3.4 Considerações Finais

Após uma visão geral dos requisitos funcionais do projeto, das tecnologias necessárias para seu desenvolvimento, e da montagem do ambiente de trabalho, é possível começar a implementação do aplicativo. O próximo capítulo irá abordar essa implementação.

4 **Implementação**

4.1 Introdução

Neste capítulo será apresentado o funcionamento do aplicativo, ilustrado através da navegação de suas diferentes telas e funcionalidades. Porém, também é importante uma abordagem sobre a organização do código e os diferentes *plugins* utilizados. As próximas subseções irão discutir esses assuntos.

4.2 Navegação pelo Aplicativo

Conforme descrito no capítulo 3, o objetivo do aplicativo é ler e tratar as informações contidas em páginas de uma determinada seção do sistema PLONE. A partir desse tratamento, as informações devem ser reorganizadas e adaptadas para exibição na tela do celular. Como o aplicativo vai fazer uma varredura e leitura de um número indefinido de páginas, foi necessária a criação de um algoritmo tratador dessas páginas (em HTML) e a criação de três modelos de páginas, onde essas informações serão exibidas. Essas página-modelo são: Página Inicial, Página de Informação e Página de Detalhe. Cada uma delas é gerada dinamicamente para acomodar as informações disponíveis no PLONE.

4.2.1 Página Inicial

A tela inicial do aplicativo, mostrada na Figura 17, exibe as seções de informação contidas da página principal do PLONE. O código responsável pela criação dessa tela identifica e exibe, na forma de botões, apenas os links presentes na página HTML inicial do PLONE. É a partir dessa tela que será feita toda a navegação pelo aplicativo.

Figura 17 - Tela inicial do aplicativo

4.2.2 Páginas de Informação

Os botões da página inicial do aplicativo levam às Páginas de Informação, que é onde reside o conteúdo presente no PLONE. As Figuras 18 e 19 mostram quatro exemplos de páginas de informação. Essas telas também são construídas dinamicamente a partir do conteúdo das páginas destino dos *links* da página inicial. O código responsável pela exibição dessas páginas de informação trata conteúdo de texto, imagens, tabelas, links externos e também cria botões para subseções dessas páginas de informação. Esses botões levam às páginas de detalhe.

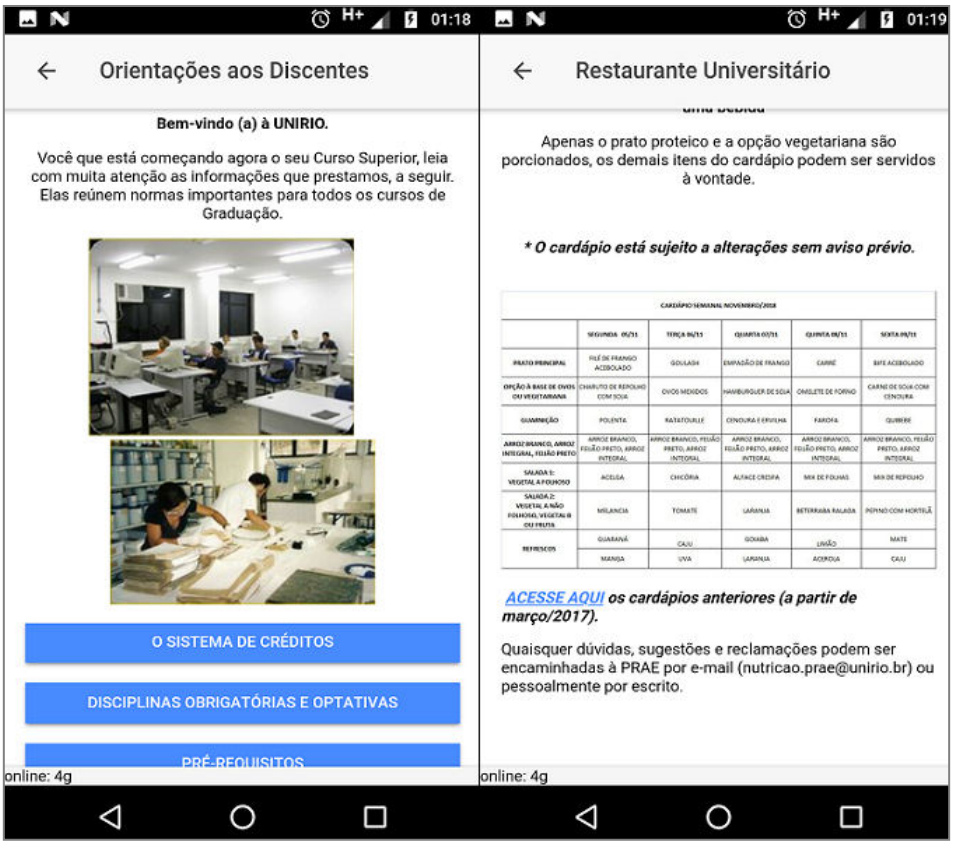

Figura 18 – Exemplos de páginas de informação com texto e imagens

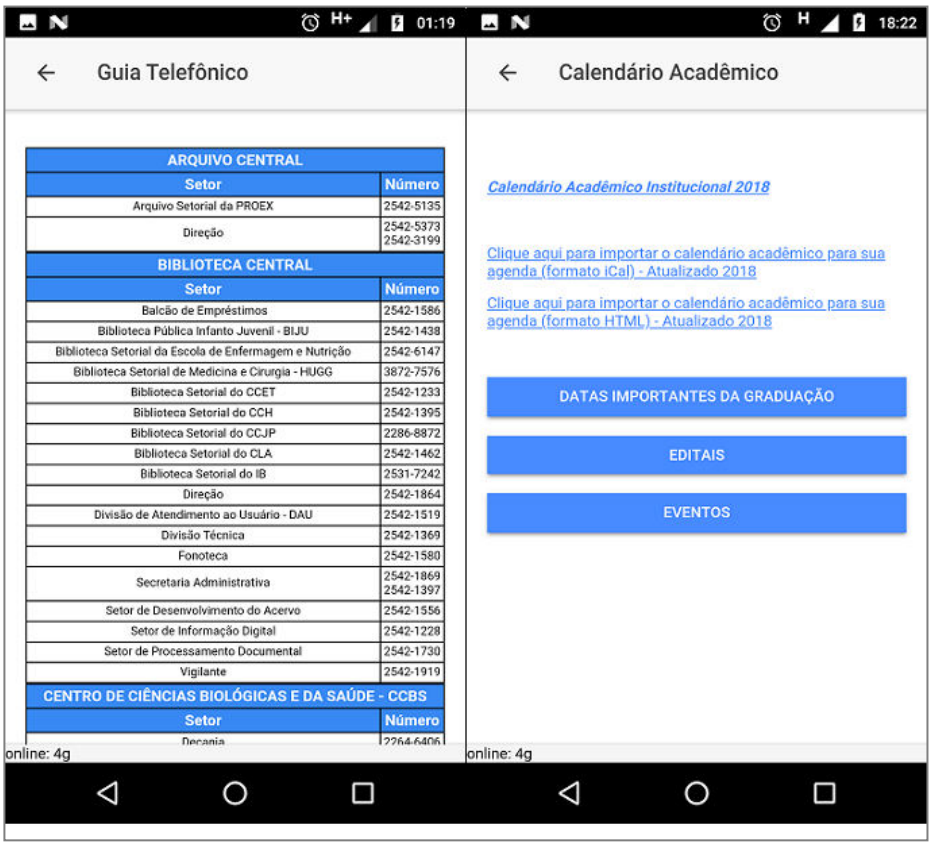

Figura 19 - Exemplos de páginas de informação com tabelas e links externos

4.2.3 Páginas de Detalhe

Essas telas são criadas a partir de subseções das telas das páginas de informação, com o intuito de evitar o excesso de informação em suas páginas de origem. O conteúdo exibido nessa página de detalhe é o HTML presente entre as *tags* com *class callout* identificadas na página de origem. As páginas de detalhe não criam botões para subseções.

Figura 20 - Exemplo de tela de detalhe exibindo apenas texto.

A Figura 20 mostra duas telas. Na direita, uma página de detalhe contendo apenas texto, e na esquerda, a página de informação que contém, em destaque, o botão para seu acesso. A Figura 21 exibe um exemplo similar, com uma tela de detalhe contendo uma tabela.

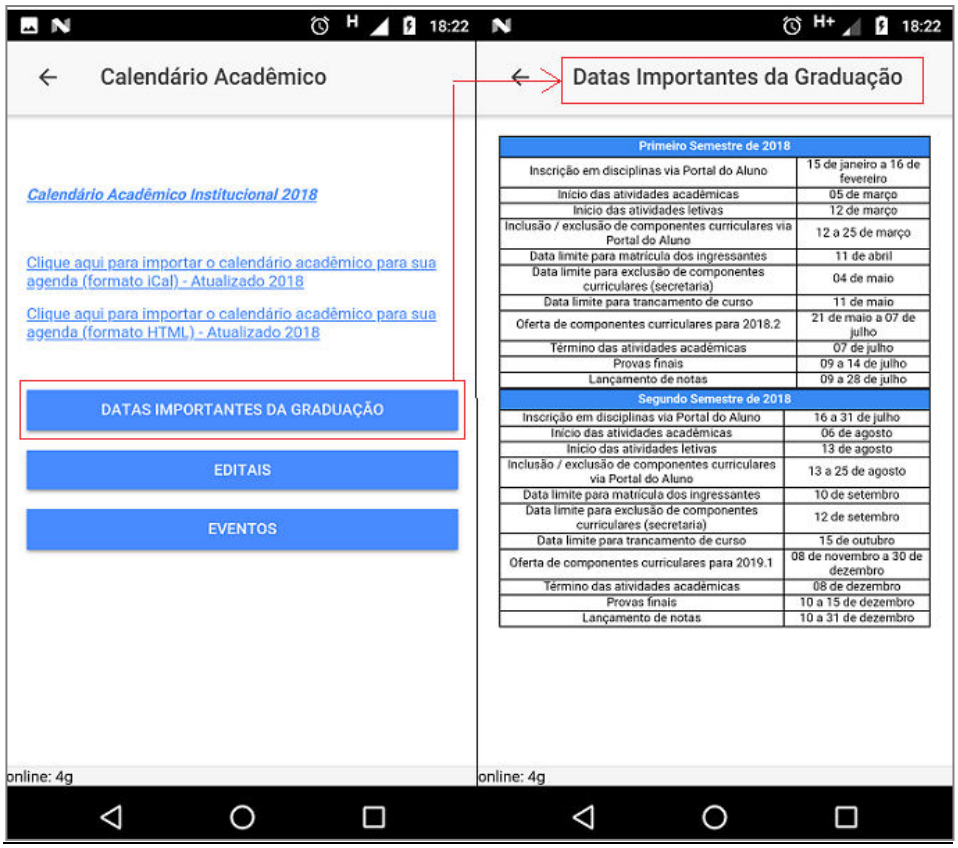

Figura 21- Exemplo de tela de detalhe exibindo uma tabela.

4.3 *Plugins*

Um dos principais requisitos do aplicativo é a capacidade de navegação *offline*, desde que, pelo menos uma vez, o usuário tenha usado o aplicativo conectado à Internet. No primeiro acesso, o aplicativo faz o *download* das páginas do PLONE para sistema de arquivos do dispositivo móvel, para que nos demais acessos sejam buscados esses arquivos caso seja identificado ausência de conectividade com a Internet. Para atender tal requisito, foi necessário o uso de três *plugins* que não fazem parte da instalação padrão do IONIC e precisaram ser instalados separadamente pela linha de comando.

4.3.1 File

Este *plugin*, desenvolvido pela própria equipe do IONIC, implementa uma API que permite acesso de leitura e gravação ao sistema de arquivo do dispositivo[13].

[13] https://ionicframework.com/docs/native/file/

4.3.2 Network Provider

Este *plugin*, também desenvolvido pela equipe do IONIC, implementa uma API que disponibiliza acesso ao status de conectividade do dispositivo móvel. É através da funcionalidade deste *plugin* que o aplicativo identifica se deve buscar os dados na Internet, ou no sistema de arquivos local do dispositivo. Caso não seja identificada uma conexão com a Internet e também não existam arquivos previamente armazenados, uma mensagem de erro será exibida. A Figura 22 mostra a tela de mensagem de erro exibida quando o primeiro acesso ao aplicativo é feito sem conectividade com a Internet.

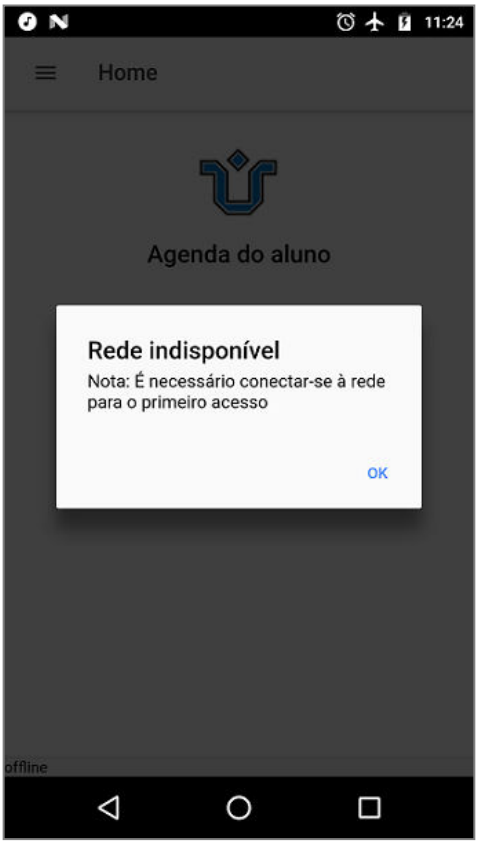

Figura 22 - Mensagem de erro

4.3.3 Loading Provider

Este *plugin* disponibiliza uma tela de *loading* que é usada enquanto o aplicativo carrega as páginas do PLONE. Essa funcionalidade é útil para, caso haja algum tipo de lentidão na conexão do dispositivo com a Internet, deixar claro ao usuário que o aplicativo não está inoperante, e sim, aguardando a carga de dados. A Figura 23 mostra a tela do aplicativo aguardando conexão com a Internet e posterior carga de dados.

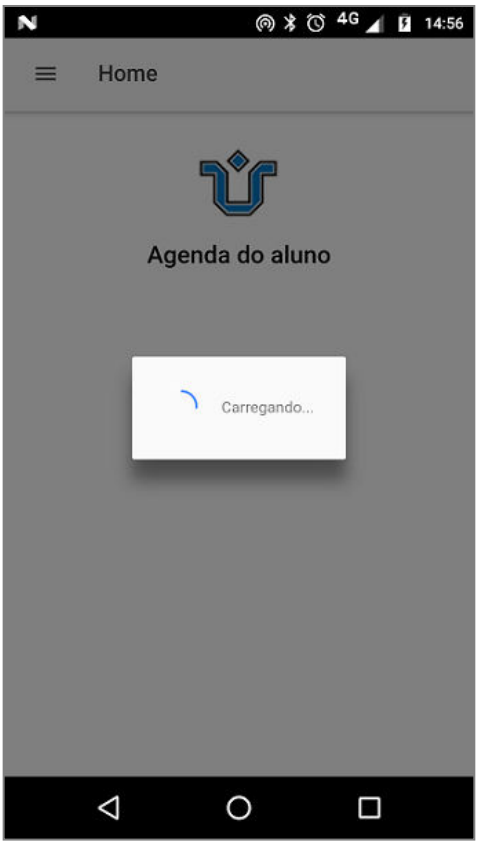

Figura 23 - Tela de *loading*

4.4 Estrutura do Código

Os principais pontos que devem ser destacados em relação à estrutura do código são: a criação das *views* das páginas descritas na seção 4.2 e suas respectivas classes controladoras, a declaração dos *plugins* e os ajustes nos arquivos de configuração. O código do aplicativo está disponível em https://github.com/rodcezar/UNIRIO-mobile.

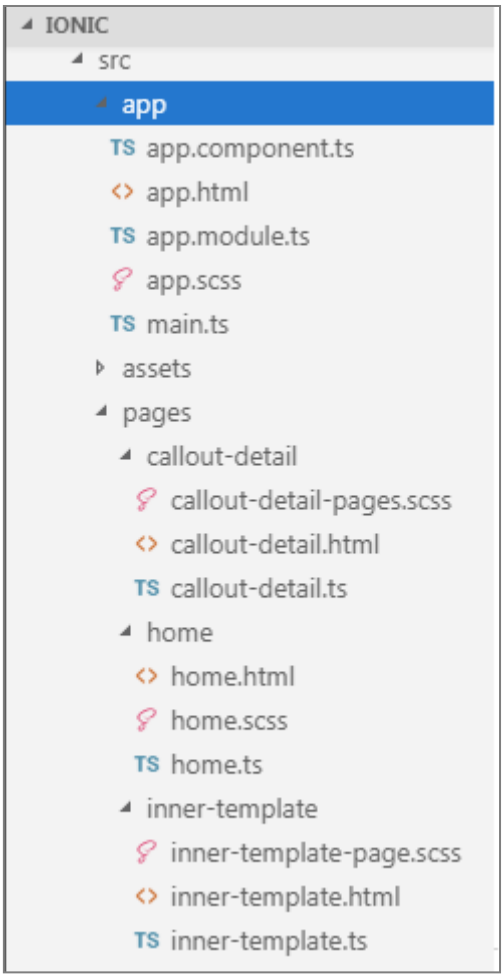

Figura 24 - Sistema de arquivos do projeto

A Figura 24 mostra uma visão geral da estrutura do projeto. A pasta *pages* traz a implementação de cada uma das páginas descritas na seção 4.2, sendo a pasta *home* referente à Página Inicial, a pasta *inner-template* referente às Páginas de Informação e a pasta *callout-detail* referente às Páginas de Detalhe. Em cada uma dessas pastas, o arquivo HTML e o CSS compõem a *view* da página e sua classe controladora é implementada no módulo *Typescript* (arquivo .ts).

Os *plugins* adicionais são declarados no módulo *app_module.ts* para que possam ser usados em outros módulos, conforme mostrado na Figura 25.

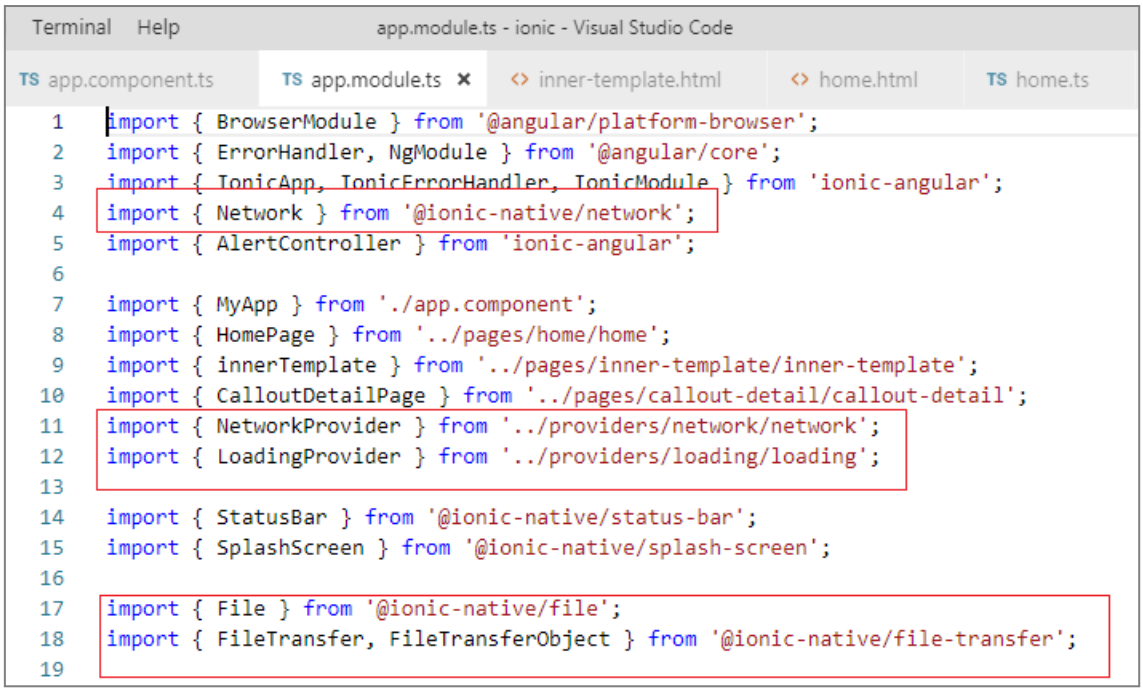

Figura 25 - Fragmento do app_module.ts

Outro ponto importante que deve ser abordado é com relação ao arquivo *config.xml*, localizado na pasta raiz do projeto. Esse arquivo foi modificado para indicar a localização do ícone da Unirio para exibição na lista de aplicativos do dispositivo. A Figura 26 mostra diversas linhas indicando caminhos para ícones da Unirio. A necessidade desses múltiplos arquivos é em razão das diversas resoluções existentes nas telas dos dispositivos móveis.

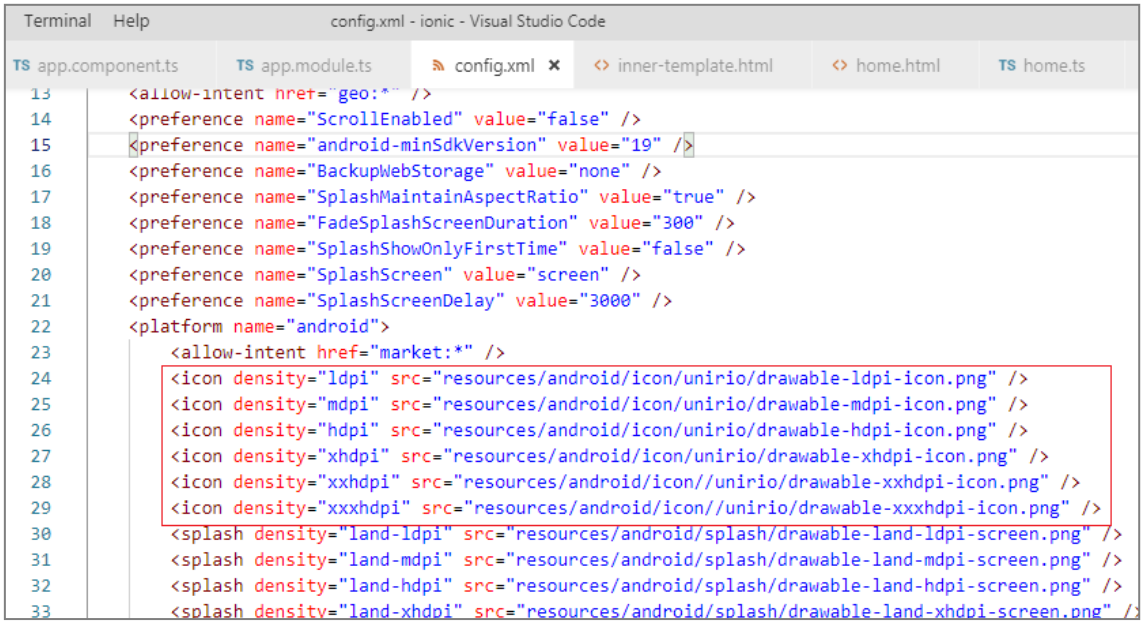

Figura 26 - Fragmento do arquivo config.xml

4.5 Considerações Finais

Conforme apresentando neste capítulo, o aplicativo UNIRIO *Mobile* é capaz de fazer uma leitura dos dados no sistema PLONE e apresentá-los em um dispositivo móvel. Além disso, o aplicativo também é capaz de armazenar esses dados no sistema local de arquivos, dispensando a necessidade de conexão com a Internet, após um primeiro acesso conectado.

Na sua atual forma, é um aplicativo simples, cuja principal ideia é ser um embrião para futuras versões com mais funcionalidades.

5 Conclusão

5.1 Considerações Finais

Neste trabalho foi desenvolvido o aplicativo UNIRIO *mobile*. Para tal, foi necessário o aprendizado e uso de diversas tecnologias e linguagens modernas de programação *Web*, assim como um estudo da arquitetura para desenvolvimento de aplicativos híbridos para aparelhos móveis. Em relação às tecnologias empregadas neste trabalho, em especial o IONIC, vale destacar algumas impressões:

- Rápida curva de aprendizado: a literatura necessária para se começar a desenvolver aplicativos utilizando o framework IONIC é bem simples se comparada à das opções de desenvolvimento nativo, como *IOS* e *Android*;
- Boa documentação: a página oficial do IONIC disponibiliza, de forma clara e organizada, um guia completo para a montagem do ambiente de trabalho e uma série de aplicativos-modelo que são ótimos pontos de partida, não apenas para o aprendizado, mas também o desenvolvimento de um novo aplicativo;
- Grande diversidade de *Plugins* e componentes: por ser uma ferramenta de código aberto e gratuita, existe grande participação de desenvolvedores de todo o mundo na elaboração de novas funcionalidades para o IONIC.

A partir dessas impressões, é razoável afirmar que o IONIC é uma excelente opção para o desenvolvimento de aplicativos como o que foi desenvolvido nesse trabalho, cuja funcionalidade é similar à de um pequeno *website*. O IONIC não só dispõe de todos os recursos necessários para atender os requisitos de um aplicativo dessa natureza, como também possui aparência e usabilidade praticamente indistintas a de um aplicativo desenvolvido nativamente.

Entretanto, é fácil enxergar as limitações do IONIC. Conforme discutido no Capítulo 2, existem certas funcionalidades de *hardware* cujo acesso é disponível apenas

através do desenvolvimento nativo. O IONIC também não dispõe de recursos para o desenvolvimento de aplicações que necessitam de alto desempenho em processamento gráfico, como jogos.

Portanto, a conclusão a respeito do uso do IONIC, ou tecnologia nativa, depende do tipo de aplicativo a ser desenvolvido. Caso o aplicativo possa ser desenvolvido usando o IONIC, sem que haja perda de funcionalidade ou desempenho em relação ao desenvolvimento usando tecnologia nativa, o IONIC é a melhor opção. Pode ser construído em menos tempo, com custo menor e mantendo a mesma qualidade.

5.2 Trabalhos Futuros

Trabalhos futuros no sentido de evoluir esse aplicativo e incluir novos recursos podem buscar uma integração mais abrangente com o sistema PLONE, permitindo que o aplicativo acesse aos horários de disciplinas e seções distintas para cada curso da UNIRIO. Isto permitiria que o aplicativo se adaptasse melhor ao contexto do aluno que o instalou, reconhecendo seu curso, suas disciplinas e apresentando informações relevantes sobre elas. Para tal, seria interessante uma integração com o portal do aluno, possibilitando aos alunos a inscrição, alteração ou cancelamento de disciplinas através de uma interface formatada para os dispositivos móveis.

6 **Referências Bibliográficas**

[1] “Number of Mobile Phone Users from 2012 to 2018”. Disponível em: http://www.statista.com/statistics/274774/forecast-of-mobile-phone-users-worldwide/ Acesso em: 02 de setembro de 2018..

[2]“The Ericsson Mobility Report Q2 2018”. Disponível em: <http://www.ericsson.com/mobility-report> Acesso em: 05 de setembro de 2018

[3] “Mobile app”. Disponivel em: <https://en.wikipedia.org/wiki/Mobile_app> Acesso em: 19 de setembro de 2018

[4] “Native, HTML5, or Hybrid: Understanding Your Mobile Application Development Options”. Disponível em: <https://developer.salesforce.com/page/Native,_HTML5,_or_Hybrid:_Understanding_Your_Mobile_Application_Development_Options > Acesso em: 19 de setembro de 2018

[5] “Mobile Operating System Market Share Worldwide”. Disponível em: < http://gs.statcounter.com/os-market-share/mobile/worldwide > Acesso em: 19 de setembro de 2018

[6] “Number of available applications in the Google Play Store from December 2009 to June 2018”. Disponível em: <https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/> Acesso em: 19 de setembro de 2018

[7] “iOS”. Disponível em: < https://en.wikipedia.org/wiki/IOS > Acesso em: 19 de setembro de 2018.

[8] “What is Ionic Framework?”. Disponível em: <https://ionicframework.com/docs/intro/concepts> Acesso em: 19 de setembro de 2018

[9] Site oficial do nodeJS. Disponível em: < https://nodejs.org/en/ > Acesso em: 19 de setembro de 2018

[10]”What is V8?”. Disponível em: < https://v8.dev/ > Acesso em: 19 de setembro de 2018

[11] Site oficial do Cordova. Disponível em: <https://cordova.apache.org/docs/en/latest/guide/overview/index.html> Acesso em: 19 de setembro de 2018

[12] “*Plugin Development Guide”*. Disponível em: < https://cordova.apache.org/docs/en/latest/guide/hybrid/plugins/index.html > Acesso em: 19 de setembro de 2018

[13] Anders Hejlsberg (2012-10-05). "What is TypeScript and why with Anders Hejlsberg". Disponível em: < https://www.hanselminutes.com/340/what-is-typescript-and-why-with-anders-hejlsberg > Acesso em: 19 de setembro de 2018

[14] Página oficial do Angular. Disponível em: < https://angular.io/guide/architecture > Acesso em: 19 de setembro de 2018

[15] Frederic Lardinois (2015-04-29). “"Microsoft Launches Visual Studio Code, A Free Cross-Platform Code Editor For OS X, Linux And Windows". Disponível em:

<https://techcrunch.com/2015/04/29/microsoft-shocks-the-world-with-visual-studio-code-a-free-code-editor-for-os-x-linux-and-windows/ > Acesso em: 19 de setembro de 2018

[16] Página oficial do Android Studio. Disponível em: < https://developer.android.com/studio/ > Acesso em: 19 de setembro de 2018