IAGO SOLIGO SANTOS

DESENVOLVIMENTO DE UM SISTEMA DE GERENCIAMENTO E CONTROLE DO LABORATÓRIO DE TOPOGRAFIA E GEODÉSIA DO IFSULDEMINAS – CAMPUS INCONFIDENTES

INCONFIDENTES / MG
2018

IAGO SOLIGO SANTOS

DESENVOLVIMENTO DE UM SISTEMA DE GERENCIAMENTO E CONTROLE DO LABORATÓRIO DE TOPOGRAFIA E GEODÉSIA DO IFSULDEMINAS – CAMPUS INCONFIDENTES

Trabalho de Conclusão de Curso apresentado como pré-requisito de conclusão do curso de Graduação em Engenharia de Agrimensura e Cartográfica no Instituto Federal de Educação, Ciência e Tecnologia do Sul de Minas Gerais – *Campus* Inconfidentes, para obtenção do título de Bacharel em Engenharia de Agrimensura e Cartográfica.

Orientador: Prof. M.e Paulo Augusto Ferreira Borges

INCONFIDENTES / MG
2018

IAGO SOLIGO SANTOS

DESENVOLVIMENTO DE UM SISTEMA DE GERENCIAMENTO E CONTROLE PARA O LABORATÓRIO DE TOPOGRAFIA E GEODÉSIA DO IFSULDEMINAS – CAMPUS INCONFIDENTES

Data de aprovação: 16 de maio de 2018

Orientador: Prof. M.e Paulo Augusto Ferreira Borges
IFSULDEMINAS – *Campus* Inconfidentes

Prof. M.e Plínio Marcos Piccin Benedito
IFSULDEMINAS – *Campus* Inconfidentes

Prof. Kleber Marcelo da Silva Rezende
IFSULDEMINAS - *Campus* Inconfidentes

Em memória de meus avós,
Maria de Lourdes e Antônio Soligo,
Onofra e Paracelso.

Que estejam orgulhosos daí de cima.

AGRADECIMENTOS

Agradeço primeiramente a Deus por me guiar, proteger, me dar forças e me abençoar.

Aos meus pais, Renata e Amaury, pelo suporte, amor, carinho e incentivo em todas as etapas da minha vida.

À minha companheira Isabela, pelo amor, carinho e dedicação, caminhando comigo em todos os momentos da minha vida, bons ou ruins.

Aos meus irmãos, Igor e Natascha, e seus companheiros, Monique e Eduardo, pelo companheirismo e apoio. Agradeço especialmente ao meu cunhado Eduardo, pelo auxílio e prestatividade na realização do presente trabalho.

A todos os professores do Setor de Agrimensura, pelo conhecimento e experiência compartilhados, em especial ao professor Paulo Augusto Pereira Borges pela paciência e orientação.

RESUMO

O gerenciamento e controle de equipamentos está presente nas mais diversas instituições, tanto públicas como privadas, sendo de grande importância operacionalizar o uso destes equipamentos, bem como a geração de informações acerca de seu uso. No Instituto Federal de Educação, Ciência e Tecnologia do Sul de Minas Gerais - *Campus* Inconfidentes, o gerenciamento dos equipamentos do Laboratório de Topografia e Geodésia, no Setor de Agrimensura, é feito atualmente através de fichas de papel. O presente trabalho busca a criação e implantação de um sistema informatizado para o gerenciamento de entrada e saída dos equipamentos do laboratório para as aulas práticas e uso em pesquisa, cumprindo a demanda do Setor de Agrimensura na geração de informações pertinentes a um gerenciamento efetivo dos equipamentos topográficos. No final do trabalho, foram gerados um banco de dados, interfaces para acesso dos alunos e professores ao sistema e um aplicativo para a leitura de etiquetas fixadas nos equipamentos, mostrando que é possível realizar o gerenciamento de forma mais eficiente através de um sistema informatizado.

Palavras Chave: Gerenciamento de Equipamentos, Desenvolvimento Web, Banco de Dados, Equipamentos Topográficos.

ABSTRACT

The management and control of equipment is present in the most diverse institutions, both public and private, and it is of great importance to operationalize the use of these equipments, as well as the generation of information about their use. At the Federal Institute of Education, Science and Technology of the Southern Minas Gerais - Campus Inconfidentes, the management of the equipment of the Topography and Geodesy Laboratory, in the Land Survey Sector, is currently done through paper forms. The present work seek the creation and implementation of a computerized system for the management of input and output of the laboratory equipment for practical classes and use in research, fulfilling the demand of the Surveying Sector in the generation of pertinent information to an effective management of topography equipment. At the end of the work, a database was generated, interfaces for students and professors access to the system and an application for reading labels fixed on the equipment, showing that it is possible to perform management more efficiently through a computerized system.

Keywords: Equipment Management, Web Development, Database, Topographic Equipment.

Sumário

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
CPF	Cadastro de Pessoa Física
CSS	*Cascading Style Sheet*
HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
IDE	*Integrated Development Environment*
IFSULDEMINAS	Instituto Federal de Educação, Ciência e Tecnologia do Sul de Minas Gerais
PHP	PHP: *Hypertext Preprocessor*
QRCODE	*Quick Response Code*
RA	Registro Acadêmico
REST	*Representational State Transfer*
SDK	*Software Development Kit*
SGBD	Sistema de Gerenciamento de Bancos de Dados
SIAPE	Sistema Integrado de Administração de Recursos Humanos
SQL	*Structured Query Language*
W3C	*World Wide Web Consortium*

LISTA DE FIGURAS

LISTA DE QUADROS

1. INTRODUÇÃO

Em diversas áreas do conhecimento o uso de equipamentos e produtos faz parte do cotidiano de trabalho, os quais realizam as mais diversas tarefas e, por isso, devem possuir a confiabilidade necessária em suas observações, quaisquer que sejam, evitando desvios e erros nos trabalhos.

Na área de agrimensura e cartografia existem os equipamentos topográficos, como teodolitos e estações totais, que praticamente todas as instituições do ramo utilizam, tanto para realização de serviços ou voltado para o ensino profissional. Segundo Suárez e Silva (2014), a utilização de estações totais simplificou a tarefa de se obter ângulos e distâncias, parâmetros essenciais para a área de agrimensura e cartografia, porém, o uso frequente destes instrumentos, juntamente com o tempo de uso e exposição às diferentes condições atmosféricas, faz com que a precisão e a acurácia de suas medições sofram desvios ao longo do tempo. Estes desvios causam um aumento nas incertezas das observações realizadas. Com o acúmulo do uso ao longo do tempo, estes desvios podem resultar em uma situação em que as observações realizadas com uma estação total sejam completamente incertas. A calibração desses instrumentos é um recurso utilizado para solucionar este problema, garantindo a qualidade das observações obtidas com estações totais, sendo necessário o uso deste recurso de maneira periódica para que não sejam comprometidos os trabalhos realizados com instrumentos topográficos.

Prezando-se como principal fator a acurácia das informações adquiridas e produzidas e, devido ao fato dos equipamentos topográficos perderem a confiabilidade das suas observações com o uso ao longo do tempo, as instituições do ramo da agrimensura devem possuir um sistema, não apenas para controlar a entrada e saída dos equipamentos, mas também para

monitorar seu uso, coletando informações sobre suas horas utilizadas, eventuais defeitos apresentados e, com isso, estabelecer a necessidade de ajustes ou manutenções, como, por exemplo, realizar a calibração dos equipamentos, de forma a manter a confiabilidade das observações adquiridas com tais instrumentos.

Um procedimento com intuito semelhante é adotado para o controle de estoque de uma empresa. Segundo Viana (2002), o controle de estoque surgiu devido à necessidade de se melhorar o controle de materiais de uma organização através da tecnologia, substituindo sistemas manuais para sistemas informatizados. Neste mesmo raciocínio, Martins e Campos Alt (2009) dispõem que, atualmente, um sistema de controle de estoque pode ser completamente informatizado com o uso de leitores ópticos para monitorar a saída de produtos, enviando esses dados para o controle de estoque, permitindo análises para aprimorar a tomada de decisões.

No Instituto Federal de Educação, Ciência e Tecnologia do Sul de Minas Gerais (IFSULDEMINAS) – *Campus* Inconfidentes, o controle de saída (empréstimo) e devolução de equipamentos topográficos é realizado, atualmente, de forma manual, onde a saída dos equipamentos é gerenciada fazendo uso de fichas impressas em papel, contendo as informações do empréstimo e devolução do equipamento. A forma atual de controle é altamente propensa a receber erros grosseiros, como, por exemplo, não for constatada a saída de algum equipamento ou a falta de alguma informação de suma importância para a realização de um controle eficaz.

A partir da importância de um controle eficaz do uso de equipamentos topográficos, este trabalho tem o objetivo de criar um sistema informatizado para o laboratório de topografia do IFSULDEMINAS – *Campus* Inconfidentes, o qual será composto por um banco de dados de todos os equipamentos topográficos existentes no laboratório e dados dos alunos dos cursos de Engenharia de Agrimensura e Cartográfica e Técnico em Agrimensura; um sistema de leitura de etiquetas, para automatizar o serviço de empréstimo e devolução dos equipamentos; um sistema de reservas de equipamentos; e um portal *web*, para o acesso dos alunos e professores ao banco de dados.

2. **OBJETIVOS**

2.1 OBJETIVO GERAL

Desenvolver um sistema de controle e gerenciamento de empréstimo dos equipamentos do laboratório de topografia do IFSULDEMINAS – *Campus* Inconfidentes.

2.2 OBJETIVOS ESPECÍFICOS

- Criar um banco de dados relacional e cadastrar os dados dos equipamentos do laboratório de topografia e dos alunos dos cursos de Engenharia de Agrimensura e Cartográfica e Técnico em Agrimensura do IFSULDEMINAS – *Campus* Inconfidentes.
- Desenvolver as funções que serão realizadas sobre o banco de dados em uma API REST para que seja possível utilizar as funções em diferentes linguagens de programação.
- Desenvolver uma página *web* para o acesso dos alunos e professores às informações do banco de dados.
- Desenvolver um sistema digital de controle de entrada e saída de equipamentos do laboratório de topografia do *Campus* Inconfidentes.
- Vincular digitalmente o equipamento emprestado às informações do usuário cadastrado.
- Registrar o número de horas de uso de cada equipamento.

- Cadastrar as informações dos defeitos ou problemas observados pelo usuário durante o uso do equipamento.

3. REFERENCIAL TEÓRICO

3.1 GERENCIAMENTO DE ESTOQUE

Segundo Viana (2000), o controle de estoque é o procedimento adotado para fins de registro, fiscalização e gestão de entrada e saída de produtos ou equipamentos. Independentemente do método, é fundamental observar as rotinas da organização com a finalidade de se evitar problemas de gerenciamento e controle de estoque, que podem vir a causar prejuízos para a empresa.

Viana (2002) dispõe ainda que o gerenciamento e controle de estoque surgiram devido à necessidade de se melhorar o controle de materiais das organizações, substituindo-se o método manual através de fichas de prateleiras ou fichas de controle. Assim, com o desenvolver da tecnologia, o controle de estoque foi aprimorado, substituindo o controle manual por um controle informatizado.

Segundo Martins e Campos Alt (2009), um controle de estoque nos dias atuais pode vir a ser completamente informatizado, permitindo que, na chegada dos produtos, o leitor ótico faça os respectivos registros no sistema. Quando ocorre a saída dos produtos, o leitor também envia esse dado para o controle de estoque, o que permite a realização de análises e comparações dos dados com outros setores e outros períodos, além de realização de novos pedidos quando o estoque atingir níveis previamente estipulados.

3.2 BANCO DE DADOS

Segundo Guimarães (2008), um banco de dados, ou base de dados, é uma coleção de dados ou informações relacionadas entre si. Elas representam aspectos do mundo real com significado próprio e que desejamos armazenar para uso futuro.

Elmasri e Navathe (2005) dispõem que a definição de banco de dados costuma ter um significado restrito à algumas propriedades implícitas, sendo elas:

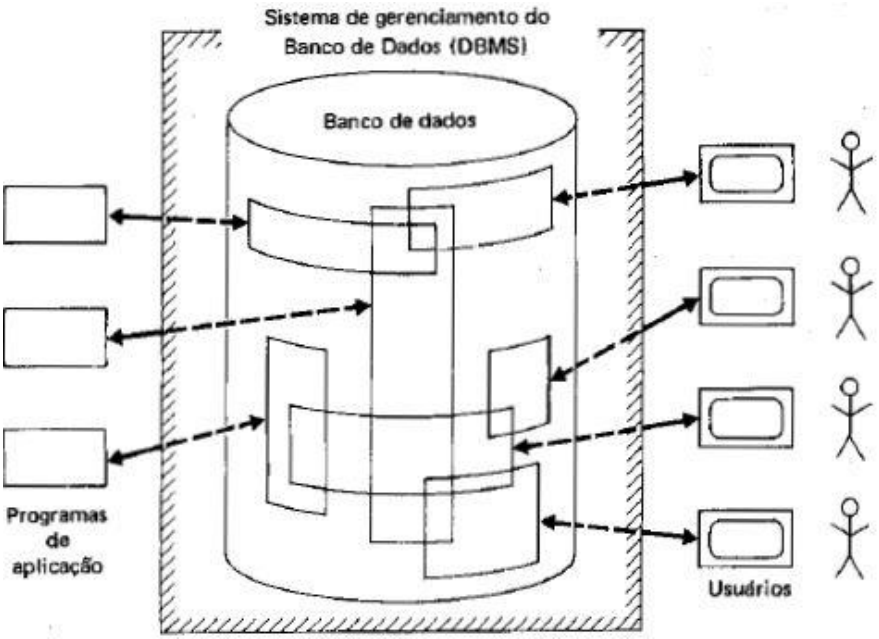
- Um banco de dados representa aspectos do mundo real, e mudanças nesses aspectos do mundo real são refletidas no banco de dados;
- Um banco de dados é uma coleção de dados que possuem significado lógico e coerente. Uma organização de dados randômicos não pode ser interpretada corretamente como um banco de dados;
- Um banco de dados é projetado, construído e povoado por dados segundo uma proposta específica, além de possuir usuários definidos e aplicações concebidas de acordo com o interesse dos usuários.

Em outras palavras, um banco de dados possui fontes no mundo real das quais os dados são derivados, além de algum nível de interação com os eventos ocorridos com as fontes no mundo real (ELMASRI; NAVATHE, 2005).

Guimarães (2008) dispõe que, ao se projetar um banco de dados, deve-se levar em consideração as aplicações que se deseja realizar sobre os dados, aplicações essas que determinarão o principal uso do banco de dados. Bancos de dados podem ser muito simples ou complexos, além de possuírem tamanhos que podem variar em ordens de magnitude entre um banco de dados e outro.

De acordo com Date (1984), um sistema de banco de dados é composto por dados (informações armazenadas no banco de dados), *hardware* (volumes de memória onde reside o banco de dados), *software* (sistemas de gerenciamento de bancos de dados e usuário programador, administrador do banco de dados e usuário final), e é, basicamente, um sistema de armazenamento de dados baseado em computador, cujo objetivo é registrar e manter qualquer informação considerada importante ou significativa para o cumprimento das aplicações às quais o banco de dados tem por objetivos. A figura 1 apresenta o funcionamento simplificado de um sistema de banco de dados.

Figura 1 – Funcionamento simplificado de um sistema de banco de dados



FONTE: (DATE, 1984)

3.2.1 Sistemas de gerenciamento de bancos de dados

Para a conexão entre o banco de dados físico e os usuários do sistema, existe uma camada de *software*. Os *softwares* com essa finalidade são comumente chamados de Sistemas de Gerenciamento de Banco de Dados (SGBD). Todas as solicitações dos usuários para acesso ao banco de dados são manipuladas pelo SGBD, em outras palavras, o SGBD permite que o usuário possua uma visão do banco de dados a níveis acima do nível de *hardware*, além de suportar as operações do usuário expressas em termos do *software* (DATE,1984).

Silberschatz, Korth e Sudarshan (1999) definem um SGBD como sendo constituído por um conjunto de dados, comumente chamado de banco de dados, relacionados a um conjunto de *softwares* que mantêm o acesso ao banco de dados, onde seu objetivo principal é proporcionar um ambiente para recuperação e armazenamento destes dados que seja eficiente e conveniente para os usuários.

Guimarães (2008) complementa indicando que um SGBD auxilia os usuários no gerenciamento do banco de dados, através de ferramentas que permitem que os usuários definam os conjuntos dos dados, construam e preencham o banco de dados, manipulem seu conteúdo através de consulta e atualizem os dados do banco de dados.

Franco (2013), por fim, dispõe algumas propriedades principais que um SGBD deve possuir, sendo elas:

- **Controle de redundância:** um sistema tradicional conta com o armazenamento realizado por diversos usuários, o que pode ocasionar problemas de redundância, ou seja, o armazenamento de uma mesma informação é realizado em locais diferentes. Em um sistema de banco de dados as informações são armazenadas em um único local, exceto quando há a intenção de se duplicar os dados;
- **Compartilhamento dos dados:** um SGBD, em um ambiente multiusuário, deve ser capaz de realizar o compartilhamento dos dados de forma que vários usuários possam realizar operações de atualização sobre o mesmo conjunto de dados, de forma correta e consistente;
- **Controle de acesso:** em um ambiente de vários usuários, é comum que nem todos eles possam ter acesso a todo o banco de dados, qualquer que seja o motivo. Dito isso, um SGBD deve ser capaz de fornecer um subsistema de autorização e segurança, para controlar o tipo de acesso e operações que cada usuário terá sobre o banco de dados;
- **Possibilidade de múltiplas interfaces:** diferentes níveis de conhecimento técnico dos vários usuários do banco de dados provêm a necessidade de que o SGBD seja capaz de apresentar diversos tipos de interfaces, de forma a suprir as necessidades de interpretação de cada usuário. Interfaces para consulta de dados, baseadas em menus ou linguagem natural, são exemplos;
- **Representação de relacionamento complexo entre dados:** um banco de dados pode possuir diversos dados que podem ser inter-relacionados de diversas maneiras. Um SGBD deve representar a variedade de relacionamentos entre dados, além de recuperar e alterá-los relacionando-os de maneira fácil e eficiente;
- **Forças restrições de integridade:** um SGBD deve possuir serviços para garantir a integridade dos dados registrados no banco de dados, como, por exemplo, a especificação de um formato específico para os dados ou assumir valores padrões;
- **Garantir *backup* e restauração de dados:** um SGBD deve disponibilizar recursos para a realização de cópias de segurança e restauração do banco de dados, em caso de falhas de *hardware* ou *software*.

3.2.2 Modelos de banco de dados

Os SGBDs evoluíram de sistemas armazenados em disco, sendo criadas novas estruturas de dados, objetivando o armazenamento de dados de forma mais eficiente e segura, passando a adotar diversas formas de representação, também chamadas de modelos de dados, para dispor

sobre a estrutura das informações contidas em seus bancos de dados (FRANCO, 2013).

Heuser (1998) dispõe que um modelo de banco de dados, ou simplesmente modelo de dados, trata da descrição dos tipos de informações que povoam um banco de dados. Basicamente, o modelo de dados é a descrição formal da estrutura do banco de dados.

> "A escolha do modelo de dados é a **principal ferramenta** no fornecimento de informações sobre a abstração realizada na parte de interesse específico no mundo real" (FRANCO, 2013).

3.2.2.1 Modelo de dados relacional

Segundo Spoto (2000), o modelo de dados relacional foi introduzido por Ted Codd em 1970, sendo um modelo baseado em uma estrutura simples e uniforme denominada *relação*.

Silberschatz, Korth e Sudarshan (1999) dispõem que o modelo relacional foi o primeiro modelo de dados estabelecido para aplicações comerciais e está em uso em diversas aplicações na área de processamento de dados tradicional.

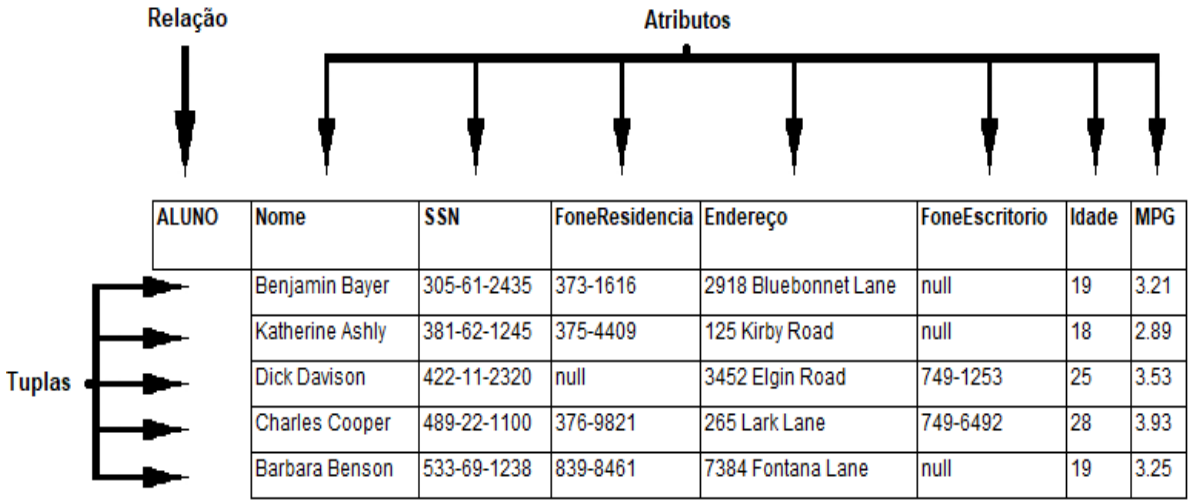
Guimarães (2008) indica que uma tabela no modelo relacional é composta por um conjunto não ordenado de linhas, cada linha representando uma entidade, e que um conjunto de linhas corresponde a um conjunto de entidades do modelo relacional. Cada linha é composta por uma lista de valores dos atributos de uma entidade (o número de valores, ou atributos, em uma linha é constante). O número de linhas de uma tabela varia com o tempo, representando o número corrente de entidades.

Guimarães (2008) indica ainda uma relação de conceitos tradicionais do modelo relacional, onde:

- Relação = Tabela = Arquivo;
- Tupla = Linha = Registro;
- Atributo = Coluna = Campos;
- Domínio = Tipo de dados da coluna = Tipo de dados do campo.

A Figura 2, adaptada de Elmasri e Navathe (2005), ilustra uma relação denominada ALUNO no modelo relacional.

Figura 2 - Os atributos e as tuplas de uma relação ALUNO



ALUNO	Nome	SSN	FoneResidencia	Endereço	FoneEscritorio	Idade	MPG
	Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	null	19	3.21
	Katherine Ashly	381-62-1245	375-4409	125 Kirby Road	null	18	2.89
	Dick Davison	422-11-2320	null	3452 Elgin Road	749-1253	25	3.53
	Charles Cooper	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
	Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	null	19	3.25

Fonte: Adaptado de Elmasri e Navathe (2005)

Desta forma, Franco (2013) define que um chamado esquema de uma relação pode ser definido conforme o exemplo abaixo, de uma relação chamada aluno com os atributos nome, número, turma e departamento:

ALUNO = {nome, número, turma, departamento}.

3.2.2.2 Linguagem de banco de dados relacional – *Structured Query Language* (SQL)

Guimarães (2008) define que o modelo relacional, desenvolvido por Ted Codd em 1970, definiu os conceitos do modelo, suas metalinguagens e sua base na álgebra e cálculo relacionais, trazendo a linguagem SQL, desenvolvida nos laboratórios da IBM nos anos 70, de definição e manipulação de dados relacionais.

Segundo Spoto (2000), a linguagem SQL foi projetada como sendo uma interface para um sistema de banco de dados relacional, sendo, atualmente, a linguagem mais utilizada pelos sistemas de gerenciamento de banco de dados relacionais.

Silberchatz, Korth e Sudarshan (1999) dispõem que a linguagem SQL utiliza uma combinação de construtores em álgebra e cálculo relacional, além de possuir diversos recursos, não apenas a consulta ao banco de dados, como também fornece meios para a definição da estrutura de dados, modificações dos dados no banco de dados e especificações de restrições de segurança.

Spoto (2000) dispõe ainda que a SQL utiliza os termos *tabela*, *linha* e *coluna* para representar uma *relação*, *tupla* e *atributo*, respectivamente. Isso ocorre pelo fato de que a SQL permite que duas ou mais tuplas idênticas possam existir, enquanto não é permitida essa

existência no modelo relacional formal.

Sistemas de banco de dados comerciais não utilizam o modelo relacional em sua forma concisa e formal, mas sim uma linguagem baseada neste modelo formal com uma sintaxe mais simples, além de incorporarem construtores para atualização, inserção e remoção de informações, realizadas como consultas ao banco de dados (SILBERCHATZ; KORTH; SUDARSHAN, 1999).

3.2.2.3 Programação SQL via linguagem hospedeira

Segundo Silberchatz, Korth e Sudarshan (1999), os sistemas de banco de dados atuais podem ser complexos, de forma que possam existir diversas partes independentes, mas que necessitam interagir entre si. Dessa forma, os programas voltados para o lado do cliente devem interagir com os sistemas do lado do servidor.

Guimarães (2008) dispõe, então, que a programação com SQL direto não é a forma mais utilizada para que aplicações acessem um banco de dados relacionais, uma vez que uma aplicação moderna normalmente emprega o uso de interfaces gráficas para utilização pelos usuários. Dessa forma, é necessário um mecanismo que torne possível a inclusão de comandos SQL na linguagem hospedeira, além de uma maneira de enviá-los para o SGBD, receber e processar os resultados dos comandos.

Silberchatz, Korth e Sudarshan (1999) citam como exemplos, as linguagens de programação Pascal e C, que podem ser utilizadas para a criação de interface que pode se comunicar com um banco de dados. Guimarães (2008) complementa esses exemplos com as linguagens C++, Java, Delphi, PHP, Perl, dentre outras, além de denominá-las como linguagens hospedeiras.

3.3 TECNOLOGIAS E LINGUAGENS DE PROGRAMAÇÃO

Esta seção tem como objetivo definir os objetivos e utilizações de tecnologias e linguagens de programação envolvidas e utilizadas no presente trabalho, bem como bibliotecas baseadas nessas linguagens.

3.3.1 *HyperText Markup Language* - HTML

Segundo Silva (2008), HTML é a sigla para *HyperText Markup Language* (linguagem para marcação de hipertexto). Lubbers, Albers e Salim (2013) dispõem que a versão atual da linguagem HTML é a HTML5, que é desenvolvida por três importantes organizações: *Web*

Hypertext Application Technology Working Group (WHATWG), *World Wide Web Consortium* (W3C) e *Internet Engineering Task Force* (IETF).

Silva (2012) dispõe que a marcação HTML foi criada com o propósito de marcação e estruturação de conteúdos. A W3C (2018) define que HTML é a linguagem de marcação básica da *World Wide Web*, sendo projetada originalmente como uma linguagem para descrever semanticamente documentos científicos. Seu design geral, no entanto, permitiu que ele fosse adaptado, ao longo dos anos subsequentes, para descrever diversos outros tipos de documentos e até mesmo aplicativos.

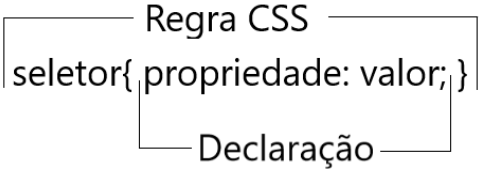
3.3.2 *Cascading Style Sheet* - CSS

Segundo Silva (2012), CSS é a sigla para *Cascading Style Sheet* (Folhas de estilo em cascata). A W3C define CSS da seguinte maneira:

> "Folha de estilo em cascata é um mecanismo simples para adicionar estilos (por exemplo: fontes, cores, espaçamentos) aos documentos web."

Silva (2012) define a chamada Regra CSS como sendo a unidade básica de uma folha de estilo, ou seja, é a menor parcela de um código passível de se produzir algum efeito de estilização, sendo formada por duas partes: o seletor e a declaração. A figura 3 demonstra a sintaxe para se descrever uma regra CSS.

Figura 3 - Sintaxe da Regra CSS



Fonte: (SILVA, 2012)

Onde:

- **Seletor:** é o alvo ao qual a regra CSS será aplicada;
- **Declaração:** determina a estilização que será aplicada ao seletor. É determinada através de dois parâmetros: propriedade e valor.
  - ✓ **Propriedade:** define a característica do seletor que será estilizada (por exemplo: cor, fonte, espaçamentos, preenchimento);
  - ✓ **Valor:** é a quantificação ou qualificação da propriedade (por exemplo: preto, azul, Arial).

3.3.3 Javascript

Segundo Flanagan (2013), JavaScript é uma linguagem de programação da web de alto nível, dinâmica, interpretada e não tipada, onde todos os navegadores modernos possuem inclusos interpretadores JavaScript, fazendo parte da tríade de tecnologias web que todo desenvolvedor de conteúdo *web* deve conhecer, e sua função nessa tríade é a de especificar o comportamento de uma página web.

Bortolossi (2012) descreve JavaScript como sendo uma linguagem de programação interpretada disponível nos navegadores de internet, que disponibiliza diversos recursos de interface gráfica (botões, campos de entrada, seletores) que viabilizam a construção de páginas *web* interativas, além de permitir modificações e integrações, de maneira dinâmica, do conteúdo e da aparência dos diversos elementos que compõem o documento.

3.3.3.1 jQuery

Segundo Silva (2010), jQuery é uma biblioteca Javascript, criada por John Resig, cujo foco é simplificar o desenvolvimento com a linguagem Javascript, com o uso da sintaxe jQuery no desenvolvimento de scripts sem exigir profundos conhecimentos da linguagem.

jQuery é uma pequena e rápida biblioteca JavaScript, que faz coisas como manipulação de eventos, animações e ajax de documentos HTML se tornarem muito simples e fáceis com uma API que funciona em múltiplos navegadores (THE JQUERY FOUNDATION, 2018).

Silva (2009), define Ajax como sendo a sigla em inglês para *Asynchronous JavaScript and XML*, tratando-se de um método de carregar conteúdos de uma página *web* utilizando JavaScript e XML, HTML, TXT, PHP, ASP, JSON ou qualquer linguagem de marcação ou programação que pode ser recuperada de um servidor. As funções para requisições Ajax são simplificadas com a utilização da biblioteca jQuery.

Silva (2010) indica, então, que o jQuery destina-se a facilitar a adição de interatividade e dinamismo às páginas *web*, dispondo ao desenvolvedor funções necessárias à criação de scripts com o objetivo de incrementar a usabilidade, acessibilidade e design com funções de efeitos visuais, busca de informações no servidor sem necessidade de recarregar a página, disposição de interatividade, alteração de conteúdos, modificação de apresentações e estilizações, simplificação de tarefas específicas de JavaScript, dentre outras. A biblioteca jQuery também foi criada em conformidade com os padrões *web*, compatível com qualquer sistema operacional e navegador, além de suporte para as CSS3.

3.3.4 PHP: *Hypertext Preprocessor* - PHP

A linguagem de programação PHP foi criada em 1994 por Rasmus Lerdorf, sendo, inicialmente, apenas um conjunto de scripts voltados para o desenvolvimento de páginas dinâmicas. Com o passar do tempo, outras funcionalidades foram sendo adicionadas à linguagem, até que Rasmus implementou-a em linguagem C, disponibilizando-a em 1995, permitindo o desenvolvimento de aplicações para web de forma muito simples (DALLOGLIO, 2009).

Ainda segundo Dalloglio (2009), a comunidade, a partir de então, passou a contribuir de maneira significativa para o desenvolvimento da linguagem PHP, culminando em novas versões com diversas novas funcionalidades, possibilidade de conexões com múltiplos bancos de dados, suporte a diversos servidores web, dentre outras.

PHP, que significa "PHP: *Hypertext Preprocessor"*, é uma linguagem de programação interpretada, que pode ser utilizada para desenvolvimento *web* e mesclada dentro do código HTML, permitindo o desenvolvimento de páginas geradas dinamicamente (PHP DOCUMENTATION GROUP, 2018).

O código PHP em um arquivo deve estar contido entre as instruções de processamento, ou tags, [**<?php]** e [**?>]**, possibilitando a sua utilização sempre que conveniente , como no exemplo abaixo:

```
<?php
código;
?>
```

O código PHP é executado no lado do servidor, gerando um arquivo HTML que é, então, enviado para o navegador, que exibirá o HTML gerado. (PHP DOCUMENTATION GROUP, 2018).

A linguagem PHP, assim como já disposto por Guimarães (2008), pode ser utilizada como linguagem hospedeira, que permite a execução de comandos da linguagem SQL com o uso de uma interface gráfica, para uso no lado do cliente. Dalloglio (2009) dispõe sobre a criação de conjuntos de classes, que podem oferecer uma interface para manipulação e armazenamento de informações em bancos de dados através da linguagem SQL e que essas conexões com o banco de dados podem ser desenvolvidas através de um ponto central, fora do código-fonte da aplicação, e utilizadas quando necessário.

3.3.4.1 Laravel

Laravel é uma biblioteca em linguagem PHP. Uma das funcionalidades do Laravel é

tornar as interações com bancos de dados extremamente simples contando com uma variedade de APIs de banco de dados utilizando SQL, o construtor de consultas de banco de dados e a ferramenta Eloquent ORM (LARAVEL, 2018).

O construtor de consultas de banco de dados do Laravel fornece uma interface para criação e execução de consultas de bancos de dados, podendo ser usado para a execução da maioria das operações de banco de dados no aplicativo desenvolvido, além de funcionar em todos os sistemas de bancos de dados suportados. Já a ferramenta Eloquent ORM fornece uma implementação simples para trabalhar com banco de dados, onde é gerado um modelo para interagir com cada tabela correspondente, permitindo consultas ou inserção de novos dados nessas tabelas (LARAVEL, 2018).

3.3.5 Typescript

A linguagem denominada TypeScript surgiu como um superset (superconjunto, em tradução livre) da linguagem JavaScript que adicionou funcionalidades que não são disponíveis nativamente ou requerem um grande esforço para utilização (DEVMEDIA, 2018).

Segundo Bierman, Abadi e Torgersen (2014), o TypeScript é uma extensão do JavaScript criada com o intuito de facilitar o desenvolvimento de aplicações em grande escala, oferecendo um sistema de módulos, classes e interfaces.

3.3.6 Ionic Framework

O Ionic Framework é um SDK (Kit de Desenvolvimento de Software) de código livre que permite a criação de aplicativos móveis utilizando tecnologias mais comumente difundidas, como HTML, CSS e JavaScript (IONIC, 2018).

O Ionic Framework é voltado para a aparência e interação com a interface do usuário de um aplicativo, buscando simplificar uma grande parte do processo de desenvolvimento de um aplicativo: o *front-end* (IONIC, 2018).

3.3.7 *Application Programming Interface* – API

A sigla API vem do inglês *Application Programming Interface* (interface de programação de aplicações) e trata-se de um conjunto de rotinas e padrões estabelecidos por determinada aplicação que torna possível que outras aplicações executem as funções da primeira sem conhecimento dos detalhes de sua implementação. Desta forma, uma API permite a interoperabilidade entre diferentes aplicações, além de comunicação entre aplicações e

usuários (PIRES, 2018)

3.3.8 *Representational State Transfer* – REST

Segundo Pires (2018), REST significa *REPRESENTATIONAL STATE TRANSFER* (transferência de estado representacional) e trata-se de uma abstração da arquitetura *web*, consistindo de regras que permitem a criação de um projeto com interfaces definidas.

Lecheta (2015) dispõe que o REST vem sendo cada vez mais utilizado para o auxílio na integração de sistemas, ele utiliza o protocolo HTTP (*HyperText Transfer Protocol*) para a criação de serviços que retornam dados.

Lecheta (2015) dispõe sobre algumas regras e padrões REST, que utiliza os protocolos HTTP, como exemplo:

- GET é utilizado para se realizar uma consulta de dados;
- POST é utilizado para se realizar uma inserção de dados;
- PUT é utilizado para se realizar uma atualização de dados;
- DELETE é utilizado para deletar dados.

4. METODOLOGIA

Esta seção tem como objetivo apresentar os procedimentos adotados para a execução do trabalho, bem como os softwares utilizados.

4.1 SOFTWARES UTILIZADOS

4.1.1 MariaDB versão 10.2.13

O software MariaDB é um servidor de banco de dados em linguagem SQL (Structured Query Language) de código aberto gratuito, apoiado por diversas organizações, sendo o principal administrador o Programa Monty Ab, onde a maioria de seus funcionários faziam parte do núcleo de desenvolvimento do MySQL (THE MARIADB FOUNDATION, 2018).

4.1.2 HeidiSQL versão 9.5

O software HeidiSQL é um Sistema Gerenciador de Banco de Dados projetado para desenvolvedores *web*. Entre suas funções, encontram-se a possibilidade de se navegar e editar dados, criar e editar tabelas, visualizações, procedimentos e eventos agendados, além exportar estruturas e dados para um arquivo SQL, área de transferência ou para outros servidores (BECKER, 2018).

4.1.3 Visual Studio Code

O Visual Studio Code é um editor de códigos-fonte com suporte para JavaScript, TypeScript e Node.js, além de possuir suporte para extensões para outras linguagens, como

C++, C#, Java, Python, PHP, GO (MICROSOFT, 2018).

4.1.4 Android Studio e SDK Tools

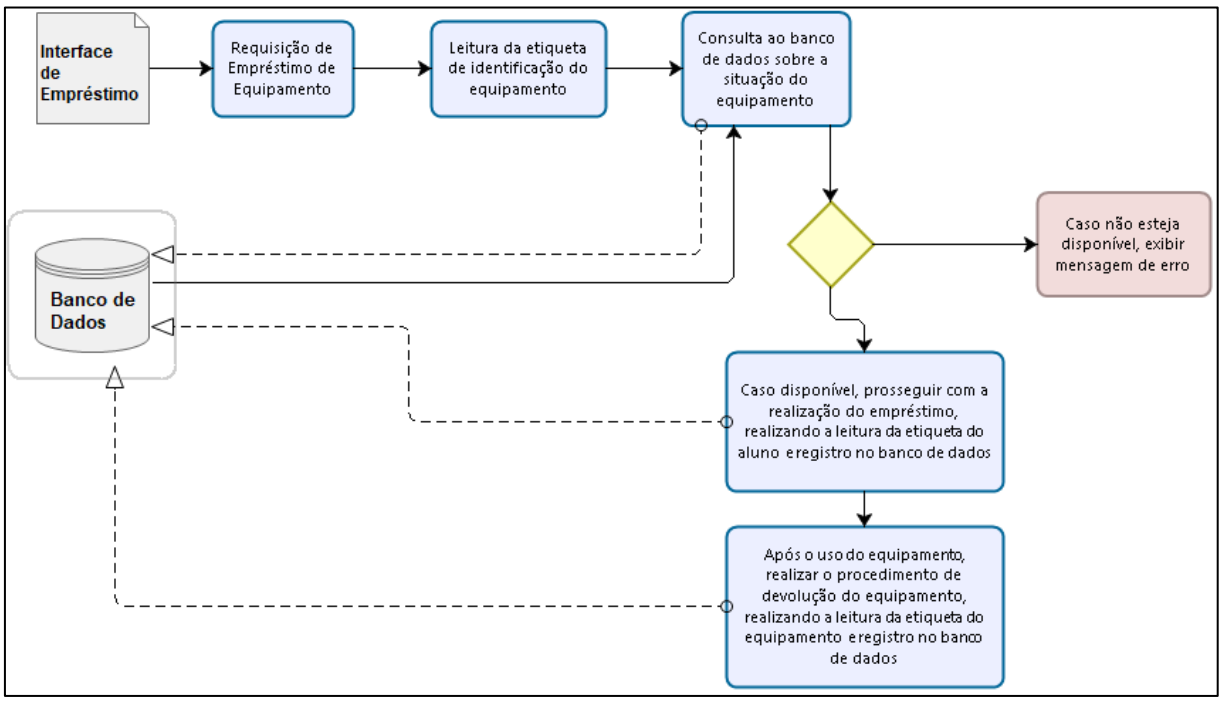
Definido como o IDE (Ambiente de Desenvolvimento Integrado) oficial do Android, oferece as ferramentas para criação de aplicativos em dispositivos Android, além de um flexível sistema de compilação (ANDROID STUDIO, 2018).

4.2 DEFINIÇÃO DAS FUNÇÕES A SEREM IMPLEMENTADAS NO SISTEMA

Para a criação do sistema, foi necessária a definição das funções que seriam implementadas para cumprir com o objetivo de se realizar um gerenciamento dos equipamentos de forma eficaz.

Para o cumprimento do objetivo do trabalho, foi definido que o sistema, para gerenciar os empréstimos realizados pelo laboratório de topografia, deveria ser capaz de armazenar os dados dos equipamentos, alunos e professores do setor de agrimensura do IFSULDEMINAS – *Campus* Inconfidentes e dados dos empréstimos realizados, além de mostrar estes dados em tela, permitindo alterações e exclusões desses dados por parte dos professores. A Figura 4 demonstra um fluxograma com o funcionamento de uma função de empréstimo.

Figura 4- Fluxograma com o funcionamento de uma função de empréstimo



Fonte: Autor

Também foi definido que o sistema possuiria uma função de reserva de equipamentos, onde os alunos poderiam acessar uma interface com autenticação por usuário e senha e reservar um determinado equipamento para determinada data e hora. Desta forma, o sistema também deveria ser capaz de armazenar os dados das reservas efetuadas pelos alunos.

Para o armazenamento destes dados, foi necessária a criação de um banco de dados capaz de suprir as necessidades do sistema, cuja criação está descrita no próximo tópico.

4.3 ESTRUTURAÇÃO E CRIAÇÃO DO BANCO DE DADOS

Para o início do desenvolvimento do sistema, foi necessário, primeiramente, a definição da estrutura do banco de dados de forma a fazer com que ele suportasse as funções as quais o sistema foi proposto. Como a proposta inicial é o controle de empréstimo (saída) e devolução (entrada) de equipamentos, além do controle de uso e calibração do equipamento, uma tabela de nome “equipamento” foi gerada, para armazenar os dados dos equipamentos do laboratório de topografia, contendo como atributos os dados do número do patrimônio, tipo, marca, modelo, status, observações, foto e data da última calibração do equipamento. Desta forma, o esquema da tabela “equipamento” foi definido como demonstrado abaixo.

EQUIPAMENTO = {ID; número do patrimônio; tipo; marca; modelo; observações; data da última calibração}

Além do esquema, foi definido também o domínio de cada atributo, bem como seu nome no banco de dados (uma vez que não é aconselhável a utilização de acentos, espaço ou pontuações para criação de atributos) como é apresentado no Quadro 1:

Quadro 1 - Atributos, domínios e nomes da tabela "equipamento"

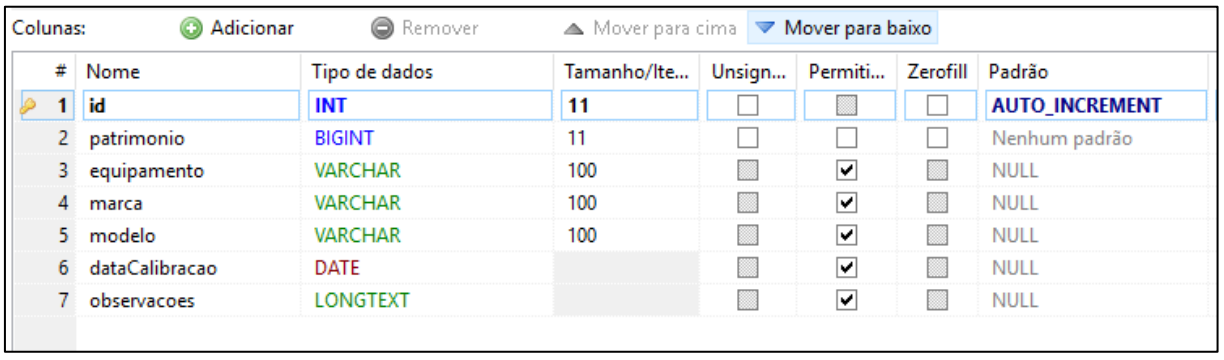
Atributo	Domínio	Nome
ID	Número inteiro	id
Número de patrimônio	Número inteiro	patrimonio
Tipo do equipamento	Cadeia de caracteres	equipamento
Marca do equipamento	Cadeia de caracteres	marca
Modelo do equipamento	Cadeia de caracteres	modelo
Observações sobre o equipamento	Cadeia de caracteres	observacao
Data da última calibração	Data	dataCalibracao

Fonte: Autor

Os domínios apresentados no Quadro 1 foram escolhidos com base na natureza dos dados que serão armazenados, por exemplo, o número de patrimônio refere-se a um número inteiro de 7 dígitos, portanto seu domínio foi definido como número inteiro, o tipo do equipamento trata-se de algumas palavras, portanto seu domínio foi definido como cadeia de caracteres, e assim por diante.

A partir da estrutura definida para a tabela “equipamento”, criou-se a tabela no banco de dados MariaDB com o uso do SGBD HeidiSQL. A Figura 5 ilustra essa tabela.

Figura 5 - Tabela "equipamento" criada com o HeidiSQL



Fonte: Autor

Seguindo o mesmo raciocínio para a criação da tabela “equipamento”, foram criadas as tabelas:

- Aluno: para armazenar os dados dos alunos dos cursos de Técnico em Agrimensura e Engenharia de Agrimensura e Cartográfica do *Campus* Inconfidentes. O esquema para a tabela “Aluno” foi definido como:

ALUNO = {ID; RA; número do celular; nome; CPF; e-mail; senha}

O Quadro 2 apresenta os atributos, domínios e nomes dos atributos da tabela “aluno”.

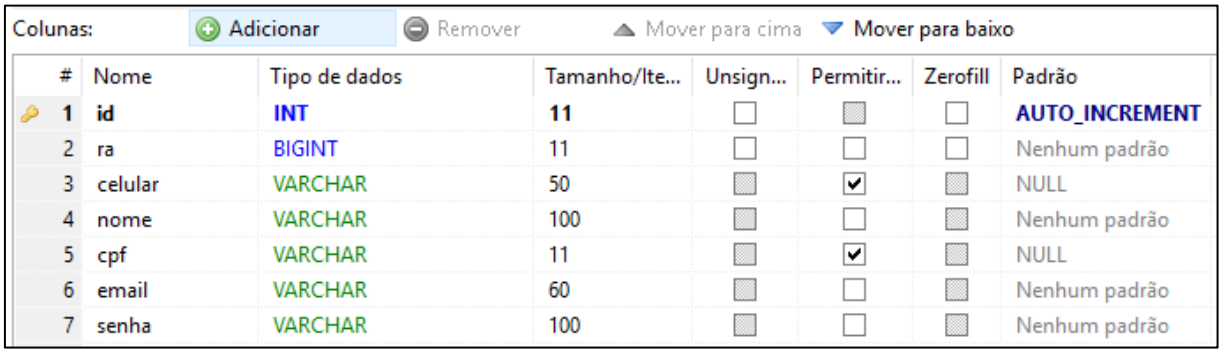
Quadro 2 - Atributos, domínios e nomes da tabela "aluno"

Atributo	Domínio	Nome
ID	Número inteiro	id
RA do aluno	Número inteiro	ra
Número do celular do aluno	Cadeia de caracteres	celular
Nome do aluno	Cadeia de caracteres	nome
CPF do aluno	Cadeia de caracteres	cpf
E-mail do aluno	Cadeira de caracteres	email
Senha do aluno	Cadeia de caracteres	senha

Fonte: Autor

A Figura 6 apresenta a tabela “aluno” efetivamente criada no banco de dados com o HeidiSQL.

Figura 6 - Tabela “aluno” criada com o HeidiSQL



Fonte: Autor

- Professor: para armazenar os dados dos professores do setor de agrimensura do *Campus* Inconfidentes. O esquema para a tabela “professor” foi definido como:

PROFESSOR = {ID; código siape; nome; cpf; e-mail; senha}

O Quadro 3 apresenta os atributos, domínios e nomes dos atributos da tabela “professor”.

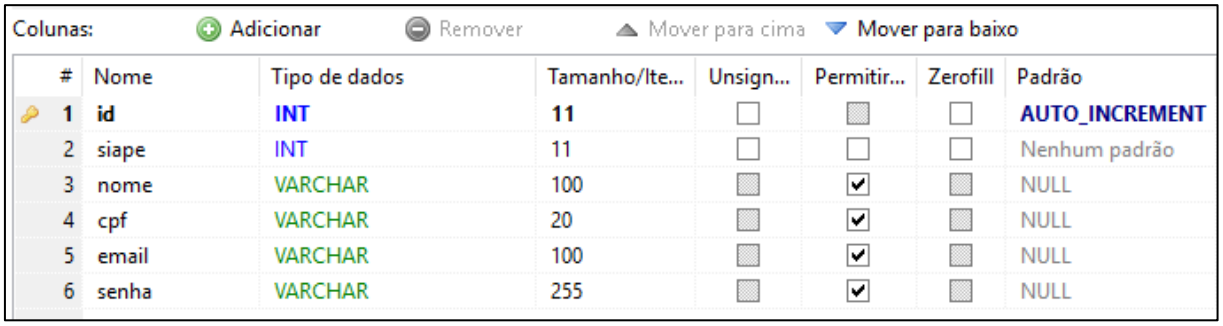
Quadro 3 - Atributos, domínios e nomes da tabela "professor"

Atributo	Domínio	Nome
ID	Número inteiro	id
Código SIAPE do professor	Número inteiro	siape
Nome do professor	Cadeia de caracteres	nome
CPF do professor	Cadeia de caracteres	cpf
E-mail do professor	Cadeia de caracteres	email
Senha do professor	Cadeia de caracteres	senha

Fonte: Autor

A Figura 7 apresenta a tabela “professor” efetivamente criada no banco de dados com o HeidiSQL.

Figura 7 - Tabela “professor” criada com o HeidiSQL



Fonte: Autor

- Empréstimo: para armazenar o histórico de empréstimos realizados no laboratório. Nota-se que esta tabela do banco de dados trata-se de uma tabela de relacionamento entre outras duas tabelas, contando com o uso de chaves estrangeiras para armazenar os dados do aluno e do equipamento que fizeram parte do processo de empréstimo, uma vez que existem tabelas específicas para os dados dos alunos e dos equipamentos. O esquema para a tabela “emprestimo” foi definido como:

EMPRESTIMO = {ID; data e hora do empréstimo; data e hora da devolução; equipamento; aluno; observação}

O Quadro 4 apresenta os atributos, domínios e nome dos atributos da tabela “emprestimo”.

Quadro 4 - Atributos, domínios e nomes da tabela "emprestimo"

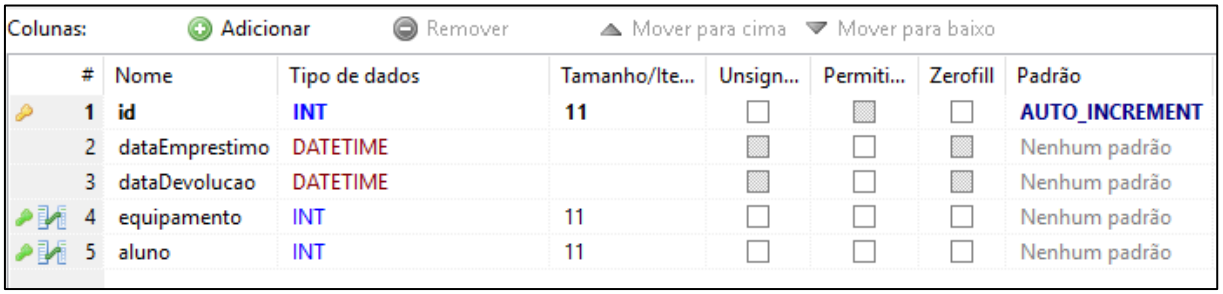
Atributo	Domínio	Nome
ID	Número inteiro	id
Data e hora do empréstimo do equipamento	Data e hora	dataEmprestimo
Data e hora da devolução do equipamento	Data e hora	dataDevolucao
Equipamento	Número inteiro	equipamento
Aluno	Número inteiro	aluno
Observações	Cadeia de caracteres	observacao

Fonte: Autor

Os atributos Equipamento e Aluno tratam-se de chaves estrangeiras, o que justifica o uso do domínio “inteiro”, os dados dos equipamentos e alunos envolvidos no processo de empréstimo serão buscados nas respectivas tabelas através do ID de cada atributo.

A Figura 8 apresenta a tabela “emprestimo” efetivamente criada no banco de dados com o HeidiSQL.

Figura 8 - Tabela “emprestimo” criada com o HeidiSQL



Fonte: Autor

- Reserva: para armazenar os dados das reservas efetuadas pelos alunos dos cursos de Técnico em Agrimensura e Engenharia de Agrimensura e Cartográfica do *Campus* Inconfidentes para posteriores empréstimos. Nota-se que esta tabela, assim como a tabela

“emprestimo”, será uma tabela de relacionamento, contando com o uso de chaves estrangeiras para armazenar os dados dos alunos e dos equipamentos reservados. O esquema para a tabela “reserva” foi definido como:

RESERVA = {ID; Data e hora da reserva; status da reserva; data e hora previstas para o empréstimo; data e hora previstas para devolução; aluno; equipamento; observação}

O Quadro 5 apresenta os atributos, domínios e nomes dos atributos da tabela “reserva”.

Quadro 5 - Atributos, domínios e nomes da tabela "reserva"

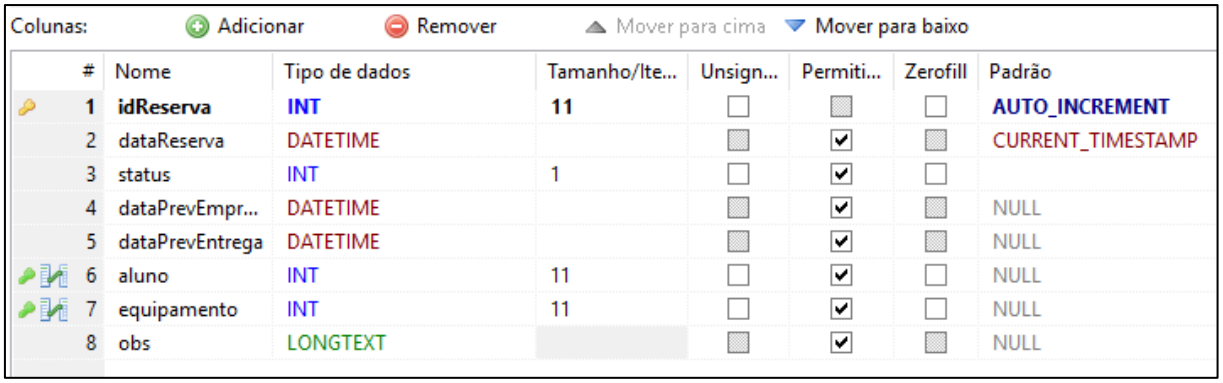
Atributo	Domínio	Nome
ID	Número inteiro	idReserva
Data e hora da reserva	Data e hora	dataReserva
Situação da reserva	Número inteiro	status
Data e hora previstas para o empréstimo	Data e hora	dataPrevEmprestimo
Data e hora previstas para a devolução	Data e hora	dataPrevEntrega
Aluno	Número inteiro	aluno
Equipamento	Número inteiro	equipamento
Observação	Cadeia de caracteres	obs

Fonte: Autor

Assim como na tabela “emprestimo”, os domínios dos atributos “aluno” e “equipamento” são números inteiros devido ao uso de chave estrangeira.

A Figura 9 apresenta a tabela “reserva” criada no banco de dados com o HeidiSQL.

Figura 9 - Tabela “reserva” criada com o HeidiSQL



#	Nome	Tipo de dados	Tamanho/Ite...	Unsign...	Permiti...	Zerofill	Padrão
1	**idReserva**	**INT**	**11**	☐	☐	☐	**AUTO_INCREMENT**
2	dataReserva	DATETIME		☐	☑	☐	CURRENT_TIMESTAMP
3	status	INT	1	☐	☑	☐	
4	dataPrevEmpr...	DATETIME		☐	☑	☐	NULL
5	dataPrevEntrega	DATETIME		☐	☑	☐	NULL
6	aluno	INT	11	☐	☑	☐	NULL
7	equipamento	INT	11	☐	☑	☐	NULL
8	obs	LONGTEXT		☐	☑	☐	NULL

Fonte: Autor

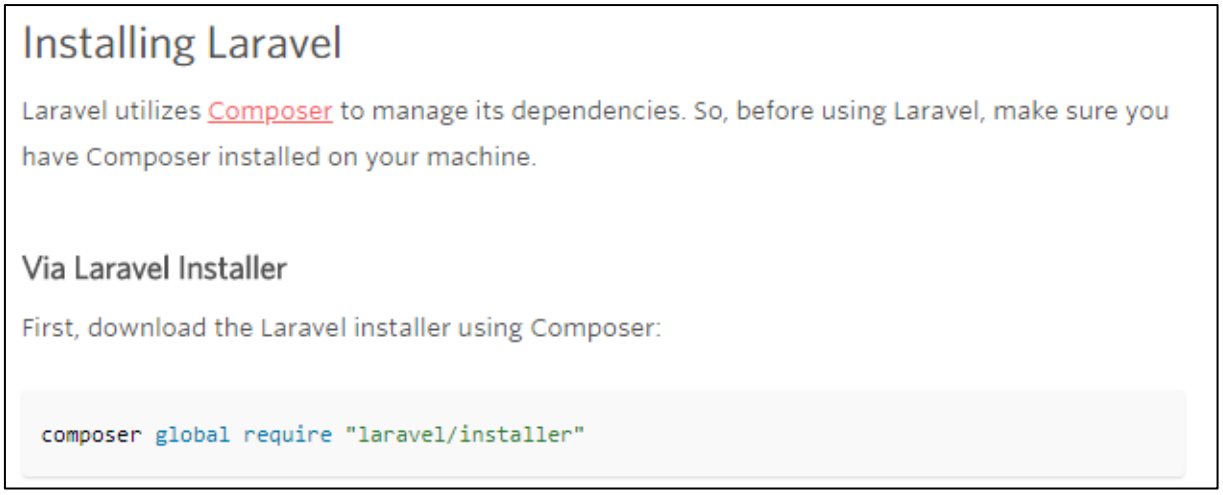
4.4 CRIAÇÃO DA API REST PARA A EXECUÇÃO DE OPERAÇÕES SOBRE O BANCO DE DADOS

Após a criação do banco de dados, que será responsável pelo armazenamento dos dados pertinentes ao sistema, prosseguiu-se para a criação de uma API (*Application Programming Interface*) usando o método *REST*, que será responsável pela execução das operações sobre as tabelas do banco de dados do sistema. Para isso, utilizou-se o *framework* (biblioteca) Laravel, na sua versão 5.6.

A utilização de uma API se justifica pelo fato de que, no planejamento do sistema, foi estabelecida a utilização de um aplicativo para a realização de leituras em etiquetas fixadas nos equipamentos do laboratório, sendo necessária a utilização da API para a integração das funções que serão executadas sobre o banco de dados em diferentes linguagens e aplicações.

Por se tratar de um *framework* em linguagem PHP, primeiramente se fez necessária a instalação do próprio PHP na sua versão estável mais recente (7.2.3), além do *Composer*, que é um gerenciador de dependências para PHP utilizado pelo Laravel e que foi utilizado para sua instalação.

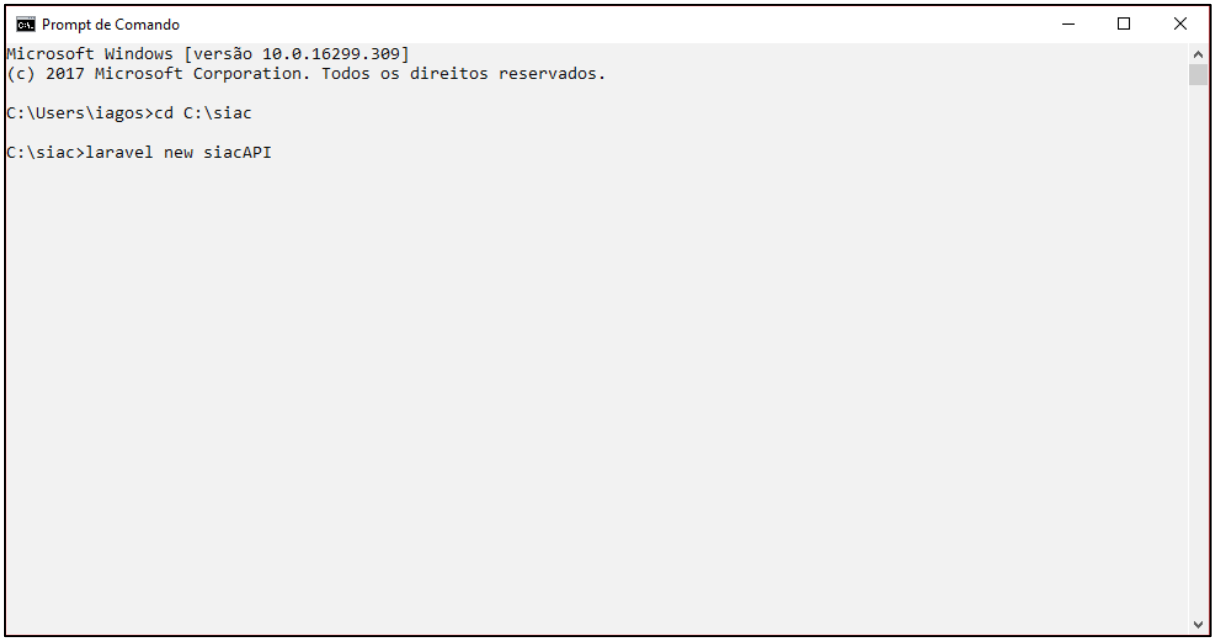
Figura 10 - Instruções para instalação do Laravel



Fonte: (LARAVEL, 2018)

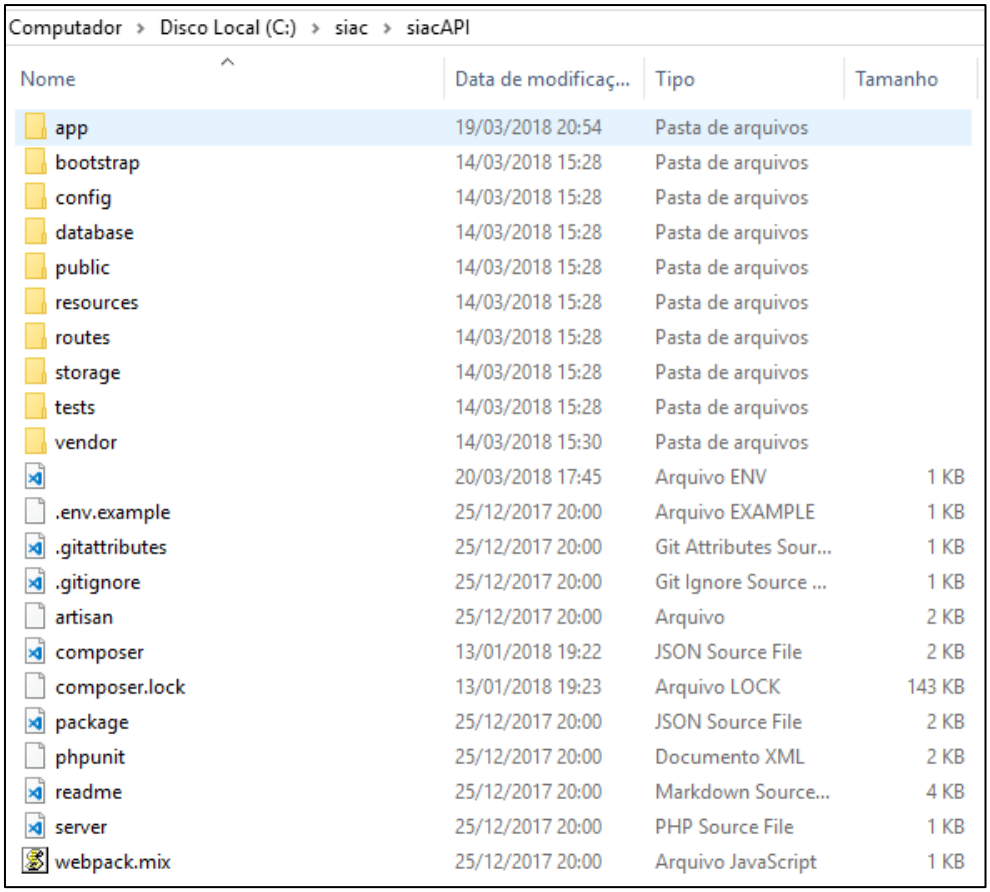
Após a instalação do Laravel, prosseguiu-se para a criação de um diretório para ser utilizado para a API, através do comando *new* do próprio *framework*, conforme instruções de sua documentação. A Figura 11 demonstra a utilização desse comando para a criação do diretório que será armazenada a API e a Figura 12 demonstra o diretório e os arquivos criados pelo Laravel.

Figura 11 - Comando utilizado para a criação de um novo diretório com uso do Laravel



Fonte: Autor

Figura 12 - Diretório e arquivos criados com o comando *new* do Laravel

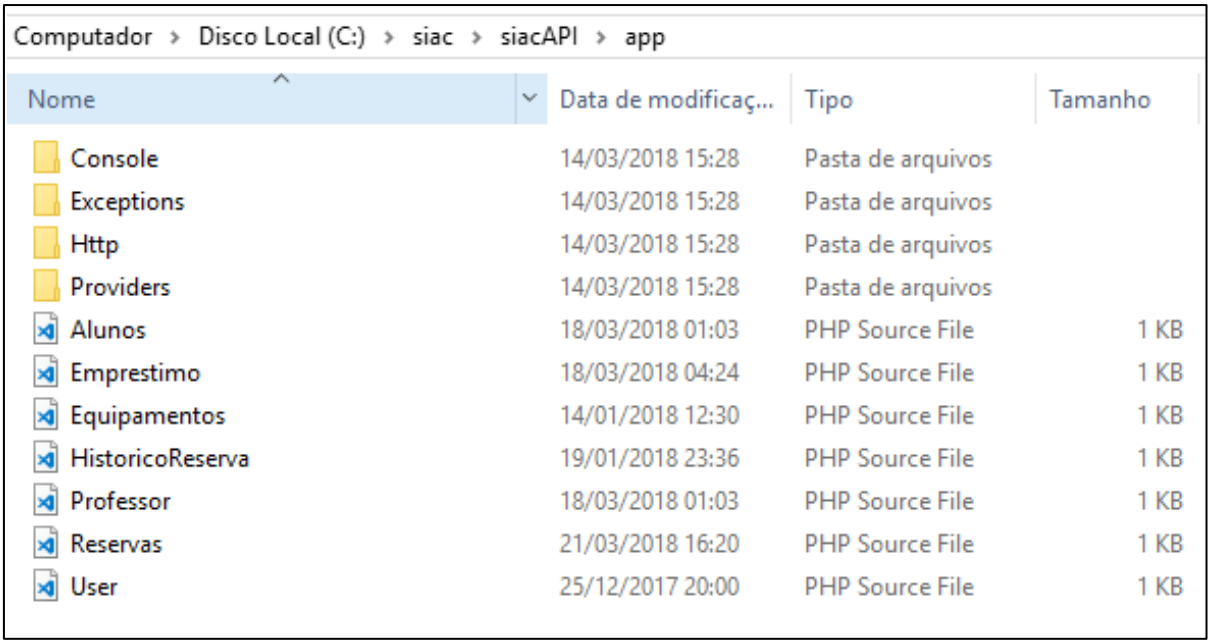


Fonte: Autor

Tendo definido o diretório a ser utilizado, bem como os arquivos base para a criação da API, prosseguiu-se para a criação das conexões e funções que serão executadas sobre as tabelas que compõem o banco de dados. Para isso, foram utilizados comandos *make* do Laravel para a criação de arquivos de extensão .php com estrutura já pré-definida para a finalidade desejada.

Inicialmente foram criados os arquivos que realizarão a conexão com as tabelas do banco de dados, recuperando os dados contidos em suas linhas. Para cada tabela foi criado um arquivo com o nome da tabela e a extensão .php, utilizando o comando *make* do Laravel, por já vir com uma estrutura pré-definida. A Figura 13 demonstra os arquivos criados no diretório e a Figura 14 demonstra o arquivo de conexão com a tabela “alunos”, que foi chamado de “alunos.php”.

Figura 13 - Arquivos criados para a conexão da API com o banco de dados



Fonte: Autor

Figura 14 - Arquivo "Alunos.php" realizando a conexão da API com a tabela "aluno"

Alunos.php

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Alunos extends Model
{
    protected $table = "aluno";
    public $timestamps = false;
    protected $primarykey = "id";

    protected $fillable = [
        "id",
        "ra",
        "celular",
        "nome",
        "cpf",
        "email",
        "senha"
    ];
```

Fonte: Autor

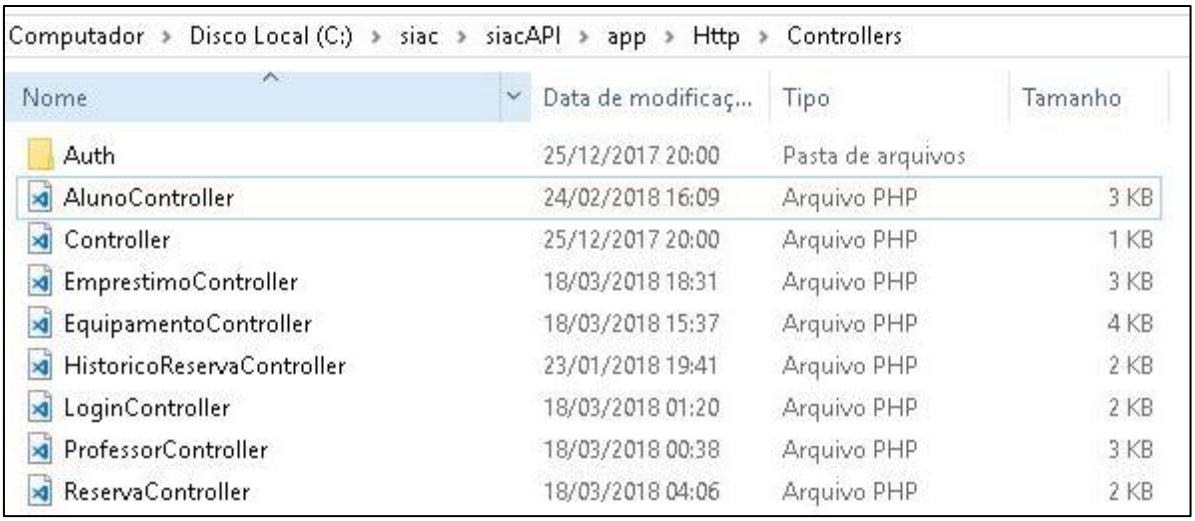
Desta forma, os arquivos de conexão possuem todos a mesma estrutura apresentada na

Figura 14, onde as linhas do código-fonte acima da primeira declaração de *class* são pré-estabelecidas pelo Laravel e a declaração da classe indica a conexão à tabela “alunos”, a desativação da função “timestamps” do Eloquent ORM do Laravel, a declaração da chave primária do banco de dados e os nomes dos atributos de suas linhas.

Após a conexão com o banco de dados ser estabelecida pelos arquivos demonstrados na Figura 13, prosseguiu-se com a criação dos arquivos responsáveis pela definição das funções que seriam realizadas sobre as tabelas do banco de dados. Para isso, foram criados arquivos referentes aos nomes das tabelas do banco de dados com o comando *make:controller* do Laravel.

Em cada arquivo, referente a cada tabela, foram implementadas algumas funções padrões, como recuperar todas as linhas da tabela, recuperar uma linha da tabela com o ID especificado, criar uma nova linha e de atualizar uma linha da tabela. Um arquivo para a função de *login* também foi criado. A Figura 15 demonstra os arquivos criados para cada tabela, além do arquivo de *login*.

Figura 15 – Arquivos que definem as funções da API sobre cada tabela do banco de dados



Fonte: Autor

A documentação do Laravel apresenta a estrutura de um arquivo genérico de *controller*, apresentado na Figura 16.

Figura 16 - Estrutura de um arquivo genérico *controller*

```php
<?php

namespace App\Http\Controllers;

use App\User;
use App\Http\Controllers\Controller;

class UserController extends Controller
{
    /**
     * Show the profile for the given user.
     *
     * @param  int  $id
     * @return Response
     */
    public function show($id)
    {
        return view('user.profile', ['user' => User::findOrFail($id)]);
    }
```

FONTE: (LARAVEL, 2018)

Desta forma, os arquivos criados conforme a Figura 15 apresentam cada um a classe com o nome do arquivo, uma declaração do diretório dos arquivos que realizam a conexão com o banco de dados (conforme Figura 14) e, dentro da classe, a declaração de cada função (*public function*) a ser realizada sobre a tabela correspondente do banco de dados.

No exemplo da Figura 16, é demonstrada a função *show*($id) que representa a função de recuperar os dados de determinada linha da tabela com o atributo especificado. Nos arquivos criados para cada tabela do banco de dados, as outras funções citadas anteriormente, além de outras consideradas necessárias, também foram implementadas.

Após a criação das funções a serem executadas sobre o banco de dados, prosseguiu-se para a criação das rotas a serem utilizadas para a execução efetiva destas funções na interface do sistema. Para isso, um arquivo com extensão .php foi criado, denominado “v1.php”, com a estrutura pré-definida de um arquivo *route* pelo Laravel, onde foram estabelecidas essas rotas.

Na criação das rotas, foram definidas rotas para a função de login e as funções sobre as tabelas do banco de dados. A Figura 18 demonstra a estrutura do arquivo “v1.php”.

Figura 17 - Estrutura do arquivo v1.php

v1.php

```php
<?php

use Illuminate\Http\Request;

/*
|--------------------------------------------------------------------------
| API Routes
|--------------------------------------------------------------------------
|
| Here is where you can register API routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| is assigned the "api" middleware group. Enjoy building your API!
|
*/

// Route::middleware('auth:api')->get('/user', function (Request $request) {
//     return $request->user();
// });

Route::group(['prefix' => 'acesso'], function(){
    Route::post('login','LoginController@login')->name('acesso.login');
});

Route::group(['prefix' => 'cadastros'], function(){
    Route::group(['prefix' => 'alunos'],function(){
        Route::get('lista','AlunoController@index')->name('alunos.lista');
        Route::post('cria','AlunoController@store')->name('alunos.cria');
        Route::get('mostra/{id}','AlunoController@show')->name('alunos.mostra');
        Route::put('altera/{id}','AlunoController@update')->name('alunos.altera');
    });
```

Fonte: Autor

Na Figura 17, além da estrutura do arquivo, é possível observar, na linha 21, a declaração da função de “login” inserida na rota “acesso” no arquivo “v1.php”. Logo abaixo da declaração da rota para a função de login, na linha 26, observa-se também as funções de listar todas as linhas (lista); na linha 27, de criar uma nova linha (cria); na linha 28, de recuperar uma linha específica (mostra); e, na linha 29, alterar os dados de uma linha da tabela (altera), todas relacionadas à tabela “aluno”, na rota “cadastros” e “aluno”, cada função com sua respectiva declaração e referenciamento ao arquivo *controller* da respectiva tabela. Também é possível observar a utilização do método REST para a criação destas rotas, através das declarações “get”, “post” e “put”.

O mesmo procedimento foi adotado para a criação das rotas para as funções das tabelas “professor”, “empréstimo”, “equipamento” e “reserva”, todas armazenadas no mesmo arquivo “v1.php”.

Desta forma, é concretizada a criação da API para a realização das funções a serem realizadas sobre as tabelas do banco de dados, tornando possível a utilização destas funções na interface do sistema, cuja criação é descrita na seção 4.5.

4.5 CRIAÇÃO DAS INTERFACES PARA ACESSO DOS ALUNOS E PROFESSORES ÀS FUNÇÕES DO SISTEMA

Como se trata de um sistema que será utilizado por professores e alunos do Setor de Agrimensura e Cartografia do *Campus* Inconfidentes, o sistema deve ser capaz de realizar suas funções criadas na API em interfaces de simples funcionamento e entendimento.

Além disso, foi definido que os professores seriam responsáveis pela manutenção do sistema, realizando as funções de manter os cadastros de equipamentos e alunos, além da realização de empréstimos, enquanto que os alunos teriam acesso à algumas informações sobre os equipamentos (sem poderem realizar cadastros ou alterações no banco de dados) e ao sistema de reserva.

Desta forma, a interface do sistema foi estruturada segmentando-o em duas "áreas" denominadas "área do professor" e "área do aluno", cada uma com o acesso às respectivas funções e dados que foram pré-estabelecidos, além de uma interface de acesso (login) a cada área. O Quadro 6 simplifica as funções e acesso aos dados de cada área.

Quadro 6 – Esquematização das funções e acesso aos dados de acordo com a área.

	Área do aluno	**Área do professor**
Alunos	- Sem acesso.	- Acesso aos dados dos alunos; - Cadastro de novos alunos; - Modificação de dados dos alunos cadastrados.
Equipamentos	- Acesso às informações dos equipamentos.	- Acesso às informações dos equipamentos; - Cadastro de novos equipamentos; - Modificação de dados dos equipamentos cadastrados.
Empréstimos e Reservas	- Acesso ao sistema de reservas.	- Acesso ao histórico de reservas; - Acesso ao histórico de empréstimos; - Acesso ao sistema de empréstimos.

FONTE: Autor

Após a estruturação e definição das funções a serem desenvolvidas para cada área, prosseguiu-se para sua criação. Para isso, foram utilizadas as linguagens HTML5, para estruturação do menu e formulários de cada área; CSS3 para estilização das páginas; e Javascript, para dinamização e recuperação dos dados do banco de dados, através da API. Além das linguagens, também foram utilizados os *frameworks* jQuery, Bootstrap e Datatables, todos para facilitar a criação das interfaces e recursos definidos.

As recuperações dos dados do banco de dados foram realizadas utilizando o comando ajax do jQuery, utilizando as rotas criadas na API. A Figura 18 demonstra uma requisição ajax no código fonte da página de alunos da área do professor.

Figura 18 - Requisição ajax para listagem dos alunos

```
ajax:{
    url:'http://localhost:8000/v1/cadastros/alunos/lista',
    dataSrc:''
},
```

Fonte: Autor

Pode-se observar a utilização da rota criada no arquivo “v1.php”, que relaciona a função de recuperação de dados da tabela de alunos.

4.6 CRIAÇÃO DO APLICATIVO DE LEITURA DE QRCODE PARA A REALIZAÇÃO DE EMPRÉSTIMOS

Para a realização dos empréstimos, foi definido que eles seriam realizados por meio de leitura de etiquetas contendo códigos de barras QRCode, cada um contendo a identificação do equipamento ou aluno.

Para o cumprimento desta etapa, foi criado um aplicativo em linguagem typescript, com a utilização do *framework* Ionic, além dos plugins de http e qrscanner para a recuperação dos comandos da API e leitura dos QRCodes, respectivamente. Também foi necessária a utilização do Android Studio e SDK Tools, para posterior compilação e instalação em smartphone android.

O aplicativo foi estruturado de maneira com que ele fosse capaz de cumprir as etapas apresentadas no fluxograma da Figura 4, iniciando com uma interface de login para os professores, uma interface posterior que apresenta os empréstimos em aberto (aqueles em que não constar o processo de devolução do equipamento) que também possui a opção de finalizar

o empréstimo aberto, realizando a leitura da etiqueta do equipamento, e a opção de criar novo empréstimo, realizando a leitura tanto da etiqueta do equipamento quanto a etiqueta do aluno.

Realizada a requisição para um novo empréstimo, a câmera do dispositivo abrirá para a leitura do código do equipamento. Feita a leitura, o sistema procurará no banco de dados se o equipamento em questão contém alguma reserva realizada para o mesmo dia, caso contenha, uma mensagem será exibida com duas opções, de continuar o empréstimo ou cancelar, caso continue o empréstimo ou caso o equipamento não tenha reservas para o dia, a câmera do dispositivo abrirá novamente para a leitura da etiqueta do aluno, finalizando o processo de empréstimo de equipamento e registro no banco de dados no atributo “dataEmprestimo” com a data e hora que foi realizado o processo. Após a leitura da etiqueta do aluno, voltará para a página onde são listados os empréstimos em aberto, e o novo empréstimo estará constado nessa lista.

Para a finalização de um processo de empréstimo, bastará que o usuário selecione o empréstimo da lista dos empréstimos em aberto que queira realizar o processo de devolução do equipamento. Feito isso, a câmera do dispositivo se abrirá novamente, para a realização da leitura da etiqueta apenas do equipamento, finalizando o processo de devolução do equipamento, excluindo a linha correspondente da tabela e alterando seu atributo “dataDevolucao” para a data e hora atuais.

O sistema de empréstimo constante no aplicativo complementa o sistema, alimentando a tabela “empréstimo” do banco de dados com os empréstimos realizados. O histórico desses empréstimos poderá ser acessado pelos professores cadastrados na página de empréstimos na área do professor.

5. RESULTADOS E DISCUSSÕES

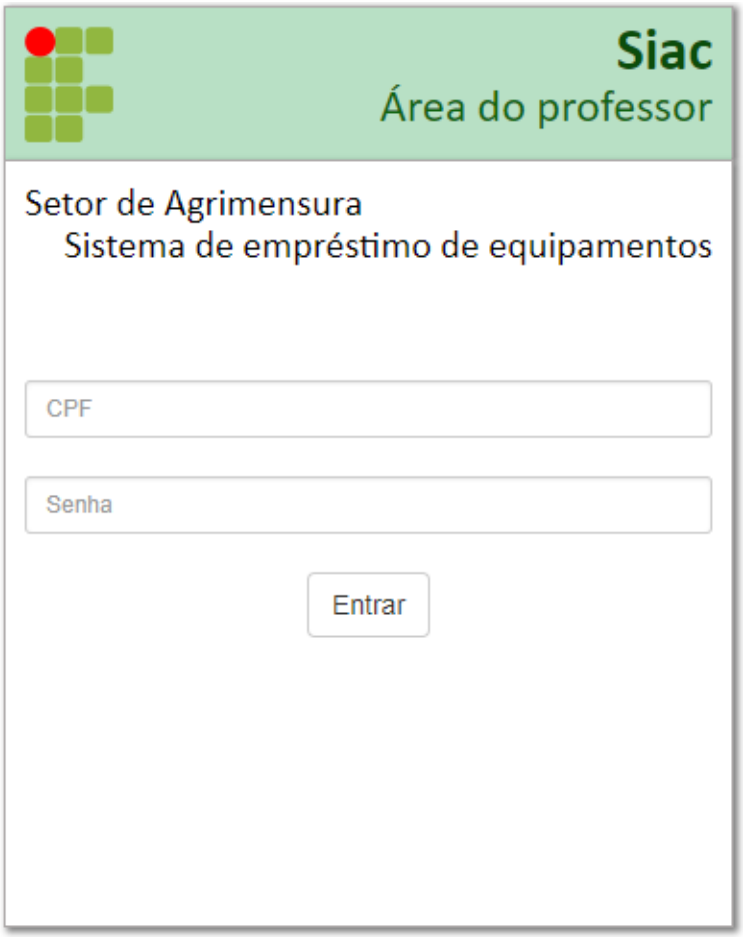
O desenvolvimento do sistema para o gerenciamento dos equipamentos do laboratório resultou em um site de fácil entendimento e acesso, contendo as funcionalidades necessárias para um gerenciamento efetivo dos equipamentos, tendo seus dados armazenados em um banco de dados através de uma API, o que pode vir a possibilitar aprimoramentos e adição de novas funcionalidades ao sistema.

Nos sub-tópicos a seguir, serão apresentadas as interfaces de cada área criada, além da interface do aplicativo e discussões sobre cada interface.

5.1 ÁREA DO PROFESSOR

A área do professor foi estruturada para ser aquela com maior acesso às funcionalidades do sistema. Como sendo de uso exclusivo dos professores do setor de agrimensura, inicialmente a área do professor conta com uma interface de login, onde deverá ser preenchido um formulário de acesso com o CPF e Senha do professor, que esteja cadastrado na tabela “professor” do banco de dados. A Figura 19 demonstra o quadro com o formulário de acesso da página de login da área do professor do sistema.

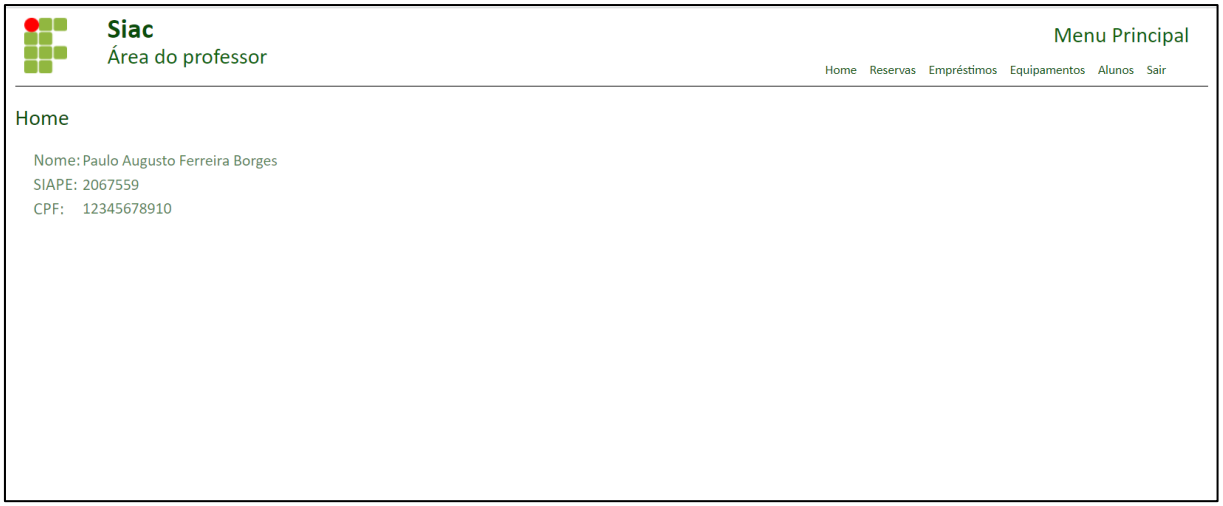
Figura 19 - Formulário de acesso para a área do professor do sistema



Fonte: Autor

Após o preenchimento do formulário e confirmação do envio, os dados são processados pela API, onde são buscados no banco de dados e retornarão com a validação ou não do acesso. Sendo confirmados o CPF e Senha do professor, serão armazenados os dados do professor que acessou o sistema no navegador, que o redirecionará para a próxima página chamada “Home”, que contará com o nome, CPF e Siape do professor, além de um menu para o redirecionamento às outras páginas na parte superior direita. A Figura 20 demonstra a página “Home” da área do professor.

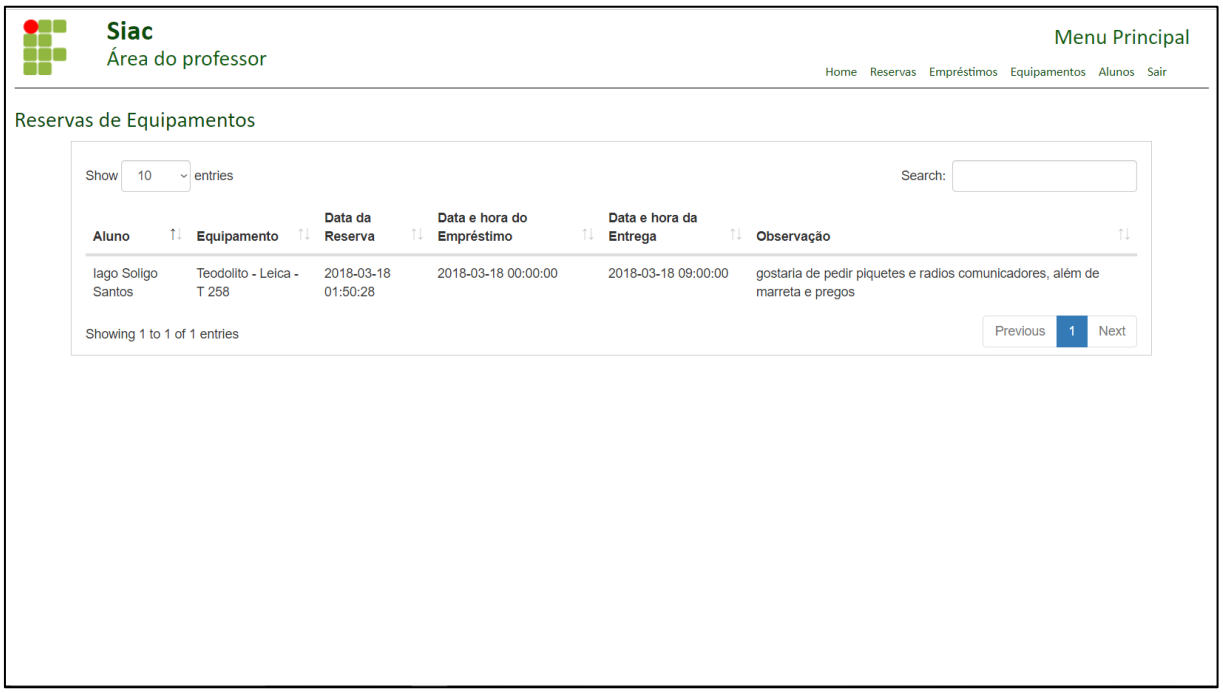
Figura 20 - Página "Home" da área do professor



Fonte: Autor

A Figura 21 demonstra a página “Reservas” da área do professor, nela o professor tem acesso ao histórico das reservas realizadas pelos alunos.

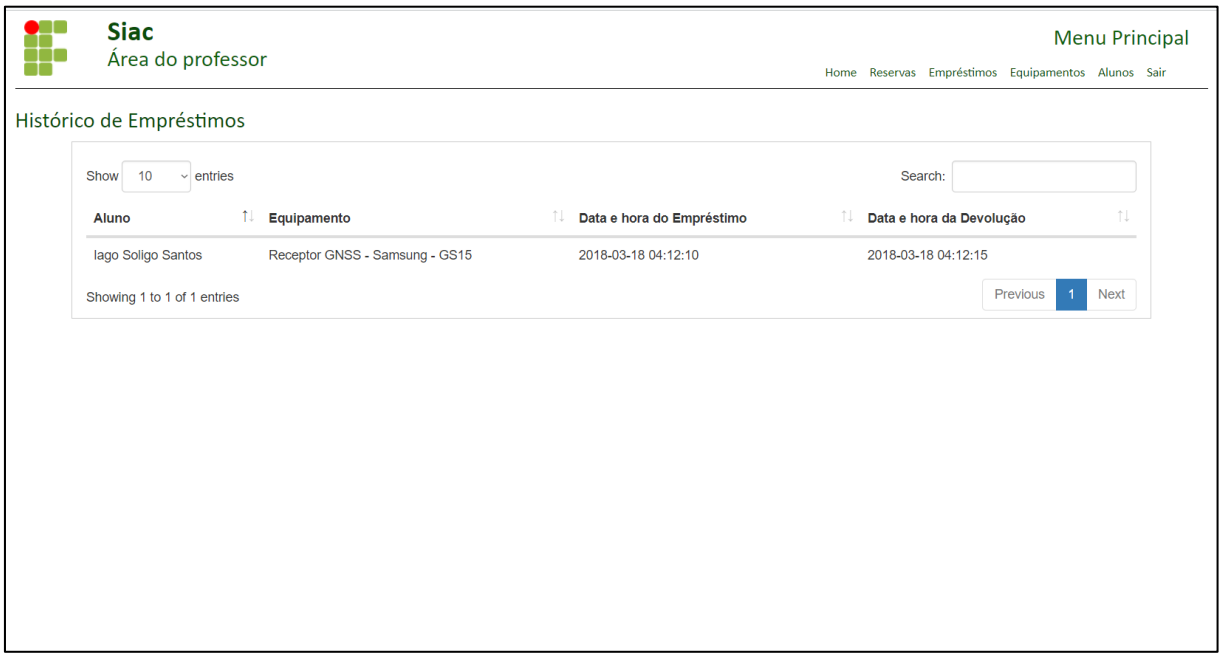
Figura 21 - Página "Reservas" da área do professor



Fonte: Autor

A Figura 22 demonstra a página “Empréstimos” da área do professor, onde os professores terão acesso ao histórico dos empréstimos já realizados, tanto finalizados quanto abertos (sem o processo de devolução do equipamento).

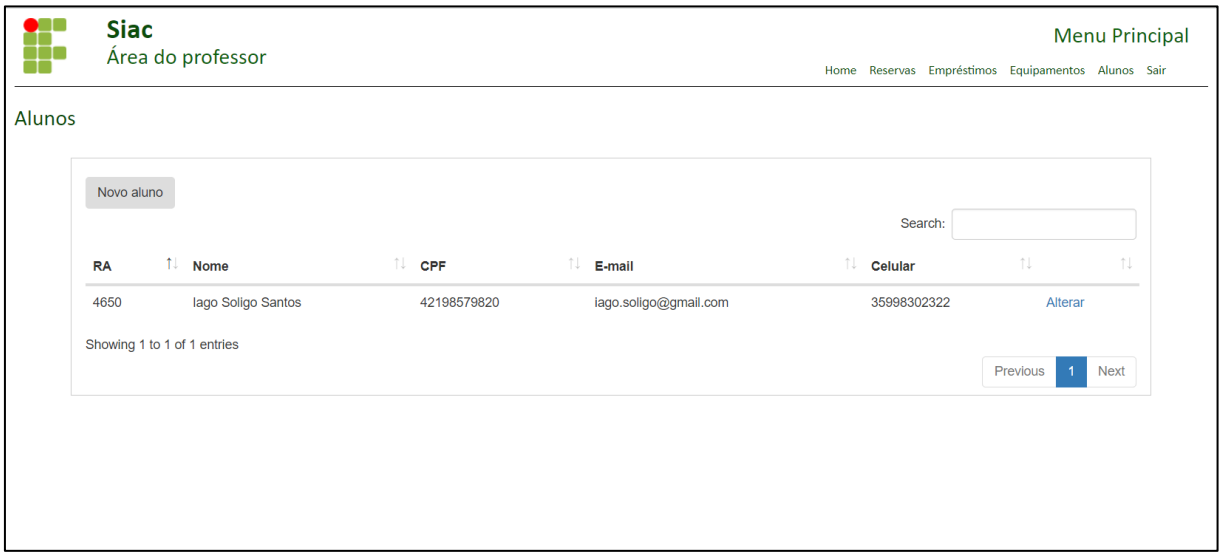
Figura 22 - Página "Empréstimos" da área do professor



Fonte: Autor

A Figura 23 demonstra a página “Alunos” da área do professor, onde será possível que os professores acessem os dados dos alunos cadastrados, bem como realizem novos cadastros de alunos ou alterem os dados de alunos já cadastrados.

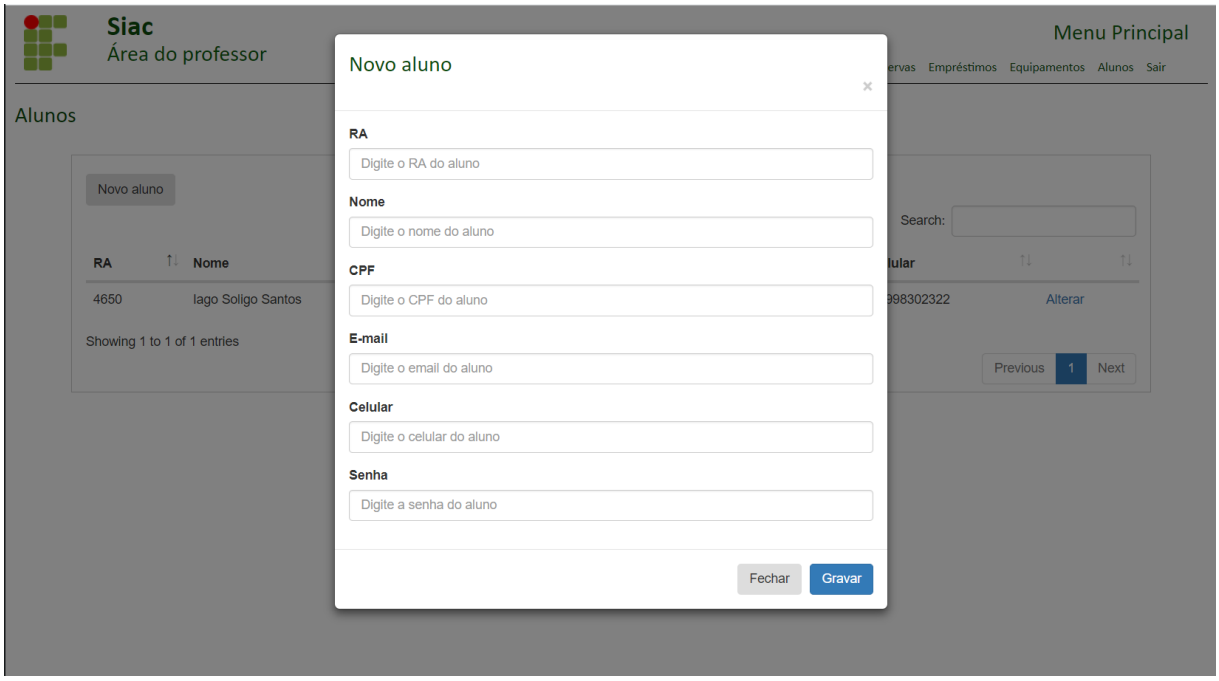
Figura 23 - Página "Alunos" da área do professor



Fonte: Autor

A Figura 24 demonstra o modal com a função de cadastrar um novo aluno, na mesma página “Alunos”.

Figura 24 - Modal de cadastro de novo aluno na página "Alunos" da área do professor

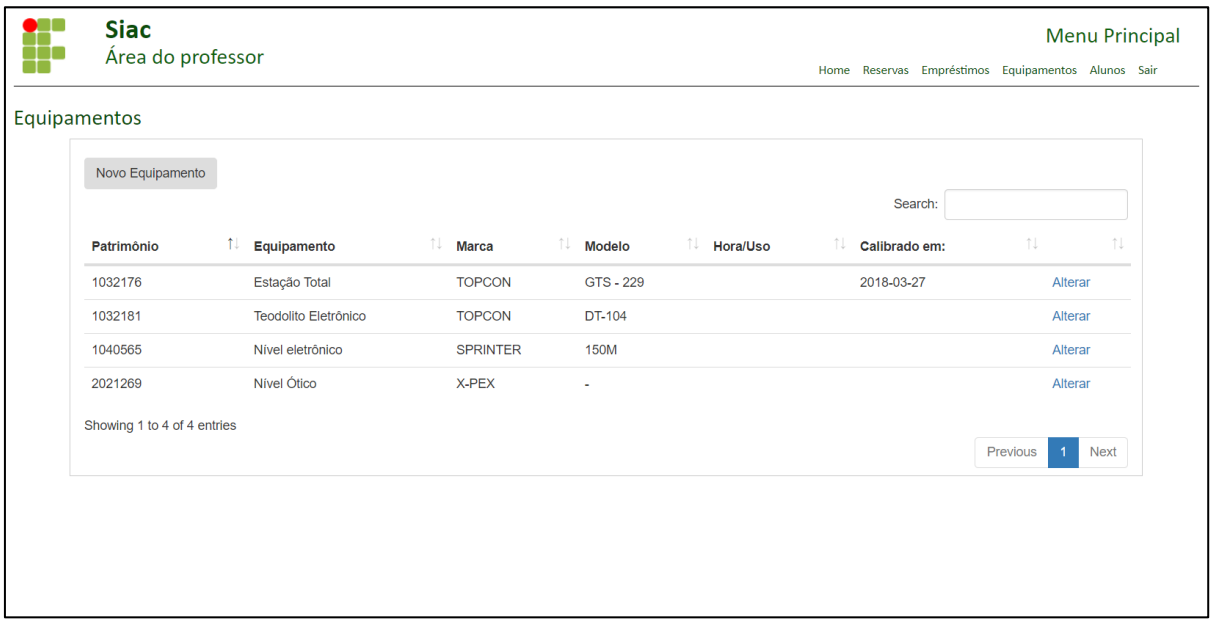


Fonte: Autor

O mesmo modal é demonstrado na função de alterar os dados de um aluno já cadastrado, porém já com autopreenchimento dos campos do formulário, com exceção da senha do aluno.

A Figura 25 demonstra a página “Equipamentos” da área do professor, onde será possível que os professores acessem os dados dos equipamentos cadastrados, bem como realizem novos cadastros de equipamentos ou alterem os dados de equipamentos já cadastrados.

Figura 25 - Página "Equipamentos" da área do professor



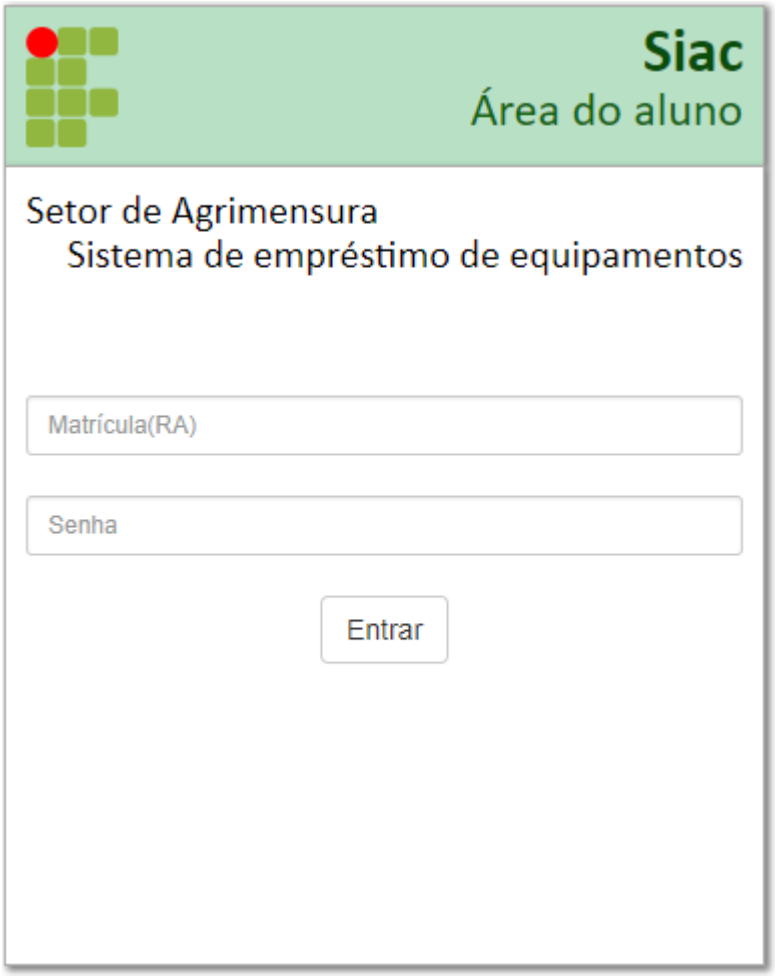
Fonte: Autor

Na página “Equipamentos”, um modal semelhante ao apresentado na Figura 24 é mostrado ao se executar a ação de cadastrar um novo equipamento ou alterar os dados de um equipamento já cadastrado.

5.2 ÁREA DO ALUNO

Assim como a área do professor, a área do aluno possui inicialmente um painel de acesso onde é requisitado o RA e senha do aluno, que esteja cadastrado na tabela “aluno” do banco de dados. A Figura 26 demonstra este painel de acesso da área do aluno.

Figura 26 - Painel de acesso da área do aluno



Fonte: Autor

Após o preenchimento do formulário e confirmação do envio, os dados são processados pela API, onde são buscados no banco de dados e retornarão com a validação ou não do acesso. Sendo confirmados o RA e senha do aluno, serão armazenados os dados do aluno que acessou o sistema no navegador, que o redirecionará para a próxima página chamada “Home”, que

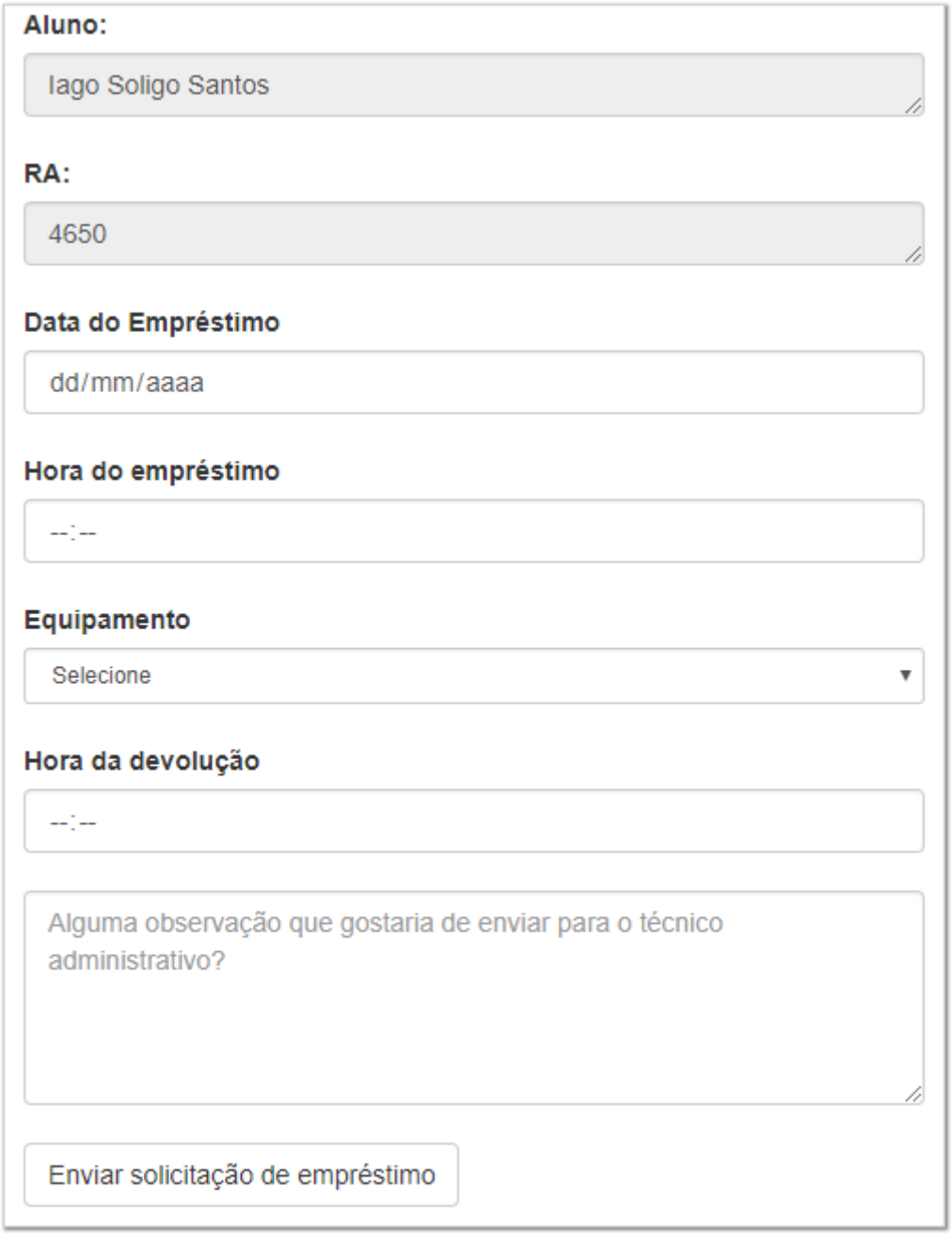
contará com o nome e RA do aluno, além de um menu para o redirecionamento às outras páginas na parte superior direita. A Figura 27 demonstra a página “Home” da área do aluno.

Figura 27 - Página "Home" da área do aluno

Fonte: Autor

A Figura 28 demonstra o formulário da página “Reservar Equipamento” da área do aluno. Nela os alunos poderão preencher um formulário para a requisição de uma reserva de um equipamento, que será armazenada no banco de dados do sistema e apresentada na página “Reservas” da área do professor.

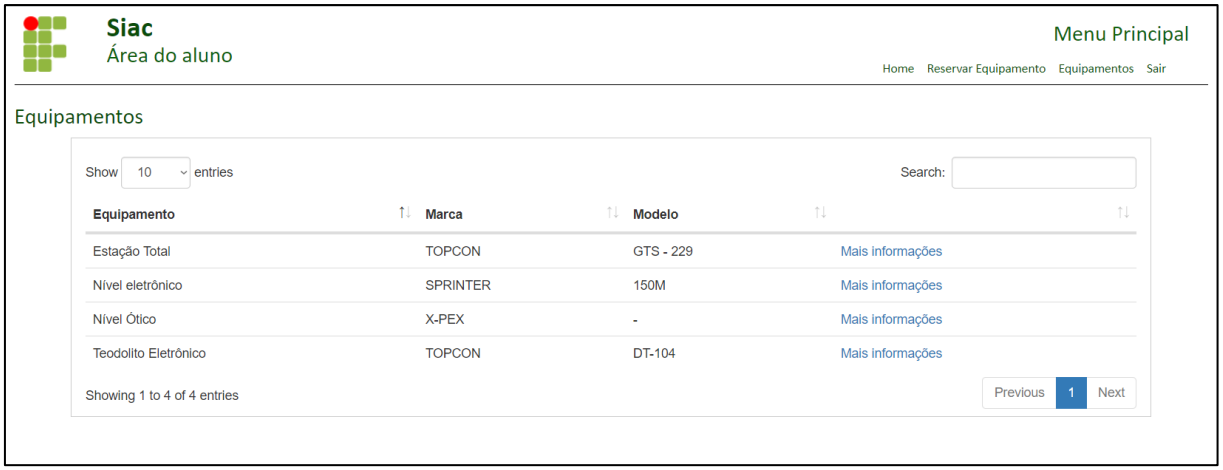
Figura 28 - Formulário para reservar um equipamento na página "Reservar Equipamento" da área do aluno



Fonte: Autor

A Figura 29 demonstra a página “Equipamentos” da área do aluno. Nela os alunos poderão verificar as informações dos equipamentos cadastrados no sistema, porém sem acesso a cadastrar um novo equipamento ou alterar dados de um equipamento já cadastrado.

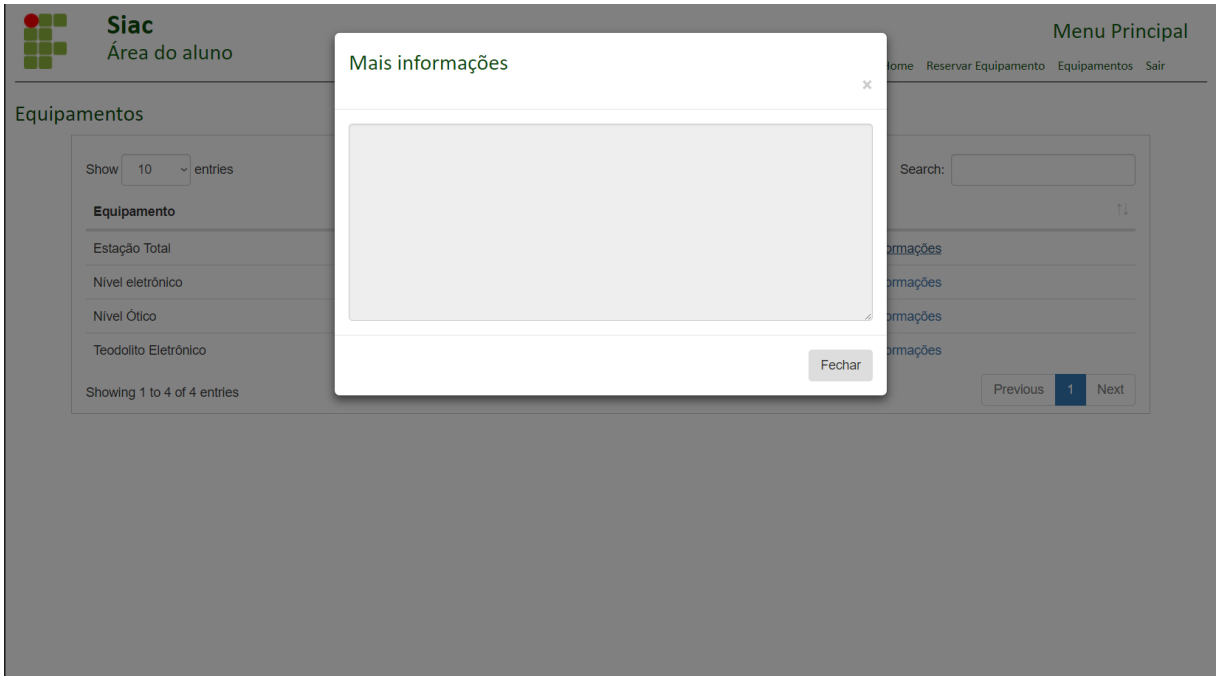
Figura 29 - Página "Equipamentos" da área do aluno



Fonte: Autor

A Figura 30 demonstra a função de “Mais informações” sobre determinado equipamento. No modal apresentado, consta os dados cadastrados no campo “Observações” do equipamento

Figura 30 - Modal de "Mais informações" aberto na página "Equipamentos" da área do aluno



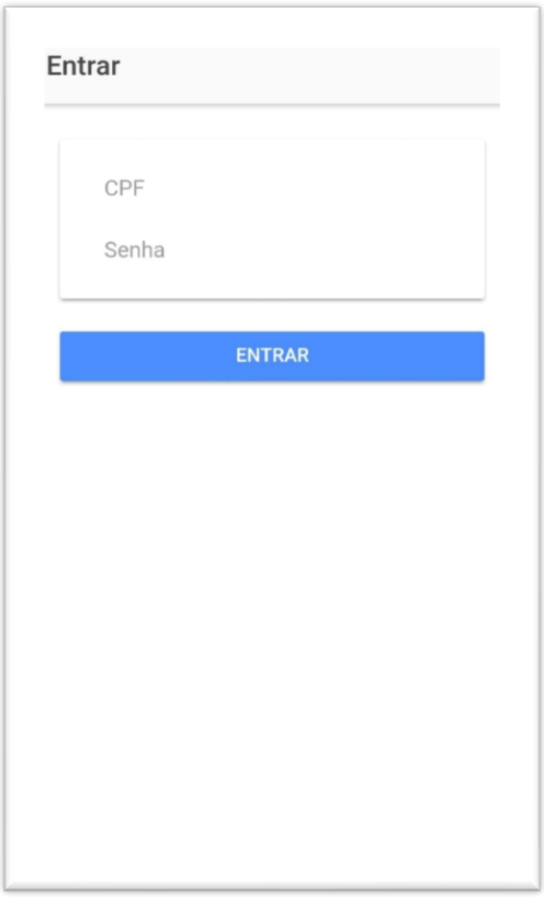
Fonte: Autor

5.3 APLICATIVO

Como o aplicativo foi definido como responsável por realizar os processos de empréstimos e devoluções de equipamentos, seu uso ficou restrito apenas aos professores. Dessa forma, a interface inicial do aplicativo também é uma interface de acesso, que contém

um formulário para preenchimento do CPF e senha do professor. A Figura 31 demonstra essa interface de acesso.

Figura 31 - Painel de acesso do aplicativo

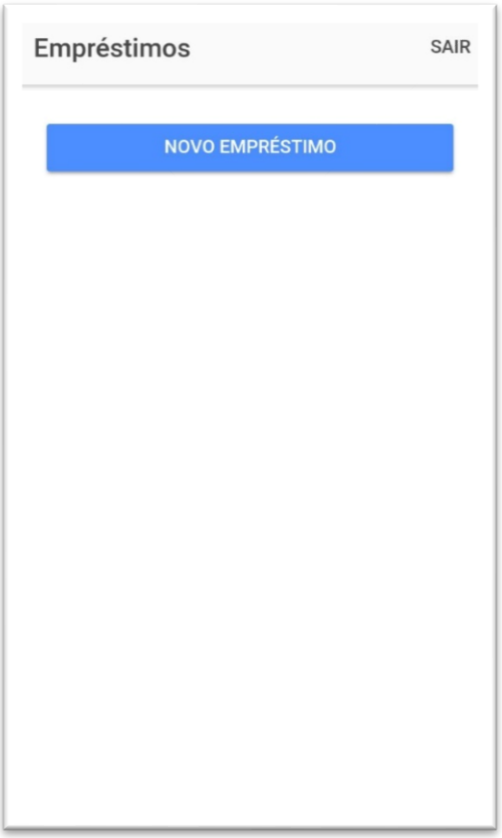


Fonte: Autor

Após o preenchimento do formulário de acesso ao aplicativo, o professor será redirecionado à página de empréstimos. Nela estarão listados os empréstimos em aberto (aqueles em que o equipamento já foi emprestado para o aluno e está em uso, ou seja, ainda não foi realizado o processo de devolução do equipamento), onde não consta no banco de dados um valor para o atributo “dataDevolucao”. Além da listagem dos empréstimos, também consta um campo para iniciar um novo empréstimo.

A Figura 32 demonstra a página “Empréstimos” ainda sem um empréstimo em aberto.

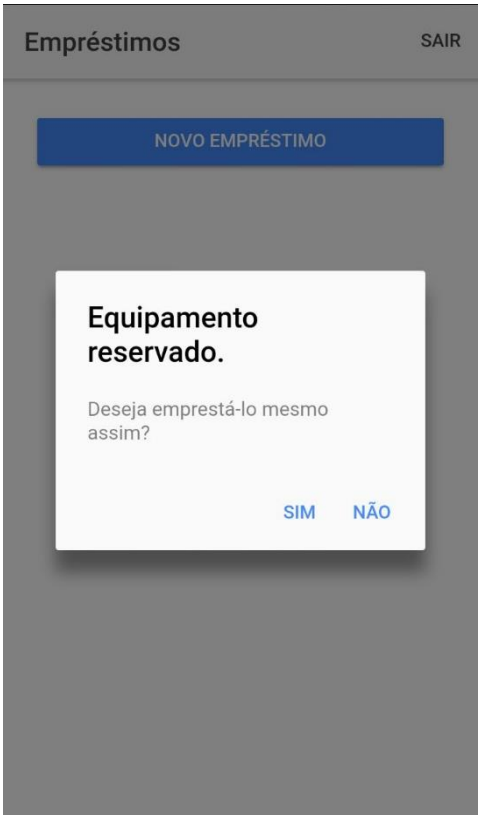
Figura 32 - Página "Empréstimos" do aplicativo



Fonte: Autor

Caso seja chamada a função de criar um novo empréstimo, a câmera do dispositivo é aberta para a realização da leitura da etiqueta do equipamento. A informação é validada pela API e buscada no banco de dados. Caso o equipamento lido contenha uma reserva efetuada para o mesmo dia do empréstimo, uma mensagem será mostrada questionando se o professor deseja prosseguir com o empréstimo. A Figura 33 demonstra esta situação.

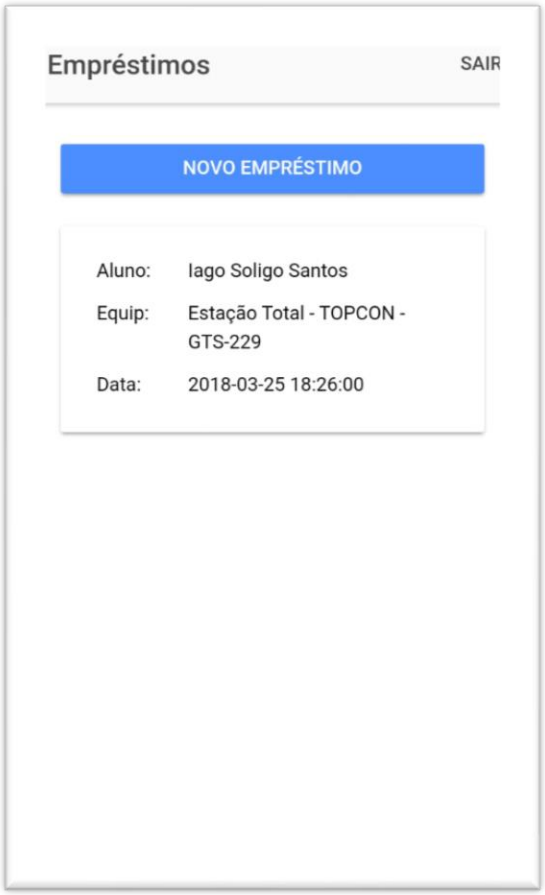
Figura 33 - Aviso de equipamento já reservado no aplicativo



Fonte: Autor

Caso o professor confirme que o aluno é o mesmo para o qual o equipamento foi reservado, ele poderá prosseguir com o processo de empréstimo, onde será aberta a câmera do dispositivo novamente para a realização da leitura da etiqueta do aluno e os dados lidos enviados para o banco de dados e listados na página “Empréstimos” novamente. A Figura 34 demonstra a situação de um empréstimo realizado e sua listagem na página “Empréstimos”.

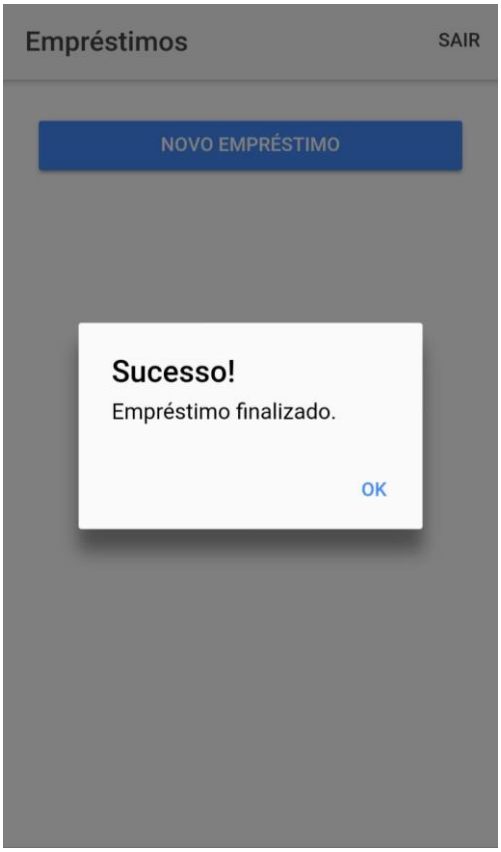
Figura 34 - Listagem do empréstimo em aberto recém-criado



Fonte: Autor

Para finalizar o empréstimo em aberto, no processo de devolução do equipamento, basta que o professor selecione o empréstimo a ser finalizado na lista e a câmera do dispositivo será aberta para a leitura da etiqueta do equipamento, um aviso será mostrado indicando que o empréstimo foi finalizado, atualizando sua linha no banco de dados e adicionando suas horas de uso na página “Equipamentos” da área do professor. A Figura 35 demonstra um empréstimo finalizado.

Figura 35 - Empréstimo finalizado no aplicativo



Fonte: Autor

Nota-se na Figura 35 que, ao finalizar o empréstimo criado e listado na Figura 35, o mesmo não aparece mais na listagem dos empréstimos em aberto, pois o mesmo agora já possui um valor no atributo “Data de Devolução”.

6. CONCLUSÃO

Com a finalização do sistema, pôde-se concluir que é possível realizar o gerenciamento dos empréstimos através de um sistema digital informatizado, diminuindo a ocorrência de erros, aumentando as informações sobre os equipamentos do laboratório e, com isso, fornecendo apoio à tomada de decisões do setor de agrimensura do Campus Inconfidentes.

Também se concluiu que o processo de empréstimo de equipamento, utilizando um sistema informatizado, é mais rápido, eficiente e prático do que com a utilização de fichas de papel para controlar os empréstimos e devoluções dos equipamentos. A utilização de um aplicativo de leitura para etiquetas fixadas nos equipamentos possibilitou que fosse garantido o armazenamento das informações de empréstimos no banco de dados, e a obrigatoriedade de leitura das etiquetas contendo os QRCodes para a realização da devolução do equipamento traz a garantia da devolução do equipamento.

O sistema criado, além de suprir as necessidades para o gerenciamento dos equipamentos do laboratório, também abre espaço para a criação de novas ferramentas e funcionalidades devido à utilização de uma API. Podendo ser expandida sua área de controle, criadas novas páginas, realizados aprimoramentos no sistema atual e até mesmo no aplicativo desenvolvido, sem necessidade de se substituir o sistema atual ou seu banco de dados.

7. REFERÊNCIAS BIBLIOGRÁFICAS

ANDROID STUDIO. **Android Studio:** O IDE oficial do android. Disponível em: <https://developer.android.com/studio/index.html>. Acesso em: 26 mar. 2018

BECKER, A. **HeidiSQL.** Disponível em: <heidisql.com>. Acesso em: 27 mar. 2018.

BIERMAN, G.; ABADI, M.; TORGENSEN, M. Understanding Typescript. In: ECOOP 2014 – OBJECT-ORIENTED PROGRAMMING. ECOOP 2014, 28., 2014, Uppsala, Sweden. **Proceedings.** Springer, Berlin, Heidelberg, 2014. p. 257 - 281. Disponível em: <https://link.springer.com/chapter/10.1007/978-3-662-44202-9_11>. Acesso em: 31 mar. 2018.

BORTOLOSSI, H. J. Criando conteúdos educacionais digitais interativos em matemática e estatística com o uso integrado de tecnologias: GeoGebra, JavaView, HTML, CSS, MathML e JavaScript. In: CONFERÊNCIA LATINO AMERICANA DE GEOGEBRA, 2012. p. 28 - 37.

DALL'OGLIO, P. **PHP:** Programando com Orientação a Objetos. 2. ed. São Paulo: Novatec Editora, 2009.

DATE, C. J. **Introdução a sistemas de bancos de dados.** Rio de Janeiro: Campus, 1984. Tradução de Hélio Auro Gouveia.

DEVMEDIA. **Introdução ao Typescript.** Disponível em: <https://www.devmedia.com.br/introducao-ao-typescript/36729>. Acesso em: 27 mar. 2018.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados.** São Paulo: Pearson Addison Wesley, 2005.

FLANAGAN, D. **JavaScript:** o guia definitivo; tradução: João Eduardo Nóbrega Tortello; revisão técnica: Luciana Nedel. 6 ed. Porto Alegre: Bookman, 2013.

FRANCO, M. **Sistemas de gerenciamento de banco de dados**. São João da Boa Vista: Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, 2013.

GUIMARÃES, C. C. **Fundamentos de banco de dados:** Modelagem, projeto e linguagem SQL. Campinas, SP: Editora da UNICAMP, 2008.

HEUSER, C. A. **Projeto de Banco de Dados.** 4. ed. Porto Alegre: UFRGS, 1998.

IONIC. **Core Concepts:** What is Ionic Framework?. Disponível em: <https://ionicframework.com/docs/intro/concepts/>. Acesso em: 27 mar. 2018.

LARAVEL. **Laravel Documentation.** Disponível em: < https://laravel.com/docs/5.6>. Acesso em 06 mar. 2018.

LECHETA, R. R. **Web Services RESTful:** Aprenda a criar web services RESTful em Java na nuvem do Google. São Paulo: Novatec Editora, 2015.

LUBBERS, P.; ALBERS, B.; SALIM, F. **Programação Profissional em HTML5:** APIs Poderosas para o Desenvolvimento de Aplicações para a Internet com Mais Recursos. Rio de Janeiro: Alta Books, 2013.

MARTINS, P. G.; CAMPOS ALT, P. R. **Administração de Materiais e Recursos Patrimoniais.** São Paulo: Saraiva, 2009.

MICROSOFT. **Getting Started.** Disponível em: <https://code.visualstudio.com/docs>. Acesso em: 27 mar. 2018.

PHP DOCUMENTATION GROUP. **PHP Documentation.** Disponível em: <php.net/manual/pt_BR/>. Acesso em: 27 mar. 2018.

PIRES, J. **O que é API? REST e RESTful? Conheça as definições e diferenças!** Disponível em: <https://becode.com.br/o-que-e-api-rest-e-restful/>. Acesso em: 06 mar. 2018.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Sistema de Banco de Dados.** 3. ed. São Paulo: Pearson Makron Books, 1999.

SILVA, M. S. **Ajax com jQuery:** requisições Ajax com a simplicidade de jQuery. São Paulo: Novatec Editora, 2009.

SILVA, M. S. **Criando sites com HTML:** sites de alta qualidade com HTML e CSS. São Paulo: Novatec Editora, 2008.

SILVA, M. S. **CSS3:** desenvolva aplicações web profissionais com uso dos poderosos recursos de estilização das CSS3; tradução: Rafael Zanolli. São Paulo: Novatec Editora, 2012.

SILVA, M. S. **jQuery:** a biblioteca do programador JavaScript. 2. ed. rev. e ampl. São Paulo: Novatec Editora, 2010.

SPOTO, E. S. **Teste estrutural de programas de aplicação de banco de dados relacional.** 2000. 204 f. Tese (Doutorado) - Curso de Engenharia de Computação, Universidade Estadual de Campinas, Campinas, 2000.

SUÁREZ, H. A.; SILVA, I. **Experiência com o uso do método compacto para calibração de estações.** V Simpósio Brasileiro de Ciências Geodésicas e Tecnologias da Geoinformação, Recife, 2014.

THE JQUERY FOUNDATION. **JQuery API.** Disponível em: <http://api.jquery.com>. Acesso em: 06 mar. 2018.

THE MARIADB FOUNDATION. **About MariaDB**. Disponível em: <https://mariadb.org/about/>. Acesso em: 01 abr. 2018.

VIANA, J. J. **Administração de Materiais**. São Paulo: Atlas, 2000.

VIANA, J. J. **Administração de materiais**, um Enfoque Prático. 1. ed. São Paulo: Editora Atlas, 2002.

W3C. **W3C Recommendation.** Disponível em: <https://www.w3.org/TR/html52/introduction.html#background>. Acesso em: 06 mar. 2018.