Matheus Ramalho Silva

Ramon Rodrigues Pêgo

ESTUDO DA VIABILIDADE DO DESENVOLVIMENTO DE UM SISTEMA WEB PARA CONTROLE DE HORÁRIO EM ACADEMIAS

FACULDADES UNIFICADAS DE TEÓFILO OTONI

TEÓFILO OTONI - MG

2017

Matheus Ramalho Silva

Ramon Rodrigues Pêgo

ESTUDO DA VIABILIDADE DO DESENVOLVIMENTO DE UM SISTEMA WEB PARA CONTROLE DE HORÁRIO EM ACADEMIAS

Monografia apresentada ao curso de Sistemas de Informação das Faculdades Unificadas de Teófilo Otoni como requisito parcial à obtenção do título de Bacharel em Sistemas de Informação.

Área de Concentração: Desenvolvimento Web

Prof. Orientador: Luiz Fernando Alves Souza

FACULDADES UNIFICADAS DE TEÓFILO OTONI

TEÓFILO OTONI-MG

2017

rede de ensino DOCTUM

FACULDADES UNIFICADAS DE TEÓFILO OTONI
NÚCLEO DE TCC / SISTEMAS DE INFORMAÇÃO
Autorizado pela Portaria 4.012 de 06/123/2004 – MEC

FOLHA DE APROVAÇÃO

A monografia intitulada: *Estudo da viabilidade do desenvolvimento de um sistema web para controle de horário em Academias,*

elaborada pelos alunos Matheus Ramalho Silva
Ramon Rodrigues Pego,

foi aprovada por todos os membros da Banca Examinadora e aceita pelo curso de Sistemas de Informação das Faculdades Unificadas de Teófilo Otoni, como requisito parcial da obtenção do título de

BACHAREL EM SISTEMAS DE INFORMAÇÃO.

Teófilo Otoni, 20 de novembro de 2017

Professor Orientador: Luiz Fernando Alves Souza

Professor Examinador: Wilbert Viana Barbosa

Professor Examinador: Salim Ziad Pereira Aouar

Dedico essa conquista a Deus, sempre como guia. A meus pais pela educação e por formarem o homem que sou hoje, pela dedicação, amor e esforço para que eu atinja meus objetivos. (Matheus Ramalho Silva).

A minha família dedico mais esta conquista, pois eles foram, e são ainda minha base para que eu pudesse chegar aqui, e ainda onde poderei chegar. Aos meus amigos, por terem feito essa caminhada ainda melhor, e ao ‘W.P.L.’. (Ramon Rodrigues Pêgo).

AGRADECIMENTOS

Agradecemos a Deus pelos cuidados e pela vida que nos concedeu, por nos dar força e persistência para aproveitar essa oportunidade de crescimento pessoal e profissional.

Aos nossos pais João e Solange, Katia e Renato pela criação que apesar das dificuldades nunca faltou amor, pelos ensinamentos, pela confiança de que nós venceríamos essa etapa. São a base para todos os passos que demos e daremos ainda. No jogo da vida serão sempre os eternos e melhores professores, muito obrigado.

Aos nossos colegas de turma e mais do que amigos Maria Laura Chaves eterna “chefa”, Mayra Nunes a “doida”, Jairo Souza o consagrado “Jairin”, Thiago oliveira “th” e Caio Sander o “indeciso”, nosso muito obrigado pelas risadas, pelas conversas, as brincadeiras, os ensinamentos não só sobre as disciplinas, mas também da vida e principalmente pela confiança e companheirismo, esperamos que essas amizades nos acompanhem sempre independente do tempo e distância.

Aos nossos professores que contribuíram para a formação profissional de cada estudante desta turma de Sistemas de Informação, pessoas de bem e grandes exemplos de profissionais em especial o professor Salim Aouar e nosso orientador Luiz Fernando, nosso muito obrigado pelos ensinamentos, cobranças e incentivos.

Aos nossos amigos e familiares que nos apoiaram nessa caminhada, e diversas vezes se fizeram presentes dando força, ou até em momentos de descontração, amigos esses que são essenciais em nossas vidas, agradecemos.

“Medir o progresso da programação por linhas de códigos é como medir o progresso da construção de aeronaves em peso”.

Bill Gates

ABREVIATURAS E SIGLAS

ANSI - *American National Standards Institute*

CSS - *Cascading Style Sheets*

DER - Diagrama de Entidade e Relacionamento

DNS - *Domain Name System*

DOM - *Document Object Model*

HTML - *HyperText Markup Language*

HTTP - *Hypertext Transfer Protocol*

IDE - *Integrated Development Environment*

ISO - *International Standsrds Organization*

MVC - *Model View Controller*

NoSQL - *Not only SQL*

PHP/FI - *Personal Home Page/Forms Interpreter*

PHP - *PHP Hypertet Preprocessor*

RF - Requisitos Funcionais

RNF - Requisitos Não Funcionais

SGBD - Sistema Gerenciador de Banco de Dados

SO - Sistema Operacional

SSH - *Secure Shell*

SQL - *Structured English Query Language*

UML - *Unified Modeling Language*

WebE - Engenharia Web

LISTA DE FIGURAS

Figura 1 - Tecnologia em camadas. 17
Figura 2: Diagrama de caso de uso preliminar para o sistema CasaSegura 22
Figura 3: Modelo Conceitual 30
Figura 4: Modelo Lógico. 31
Figura 5: Modelo Físico 31
Figura 6: Popularidade das bibliotecas JavaScript 36
Figura 7: Reprodução de um formulário em navegadores diferentes 37
Figura 8: Conexão via SSH pelo PHPStrom 45
Figura 9: Diagrama de caso de uso cadastro de horário 48
Figura 10: Tabela aluno 51
Figura 11: Tabela instrutor 52
Figura 12: Tabela modalidade 52
Figura 13: Tabela aula 53
Figura 14: Arquivo env 55
Figura 15: Tela de login 65
Figura 16: Tela de Registro 66
Figura 17: Tela Home do sistema 67
Figura 18: Tela de Cadastro de aluno responsiva 68
Figura 19: Tela de Cadastro de aula 69
Figura 20: Tela de Matricula 69
Figura 21: Cadastrando uma aula 70
Figura 22: Cadastrando aluno na aula 71
Figura 23: Exibição de horário 72

LISTA DE CÓDIGOS

Código 1 - Migration 50
Código 2: Trecho onde é chamado as demais views para a view app 56
Código 3: View 57
Código 4: Routes 58
Código 5: Controller 59
Código 6: Model 60
Código 7: Função para preencher endereço através do cep 61
Código 8: Função validação 62

RESUMO

Esta monografia tem como objetivo desenvolvimento de um sistema web para controle de horários, para auxiliar o usuário e gerente de uma empresa provedora de atividades físicas no ramo de academias. Sendo construída sobre o tema “Estudo da viabilidade do desenvolvimento de um sistema web para controle de horário em academias”. Apesar de a área de concentração estar situada no desenvolvimento web, trata-se também de um estudo teórico, onde material bibliográfico consagrado foi levantado sobre disciplinas da ciência da tecnologia como Engenharia de Software, Banco de Dados e linguagens de programação com objetivo de embasar o conhecimento e fundamentar a estruturação de um projeto de desenvolvimento. Portanto, o presente trabalho de conclusão de curso tem como foco possibilitar a solução dos problemas e atender as necessidades de uma empresa do ramo de academias em relação ao controle de horários das aulas ofertadas pela mesma, através do desenvolvimento de um sistema web capaz de modelar as especificidades do ambiente da empresa.

Palavras-Chave: desenvolvimento, web, sistema, Laravel, responsividade.

SUMÁRIO

INTRODUÇÃO 13

1 REFERENCIAL TEÓRICO 16

1.1 Software 16

1.2 Engenharia de Software 16

1.2.1 Processo de Software 18

1.2.1.1 Modelo de Processo de Software 18

1.2.2 Etapas do Desenvolvimento 19

1.2.3 Requisitos 19

1.2.4 Linguagem de Modelagem Unificada (UML) 20

1.2.4.1 Caso de Uso 21

1.3 Engenharia Web 24

1.3.1 Camadas da Engenharia da Web 24

1.4 Banco de Dados 25

1.4.1 Sistema Gerenciador de Banco de Dados (SGBD) 26

1.4.1.1 Arquitetura do SGBD 27

1.4.1.2 Organização do SGBD 28

1.4.2 Projeto de Banco de dados 29

1.4.3 SQL 32

1.5 Front-end 32

1.5.1 Hypertext Markup Language (HTML) 32

1.5.1.1 HTML5 33
1.5.2 JavaScript 33
1.5.2.1 JQuery 35
1.5.3 Cascading Style Sheets (CSS) 36
1.6 Back-end 37
1.6.1 PHP Hypertext Preprocessor (PHP) 37
1.6.2 Servidor Web 39
1.7 Frameworks 39
1.7.1 Laravel 40
1.7.2 Bootstrap 40
2 MATERIAIS E MÉTODOS 42
2.1 Sistema Operacional 42
2.2 XAMPP 42
2.3 Composer 43
2.4 PHPStorm 43
2.5 Laravel 43
2.6 Vagrant 44
3 DESENVOLVIMENTO 46
3.1 Desenvolvimento do sistema 46
3.1.1 Requisitos 46
3.1.2 Banco de Dados 49
3.1.3 Sistema Web 53
3.1.4 Testes do sistema 70
4 RESULTADOS OBTIDOS 73
CONSIDERAÇÕES FINAIS 75
REFERÊNCIAS 80
APÊNDICE I 84

INTRODUÇÃO

A presente monografia, concentrada na área do desenvolvimento web, tem como objetivo o desenvolvimento de um sistema web para substituir e melhorar os processos das academias, a mesma foi intitulada de: “Estudo da viabilidade do desenvolvimento de um sistema web para controle de horários em academias”, otimizando e facilitando a organização de determinadas atividades na empresa.

As empresas de diversos seguimentos atualmente necessitam acompanhar o desenvolvimento tecnológico para atualizar e melhorar seus serviços internos e serviços ou produtos prestados ao público. Uma forma viável de se encaixar nesse quadro é a melhora dos equipamentos e processos que uma empresa realiza, nesse sentido, empresas do ramo de atividades esportivas devem acompanhar essas mudanças e substituir o sistema antigo por um novo sistema que fosse atrativo e eficiente.

As informações atualmente são o bem mais precioso que uma empresa pode ter, diante desse quesito. Portanto, a presente monografia parte da seguinte questão investigativa: Seria viável o desenvolvimento de um sistema web para controle de horários em academias?

Sobre os presentes fatos o projeto tem como objetivo possibilitar a solução dos problemas e atender as necessidades das academias em relação ao controle de horários das aulas ofertadas pela mesma, através do desenvolvimento de um sistema web capaz de modelar as especificidades do ambiente da empresa.

É valido ressaltar que a comunicação entre usuário e sistema é extremamente importante, do mesmo modo que a relação entre a empresa e o cliente seja clara, objetiva, correta e útil. Portanto uma empresa se apoiar na utilização de um software atualizado com as normas e regras do mercado e que atenda exatamente ao tipo de serviço prestado é de grande relevância.

Durante o decorrer da pesquisa foram levantadas hipóteses com intuito de

obter a resposta para a questão investigativa, sendo elas:

Hipótese 0: Não seria viável o desenvolvimento deste sistema, pois a gestão da empresa por uso de planilhas do Microsoft Excel viria logrando o êxito esperado pela gestora.

Hipótese 1: O desenvolvimento do sistema não seria viável, pois não traria maior segurança para os dados da empresa do que uma planilha feita no Microsoft Excel.

Hipótese 2: A implantação do sistema seria viável para que a empresa consiga se organizar melhor, contemplando uma administração de qualidade e atualizada.

Hipótese 3: O desenvolvimento e implantação da aplicação web não seria viável, pois a empresa não conseguiria se manter no mercado, que está cada vez menos interessado no nicho de mercado em questão.

Com foco em atingir o objetivo geral da pesquisa, foram desenvolvidos objetivos específicos, sendo eles:

- Compreender as regras de negócio para o levantamento dos requisitos sobre as funcionalidades que devem ser contempladas pelo software.
- Garantir a plena adequação do sistema às necessidades reais da empresa em questão.
- Através de análise, garantir que o software sucessor contemple as necessidades mínimas do antecessor.
- Através da análise das funcionalidades do software anterior, compreender as falhas que o tornaram inviável para empresa, evitando assim que o novo software possua tais erros.
- Compreender as vantagens de se possuir um sistema web e garantir o entendimento do cliente sobre os benefícios que o mesmo trará a empresa.
- Garantir que o sistema possua um local de hospedagem seguro e de qualidade, além de um domínio, evitando eventuais problemas relacionados a problemas de acesso e funcionamento do sistema na web.
- Aplicação de um ambiente de testes para que seja descoberto falhas e erros nas partes já desenvolvidas do software e corrigi-las antes de chegar até as mãos do usuário final.
- Desenvolver o software de forma iterativa e incremental, sempre em contato constante com o cliente, garantindo que ele compreenda as funcionalidades

desenvolvidas e garantir que estão atendendo a sua necessidade.

- Garantir que a gestora e os demais usuários aprendam a utilizar o sistema corretamente através de treinamento, evitando utilização errada do software que possa vir a torna-lo inviável para empresa.
- Analisar e compreender o feedback do cliente sobre a apresentação/teste do sistema completo e corrigir pontos solicitados pelo mesmo.
- Validar hipóteses para que o sistema web tenha tido êxito no desenvolvimento e garanta que a implantação e utilização do mesmo na empresa traga benefícios.

Quanto aos fins, esta pesquisa tem caráter descritiva e laboratorial, uma vez que se baseia na análise do método já utilizado pela gerência da empresa em questão e para que o sistema se torne viável se fez necessário a pesquisa e estudo de linguagens, técnicas e ferramentas que serviram como base e guia para o decorrer do projeto, buscando mostrar que o sistema web é de grande relevância para a solução do problema apresentado.

A pesquisa se encaixa no modelo de pesquisa bibliografia sendo que foi necessário o levantamento, estudo e análise de bibliografias consagradas no mercado de tecnologia e desenvolvimento, com objetivo de que o sistema fosse construído baseado em boas práticas. Também inclui-se no tipo documental devido à utilização de material audiovisual para complemento dos estudos.

Esta pesquisa foi dívida em quatro capítulos sendo o primeiro intitulado de referencial teórico, onde é abordado teoricamente as práticas, métodos e filosofias utilizadas para se realizar um desenvolvimento de software de qualidade, alguns desses conceitos são Banco de dados, Engenharia de software e linguagens de programação.

O segundo capítulo é de materiais e métodos, encabeçado por explicar quais foram os softwares e frameworks utilizados para realizar o desenvolvimento do sistema web, ou seja, é um capítulo sobre a instalação e configuração do ambiente.

O terceiro capítulo possui título de Desenvolvimento, é responsável pelas informações sobre o desenvolvimento do sistema web, como análise dos requisitos, criação do banco de dados, codificação do sistema e amostra das telas do mesmo.

E por último o quarto capítulo, intitulado de resultados obtidos, trazendo os dados e informações pelos resultados obtidos com essa pesquisa e projeto.

1 REFERENCIAL TEÓRICO

1.1 Software

Para Pressman (2011, p.32) software são “instruções (programas de computador) que quando executadas fornecem características, funções e desempenhos desejados”.

A maioria das pessoas acredita que a palavra software está delimitada apenas a ser um programa de computador, no entanto é uma visão restritiva do intuito da palavra, assim afirma Sommerville (2007, p.4) que diz “Software não é apenas um programa, mas também todos os dados de documentação e configuração associados, necessários para que o programa opere corretamente”.

Na década de 70 menos de 1% da população sabia descrever o significado de software de computadores (PRESSMAN, 2006, p.4). Atualmente, apesar do avanço tecnológico, pessoas de diversas áreas e até do público leigo pensam saber o significado de software, mas na verdade desconhecem o significado real, pois, misturam software e hardware em uma definição só. No entanto “software é um elemento de um sistema lógico e não físico, assim ele possui características que são consideravelmente diferentes daquelas do hardware” (PRESSMAN, 2006, p.4).

Como a deterioração do hardware não está ligada a vida do software, já que o mesmo não sofre influência do ambiente como acontece com hardware, a vida útil de um software está ligada basicamente a constante evolução da tecnologia, ou seja, ele depende de atualizações para continuar vivo e funcional. (PRESSMAN, 2011, p.32).

1.2 Engenharia de Software

A Engenharia de Software é uma disciplina ligada a produção de software, mas não somente a isso, está ligada desde a fase inicial do projeto até sua finalização, ou seja, é responsável por todo o projeto de produção de um software. Junior (apud BAUER, p.9) afirma que “o estabelecimento e uso de sólidos princípios de engenharia para que se possa obter economicamente um software que funcione efetivamente com maquinas reais”. Conduzir um projeto de software utilizando a Engenharia de Software é uma prática recomendada para que o mesmo seja finalizado com maior êxito. Junior retrata que

> A engenharia de software é uma derivação da Engenharia de Sistemas e de Hardware. Ela abrange um conjunto de três elementos fundamentais – métodos, ferramentas e procedimentos – que possibilita ao gerente o controle do processo de desenvolvimento do software e oferece ao profissional uma base para a construção de software de alta qualidade produtivamente. (JUNIOR, apud BAUER, p.9).

A Engenharia de Software tem o objetivo de auxiliar todo o processo de construção de um projeto de software, atuando nas áreas de planejamento e execução do mesmo, para que isso se torne possível e eficiente, trabalha com uma tecnologia em camadas, conforme pensa Pressman.

> A Engenharia de Software é uma ciência para estudo e produção de software que se baseia em uma tecnologia em camadas, qualquer abordagem de engenharia deve se apoiar num compromisso organizacional com qualidade que ajudam a dar maior segurança e qualidade ao processo utilizado como base para se obter um produto de qualidade e eficiência (PRESSMAN, 2006, p.17).

Figura 1 - Tecnologia em camadas.

Fonte: PRESSMAN, 2006, p.39

As camadas da Engenharia de Software são quatro, ferramentas, métodos, processo e foco na qualidade, cada uma responsável por completar a outra e levam a uma mais alta qualidade em todo o processo. Pressman as define:

> O processo de engenharia de software é a liga que mantém as camadas de tecnologia coesas e possibilita o desenvolvimento de software de forma racional e dentro do prazo, já os métodos fornecem as informações técnicas para desenvolver o software, enquanto as ferramentas fornecem suporte automatizado ou semi-automatizado para o processo e para os métodos. (PRESSMAN, 2011, p.39-40).

1.2.1 Processo de Software

O processo de software é uma maneira da equipe de desenvolvimento conduzir e validar a produção de software seguindo algumas etapas que auxiliarão na condução do projeto, com objetivo de se obter um produto. Pressman (2011, p.40) afirma que “o processo de software é um conjunto de atividades, ações e tarefas realizadas na criação de algum produto de trabalho (*work product*).”

O processo de software é constituído e dividido pelas seguintes etapas: especificação, desenvolvimento, validação e evolução do software, para obter-se um produto de software é necessário passar por todas as etapas, Sommerville as define:

> A etapa de especificação do software é onde clientes e engenheiros definem o software a ser produzido e as restrições para a sua operação, prosseguindo vem a etapa de desenvolvimento do software onde o mesmo é projetado e programado, após a conclusão da codificação é dado início a etapa de validação de software onde é realizada uma verificação para garantir que é o que o cliente deseja, e por fim a etapa de evolução de software onde são feitas modificações para que se adapte as mudanças dos requisitos do cliente e do mercado. (SOMMERVILLE, 2007, p.6).

1.2.1.1 *Modelo de Processo de Software*

Os modelos de processo foram propostos inicialmente para acabar com a desordem no desenvolvimento de software, eles são responsáveis por trazer uma estruturação para o projeto de desenvolvimento, como um guia de atividades para o projeto. (PRESSMAN, 2006, p.37).

A maioria dos modelos de processo de software é baseada nos modelos ou paradigmas de desenvolvimento de software, Cascata e Incremental

(SOMMERVILLE, 2007, p.7).

O Modelo em Cascata ou ciclo de vida clássico é um modelo mais rígido e recomendado quando os requisitos estão muito bem compreendidos e muito dificilmente sofrerão alguma alteração. Pressman (2006, p.39) sugere para este modelo uma abordagem sistemática e sequencial que é composta pelas etapas de Comunicação, Planejamento, Modelagem, Construção e Implantação.

O Modelo Incremental é comumente recomendado para projetos em que os requisitos de software não estão claramente compreendidos, e/ou que o cliente necessita de ao menos partes do sistema funcionando, assim o software é desenvolvido aos poucos. Pressman (2006, p.40) descreve que “o modelo incremental combina elementos do modelo em cascata aplicado de maneira iterativa”.

1.2.2 Etapas do Desenvolvimento

O desenvolvimento de software não é composto apenas pela codificação do sistema, mas sim por todas as etapas que comportam todo o projeto, desde a definição e/ou identificação do problema a ser resolvido, até a entrega final do software completo para o cliente.

O desenvolvimento de software é dividido em três etapas: fase de definição, desenvolvimento e fase de verificação, liberação e manutenção.

A fase de definição é a etapa de planejamento do software, é descrito o escopo e feitas estimativas de prazo e custo, análise de requisitos. A fase de desenvolvimento comporta a descrição da estrutura modular, definição de interfaces, estrutura de dados estabelecidas e a codificação (programação do software). Na fase final é realizada a verificação, liberação e manutenção do software, onde os testes são executados a fim de encontrar o máximo de erros possíveis antes da liberação do software, também é realizada a manutenção durante sua vida útil. (JUNIOR, apud PRESSMAN, 1995, p.1088).

1.2.3 Requisitos

Os requisitos de um sistema são funcionalidades, práticas, funções que o software deverá possuir para satisfazer a necessidade de sua construção, ou seja, as operações que o software deve realizar para atingir o propósito de sua criação. Esses

requisitos são descrições dos serviços fornecidos pelo sistema e as suas restrições operacionais (SOMMERVILLE, 2007, p.79).

Os requisitos de sistemas de software são classificados frequentemente como requisitos funcionais e não funcionais.

> Requisitos funcionais: São declarações de serviços que o sistema deve fornecer, como o sistema deve reagir a entradas periféricas específicas e como o sistema deve comportar em determinadas situações. Em alguns casos os requisitos funcionais podem especificar o que o sistema não deve fazer.
>
> Requisitos não funcionais: são restrições sobre os serviços ou as funções oferecidas pelo sistema. Eles incluem restrições de *timing*, restrições sobre o processo de desenvolvimento e padrões. (SOMMERVILLE, 2007, p.80).

1.2.4 Linguagem de Modelagem Unificada (UML)

UML é uma linguagem de modelagem utilizada no desenvolvimento de *softwares* para documentar e visualizar objetos que são especificados e construídos na análise e no desenho de um sistema, sendo uma sintaxe para criação de um modelo lógico. Frequentemente, utiliza-se a UML para descrever um sistema de um computador da maneira como ele é entendido nas várias posições as quais ele é desenvolvido durante análise e desenho.

A sintaxe da UML foi proposta originalmente e definida por Jim Rumbaugh e Grady Booch para possuir construções disponíveis em cada método com representações gráficas comuns. Sucessivamente, Ivair Jacobson acrescentou a sintaxe para definição de requisitos com casos de uso e logo após a linguagem foi concluída por um comitê de especialistas em linguagem orientada a objetos.

A sintaxe de UML tem sido projetada para ser independente de qualquer linguagem-alvo, processo de desenvolvimento de software ou ferramenta, no entanto, é suficientemente genérica e flexível a ponto de poder ser utilizada de uma forma personalizada utilizando extensões definidas pelo próprio usuário para organizar praticamente qualquer linguagem, ferramenta ou requisito de processo.

A Linguagem Unificada de Modelagem (UML), une as melhores práticas de engenharia para fins de modelagem de sistemas, produzindo um conjunto de artefatos

que podem ser disponibilizados no mercado e possui alcance mundial. Seus objetivos podem ser estabelecidos como: (1) Linguagem de modelagem visual que seja simples e extensível; (2) possuir independência de qualquer linguagem de programação; (3) independência de processo; (4) suporte a conceitos em alto nível como estrutura, padrões e componentes; (5) possuir flexibilidade e ser amplamente aplicável.

A UML produz produtos e serviços que focam em demandas e requisitos para seus clientes e constitui-se o mecanismo preferido da indústria de *software*. Requisitos de *software* é o problema e os produtos e serviços a solução deste. O problema e a solução acontecem dentro de algum domínio. Para que uma boa solução seja gerada primeiramente deve-se entender o problema. A solução deve ser ajustada e planejada a fim de facilitar a consecução e aderir às restrições de domínio. Dessa maneira, para resolução de problemas, os apropriados conhecimentos do problema e/ou solução precisam ser capturados(modelados), organizados (arquitetura) e representados (diagrama) apropriando-se de mecanismos que permitam a comunicação e alavancagem do conhecimento.

A UML possui uma série de diagramas que podem ser utilizados para análise e projeto em nível de sistema como de *software.* Os diagramas presentes na UML são: (1) Diagramas de casos de uso; (2) Diagramas de estrutura estática, (3) Diagramas de objeto, (4) Diagramas de Classe, (5) Diagramas de Interação, (6) Diagramas de sequência, (7) Diagramas de colaboração, (8) Diagramas de estado, (9) Diagramas de atividade, (10) Diagramas de implementação, (11) Diagramas de componentes e (12) Diagramas de implantação. Porém, os mais importantes presentes na sintaxe UML são os diagramas de caso de uso, diagramas de classe, diagramas de interação e diagramas de estado. (LEE; TEPFENHART,2002, p.505-507).

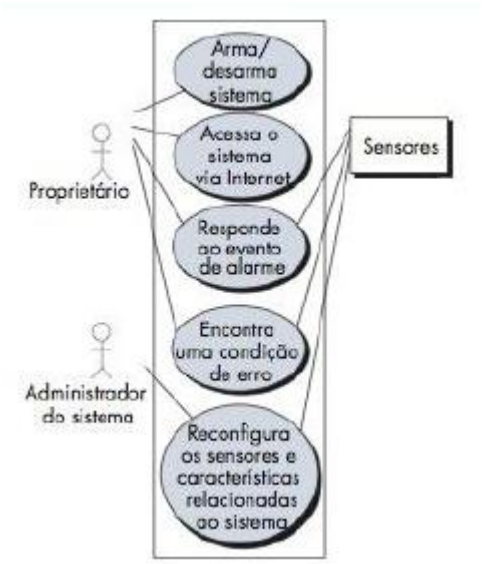
1.2.4.1 *Caso de Uso*

É um diagrama que tem como objetivo dar auxílio a comunicação entre o analista e o cliente. Este tipo de diagrama é responsável por descrever o cenário que mostre funcionalidades do sistema do ponto de vista do usuário. O caso de uso deve deixar claro para o cliente as principais funcionalidades do sistema descrito no

mesmo.

O diagrama é representado por 3 itens, sendo eles atores, casos de uso e relacionamento. Os atores são o usuário do sistema, enquanto os casos de usos são as funções do sistema e são representados por uma elipse e por fim os relacionamentos que ajudam a descrever o caso de uso e são representados por setas e são divididos em três tipos diferentes, o primeiro é associação entre ator e caso de uso, a segunda é a relação entre atores, em terceiro a relação entre casos de uso que possui os tipos include, *extend* e generalização ou especialização.[1]

Figura 2: Diagrama de caso de uso preliminar para o sistema CasaSegura



Fonte: PRESSMAN, 2006, p.157

Caso de uso: IniciarMonitoração.

Ator principal: Proprietário.

Meta de Contexto: Iniciar o sistema para monitoramento dos sensores quando o proprietário estiver em casa ou sair.

Precondições: Sistema programado para uma senha e reconhecer vários

[1] http://www.dsc.ufcg.edu.br/~sampaio/cursos/2007.1/Graduacao/SI-II/Uml/diagramas/usecases/usecases.htm

sensores.

Disparo: O proprietário quem decide ligar as funções de alarme.

Cenário:

1. Proprietário: observa o painel de controle.
2. Proprietário: insere a senha
3. Proprietário: escolhe “dentro” ou “fora” de casa.
4. Proprietário: constata se a luz vermelha do alarme que indica que o CasaSegura foi ligado e está em operação.

Exceções:

1. Painel de controle indica não disponível: o proprietário analisa todos os sensores.
2. Senha inserida incorretamente: proprietário introduz a senha correta;
3. Senha não reconhecida: o subsistema de monitoração e resposta deve ser contatado para reprogramar a senha;
4. “Dentro” é selecionado: painel de controle toca duas vezes e a luz “dentro” acende, neste momento, os sensores periféricos são ativados.
5. “Fora” é selecionado: o painel de controle toca três vezes e a luz “fora” acende ativando os sensores periféricos.

Prioridade: Essencial, deve ser implementada.

Quando disponível: Primeiro incremento.

Frequência de utilização: inúmeras vezes ao dia.

Canal para o ator: interface do painel de controle.

Atores secundários: técnico de apoio, sensores.

Canais para atores secundários:

- Técnico de apoio: linha telefônica.
- Sensores: interfaces em *hardwares* e sem fio.

É importante ressaltar que cada caso de uso deve ser examinado, conferido e visto com cuidado, para que não haja elemento ambíguo na iteração.

1.3 **Engenharia Web**

A *World Wide Web* e a Internet são os dois desenvolvimentos mais importantes da história da computação, afirma Pressman (2006, p.378). Os sistemas e aplicações baseados na web são conhecidos como WebApps, a engenharia da web é um processo utilizado para a criação desses WebApps com uma alta qualidade.

A Engenharia Web (WebE) não é a mesma coisa que a Engenharia de Software, são muito parecidas, pois, utilizam muitos dos conceitos e princípios fundamentais da engenharia de software.

> A WebE aplica uma abordagem genérica que é combinada com estratégias, táticas e métodos especializados. O processo de WebE começa com uma formulação do problema a ser resolvido pela WebApp. O projeto WebE é planejado e os requisitos e o projeto da WebApp são modelados. O sistema é construído usando tecnologias e ferramentas especializadas associadas com a web. (PRESSMAN, 2006, p.378).

Os sistemas desenvolvidos na web, os WebApps são considerados diferentes de um desenvolvimento de software tradicional. Pressman (2006, p.378) afirma que estes sistemas “envolvem uma mistura de publicação impressa e desenvolvimento de software, de comercialização e computação, de comunicações internas e relações externas, e de arte e tecnologia”.

1.3.1 Camadas da Engenharia da Web

Assim como o desenvolvimento de um software utilizando Engenharia de Software como base e guia, o desenvolvimento web segue pelo mesmo caminho, utilizando também modelos de processos especializados, métodos de engenharia de software adaptados às características do desenvolvimento web.

O desenvolvimento web é dividido em camadas, são três: os processos, os métodos e ferramentas e tecnologia. Pressman (2006, p.381) define que processos

utilizam a filosofia de desenvolvimento simples que incorpora ciclos rápidos de desenvolvimento, como acontece no modelo iterativo e incremental. Já os métodos são compostos por um conjunto de tarefas técnicas que habilitam o engenheiro web a entender, caracterizar e então construir uma WebApp de alta qualidade, esses métodos são divididos em:

> Métodos de comunicação que são responsáveis por facilitar a comunicação entre engenheiros e todos os outros interessados no projeto. Métodos de análise de requisitos que fornecem a base para o conteúdo a ser entregue pelo sistema, métodos de projeto que são técnicas de projeto que cuidam do conteúdo, arquitetura da aplicação e da informação, projeto da interface e estrutura da navegação. Métodos de teste responsáveis por revisões técnicas formais do conteúdo e modelo de projeto e uma ampla variedade de técnicas de teste. (PRESSMAN, 2006, p.381).

E por último as ferramentas e tecnologia utilizadas no auxílio do desenvolvimento web, como linguagens de programação, navegadores, sites, linguagens de marcação.

1.4 **Banco de Dados**

Antigamente os sistemas utilizados eram primitivos e não possuíam conexão uns com os outros, era utilizado um sistema para cada tarefa, além de que trabalhavam com sistemas de arquivos, como afirma Cayres (2015, p.1).

Com a evolução da tecnologia, para que uma aplicação possa ser eficiente para finalidade a qual foi desenvolvida, é necessário que a mesma já não necessite de um sistema de arquivos para funcionar, onde é suprida por um banco de dados estruturado ou não estruturado. Elmasri e Navathe (2005, p.4) definem um banco de dados como “uma coleção de dados relacionados. Os dados são fatos que podem ser gravados e que possuem um significado implícito”.

Apesar de “banco de dados” ser uma expressão técnica, a maioria das pessoas atualmente tem contato direto com ele, devido à grande quantidade de equipamentos cuja função é armazenar informações. Afirma Carvalho (2015, p.3) “um local no qual é possível armazenar informações para consulta ou utilização, quando necessário”.

Como descrito anteriormente, os softwares trabalhavam com arquivos físicos como base, que necessitavam de vários usuários e maior tempo gasto para resgatar

informação. Atualmente os arquivos físicos foram substituídos por bancos de dados informatizados e para que os sistemas de bancos de dados apresentem máxima eficiência, alguns objetivos devem ser alcançados, conforme Cayres (2015, p.1)

- Gerenciamento de grande quantidade de informação: Um sistema de banco de dados teria de possibilitar o armazenamento de informações tanto de sistemas simples, quanto de sistemas complexos.
- Evitar redundância e inconsistência de dados: Um Sistema de banco de dados teria de ter capacidade de reduzir ao máximo, ou mesmo eliminar, a redundância da informação em lugares diferentes.
- Facilidade de acesso: Um sistema de banco de dados deveria facilitar ao máximo o acesso aos dados, se preocupando com o possível acesso concorrente.
- Segurança de dados: O sistema de banco de dados deveria garantir a segurança de acesso aos dados por meio da implementação de usuários e senhas de acessos.
- Garantia de Integridade: Fazer com que os valores dos dados atribuídos e armazenados em um banco de dados devam satisfazer certas restrições para manutenção de consistência e concorrência.
- Facilidade de migração: O sistema de banco de dados deveria garantir a possível transferência de dados entre banco de dados.

1.4.1 Sistema Gerenciador de Banco de Dados (SGBD)

Para que seja possível o manuseio de um banco de dados se faz necessário a utilização de um Sistema Gerenciador de Banco de Dados (SGBD), que surgiram no final de 1960. Os SGBDs são responsáveis por executar funções extremamente necessárias para o banco, como segurança e integridade. O “SGBD é um software que incorpora e facilita as funções de definição, recuperação e alteração de dados em um banco de dados”. (CAYRES, 2015, p.3).

Atualmente existem diversas soluções de SGBD no mercado de banco de dados, como MySQL, PostgreSQL, Microsoft SQL Server, MongoDB, Cassandra,

entre outros. Diversos desses produtos são mundialmente conhecidos e tem sua licença de utilização proprietária, porém existem alternativas *open source* (código aberto) como os SGBDs OrientDB, Cassandra, Couchbase e MariaDB.

O MariaDB foi desenvolvido por David Axmark, Allan Larsson e Michael “Monty” Widenius, uma opção *open source* baseada no SGBD MySQL, que atualmente é mantida pela comunidade MariaDB com a *Monty Program Ab* como principal administrador. A maioria dos componentes funciona exatamente da mesma forma que o MySQL, interfaces, comandos, bibliotecas e APIs[2].

Para acesso ao sistema gerenciador de banco de dados são utilizados softwares com interface gráfica amigável, com objetivo de facilitar e agilizar o manuseio do banco de dados, existem diversos softwares que possibilitam esse uso, como o PHPMyAdmin que possibilita ao usuário de banco de dados a trabalhar através de uma interface gráfica, de forma mais simples, enquanto outros proporcionam uma experiência mais completa, como no caso do MySQL Workbench.

1.4.1.1 *Arquitetura do SGBD*

A arquitetura de SGBD varia de uso da sua aplicação, podendo ser mais de uma, ou um pouco de cada, chamada de híbrida. As primeiras arquiteturas eram centralizadas, isto é, um computador com uma grande capacidade de processamento que fará o processamento para os clientes desse SGBD.

Ao passar do tempo novas arquiteturas de SGBDs foram desenvolvidas, como cliente-servidor, onde são conectadas várias estações de trabalho juntas em uma rede, e a definição de servidores especializados onde cada um tem uma utilização específica, como servidor de DNS-*Domain Name System* (Sistema de Nomes e Domínios), servidor de impressoras ou servidores de arquivos. (SANCHES, 2005).

A arquitetura dos SGBDs tem como função separar a aplicação dos usuários dos dados físicos armazenados. São divididos em três esquemas: nível interno, que usa um modelo que mostra a estrutura do armazenamento físico dos dados; nível conceitual, que oferece uma descrição total da estrutura do banco, mas sem detalhes dos dados; e de nível externo onde descreve as visões do banco de dados para um grupo de

[2] https://mariadb.com/kb/pt-br/sobre-o-mariadb/

usuários que acessarão este banco. (CARVALHO, s/d, p.29).

1.4.1.2 *Organização do SGBD*

Devido ao desenvolvimento de hardware e software, abriu-se um caminho para que os SGBDs pudessem ser divididos em modelos para sua organização, atendendo a diferentes tipos de dados e sistemas. Foram desenvolvidos seis modelos diferentes de organização de SGBDS, Modelo hierárquico, modelo de rede, modelo relacional, modelo orientado a objetos, modelo objeto-relacional e modelo NoSQL.

O Modelo Hierárquico foi o primeiro modelo a ser reconhecido como um modelo de dados.

> Nesse modelo de dados, os dados são estruturados em hierarquias ou árvores. Os nós das hierarquias contem ocorrências de registro, onde cada registro é uma coleção de campos (atributos), cada um contendo apenas uma informação. (CAYRES, 2005, p.4-5).

O Modelo em Rede foi desenvolvido como uma extensão do modelo hierárquico, eliminando o conceito de hierarquia e permitindo que um mesmo registro estivesse envolvido em várias associações, segundo Cayres (2005, p.5). Nesse modelo os registros ficam organizados na forma de grafos que contempla apenas um único tipo de associação (set) responsável por definir uma relação 1: N entre dois tipos de registros, proprietário e membro.

O Modelo Relacional “representa o banco de dados como uma coleção de relações. Informalmente, cada relação se parece com uma tabela de valores”. (ELMASRI, NAVATHE, 2005, p.90). O modelo relacional foi desenvolvido para solucionar as seguintes necessidades: aumentar a independência de dados nos sistemas gerenciadores de banco de dados, prover um conjunto de funções apoiadas em álgebra relacional para armazenamento e recuperação de dados e permitir processamento dedicado e exclusivo.

Como o modelo relacional é estruturado em relações, para se trabalhar com ele é necessário tomar conhecimento de algumas restrições que irão ajudar a evitar repetições de informações, perda de informações e outros. Essas restrições formam um tipo de nomenclatura para melhor entendimento, contendo item de dado que seria campo, coluna, atributo, registro chamado de linha (tupla) e a manipulação com SQL

descrita como operações CRUD[3]. (CAYRES, 2005, p.5-6).

Segundo Elmasri e Navathe, “na terminologia do modelo relacional formal, uma linha é conhecida como tupla, um cabeçalho de coluna é conhecido como atributo, e a tabela é chamada relação” (ELMASRI, NAVATHE, 2005, p.90);

O Modelo Orientado a Objetos surgiu devido aos limites que o modelo relacional possui, por exemplo armazenar dados provindos de linguagens de programações orientadas a objetos. Apesar de ter aparentemente resolvido os problemas encontrados no modelo relacional, não ganhou mercado como o esperado, e atualmente é mais utilizado em aplicações especializadas.

O Modelo Objeto-Relacional prevê a implementação de uma camada de abstração de dados em cima dos métodos relacionais, o que o torna possível a manipulação de dados mais complexos.

O Modelo NoSQL (Not only SQL - Não só SQL) “é um termo utilizado para definir um tipo de banco de dados que não segue normas de tabelas presente no banco de dados relacional” (CAYRES, 2005, p.6). Esse tipo de modelo foi desenvolvido com objetivo de conseguir armazenar e manipular dados no contexto de Big Data.

1.4.2 Projeto de Banco de dados

Um projeto de banco de dados que vai dar suporte a um sistema desenvolvido, deve ser planejado e executado com cuidado, já que o mesmo será a base de todo o sistema, sem ele o sistema pode se tornar inviável. Para que um projeto de banco de dados obtenha êxito, deve ser desenvolvido passando por algumas etapas, que seriam os níveis de abstração formados pelo modelo conceitual (DER), modelo Lógico e Modelo físico. Essas etapas aparentemente simples, são umas das mais importantes dentro de um projeto de software afirma Cayres:

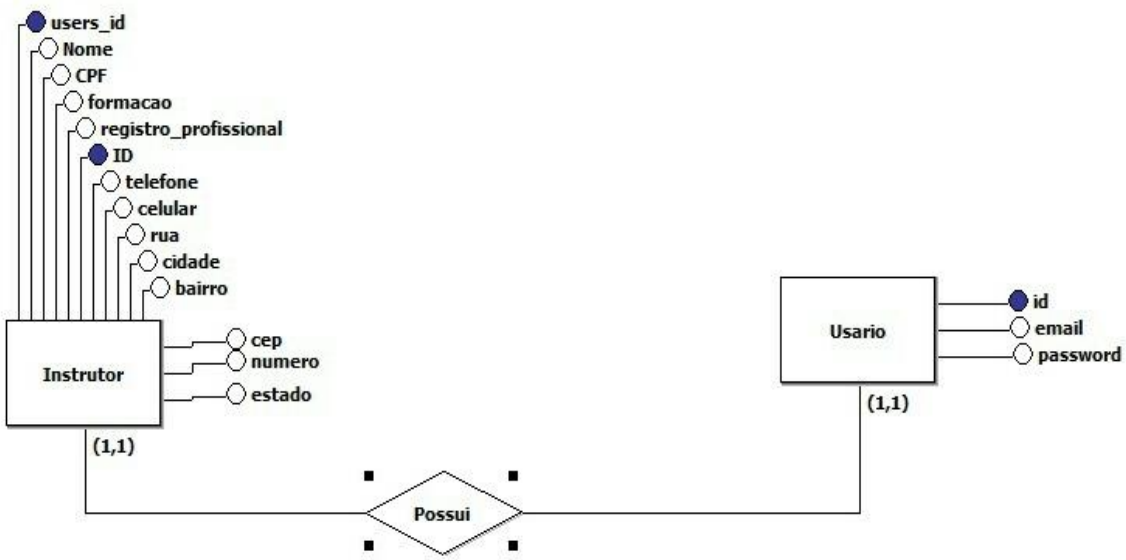
> São enormes as pressões para que a equipe de informática possa responder o mais rapidamente possível as demandas para colocar em funcionamento o novo sistema. Assim, não é incomum tomar conhecimento de projetos de desenvolvimento onde as etapas de levantamento de requisitos e modelagem são relegadas a um segundo plano. Esse é um erro comum que costuma cobrar seu preço, em geral muito significativo, nas etapas seguintes do processo de desenvolvimento. (CAYRES, 2005, p.10).

[3] CRUD, é um termo utilizado para designar as quatro operações básicas em Banco de Dados; Create, Read, Update e Delete em Inglês.

O Modelo Conceitual (DER) foi criado na década de 70, e “seu objetivo é facilitar a compreensão por parte do usuário, sendo visto como ferramenta útil durante o processo de projeto da base de dados” (CAYRES, 2005, p.15). É um modelo de dados abstrato que descreve a estrutura de um banco de dados, é também conhecido como Diagrama Entidade-Relacionamento (DER ou Modelo E-R).

O diagrama de modelo DER é composto por três tipos de elementos, entidades que representa o objeto no mundo real, atributos que qualificam uma entidade e relacionamentos que são as dependências entre entidades associadas.

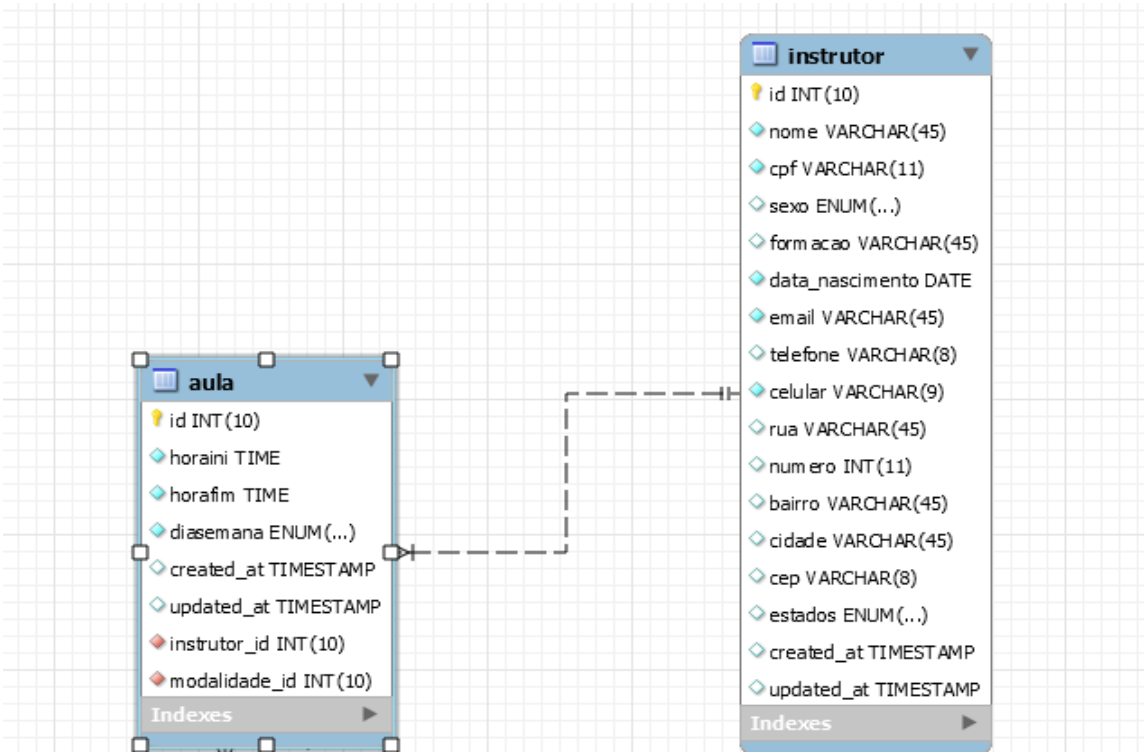
Figura 3: Modelo Conceitual



Fonte: Dos próprios autores.

Modelo Lógico (Esquema do banco de dados) que tem como objetivo “transformar o modelo conceitual em um modelo que define como o banco de dados será implementado em um SGBD especifico”. (CAYRES, 2005, p.10).

Figura 4: Modelo Lógico.



Fonte: Dos próprios autores.

O Modelo Físico (*Script* do banco de dados em SQL) “nessa fase o modelo do banco de dados é enriquecido com detalhes que influenciam no desempenho do banco de dados, mas não interferem na sua funcionalidade” (CAYRES, 2005, p.11).

Figura 5: Modelo Físico

```
CREATE TABLE `users` (
  `id` int(10) UNSIGNED NOT NULL,
  `name` varchar(45) COLLATE utf8mb4_unicode_ci NOT NULL,
  `email` varchar(45) COLLATE utf8mb4_unicode_ci NOT NULL,
  `password` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `remember_token` varchar(100) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE `instrutor` (
  `id` int(10) UNSIGNED NOT NULL,
  `nome` varchar(45) COLLATE utf8mb4_unicode_ci NOT NULL,
  `cpf` varchar(11) COLLATE utf8mb4_unicode_ci NOT NULL,
  `sexo` enum('masculino','feminino','outro') COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `formacao` varchar(45) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `data_nascimento` date NOT NULL,
  `email` varchar(45) COLLATE utf8mb4_unicode_ci NOT NULL,
  `telefone` varchar(8) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `celular` varchar(9) COLLATE utf8mb4_unicode_ci NOT NULL,
  `rua` varchar(45) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `numero` int(11) DEFAULT NULL,
  `bairro` varchar(45) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `cidade` varchar(45) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `cep` varchar(8) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `estados` enum('AC','AL','AP','AM','BA','CE','DF','ES','GO','MA','MT','MS','MG','PA','PB','PR','PE','PI','RJ','RN','RS','RO','RR','SC','SE','SP','TO')
              COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

Fonte: Dos próprios autores.

1.4.3 SQL

A linguagem SQL é a linguagem padrão utilizada pelos bancos de dados, tornando mais fácil a comunicação e compreensão da forma de comunicação dos bancos de dados de maneira generalizada. Elmasri e Navathe (2005, p.148) afirmam que “a linguagem SQL pode ser considerada uma das maiores razões para o sucesso dos bancos de dados relacionais no mundo comercial”.

O termo SQL significa *Structured English Query Language* - Linguagem Estruturada de Consulta, foi projetada e implementada pela IBM Research como uma interface para um sistema experimental de um banco de dados relacional chamado SISTEMA R.

A linguagem SQL se tornou a linguagem padrão para os SGBDs relacionais devido ao esforço conjunto da ANSI (*American National Standards Institute* - instituto Nacional Americano de Padrões) e da ISO (*International Standsrds Organization* - Organização Internacional de Padrões) após essa padronização a linguagem SQL teve várias modificações e melhorias até chegar na versão padrão definitiva chamada de SQL3, que atualmente é conhecida como SQL-99.

A linguagem SQL possui os comandos necessários para manipulação de um banco de dados como definição de dados, consultas e atualizações. “Para realizar comandos, o SQL utiliza a nomenclatura de termos tabela, linha e coluna em vez dos termos relação, tupla e atributo que é utilizado no modelo relacional formal” (ELMASRI, NAVATHE, 2005, p.149).

1.5 ***Front-end***

1.5.1 Hypertext Markup Language (HTML)

HTML vem de *HyperText Markup Language* que traduzindo significa linguagem de marcação de hipertexto. “*HyperText* está ligado a ideia de um texto que apresenta diversos caminhos diferentes de leitura, cujas partes estão interconectadas”. HTML é uma linguagem usada para criar páginas da web com as marcações em seus conteúdos (FERREIRA, 2013, E-book).

Essa linguagem de marcação é interpretada pelos navegadores para a criação

de páginas da web, a linguagem é constantemente confundida com linguagem de programação, o que não é verdade, visto que uma linguagem de programação é um conjunto de instruções usadas para definir um programa, nesse sentido HTML sozinha não introduz nenhuma instrução para o computador.

Em “1989 Tim Berners-Lee inventa a Web com HTML como sua linguagem de publicação” (LONGMAN, 1998).

O HTML não é uma linguagem case *sensitive*, isto é, não difere letras minúsculas de maiúsculas, outra peculiaridade do HTML é que se divide em três componentes principais o <html> onde é iniciado o arquivo, <head> é literalmente a cabeça do arquivo, onde é realizado a importação de outros códigos externos como JavaScript e CSS (*Cascading Style Sheets)* e por fim o <body> onde fica o conteúdo da página, o que o usuário enxerga através do navegador.

1.5.1.1 *HTML5*

O HTML5, veio com a intenção de conceber aos programadores a opção de gerar códigos mais organizados, utilizando agora a marcação do conteúdo, e deixando o estilo somente para as folhas de estilo, aderindo o foco agora apenas na parte semântica. (FERREIRA, 2013, E-book).

Como Botelho conclui sobre as novidades do HTML5:

> Com a chegada da nova versão da linguagem de marcação HTML só temos a ganhar, pois além de contarmos com novos recursos que permitem uma maior estruturação dos documentos, mais funcionalidades, independência de plataforma, tratamento de exceção e suporte nativo aos recentes conteúdos multimídia, ainda podemos ficar livres dos incômodos estresses que temos ao utilizar bibliotecas e *plugins* externos que nem sempre funcionam como deveriam nos navegadores distintos.[4]

1.5.2 JavaScript

JavaScript criado em maio de 1995 por Brendan Eich, ganhou este nome

[4] http://www.devmedia.com.br/as-novidades-do-html5/23992

através de uma jogada de marketing em cima do nome Java, pois nesta época era bastante popular[5]. Esta jogada teve uma grande consequência que ainda assombra a linguagem até os dias atuais, pois comumente se confunde JavaScript com Java, sendo que são linguagens totalmente diferentes.

JavaScript se tornou uma linguagem importante na atualidade, como Balduino (2014, prefacio) demonstra: “o JavaScript passou do estigma de linguagem odiada e mal compreendida a uma das mais interessantes e essenciais”, com sua praticidade para manipulação do DOM *(Document Object Model)*, é mais simples ainda com a utilização da biblioteca JQuery, o JavaScript pode mudar totalmente a estrutura de uma página web ou somente atribuir funções a mesma.

Escrever grandes trechos ou partes importantes com JavaScript através do DOM, pode ocasionar um mal funcionamento da página, levando em conta que nem todos os usuários usam navegadores atualizados, o que acarretaria um mal funcionamento da sua página web, já que JavaScript não é suportado em todas as versões nos navegadores.

JavaScript é implementada no HTML, desde até exibir uma mensagem para o usuário ou validação de campos como CPF, CEP ou resolução de cálculos, tudo no *client-side* (lado cliente de uma aplicação). Dessa forma a linguagem desafoga muito o tráfego de processamento do lado servidor da aplicação, pois não é mais necessário enviar tudo para que o servidor processe. Efeito disso vai ser um servidor mais leve e uma sensação de fluidez na página.

Neste projeto, JavaScript será usado para efeitos de transições e ambientação da página, assim como será utilizado folhas de estilos CSS3 (abordadas futuramente nesta pesquisa). Muito além dessa função de estilo de página, JavaScript vem crescendo bastante em todas as áreas como Balduino apresenta:

> Se faz presente no *back-end*, em jogos, servindo de motor para implementações de novas máquinas virtuais, emulação de hardwares antigos e até como linguagem padrão de plataformas inteiras, como Firefox OS e Windows 8. (BALDUINO, 2014, prefacio).

[5] Traduzido pelo autor. https://www.w3.org/comunity/webed/wiki/A_Short_History_of_JavaScript.

Onde até mesmo podendo ser utilizada no lado servidor assim como PHP (*PHP Hypertet Preprocessor*), (abordada futuramente), “JavaScript é uma linguagem completa: você pode usar isso em muitos contextos e fazer de tudo, com isso você pode alcançar o que as outras linguagens consegue”[6].

1.5.2.1 *JQuery*

JQuery é uma biblioteca JavaScript gratuita criada com o objetivo de tornar mais simples e intuitivo os códigos JavaScript. JQuery é construída seguindo a linhagem de JavaScript, sendo assim tudo que JavaScript faz, JQuery faz, porém de forma mais simples e com menor uso de linhas de código. Não se pode confundir JQuery como uma linguagem de programação, pois na verdade JQuery é apenas uma biblioteca JavaScript. (SCHEID, 2015, E-book).

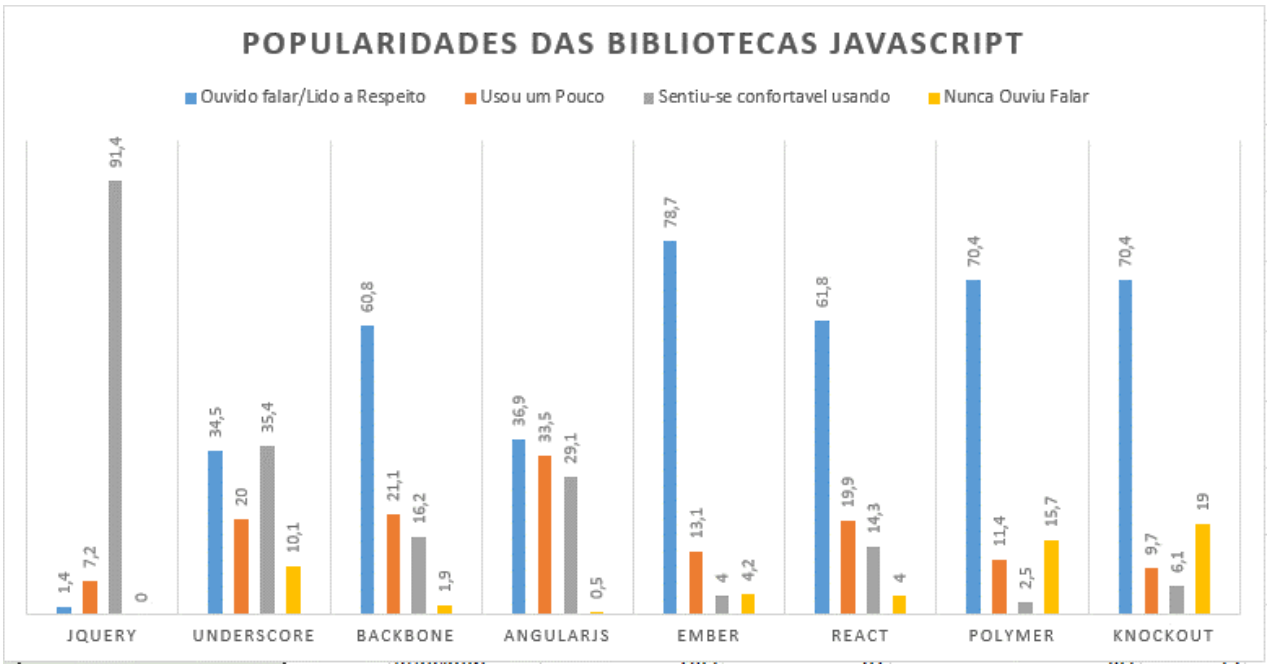
A simplicidade imposta pelo JQuery é de “brilhar os olhos”, pode haver casos de códigos escritos com centenas de linhas de código, serem reescritos com apenas dezenas utilizando a sintaxe JQuery. Esta funcionalidade é um aliado e tanto para os programadores desde iniciantes até profissionais mais experientes, pois, a velocidade de escrita é fortemente acrescida com esta biblioteca.

Assim como Scheid afirmou que JQuery foi feito para facilitar, Duckett reforça a ideia. O lema do JQuery é “escrever menos e fazer mais, porque ele permite alcançar os mesmos objetivos, mas em menos linhas de código do que você precisaria escrever com JavaScript simples” (DUCKETT, 2015, p.301).

Como abordado anteriormente JQuery é a mais famosa e utilizada biblioteca JavaScript, como é exibido no gráfico a seguir:

[6] http://nodebr.com/javascript-no-servidor-com-node-js

Figura 6: Popularidade das bibliotecas JavaScript



Fonte: Adaptada de: https://ashleynolan.co.uk/blog/frontend-tooling-survey-2015-results

A Pesquisa realizada em 2015 sobre as bibliotecas de JavaScript com um total de 2048 pessoas, apontou que 91,4% dos desenvolvedores se sentiam confortáveis com a utilização desta biblioteca, e em nível de conhecimento foi de longe a mais abrangente e com índice de 0% nunca terem ouvido falar a respeito da biblioteca, com isso é quase que certo constar que escolher JQuery como sua biblioteca vai lhe proporcionar menos trabalho e maior aproveitamento, pois a consequência dessa popularidade é um melhor suporte, ou pesquisas em documentação sobre o mesmo.

1.5.3 *Cascading Style Sheets* (CSS)

O *CASCADING STYLE SHEETS* (CSS) teve origem com Håkon Wium Lie em 1994, Lie trabalhava na CERN, onde viu que os sites estavam começando a serem utilizados para realizar publicações eletrônicas. Com isso surgiu a dificuldade, escrever um *layout* em forma de um jornal em uma página Web. Já tendo trabalhado no MIT (*Massachusetts Institute of Technology*), Lie viu a necessidade de criar uma linguagem de folha de estilo, porém não era uma novidade na época, a separação da

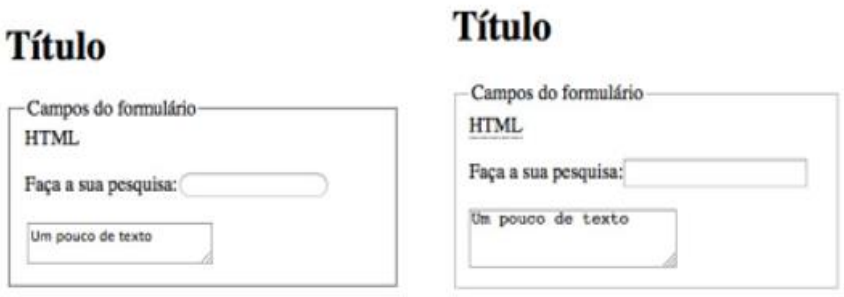
estrutura do documento do seu *layout* já havia sido pensada por Tim Berners-Lee em 1990.

Em 1995 no *workshop* situado em Paris foi o marco na garantia que o CSS tivesse seu lugar na *web*. [7]

Segundo Miyagusku a CSS surgiu com objetivo de solucionar as deficiências e limitações do HTML, sempre focando na estética das páginas, já que páginas HTML estavam ficando muito complexas devido à mistura de conteúdo e formatação no mesmo arquivo, as folhas de estilo vieram para fazer essa separação do conteúdo da formatação. (MIYAGUSKU, 2007, E-book).

O mesmo site em navegadores diferentes apresentam padrões de exibição diferentes, como é exemplificado a seguir.

Figura 7: Reprodução de um formulário em navegadores diferentes



Fonte: Dos próprios autores.

A solução para esse incomodo, é o uso de resets como retrata Mazza (2014, p.36) “os arquivos de reset, como são chamados, possuem uma gama de regras para alinhar os navegadores em um mesmo patamar de estilo, seja corrigindo problemas ou resolvendo inconsistências”. Dessa forma a preocupação com fontes, tamanhos e posicionamentos em cada navegador diminui, porém não significa que deve se desapegar disso e não realizar testes em todos os navegadores.

1.6 **Back-end**

1.6.1 *PHP* Hypertext Preprocessor (PHP)

[7] Traduzido pelo autor: https://www.w3.org/Style/CSS20/history.html

O PHP é uma linguagem de programação web voltada para a internet. O foco dessa linguagem é o lado do servidor de uma aplicação, ou seja, ele é responsável por fazer a comunicação do *front-end* com o *back-end*, o PHP que realiza as buscas necessárias no servidor e no banco de dados para que seja obtido uma saída em formato HTML para o usuário que está navegando.

> A primeira versão do PHP surgiu em 1995, quando Rasmus Lerdorf criou para uso pessoal uma ferramenta chamada PHP/FI (*Personal Home Page/Forms Interpreter*). Porém, ele não imaginava que estava criando uma das mais poderosas linguagens para desenvolvimento de aplicações na web. (NIEDERAUER, 2017, p.16)

Em 1998 dois israelenses Zeev Suraski e Andi Gutmans reescreveram por completo o PHP, transformando-o realmente em uma linguagem de programação que foi renomeada para *PHP Hypertext Preprocessor*, como explica Guanabara [8]

> O PHP pode ser utilizado em diversos sistemas operacionais, como Microsoft Windows, MAC OS, Linux e várias distribuições do Unix, também é suportado pela maioria dos servidores web existentes atualmente, pode ser utilizado tanto em programação estruturada quanto com programação orientada a objeto (OOP), assim como suporta uma ampla variedade de banco de dados.[9]

A linguagem PHP pode ser utilizada embutida no HTML, importa-la em arquivos fontes separados que são carregados pelo servidor trabalhando com o arquivo HTML.

Devido ser uma linguagem *server-side*, o código escrito em PHP não é acessado pelo usuário, ou seja, um visitante na página web, não tem acesso ao lado servidor de uma aplicação, como ocorre em JavaScript, apesar de também ser uma linguagem de script, é possível que o usuário tenha acesso ao código JavaScript executado pelo browser através do código fonte da página, onde é possível visualizar a estrutura do JavaScript, essa é uma das principais diferenças entre as linguagens.

No entanto, como é uma linguagem de servidor, PHP pode acarretar uma pequena sobrecarga no servidor, pois a mesma é executada diretamente no servidor

[8] http://www.cursoemvideo.com/course-status/
[9] https://secure.php.net/manual/pt_BR/intro-whatcando.php

(BALDUINO, 2014, p07).

De toda forma PHP é a linguagem altamente usada em sites atualmente, segundo a W3Tech a linguagem PHP é utilizada em 82,8% dos sites, enquanto linguagens como JavaScript conseguiu apenas 0,4%[10]. Não dá para pensar em escrever uma aplicação web prática e rápida sem pensar em PHP atualmente.

1.6.2 Servidor Web

Servidores Web são computadores com software específico que permitem a aceitação de solicitações de computadores de clientes e retornam respostas à essas solicitações. Os servidores Web permitem que se compartilhe informações pela Internet, intranet ou extranet.[11]

> Um programa de computador responsável por aceitar pedidos HTTP de clientes, geralmente os navegadores, e servi-los com respostas HTTP, incluindo opcionalmente dados, que geralmente são páginas web, tais como documentos HTML com objetos embutidos (imagens, etc.)[12].

O servidor web mais utilizado no mundo é o Apache. O Apache HTTP Server Project é um esforço colaborativo de desenvolvimento de software destinado a criar uma implementação de código fonte robusta, comercial, funcional e livremente disponível de um servidor HTTP (Web)[13].

1.7 **Frameworks**

Frameworks basicamente são um conjunto de soluções para uma gama geral de problemas, isto é uma aplicação quase completa, mas com pedaços faltando. Ai que entra uma aplicação que proverá estes pedações que no final resultará no produto final[14]. Frameworks são grandes facilitadores na vida de um programador, ajudam a agilizar o processo de desenvolvimento. Segundo Turini (2015, p.8), independentemente da linguagem ou tecnologia escolhida, ninguém quer ficar se

[10] Traduzido pelo autor: https://w3techs.com/technologies/overview/programming_language/all
[11] https://technet.microsoft.com/pt-br/library/cc770634(v=ws.11).aspx
[12] https://www.oficinadanet.com.br/artigo/servidores/o_que_e_um_servidor_web
[13] Tradução livre do autor: http://httpd.apache.org/ABOUT_APACHE.html
[14] http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/frame/oque.htm

preocupando com infraestrutura, aí que os frameworks entram, com sua organização e seu bom aproveitamento de código.

1.7.1 Laravel

Laravel é um *framework* PHP consideravelmente famoso, a frente de outros grandes como Angular, até a presente data Laravel é o 7º *framework* mais utilizado.[15]

Este é utilizado para desenvolvimento web, onde uma de suas características é o desenvolvimento seguro de aplicações e de rápida performance, com um código incrivelmente simples de se escrever e ler.

Lavarel utiliza o conceito MVC (*Model View Controller*) para organização de seus diretórios, o modelo MVC é bastante conhecido e utilizado, onde separa a aplicação em três diretórios, onde cada componente fica em seu respectivo local de uso. Utilizar este modelo traz consigo um benefício de um projeto organizado, modelo padrão, e de fácil manutenção, além de uma aplicação mais limpa e leve. Além disso, adicionar ou remover funcionalidades em sua aplicação pode se tornar uma tarefa bastante simples e rápida.[16]

O Composer faz parte da instalação do Laravel, é uma ferramenta para gerenciamento de dependências do PHP, que permite incluir pacotes, códigos e bibliotecas de outras pessoas em seu projeto, ele sozinho cria um *autoloader* para deixa-las prontas para utilização. Atualmente vem se tornando bastante famoso.

Sua similaridade com outro *framework* bastante conhecido, o Ruby on Rails da linguagem Ruby. Laravel utiliza uma *engine* de template chamada Blade, que por si só traz uma série de ferramentas que ajudam a criar bonitas interfaces. Sua comunicação com o banco de dados é feita por meio da Ative Record chamada Eloquent ORM, que traz várias funcionalidades para facilitar a manipulação do banco de dados.

1.7.2 Bootstrap

[15] Traduzido pelo autor: https://hotframeworks.com/
[16] Traduzido pelo autor: https://book.cakephp.org/1.3/pt/The-Manual/Beginning-With-CakePHP/Understanding-Model-View-Controller.html

Um *framework front-end,* atualmente sua versão é a 4.0 beta, uma ferramenta de código aberto desenvolvido pelo Twitter e atualmente mantido pela equipe que o fundou e mais um grupo de colaboradores. Esse *framework* foi criado segundo Silva, para solucionar um problema de compatibilidade e inconsistências, pois cada desenvolvedor da equipe no Twitter usava sua biblioteca que era familiar (SILVA, 2015, p.21). Após sua criação, visto que o *framework* seria de grande suporte aos desenvolvedores, liberado então de forma gratuita regido sobre a MIT *license*[17] o uso do *Bootstrap* com o objetivo de ajudar desenvolvedores na criação de sistemas web.

Zamel afirma que “Um site com web design responsivo - ou *responsive* web design pode ser acessado de um PC, notebook, smartphone, tablet, TV, geladeira, banheira” (ZAMEL, 2012, p.9) com a Internet das Coisas até os periféricos mais rústicos atualmente acessam a internet. O fato é que um sistema web responsivo é acessível a todos, e onde for possível acessar a internet, será possível visualizar seu site de uma forma que continue bem apresentável.

Segundo Zamel (2012, prefácio). “O Web Design Responsivo é a chave para essa nova Web. É pensar em páginas que se adaptem a todo tipo de dispositivo e contexto de uso”, neste sentido é deixar de pensar em limitações de tela e começar a imaginar telas com tamanho flexível. Pensar em um site que pode ser acessado em qualquer dispositivo. Responsividade é um ponto mais que essencial quando se pensa em uma aplicação web atualmente.

> Em outras palavras, o design responsivo ou layout responsivo expande e contrai com a finalidade de se acomodar de maneira usável e acessível à área onde é visitado ou, mais genericamente, ao contexto onde é renderizado, seja um smartphone, um tablet, um leitor de tela, um mecanismo de busca etc. (SILVA, 2014, p.35).

[17] Traduzido pelo autor: https://v4-alpha.getbootstrap.com/about/license/

2 MATERIAIS E MÉTODOS

2.1 Sistema Operacional

O Sistema Operacional escolhido para o desenvolvimento foi o Microsoft Windows 10, para que o sistema fique totalmente apto para se desenvolver um programa web, foi necessário a instalação de mais softwares e/ou configurações no Sistema Operacional (SO). Algumas delas foram instalação da IDE (*Integrated Development Environment*) escolhida o PHPStorm, e outros softwares como XAMPP (Sendo ‘X’ para retratar qualquer um dos sistemas operacionais, e ‘A’ Apache, ‘M’ MariaDB, ‘P’ PHP e ‘P’ Perl), Composer, Laravel e Vagrant.

2.2 XAMPP

O primeiro, e praticamente o mais importante, que o ambiente de desenvolvimento deva estar munido é o XAMPP, pois é possível a manipulação do banco de dados, devido à ferramenta PHPMyAdmin que já vem incluso no mesmo, assim como o servidor web Apache e o SGBD MariaDB responsável pelo banco de dados utilizado pela aplicação. O XAMPP é um ambiente de desenvolvimento PHP, gratuito munido pela licença GNU General *Public Licenc.*

O XAMPP pode ser baixado pelo seu site oficial https://www.apachefriends.org/pt_br/download.html, após fazer o *download*, deve-se executar o *installer* e seguir as telas clicando em *next* até finalizar a instalação.

Após instalado deve-se executá-lo e iniciar os serviços Apache e MySQL, para uma maior comodidade, o XAMPP pode ser configurado para inicializar juntamente com o sistema operacional e também estes mesmos serviços.

2.3 **Composer**

Para que possa desenvolver sistemas WEB com o Laravel, deve-se primeiro ser instalado o Composer no ambiente. O mesmo pode ser obtido pelo link: https://getcomposer.org/Composer-Setup.exe, e executado normalmente via o próprio *installer*, e seguir as telas com o *next*.

2.4 **PHPStorm**

Como já mencionado anteriormente a IDE ou Ambiente de Desenvolvimento Integrado escolhida foi a da empresa JetBrains inicialmente na sua versão 2017.1, agora na versão 2017.2.4, que pode ser obtida através do link: https://www.jetbrains.com/phpstorm/download. Esta ferramenta é paga com preços para desenvolvimento individual de USD $ 8,00 mensais ou USD $ 89,00 anuais, no entanto a JetBrains oferece planos com descontos para várias categorias, como de estudantes, que foi a licença requerida pelos desenvolvedores, que recebe um semestre de acesso completo e gratuito a IDE com a possibilidade de renovação.

Após o download da IDE, deve executar o instalador e seguir os passos de clicando em *Next* até que se conclua a instalação. Após a conclusão da instalação na primeira inicialização é requerido que entre com sua licença de uso, escolhida como dito previamente.

2.5 **Laravel**

Após a instalação do Composer, já é possível fazer o download do Laravel para o ambiente, acessando o site do laravel.com/docs o usuário será redirecionado a documentação/versão mais recente do framework. Nesta página contém as instruções para a instalação do mesmo, que também serão descritas aqui as principais.

Primeiro deve ser aberto um terminal de sua escolha (CMD ou PowerShell, ambos estão aptos a essa tarefa), depois de aberto o usuário deve executar o seguinte comando: **composer global require "laravel/installer”** que será responsável por realizar o download do Laravel e suas dependências.

Após o download completo das dependências do Laravel o ambiente estará finalmente completo para desenvolver sua primeira aplicação WEB com o framework Laravel. Para iniciar seu primeiro projeto o usuário pode dirigir-se ao diretório onde o mesmo quiser que se armazene suas pastas de projeto e executar o seguinte comando: **laravel new Projeto**, que criará um novo projeto com toda estrutura do Laravel.

Após isso, pode-se abrir o projeto no PHPStorm e iniciar seu desenvolvimento a maneira que o agradar. Onde “Projeto” é o nome do projeto a ser criado, o usuário tem total liberdade de escolher este nome como desejar.

2.6 **Vagrant**

Para simular um ambiente virtual e de certa forma, a uma hospedagem online do projeto. Foi escolhido a Vagrant, para simular um ambiente virtual Linux na máquina Windows. Para instalá-la é bastante simples acessando o link: https://www.vagrantup.com/downloads.html escolha o sistema operacional e faça o download do pacote.

Para que a Vagrant funcione corretamente ela deve primeiramente baixar o virtualbox que pode ser obtido pelo link: https://www.virtualbox.org/wiki/Downloads, após o download, siga os passos do instalador.

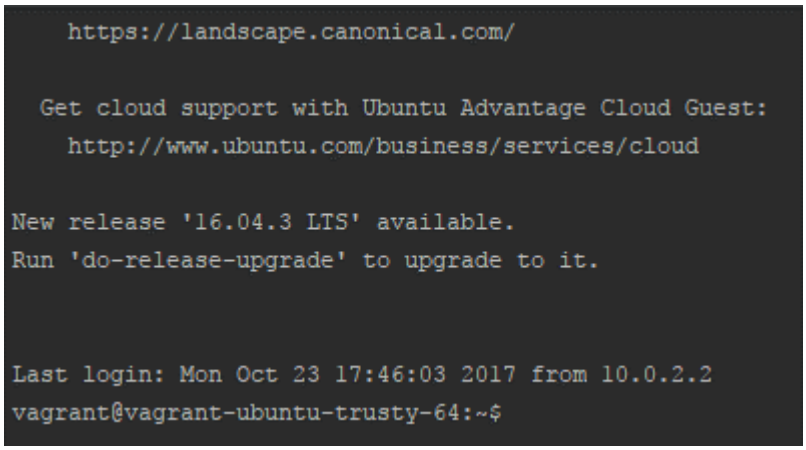
Após os dois serem instalados, para facilitar ainda mais a instalação e configuração do Vagrant é recomendado baixar estes arquivos fornecidos pela Especializa TI onde já vem pré-configurado a Vagrant, disponível no link: https://github.com/carlosfgti/vagrant-setup-php. Após o download concluído, deve-se descompacta-los em um diretório de sua escolha (recomenda-se na raiz do disco), onde deve-se atentar para que não haja espaços e caracteres especiais no nome do diretório.

Finalmente após estes passos, dentro do repositório criado e com os arquivos extraídos no mesmo, deve se abrir o terminal (CMD ou PowerShell) e executar o seguinte comando: **vagrant up**, responsável por inicializar a máquina virtual vagrant. Após a execução será baixado e configurado automaticamente tudo que será

necessário para desenvolver no ambiente Vagrant.

Para que se possa conectar a Vagrant é necessário uma conexão via SSH (*Secure Shell*), no sistema Windows não é possível fazer esta conexão via terminal, pode ser usado programas como o Putty, mas como a IDE PHPStorm já oferece essa opção não é necessário o download de mais um software.

Figura 8: Conexão via SSH pelo PHPStrom



Fonte: Dos próprios autores.

3 DESENVOLVIMENTO

3.1 Desenvolvimento do sistema

Este capítulo visa apresentar a forma com que foi realizado o desenvolvimento do sistema web, desde o levantamento dos requisitos até os testes realizados.

3.1.1 Requisitos

O primeiro passo de um projeto de desenvolvimento é a coleta e análise dos requisitos que comportam o sistema, é uma etapa extremamente importante já que ela informará o que o *software* deverá realizar.

O processo para coletar requisitos para o sistema web seguiu pelo caminho onde diálogos e entrevistas foram realizadas para coletar os dados e informações necessários para compreender quais funcionalidades o sistema deveria realizar. Os requisitos coletados algumas vezes recebem mudanças ao longo do desenrolar do projeto, tal como em determinados casos o usuário não sabe definir exatamente o que deseja, cabendo aos desenvolvedores explicar e exemplificar possíveis soluções para o determinado caso.

Dentre os requisitos coletados durante as entrevistas com a cliente, foram separados em duas categorias, os requisitos funcionais (RF) e os requisitos não funcionais (RNF).

Os requisitos funcionais são aqueles que descrevem quais serão os serviços oferecidos pelo sistema. Enquanto os requisitos não funcionais são os que apresentam as características que o sistema deve possuir, suas funções.

Quadro 01 - RF

Código	Descrição
RF01	Para utilizar o sistema o usuário deve possuir acesso à internet
RF02	O sistema deve possuir entrada por meio de login por e-mail e senha
RF03	O sistema deve exibir, assim que o usuário logar, os horários de aulas cadastrados
RF04	O sistema deve atender aos vários tipos de cadastros necessários para exibir os horários de aulas
RF05	As informações devem ser exibidas de acordo com o que o usuário selecionou
RF06	Realizar cadastros de usuário, aluno, instrutor e modalidade

Fonte: Dos próprios autores.

Quadro 02 - RNF

Código	Descrição
RFN01	O usuário do sistema deve possuir um computador ou notebook com acesso à internet
RFN02	O sistema deve ser de fácil entendimento e utilização
RFN03	O sistema deverá tratar falhas de conexão com a internet
RFN04	A aplicação deve retornar apenas dados verdadeiros
RFN05	O sistema deve atender as necessidades do usuário de forma satisfatória
RFN06	O sistema deve validar a integridade dos dados inseridos
RFN07	O sistema deve possuir um nome junto a uma logo marca

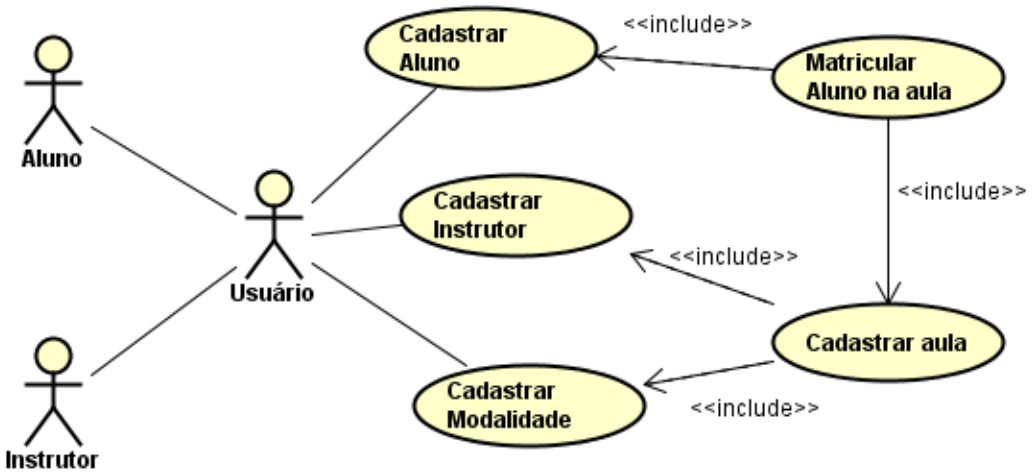
Fonte: Dos próprios autores.

Após a coleta dos requisitos o sistema web recebeu o título de WebFit, devido relacionar prática de exercícios para uma melhor saúde com o tipo de sistema desenvolvido, sistema web.

Utilizando a modelagem UML (*Unified Modeling Language*), foi desenvolvido um caso de uso do sistema sobre os horários a serem exibidos para o usuário, devido

ser a funcionalidade mais importante do sistema web. Para a construção do mesmo foi utilizado os itens que um caso de uso possui que são atores, caso de uso e relacionamento. Como é descrito na figura 9.

Figura 9: Diagrama de caso de uso cadastro de horário



Fonte: Dos próprios autores.

Caso de uso: Criar horário.

Ator principal: Usuário.

Atores secundários: Instrutor e aluno.

Meta de Contexto: Cadastrar os horários de aulas utilizando os dados de instrutor, modalidade, alunos e aula já cadastrados.

Precondições: Usuário estar logado no sistema, instrutor, aluno e modalidade estarem cadastrados.

Disparo: O usuário decide o horário das aulas ofertadas.

Cenário:

1. Usuário: loga no sistema com e-mail e senha.
2. Usuário: realiza cadastro de instrutor.
3. Usuário: realiza cadastro de aluno.
4. Usuário: realiza cadastro de modalidade.
5. Usuário: realiza cadastro de aula, selecionando uma modalidade e um instrutor já cadastrado.
6. Usuário: realiza a matrícula de um aluno na aula, selecionando a aula já cadastrada e o aluno também já cadastrado no sistema.

Exceções:

1. Senha inserida incorretamente: usuário introduz a senha correta;
2. Cadastro de instrutor incorretamente: usuário introduz dados corretos;
3. Cadastro de aluno incorretamente: usuário introduz dados corretos;

3.1.2 Banco de Dados

Diante da coleta e análise dos requisitos levantados durante as entrevistas, foi planejado e modelado um banco de dados que irá dar suporte à aplicação, armazenando dados e retornando os mesmos através de consultas, na forma de informação para o usuário. Para suprir as necessidades do sistema, foi utilizado o SGBD Maria DB, que além de ser *open source,* é eficiente e rápido.

O banco de dados denominado de “webfit” é composto por 8 (oito) tabelas onde cada uma possui uma função, cada uma delas possui seus respectivos campos, e um nome, que diferencia uma das outras, essas tabelas são: *users*, *password_resets*, *migrations*, instrutor, aula, modalidade, aluno, fatura e ficha.

A criação desse banco de dados e suas respectivas tabelas se deu através da utilização do framework Laravel, onde através da confecção das “*migration*” que são arquivos onde é definido a tabela e seus campos, de forma um pouco diferente, sem necessitar da linguagem SQL padrão dos SGBDs. Ao rodar o comando “***php artisan migration***” é executado todas as *migrations* criadas e dessa forma as tabelas são criadas.

A utilização do banco de dados dessa forma se torna viável, pois para

alterações nas estruturas das tabelas, como adição de novos campos, não se faz necessário a alteração direta no banco de dados através da linguagem SQL, alterando apenas a *migration* da tabela desejada e executar o comando "***php artisan migrate:refresh***", pronto, o banco está atualizado.

Código 1 - Migration

```
class CreateUsersTable extends Migration {

 public function up() {
  Schema::create('users', function (Blueprint $table) {
   $table->increments('id');
   $table->string('name', 45);
   $table->string('email', 45)->unique();
   $table->string('password');
   $table->rememberToken();
   $table->timestamps();
  });
 }

 public function down() {
  Schema::dropIfExists('users');
 }
}
```

Fonte: Dos próprios autores.

A função de criação de uma *migration* recebe o nome da tabela a ser criada e os respectivos campos que vai possui, com o tipo e o tamanho de cada um, a sintaxe é parecida com a de uma criação de uma tabela por SQL.

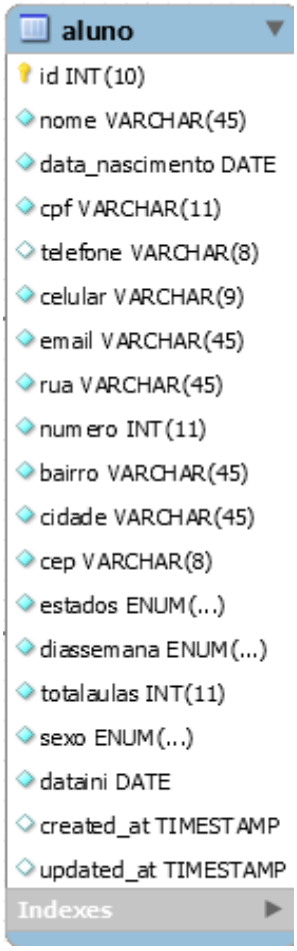
Dentre as tabelas citadas anteriormente, será descrito abaixo algumas das tabelas com maior grau de relevância para o objetivo principal do sistema WebFit.

A tabela "aluno" é responsável por armazenar os dados dos alunos cadastrados no sistema, alunos que nesse caso são os clientes da empresa. A tabela aluno possui 15 (quinze) campos, sendo eles: id, nome, data_nascimento, cpf, sexo, email, telefone, celular, cep, estado, cidade, bairro, rua, número e data_matricula, possui cada um deles um tipo de dado específico para receber, como *string*, date, *enum* e outros.

Esta tabela possui campos do tipo *unique*, ou seja, são campos importantes e que não devem possuir registros no banco de dados de mais de um aluno com o valor

igual ao de outro aluno cadastrado, esses campos são cpf e email, além dessa verificação no banco esses campos possuem também validação, ou seja, existe uma verificação através de uma função *javascript* que verifica e valida se o cpf e o email informado são verdadeiros.

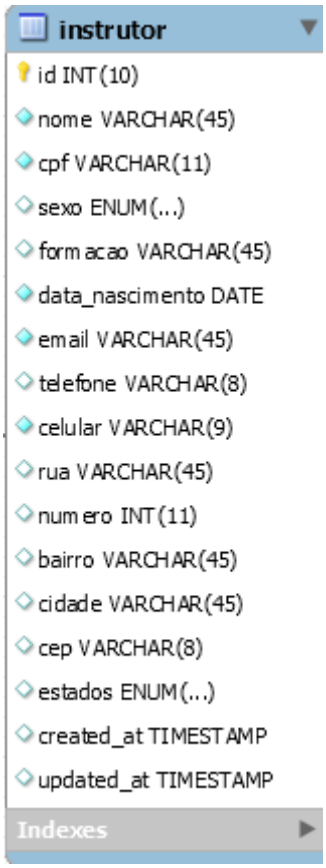
Figura 10: Tabela aluno



Fonte: Dos próprios autores.

Na tabela “instrutor” fica armazenado os dados dos instrutores responsáveis por conduzir as aulas em que os alunos serão matriculados. Ela possui 15 (quinze) campos, onde a grande maioria é igual à da tabela aluno assim como a verificação e validação dos campos, sendo eles: id, nome, data_nascimento, cpf, sexo, e-mail, formação, telefone, celular, cep, estado, cidade, bairro, rua e número, como mostra a figura 11.

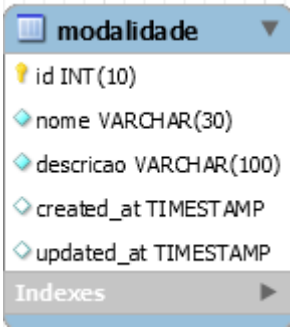
Figura 11: Tabela instrutor



Fonte: Dos próprios autores.

A tabela “modalidade” é responsável pelo armazenamento das modalidades de aulas ofertadas pela empresa que utilizará o sistema. Essa tabela possui apenas 3 (três) campos, que são: id, nome e descrição.

Figura 12: Tabela modalidade



Fonte: Dos próprios autores.

A tabela “aula” é a tabela principal do banco de dados do sistema web

desenvolvido, ela é responsável por armazenar os dados que serão utilizados para exibir os horários de aulas marcados para a usuária do sistema. Essa tabela possui 4 (quatro) campos, sendo eles: id, horaini, horafim diasemana e recebe o id das tabelas instrutor, modalidade e aluno, sendo a relação de aluno N:M gerando uma nova tabela “aluno_aula”.

Figura 13: Tabela aula

Fonte: Dos próprios autores.

Durante a criação do banco de dados registros foram inseridos diretamente no banco através dos *softwares* phpMyAdmin e MySQLWorkbench. Após a finalização, os registros foram inseridos através das telas de cadastros do sistema, a fim de descobrir se todos os dados dos campos estavam sendo enviados para o banco de dados. Também foi utilizado um recurso do *framework* Laravel para inserção de vários registros com finalidade de testes durante o desenvolvimento do sistema, esse recurso é o “*factorie*” que gera uma quantidade de registro definida pelo desenvolvedor para as tabelas também escolhidas pelo desenvolvedor.

3.1.3 Sistema Web

Após a finalização do banco de dados, iniciou-se o desenvolvimento do sistema, devido a concentração do projeto ser web, pode-se basicamente ser divido em duas partes, o *back-end* (lado servidor) e *front-end* (lado cliente) de uma aplicação.

Para a construção desse projeto, foi utilizado o *framework* php, Laravel, que

tem foco no lado servidor da aplicação, facilitando a configuração, criação e manipulação de arquivos php responsáveis por estabelecerem a conexão com o servidor e o banco de dados.

Outro grande facilitador para desenvolvimento com Laravel é a organização de sua estrutura de pastas e arquivos, que utiliza o padrão MVC - *Model-View-Controller* (Modelo-Visão-Controlador) que trabalha com uma organização de hierarquias de pastas que deixa o projeto organizado e com maior segurança.

No lado *back-end* da aplicação, um dos arquivos mais importantes do projeto é o ".env", o arquivo responsável pelas configurações de servidor e banco de dados, como descrito na figura 14, esse arquivo é composto pelos componentes:

- Os componentes iniciados com APP, estão relacionadas as configurações do projeto, como por exemplo o nome do sistema, a *url* (*Uniform Resource Locator* - Localizador Padrão de Recursos) de acesso pelo *browser*.
- Os componentes iniciados com DB, estão relacionadas as configurações de conexão com o banco de dados, onde é informado qual o tipo de SGBD, no caso o "mysql", o ip do host, a porta de acesso do apache, o nome da base de dados "webfit", o admin e senha de acesso do banco de dados.
- Os componentes com inicial MAIL são para auxilio de troca de senhas e outras informações que podem ser enviadas por e-mail para um usuário, nele são configuradas o e-mail e senha do e-mail que irá enviar a informação para o e-mail do usuário.

Figura 14: Arquivo env

Fonte: Dos próprios autores.

Após a configuração de acesso ao servidor externo e ao banco de dados, entramos na fase de codificação do sistema em si, que gira em torno de 4 (quatro) tipos de arquivos extremamente importantes, sendo eles: *Controllers, Models, Views e Routes*.

As *Views* são arquivos onde são desenvolvidas as telas da aplicação, ou seja, são os arquivos responsáveis por receberem e informarem dados e informações para o usuário do sistema. Utilizando o Laravel, as *views* não são arquivos com extensões ".html", mas sim ".*blade*.php" onde o *blade* possui *tags* e comandos que se assemelham aos do html. O trecho de código 2 exemplifica um trecho de código de uma *view* utilizando o *blade*.

O projeto roda em torno de uma *view* chamada app.blade.php que é a view principal do sistema, ou podemos chama-la de *template*, pois, todas as outras *views* são chamadas e executadas dentro dessa app. A app é composta do *header* (cabeçalho) onde é importado arquivos de *stylesheet*, devido as demais views serem

executadas dentro da app, todas as importações de arquivos externos de css, javascript, imagens e plugins são realizados nela. O body (corpo) é a área onde o sistema é exibido para o usuário, nele está codificado o menu superior e menu lateral, assim como o rodapé do sistema, dentro desse body existe um código conforme é demonstrado no trecho de código 2 que é responsável por receber o conteúdo das demais views.

Código 2: Trecho onde é chamado as demais views para a view app

```
<div class="content-page">
 <div class="content">
  <div class="container">
   @yield('container')
  </div>
 </div>

 <footer class="footer text-right"> 2017 © WebFit. </footer>
</div>
```

Fonte: Dos próprios autores.

Pode-se observar que dentro da abertura das div's existe o seguinte trecho de código `@yield('container')` que é responsável por receber o conteúdo das outras views, como as de cadastro de aluno, instrutor, modalidade e etc. Para que isto funcione, o arquivo de cada view recebe as seguintes linhas de códigos: `@extends('layouts.app')` e `@section('container').`

O trecho de código 3 é um exemplo de um formulário de cadastro de uma view utilizando as tags do blade no lugar das tags de um código html. Onde no lugar das tags para formulários, *labels*, inputs e botões, todas elas são iniciadas da mesma forma sendo a sintaxe do blade, todas iniciadas com "{!! Form::". Pode-se observar que apesar da diferença, as tags do blade permitem inserir classes, id, *names*, tipo e outros atributos aos elementos, assim como no html convencional.

Código 3: View

```
  @if (isset($ficha))
    {!!Form::model($ficha,['route'=>['ficha.update',$ficha->id],
'class'=>'form-horizontal','role'=>'form','method'=>'PUT'])!!}
  @else
    {!!Form::open(['route'=>['ficha.store'],'class'=>'form-
horizontal','role'=>'form',’method'=>'post'])!!}
  @endif
    {{csrf_field()}}

    <div class="col-md-10">
      <div class="form-group">
        {!!Form::label('exercicio','Exercício',['class'=>'control-
label','for'=>'exercicio'])!!}
        {!!Form::text('exercicio',null,['class'=>'form-control',
'required','id'=>'exercicio'])!!}
      </div>

      <div class="form-group">
        {!!Form::label('serie','N° de séries',['class'=>'control-
label','for'=>'serie'])!!}
        {!!Form::number('serie',null,['class'=>'form-
control','required','minlength'=>'1','maxlength'=>'2','id'=>'serie
'])!!}
      </div>

      <div class="form-group">
        {!!Form::label('repeticoes', 'N° de Repetições',
['class'=>' control-label', 'for'=>'repeticoes']) !!}
        {!! Form::number('repeticoes', null, ['class'=>'form-
control', 'required', 'minlength'=>'1', 'maxlength'=>'2',
'id'=>'repeticoes']) !!}
      </div>
```

Fonte: Dos próprios autores.

Após uma *view* receber dados informados pelo usuário, esses dados precisam ser enviados para o banco de dados, para isso a *view* chama uma rota, que está localizada no diretório *routes*, essas rotas são caminhos para comunicação entre uma *view* e um *controller*, onde a rota recebe o caminho até o arquivo *controller* desejado. O trecho de código 4 mostra como funciona uma rota.

Código 4: Routes

```
Route::get('/', function () {return view('welcome');});
Auth::routes();

Route::get('/home','HomeController@index')->name('home');
Route::get('/aluno/listar','AlunoController@listar');
Route::resource('/aluno','AlunoController');
```

Fonte: Dos próprios autores.

As rotas acima são caminhos para a view *welcome*, que é a página de apresentação do sistema, enquanto as demais rotas chamam métodos nos *controllers* dos arquivos home e aluno. O primeiro parâmetro recebido pela rota é o caminho que o usuário percorre até uma view, e o segundo parâmetro é uma função do controller dessa view que vai ser executada para chamar essa view que o usuário deseja acessar.

O *controller* é responsável por controlar as regras da aplicação, controlar quais funções ira possuir, quais parâmetros irá receber e retornar. Ele recebe esses dados diretamente na função para o qual foi chamado, funções como de inserção, edição, exclusão e listagem de dados. Cada uma das funções construídas no *controller* possui também um redirecionamento para uma outra *view,* conforme demonstrado no trecho de código 5.

Código 5: Controller

```
class AlunoController extends Controller {
  protected $aluno;

  public function __construct(Aluno $aluno){
    $this->middleware('auth');
    $this->aluno = $aluno;
  }

  public function index() {
    return view('aluno');
  }

  public function create() {
    //
  }

  public function store(RegrasAluno $request, Aluno $aluno){
    $dataform = $request->except('_token');
    $insert = $aluno->insert($dataform);

    if ($insert) {
      return redirect()->route('aluno.index')-
>with('sucess','Aluno foi inserido corretamente');
    } else {
      return redirect()->back()->with('error','Falha na inserção
do Aluno');
    }
  }
```

Fonte: Dos próprios autores.

O *controller* de aluno é responsável por executar as funções descritas anteriormente que cabem o *controller* gerenciar, nesse trecho de código algumas funções que são necessárias para a aplicação, onde a *__construct* só permite que seja carregada a página do *controller* de aluno se tiver um usuário logado no sistema, enquanto a função de index, fica responsável por chamar a *view* de aluno. E, por fim, a função *store* recebe um aluno como parâmetro e faz a inserção do mesmo no banco de dados, e retorna para o usuário à função *index* que chama a *view* de aluno se foi inserido com sucesso, caso não tenha êxito na inserção retorna uma mensagem de falha.

Já os arquivos denominados como *models* são responsáveis pela definição das tabelas do banco de dados, as mesmas criadas através das *migrations*. Na *model* é definido dois atributos, um contendo o nome da tabela e o outro os seus campos. Diferentemente de um banco de dados tradicional onde as relações entre tabelas são

criadas na própria tabela, no Laravel é a *model* que fica responsável por definir o tipo de relação de uma tabela com a outra (1:1, 1:N ou N:M). Conforme demonstrado no trecho de código 6.

Código 6: Model

```
class Aluno extends Model {
  protected $table = 'aluno';
  protected $fillable = ['nome',
'data_nascimento','cpf','telefone','celular','email','rua','numero
','bairro','cidade','cep','estado', 'dataini','users_id'];

  public function fatura () {
    return $this->hasMany(Fatura::class);
  }
}
```

Fonte: Dos próprios autores.

Outro arquivo importante, porém, que não é exclusivo do Laravel é o arquivo script.js, arquivo criado para conter funções JavaScript e JQuery que foram utilizadas no projeto. Esse arquivo tem uma grande relevância, pois apesar do banco de dados e do próprio Laravel realizar validações de alguns campos, como e-mail e senha, outros campos presentes nos formulários de cadastro se faz necessário validação, não somente se foram informados corretamente, mas sim se são realmente válidos ou se já existem cadastrados no banco de dados como é o caso do cpf.

Esse arquivo realiza complementos para uma melhora nos cadastros e na estética para o usuário, como no caso dos campos de endereço, onde ao informar o cep, o sistema retorna os outros campos que completam o endereço do aluno ou do instrutor, e mensagens de confirmação de cadastros realizados, para melhor compreensão do usuário.

Os trechos de códigos a seguir exibem as respectivas funções citadas acima, sobre as funções javascript que complementam e melhoram o sistema.

Código 7: Função para preencher endereço através do cep

```
$(document).ready(function() {
  function limpa_formulário_cep() {
    $("#cep").val("");
    $("#estados").val("");
    $("#cidade").val("");
    $("#bairro").val("");
    $("#rua").val("");
  }
  $("#cep").blur(function() {
    var cep = $(this).val().replace(/\D/g, '');
    if (cep != "") {

      var validacep = /^[0-9]{8}$/;
      if(validacep.test(cep)) {
        $("#rua").val("...");
        $("#bairro").val("...");
        $("#cidade").val("...");
        $("#estados").val("...");
        $("#ibge").val("...");
        $.getJSON("//viacep.com.br/ws/"+ cep +"/json/?callback=?",
function(dados) {
          if (!("erro" in dados)) {
            $("#rua").val(dados.logradouro);
            $("#bairro").val(dados.bairro);
            $("#cidade").val(dados.localidade);
            $("#estados").val(dados.uf);
            $("#ibge").val(dados.ibge);
          } else {
            limpa_formulário_cep();
            alert("CEP não encontrado.");
          }
        });
      } else {
        limpa_formulário_cep();
        alert("Formato de CEP inválido.");
```

```
        }
    } else {
        limpa_formulário_cep();
    }
  });
});
```

Fonte: Dos próprios autores.

Essa função realiza uma consulta no viacep.com.br assim que o usuário retira o foco do campo cep, nessa busca é retornado e preenchido automaticamente os valores dos campos estado, cidade, bairro e rua, isso quando a cidade possui mais de um cep, e quando possui apenas um a função preenche apenas os campos de estado e cidade, deixando bairro e rua para que o próprio usuário preencha. Enquanto a pesquisa é realizada trecho com "*if*(validacep.test(cep))" preenche os inputs do formulário com três pontinhos até que os valores retornem da consulta e preencham os campos, o código também verifica se o formato do cep digitado está correto, caso não esteja, retorna uma mensagem informando que o formato do cep é inválido.

Como dito acima, funções JavaScript também foram criadas para realizar validações de alguns campos no banco de dados, como exibe o trecho de código 8.

Código 8: Função validação

```
function valida(campo,valor) {
    var campo2="#"+campo;
    var tabela;

    //Verifica por classe se é do form Instrutor ou ALuno
    if($(campo2).hasClass("instrutor")){
        tabela = "instrutor";
    }else if($(campo2).hasClass("aluno")){
        tabela = "aluno";
    }
```

```
    $(campo2).change(function () {
        $(campo2).removeClass("alert-danger");
    });

    if(!$(campo2).hasClass("alert-danger")){
        $.ajax({
            type: "POST",
            url: "email.php",
            dataType: "html",
            data: {'campo':campo, 'valor':valor,'tabela':tabela },
            success: function(response){
                if(response === '0'){     // NO ARQUIVO PHP ELE
VAI RETORNAR 0 caso o campo não exista no banco de dados
                    if ($("#email").hasClass("alert-danger")||
$("#cpf").hasClass("alert-danger")){
                      $("#btn").attr("disabled","disabled");
                    }else{
                      $('#btn').removeAttr('disabled');
                      $(campo2).removeClass('alert-danger');
                    }
                }
                if(response === '1'){     // NO ARQUIVO PHP ELE
VAI RETORNAR 1 caso o campo exista no banco de dados
                    $(campo2).addClass('alert-danger');
                    $('#btn').attr('disabled','disabled');
                }

            }
        });
    }
}
```

Fonte: Dos próprios autores.

A função para validar a existências dos campos de CPF e EMAIL no banco de dados, funciona da seguinte forma: espera dois parâmetros, o valor a ser consultado no banco e o id do campo a que esse valor se refere, assim que inicializada ela pega

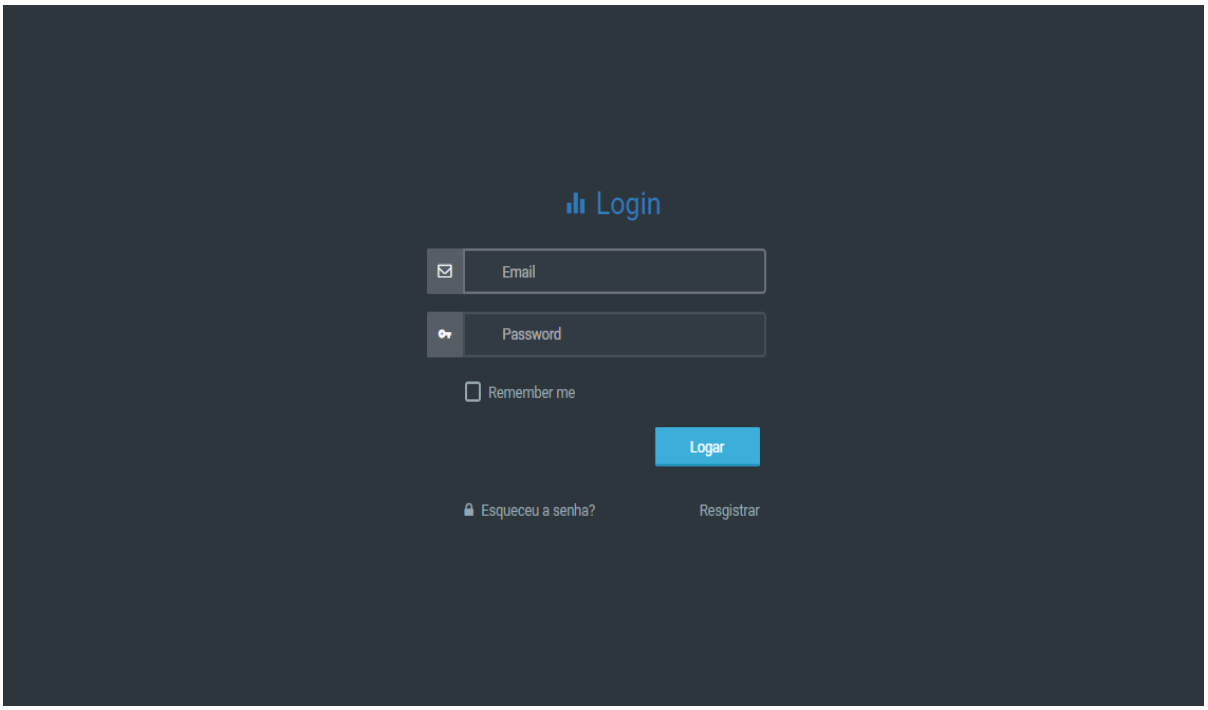
o id do campo que foi recebido e o verifica se existe a classe “aluno” ou “instrutor” e adiciona em uma variável tabela para alimentar a consulta, pois é uma função genérica que serve para os dois formulários. Após isso ela faz uma requisição via ajax ao documento email.php, onde passa três parâmetros: campo, valor e a tabela, os parâmetros vão ser enviados no tipo POST e com os dados do tipo html. Na função email.php, recebe estes valores abre uma conexão com o banco de dados e monta uma sql de consulta no mesmo, de acordo com a resposta dessa consulta o email.php irá retornar ao ajax 0 ou 1, sendo 0 caso o campo não exista e 1 caso exista.

Assim, a função ajax de acordo com a resposta acusa o erro no campo que existe e desativa o botão cadastrar, impedindo que o usuário tente enviar valores iguais ao banco ou remova o impedimento do botão caso o mesmo atualize os campos com valores validos.

Agora que foi explicado o que cada componente é responsável dentro do projeto será descrito as *views* que o projeto possui, dando ênfase nas telas com maior relevância para o projeto.

A tela inicial do sistema é a tela de login de usuário, é responsável por conceder ao usuário acesso ao sistema solicitando e-mail e senha, contanto que o mesmo já esteja cadastrado no banco de dados. A figura 15 exibe a tela de login do sistema.

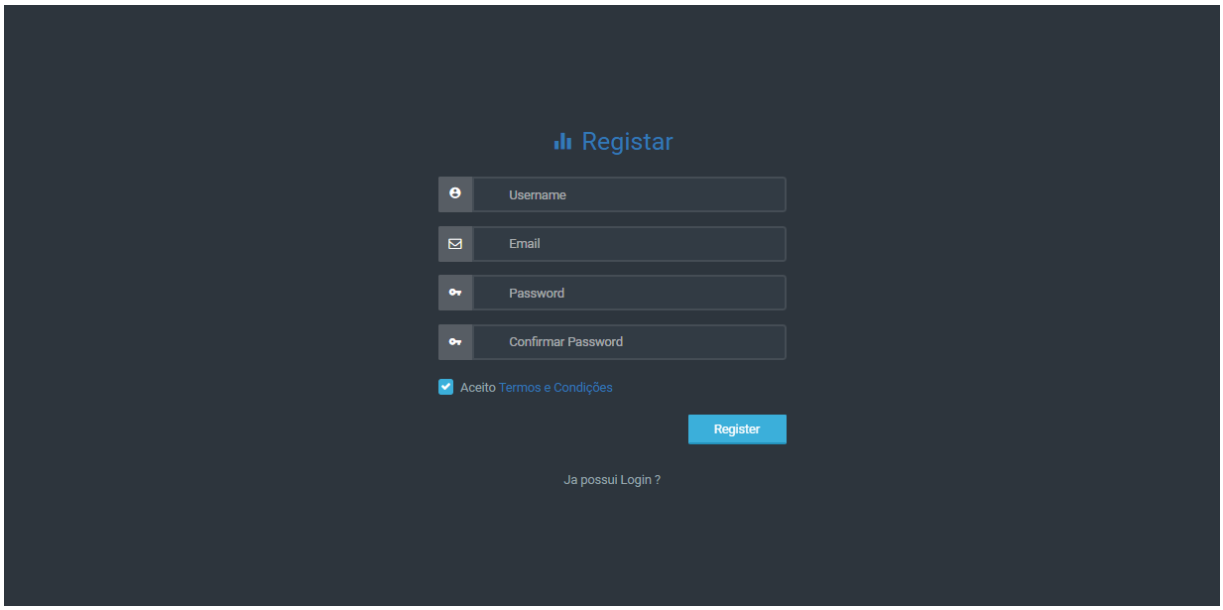
Figura 15: Tela de login



Fonte: Dos próprios autores.

Caso o usuário não possua cadastro no banco, seu login não será permitido, dessa forma o mesmo deve se registrar, informando os dados necessários para o cadastro de usuário, que são: nome, e-mail, senha e confirmação da senha. A figura 16 exibe a tela de registro de usuário.

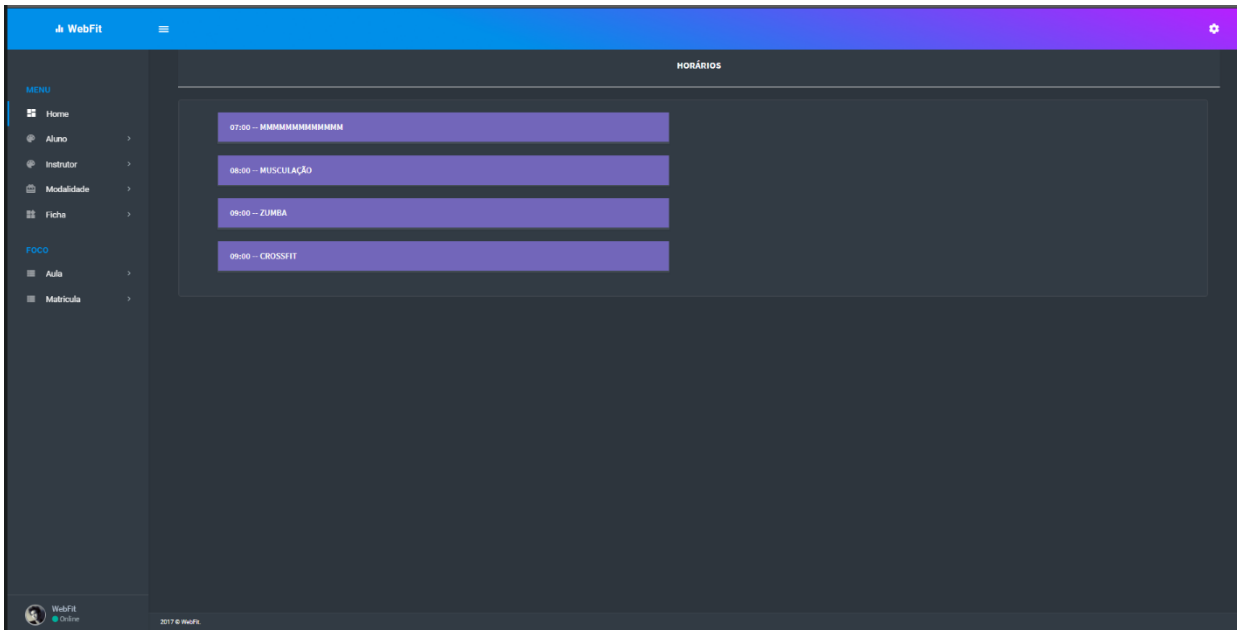
Figura 16: Tela de Registro



Fonte: Dos próprios autores.

Assim que o acesso ao usuário for permitido, será redirecionado para a tela home do sistema, que possui um menu lateral com as opções de navegação pelo sistema, que o da permissão de cadastrar e listar alunos, instrutores, modalidades, fichas, aulas e matrículas. A home também possui uma área onde já será exibido os horários de aulas marcadas, devido ser o ponto central do sistema, exibir esses horários para o usuário, foi acordado o mesmo já estar localizado na tela inicial assim que o usuário logar no sistema. Conforme demonstrado na figura 17.

Figura 17: Tela Home do sistema



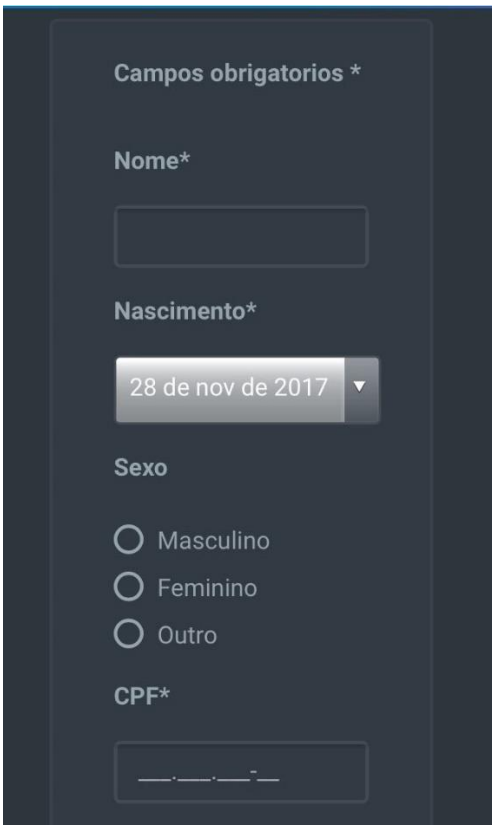
Fonte: Dos próprios autores.

As telas de cadastro do sistema WebFit são bem simples, todas são encontradas através do menu lateral, escolhendo a opção desejada entre aluno, instrutor, modalidade, ficha, aula e matrícula, ao click do mouse é aberto um sub-menu em cada uma delas com opções de cadastro e listagem.

As telas de cadastro são relativamente parecidas, seguindo o mesmo designer, com formatação dos campos e botões iguais para todas, assim como sua exibição, a seguir será exemplificado através de figuras as 3 (três) telas de cadastro com maior relevância para o sistema.

A primeira tela a ser mostrada é a de cadastro de aluno, já que o sistema deve exibir quais alunos estão matriculados em determinadas aulas, ele tem grande relevância, e as diferenças desse cadastro para o cadastro de instrutor, modalidade e ficha, são mínimas.

Figura 18: Tela de Cadastro de aluno responsiva

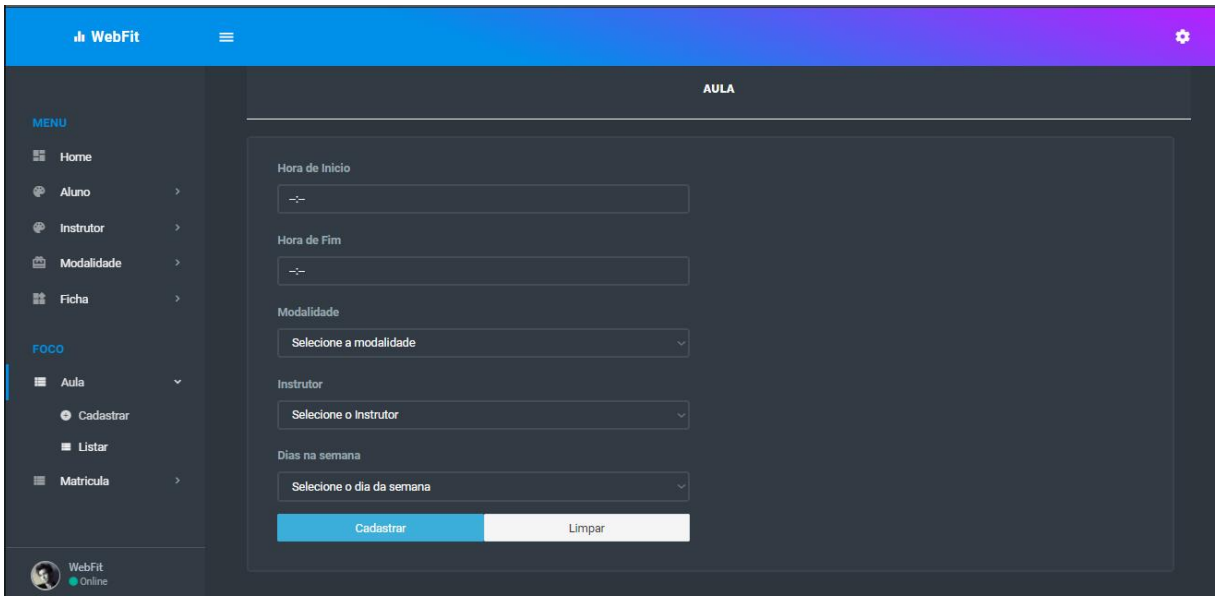


Fonte: Dos próprios autores.

A tela de cadastro de aula é responsável por criar uma aula, onde conterá informações de outras tabelas, como modalidade e instrutor. Nessa view contém dois campos de horários que são responsáveis pelos horários de início e fim de uma aula, que possui duração de 1 (uma) hora cada. Essa tela possui 3 (três) campos do tipo *select*, onde o usuário seleciona um instrutor e uma modalidade que já foram cadastrados no banco de dados para serem integrados a essa aula, o terceiro campo é responsável por escolher qual dia da semana aquela aula pertence podendo escolher de segunda-feira à sábado.

É valido ressaltar que para a criação de uma aula, necessita apenas de um aluno. A figura 19 representa a tela de cadastro de aula.

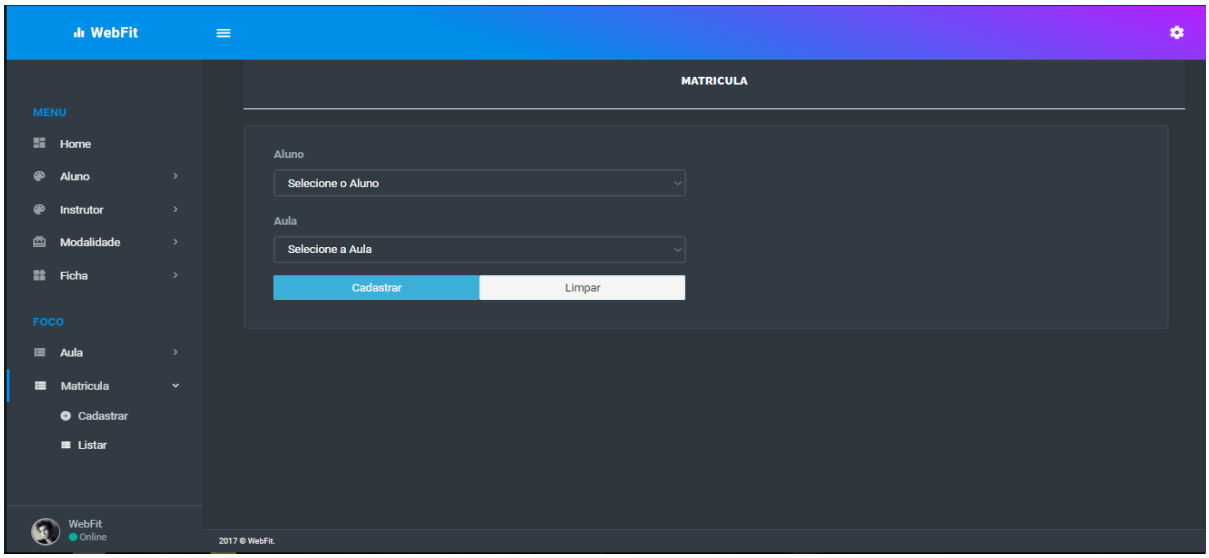
Figura 19: Tela de Cadastro de aula



Fonte: Dos próprios autores.

E, por fim, a view de matricula que é responsável por matricular um aluno em uma aula, esta tela possui apenas dois campos do tipo *select* que trazem do banco de dados as aulas e os alunos já cadastrados, selecionando o aluno e matriculando-o em uma aula. É valido ressaltar que uma aula pode ter no máximo 5 (cinco) alunos cadastrados nela.

Figura 20: Tela de Matricula



Fonte: Dos próprios autores.

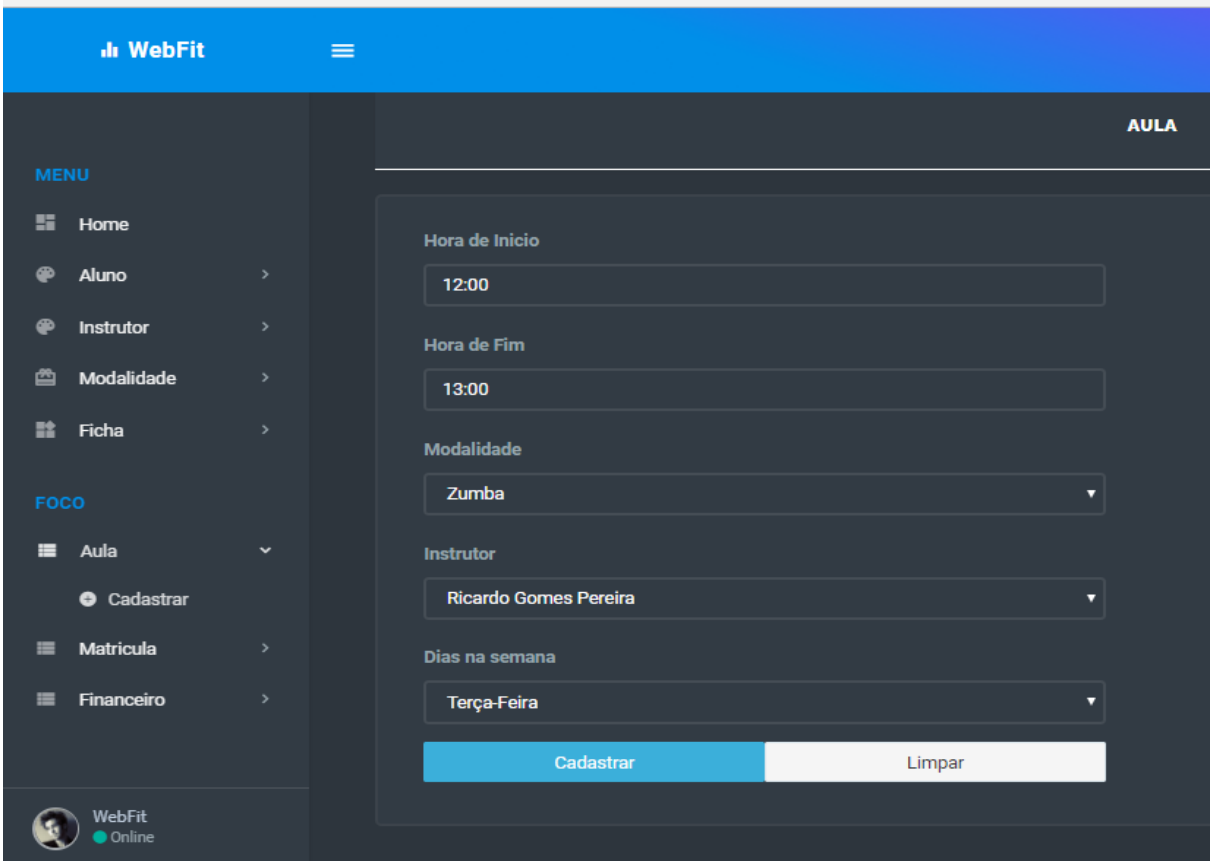
3.1.4 Testes do sistema

Neste subcapitulo será abordado o teste do sistema a fim de comprovar o objetivo do desenvolvimento do mesmo, ressaltando que devido ao fechamento da empresa a qual o software seria desenvolvido inicialmente, cabido aos requisitos serem utilizáveis para outras empresas, foi possível adaptar o sistema para a mesma e realizar os testes em outra academia.

Devido o foco do sistema web ser gestão de horários de aulas para academias, será efetuado a inclusão de um aluno em uma determinada aula, que deve ser exibida para o usuário do sistema na tela *home*.

Na tela de cadastro de aula será apresentado as informações de instrutor e de modalidades cadastradas no banco de dados, onde cabe ao usuário selecionar a modalidade da aula desejada, assim como o instrutor que conduzirá a aula; o usuário deve informar também o dia e o horário desta aula, como mostra a figura 21.

Figura 21: Cadastrando uma aula

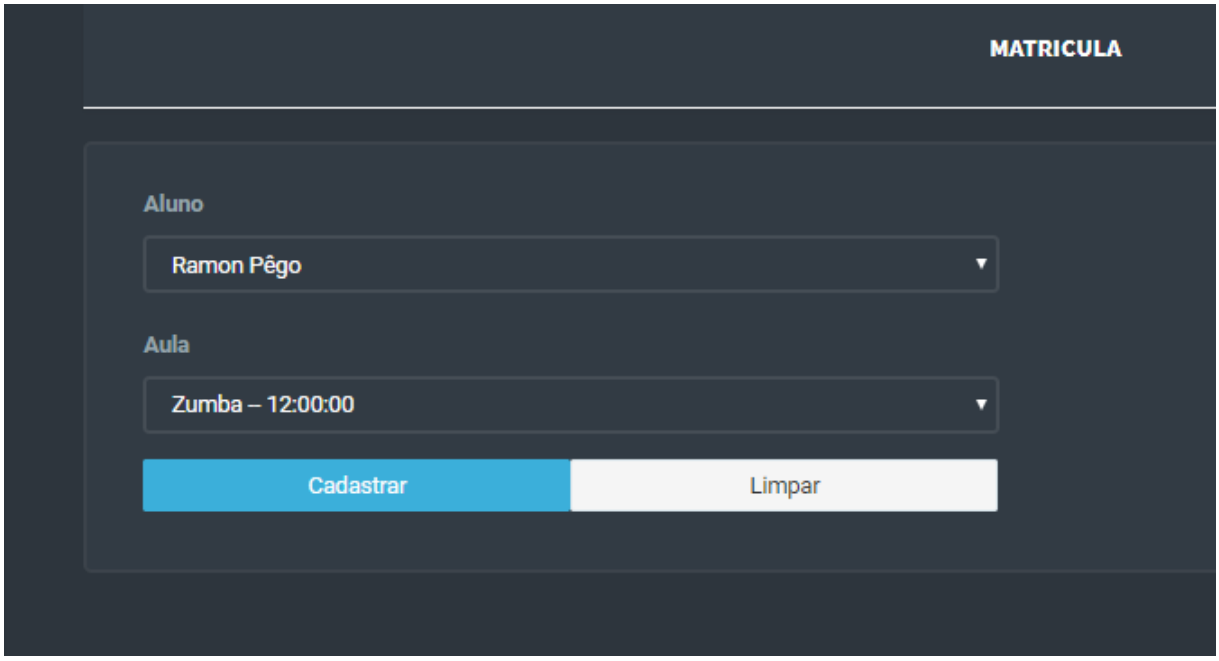


Fonte: Dos próprios autores.

Após a conclusão do cadastro da aula, o usuário deve ir à seção “Matrícula” para finalmente cadastrar o aluno em alguma aula que anteriormente foi cadastrada pelo usuário.

Nesta tela o usuário irá encontrar as duas caixas de seleção onde será exibido todos os alunos, que podem ser matriculados em alguma aula, e todas as aulas disponíveis, conforme exibe a figura 22.

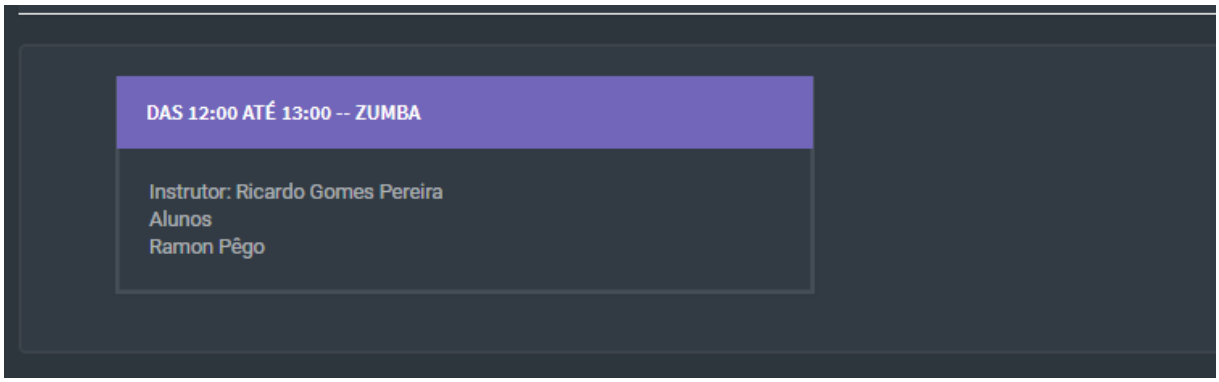
Figura 22: Cadastrando aluno na aula



Fonte: Dos próprios autores.

Após a finalização da matrícula de um determinado aluno em uma aula, o usuário poderá se dirigir à tela “home” do sistema, onde é exibido todas as aulas e seus respectivos dados, gerando as informações necessárias para auxílio do controle de horários das aulas ofertadas pelo estabelecimento. A figura 23 exibe uma aula cadastrada.

Figura 23: Exibição de horário



Fonte: Dos próprios autores.

4 RESULTADOS OBTIDOS

Este capítulo tem por finalidade apresentar os resultados obtidos após a utilização do sistema WebFit em academia com finalidade de testar a eficiência do sistema.

Em um estabelecimento do ramo de atividades físicas como academias de musculação, sabe-se que é necessário ter um controle das pessoas que frequentam a mesma, para que se tenha esse controle um cadastro se faz necessário. Atualmente grande parte das academias oferecem tipos de práticas variadas, como aulas de artes marciais, de dança e outras, para esse tipo de serviço ofertado, uma forma de marcar e controlar essas aulas é necessária, devido ao fato de acontecerem em horários diferentes, com alunos diferentes.

Portanto, para que uma academia deixe a utilização de gerenciamento manual, com papel e/ou planilhas, um sistema que atenda essas especificações e não dependa que a empresa armazene arquivos físicos de seus clientes e alunos, foi desenvolvido um sistema web, capaz de realizar cadastros dos alunos que se matriculam em academias e também a criação e controle de aulas que podem ser utilizadas para as mais variadas práticas de serviços oferecidos pela empresa. O sistema WebFit foi testado por uma academia localizada na cidade de Setubinha – MG, a empresa em questão teve um prazo determinado para utilização do sistema web, onde após o vencimento seria submetido um questionário com questões relacionadas ao sistema com objetivo de levantar opiniões e possíveis melhorias para a aplicação, questionário encontra-se no APÊNDICE I.

Após o teste do sistema WebFit em academia do público alvo, e analise do questionário, o mesmo apresentou algumas vantagens, sendo elas:

- Facilidade em gerir aulas com determinados alunos e instrutor;
- A praticidade e rapidez para realizar cadastros dos dados dos clientes da

academia;

- Utilização em qualquer dispositivo, desde que o usuário possua acesso à internet;

Assim como os resultados obtidos evidenciam as vantagens, também é exibido os pontos negativos do sistema, sendo eles:

- Dificuldade na utilidade do sistema inicialmente;
- Falta da funcionalidade para controle do financeiro;
- A inutilidade do sistema, quando o acesso à internet não é possível;

O sistema WebFit proporciona aos usuários e gerentes de academias utilizar um sistema informatizado atual e relevante para o ramo em que se encontram no mercado, trazendo agilidade, praticidade e eficiência para gestão da empresa, e um melhor controle do que acontece na mesma. É claro que melhorias devem ser realizadas para que o sistema se mantenha útil, como por exemplo um módulo de controle financeiro como foi sugerido pelo usuário ao responder o questionário.

CONSIDERAÇÕES FINAIS

O projeto desenvolvido nesta monografia, cujo objetivo é o desenvolvimento de um sistema web para auxiliar gestores de academias em relação ao controle de horários de aulas ofertadas. No decorrer do desenvolvimento deste projeto foi proporcionado aos desenvolvedores adquirir conhecimentos nas linguagens e ferramentas de programação, que facilitam e agilizam o dia-a-dia de um programador, tal como é feita a condução e organização de um projeto através da Engenharia de Software. Desta forma o projeto foi planejado, executado e finalizado em um curto período de tempo.

Para alcançar o objetivo geral do trabalho, fez-se necessário percorrer por diversos objetivos específicos que ao conquistá-los nos aproximava mais do objetivo principal do projeto. A seguir serão listados os objetivos específicos utilizados:

Compreender as regras de negócio para o levantamento dos requisitos sobre as funcionalidades que devem ser contempladas pelo software:
Para o desenvolvimento de qualquer software é necessário buscar e conhecer quais são as necessidades do cliente, assim como o que ele realmente precisa que o software contenha, para isso foram realizados diálogos com o gerente da empresa e realizado um levantamento de requisitos, estes estão disponíveis na sessão “3.1.1 Requisitos”, na página 47.

Garantir a plena adequação do sistema às necessidades reais da empresa em questão:
Dever dos desenvolvedores conduzirem o desenvolvimento do software de forma que o mesmo se adeque às necessidades da empresa, caso contrário o sistema não teria razão de existir, seguindo os requisitos coletados nas entrevistas com o cliente e utilizando a Engenharia de Software para aplica-los no projeto da melhor maneira.

Através de análise, garantir que o software sucessor contemple as necessidades mínimas do antecessor:

Garantir que o sistema WebFit possuísse as mesmas funções que eram encontradas nas práticas manuais da empresa como foi constatado no processo de levantamento de requisitos. Dessa forma, o sistema deveria informar ao usuário os horários de aulas cadastrados, como era feito no método anterior, através de tabelas em uma planilha.

Através da análise das funcionalidades do software anterior, compreender as falhas que o tornaram inviável para empresa, evitando assim que o novo software possua tais erros:

Descobrir o que causava transtorno e dificuldade para os usuários com softwares não específicos para suas necessidades através das entrevistas onde os requisitos foram levantados. Tendo que se adaptarem a planilhas e/ou serviços manuais como fichas cadastrais em papel, desta forma o sistema foi desenvolvido para facilitar essas atividades, além de torná-las mais eficiente e seguras.

Compreender as vantagens de se possuir um sistema web e garantir o entendimento do cliente sobre os benefícios que o mesmo trará a empresa:

Um sistema informatizado voltado para a web seria de grande benéfico para as empresas devido sua facilidade de acesso e utilização desde que a mesma possua internet, além de possuir todas as informações que uma gerencia manual teria, tem uma segurança maior para a empresa já que os dados inseridos na mesma ficam armazenados em nuvem.

Garantir que o sistema possua um local de hospedagem seguro e de qualidade, além de um domínio, evitando eventuais problemas relacionados a acesso e funcionamento do sistema na web:

O domínio atribuído para o sistema web desenvolvido foi “webfit.life” e o servidor em que o mesmo se encontra em funcionamento é o servidor da Amazon.

Aplicação de um ambiente de testes para que seja descoberto falhas e erros nas partes já desenvolvidas do software e corrigi-las antes de chegar até as mãos do usuário final:

Um ambiente de testes para o cliente não foi possível de ser projetado, por diversos fatores, porém foi criado um ambiente de desenvolvimento em que os desenvolvedores tiveram oportunidades de testar as funcionalidades desenvolvidas antes que o sistema chegasse a ser testado pelo cliente.

Desenvolver o software de forma iterativa e incremental, sempre em

contato constante com o cliente, garantindo que ele compreenda as funcionalidades desenvolvidas e garantir que estão atendendo a sua necessidade:

Desenvolver os testes aos poucos e sempre estar em constante contato com o cliente, facilitou algumas mudanças que ocorreram durante a execução do projeto, como foi o caso em que a empresa inicialmente contratante do projeto teve seu fechamento. E o sistema foi adaptado e desenvolvido para academias de forma geral, devido à grande compatibilidade com o ramo da empresa para qual inicialmente seria desenvolvido, evitando que funcionalidades inadequadas fossem implementadas.

Garantir que a gestora e os demais usuários aprendam a utilizar o sistema corretamente através de treinamento, evitando utilização errada do software que possa vir a torná-lo inviável para empresa:

Antes de iniciar os testes por parte do cliente, foi devidamente apresentado o sistema para o mesmo, explicando todas as telas e as funcionalidades encontradas em cada tela da aplicação. Onde na primeira utilização do sistema os desenvolvedores, juntamente com o cliente, foram passando por todas as telas do sistema, explicando qual a função de cada uma delas e a forma com que o usuário deveria utilizar a mesma.

Analisar e compreender o feedback do cliente sobre a apresentação/teste do sistema completo e corrigir pontos solicitados pelo mesmo:

Ao final dos testes foi apresentado ao cliente um questionário com objetivo de descobrir o que o mesmo achou do sistema, assim como descobrir possíveis erros, falhas e sugestões de melhorias, como por exemplo novas funcionalidades.

É relevante ressaltar que apesar do fechamento da empresa por motivos pessoais, a qual o sistema inicialmente seria desenvolvido, os requisitos levantados pela mesma poderão ser utilizados e adaptados para academias em geral, de modo que os dois tipos de empresas seguem uma mesma linha de mercado, o que facilitou a adaptação tanto dos requisitos quanto do sistema, possibilitando que os testes fossem realizados sem nenhum problema e as hipóteses fossem validadas ou invalidadas.

Diante do objetivo do projeto que era o desenvolvimento de um sistema web para controle de horários em academias, para melhor auxílio dos usuários na gestão do estabelecimento, foi realizado testes com objetivo de analisar o desempenho do sistema para validar ou não as hipóteses levantadas.

H0: Não seria viável o desenvolvimento deste sistema, pois a gestão da empresa por uso de planilhas do Microsoft Excel viria logrando o êxito esperado pela gestora.

A hipótese descrita acima **não foi validada**, devido que a proposta de desenvolvimento do sistema web foi justificado pois, métodos inadequados utilizados pela empresa, por meio de gestão manual através de planilhas não ofereciam eficiência a ponto da empresa se manter utilizando apenas este método. Diante disso o desenvolvimento do sistema web foi proposto para que a empresa tivesse uma gestão adequada que atendesse suas necessidades e expectativas.

H1: O desenvolvimento do sistema não seria viável, pois não traria maior segurança para os dados da empresa do que uma planilha feita no Microsoft Excel.

A hipótese h1 **não foi validada**, diante que a segurança de dados transcritos em planilhas e/ou em papéis não possui uma segurança apropriada para arquivamento de informações de clientes de uma empresa. Portanto, o sistema web com armazenamento dos dados em banco de dados hospedado em um servidor nas nuvens possui uma segurança muito maior, sendo necessário autorização através de *login* e senha para se ter acesso aos dados.

H2: A implantação do sistema seria viável para que a empresa consiga se organizar melhor, contemplando uma administração de qualidade e atualizada.

A hipótese acima é **válida**, devido que os resultados dos testes realizados, mostraram que a utilização do sistema WebFit traz benefícios e facilidade para gerenciamento da administração da empresa, tendo os dados e informações armazenados em banco de dados e podendo consultar os mesmos sempre que necessário, além de se manter atualizada com um sistema informatizado.

H3: O desenvolvimento e implantação da aplicação web não seria viável, pois a empresa não conseguiria se manter no mercado, que está cada vez menos interessado no nicho de mercado em questão.

A hipótese acima **não foi validada**, devido que o ramo de práticas esportivas sempre se fez presente no mercado e dificilmente deixará de fazer. Pessoas praticam

atividades físicas por inúmeras razões, que as levam buscar academias que lhes proporcione atingir essas razões. Portanto, a implantação do sistema web é viável devido que, mesmo diante do fechamento da primeira empresa, foi possível a implantação do sistema em outra, e após a implantação, ficou comprovado através dos testes executados pelo cliente os benefícios e relevância que o sistema trouxe para a empresa, facilitando gerir o controle de aulas, visto que a gestão da empresa se fazia por meios manuais.

Para prosseguir com a viabilidade do sistema perante o mercado, novas funcionalidades poderão ser desenvolvidas, como controle financeiro e controle de acesso através de catracas, utilizando senhas numéricas ou biométricas.

REFERÊNCIAS

BALDUÍNO, Plínio. *Dominando JavaScript com JQuery*. s/l: Editora Casa do Código. 2014.193p. Disponível em: <https://www.casadocodigo.com.br/products/livro-javascript-jquery>. Acesso em: 22 mai. 2017.

BOTELHO, Fernando. *As Novidades do HTML5*. Disponível em: < http://www.devmedia.com.br/as-novidades-do-html5/23992>. Acesso em 23 de ago. 2017.

CAYRES, Paulo Henrique. *Modelagem de Banco de Dados. Rio de Janeiro: Escola Superior de Redes*, 2015. 164p.

CARVALHO, Vinícius. *MYSQL: Comece com o principal banco de dados open source do mercado*. São Paulo: Casa do Código, 2015. 172p. Disponível em: < https://www.casadocodigo.com.br/products/livro-banco-mysql>. Acesso em: 21 mai. 2017.

DUCKET, Jon. *JavaScript & JQuery*: *Desenvolvimento de interfaces web interativas.* Rio de Janeiro: Alta books. 2015. 640p.

ELMASRI, Ramez; NAVATHE, Shamkant B. *Sistemas de banco de dados*. 4. ed. São Paulo: Pearson Addison Wesley, 2005. 730p.

FERREIRA, Silvio. *Guia Prático de HTML5.* São Paulo: Universo dos Livros Editora, 2013. 168p. Disponível em: < https://www.amazon.com.br/Guia-Prático-HTML5-Silvio-Ferreira-ebook/dp/B00DJ6QF5Q/ > Acesso em: 24 mar. 2017.

GOMES, Eduardo Henrique. *Sistema Gerenciador de Banco de Dados*. Disponível em: <http://ehgomes.com.br/disciplinas/bdd/sgbd.php> Acesso em: 24 mai. 2017.

JUNIOR, Walteno Martins Parreira. *Apostila Engenharia De Software*. Disponível em:< http://www.waltenomartins.com.br/es_aps.pdf> Acesso em: 25 mai. 2017.

LONGMAN, Addison Wesley.*Chapter 2. A history of HTML*, 1998. Disponível em: <https://www.w3.org/People/Raggett/book4/ch02.html> Acesso em: 24 mar. 2017.

MAZZA, Lucas. HTML5 e CSS3: Domine a web do futuro. s/l: Editora Casa do Código, 2014. 217p. Disponível em: <https://www.casadocodigo.com.br/products/livro-html-css>. Acesso em: 22 mai. 2017.

MIYAGUSKU, Renata Hiromi Minami. D*esvendando os Recursos do CSS*. Universo dos Livros Editora, 2007. s/p. Disponível em: < https://www.amazon.com.br/Desvendando-os-recursos-do-CSS-ebook/dp/B00DN8P754/>. Acesso em 21 mar. 2017.

NIEDERAUER, Juliano*. Desenvolvendo websites com PHP: aprenda a criar websites dinâmicos e interativos com PHP e bancos de dados*. 3. Ed. São Paulo: NOVATEC, 2017. 320p.

NOLAN, Ashley. *The State of Front-End Tooling*.2015. Disponível em: <https://ashleynolan.co.uk/blog/frontend-tooling-survey-2015-results>. Acesso em 23 de ago. 2017.

PRESSMAN, Roger S. *Engenharia de Software*. 6ª edição. São Paulo: McGraw-Hill, 2006.

_____. *Engenharia de Software: Uma Abordagem Profissional*. 7ª edição. São Paulo: McGraw-Hill, 2011. 780p.

SCHEIDT, Felippe Alex. *Introdução ao jQuery*. s/l: Editora Itacaiunas, 2015. Disponível em:<https://www.amazon.com.br/Introdução-jQuery-Felippe-Alex-Scheidt-ebook/dp/B00X3AJMA8>. Acesso em: 28 abr 2017.

SILVA, Maurício Samy. *Bootstrap 3.3.5*. *Aprenda a usar o framework Bootstrap para criar layouts CSS complexos e responsivos*. São Paulo: Novatec Editora Ltda. 225p. Disponível em: <http://livrosdomaujor.com.br/bootstrap3/codigos.html> Acesso em: 26 mai. 2017.

SOMMERVILLE, Ian. *Engenharia de Software.* 8ª edição. São Paulo: Pearson

Addison- Wesley, 2007. 552p.

TURINI, Rodrigo. *PHP e Laravel: crie aplicações web como um verdadeiro artesão*. Editora Casa do Código, 2015. 219p. Disponível em <https://www.casadocodigo.com.br/products/livro-laravel-php>. Acesso em 21 set. 2017.

ZAMEL, Tárcio. *Web design responsivo.* s/l: Editora Casa do Código, 2012.160p. Disponível em < https://www.casadocodigo.com.br/products/livro-web-design-responsivo>. Acesso em 23 de ago. 2017.

APACHE HTTP SERVER PROJECT. *What is the Apache HTTP Server Project*?. Disponível em < http://httpd.apache.org/ABOUT_APACHE.html> Acesso em: 25 de ago. 2017.

CAKE PHP. *Entendendo o Model-View-Controller (MVC)*. Disponível em: < https://book.cakephp.org/1.3/pt/The-Manual/Beginning-With-CakePHP/Understanding-Model-View-Controller.html> Acesso em: 24 de ago. 2017.

BOTSTRAP. About. Disponível em: <https://v4-alpha.getbootstrap.com/about/license/>. Acesso em 14 set. 2017.

HOTFRAMEWORKS. *Find your new favorite web framework*. Disponível em < https://hotframeworks.com/> Acesso em: 24 de ago. 2017.

MICROSOFT. *Visão geral da função de Servidor Web (IIS)*. Disponível em: < https://technet.microsoft.com/pt-br/library/cc770634(v=ws.11).aspx>. Acesso em 26 de ago. 2017.

MARIADB. *Sobre o Maria DB*. Disponível em: < https://mariadb.com/kb/pt-br/sobre-o-mariadb/> Acesso em:10 set. 2017.

NODEBR. *JavaScript no servidor com Node.js*. Disponível em:< http://nodebr.com/javascript-no-servidor-com-node-js/ >. Acesso em 21 ago. 2017.

OFICINA DA NET. *O que é um Servidor web*?. Disponível em: <https://www.oficinadanet.com.br/artigo/servidores/o_que_e_um_servidor_web> Acesso em: 25 de ago. 2017.

PHP. *O que o PHP pode fazer*?. Disponível em: <https://secure.php.net/manual/pt_BR/intro-whatcando.php>. Acesso em 24 de ago. 2017.

UNIVERSIDADE FERDERAL DE CAMPINA GRANDE. *Frameworks: O que é um framework?*. Disponível em: <http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/frame/oque.htm>. Acesso em 14 set. 2017.

_____. *Casos de Uso: diagrama de casos de uso*. Disponível em < http://www.dsc.ufcg.edu.br/~sampaio/cursos/2007.1/Graduacao/SI-II/Uml/diagramas/usecases/usecases.htm >. Acesso em: 15 de set. 2017

WORLD WIDE WEB CONSORTIUM. *A Briefl History of CSS until* 2016. Disponível em: <https://www.w3.org/Style/CSS20/history.html> Acesso em: 22 mar. 2017.

_____. *A Short History of JavaScript.* Disponível em:< https://www.w3.org/community/webed/wiki/A_Short_History_of_JavaScript>. Acesso em: 11 de mai. 2017.

_____. *Usage of server-side programming languages for websites*. Disponível em <https://w3techs.com/technologies/overview/programming_language/all>. Acesso em: 25 de ago. 2017.

UNIVERSIDADE FEDERAL DE CAMPINA GRANDE. Casos de Uso: diagrama de casos de uso. Disponível em < http://www.dsc.ufcg.edu.br/~sampaio/cursos/2007.1/Graduacao/SI-II/Uml/diagramas/usecases/usecases.htm >. Acesso em 23 de out. 2017.

APÊNDICE I

QUESTIONÁRIO DE TESTE DO SISTEMA

Sistema WebFit

O objetivo deste questionário é obter um feedback do cliente/usuário sobre a utilização do sistema.

Nome do proprietário (a) *

[redacted]

Nome da academia

Academia Fatal Fitness

E-mail:

academiafitnesssetubinha@outlook.com.br

Quais foram as maiores dificuldades encontradas para manoseio do sistema? *

dificuldades em operar o sistema no inicio

Foi encontrado algum erro em alguma funcionalidade do sistema? *

Não! Nas funcionalidades que estavam prontas, funcionaram todas corretamente.

O sistema auxiliou no processo de gerenciamento dos horários da academia? *

Sim.

Qual(is) funcionalidade(s) que mais chamou atenção no sistema?

A tela inicial do sistema, mostra informações importantes relacionadas a academia.

Que nota daria para o sistema? *

	1	2	3	4	5	
1 Muito ruim	○	○	○	◉	○	5 Muito bom

Sobre aparência do sistema *

	1	2	3	4	5	
1 Aparência não agrada em nada.	○	○	○	○	◉	5 Aparência agradável.

Probabilidade de aquisição desse sistema *

	1	2	3	4	5	
1 Não adquiriria esse sistema	○	○	○	○	◉	5 Com certeza adquiria esse sistema

Você indicaria esse sistema para outras academias? *

	1	2	3	4	5	
Sendo 1 não indicaria	○	○	○	○	◉	5 certamente indicaria

Você contrataria esse sistema para que a sua academia possua um sistema informatizado? Justifique. *

Sim. Pois todos os processos na academia são feitos de forma manual através de papeis,então o sistema ajudou bastante.

Em qual dispositivo acessou o sistema WebFit? *

- ( ) Desktop(Computador de mesa)
- (•) Notebook (Ou Laptop e outros do genero)
- ( ) Celular
- ( ) Tablet
- ( ) Outro: .....

Na sua opinião esse sistema tem maior relevância do que controle de gerencia manual ou por softwares como Microsoft Excel? Justifique *

Sim em partes. por que Se faltar internet não consigo acessar o sistema!

Sugestões de melhoria para o sistema: *

Colocar uma forma de controlar o financeiro.

Informações adicionais sobre sua experiencia com o sistema: *

Bom Sistema, supri as necessidades da minha academia.