

UNIVERSIDADE DO VALE DO SAPUCAÍ
ALLAN THOMAS OLIVEIRA FRANCO
FLÁVIO FRANCISCO DO NASCIMENTO

APLICAÇÃO MÓVEL HÍBRIDA PARA A COMISSÃO PRÓPRIA DE
AVALIAÇÃO USANDO IONIC

POUSO ALEGRE, MG

2016

UNIVERSIDADE DO VALE DO SAPUCAÍ
ALLAN THOMAS OLIVEIRA FRANCO
FLÁVIO FRANCISCO DO NASCIMENTO

APLICAÇÃO MÓVEL HÍBRIDA PARA A COMISSÃO PRÓPRIA DE
AVALIAÇÃO USANDO IONIC

Trabalho de Conclusão de Curso
apresentado ao Curso de Sistemas de
Informação da Universidade do Vale do
Sapucaí como requisito parcial para a
obtenção do título de bacharel em Sistemas
de Informação.

Orientador: Prof. MSc. Márcio Emílio Cruz
Vono de Azevedo.

POUSO ALEGRE, MG

2016

FRANCO, Allan Thomas Oliveira; NASCIMENTO, Flávio Francisco.
**APLICAÇÃO MÓVEL HÍBRIDA PARA A COMISSÃO PRÓPRIA DE
AVALIAÇÃO USANDO IONIC.** Trabalho de Conclusão de Curso –
Universidade do Vale do Sapucaí, Univás, Bacharelado em Sistemas de
Informação – Pouso Alegre – MG: Univás, 2016. 57p.

1. Ionic. 2. CPA. 3. Mobile. CDD: 004

UNIVERSIDADE DO VALE DO SAPUCAÍ
ALLAN THOMAS OLIVEIRA FRANCO
FLÁVIO FRANCISCO DO NASCIMENTO

APLICAÇÃO MÓVEL HÍBRIDA PARA A COMISSÃO PRÓPRIA DE
AVALIAÇÃO USANDO IONIC

Trabalho de conclusão de curso defendido e aprovado em 24/11/2016 pela banca
examinadora constituída pelos professores:

Prof. MSc. Márcio Emílio Cruz Vono de Azevedo
Orientador

Prof. Ednardo David Segura
Examinador

Prof. MSc. Valéria Santos Paduan Silva
Examinador

AGRADECIMENTOS

De Allan Thomas Oliveira Franco.

Agradeço em primeiro lugar a Deus, por ter me concedido saúde e disposição para prosseguir com essa jornada exaustiva durante quatro anos, sem nunca ter perdido a vontade de crescer, evoluir e seguir em frente. Ao meu amado pai, Gilmar Batista Franco, que me deu forças e me motivou para que eu conseguisse terminar meus estudos, sempre se sacrificando ao máximo para me ajudar. A todos os professores que tanto contribuíram para minha formação durante esse período de faculdade, em especial ao orientador Márcio Emílio Cruz Vono de Azevedo, que sempre esteve disposto a ajudar no que fosse necessário. Ao meu companheiro de TCC, Flávio Francisco do Nascimento, que, com muita força de vontade caminhou comigo durante todos esses anos. Agradeço a todos os meus amigos e familiares, que de maneira direta ou indireta contribuíram neste trabalho. E também a todos os meus colegas de faculdade, que estiveram comigo durante esses quatro anos, sempre dispostos a ajudar no que fosse possível.

De Flávio Francisco do Nascimento.

Agradeço primeiramente a Deus por me dar força e persistência para vencer todas as etapas do curso de Sistemas de Informação. Agradeço a meus pais Irineu Gonçalves do Nascimento e Maria José de Oliveira Nascimento que sempre me apoiaram nessa luta. Agradeço também a uma pessoa muito especial que me ajudou muito nesse caminho desde o início do curso, Angelita de Moraes. E a meu amigo Allan Tomas Oliveira Franco, que desde o início do curso, lutou junto e me ajudou muito, sempre trabalhando em equipe. Agradeço a meus sobrinhos Joao Vitor do Nascimento Silva, Ana Clara do Nascimento e João Gabriel Nascimento da Silva, por serem uma das razões de tanto esforço. Agradeço aos professores Márcio Emílio Cruz Vono de Azevedo e José Luiz da Silva representando todos os professores que me instruíram em minha formação. Agradeço a meus amigos Jonas Santos do Couto que me ajudou muito nos estudos, Diógenes Aparecido Rezende por nos conceder ótimas ideias e ajuda para o trabalho de conclusão de curso. E por fim agradeço a todos colegas de sala e a todos que contribuíram para que eu pudesse conseguir realizar esse objetivo de minha vida.

LISTA DE FIGURAS

Figura 1 - Diagrama de casos de uso do projeto.	23
Figura 2 - Instalação do Slim Framework.	26
Figura 3 - Instalação do Cordova e do Ionic.	27
Figura 4 - Criação de um novo projeto Ionic.	28
Figura 5 - Diretório raiz do projeto Ionic.	28
Figura 6 - Instalação da plataforma Android no projeto Ionic.	29
Figura 7 - Compilação do aplicativo.	29
Figura 8 - Interface do phpMyAdmin.	30
Figura 9 - Diagrama do banco de dados.	31
Figura 10 - Diretório raiz da API.	37
Figura 11 - Tela inicial do aplicativo.	47
Figura 12 - Tela do questionário.	48
Figura 13 - Tela de encerramento do questionário.	49
Figura 14 - Tela de questionário já respondido.	49
Figura 15 - Tela de login do administrador.	50
Figura 16 - Tela do administrador.	51
Figura 17 - Exemplo de gráfico.	52
Figura 18 - Confirmação para alterar status.	53
Figura 19 - Notificação da CPA	54

LISTA DE LISTAGENS

Listagem 1 - Configuração inicial do index.php.	26
Listagem 2 - Classe ConectionFactory.	32
Listagem 3 - Classe CPAService.	33
Listagem 4 - index.php.	35
Listagem 5 - ReportService, classe responsável pelos relatórios.	37
Listagem 6 - Operação fazerLogin do arquivo index.js.	39
Listagem 7 - Método buscaDisciplinas.	40
Listagem 8 - Método guardarDados.	41
Listagem 9 - Método salvarRespostas.	41
Listagem 10 - Métodos iniciar e alterarStatus.	42
Listagem 11 - Método responsável pela notificação.	43
Listagem 12 - Método gerarRelatorio.	44
Listagem 13 - Método converteData.	44
Listagem 14 - Host.js.	45

LISTA DE SIGLAS E ABREVIATURAS

API	<i>Application Programming Interface</i>
CONAES	Comissão Nacional de Avaliação da Educação Superior
CPA	Comissão Própria de Avaliação
CSS	<i>Cascading Style Sheets</i>
HTML	<i>HyperText Markup Language</i>
HTTP	<i>HyperText Transfer Protocol</i>
INEP	Instituto Nacional de Estudos e Pesquisas Educacionais
JSON	<i>JavaScript Object Notation</i>
MEC	Ministério da Educação
MVC	<i>Model-View-Controller</i>
MVP	<i>Model-View-Presenter</i>
MVVM	<i>Model-View-ViewModel</i>
MVW	<i>Model-View-Whatever</i>
NPM	<i>Node Package Manager</i>
PHP	<i>Hypertext Preprocessor</i>
REST	<i>Representational State Transfer</i>
SDK	<i>Software Development Kit</i>
SGBD	Sistema de Gerenciamento de Bancos de Dados
SINAES	Sistema Nacional de Avaliação da Educação Superior
SQL	<i>Structured Query Language</i>
UNIVÁS	Universidade do Vale do Sapucaí
URI	<i>Uniform Resource Identifier</i>
URL	<i>Universal Resource Locator</i>
WHATWG	<i>Web Hypertext Application Technology Working Group</i>
XML	<i>eXtensible Markup Language</i>

FRANCO, Allan Thomas Oliveira; NASCIMENTO, Flávio Francisco. **APLICAÇÃO HÍBRIDA PARA A COMISSÃO PRÓPRIA DE AVALIAÇÃO USANDO IONIC**. 2016. Monografia – Curso de SISTEMAS DE INFORMAÇÃO, Universidade do Vale do Sapucaí, Pouso Alegre – MG, 2016.

RESUMO

A presente pesquisa, que é chamada de aplicada, se direciona para o desenvolvimento de um aplicativo móvel que coleta opiniões dos alunos de uma universidade com o questionário da CPA (Comissão Própria de Avaliação), obrigatório pelo CONAES (Comissão Nacional de Avaliação da Educação Superior). O administrador gera relatórios com as informações que são coletadas e disponibiliza o questionário a partir de um ambiente paralelo na aplicação. Os alunos com o aplicativo instalado recebem notificações quando o questionário está disponível. Atualmente há uma enorme quantidade de tecnologias disponíveis no mercado, principalmente quando estamos falando dos *smartphones*, que são utilizados a todo momento para conseguir informações em tempo real. A tecnologia utilizada, chamada de híbrida, permite que um mesmo aplicativo escrito a partir de linguagens web, funcione em mais de um sistema operacional, utilizando o mesmo código, funcionalidade possível graças ao *framework* Ionic. Foi utilizado também o AngularJS, outra tecnologia extremamente poderosa para aplicações web.

Palavras-chave: Ionic. CPA. Mobile.

FRANCO, Allan Thomas Oliveira; NASCIMENTO, Flávio Francisco. **APLICAÇÃO HÍBRIDA PARA A COMISSÃO PRÓPRIA DE AVALIAÇÃO USANDO IONIC**. 2016. Monografia – Curso de SISTEMAS DE INFORMAÇÃO, Universidade do Vale do Sapucaí, Pouso Alegre – MG, 2016.

ABSTRACT

This research, which is called applied research, is directed to the development of a mobile application that collects students' opinions from a university with the questionnaire from CPA (Comissão Própria de Avaliação), required by the CONAES (Comissão Nacional de Avaliação da Educação Superior). The system administrator generates reports with the collected information and offers the questionnaire from a parallel environment in the application. Students with the application installed receive notifications when the questionnaire is available. Nowadays, there is a huge amount of technology available in the market, especially when we are talking about smartphones, which are used all the time to get real-time information. The technology used, called hybrid, allows that the same application, written from web languages, work in more than one operational system, using the same code; functionality that is possible thanks to the Ionic framework. It was also used, in this work, the AngularJS, another extremely powerful technology for web applications.

Key words: Ionic. CPA. Mobile.

SUMÁRIO

1 INTRODUÇÃO	12
2 QUADRO TEÓRICO	15
2.1 Ionic	15
2.2 Cordova	16
2.3 AngularJS 1	16
2.4 JavaScript	17
2.5 CSS	17
2.6 Bootstrap	18
2.7 HTML5	18
2.8 PHP	19
2.8.1 Slim Framework	19
2.8.2 NotORM	19
2.9 REST	20
2.10 MySQL	21
3 QUADRO METODOLÓGICO	22
3.1 Tipo de pesquisa	22
3.2 Contexto de pesquisa	22
3.3 Participantes	23
3.4 Instrumentos	24
3.5 Procedimentos e Resultados	24
3.5.1 Levantamento de Requisitos	24
3.5.2 Configuração do ambiente da API	25
3.5.3 Configuração do ambiente do aplicativo	27
3.5.4 Configuração do banco de dados	29
3.5.5 Desenvolvimento da API	31
3.5.6 Desenvolvimento do aplicativo	38
3.5.6.1 Ambiente do aluno	38
3.5.6.2 Ambiente do administrador	42
4 DISCUSSÃO DE RESULTADOS	46
5 CONSIDERAÇÕES FINAIS	55

1 INTRODUÇÃO

No mundo em que vivemos, com o desenvolvimento tecnológico e os *smartphones* em ascensão, as pessoas sempre olham seus dispositivos como recurso para resolver um problema ou como ele pode facilitar uma determinada tarefa, como uma transação bancária, uma pesquisa em um site de buscas ou simplesmente para acessar uma rede social. Esses dispositivos, que estão na mão de praticamente todos os universitários, podem ser usados como grandes aliados de uma instituição de ensino. Hoje em dia, o desenvolvimento e a utilização de *softwares* não se limita apenas aos *desktops* como era há alguns anos, cada vez mais os sistemas seguem atrelados aos dispositivos móveis, com toda a facilidade e mobilidade que eles nos proporcionam.

Diante deste cenário, o presente trabalho consiste em criar uma aplicação que possibilite a uma instituição de ensino coletar opiniões de seus alunos em relação à sua infraestrutura, professores, instalações, entre outros, seguindo o modelo disponibilizado pela CPA¹. A CPA é uma comissão de avaliação destinada à comunidade de professores, estudantes e técnicos administrativos dentro das instituições de ensino superior além de toda a sua comunidade. Segundo Sinaes (2004), a CPA tem como objetivo a identificação de deficiências dentro da instituição e aumentar a consciência pedagógica dos profissionais, assim também como prestar contas de seus serviços perante a sociedade. O foco da pesquisa e o desenvolvimento da aplicação são direcionados para alunos da UNIVÁS², com o objetivo de facilitar o acesso ao questionário da avaliação institucional por meio de um aplicativo móvel. A avaliação realizada pelos discentes que já é feita semestralmente por meio do portal do aluno, será aprimorada com o aplicativo móvel híbrido. A aplicação conta com um banco de dados próprio e informações fictícias, utilizadas somente para testes.

A tecnologia utilizada no desenvolvimento do projeto, que é chamada de híbrida, permite que um único aplicativo possua suporte para mais de uma plataforma *mobile*, como por exemplo o *Android* e o *iOS*, tudo isso a partir do *framework* Ionic. De acordo com Khanna e Harlington (2016), aplicativos híbridos são semelhantes aos aplicativos nativos, porém eles são desenvolvidos com um único código base e utilizados em mais de uma plataforma, além de possuírem comunicação com o *hardware* e também serem instalados no dispositivo. Segundo

¹ Comissão Própria de Avaliação

² Universidade do Vale do Sapucaí

IONIC (2016) o Ionic é um poderoso SDK¹ HTML5 que ajuda a construir aplicativos móveis usando tecnologias Web como HTML², CSS³ e JavaScript, que são voltados principalmente para a interface gráfica do aplicativo.

Segundo INEP⁴ (2016), a Avaliação Institucional é obrigatória para as universidades brasileiras e está relacionada diretamente com a qualidade da educação, aumento da eficiência acadêmica e social, aprofundamento de compromissos, responsabilidades sociais dentre outros itens. A presente pesquisa se direciona a uma das etapas da Avaliação Institucional, que é a autoavaliação, coordenada pela Comissão Própria de Avaliação (CPA) de cada instituição de ensino e orientada pelas normas da CONAES⁵. Segundo CONAES (2016), a Comissão Nacional de Avaliação da Educação Superior (CONAES) é o órgão de coordenação e supervisão do Sistema Nacional de Avaliação da Educação Superior (SINAES), instituído pela Lei nº 10.861, de 14 de Abril de 2004. A pesquisa que é realizada pela CPA, também abrange a sociedade, os professores e demais colaboradores da instituição, porém o trabalho tem a definição de seu escopo voltada para os alunos.

A escolha por desenvolver um aplicativo usando Ionic e seus conceitos, deve-se pela sua excelente forma de construir um aplicativo móvel multiplataforma, além das tecnologias usadas por ele que estão mais robustas e em constantes atualizações. A principal plataforma utilizada será o *Android* devido à sua grande popularidade e por ser muito utilizado.

A relevância do aplicativo para uma universidade é grande, pois facilita a pesquisa feita pela CPA, a qual muitos alunos deixam de responder por falta da praticidade. Um aplicativo destinado especialmente para este fim, com fácil utilização, intuitivo e que emite alertas aos usuários quando a pesquisa estiver disponível, pode ajudar a pesquisa a atingir seu público por mais de um meio, aumentando a quantidade de entrevistados, o que, no final, garante ainda mais dados e informações mais consistentes. O trabalho também auxilia os alunos que se interessarem por aprender como funciona o desenvolvimento de um aplicativo híbrido, além de todas as tecnologias e *frameworks* que estão envolvidos no processo, que são atuais e muito requisitadas no mercado de trabalho.

O objetivo principal da pesquisa é desenvolver um aplicativo móvel multiplataforma a partir do *framework* Ionic, que possibilite aos alunos da UNIVÁS responderem ao questionário

¹ *Software Development Kit*

² *HyperText Markup Language*

³ *Cascading Style Sheets*

⁴ Instituto Nacional de Estudos e Pesquisas Educacionais

⁵ Comissão Nacional de Avaliação da Educação Superior

semestral que é disponibilizado pela CPA com o intuito de coletar informações e opiniões referentes à universidade. A fim de atingir os objetivos da pesquisa, foi efetuado o levantamento de requisitos do sistema, a escolha das tecnologias que foram utilizadas no desenvolvimento e o planejamento do projeto a partir dos requisitos coletados, para então dar início ao desenvolvimento de um aplicativo que permita aos alunos responderem ao questionário da CPA.

Nos capítulos a seguir, serão descritas as teorias sobre as tecnologias utilizadas no desenvolvimento do projeto, as metodologias utilizadas, bem como o tipo de pesquisa, instrumentos, procedimentos e resultados, após a metodologia é realizada uma discussão com os resultados ressaltando o que se obteve com o trabalho, por fim, são apresentadas as considerações finais.

2 QUADRO TEÓRICO

Para que se possa desenvolver uma aplicação são necessárias algumas tecnologias, tais como linguagens de programação e *frameworks*. Discutiremos neste capítulo quais são eles e suas principais características.

2.1 Ionic

O Ionic é uma ferramenta de desenvolvimento de aplicativos *mobile*, que conta com a particularidade de trabalhar com uma tecnologia híbrida, o que permite com que um mesmo aplicativo possa ser compatível com mais de um sistema operacional e funcionar como se fosse escrito em uma linguagem nativa.

Segundo IONIC (2016) o *framework* foi construído por Ben Sperry, Adam Bradley e Max Lynch em Drifty, uma empresa de *software* independente e que possui diversos produtos reconhecidos no mercado. O Ionic foi criado para possibilitar que aplicativos *mobile* fossem desenvolvidos em sua base com HTML5. Mesmo com conhecimentos não muito aprofundados em tecnologias Web, é possível criar um aplicativo usando Ionic, que é um aglomerado de ferramentas que trabalham juntas para fazer do desenvolvimento Web um caminho estreito para o mundo *mobile*.

Existem três principais métodos de construção de um aplicativo móvel, são eles: aplicativos nativos, sites móveis e aplicativos híbridos, além dos recém chegados *progressive web apps*, sendo que cada um deles apresenta diversas vantagens e desvantagens. Wilken (2015) afirma que o desenvolvimento de aplicativos móveis se tornou um requisito inicial para qualquer desenvolvedor. O Ionic facilita esse desenvolvimento, pois faz com que um aplicativo móvel híbrido seja como um aplicativo nativo, através de recursos próprios que interpretam o código por partes, nas quais a interface gráfica é controlada por uma espécie de navegador isolado chamado de WebView, e também um aplicativo nativo que controla o hardware. Além do AngularJS, o Ionic possui o Cordova, que funciona carregando todo o código da aplicação Web para apresentá-lo ao usuário, além de realizar a comunicação com o dispositivo. Informações detalhadas sobre as tecnologias que foram mencionadas, serão apresentadas mais adiante no quadro teórico.

O uso de AngularJS atualmente é exigido pelo Ionic, isso a fim de potencializar seu desenvolvimento. Além de AngularJS, o Ionic também usa JavaScript e possui suporte para CSS e Bootstrap.

2.2 Cordova

Segundo Apache (2016), o Apache Cordova é um *framework* de código aberto para desenvolvimento móvel. Com ele os aplicativos são direcionados para cada plataforma, contando com ferramentas específicas para realizar o acesso ao *hardware* de cada dispositivo. Sem os recursos do Cordova, o Ionic também pode ser executado, mas seu resultado só é visível através de um navegador, pois o Cordova é necessário para que a aplicação seja executada nos dispositivos.

2.3 AngularJS 1

Segundo AngularJS (2016), o AngularJS é um *framework* para desenvolvimento de aplicações Web, criado em 2009 por Misko Hevery e Adams Abrons, porém adotado posteriormente pelo Google.

Branas (2014) diz que devido à sua forma de escrita simples e direta, reutilizável e que possibilita uma aplicação sustentável, a escrita do código com Angular ao seu final elimina uma enorme quantidade de código desnecessário, assim uma equipe de desenvolvimento mantém seu foco no trabalho, o que é realmente importante.

De acordo com Green e Seshadri (2014), com a evolução das tecnologias Web, as aplicações foram se tornando maiores e mais robustas, que acarretou um aumento da complexidade para os seus administradores. Desenvolver utilizando JavaScript/JQuery não estava gerando desempenho suficiente e a manutenção de código à longo prazo estava prejudicada. O AngularJS foi criado para atender essas necessidades que surgiram, com ele muitas tecnologias acabaram por ser dispensáveis ou pouco utilizadas.

O AngularJS utiliza como modelo de arquitetura o padrão MVC, que divide o sistema em três partes distintas e modulares: o modelo (model), a visão (view) e o controlador (controller). Cada uma dessas partes separa o código em grupos, o que proporciona vantagens

como o dimensionamento e a organização de tarefas, na qual cada uma faz somente o que é designado, além disso o código se torna reutilizável e de fácil manutenção. Além do MVC, o AngularJS também utiliza os padrões MVVM, MVP e MVW.

2.4 JavaScript

Segundo Balduino (2012), o JavaScript teve sua primeira versão desenvolvida em 1995, para o *browser* Mozilla, e o seu próprio nome, JavaScript, foi uma jogada de marketing, visando aproveitar o sucesso da linguagem Java. Apesar de levar parte do nome, o JavaScript não é uma versão simplificada do Java. O JavaScript é comumente utilizado no lado do cliente (*client-side*) realizando diversas funções, dentre elas, o trabalho de intermediador para com o servidor. Atualmente também existem ferramentas que possibilitam o uso do JavaScript no lado do servidor.

De acordo com Flanagan (2006), o JavaScript é interpretado ao invés de compilado, por isso muitas vezes é considerado uma linguagem de *script* ao invés de linguagem de programação.

2.5 CSS

Segundo Silva (2011), CSS é a abreviação para o termo em inglês *Cascading Style Sheet*, traduzido para o português como folhas de estilo em cascata. A linguagem CSS tem por finalidade estilizar uma estrutura HTML para a apresentação de elementos. Como por exemplo: cores de fontes, tamanhos de texto, bordas arredondadas, entre outros. De acordo com Lie e Bos (2005), os primeiros vestígios de CSS são do ano de 1994, quando surgiu a necessidade de melhorar a aparência de uma página Web tendo como layout um jornal.

Sendo assim, entendemos que o CSS faz todo o trabalho de estilizar e deixar páginas Web com uma interface amigável e melhor apresentada do que somente quando se tem HTML puro.

2.6 Bootstrap

Bootstrap é um *framework* destinado ao desenvolvimento *front-end* de projetos responsivos e focado no desenvolvimento para dispositivos móveis. Conforme Bootstrap (2016), sua origem foi por volta de 2010 por um desenvolvedor no Twitter e antes de ser uma estrutura de código aberto, era conhecido como Twitter Blueprint, foi lançado em 19 de agosto de 2011, passando por vários lançamentos estando hoje na versão 3. Sua alta popularidade está por sua simplicidade na implementação e produção.

2.7 HTML5

Segundo Eis e Ferreira (2012), o HTML é um dos três pilares da Web, uma linguagem de marcação para publicação de informações (texto, audio, imagem etc) para Web. Seu conceito de Hipertexto é uma forma de organizar o conteúdo de maneira não linear, as marcações que referenciam os elementos (tags), também os identificam para o CSS e o JavaScript tratá-los.

Ainda segundo Eis e Ferreira (2012), o HTML que usamos hoje em dia, na versão 5, é uma evolução da versão criada por Tim Berners-Lee. O HTML5 chegou ao seu estágio final a partir de um grupo chamado WHATWG, formado por desenvolvedores de empresas como Apple, Mozilla e Opera, que estavam insatisfeitos com o caminho que a Web estava tomando. A versão 5 trouxe para o HTML uma série de melhorias que o tornou muito mais potente, como a manipulação de características de elementos de forma que a aplicação continue funcional e leve, descartando a necessidade de scripts enormes. A validação de formulários é outro recurso que foi implementado no HTML5, que reduz drasticamente o trabalho no tratamento de informações que são enviadas para o servidor.

No HTML5 a semântica se faz mais presente, com menor quantidade de códigos, pouca necessidade de instalação de *plugins* e consequentemente ganho de performance.

2.8 PHP

O PHP¹ é uma linguagem *open source*, que em tradução livre, significa código aberto, ou seja, qualquer um pode contribuir para seus aprimoramentos. De acordo com PHP (2016), o PHP é executado no lado do servidor, o que faz com que o código seja processado por um sistema dedicado especialmente para essa função, e não na máquina do usuário. Desta forma, o navegador recebe apenas os resultados do processamento, o que é indispensável quando se é necessário ter sigilo e segurança com as informações.

Segundo Niederauer (2004), o PHP é uma das linguagens de programação mais conhecidas e utilizadas mundialmente. Sua função é, principalmente, trazer interações e conteúdos dinâmicos para páginas Web, em que apenas o HTML não é suficiente. Em um site de notícias, por exemplo, o PHP tem a função de recuperar informações de um banco de dados e transmiti-las para o *front-end* de uma aplicação ou *website*, que apresentará essas informações para o usuário.

2.8.1 Slim Framework

Para que seja possível a comunicação entre uma aplicação móvel e um servidor de banco de dados através do PHP, é necessário um serviço Web realizando esse intermédio. Para essa função será utilizado o Slim Framework, que segundo Slim (2016), é uma ferramenta que possibilita a criação de APIs² REST³ de maneira sucinta e poderosa através do protocolo HTTP⁴.

2.8.2 NotORM

O NotORM é uma biblioteca do PHP que trabalha de maneira eficiente com os dados de um banco de dados, realiza de maneira simples e rápida consultas SQL em uma aplicação,

¹ *Hypertext Preprocessor*

² *Application Programming Interface*

³ *Representational State Transfer*

⁴ *Hypertext Transfer Protocol*

funcionando a partir do PHP 5.1. De acordo com Shameer C (2012), o NotORM ajuda na criação de consultas SQL sem se preocupar muito com a escrita, evitando possíveis erros de sintaxe e consultas complexas com códigos extensos.

2.9 REST

Segundo Saudate (2014), o REST, criado por Roy Fielding, é uma forma de arquitetura baseada em redes de computadores que fornece uma interface simples para disponibilizar serviços. Quando amplamente explorado para o desenvolvimento de *web services* e seus princípios são todos utilizados, um sistema REST é chamado de RESTful.

A URI¹ deve ser única para cada recurso que se deseja ter, pois é ela quem define o procedimento que será realizado e qual será a interação com o lado do cliente. Em conjunto com a URI, é necessário trabalhar com os métodos HTTP, pois são eles quem a API interpreta para saber o que deverá ser feito. Na versão corrente do HTTP, estão disponíveis oito principais métodos, sendo eles:

- GET
- POST
- PUT
- DELETE
- OPTIONS
- HEAD
- TRACE
- CONNECT
- PATCH

REST trabalha com dois principais formatos de envio e retorno de informações, que são o XML² e o JSON³. Ambos os formatos possuem a mesma proposta, porém, com algumas diferenças, principalmente na implementação. Neste projeto será usado o formato JSON, por

¹ Uniform Resource Identifier

² eXtensible Markup Language

³ JavaScript Object Notation

conta de sua menor complexidade, tamanho reduzido quando comparado com o XML, e também por ser muito usado atualmente.

De acordo com Webber, Parastatidis e Robinson (2010), a Web atual é, em sua maioria, desenvolvida para pessoas, e a necessidade de várias formas de acesso ou mais de uma plataforma para um único recurso ou aplicativo, faz a exigência da construção de um sistema distribuído. A facilidade para trabalhar com sistemas distribuídos é a grande vantagem de uso das REST API's, pois elas possibilitam a comunicação de dados entre domínios distintos, justamente a necessidade de uma aplicação *mobile*, diferente do que acontece nos websites por exemplo, em que ambos estão na mesma rede e se comunicam diretamente, sem a necessidade de um Web service realizando um intermédio.

2.10 MySQL

Segundo Jobstraibizer (2010), o MySQL foi criado na Suécia por três desenvolvedores, Allan Larsson, David Amark e Michael Widenius, o sistema foi posteriormente adquirido pela Sun Microsystems, empresa que hoje pertence à Oracle.

Niederauer (2005) diz que o MySQL é um SGBD¹ relacional que utiliza como padrão a linguagem SQL², largamente utilizado para websites e aplicações Web, principalmente quando se necessita de velocidade ao acessar um banco de dados muito grande. Sua alta popularidade está também relacionada à disponibilidade para diferentes sistemas operacionais como o Linux, Mac e Windows. Entre suas vantagens estão o desempenho, a escalabilidade e a confiabilidade, além de ser compatível com diversas linguagens de programação como o PHP, Java, Python e C++.

¹ Sistema de Gerenciamento de Bancos de Dados

² *Structured Query Language*

3 QUADRO METODOLÓGICO

Nesse capítulo serão apresentados os métodos adotados para o desenvolvimento da pesquisa, tais como tipo de pesquisa, participantes e principalmente os procedimentos realizados e os resultados obtidos durante o desenvolvimento do projeto.

3.1 Tipo de pesquisa

Existem diversos tipos de pesquisas, e dentre todos os tipos, a que melhor se enquadra no projeto é a pesquisa aplicada. Isso porque ela parte do estudo teórico de uma tecnologia específica até o desenvolvimento de uma aplicação para coletar dados dos alunos da universidade.

De acordo com Barros e Lehfeld (2000), a pesquisa aplicada visa a solução de um problema encontrado na realidade, produzindo conhecimentos para uma melhor solução. Podemos enxergar como problema, a quantidade de alunos que não respondem ao questionário, o que prejudica no resultado final da pesquisa, que coletará ao seu término, uma menor quantidade de informações.

3.2 Contexto de pesquisa

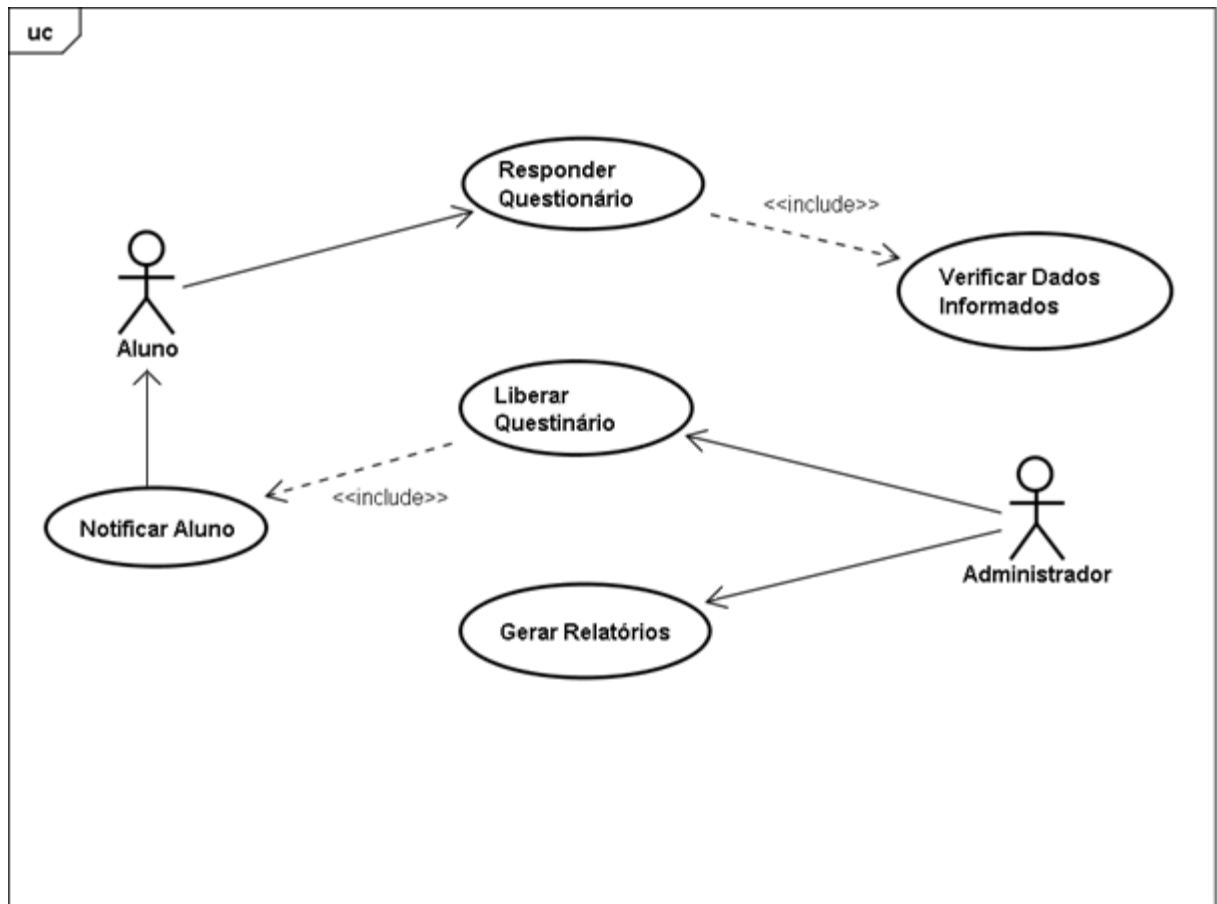
Essa pesquisa beneficia qualquer instituição de ensino que possua um método de pesquisa de satisfação de seus alunos em relação à instituição como um todo. O objetivo é criar uma aplicação para dispositivos móveis utilizando tecnologia híbrida, o que permite com que o mesmo aplicativo seja utilizado em diferentes sistemas, sem a necessidade de uma reescrita para outras linguagens. Tendo dentro deste contexto o módulo da aplicação que será usado apenas pelos alunos, e tendo como campo de pesquisa focada para alunos da UNIVÁS.

O aluno, portador de um *smartphone* instala o aplicativo da CPA, e a partir daí ele passa a receber notificações assim que a pesquisa estiver disponível, caso esteja, o aluno preenche o questionário, assim como é feito através do portal da universidade.

3.3 Participantes

O diagrama de casos de uso descreve uma proposta para o sistema que será projetado, além de ser uma ferramenta para levantamento de requisitos. Segue abaixo na Figura 1 o diagrama representando os participantes do sistema e suas principais atividades.

Figura 1 - Diagrama de casos de uso do projeto.



Fonte: Elaborado pelos autores.

No diagrama acima estão apresentadas as duas principais funcionalidades do aplicativo, que são: a possibilidade dos alunos da universidade de responderem ao questionário da CPA e os recursos para o administrador, de liberar o questionário para os alunos e gerar relatórios a partir dos dados que foram coletados.

3.4 Instrumentos

O material foi coletado através de pesquisas em sites, artigos, vídeos da internet, e-mails e reuniões com a comissão da CPA da UNIVÁS.

Foram realizadas pesquisas em sites oficiais sobre as tecnologias e através do SINAES (órgão que avalia as instituições em três componentes principais que são avaliação das instituições, dos cursos e desempenho dos alunos), buscando a melhor maneira de trabalhar com cada tecnologia. Os artigos servirão como parte das pesquisas em documentações coerentes com o tema. Foram também utilizados livros de autores estrangeiros, por conta das poucas informações relevantes existentes em português, por se tratarem de tecnologias novas.

O intuito das reuniões e e-mails foi de coletar dados relevantes e apresentar propostas sobre o projeto, de tal forma que os integrantes definam o andamento do projeto e troquem informações do que pode ser melhorado para realização de um trabalho de qualidade.

3.5 Procedimentos e Resultados

Após os estudos das ferramentas e teorias de desenvolvimento de software e integração entre serviços e uma aplicação Ionic, iniciou-se o período do desenvolvimento do sistema.

3.5.1 Levantamento de Requisitos

A execução do projeto requer seguir alguns procedimentos, para melhor aproveitamento do tempo e desenvolvimento com a maior qualidade possível, o projeto seguiu alguns procedimentos pré-determinados pela equipe, que são:

- Levantamento dos requisitos: Foram coletadas informações sobre o funcionamento da CPA que já é utilizada através do portal da Universidade e sobre os participantes e suas funções dentro do sistema.

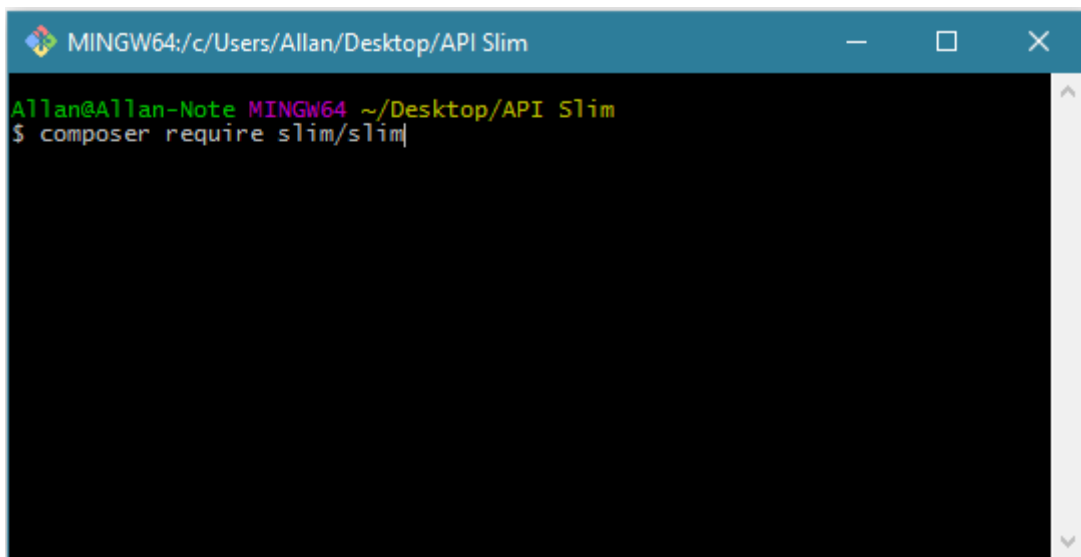
- Criação de protótipos para melhor compreensão das ferramentas e recursos: As telas foram pré projetadas em programas de prototipação e as tabelas do banco de dados testadas manualmente até a integração do front-end com o back-end.
- Criação da aplicação: Desenvolvimento do aplicativo que permita aos alunos responderem o questionário da CPA, utilizando o Ionic e seu método de desenvolvimento híbrido.
- Testes de funcionamento e qualidade da aplicação.

3.5.2 Configuração do ambiente da API

Para a construção da API REST foi necessária a instalação e configuração de um ambiente de desenvolvimento, compatível com as tecnologias utilizadas e em comum acordo com as necessidades da aplicação.

Para que houvesse uma facilidade maior na configuração do ambiente durante a execução de comandos, foi utilizado o *Git*, que foi instalado atrás do *download* em seu site oficial. O *Git* também foi utilizado como controlador de versão do projeto, através do *GitHub*, ferramenta que facilitou muito do trabalho da equipe durante o desenvolvimento, para organização e *backup* dos códigos da aplicação.

Após a instalação do *Git*, foi feito o *download* do Composer, em seu site oficial, que é uma ferramenta de gerenciamento de dependências em PHP, após baixado o instalador do composer, o mesmo solicita durante sua instalação, o diretório que foi instalado o PHP no computador do usuário, que deve ser selecionado para que a instalação prossiga. Depois de instalado o *Composer*, foi executado o comando conforme a Figura 2, para realizar a instalação do *Slim Framework*.

Figura 2 - Instalação do Slim Framework.

Fonte: Elaborado pelos autores.

Após a instalação do *Slim Framework*, foi criado no seu diretório raiz, o arquivo `index.php`, com o código de exemplo conforme a Listagem 1.

Listagem 1 - Configuração inicial do `index.php`.

```
<?php
use \Psr\Http\Message\ServerRequestInterface as Request;
use \Psr\Http\Message\ResponseInterface as Response;

require 'vendor/autoload.php';

$app = new \Slim\App;
$app->get('/hello/{name}', function (Request $request, Response $response) {
    $name = $request->getAttribute('name');
    $response->getBody()->write("Hello, $name");

    return $response;
});
$app->run();
```

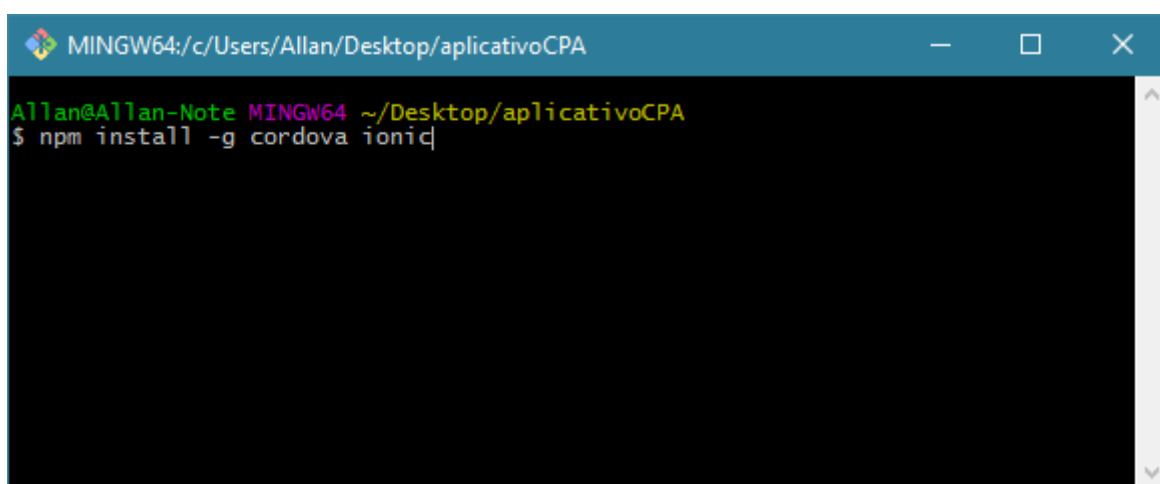
Fonte: Elaborado pelos autores.

O arquivo `index.php` é necessário para fazer a interpretação das requisições da API através dos métodos HTTP por um endereço específico, a partir dele são invocadas as classes e métodos específicos para cada requisição.

3.5.3 Configuração do ambiente do aplicativo

Para dar início na configuração do ambiente da aplicação, foi instalado o *NodeJS*, através do *download* do arquivo executável pelo seu site oficial. Segundo Pereira (2014), o *NodeJS* é uma tecnologia que conta com um modelo inovador em sua arquitetura, que possui uma excelente performance, principalmente em sistemas que necessitam de muito processamento. Assim como o *Maven* no Java, o *NodeJS* possui seu próprio gerenciador de pacotes, o NPM¹, que mesmo utilizando outra linguagem, ele é útil para executarmos procedimentos de configuração e instalações no ambiente de desenvolvimento. Após instalado o *NodeJS*, foi executado o comando conforme mostra a Figura 3, para a instalação da última versão do Cordova juntamente com o Ionic.

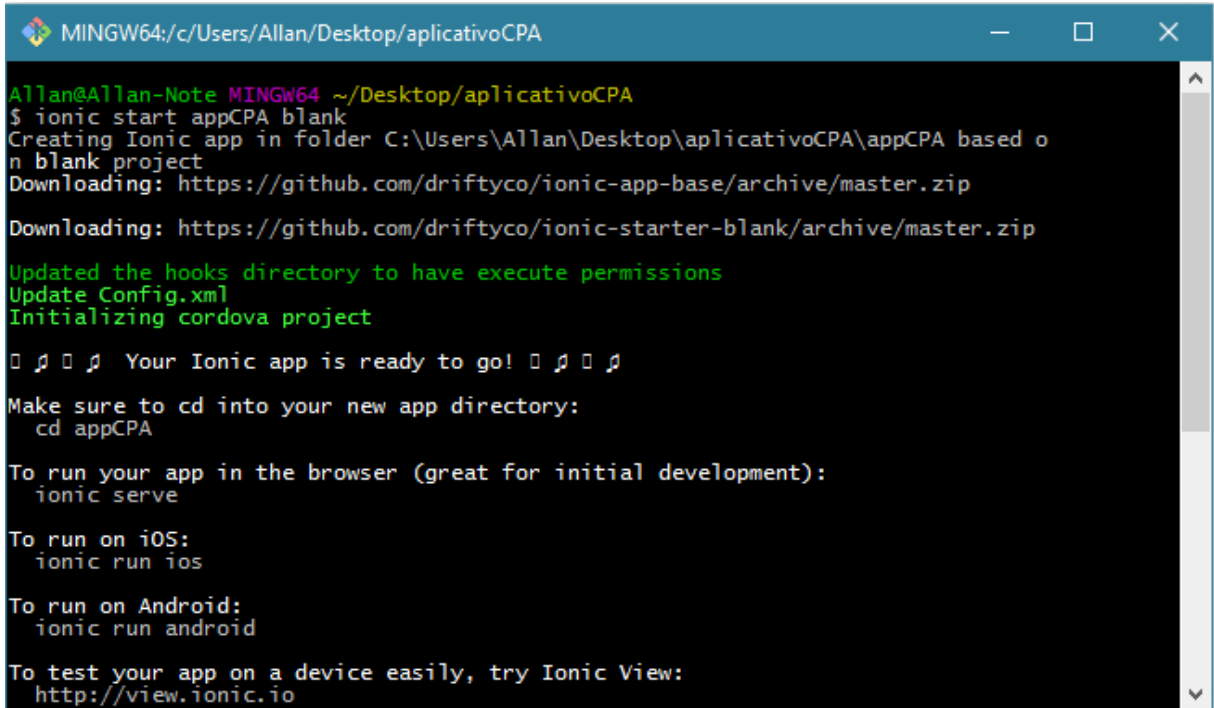
Figura 3 - Instalação do Cordova e do Ionic.



Fonte: Elaborado pelos autores.

Após a instalação do Cordova e do Ionic, foi necessário criar um projeto inicial em branco com o Ionic, através do comando `ionic start appCPA blank`, como pode ser visto na Figura 4.

¹ Node Package Manager

Figura 4 - Criação de um novo projeto Ionic.


```

MINGW64/c:/Users/Allan/Desktop/aplicativoCPA
Allan@Allan-Note MINGW64 ~/Desktop/aplicativoCPA
$ ionic start appCPA blank
Creating Ionic app in folder C:\Users\Allan\Desktop\aplicativoCPA\appCPA based o
n blank project
Downloading: https://github.com/driftyco/ionic-app-base/archive/master.zip
Downloading: https://github.com/driftyco/ionic-starter-blank/archive/master.zip
Updated the hooks directory to have execute permissions
Update Config.xml
Initializing cordova project

 ionic  Your Ionic app is ready to go!

Make sure to cd into your new app directory:
  cd appCPA

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

To run on iOS:
  ionic run ios

To run on Android:
  ionic run android

To test your app on a device easily, try Ionic View:
http://view.ionic.io

```

Fonte: Elaborado pelos autores.

Depois de iniciado um projeto em branco do Ionic, um diretório novo foi criado com o mesmo nome atribuído para o aplicativo. O diretório raiz contém as configurações e arquivos *default* do Ionic, além da pasta *www*, que é onde ficará todo o *front-end* da aplicação. O diretório principal do Ionic pode ser visto na Figura 5.

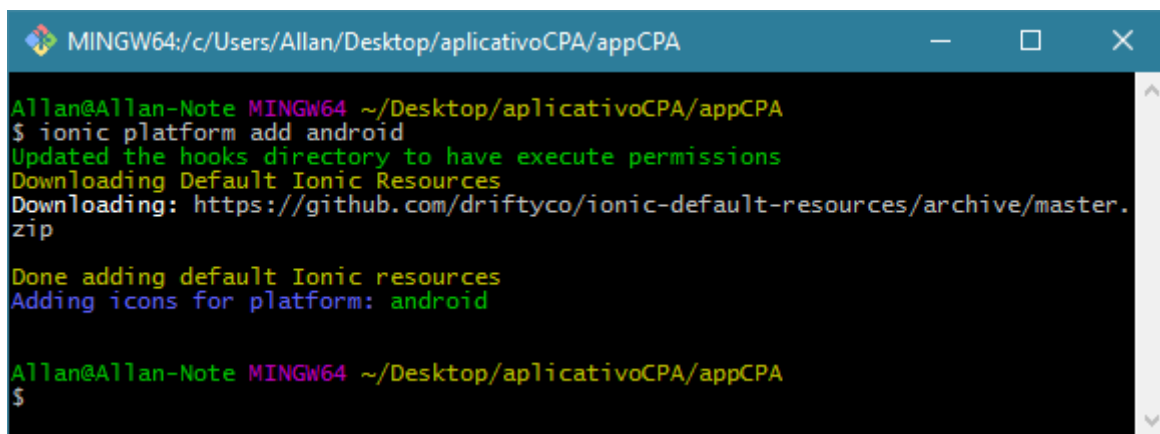
Figura 5 - Diretório raiz do projeto Ionic.

Nome	Data de modificaç...	Tipo	Tamanho
hooks	25/08/2016 09:42	Pasta de arquivos	
scss	25/08/2016 09:42	Pasta de arquivos	
www	25/08/2016 09:42	Pasta de arquivos	
	25/08/2016 09:42	Bower RC Source ...	1 KB
	25/08/2016 09:42	Editor Config Sour...	1 KB
	25/08/2016 09:42	Documento de Te...	1 KB
bower	25/08/2016 09:42	Arquivo JSON	1 KB
config	25/08/2016 09:42	Documento XML	1 KB
gulpfile	25/08/2016 09:42	Arquivo JavaScript	2 KB
ionic.project	25/08/2016 09:42	Arquivo PROJECT	1 KB
package	25/08/2016 09:42	Arquivo JSON	1 KB

Fonte: Elaborado pelos autores.

Para que fosse possível criar um arquivo final de instalação para dispositivos Android no formato `.apk`, também foi necessário definir a plataforma para compilar o projeto, executando um comando dentro do diretório principal, conforme mostra a Figura 6.

Figura 6 - Instalação da plataforma Android no projeto Ionic.



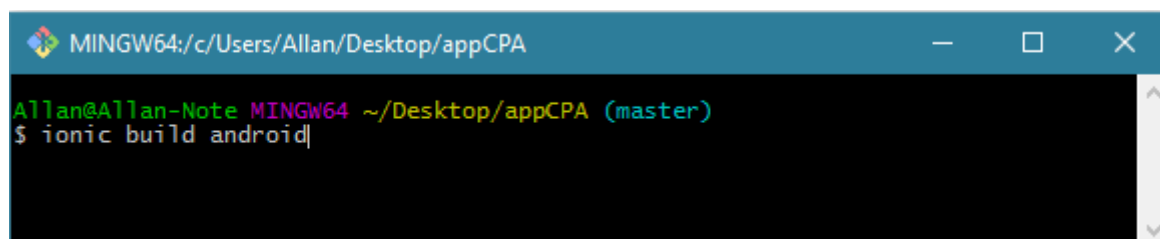
```
MINGW64:/c:/Users/Allan/Desktop/aplicativoCPA/appCPA
Allan@Allan-Note MINGW64 ~/Desktop/aplicativoCPA/appCPA
$ ionic platform add android
Updated the hooks directory to have execute permissions
Downloading Default Ionic Resources
Downloading: https://github.com/driftyco/ionic-default-resources/archive/master.zip
Done adding default Ionic resources
Adding icons for platform: android

Allan@Allan-Note MINGW64 ~/Desktop/aplicativoCPA/appCPA
$
```

Fonte: Elaborado pelos autores.

O processo de compilação da aplicação foi feito através do comando que pode ser visto na Figura 7.

Figura 7 - Compilação do aplicativo.



```
MINGW64:/c:/Users/Allan/Desktop/appCPA
Allan@Allan-Note MINGW64 ~/Desktop/appCPA (master)
$ ionic build android
```

Fonte: Elaborado pelos autores.

Após ter sido compilado com sucesso, o arquivo final para a instalação do aplicativo fica disponível no diretório `\appCPA\platforms\android\build\outputs\apk` nomeado automaticamente para `android-debug.apk`.

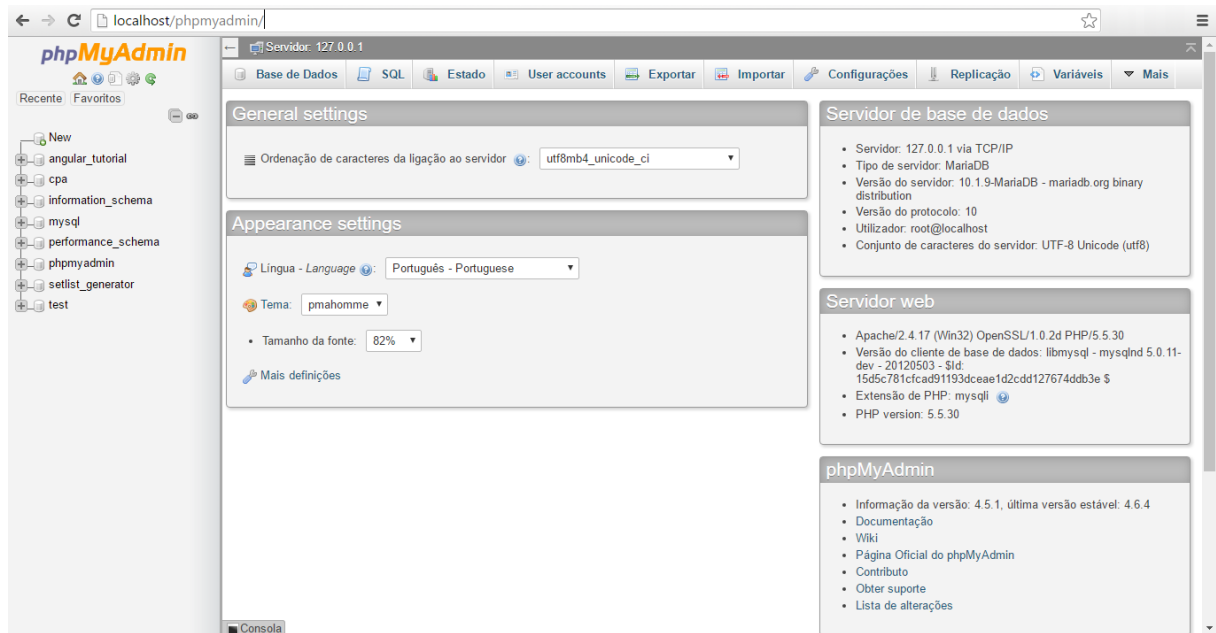
3.5.4 Configuração do banco de dados

Para facilitar o manejo das informações do banco de dados, além da criação das tabelas e suas relações, foi utilizada a ferramenta *phpMyAdmin*, que é acessada por um navegador

disponível no servidor pelo endereço `http://localhost/phpmyadmin`. O *phpMyAdmin* é uma ferramenta de gerenciamento de bancos de dados MySQL, que foi extremamente útil e eficaz para o desenvolvimento da aplicação.

A Figura 8 mostra a interface principal do *phpMyAdmin*.

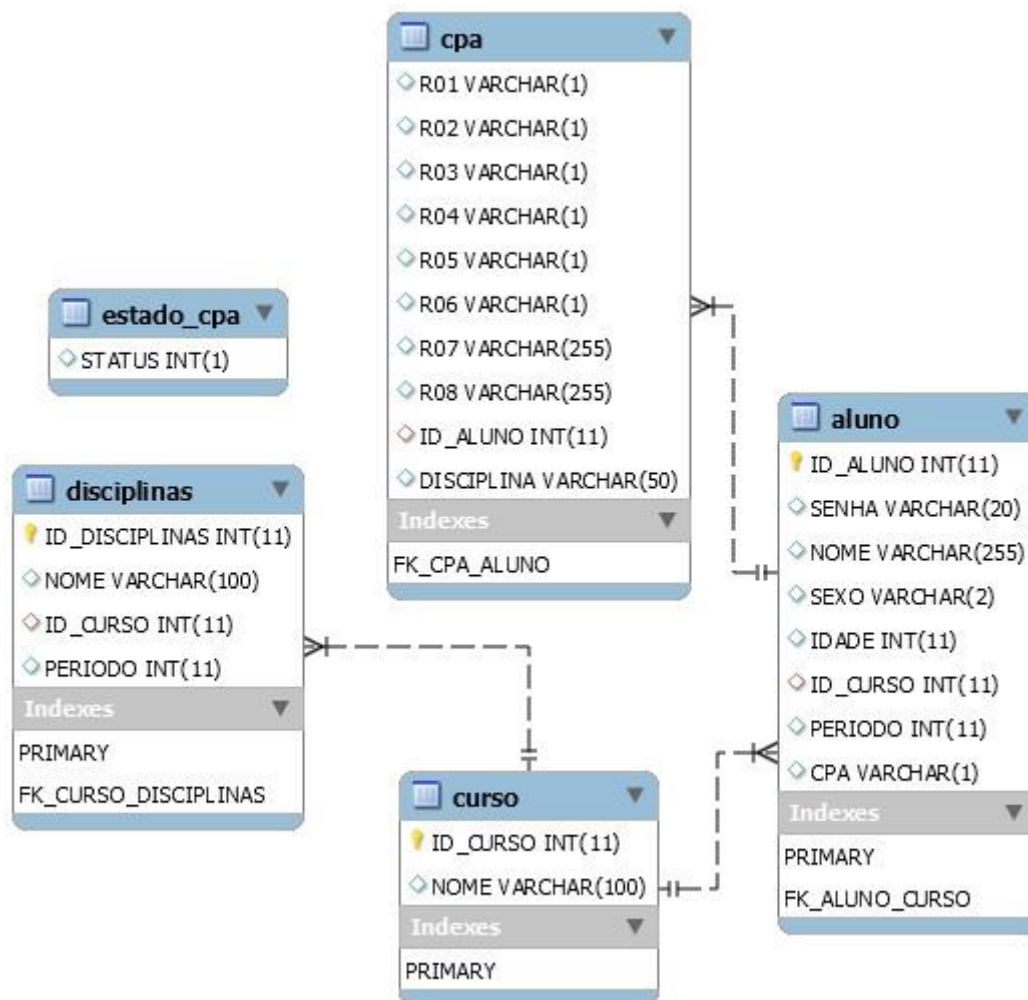
Figura 8 - Interface do phpMyAdmin.



Fonte: Elaborado pelos autores.

O *phpMyAdmin* foi utilizado também para a criação das tabelas do sistema e a definição de seus respectivos relacionamentos. O diagrama que representa os relacionamentos entre as tabelas está representado na Figura 9.

Figura 9 - Diagrama do banco de dados.



Fonte: Elaborado pelos autores.

Para o desenvolvimento da aplicação, testes de funcionamento e manejo de dados, as tabelas do sistema e seus relacionamentos foram criados isoladamente, sem qualquer ligação com o banco de dados da universidade, além disso dados utilizados para os testes de funcionamento foram fictícios.

3.5.5 Desenvolvimento da API

Anteriormente foi mostrada a criação do ambiente de desenvolvimento da API REST, a partir de realizado esse procedimento, iniciou-se o desenvolvimento do *back-end*, que funciona no lado do servidor conectado ao banco de dados e provê informações para a aplicação através do protocolo HTTP.

Com o banco de dados já configurado, o primeiro passo foi criar a classe `ConnectionFactory.php`, essa classe faz a comunicação de toda a API REST com o banco de dados da aplicação, não sendo necessário reescrever o mesmo código em cada nova interação com o banco.

O código da classe pode ser visto na Listagem 2.

Listagem 2 - Classe `ConnectionFactory`.

```
<?php
header('Access-Control-Allow-Origin: *');

class ConnectionFactory {
    public static function getDB() {
        $connection = self::getConnection();
        $db = new NotORM($connection);
        return $db;
    }
    public static function getConnection() {
        $base_dados = 'cpa';
        $usuario_bd = 'root';
        $senha_bd = '';
        $host_db = 'localhost';
        $charset_db = 'utf8';
        $conexao_pdo = null;
        $detalhes_pdo = 'mysql:host=' . $host_db;
        $detalhes_pdo .= ';dbname=' . $base_dados;
        $detalhes_pdo .= ';charset=' . $charset_db;

        try {
            $conexao_pdo = new PDO($detalhes_pdo, $usuario_bd, $senha_bd);
        } catch (PDOException $e) {
            print "Erro: " . $e->getMessage() . "<br/>";
            die();
        }

        return $conexao_pdo;
    }
}
?>
```

Fonte: Elaborado pelos autores.

Com a classe `ConnectionFactory.php` devidamente criada, foram realizados testes de comunicação com o banco de dados, e a partir daí foi escrita a classe `CPAService.php`, que provê as principais funções da aplicação, como as validações de senha do usuário verificações sobre o *status* da CPA entre outros, seu código pode ser visto na Listagem 3.

Listagem 3 - Classe CPAService.

```

class CPAService {
    public static function validaLogin($usuario, $senha) {
        $db = ConnectionFactory::getDB();
        $result = array();
        $aluno = $db->ALUNO()->select("ID_ALUNO, NOME, CPA, PERIODO")
            ->where("ID_ALUNO = $usuario AND SENHA = $senha");
        if($data = $aluno->fetch()){
            echo json_encode(
                array(
                    'ID_ALUNO' => $data['ID_ALUNO'],
                    'NOME' => $data['NOME'],
                    'CPA' => $data['CPA'],
                    'PERIODO' => $data['PERIODO']
                )
            );
        } else {
            return 0;
        }
    }

    public static function salvarQuestionario($JSONRespostas) {
        $db = ConnectionFactory::getDB();

        //Insere as respostas na tabela CPA, uma linha para cada disciplina.
        for ($index = 0; $index < sizeof($JSONRespostas); $index++) {
            $application = $db->CPA()->insert($JSONRespostas[$index]);
        }
        //Adiciona ao registro do usuário que ele já respondeu.
        $pdo = ConnectionFactory::getConnection();
        $aluno = $JSONRespostas[0];
        $id_aluno = $aluno['id_aluno'];
        $atualiza = $pdo->prepare("UPDATE ALUNO SET CPA = 1 WHERE ID_ALUNO =
        '$id_aluno'");
        $atualiza->execute();
        return "OK";
    }

    public static function alteraStatusCPA($status) {
        $pdo = ConnectionFactory::getConnection();
        if ($status == 1) {
            $pdo->prepare("UPDATE ESTADO_CPA SET STATUS = 0")->execute();
            return "A CPA foi fechada!";
        } else {
            $pdo->prepare("UPDATE ESTADO_CPA SET STATUS = 1")->execute();
            $pdo->prepare("UPDATE ALUNO SET CPA = 0")->execute();
            return "A CPA foi aberta!";
        }
    }
}

```

Fonte: Elaborado pelos autores.

A classe apresentada anteriormente, realiza os procedimentos que se referem principalmente ao caso de uso “Aluno” e também do “Administrador”, com exceção dos relatórios, que devido à sua complexidade, ganhou uma classe especialmente para este propósito. As funções da classe CPAService(Serviço CPA) foram divididas uma para cada finalidade específica, que são respectivamente na ordem do código:

- Validar os dados que foram informados durante o login no aplicativo;
- Buscar as disciplinas cursadas por cada aluno, com base no período atual e no curso;
- Salvar o questionário já preenchido e validado pelo *front-end* no banco de dados;
- Verificar o *status* atual da CPA(verifica se o questionário está ou não disponível para ser preenchido);
- Alteração do status da CPA(faz a liberação do questionário para que o mesmo possa ser acessado e respondido pelos alunos.

Paralelamente ao desenvolvimento da classe apresentada acima, foi escrita a página `index.php`, já mencionada anteriormente, que faz a interpretação de cada requisição HTML e toma a decisão pertinente a cada método, a partir daí, invocando uma função específica dos serviços da API.

Na Listagem 4, são mostrados os métodos que são invocados na API REST a partir das requisições HTML.

Listagem 4 - index.php.

```

header('Access-Control-Allow-Origin: *');
require 'vendor/autoload.php';
require 'database/ConnectionFactory.php';
require 'database/CPAService.php';
require 'database/ReportService.php';
require 'database/notorm/NotORM.php';
$app = new \Slim\Slim();
// Fazer Login
$app->post('/fazerLogin/', function() use ( $app ) {
    $login = $app->request()->getBody();
    $login = json_decode($login, true);
    $return = CPAService::validaLogin($login['usuario'], $login['senha']);
    echo $return;
});
$app->post('/fazerLoginAdm/', function() use ( $app ) {
    $login = $app->request()->getBody();
    $login = json_decode($login, true);
    if ($login['usuario']=="admin" && $login['senha']=="admin" )
        echo 1;
    else
        echo 0;
});
// Busca Disciplinas
$app->post('/buscaDisciplinas/', function() use ( $app ) {
    $ID_ALUNO = $app->request()->getBody();
    $ID_ALUNO = json_decode($ID_ALUNO, true);
    $return = CPAService::buscaDisciplinas($ID_ALUNO);
    echo json_encode($return);
});
// Salvar Respostas no Banco de Dados
$app->post('/salvarRespostas/', function() use ( $app ) {
    $JSONRespostas = $app->request()->getBody();
    $JSONRespostas = json_decode($JSONRespostas, true);
    echo CPAService::salvarQuestionario($JSONRespostas);
});
// Verificar se a CPA está aberta / alteração de Status
$app->get('/verificaCPA/', function() use ( $app ) {
    echo CPAService::verificaCPA();
});
// Modifica o status da CPA
$app->get('/alteraStatusCPA/', function() use ( $app ) {
    echo CPAService::alteraStatusCPA(CPAService::verificaCPA());
});
// Gerar Relatórios
$app->post('/gerarRelatorio/', function() use ( $app ) {
    $relatorio = $app->request()->getBody();
    $relatorio = json_decode($relatorio, true);
    echo ReportService::gerarRelatorio($relatorio);
});

```

Fonte: Elaborado pelos autores.

Nas primeiras linhas do arquivo `index.php` foram incluídas as classes que contêm os métodos específicos para cada situação, e que são invocados nas requisições da API. Abaixo delas estão todos os recursos disponibilizados pelo *web service*, como as validações de senhas e usuários, busca de informações referentes ao questionário, verificações sobre o *status* da CPA entre outros.

Após a criação do arquivo que trata as requisições da API(`index.php`) e com os serviços referentes ao questionário funcionando corretamente e devidamente testados, foi implementado o serviço de relatórios(`ReportService.php`), desenvolvido para gerar os relatórios pertinentes à CPA baseados nas informações coletadas com o questionário, a partir de pontos específicos como: quantidade de alunos que responderam ao questionário e sua diferenciação por sexo, qualidade das respostas em relação à universidade, além de um relatório principal, contendo todas as informações e percentuais de resposta com base nas questões apresentadas e as alternativas disponíveis. Na Listagem 5 é mostrado um resumo da classe `ReportService`.

Listagem 5 - `ReportService`, classe responsável pelos relatórios.

```
class ReportService {
    public static function gerarRelatorio($relatorio) {
        switch ($relatorio['tipo']) {
            case 'G_Universidade' :
                return ReportService::G_Universidade($relatorio['dataInicial'],
                    $relatorio['dataFinal']);
        }
    }








    private static function G_Universidade($dataInicial, $dataFinal) {
        $pdo = ConnectionFactory::getConnection();

        $consulta = $pdo->prepare("...
        $consulta->execute();
        $result = $consulta->fetchAll(PDO::FETCH_ASSOC);
        $A = 0; $B = 0; $C = 0;
        foreach($result as $linha) {
            $A = $A + $linha['A'];
            $B = $B + $linha['B'];
            $C = $C + $linha['C'];
        }
        return json_encode(array('A'=> $A, 'B'=> $B, 'C'=> $C));
    }
}
?>
```

Fonte: Elaborado pelos autores.

O diretório da API, ao término do seu desenvolvimento é apresentado na Figura 10.

Figura 10 - Diretório raiz da API.

Nome	Data de modificaç...	Tipo	Tamanho
 .git	06/09/2016 16:18	Pasta de arquivos	
 database	12/09/2016 16:13	Pasta de arquivos	
 vendor	09/08/2016 16:10	Pasta de arquivos	
 .htaccess	17/02/2016 11:11	Arquivo HTACCESS	1 KB
 composer	22/03/2016 15:16	Arquivo JSON	1 KB
 composer.lock	22/03/2016 15:18	Arquivo LOCK	3 KB
 index	12/09/2016 13:42	Arquivo PHP	2 KB

Fonte: Elaborado pelos autores.

As classes de serviços e a fábrica de conexões com o banco de dados ficaram localizados dentro da pasta database. O arquivo *.htaccess* tem o objetivo de encurtar a URL ¹ para o acesso à API e facilitar seu acesso, não sendo necessário especificar o arquivo *index.php* para requisições com a API. Com a API REST pronta, os serviços criados e disponíveis para acesso, iniciou-se o desenvolvimento do aplicativo, tema do próximo capítulo.

3.5.6 Desenvolvimento do aplicativo

A partir daqui serão apresentados os procedimentos necessários para a criação do aplicativo, separados pelo ambiente utilizado pelo aluno, e o utilizado pelo administrador respectivamente.

3.5.6.1 Ambiente do aluno

Com o ambiente de desenvolvimento pronto e organizado, a aplicação começou a ser desenvolvida. O ponto de partida inicial foi a construção de uma página de *login*, *index.html*, para que o aluno pudesse se logar no sistema e responder ao questionário. Ao fazer a validação do usuário e da senha fornecidos pelo aluno, é realizada também a verificação se a CPA está disponível e se o aluno não respondeu ao questionário. Caso esteja tudo correto,

¹ *Universal Resource Locator*

o aluno é redirecionado para a página `questionário.html`, onde as questões são apresentadas dinamicamente para todas as disciplinas cursadas em seu período atual. Se a CPA não estiver disponível ou se o aluno já respondeu ao questionário, uma mensagem é apresentada na tela de *login* informando o usuário do que houve.

Durante a verificação dos dados informados pelo aluno, também são retornados pela API informações à respeito do aluno, como curso e o período, que serão armazenados localmente em uma *session* para que possam ser utilizados futuramente durante o preenchimento do questionário. A lógica de funcionamento das operações citadas podem ser vistas na Listagem 6, que representa o arquivo `index.js`.

Listagem 6 - Operação `fazerLogin` do arquivo `index.js`.

```
var app = angular.module("CPA", []);

app.controller("CPAController", function($scope, $http) {
    sessionStorage.clear();

    $scope.fazerLogin = function(login) {
        $http.get("http://" + host +
            "/api/verificaCPA/").success(function(data) {
            if (data == 1) {
                $http({
                    method: "post",
                    url: "http://" + host + "/api/fazerLogin/",
                    headers: {'Content-Type': 'application/x-www-form-
                        urlencoded'},
                    data: login
                }).success(function(data) {
                    if (data == 0) {
                        $scope.avisos = "Usuário ou Senha Incorretos!";
                    } else {
                        sessionStorage.setItem("ID_ALUNO",
                            data['ID_ALUNO']);
                        sessionStorage.setItem("NOME", data['NOME']);
                        sessionStorage.setItem("PERIODO", data['PERIODO']);
                        sessionStorage.setItem("CPA", data['CPA']);
                        window.location.href = ("questionario.html");
                    }
                });
            } else {
                $scope.avisos = "O questionário não está disponível!";
            }
        });
    }
});
```

Fonte: Elaborado pelos autores.

Após ser redirecionado para a página do questionário, o sistema verifica se foram adicionadas na *session* as informações referentes ao aluno. A partir daí, é feita uma busca a partir do curso do aluno e seu período atual, de todas as disciplinas que ele está cursando para montar o questionário. As disciplinas retornadas pela API são armazenadas em um *array* local, conforme mostra a Listagem 7.

Listagem 7 - Método buscaDisciplinas.

```
buscaDisciplinas = function(ID_ALUNO) {
    $http({
        method: "post",
        url: "http://" + host + "/api/buscaDisciplinas/",
        headers: {'Content-Type': 'application/x-www-form-urlencoded'},
        data: ID_ALUNO
    }).success(function(data) {
        for (; index<99; index++) {
            var aux = data[index];
            if (data[index] == null) break;
            arrayDisciplinas.push(aux["NOME"]);
        }
        //Adiciona na tela o nome da primeira disciplina do questionário
        $scope.idAluno = sessionStorage.getItem("ID_ALUNO");
        $scope.nomeAluno = sessionStorage.getItem("NOME");
        $scope.nomeDisciplina = arrayDisciplinas[0];
    });
}
```

Fonte: Elaborado pelos autores.

As perguntas são apresentadas agrupadas por disciplina, de maneira dinâmica, utilizando apenas a página `questionario.html`. Para que as questões ficassem organizadas e a estilização do formulário ficasse mais simples, elas foram importadas do arquivo `template-questoes.html`, localizado na pasta `templates`. Foram feitas validações no formulário para que não ocorresse de questões serem enviadas sem terem sido preenchidas, e, ao fim de responder cada disciplina o questionário é zerado. À medida que o aluno vai respondendo às questões, o sistema cria dinamicamente um objeto JSON com as respostas fornecidas pelo aluno. Como pode ser observado na Listagem 8.

Listagem 8 - Método guardarDados.

```

$scope.guardarDados = function(resposta) {
    resposta.id_aluno = sessionStorage.getItem("ID_ALUNO");
    resposta.disciplina = arrayDisciplinas[aux];
    var data_atual = new Date();
    var dataFormatada = data_atual.toISOString().substr(8,2) +
                        data_atual.toISOString().substr(5,2) +
                        data_atual.toISOString().substr(0,4);
    resposta.data_quest = dataFormatada;

    //Faz uma cópia da resposta atual para não enviar a mesma referência de
    memória
    var novaResposta = angular.copy(resposta);

    //Adiciona a resposta ao Objeto JSON
    JSONRespostas[aux] = novaResposta;
    console.log(JSONRespostas);
    //Prepara para a resposta seguinte
    aux = aux + 1;
    $scope.nomeDisciplina = arrayDisciplinas[aux];
    limparFormulario(resposta);
    $('html, body').animate({scrollTop:0}, 'slow');

    //Verifica se as disciplinas já acabaram encerra o questionário
    if (aux == index) {
        salvarRespostas();
        sessionStorage.clear();
        window.location.href = ("questionario_finalizado.html");
    }
}

```

Fonte: Elaborado pelos autores.

Após o aluno ter respondido as perguntas referentes à todas as disciplinas cursadas no seu período atual, o sistema finaliza o questionário, enviando o dados informados para API, como é mostrado na Listagem 9.

Listagem 9 - Método salvarRespostas.

```

salvarRespostas = function() {
    $http({
        method: "post",
        url: "http://" + host + "/api/salvarRespostas/",
        headers: {'Content-Type': 'application/x-www-form-urlencoded'},
        data: JSONRespostas
    }).success(function(data) {
        console.log("POST COM SUCESSO");
        console.log(data);
    });
}

```

Fonte: Elaborado pelos autores.

Depois de gravar as respostas no banco de dados, o sistema faz também a gravação de uma *flag* junto à ID do aluno, informando que ele já respondeu ao questionário, para que o mesmo não possa vir a responder novamente naquele período. Após finalizar o questionário, o aluno é redirecionado para uma página final de agradecimentos.

3.5.6.2 Ambiente do administrador

O acesso do administrador ao sistema é feito de maneira semelhante ao aluno, na página inicial do aplicativo, o administrador acessa sua área através do logotipo da CPA no topo da tela. Logo em seguida o usuário é redirecionado para uma página de *login* semelhante, onde informa seu nome de usuário e senha de acesso.

Após logar, o administrador é redirecionado para outra tela, na qual logo no início ele se depara com um botão para abrir ou fechar a CPA e uma mensagem de *status* informando se ela está aberta ou fechada. Os métodos que disponibilizam esse recurso são mostrados na Listagem 10.

Listagem 10 - Métodos iniciar e alterarStatus.

```
$scope.iniciar = function() {
    $http.get("http://" + host + "/api/verificaCPA/").success(function(data) {
        if (data == 1) {
            $scope.status = "ON";
            $scope.statusStyle = "statusStyleAberto";
        } else {
            $scope.status = "OFF";
            $scope.statusStyle = "statusStyleFechado";
        }
    });
}

$scope.alterarStatus = function() {
    if (confirm("Deseja continuar?")) {
        $http.get("http://" + host + "/api/alteraStatusCPA/").success(function(data) {
            $scope.iniciar();
        });
    }
}
```

Fonte: Elaborado pelos autores.

O recurso de alterar o *status* da CPA funciona em conjunto com o sistema de notificação do aplicativo, que, quando o questionário fica disponível, uma notificação é enviada para todos

os dispositivos que possuem a aplicação instalada. O código que realiza essa operação é apresentado na Listagem 11.

Listagem 11 - Método responsável pela notificação.

```
angular.module('starter', ['ionic', 'ngRoute'])
.run(function($ionicPlatform) {
  $ionicPlatform.ready(function() {

    var push = PushNotification.init({
      android: {
        senderID: "64422910691"
      },
      browser: {
        pushServiceURL: 'http://push.api.phonegap.com/v1/push'
      },
      ios: {
        alert: "true",
        badge: "true",
        sound: "true"
      },
      windows: {}
    });

    //Captura o ID do aplicativo que receberá a notificação
    push.on('registration', function(data) {
      var ID = data.registrationId;
    });

    push.on('notification', function(data) {
      // data.message,
      // data.title,
      // data.count,
      // data.sound,
      // data.image,
      // data.additionalData
    });

    push.on('error', function(e) {
      // e.message
    });
  });
});
```

Fonte: Elaborado pelos autores.

Também estão disponíveis para o acesso do administrador, uma lista com relatórios, que são impressos na tela em formato de texto ou com gráficos, referentes às questões que foram respondidas pelos alunos. Para utilizar, o administrador deve informar o tipo de relatório desejado e o intervalo de datas que ele deseja que o relatório seja processado. O método que captura a escolha do tipo de relatório a ser impresso e o intervalo entre as datas, é mostrado na Listagem 12.

Listagem 12 - Método gerarRelatorio.

```

$scope.gerarRelatorio = function(relatorio) {
    // O objeto foi copiado, pois o Angular estava acusando erros no Log
    var relatorio_ok = angular.copy(relatorio);
    relatorio_ok.dataInicial = converteData(relatorio_ok.dataInicial);
    relatorio_ok.dataFinal = converteData(relatorio_ok.dataFinal);

    $http({
        method: "post",
        url: "http://" + host + "/api/gerarRelatorio/",
        headers: {'Content-Type': 'application/x-www-form-urlencoded'},
        data: relatorio_ok
    }).success(function(data) {
        console.log(data);
        switch (relatorio.tipo) {
            case 'G_Universidade' :
                $scope.descricao = "Gráfico Geral da Universidade";
                document.getElementById('grafico').innerHTML =
                    '';
                break;
        }
    });
}

```

Fonte: Elaborado pelos autores.

O *input* de captura de datas pelo HTML, retornava uma *string* extensa e com mais informações do que era necessário, além de ser incompatível com o formato de datas do banco de dados. Por esse motivo foi criado o método `converteData`, que faz uma conversão das datas selecionadas para o formato presente no banco de dados, o método citado é mostrado na Listagem 13.

Listagem 13 - Método converteData.

```

converteData = function(data) {
    return data.toISOString().substr(8,2) +
        data.toISOString().substr(5,2) +
        data.toISOString().substr(0,4); }

```

Fonte: Elaborado pelos autores.

Para que a comunicação com a API fosse de fácil manutenção e compreensão do código, foi criado um arquivo que alocasse o endereço responsável por prover os serviços. O arquivo `host.js`, conta com uma única variável que contém a URL necessária para a requisição

HTTP. O arquivo é importado por todas as páginas que necessitam de comunicação com o *web service*, sua sintaxe é apresentada na Listagem 14.

Listagem 14 - Host.js.

```
//var host = "localhost";  
var host = "192.168.0.50";
```

Fonte: Elaborado pelos autores.

Os relatórios presentes no sistema em formatos de gráfico, são gerados a partir da API Google Charts, fornecida gratuitamente pelo Google, sua implementação foi simples e funcional, apresentando as informações de maneira direta e descomplicada para o utilizador. Os formatos de relatório disponíveis foram baseados no relatório geral da CPA da UNIVÁS, com base no ano de 2015.

4 DISCUSSÃO DE RESULTADOS

Nesse capítulo serão discutidos os resultados obtidos com esta pesquisa, focando nos resultados que fazem ligação com a experiência do usuário com o aplicativo, apresentando as telas e sua interface gráfica.

A presente pesquisa teve como objetivo principal, desenvolver um aplicativo móvel híbrido que possibilitasse aos alunos de uma instituição de ensino responderem ao questionário da CPA que é disponibilizado pelo MEC¹. O escopo da aplicação foi definido na intenção de construir o aplicativo obedecendo os padrões que já são presentes na UNIVÁS e o limitando apenas aos alunos da instituição, visto que a CPA também coleta informações na sociedade, dos professores da universidade e de inúmeras outras formas.

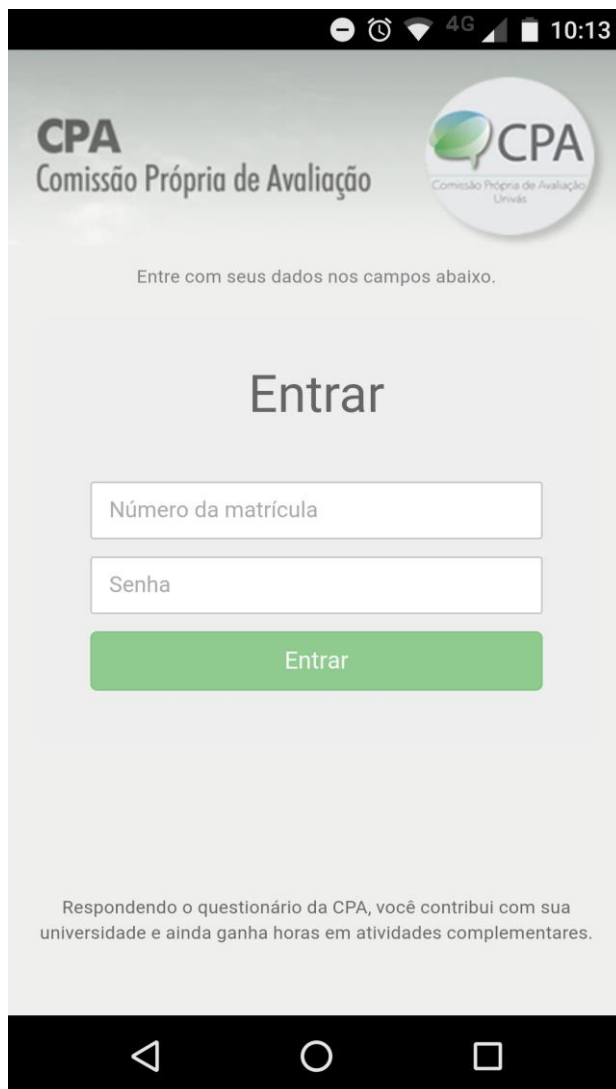
A aplicação consome um serviço Web que foi desenvolvido utilizando a tecnologia RESTful a partir do *framework* Slim, de forma que o código ficasse alocado em um servidor contendo também o banco de dados da aplicação e a API que provê informações através de requisições por métodos HTTP. Desta forma o aplicativo móvel funciona de maneira independente e isolada da API. A escolha do *framework* Slim, utilizado para o desenvolvimento da API foi excelente, pois a ferramenta se mostrou muito efetiva e descomplicada, o que facilitou bastante durante o desenvolvimento.

Durante a fase de desenvolvimento, o aplicativo foi testado utilizando o navegador *Google Chrome* simulando uma resolução de um dispositivo móvel, por conta de sua rapidez para apresentar os resultados durante o desenvolvimento, além do excelente auxílio que a sua ferramenta de log proporcionou. Para realizar testes da aplicação com um contato maior com o hardware, testando a experiência final do usuário, foram utilizados os *smartphones* Motorola Moto X2 e Motorola Moto G3. Como se trata de uma aplicação híbrida, a mesma pode ser compilada para funcionar tanto em dispositivos com Android como iOS, sendo que os testes ocorreram apenas em dispositivos Android, pelo motivo de que para compilar um aplicativo Ionic para iOS, é necessário um computador com o sistema operacional da Apple, equipamento que não contamos para o desenvolvimento da aplicação. Por conta dos testes terem sido feitos sempre nos mesmos *smartphones*, pode ser que ocorram erros com o *layout* da aplicação dependendo do tamanho da tela do dispositivo, contudo, a lógica e o funcionamento da aplicação não serão afetados.

¹ Ministério da Educação

A seguir, na Figura 11, é apresentada a tela inicial do aplicativo, na qual o aluno entra com seu número de matrícula e senha para responder ao questionário.

Figura 11 - Tela inicial do aplicativo.



Fonte: Elaborado pelos autores.

O aplicativo desenvolvido possui fácil usabilidade, *interface* gráfica agradável e manteve visualmente parecido com o recurso já existente na universidade. A operação de rolagem de tela para responder ao questionário foi mantida, visto que, essa operação realizada em um *smartphone* é consideravelmente mais rápida e confortável do que quando executada com um mouse comum, o que foi um ponto positivo para a aplicação, pois se mostrou bastante eficiente. Após logado, o aluno é redirecionado para outra página para responder ao questionário, como é mostrado na Figura 12.

Figura 12 - Tela do questionário.

CPA
Comissão Própria de Avaliação

Aluno(a): 88888888 - Carla Sousa Araujo

Algoritmos II

1 - METODOLOGIA DE ENSINO: refere-se às estratégias de ensino utilizadas pelo(a) professor(a) para favorecer a aprendizagem dos(as) graduandos(as). Indique se o(a) professor(a).

☐ A. utiliza estratégias que favorecem a aprendizagem.

☐ B. às vezes utiliza estratégias que favorecem a aprendizagem.

☐ C. utiliza estratégias que não favorecem a aprendizagem.

2 - CLAREZA DE COMUNICAÇÃO: é esperado que o(a) professor(a) se faça entender pelos(as) graduandos(as). Indique se o(a) professor(a).

☐ A. comunica-se de forma clara, facilitando o entendimento.

☐ B. empenha-se na comunicação, mas é parcialmente entendido(a).

☐ C. comunica-se de forma que dificulta o entendimento.

3 - PLANO DE ENSINO: é esperado que o(a) professor(a) apresente e desenvolva o Plano de Ensino. Indique se o(a) professor(a).

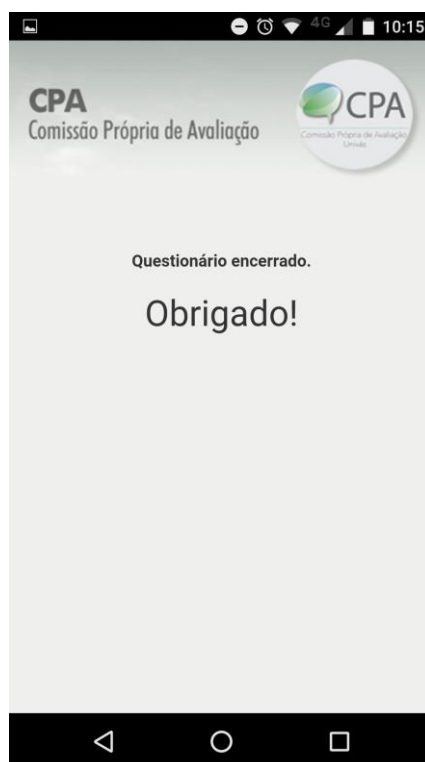
☐ A. apresenta e desenvolve o plano de ensino proposto.

Fonte: Elaborado pelos autores.

A tela apresentada na Figura 12 se refere à primeira sequência de respostas de um aluno, com nome e número de matrícula fictícios, utilizados apenas para testes da aplicação. Um dos problemas enfrentados durante o desenvolvimento, foi de que as questões do questionário possuem uma grande quantidade de texto, tanto nas perguntas quanto nas alternativas de resposta, o que impossibilitou a criação de uma *interface* mais enxuta e aprimorada, nos forçando a manter um design semelhante ao já existente na UNIVÁS.

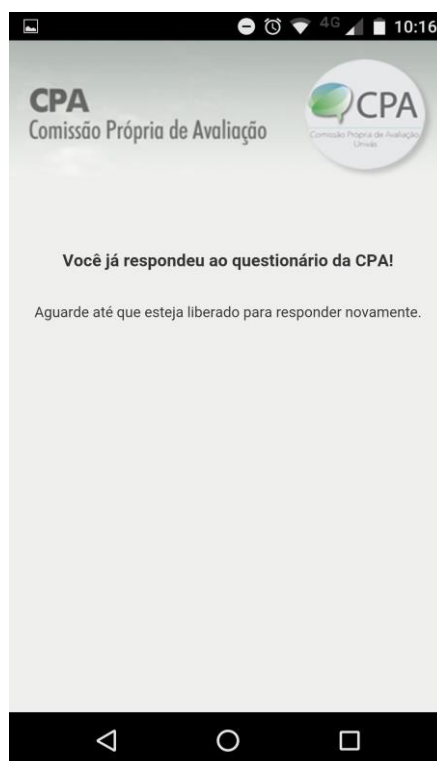
Após terminar o questionário, o aluno é redirecionado para a página final de agradecimentos(Figura 13), a partir daí ele somente poderá responder novamente após o administrador reabrir o questionário. Caso o aluno já tenha respondido, a aplicação apresenta uma mensagem para o usuário na tela, como pode ser visto na Figura 14.

Figura 13 - Tela de encerramento do questionário.



Fonte: Elaborado pelos autores.

Figura 14 - Tela de questionário já respondido.

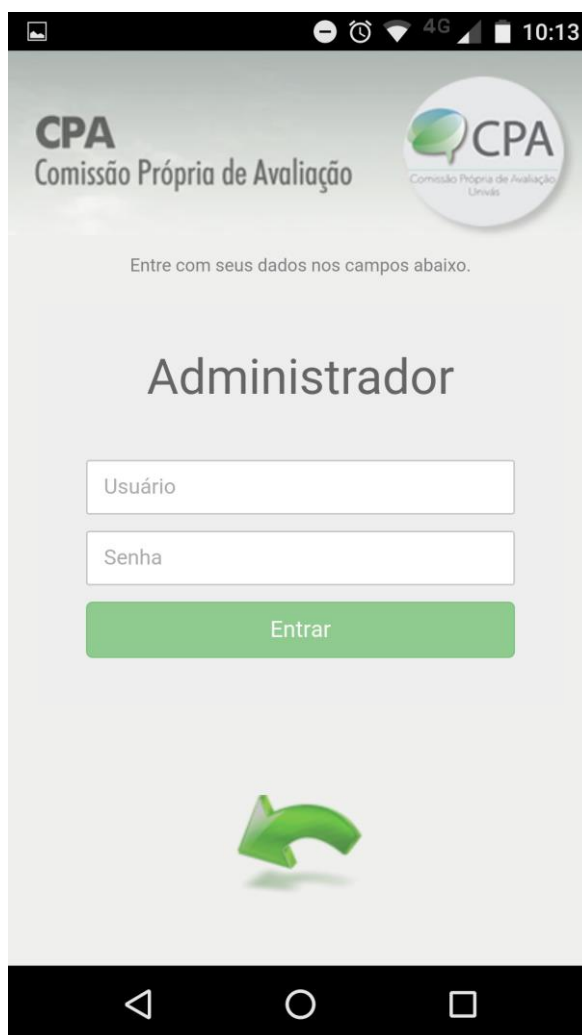


Fonte: Elaborado pelos autores.

Como foi informado anteriormente, também foi desenvolvida a área de acesso para o administrador do sistema, onde o mesmo pode realizar a abertura e o fechamento da CPA além de gerar gráficos e relatórios a partir dos dados coletados. A construção desta segunda área de acesso de nível superior, era essencial para o bom funcionamento da aplicação e para que suas informações pudessem ser acessadas de maneira rápida e descomplicada.

O administrador faz o acesso ao sistema pela da página de *login* do aluno, através do logotipo da CPA localizado no canto superior direito da tela da aplicação, após acessar, ele é redirecionado para outra página de login e deve entrar com seus dados, como pode ser visto na Figura 15.

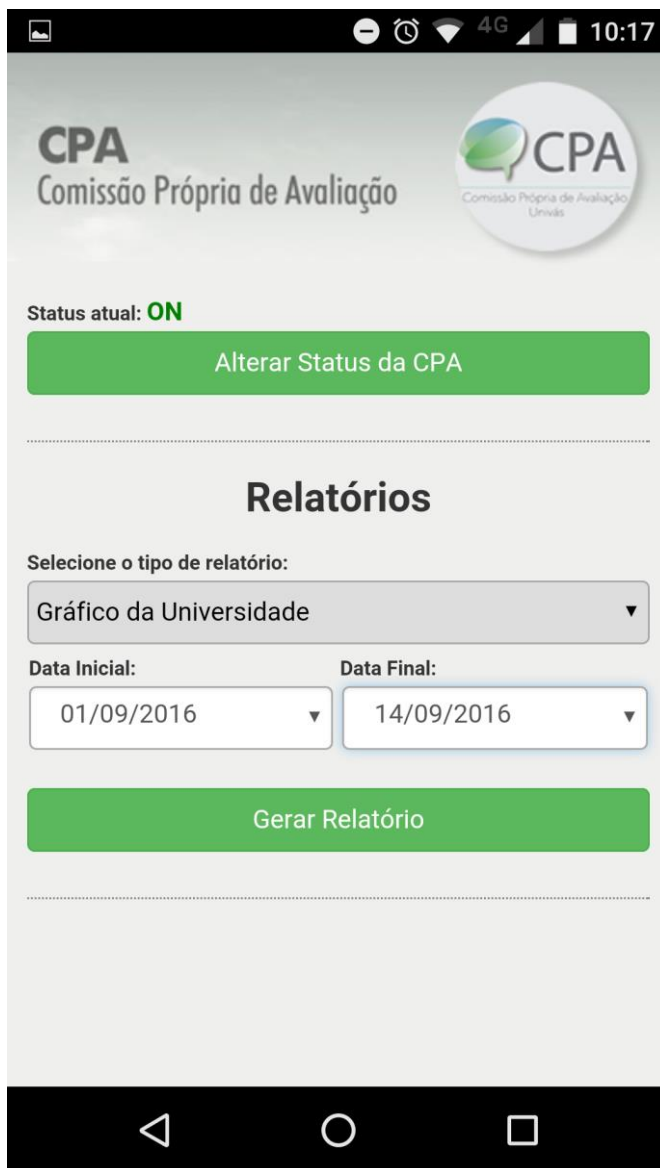
Figura 15 - Tela de login do administrador.

A imagem mostra a interface de login para o administrador de uma aplicação móvel. No topo, há uma barra de status com ícones de bateria, Wi-Fi, 4G e o horário 10:13. Abaixo, o cabeçalho contém o texto "CPA Comissão Própria de Avaliação" à esquerda e um logotipo circular da CPA à direita. O logotipo também contém o texto "Comissão Própria de Avaliação" e "Unidade". Abaixo do cabeçalho, há o texto "Entre com seus dados nos campos abaixo." e o título "Administrador" em uma fonte grande e escura. Seguem-se dois campos de entrada: "Usuário" e "Senha". Abaixo desses campos, há um botão verde com o texto "Entrar". Na base da tela, há um ícone de uma seta verde curva apontando para cima e para a esquerda. A barra de navegação do Android está visível na base da tela.

Fonte: Elaborado pelos autores.

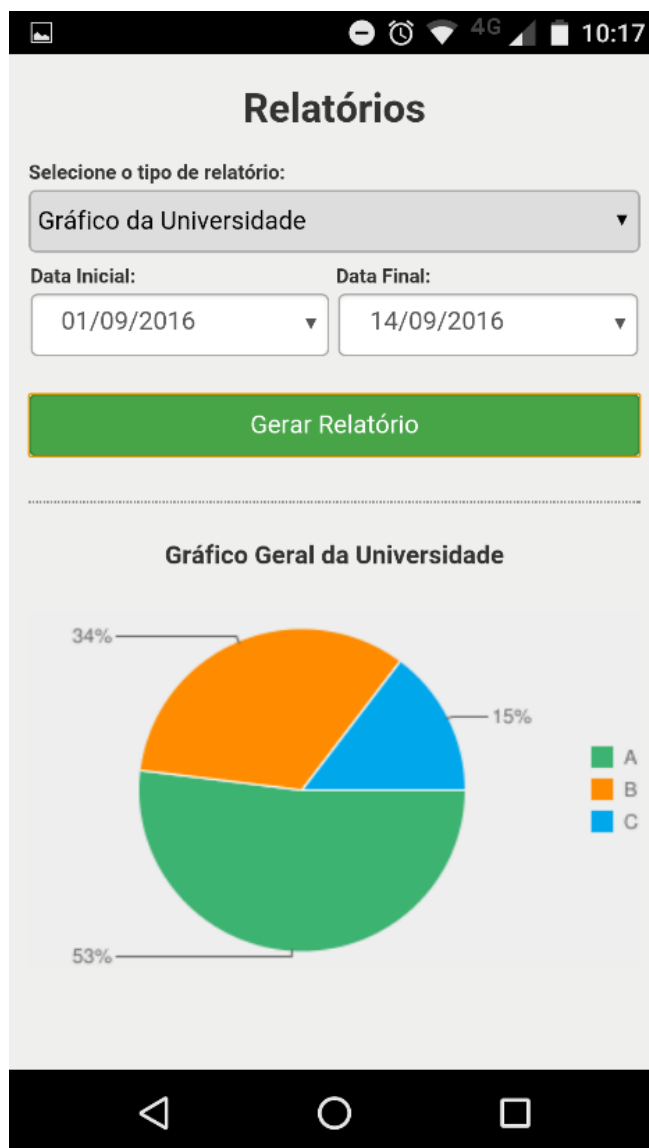
Após efetuar o *login*, o administrador é redirecionado para outra tela, na qual conta com recursos para abrir ou fechar a CPA e gerar relatórios em texto ou gráficos com os dados coletados a partir das respostas dos alunos. A tela do administrador é apresentada na Figura 16.

Figura 16 - Tela do administrador.



Fonte: Elaborado pelos autores.

No processo de geração de relatório, o administrador escolhe seu tipo, o intervalo de datas que deseja que os dados sejam coletados e pressiona o botão Gerar Relatório, uma requisição é enviada para o servidor e retorna os dados para a aplicação, que gera na tela do usuário o relatório que foi escolhido, um exemplo pode ser visto na Figura 17.

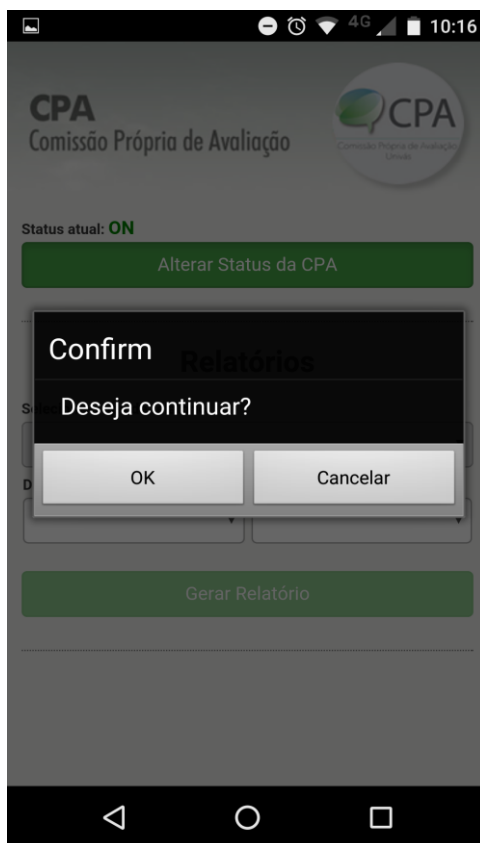
Figura 17 - Exemplo de gráfico.

Fonte: Elaborado pelos autores.

A opção de definir um intervalo de datas personalizado, permite que se obtenham relatórios de questionários realizados em um espaço de tempo maior ou menor do que apenas um semestre, o que pode ser interessante para uma tomada de decisão ou para a construção de um relatório específico.

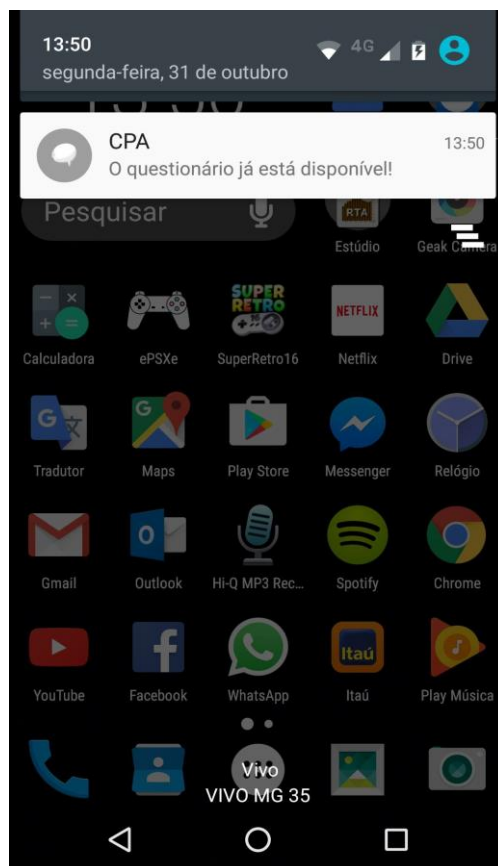
Por questões de segurança, foi implementado também na página do administrador, uma confirmação durante a abertura e fechamento da CPA, para que não ocorresse do botão “Alterar Status CPA” ser pressionado por acidente e assim alterando informações dos alunos. A tela com a mensagem de confirmação é mostrada na Figura 18.

Figura 18 - Confirmação para alterar status.



Fonte: Elaborado pelos autores.

No momento em que o administrador altera o *status* da CPA, o questionário fica disponível para os alunos responderem. No mesmo instante uma notificação é enviada para cada dispositivo que possui o aplicativo instalado, mesmo que ele não esteja aberto no momento, informando aos alunos que o questionário já se encontra disponível. A tela de notificação é apresentada na Figura 19.

Figura 19 - Notificação da CPA

Fonte: Elaborado pelos autores.

Por fim, sabendo-se que a presente pesquisa se enquadra no tipo de pesquisa aplicada, na qual se faz necessário criar um produto real para resolver determinado problema ou facilitar determinada tarefa, o aplicativo desenvolvido supriu muito bem as necessidades de uma universidade para coletar a opinião de seus alunos com o questionário da CPA, pois é de fácil utilização, intuitivo e mostrou-se bastante eficaz durante sua operação.

5 CONSIDERAÇÕES FINAIS

Com a realização deste trabalho, foi desenvolvido um aplicativo que permite a uma instituição de ensino coletar opiniões dos seus alunos seguindo o modelo que é disponibilizado pela CPA e obrigatório pelo INEP. Além do questionário, foi desenvolvida também uma área de acesso para o administrador, para permitir que o mesmo possa gerar relatórios com base nas informações obtidas além de liberar ou bloquear o questionário, operação que automaticamente envia notificações para os dispositivos com o aplicativo instalado.

Para que houvesse troca de informações com um servidor e um banco de dados, foi construído um *web service* utilizando a tecnologia REST, que provê requisições e respostas de informações através do protocolo HTTP, informações essas que são transmitidas através do formato JSON.

A escolha por utilizar o Ionic, o AngularJS e as outras linguagens e *frameworks* para o desenvolvimento da aplicação, foi extremamente feliz, pois as mesmas se mostraram extremamente eficazes em seus funcionamentos e sem complicações durante a implementação dos códigos. Porém, foi notado que o desempenho das mesmas pode não ser satisfatório caso sejam utilizadas para o desenvolvimento de uma aplicação com recursos muito complexos e que exija muito processamento. Nesse caso, é recomendado que o desenvolvedor procure utilizar uma linguagem de programação nativa para desenvolvimento no sistema operacional *mobile*, desta forma, ganha-se desempenho e o resultado final é uma aplicação mais rápida e com maior fluidez.

Apesar das dificuldades encontradas durante o projeto, percebeu-se que a aplicação desenvolvida se comportou de maneira excelente para o que foi proposto, e que se implantada da maneira correta, uma universidade pode conseguir ótimos resultados a partir dos recursos disponibilizados, podendo alcançar muito mais informações no questionário da CPA do que conseguiria sem o uso do aplicativo.

Devido ao tempo escasso para o desenvolvimento, a aplicação desenvolvida não trata com vigor a parte de segurança, podendo ser implementada em outra oportunidade por uma eventual continuação da presente pesquisa. Uma sugestão para pesquisas futuras, é em relação à implementação de outros módulos para o aplicativo, com o propósito de coletar informações da sociedade e dos professores da universidade.

Esta pesquisa foi de grande relevância para os participantes do projeto, pois contribuiu com uma enorme evolução na capacidade de resolução de problemas, além de um vasto conhecimento nas tecnologias utilizadas. Diante das conclusões apresentadas, entende-se que a pesquisa em questão conseguiu corresponder às expectativas iniciais e atender aos objetivos estabelecidos.

REFERÊNCIAS

ANGULARJS. : **AngularJS Documentation**: what is angular? 2016. Disponível em: <<https://docs.angularjs.org/guide/introduction>>. Acesso em: 15 de Fevereiro de 2016.

APACHE, S. F. : **Cordova Documentation**. 2016. Disponível em: <<https://cordova.apache.org/docs/en/6.0.0/guide/overview/>>. Acesso em: 15 de Fevereiro de 2016.

BARROS, A. J. S. e LEHFELD, N. A. S. **Fundamentos de Metodologia**: Um Guia para a Iniciação Científica. 2 Ed. São Paulo: Makron Books, 2000.

BALDUINO, P. : **Dominando JavaScript com jQuery**. São Paulo: Casa do Código Ltda, 2012.

BOOTSTRAP. : **Bootstrap**. 2016. Disponível em: <<http://getbootstrap.com.br/about/>>. Acesso em: 11 de Agosto de 2016.

BRANAS, R. : **AngularJS Essentials**. Birmingham: Packt Publishing Ltd, 2014.

CONAES. : **Comissão Nacional de Avaliação da Educação Superior**. 2016. Disponível em: <<http://portal.mec.gov.br/conaes-comissao-nacional-de-avaliacao-da-educacao-superior/>>. Acesso em 01 de Dezembro de 2016.

EIS, D.; FERREIRA, E. : **HTML5 e CSS3** com farinha e pimenta. São Paulo: Tableless, 2012.

FLANAGAN, D. : **JavaScript**: the definitive guide. Sebastopol: O'Reilly Media, Inc, 2006.

GREEN, B.; SESHADRI, S. : **AngularJS**. São Paulo: Novatec Editora Ltda, 2014.

INEP. : **Inep**. 2016. Disponível em: <http://portal.inep.gov.br/superioravaliacao_institucional>. Acesso em: 23 de Setembro de 2016.

IONIC. : **Ionic Documentation**. 2016. Disponível em: <<http://ionicframework.com/docs/overview/>>. Acesso em: 13 de Fevereiro de 2016.

JOBSTRAIBIZER, F. : **Criação de bancos de dados com MySQL**. São Paulo: Universo dos Livros Editora Ltda, 2010.

KHANNA, R.; HARLINGTON, M. : **Getting Started with Ionic**. Birmingham: Packt Publ., 2016.

LIE, H. W.; BOS, B. : **Cascading Style Sheets**: designing for the Web, portable documents. Boston: Addison-Wesley Professional, 2005.

NIEDERAUER, J. : **Desenvolvendo websites com PHP**. São Paulo: Novatec Editora Ltda, 2004.

NIEDERAUER, J. : **Integrando PHP 5 com MySQL**. São Paulo: Novatec Editora Ltda, 2005.

NOTORM. : **NotORM Documentation**. Disponível em: <<http://notorm.com/#persistence>>. Acesso em: 11 de Agosto de 2016.

PEREIRA, C. : **Node.js**: aplicações web real-time com Node.js. São Paulo: Editora Casa do Código, 2014.

PHP. : **PHP Documentation**: o que é o php? 2016. Disponível em: <http://php.net/manual/pt_BR/intro-what-is.php>. Acesso em: 16 de Março de 2016.

SAUDATE, A. : **REST**: construa api's inteligentes de maneira simples. São Paulo: Editora Casa do Código, 2014.

SILVA, M. S. : **CSS3**: desenvolva aplicações Web profissionais com uso dos poderosos recursos de estilização das css3. São Paulo: Novatec Editora Ltda, 2011.

SLIM. : **Slim Framework Documentation**. 2016. Disponível em: <<http://www.slimframework.com/docs/>>. Acesso em: 16 de Março de 2016.

SINAES. : **Sistema Nacional de Avaliação da Educação Superior**. 2016. Disponível em: <<http://portal.inep.gov.br/superior-sinaes/>>. Acesso em 11 de Agosto de 2016.

WEBBER, J.; PARASTATIDIS, S.; ROBINSON, I. : **REST in practice**: hypermedia and systems architecture. Sebastopol: O'Reilly Media, Inc., 2010.

WILKEN, J. : **Ionic In Action**: hybrid mobile apps with ionic and angularjs. Shelter Island: Manning Publications, 2015.