UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

VINICIUS DALSASSO

SISTEMA WEB PARA CONTROLE DE JORNADA DE TRABALHO DE MOTORISTAS PROFISSIONAIS

TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO

PATO BRANCO
2020

VINICIUS DALSASSO

SISTEMA WEB PARA CONTROLE DE JORNADA DE TRABALHO DE MOTORISTAS PROFISSIONAIS

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Conclusão de Curso 2, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, como requisito parcial para obtenção do título de Tecnólogo.

Orientadora: Profa. Rúbia Eliza de Oliveira Schultz Ascari

PATO BRANCO
2020

Ministério da Educação
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
UTFPR - CAMPUS PATO BRANCO
DIRETORIA GERAL - CÂMPUS PATO BRANCO
DIR. DE GRAD. E EDUCAÇÃO PROFISSIONAL-PB
DEP. ACADÊMICO DE INFORMÁTICA DAINF-PB

TERMO DE APROVAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO

Sistema Web para Controle de Jornada de Trabalho de Motoristas Profissionais

POR

Vinicius Dalsasso

Este trabalho de conclusão de curso foi apresentado no dia **25 de novembro de 2020**, como requisito parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, pela Universidade Tecnológica Federal do Paraná. O acadêmico foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Banca examinadora:

Profa. Dra. Rúbia Eliza de Oliveira Schultz Ascari	**Profa. Msc Andreia Scariot Beulke**	**Prof. Dr. Robison Cris Brito**
Professora orientadora	Professora convidada	Professor convidado

Assinam também:

Prof. Dr. Edilson Pontarolo	**Profa. Dra. Mariza Miola Dosciatti**
Coordenador do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas	Responsável pela Atividade de Trabalho de Conclusão de Curso

Documento assinado eletronicamente por (Document electronically signed by) **ANDREIA SCARIOT BEULKE, PROFESSOR ENS BASICO TECN TECNOLOGICO**, em (at) 03/12/2020, às 20:00, conforme horário oficial de Brasília (according to official Brasilia-Brazil time), com fundamento no (with legal based on) art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por (Document electronically signed by) **RUBIA ELIZA DE OLIVEIRA SCHULTZ ASCARI, PROFESSOR(A) ORIENTADOR(A)**, em (at) 03/12/2020, às 20:31, conforme horário oficial de Brasília (according to official Brasilia-Brazil time), com fundamento no (with legal based on) art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por (Document electronically signed by) **MARIZA MIOLA DOSCIATTI, PROFESSOR ENS BASICO TECN TECNOLOGICO**, em (at) 04/12/2020, às 09:34, conforme horário oficial de Brasília (according to official Brasilia-Brazil time), com fundamento no (with legal based on) art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por (Document electronically signed by) **ROBISON CRIS BRITO, PROFESSOR ENS BASICO TECN TECNOLOGICO**, em (at) 06/12/2020, às 09:09, conforme horário oficial de Brasília (according to official Brasilia-Brazil time), com fundamento no (with legal based on) art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por (Document electronically signed by) **EDILSON PONTAROLO, COORDENADOR(A) DE CURSO/PROGRAMA**, em (at) 07/12/2020, às 10:28, conforme horário oficial de Brasília (according to official Brasilia-Brazil time), com fundamento no (with legal based on) art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site (The authenticity of this document can be checked on the website) https://sei.utfpr.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador (informing the verification code) **1771371** e o código CRC (and the CRC code) **CC692C85**.

RESUMO

Este trabalho apresenta o processo de desenvolvimento de um sistema web de controle de jornada de motoristas. O trabalho visa atender a Lei 13.103/2015, que regulamenta a jornada de trabalho de motoristas profissionais. Com o sistema desenvolvido, será possível realizar o registro de jornadas que são associadas a regras de jornada, configuráveis via sistema, que assegura que motoristas não exerçam uma jornada sobrecarregada, e ainda, permitir adequações que se adaptem à possíveis atualizações da lei, com o objetivo de evitar que este controle seja feito de forma manual. Para o desenvolvimento deste trabalho, foram utilizadas tecnologias tais como a linguagem de programação PHP, para o *backend* do sistema, JavaScript, para programar os comportamentos das páginas, e MySQL, para o armazenamento de dados. Neste sistema, é realizado o cadastro de motoristas, jornadas e regras de jornadas, dependendo do nível do escopo do usuário, onde cada motorista registrado no sistema é associado a uma regra de jornada, e em cada jornada realizada são armazenadas informações sobre tempo gastos em diferentes atividades que um motorista pode realizar.

Palavras-chave: Motoristas. Jornada de trabalho motoristas. Sistema web.

ABSTRACT

This work presents the development process of a web driver's journey control system. The work aims to comply with Law 13.103 / 2015, which regulates professional drivers' working hours. With the developed system, it will be possible to register the journeys that are associated to a driver’s journey rule, configurable through the system, which ensures that drivers do not exercise an overloaded journey and also allow adjustments that adapt to possible updates of the law, to prevent this control from being done manually. In order to develop this work, technologies such as the PHP programming language were used, for the backend of the system, JavaScript, to program the pages’ behaviour, and MySQL, for data storage. In this system, drivers are registered, as well as journeys and journey rules, depending on the access level of the user, where every driver registered in the system is associated to a journey rule, and in every journey information about different activities that a driver may perform during the journey are stored.

Keywords: Drivers. Drivers' workday. Web system.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTAGEM DE CÓDIGO

LISTA DE ABREVIATURAS E SIGLAS

CSS	*Cascading Style Sheets*
HTML	*Hypertext Markup Language*
MVC	*Model-View-Controller*
PHP	*PHP Hypertext Preprocessor*
RIA	*Rich Internet Applications*

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta as considerações iniciais, os objetivos e a justificativa de realização do trabalho. Por fim está a apresentação dos capítulos subsequentes.

1.1 CONSIDERAÇÕES INICIAIS

No Brasil, o transporte de cargas e passageiros é baseado fundamentalmente no modal rodoviário (um tipo de transporte de carga para longas distâncias, feito por meios terrestres). No ano de 2014, o setor de transporte de cargas foi responsável por 61% dos deslocamentos e por, aproximadamente, 96% do transporte de passageiros (CONFEDERAÇÃO..., 2020).

Devido a essa demanda de transporte utilizando-se de rodovias, muitos motoristas acabam realizando jornadas prolongadas, de modo a aumentar a sua renda. Entretanto, isso acaba se tornando um risco. Dados apontam que motoristas que trabalham por 12 horas por dia têm suas chances de envolvimento em acidentes dobradas, passando a triplicar quando se passa de 14 horas (CRAIDE, 2013).

Diante desse problema, foram criadas diferentes leis para regulamentar o tempo de jornada de motoristas profissionais, com o objetivo de reduzir o número de acidentes provocados pela exaustão de jornadas prolongadas.

A Lei nº 12.619/2012, também conhecida popularmente como “Lei do Descanso”, estabelece limites de jornada e intervalos para repouso. Poucos anos depois, a partir da Lei 13.103/2015, o controle de jornada se tornou obrigatório para transportadoras, realizada por meio da anotação em diário de bordo, fichas, sistemas, entre outros (YAMAMOTO, 2015).

Considerando a regulamentação legal, é necessário que se estabeleça um controle para limites de jornada, com registro, definição de limites, descansos, entre outras, para que as transportadoras cumpram com as definições. Esse processo é passível de automatização, como o registro dos horários de início e término de jornada e realização da verificação do cumprimento da jornada conforme as orientações das respectivas leis, entre outros.

A partir de uma aplicação web, que pode ser acessada por dispositivos móveis ou pela própria Internet, esse processo seria facilitado, uma vez que a própria aplicação emitiria avisos de jornadas que fossem extensas demais, ou que descumpririam as orientações das leis em geral, sem a necessidade de verificação humana ao cadastro de cada nova jornada de um motorista.

Considerando esse contexto, por meio da realização deste trabalho foi desenvolvido um sistema que realize o registro de jornadas de motoristas profissionais. É um sistema

desenvolvido para web, com o intuito de facilitar o registro de informações de como é utilizado o tempo de um dia de jornada de um motorista.

1.2 OBJETIVOS

A seguir são apresentados os objetivos pretendidos com a realização deste trabalho.

1.2.1 Objetivo Geral

Desenvolver um sistema web, que realize o cadastro de jornada de trabalho de motoristas profissionais.

1.2.2 Objetivos Específicos

- Possibilitar o registro de jornada de trabalho de motoristas profissionais.
- Registrar usuários administradores e motoristas no sistema, com escopos de acesso diferentes.
- Permitir a atualização de regras de registro de jornadas.

1.3 JUSTIFICATIVA

Apesar de já existirem sistemas nesta área, as opções são poucas, uma vez que esta atividade era, e ainda é desenvolvida muitas vezes de forma manual. Segundo Toledo e Neves (2012), a ausência de um controle formal de jornada ou a sua instauração de modo incompleto ou inidôneo acarretará, no âmbito administrativo, a imposição de multas pelos órgãos de inspeção do trabalho.

O processo de registro de controle de jornada de trabalho é algo repetitivo e necessita de verificações sempre que uma nova rotina é cadastrada, para que a jornada cumpra com as orientações.

Nesse sentido, este trabalho visa tornar o controle de jornada mais facilitado, permitindo a vinculação de motoristas a regras de jornada, e o registro do tempo gasto do motorista em diferentes atividades (horas de trabalho regular, horas extras, tempo gasto em refeição, tempo de descanso, tempo de espera).

1.4 ESTRUTURA DO TRABALHO

Este trabalho está organizado em capítulos. Este é o primeiro e apresenta as considerações iniciais com o contexto do sistema que foi desenvolvido, os objetivos e a justificativa. O Capítulo 2 apresenta o referencial teórico, com ênfase no tema de controle de jornadas de trabalho de motoristas profissionais e Aplicações de Internet Ricas. No Capítulo 3 estão as ferramentas e tecnologias utilizadas na modelagem do sistema e na implementação subsequente do sistema. No Capítulo 4 é apresentado o resultado da realização do trabalho, ou seja, a modelagem criada e a o sistema implementado. Por fim estão as considerações finais seguidas pelas referências.

2 CONTROLE DE JORNADAS DE TRABALHO DE MOTORISTAS PROFISSIONAIS E APLICAÇÕES DE INTERNET RICAS

Este capítulo apresenta alguns conceitos considerados importantes para o desenvolvimento deste trabalho, referentes à jornada de trabalho de motoristas profissionais, e Aplicações de Internet Ricas.

2.1 JORNADA DE TRABALHO DE MOTORISTAS PROFISSIONAIS

A jornada de trabalho consiste no período em que o trabalhador está à disposição do empregador, seja executando ordens ou a espera delas (MACHADO e GOLDSCHMIDT, 2012). O tempo efetivamente trabalhado, bem como o tempo à disposição e o tempo de deslocamento compõem a jornada de trabalho (DELGADO, 2012).

Este trabalho concentra-se na modalidade de jornada de trabalho controlada, voltada a motoristas profissionais empregados que desenvolvem suas atividades no meio urbano ou mesmo rural e, ainda, nas vias urbanas ou interurbanas. A Lei 12.619 de 2012, Lei do Motorista, regulamenta o controle sobre a jornada de trabalho dos motoristas de carga e passageiros e acrescenta disposições ao Código de Trânsito Brasileiro (Lei 9.503/97). Essa lei tornou o controle da jornada de trabalho para motoristas obrigatório, e foi complementada posteriormente pela Lei 13103/2015, que acrescentou alguns aspectos ao controle de jornada, como a possibilidade de utilizar meios eletrônicos instalados nos veículos a critério do empregador para controle de jornadas (COELHO, 2015).

Para Machado e Goldschmidt (2012) a limitação e o controle da jornada de trabalho constituem medida profilática para a manutenção da saúde do trabalhador. Assim, considerando situações em que o controle de jornada não pode ser feito de forma automática, pode-se transferir ao motorista a responsabilidade por registrar sua jornada de trabalho, o tempo de intervalo intrajornada e interjornada, e o descanso semanal.

Uma vez que a interpretação de leis, ou mesmo o surgimento de novas leis podem implicar em desafios de interpretação ou entendimentos diversos (COELHO, 2015), transferir para o empregador o registro dos critérios que devem ser seguidos pelo motorista pode ser uma opção válida para manter o sistema flexível e adaptável. Assim, o sistema proposto neste trabalho prevê o cadastro pelo empregador (como usuário administrador) das regras de jornada, sendo que os registros feitos pelo motorista devem seguir essas regras.

Nesse contexto, o desenvolvimento de uma aplicação web voltada ao cadastro de jornada de trabalho de motoristas foi visto como oportuno para permitir o registro facilitado das jornadas de trabalho realizadas pelos motoristas, assim como o acompanhamento desses registros pelo empregador.

Aplicações web podem apoiar a colaboração e o compartilhamento do conteúdo gerado pelos usuários. O desenvolvimento dessas aplicações empregando interfaces avançadas, como as baseadas em *Aplicações de Internet Ricas*, permitem entregar ao usuário uma solução com aparência, comportamento e usabilidade próximas de aplicações desktop.

2.2 APLICAÇÕES DE INTERNET RICAS

Tem ocorrido uma crescente migração de uma multiplicidade de aplicações tradicionalmente *desktop* para web. Essas aplicações migradas e desenvolvidas com um paradigma, em termos de recursos de interação do usuário, que as aproximam das aplicações *desktop* são denominadas Aplicações de Internet Ricas (*Rich Internet Applications* – RIAs*)* (DUHL, 2003). As RIAs são imaginadas ser a unificação do melhor das aplicações *desktop* tradicionais, tais como interface com o usuário interativa e tempo de resposta mais rápido pela não necessidade de redesenhar a página toda a cada interação do usuário, com o melhor das aplicações web, destacando-se a possibilidade de *download* progressivo para recuperação de conteúdo (POWELL; NAKAMURA; AKAMA, 2009) e a comunicação assíncrona entre cliente e servidor (MARTÍNEZ-RUIZ, 2010).

Em aplicações RIAs, a apresentação de conteúdo, como textos, imagens e mídias, requer tecnologias que possibilitem desenvolver interfaces muito mais ricas para o usuário, oferecendo recursos que estimulem e facilitem a interação do mesmo (CIRILO, 2010). As RIAs são caracterizadas pela interação ampla e melhorada com o usuário final que não possui um papel passivo somente como um consumidor de conteúdo armazenado e processado no lado servidor da aplicação, mas que tem assumido um papel ativo na escolha/seleção do processamento do conteúdo a ser provido no lado cliente (BERNARDI; DI LUCCA; DISTANTE, 2009).

As RIAs provêm uma experiência interativa e responsiva para o usuário (LAWTON, 2008) por meio de uma interface gráfica rica e respostas rápidas (CAMERON, 2004). Esses recursos não estão disponíveis em aplicações web clássicas ou tradicionais que são compostas por páginas sequenciais, com interatividade limitada e respostas lentas (LATOW, 2008). As RIAs assemelham-se às aplicações *desktop*, ainda que forneçam funcionalidades baseadas na web (DISSANAYAKE; DIAS, 2014). Com a introdução de HTML5 **(***Hypertext Markup*

*Language***)** e CSS3 (*Cascading Style Sheets*), as RIAs têm se tornado mais proeminentes com o aumento de portabilidade e disponibilidade (CORREIA, 2013).

Contudo, várias metodologias usadas para desenvolver as aplicações web clássicas não são automaticamente transferidas para o desenvolvimento de RIAs devido às características dessas aplicações, que são (POWELL; NAKAMURA; AKAMA, 2009):

- Descarregamento da apresentação e camadas interativas do servidor para o cliente;
- Minimização dos dados transferidos entre cliente e servidor em decorrência de atualização parcial da página;
- Distribuição da carga de computação entre servidor e cliente;
- Comunicação assíncrona e concorrente entre servidor e cliente.

As RIAs apresentam melhor usabilidade, permitindo melhor desempenho da aplicação porque elas reduzem significativamente o número de requisições ao servidor pelo processamento no cliente de muitas operações e a manipulação de dados requisitados pelo usuário, reduzindo, assim, a atualização da página e o tráfego de dados na Internet. As RIAs são baseadas em um estilo de arquitetura cliente-servidor usando requisição assíncrona composta por pequenos blocos de dados (BERNARDI; DI LUCCA; DISTANTE, 2009). As RIAS são mais similares às aplicações desktop porque elas combinam a arquitetura de distribuição da web com a interatividade de interface e a capacidade de computação das aplicações *desktop* e essa combinação melhora todos os elementos de uma aplicação web: dados, lógica de negócio, comunicação e apresentação (FRATERNALLI; ROSSI, 2010).

Sumariamente, uma RIA pode ser vista como uma aplicação web na qual a interface com o usuário é processada no lado cliente e a lógica de negócio é definida por um serviço *backend* (VALVERDE, 2009).

3 MATERIAIS E MÉTODO

A seguir estão os materiais e o método utilizados para a modelagem e a implementação do sistema obtido como resultado deste trabalho.

3.1 MATERIAIS

O Quadro 1 apresenta as tecnologias e as ferramentas utilizadas no desenvolvimento do trabalho.

Quadro 1 - Tecnologias e ferramentas utilizadas

Ferramenta / Tecnologia	Versão	Finalidade
Visual Studio Code	1.4.7	IDE para programação. Utilizado para edição de arquivos HTML, JavaScript, PHP e CSS.
MySQL Workbench	8.0.20	Modelagem do banco de dados.
MySQL	8.0.20	Banco de dados.
PHP	7.4.9	Linguagem de programação.
phpMyAdmin	5.0.1	Administrador de banco de dados MySQL.
JavaScript	ECMAScript 5	Linguagem de programação.
StarUML	2.7.0	Modelagem dos casos de uso, diagrama de classes.
CSS	3	Linguagem de estilização para HTML ou XML.
HTML	5	Linguagem de marcação utilizada para construção de página web.
Bootstrap	3.3.7	Framework HTML e CSS, utilizado para desenvolvimento de páginas web responsivas.

Fonte: Autoria própria.

Entre as tecnologias utilizadas, o Visual Studio Code foi escolhido como IDE de programação, por ser uma IDE simples e prática para o desenvolvimento web. O banco de dados escolhido foi o MySQL, por ser também um dos bancos de dados mais populares para web.

Quanto as linguagens de programação, JavaScript, HTML e CSS foram utilizados para o desenvolvimento do *front-end* do sistema, e o *PHP Hypertext Preprocessor* **(**acrônimo recursivo, PHP), como linguagem para o *back-end*, por ser uma linguagem estabelecida para o desenvolvimento de aplicações web.

Por fim, o MySQL Workbench foi utilizado para modelagem do banco de dados, e o StarUML para modelagem do banco de casos de uso.

3.1.1 PHP

A linguagem de programação PHP foi utilizada por ser uma das linguagens de desenvolvimento web mais utilizados no mercado, por ser uma linguagem de alto desempenho, compatibilidade com diversos banco de dados (desde que o *driver* de banco de dados a ser escolhido permita) e por ser multiplataforma.

O PHP é utilizado do lado de servidor do sistema. Ele é utilizado para fazer as inserções no banco, definições de classes e para realizar a estruturação das páginas de *views*, onde trechos de código de *layout* são guardados em uma variável de um arquivo de um *layout* específico, e são chamados em outros arquivos toda vez que for necessária a reutilização do código.

Essa foi a principal linguagem de programação utilizada durante o desenvolvimento deste trabalho, sendo que no lado cliente do sistema o PHP não é utilizado, que no caso desse sistema, utiliza a linguagem JavaScript.

O que distingue o PHP de algo como o JavaScript no lado do cliente é que o código é executado no servidor, gerando o HTML que é então enviado para o navegador. O navegador recebe os resultados da execução desse script, mas não sabe qual era o código fonte (PHP.NET, 2020).

3.2 MÉTODO

O método empregado neste trabalho consiste nas atividades de levantamento de requisitos, análise, projeto e desenvolvimento. A seguir estão descritas, de forma sucinta, as etapas desenvolvidas para a realização deste trabalho.

3.2.1 Levantamento de Requisitos

O levantamento de requisitos teve como objetivo o desenvolvimento de um sistema web no qual fosse possível cadastrar jornadas de motoristas profissionais e realizar as devidas verificações. Na fase de levantamento, foram pesquisadas as leis que tratam do assunto, em especial a Lei n° 13.103/2015, além de artigos e sites institucionais nos quais se abordam assuntos relacionados ao trabalho.

Com o desenvolvimento do trabalho, mais requisitos foram adicionados, bem como diversas alterações de modo a definir de maneira mais adequada as funcionalidades do sistema.

3.2.2 Análise e Projeto

Tendo como base os requisitos estabelecidos, foram criados os casos de uso do sistema, os quais foram necessários para gerar informações para a etapa da definição de banco de dados e funcionalidades do sistema.

3.2.3 Desenvolvimento

Para o desenvolvimento foi utilizada a ferramenta Visual Studio Code e banco de dados MySQL. Foi montado elaborado o modelo lógico e banco de dados inicial para o sistema no MySQL Workbench, e então, iniciado o desenvolvimento do código fonte do sistema. O desenvolvimento de cadastros seguiu uma estrutura base, padronizando as operações de inclusão, edição e exclusão.

4 RESULTADOS

Este capítulo apresenta o que foi obtido como resultado deste trabalho, que é a modelagem e implementação de um sistema web para controle da jornada de trabalho de motoristas profissionais.

4.1 ESCOPO DO SISTEMA

O sistema modelado como resultado deste trabalho permite registrar as jornadas de motoristas profissionais. O registro de jornadas é vinculado a uma série de normas definidas em lei, as quais serão definidas como as regras de jornada do sistema. A solução proposta considera o contexto apresentado a seguir.

O motorista poderá registrar a sua própria jornada por meio de uma página web. Devem ser informados diversos dados de como foi gasto o tempo durante a jornada, seja o tempo gasto em direção, espera, descanso ou refeição.

O usuário com escopo operacional do sistema, cadastrará os motoristas que terão acesso ao sistema, além de poder realizar o cadastro de jornadas pelos motoristas, assim como alterá-las.

O administrador do sistema cadastrará usuários operacionais do sistema, além de ser ele o responsável pela criação e atualização de regras de jornada (caso a legislação mude, ele poderá alterar o número de horas necessárias como intervalos entre jornadas, por exemplo). Além disso, ele terá todos os privilégios que os usuários com escopo operacional possuem.

4.2 MODELAGEM DO SISTEMA

O Quadro 2 apresenta a listagem dos requisitos funcionais identificados para o sistema. O requisito RF05 deve ser implementado futuramente em um aplicativo *mobile* que será integrado ao sistema web desenvolvido.

Quadro 2 - Requisitos Funcionais

Identificação	Nome	Descrição
RF01	Cadastrar motorista	Os motoristas são os profissionais que realizam as jornadas de trabalho. Cada jornada se refere a um único motorista.
RF02	Cadastrar regras de jornada de trabalho	As regras serão os limites de jornada que seguirão recomendações da lei. Como as regras da lei podem ser modificadas, deve ser possível customizar as regras de jornada de trabalho
RF03	Cadastrar Veículos	Veículos que compõe a frota da empresa. Cada veículo possui um identificador único, que no caso seria a sua placa.

RF04	Cadastrar Jornada através da plataforma web	Jornada cadastrada para fins de registro pelo pessoal da empresa, através do sistema web.
RF05	Cadastrar usuários	Usuários da parte web do sistema – isto é, os usuários operacionais e administrativo, possuindo diferentes permissões.

Fonte: Autoria própria.

O Quadro 3 apresenta os requisitos não-funcionais identificados para o sistema, também denominados de requisitos suplementares. Os requisitos não funcionais explicitam regras de negócio, restrições ao sistema de acesso, requisitos de qualidade, desempenho e segurança, dentre outros.

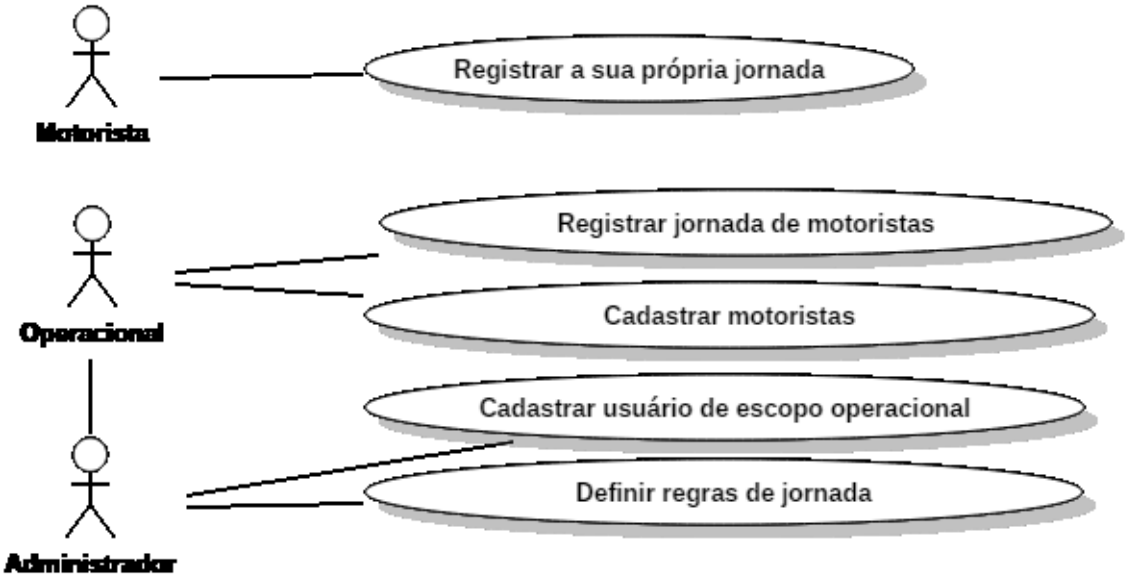
Quadro 3 - Requisitos Não Funcionais

Identificação	**Nome**	**Descrição**
RNF01	Acesso ao sistema	O acesso ao sistema será realizado por meio de *login* e senha.
RNF02	Vinculação de um motorista com uma jornada	Os motoristas devem ter ligação com uma regra de jornada específica.

Fonte: Autoria própria.

O diagrama de casos de uso apresentado na Figura 2 contém as funcionalidades essenciais do sistema realizadas pelos seus atores que são: motoristas, operacional e administrador. O administrador é responsável pelos cadastros de usuários operacional do sistema, além de definir as regras de jornada. O operacional é quem realiza a atividade de cadastro e alteração de jornadas e cadastro de motoristas. Os motoristas são os profissionais que utilizarão o sistema web para cadastrar a sua própria jornada.

Figura 1 - Diagrama de casos de uso



Fonte: Autoria própria.

O Quadro 4 apresenta a operação incluir dos casos de uso de cadastros para as funcionalidades que se referem aos cadastros de jornadas realizados no sistema.

Quadro 4 - Operação incluir no caso de uso de cadastro

Caso de uso: Cadastrar uma jornada (por meio do *back-end* do sistema web). **Descrição:** Usuário operacional ou administrador irá efetuar o cadastro de uma jornada, incluindo dados necessários como hora de início, término, entre outros. **Evento iniciador:** Acessar o menu de cadastro de jornadas. **Atores:**	
Operacional **Pré-condição** Possuir acesso ao sistema (um usuário operacional ou administrativo). **Sequência de Eventos:** 1. O usuário acessará a tela de jornadas, através de um menu “Jornada”. 2. Em seguida, ele selecionará o motorista que deseja incluir uma jornada. 3. É preenchido os dados necessários para o cadastro da jornada. 4. Em seguida pressiona o botão salvar. **Pós condição:** Dados da jornada são inseridos no banco de dados	
Nome do fluxo alternativo (extensão)	Descrição
Linha 4: Campos obrigatórios não foram preenchidos.	4.1 Ao salvar os dados, é realizada uma verificação para descobrir se um ou mais campos obrigatórios não foram preenchidos pelo usuário. É emitida uma mensagem informativa e permanece no formulário de cadastro para edição.

Fonte: Autoria própria.

O Quadro 5 apresenta a operação alterar dos casos de uso de cadastros para as funcionalidades que se referem aos cadastros de jornadas realizados no sistema.

Quadro 5 - Operação alterar do caso de uso de cadastro

<table>
<tr><td colspan="2">Caso de uso:
Alterar uma jornada (através do back-end do sistema web).
Descrição:
Usuário operacional ou administrador irá efetuar a alteração de uma jornada.
Evento iniciador:
Acessar o menu de jornadas.
Atores:</td></tr>
<tr><td colspan="2">Usuário:
Pré-condição
Possuir acesso ao sistema (um usuário operacional ou administrativo).
Sequência de Eventos:
1. O usuário acessará a tela de jornadas, através de um menu “Jornada”.
2. Em seguida, ele selecionará o motorista que deseja alterar uma jornada.
3. É preenchido os dados para a alteração da jornada.
4. Em seguida pressiona o botão salvar.
Pós condição:
Dados da jornada são atualizados no banco de dados</td></tr>
<tr><td>Nome do fluxo alternativo (extensão)</td><td>Descrição</td></tr>
<tr><td>Linha 4: Campos obrigatórios não foram preenchidos.</td><td>4.1 Ao salvar os dados, é realizada uma verificação e descobre-se que um ou mais campos obrigatórios não foram preenchidos pelo usuário. É emitida uma mensagem informativa e permanece no formulário de cadastro para edição.</td></tr>
</table>

Fonte: Autoria própria.

O Quadro 6 apresenta a operação de exclusão dos casos de uso de cadastros para as funcionalidades que se referem aos cadastros de jornadas realizados no sistema.

Quadro 6 - Operação excluir do caso de uso de cadastro

<table>
<tr><td colspan="2">Caso de uso:
Excluir uma jornada (através do back-end do sistema web).
Descrição:
Usuário operacional ou administrador irá efetuar a exclusão de uma jornada.
Evento iniciador:
Acessar o menu de jornadas.
Atores:</td></tr>
<tr><td colspan="2">Usuário:
Pré-condição
Possuir acesso ao sistema (um usuário operacional ou administrativo).
Sequência de Eventos:
1. O usuário acessará a tela de jornadas, através de um menu “Jornada”.
2. Em seguida, ele selecionará o motorista que deseja excluir uma jornada.
3. É selecionada a jornada que se deseja excluir.
4. Em seguida pressiona o botão excluir.
Pós condição:
Dados da jornada são apagados do banco de dados.</td></tr>
<tr><td>Nome do fluxo alternativo (extensão)</td><td>Descrição</td></tr>
<tr><td>Linha 3: Confirmação de exclusão.</td><td>3.1 O sistema emitirá uma mensagem de alerta confirmando se o usuário realmente deseja excluir aquele registro.</td></tr>
</table>

Fonte: Autoria própria.

4.3 APRESENTAÇÃO DO SISTEMA

A Figura 2 apresenta a tela inicial (de acesso) ao sistema, onde deverá ser informado o nome de usuário e senha de acesso.

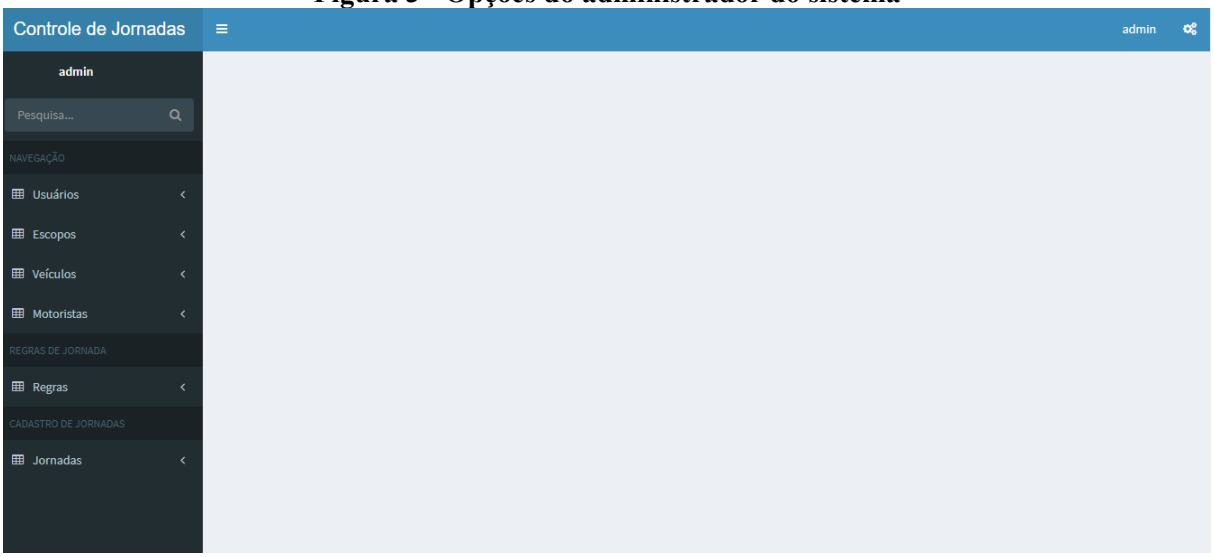
Figura 2 - Tela de acesso ao sistema

Fonte: Autoria própria.

Após informar os dados solicitados, o usuário realizará o acesso ao sistema, sendo que as opções exibidas dependerão do escopo do usuário que acessou. Usuários operacionais terão acesso ao controle de jornadas de motoristas, enquanto usuários administrativos terão acesso à configuração de regras de jornada além do controle, e motoristas apenas poderão realizar o registro de suas jornadas. Em caso de esquecimento de senha, será necessário contatar o administrador do sistema.

A Figura 3 apresenta a tela com as opções relativas ao usuário administrador do sistema.

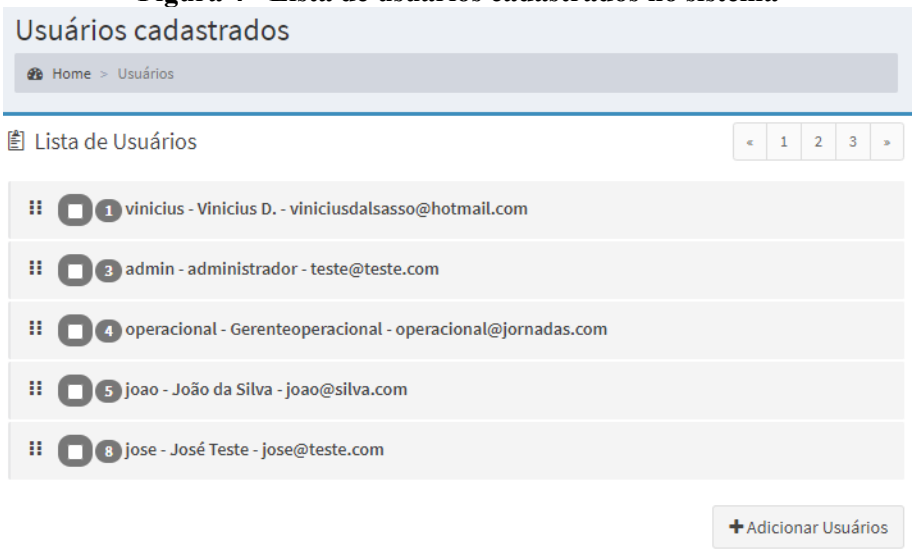
Figura 3 - Opções do administrador do sistema



Fonte: Autoria própria.

Acessando o sistema como usuário administrador, é possível acrescentar mais usuários, administradores ou operacionais, por meio do menu “Usuários”, no menu lateral esquerdo do sistema. Por meio dele, será exibida uma lista de usuários cadastrados no sistema, conforme apresentado na Figura 4.

Figura 4 - Lista de usuários cadastrados no sistema

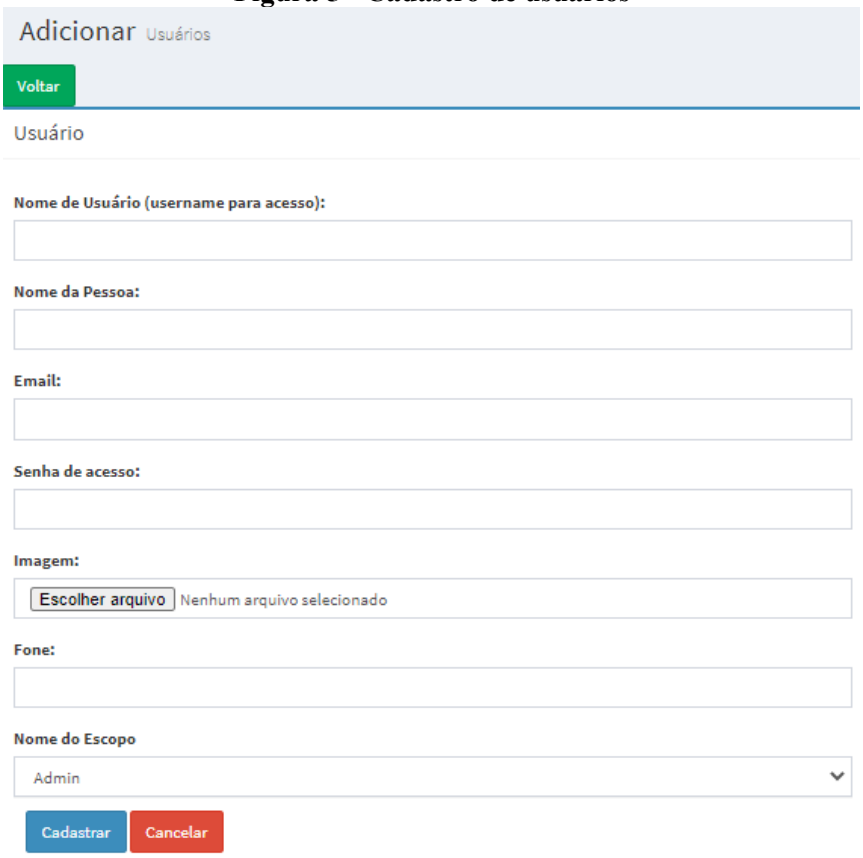


Fonte: Autoria própria.

Os usuários podem ser administrativos (com acesso a todas as funcionalidades do sistema), ou operacionais, que não possuem acesso à seção de cadastro e alteração de jornadas de trabalho.

Para cadastrar novos usuários no sistema, o administrador deverá selecionar a opções “Adicionar Usuários”, a qual abrirá a tela de cadastro de usuários, apresentada na Figura 5.

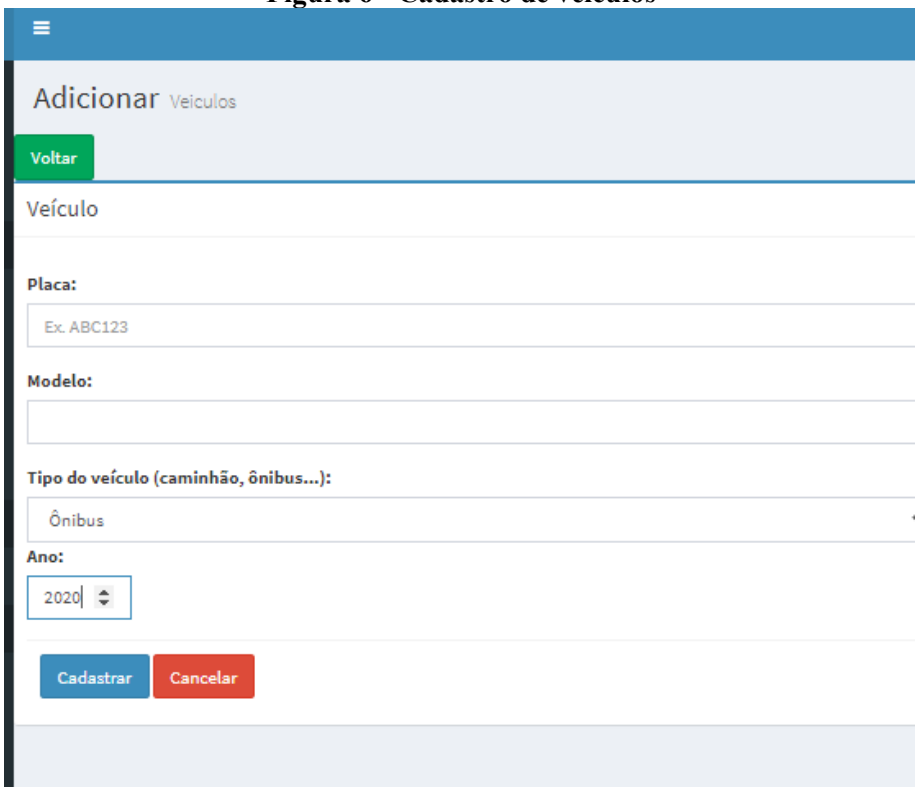
Figura 5 - Cadastro de usuários



Fonte: Autoria própria.

Na Figura 6, é apresentada a tela de cadastro de veículos, na qual são informados alguns dados para identificação de veículos que podem ser utilizados durante a realização de jornadas.

Figura 6 - Cadastro de veículos

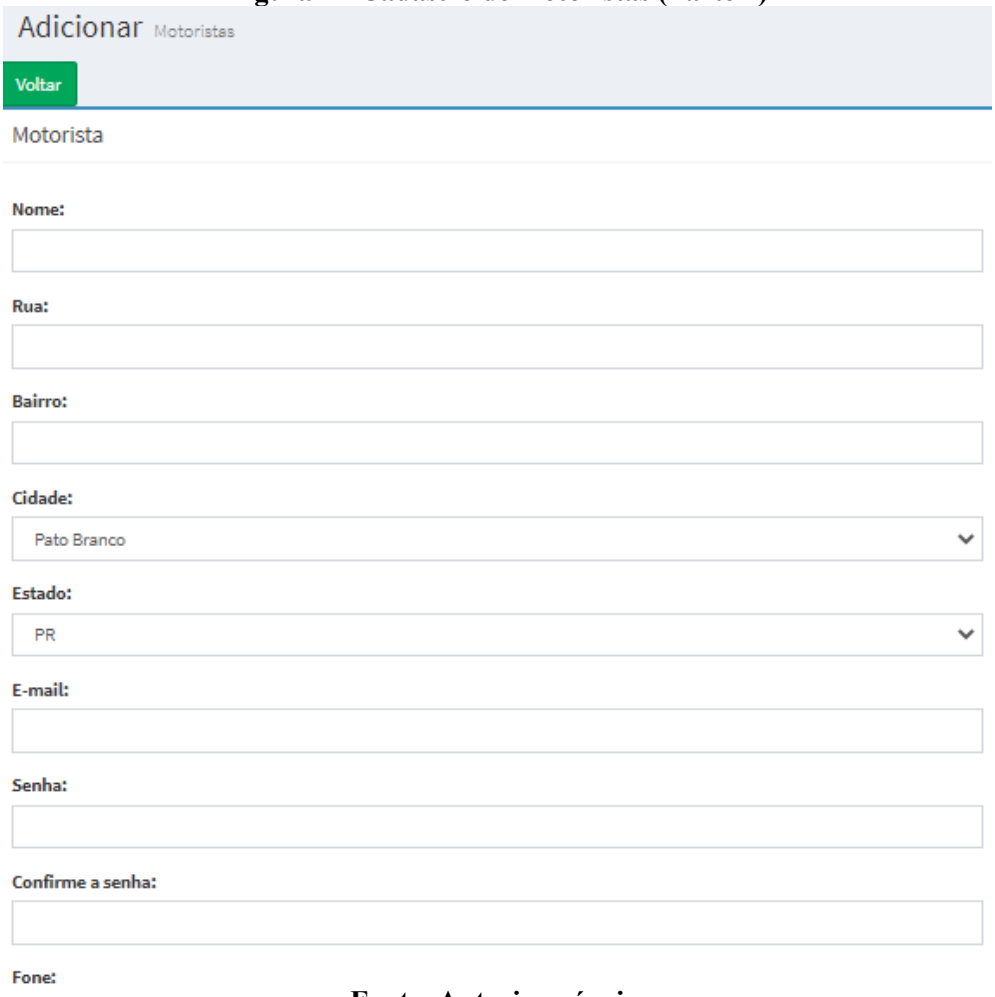


Fonte: Autoria própria.

O campo placa possui validação de formato de placa, feita por meio de um arquivo JavaScript incluído nas *tags* da página, sendo que deve seguir o formato "ABC1D23". O campo tipo do veículo apresenta um *combobox* onde é selecionado os tipos de veículo presentes no banco de dados do sistema, e por fim o campo ano, sendo um campo com um tipo de *input* numérico, com valor mínimo requerido de 1900 e máximo de 2100 em sua *tag* HTML.

A Figura 7 apresenta parte da tela de cadastro de motoristas, a qual possui diversos campos para identificação do motorista. Os campos de cidade e estado são campos de seleção, em que o usuário seleciona de uma lista de estados e cidades presentes no banco de dados, e o campo de senha também possui um campo de confirmação, para realização da validação de que a senha digitada é realmente a pretendida. A máscara de senha é exibida assim que for digitado no campo.

Figura 7 - Cadastro de motoristas (Parte 1)



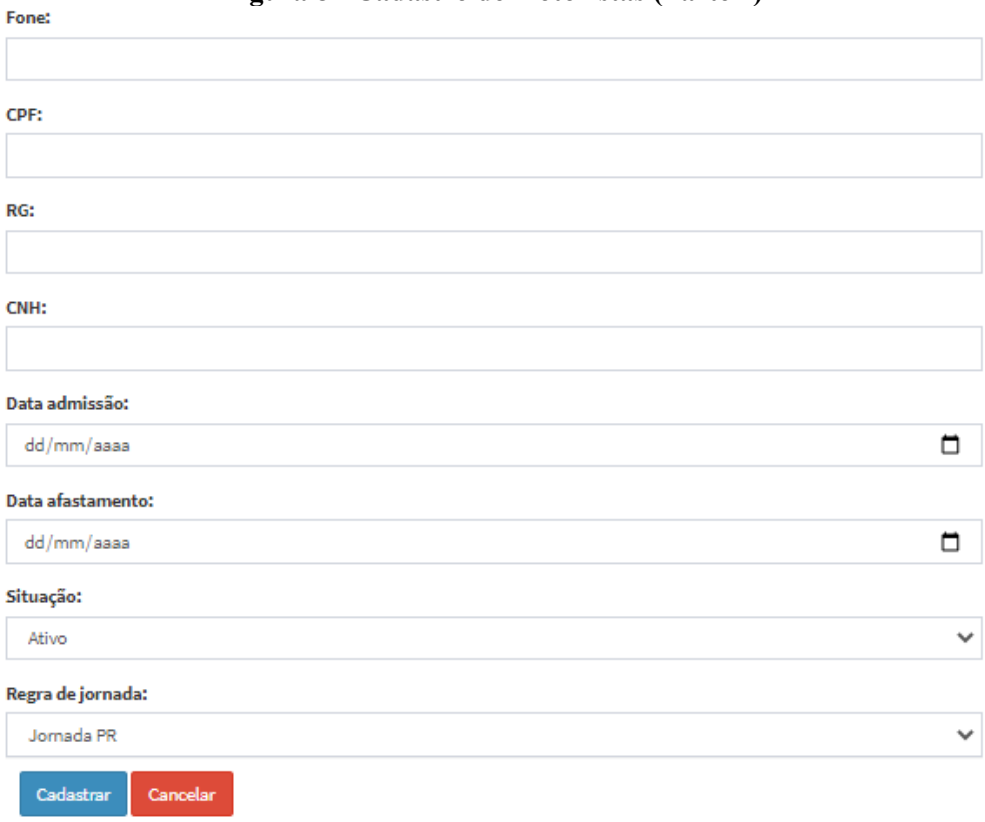
Fonte: Autoria própria.

Na Figura 8, é apresentada a continuação da tela do cadastro de motorista, exibindo os campos restantes do cadastro.

No campo “data de admissão”, de caráter obrigatório, é informada a data de admissão do motorista à empresa. O campo “data de afastamento”, de preenchimento opcional, serve para armazenar a data de afastamento do motorista, caso ele tenha sido afastado. Deve ser informada a data da última admissão e afastamento caso isso tenha ocorrido mais de uma vez. Campos de CPF, RG e CNH são validados em funções específicas através de um arquivo JavaScript que é chamado por uma *tag* no início do arquivo “addmotorista.php”. No campo de situação, deve ser selecionado no campo *combobox* entre as opções de “ativo” e “inativo”, sendo que caso o cadastro deste motorista fique inativo, é impedido o cadastro de jornadas relativas a este motorista.

No campo “regra de jornada”, o administrador deverá selecionar no *combobox* qual regra o motorista irá seguir, sendo que suas jornadas serão baseadas nos dados da regra de jornada selecionada em seu cadastro.

Figura 8 - Cadastro de motoristas (Parte 2)

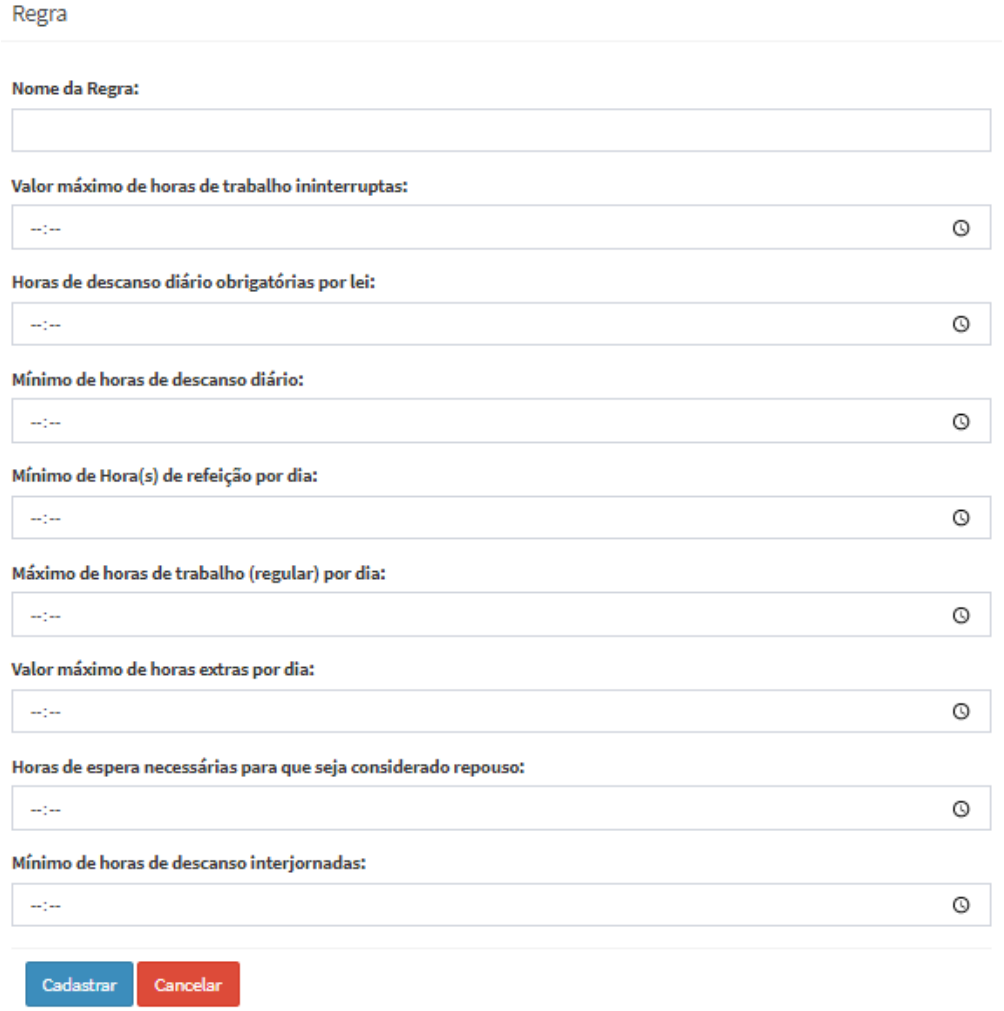


Fonte: Autoria própria.

Na Figura 9, é apresentada a tela de regras de jornada do sistema. Empresas de transporte geralmente irão seguir uma única regra, porém, a Lei nº 13.103/2015, que que rege as jornadas de trabalho de motoristas profissionais, define valores diferentes de descanso para motoristas de carga e de passageiros, portanto, a possibilidade de definição de diferentes regras permite que se façam controles para ambos os tipos de transporte no próprio sistema.

Os tempos para cada regra devem ser informados seguindo o formato “hh:mm” (horas e minutos).

Figura 9 - Cadastro de regras de jornada

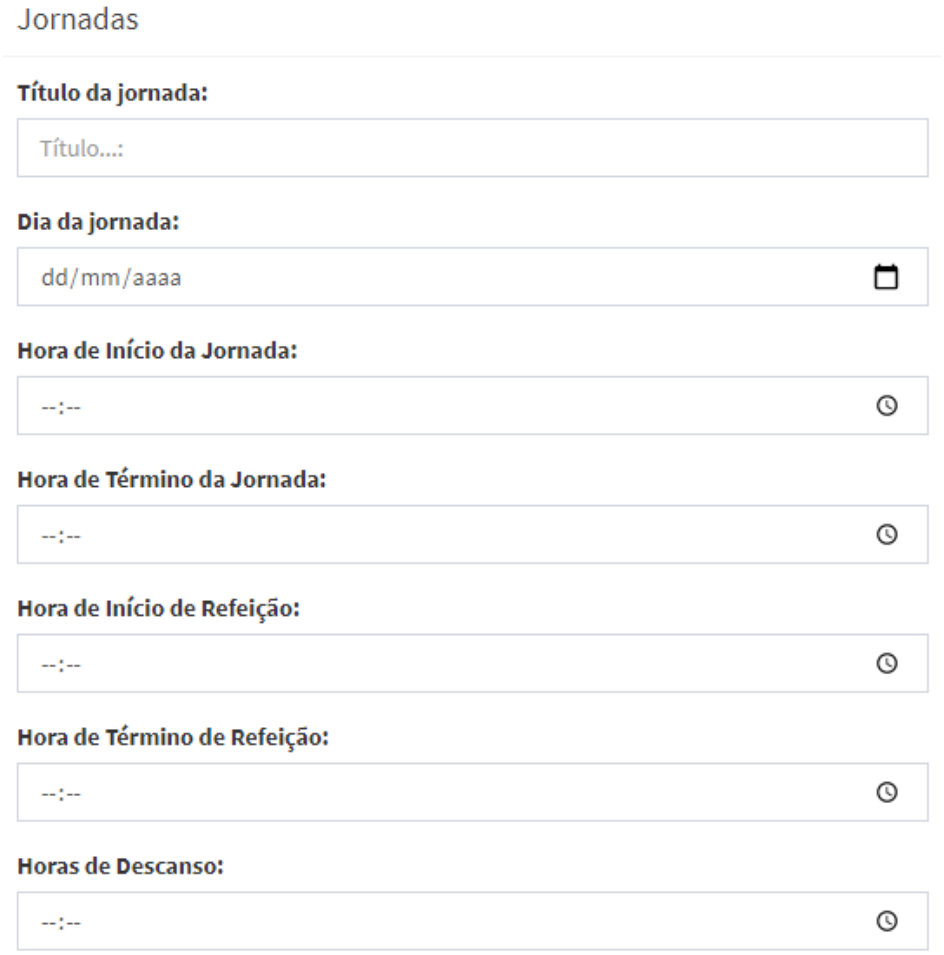


Fonte: Autoria própria.

Na Figura 10, é apresentado o cadastro de jornadas do sistema. Nele, o usuário fará o registro de um dia de jornada.

No cadastro, deve ser selecionado o dia da jornada, hora de início e fim, de qualquer período trabalhado no dia, desconsiderando possíveis pausas que são informadas nos outros campos. Também deve ser informado o tempo inicial e final de refeição, e tempo de descanso entre o tempo inicial e final da jornada (tempos fora do intervalo inicial e final de jornada são automaticamente considerados como descanso).

Figura 10 - Tela de cadastro de jornada (Parte 1)

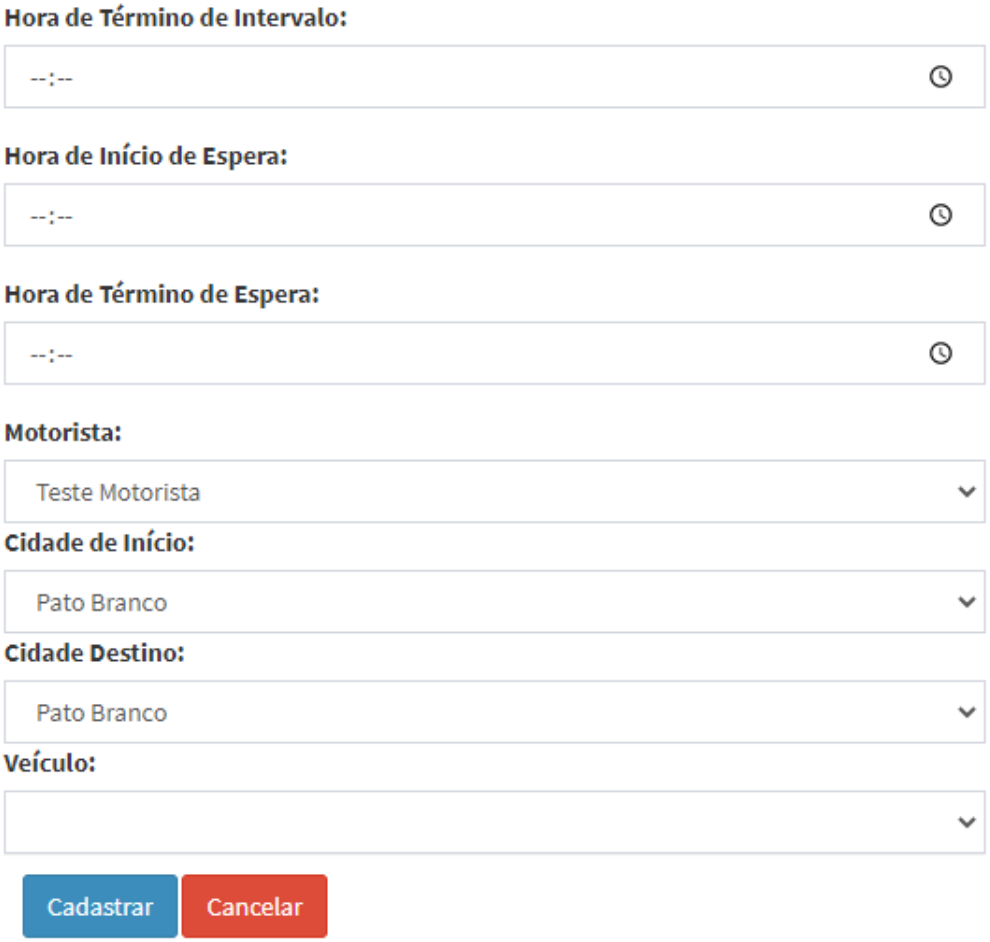


Fonte: Autoria própria.

A Figura 11 apresenta a continuação da tela do cadastro de jornadas. Deve ser informada a hora de término do intervalo realizado para descanso (caso houve durante o período de jornada), hora de início e término de espera. O tempo de espera é o tempo em que o motorista fica aguardando a carga ou descarga do veículo nas dependências do embarcador ou destinatário.

Também é selecionado o motorista que realizou a jornada, cidade de início e destino daquela jornada, e o veículo utilizado durante a realização da jornada.

Figura 11 - Tela de cadastro de jornada (Parte 2)



Fonte: Autoria própria.

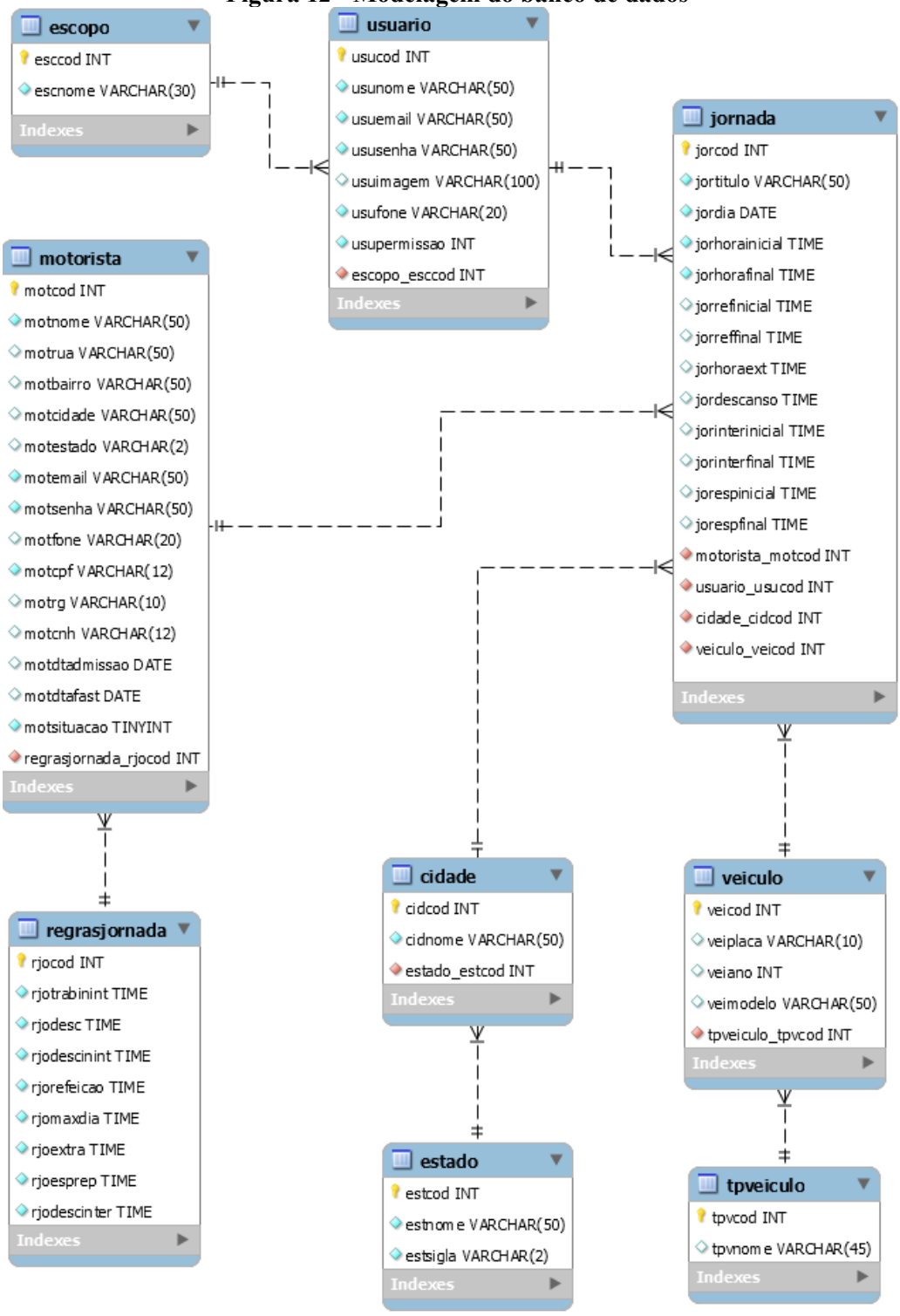
4.4 IMPLEMENTAÇÃO DO SISTEMA

O sistema foi desenvolvido utilizando como linguagem principal o PHP, responsável pela parte de servidor desenvolvida em web. Além disso, é utilizado JavaScript para *client-side*, HTML e CSS para estruturação e apresentação de conteúdo, além do banco de dados MySQL para armazenamento de dados.

De modo a auxiliar na criação de interfaces, foi o escolhido o *framework* Bootstrap, facilitando o processo de desenvolvimento de interfaces responsivas. Além disso, foi utilizado o AdminLTE, que é um tema baseado em Bootstrap que forneceu a base para a customização e construção do *dashboard* do sistema.

Por meio do MySQL Workbench, foi criado o modelo lógico do banco de dados, conforme a Figura 12.

Figura 12 - Modelagem do banco de dados



Fonte: Autoria própria.

Entre as tabelas principais do sistema, estão a tabela REGRASJORNADA, onde são configuradas as regras para registro de jornada, a tabela JORNADA, responsável pelo armazenamento de cada jornada percorrida, e a tabela MOTORISTA, que contém os dados do motorista e a regra de jornada à que ele pertence.

O Quadro 7 apresenta os atributos da tabela correspondente ao controle de regras de jornada (REGRASJORNADA).

Quadro 7 - Atributos da tabela REGRASJORNADA

REGRASJORNADA	
RJOCOD	Código da regra de jornada.
RJTRABININT	define o máximo de horas ininterruptas que o motorista pode ter em sua jornada.
RJODESC	define quantas horas de descanso diárias são necessárias.
RJODESCININT	define a quantidade de horas de descanso sem interrupção diárias são obrigatórias.
RJOREFEICAO	define o tempo reservado para refeição diário.
RJOMAXDIA	define a quantidade de horas máxima de jornada em trabalho normal.
RJOEXTRA	trata da quantidade de horas máximas de jornada diária extras, consideradas quando o máximo de horas em ritmo normal foi extrapolado.
RJOESPREP	trata da quantidade de horas em espera que são necessárias para que um tempo de espera passe a ser considerado como repouso.
RJODESCINTER	define o tempo mínimo de intervalo para que se continue a jornada após atingir o limite previsto na RJTRABININT.

Fonte: Autoria própria.

Na tabela do cadastro de jornada, são registrados todos os dados a respeito de como foi utilizado o tempo (entre o tempo inicial e final) daquela jornada. O Quadro 8 apresenta os atributos da tabela correspondente ao cadastro de jornada (JORNADA).

Quadro 8 - Atributos da tabela JORNADA

JORNADA	
JORCOD	Código da jornada.
JORTITULO	Título de identificação da jornada.
JORDIA	Dia em que a jornada foi realizada.
JORHORAINICIAL e JORHORAFINAL	Registra a hora inicial e final da jornada no dia.
JORREFINICIAL e JORREFFINAL	Horário de início e fim de refeição.
JORHORAEXT	Registra a quantidade de horas extras naquela jornada, se existir, caso o máximo de horas diária normal for extrapolado.
JORDESCANSO	Armazena a quantidade de horas de descanso naquele dia. A quantidade de horas de descanso será o total de horas do dia subtraído das horas de trabalho (normal e extras), de espera e de refeição.
JORINTERINICIAL e JORINTERFINAL	Registra o tempo gasto como intervalo durante a jornada, caso utilizado.
JORESPINICIAL e JORESPFINAL	Utilizado para registro d

Fonte: Autoria própria.

A tabela de motorista contém os dados de identificação do motorista, além da regra de jornada à qual ele é vinculado. O Quadro 9 apresenta os atributos da tabela correspondente ao cadastro de motoristas (MOTORISTA).

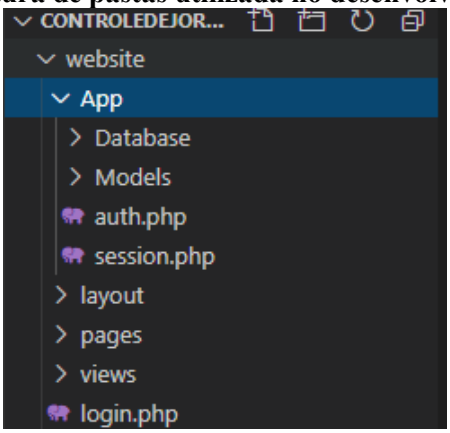
Quadro 9 - Atributos da tabela MOTORISTA

MOTORISTA	
MOTCOD	Código do motorista.
MOTNOME	Nome do motorista.
MOTRUA, MOTBAIRRO, MOTCIDADE, MOTESTADO	Dados de endereço.
MOTEMAIL	E-mail do motorista, utilizado como acesso.
MOTFONE, MOTCPF, MOTRG, MOTCNH	Dados de identificação do motorista.
MOTDTADMISSAO e MOTDTAFAST	Data de admissão do motorista e a data de afastamento dele da empresa (caso tenha sido afastado).
MOTSITUAÇÃO	Define a situação do motorista, ativo ou inativo, para acesso ao sistema.
REGRASJORNADA_RJOCOD	Armazena qual configuração de regra de jornada que aquele motorista será regido.

Fonte: Autoria própria.

Depois de criada a estrutura básica do banco de dados, foi iniciado o desenvolvimento do sistema. A Figura 13 apresenta a estrutura de pastas adotada no projeto do sistema.

Figura 13 - Estrutura de pastas utilizada no desenvolvimento do sistema



Fonte: Autoria própria.

Foram utilizados *models* para cada classe, contendo funções como *lists* e demais funções específicas. Na pasta *database*, foram tratados os *inserts* de cada classe no banco de dados, e na pasta views, é onde o cadastro está divido por pastas individuais, com seus arquivos de *layout* de inserção e edição.

Para o *front-end*, existe a pasta de *layout*, que contém predefinições de *layout* do *dashboard* do site, a pasta *pages*, que contém modelos de páginas que podem ser aproveitadas presente no tema AdminLTE.

Como página inicial, existe a página de *login*, a qual enviará através de um método *post* os dados de nome de usuário e senha que foram informados no formulário HTML para o *script* de PHP, que foi chamado de session.php, conforme Listagem 1.

Listagem 1 - Arquivo de início de sessão

```
<?php
session_start();
$username = $_POST['username'];
$password  = md5($_POST['password']);

if($username == NULL || $password == NULL){
      echo "<script>alert('Você deve digitar seu nome e senha');</script>";
      echo "<script> window.location.href='../login.php'</script>";
      exit;
}else{
      require_once 'Models/connect.php';
      $connect->login($username, $password);
}
?>
```

Nesse arquivo, é iniciada a sessão, são definidas variáveis globais do PHP ($_POST), para coletar os dados que foram enviados no formulário de *login*, e é realizada a encriptação da senha enviada com o algoritmo de encriptação md5.

O arquivo da sessão executa uma função de *login*, a qual está presente em uma classe chamada *Connect*, para realizar a conexão com o banco de dados, a qual é apresentada na Listagem 2.

Essa conexão é realizada pela extensão MySQLi, a qual é um *driver* de banco de dados relacional, que fornece uma interface para comunicação do PHP com o banco de dados MySQL, permitindo assim, a consulta ao banco de dados.

Listagem 2 - Classe Connect

```
class Connect
 {
      var $localhost = "localhost";
      var $root = "***";
      var $passwd = "***";
      var $database = "controledejornada";
      var $SQL;

      public function __construct()
      {
            $this->SQL   =   mysqli_connect($this->localhost,   $this->root,
$this->passwd);
            mysqli_select_db($this->SQL, $this->database);
            if(!$this->SQL){
                  die("Conexão   com   o   banco   de   dados   falhou!:"   .
mysqli_connect_error($this->SQL));
            }
      }
```

```
      function login($username, $password){
            $this->query = "SELECT * FROM `usuario` WHERE `ususername` =
'$username'";
            $this->result   =   mysqli_query($this->SQL,   $this->query)   or
die(mysqli_error($this->SQL));
            $this->total = mysqli_num_rows($this->result);

            if($this->total){
                  $this->dados = mysqli_fetch_array($this->result);
                  if(!strcmp($password, $this->dados['ususenha'])){
                        $_SESSION['idUsuario'] = $this->dados['usucod'];
                        $_SESSION['usuario'] = $this->dados['ususername'];
                        $_SESSION['perm'] = $this->dados['usupermissao'];
                        $_SESSION['foto'] = $this->dados['usuimagem'];
                        header("Location: ../views/");
                  }else{
                        header("Location: ../login.php?alert=2");
                  }
            }else{
                        header("Location: ../login.php?alert=1");
                  }
      }
 }

$connect = new Connect;
```

Fonte: Autoria própria.

Na classe *Connect* se encontram as propriedades de conexão do banco de dados com o MySQL. A função *login* executa uma *query* com as informações enviadas na tela de *login* do sistema e no banco, e inicia uma sessão caso os dados correspondam à informação no banco.

Por fim, a autenticação é controlada por um arquivo auth.php, que verifica se uma sessão de usuário foi declarada, e em caso contrário, retorna à tela de login. A Listagem 3 apresenta o código de controle de autenticação dos usuários.

Listagem 3 - Arquivo auth.php

```
<?php
session_start(); //Iniciando a sessão
if(!isset($_SESSION["idUsuario"]) || !isset($_SESSION["usuario"])){
                  header('Location: ../login.php');
}else{
      $idUsuario = $_SESSION["idUsuario"];
      $usuario   = $_SESSION["usuario"];
      $perm    = $_SESSION["perm"];
      $foto      = $_SESSION["foto"];
}
?>
```

Fonte: Autoria própria.

Esse arquivo auth.php é incluído em todas as *views* do sistema, por meio da declaração *require_once*, de modo que o sistema obriga que o usuário tenha iniciado a sessão para visualizar as telas.

Na classe *Connect* se encontram as propriedades de conexão do banco de dados com o MySQL. A função *login* executa uma *query* com as informações enviadas na tela de login do sistema e no banco, e inicia uma sessão caso os dados correspondam à informação no banco.

Na pasta *layout*, foi definido um arquivo chamado *script.php,* que apresenta trechos de código de *layout* do *dashboard* reutilizável, que são usados posteriormente nos *views*.

A Listagem 4 apresenta parte do código fonte de uma das views do sistema, o arquivo addveiculo.php, demonstrando uso da chamada de trechos de código pré-definidos para a *view*. Por meio do comando *echo* do PHP, são renderizados os trechos de código HTML do arquivo script.php, e com isso, são gerados os layouts de cada uma das *views* de cadastro.

Listagem 4 - Trecho da renderização do cabeçalho do cadastro de veículos

```
echo $head;
echo $header;
echo $aside;
echo '<div class="content-wrapper">';
echo '<!-- Content Header (Page header) -->
    <section class="content-header">
      <h1>
        Adicionar <small>Veiculos</small>
      </h1>
      <ol class="breadcrumb">
        <li><a href="../"><i class="fa fa-dashboard"></i> Home</a></li>
        <li class="active">Veiculos</li>
      </ol>
    </section>

    <!-- Main content -->
    <section class="content">
      <!-- Small boxes (Stat box) -->
      <div class="row">';
```

Fonte: Autoria própria.

Ainda no arquivo de *views*, existe o formulário de cadastro, o qual envia via método *post* todos os dados dos campos que foram preenchidos para um arquivo da pasta “database” que se encarregará com a inserção no banco. Na Listagem 5, é exibido trecho de código da *view* de cadastro que faz referência a ação a ser realizada pelo formulário.

Listagem 5 - Trecho da chamada do método post de um formulário de cadastro

```
echo ' <a href="./" class="btn btn-success">Voltar</a>
      <div class="row">
        <!-- left column -->
        <div class="col-md-6">
          <!-- general form elements -->
          <div class="box box-primary">
            <div class="box-header with-border">
              <h3 class="box-title">Veículo</h3>
            </div>
```

```
            <!-- /.box-header -->
            <!-- form start -->
            <form role="form" action="../../App/Database/insertveiculo.php"
method="POST">
              <div class="box-body">
                  <div class="form-group">
            </div>
```

Fonte: Autoria própria.

Na pasta Database, são recebidos os dados enviados por cada um dos formulários das *views*. Na Listagem 6, é apresentado o arquivo insertregra.php. Ele recebe os dados do *post* do formulário regra, verifica se a regra não tem o seu nome com o valor nulo, e então analisa se é uma operação de cadastro ou edição de regra. Dependendo do caso, ele irá executar a função de edição ou inserção de regra, estando essas funções no *model* da classe.

Listagem 6 - Trecho da chamada do método post de um formulário de cadastro

```
<?php
require_once '../auth.php';
require_once '../Models/regras.class.php';

if(isset($_POST['upload']) == 'Cadastrar'){

$rjoNome = $_POST['rjonome'];
$rjoTrabInint = $_POST['rjotrabinint'];
$rjoDesc = $_POST['rjodesc'];
$rjoDescInint = $_POST['rjodescinint'];
$rjoRefeicao = $_POST['rjorefeicao'];
$rjoMaxDia = $_POST['rjomaxdia'];
$rjoExtra = $_POST['rjoextra'];
$rjoEspRep = $_POST['rjoesprep'];
$rjoDescInter = $_POST['rjodescinter'];

if($rjoNome != NULL){
      if(isset($_POST['rjocod'])){

            $rjoCod = $_POST['rjoCod'];
            $regras->editRegras($rjoCod, $rjoNome, $rjoTrabInint, $rjoDesc,
$rjoDescInint, $rjoRefeicao, $rjoMaxDia, $rjoExtra, $rjoEspRep,
$rjoDescInter);
      }else{
            $regras->InsertRegras($rjoNome, $rjoTrabInint, $rjoDesc,
$rjoDescInint, $rjoRefeicao, $rjoMaxDia, $rjoExtra, $rjoEspRep,
$rjoDescInter);
      }

}else{
      header('Location: ../../views/regras/index.php?alert=3');
 }
}else{
      header('Location: ../../views/regras/index.php');
}
```

Fonte: Autoria própria.

Na pasta *layout*, foi definido um arquivo chamado script.php, que apresenta trechos de código de layout do dashboard reutilizável, que são usados posteriormente nos views. No Apêndice 1 é apresentado um trecho do script.php, da seção do menu lateral da área administrativa do sistema.

Para que seja possível que cada motorista respeite uma regra de jornadas específica, em seu cadastro é definida uma *foreign key* com a tabela REGRASJORNADA. O Apêndice 2 apresenta a tela do *model* do Motorista, com seus métodos de inserção, atualização de dados e listagem de motoristas.

5 CONCLUSÃO

O objetivo deste trabalho foi realizar o desenvolvimento de um sistema web que permita o controle de jornadas de trabalho de motoristas profissionais, com o objetivo de facilitar o cadastro e verificações de cumprimento de regras de jornada.

Um sistema web foi desenvolvido, e a modelagem criada, assim como detalhes de implementação do sistema foram apresentados. O trabalho foi idealizado inicialmente com o objetivo de se desenvolver dois sistemas, um sistema web e um aplicativo *mobile* complementar, contudo, até o momento apenas o sistema web foi desenvolvido. Assim, o desenvolvimento do aplicativo *mobile* permanece como uma possibilidade de implementação futura para ampliar a flexibilidade e facilidade no cadastro de jornadas de trabalho por parte dos motoristas.

Entre as dificuldades encontradas, pode-se citar a definição do banco de dados, pois foi utilizado o pacote de dados Xampp para instalação de PHP e MySQL, contudo, a versão atual do pacote do Xampp vem com o MariaDB, que embora seja um *fork* do MySQL, apresenta algumas diferenças que dificultaram a criação de tabelas geradas no MySQL Workbench. Foi necessário procurar outra versão do pacote, mais antiga, que realmente vinha com o MySQL. Além disso, a utilização do *driver* de banco de dados MySQLi impedia a utilização de outro banco de dados, sendo que seria necessária que fosse reescrito toda a base de código relativa a conexão com o banco para outro driver de banco de dados, como o PHP PDO, que permitiria conexão com outros bancos de dados além do SQL.

Outra dificuldade foi a validação de jornadas, que não foi totalmente concluída, principalmente pela quantidade de regras e o cálculo envolvido entre elas, por exemplo, a regra de intervalo entre jornadas deve avaliar, como variáveis, a última jornada que foi realizada por aquele motorista, o último tempo de trabalho real da última jornada (deveria desconsiderar todo o tempo de descanso ou espera que poderia ter ocorrido ao fim da última jornada) e utilizar o número de horas interjornada da regra. As outras demais regras também possuem validações específicas que deveriam ser executadas a cada inserção.

Com o controle de cadastro de jornadas desenvolvido, é possível se obter o registro de jornadas de trabalho de um motorista, processo este que era praticado costumeiramente de forma manual, passível de erros humanos.

Como perspectiva futura, existe a conclusão dos cálculos de validação de jornada para possibilitar a geração de relatórios relevantes sobre as jornadas de cada motorista, e do desenvolvimento de um aplicativo que seja voltado especificamente ao motorista, no qual ele

poderá realizar o registro de sua própria jornada, resgatar informações do GPS de sua localização, e até mesmo o registro *off-line* de jornadas, assim como a geração de relatórios avançados sobre as jornadas de motorista, na qual o empregador teria acesso detalhado às jornadas realizadas pelos seus motoristas.

REFERÊNCIAS

AKAMA, Kiyoshi; NANTAJEEWARAWAT, Ekawit; OGASAWARA, Hidemi. **Generation of correct parallel programs based on specializer generation transformations**. In: Proceedings of the 7th international conference on intelligent technologies. 2006.

CAMERON. O'Rourke. A look at rich internet application. **Oracle Magazine**. Jul./ago., 2004, p. 1-4.

CIRILO, Carlos E. et al. Model driven richubi-a model-driven process to construct rich interfaces for context-sensitive ubiquitous applications. In: **2010 Brazilian Symposium on Software Engineering**. IEEE, 2010. p. 100-109.

COELHO, Luciano Augusto de Toledo. **As Leis 12.619/2012 e 13.103/2015 e flexibilização da jornada de trabalho do motorista em transporte rodoviário de passageiros e de cargas**. Revista eletrônica [do] Tribunal Regional do Trabalho da 9ª Região, Curitiba, PR, v. 5, n. 45, p. 35-49, 2015.

CONFEDERAÇÃO NACIONAL DO TRANSPORTE. **Plano CNT de recuperação econômica**. Disponível em: http://cms.cnt.org.br/Imagens%20CNT/PDFs%20CNT/Plano%20CNT%20de%20Recupera%C3%A7%C3%A3o%20Econ%C3%B4mica/PLANO_CNT_DE_RECUPERACAO_ECONOMICA.pdf. Acesso em: 02 set. 2020.

CORREIA, E. J. **What's next for HTML 5.** Intel software adrenaline, 2013.
DISSANAYAKE, Nalaka R.; DIAS, G. Kapila A. **Best Practice for rapid application development of AJAX based Rich Internet Applications**. In: 2014 International Conference on Advances in ICT for Emerging Regions (ICTer), 2014, p. 63-66.

CRAIDE, Sabrina. **Jornada prolongada de trabalho de motoristas duplica riscos de acidente de trânsito**. Empresa Brasil de Comunicação. Disponível em: http://memoria.ebc.com.br/agenciabrasil/noticia/2013-07-20/jornada-de-trabalho-prolongada-de-motoristas-duplica-risco-de-acidentes-de-transito. 2013. Acesso em: 02 set. 2020.

DUHL, Joshua. **White paper: Rich Internet Applications**. Technical report, IDC, November 2003.

FRATERNALI, P.; Rossi, G.; SANCHEZ-FIGUEROA, F. **Rich Internet Applications**. IEEE Internet Computing, 9: 1 2 may/june, 2010. IEEE Computer Society.

LAWTON, George. New ways to build rich Internet applications. **Computer.** Published by the IEEE Computer Society, p. 10-12, 2008.

MACHADO, Caren Silva; GOLDSCHMIDT, Rodrigo. Um olhar constitucional sobre a jornada de trabalho do motorista profissional partindo de uma análise da nova Lei do Motorista. **Simpósio Internacional de Direito: Dimensões materiais e eficaciais dos direitos fundamentais**, v. 2, n. 2, p. 683-719, 2012.

MARTÍNEZ-RUIZ, Francisco J. **The Triad-based Design of Rich User Interfaces for Internet Applications**. ACM, 2010, p. 345-348.

POWELL, Courtney; NAKAMURA, Keisuke; Akama, Kiyoshi. **Towards a Formal Behavioral Model for Rich Internet Applications**. In: International Conference on Computational Intelligence and Software Engineering, 2009 (CiSE 2009), p. 1-5.

TOLEDO FILHO, Manoel Carlos; NEVES, Bruno Hiroshi Kuae. A nova disciplina da jornada de trabalho do motorista profissional. **Revista Direito Mackenzie**, v. 6, n. 1, 2012.

VALVERDE, F.; PASTOR, O. **Facing the Technological Challenges of Web 2. 0: a RIA Model-driven Engineering Approach.** In Proc. of the International Co'!ference on WISE, WISE 2009,

PHP.NET. **PHP: O que é o PHP? - Manual.** [S.I]. [2020?]. Disponível em: https://www.php.net/manual/pt_BR/intro-whatis.php. Acesso em: 27 nov. 2020.

YAMAMOTO, Lucas Zucoli. **Lei do caminhoneiro e a jornada de trabalho de motoristas.** Federação Nacional dos Trabalhadores Celetistas nas Cooperativas no Brasil. 2015. Disponível em: http://www.sintracoopsp.com.br/?p=35292. Acesso em: 02 set. 2020.

APÊNDICE 1 – CÓDIGO FONTE DO SCRIPT.PHP

```
  $aside = '<!-- Left side column. contains the logo and sidebar -->
  <aside class="main-sidebar">
    <!-- sidebar: style can be found in sidebar.less -->
    <section class="sidebar">
      <!-- Sidebar user panel -->
      <div class="user-panel">
        <div class="pull-left image">
          <img src="'.$url.''.$foto.'" class="img-circle" alt="User Image">
        </div>
        <div class="pull-left info">
          <p>'.$usuario.'</p>
          <a href="#"><i class="fa fa-circle text-success"></i> Online</a>
        </div>
      </div>
      <!-- search form -->
      <form action="#" method="get" class="sidebar-form">
        <div class="input-group">
          <input type="text" name="q" class="form-
control" placeholder="Pesquisa...">
              <span class="input-group-btn">
                <button type="submit" name="search" id="search-
btn" class="btn btn-flat"><i class="fa fa-search"></i>
                </button>
              </span>
        </div>
      </form>
      <!-- /.search form -->
      <!-- sidebar menu: : style can be found in sidebar.less -->
      <ul class="sidebar-menu">
        <li class="header">NAVEGAÇÃO</li>

        <li class="treeview">
        <a href="#">
          <i class="fa fa-table"></i>
          <span>Usuários</span>
          <span class="pull-right-container">
            <i class="fa fa-angle-left pull-right"></i>
          </span>
        </a>
        <ul class="treeview-menu">
          <li><a href="'.$url.'usuario/"><i class="fa fa-circle-
o"></i>Lista de Usuários</a></li>
          <li><a href="'.$url.'usuario/addusuario.php"><i class="fa fa-
circle-o"></i>Adicionar Usuarios</a></li>
        </ul>
      </li>

      <li class="treeview">          <a href="#">

            <i class="fa fa-table"></i>
            <span>Escopos</span>
            <span class="pull-right-container">
              <i class="fa fa-angle-left pull-right"></i>
            </span>
          </a>
```

```
          <ul class="treeview-menu">
            <li><a href="'.$url.'escopo/"><i class="fa fa-circle-
o"></i>Lista de Escopos</a></li>
            <li><a href="'.$url.'escopo/addescopo.php"><i class="fa fa-
circle-o"></i>Adicionar Escopos</a></li>
          </ul>
        </li>

        <li class="treeview">
          <a href="#">
            <i class="fa fa-table"></i>
            <span>Veículos</span>
            <span class="pull-right-container">
              <i class="fa fa-angle-left pull-right"></i>
            </span>
          </a>
          <ul class="treeview-menu">
            <li><a href="'.$url.'veiculo/"><i class="fa fa-circle-
o"></i>Lista de Veículos</a></li>
            <li><a href="'.$url.'veiculo/addveiculo.php"><i class="fa fa-
circle-o"></i>Adicionar Veículos</a></li>
          </ul>
        </li>

        <li class="treeview">
        <a href="#">
          <i class="fa fa-table"></i>
          <span>Motoristas</span>
          <span class="pull-right-container">
            <i class="fa fa-angle-left pull-right"></i>
          </span>
        </a>
        <ul class="treeview-menu">
          <li><a href="'.$url.'motorista/"><i class="fa fa-circle-
o"></i>Lista de Motoristas</a></li>
          <li><a href="'.$url.'motorista/addmotorista.php"><i class="fa fa-
circle-o"></i>Adicionar Motorista</a></li>
        </ul>
      </li>

      <li class="header">REGRAS DE JORNADA</li>

      <li class="treeview">
        <a href="#">
          <i class="fa fa-table"></i>
          <span>Regras</span>
          <span class="pull-right-container">
            <i class="fa fa-angle-left pull-right"></i>
          </span>
        </a>
        <ul class="treeview-menu">
          <li><a href="'.$url.'regras/"><i class="fa fa-circle-
o"></i>Lista de Regras</a></li>
          <li><a href="'.$url.'regras/addregra.php"><i class="fa fa-circle-
o"></i>Adicionar Regras</a></li>
        </ul>
      </li>
```

```
      <li class="header">CADASTRO DE JORNADAS</li>

      <li class="treeview">
        <a href="#">
          <i class="fa fa-table"></i>
          <span>Jornadas</span>
          <span class="pull-right-container">
            <i class="fa fa-angle-left pull-right"></i>
          </span>
        </a>
        <ul class="treeview-menu">
          <li><a href="'.$url.'regras/"><i class="fa fa-circle-
o"></i>Lista de Regras</a></li>
          <li><a href="'.$url.'regras/addregra.php"><i class="fa fa-circle-
o"></i>Adicionar Regras</a></li>
        </ul>
      </li>
    </section>
    <!-- /.sidebar -->
  </aside>';
```

APÊNDICE 2 – CÓDIGO FONTE DO MODEL MOTORISTA

```
<?php
/*
 Class Motorista
*/
 require_once 'connect.php';
class Motorista extends Connect {

  public function index() {
            $this->query = "SELECT * FROM `motorista`";
            $this->result = mysqli_query($this->SQL, $this->query) or die
( mysqli_error($this->SQL));
            if($this->result){
                  while ($row = mysqli_fetch_array($this->result)) {
                        echo '<li>
                  <!-- drag handle -->
                      <span class="handle">
                        <i class="fa fa-ellipsis-v"></i>
                        <i class="fa fa-ellipsis-v"></i>
                      </span>
                  <span class="text">
                  <!-- checkbox -->
                  <form     class="badge"     name="ativ'.$row['motcod'].'"
action="action.php" method="post">
                  <input       type="hidden"       name="id"       id="id"
value="'.$row['motcod'].'">
                  <input type="checkbox" id="status" name="status"</form>
                  <!-- todo text -->
                  <span    class="badge    left">'.$row['motcod'].'</span>
'.$row['motnome'].'</span>

                  <div class="tools right">

                    <form                                        class="right"
name="editMotorista'.$row['motcod'].'"          action="editMotorista.php"
method="post">
                      <input   type="hidden"   name="motcod"   id="motcod"
value="'.$row['motcod'].'">
                      <a               href="#"               type="button"
onclick="this.form.submit();"><i class="fa fa-edit"></i></a></form>
                      <form                                        class="right"
name="delMotorista'.$row['motcod'].'"            action="delMotorista.php"
method="post">
                      <input    type="hidden"    name="motcod"    id="motcod"
value="'.$row['motcod'].'">
                       <a               href="#"               type="button"
onclick="this.form.submit();"><i class="fa fa-trash-o"></i></a></form>
                  </div>
                </li>';
                  }
            }
      }

   public    function    InsertMotorista($motNome,    $motRua,    $motBairro,
$motCidade, $motEstado, $motEmail, $motSenha, $motFone, $motCpf, $motRg,
$motCnh, $motDtAdmissao, $motDtAfast, $motSituacao, $regrasjornada_rjocod){
```

```
     $this->query = "INSERT INTO `motorista`(`motcod`, `motnome`, `motrua`,
`motbairro`, `motcidade`, `motestado`, `motemail`, `motsenha`, `motfone`,
`motcpf`, `motrg`, `motcnh`, `motdtadmissao`, `motdtafast`, `motsituacao`,
`regrasjornada_rjocod`) VALUES (NULL, '$motNome', '$motRua', '$motBairro',
'$motCidade', '$motEstado', '$motEmail', '$motSenha', '$motFone', '$motCpf',
'$motRg',                    '$motCnh',                    '$motDtAdmissao',
'$motDtAfast','$motSituacao','$regrasjornada_rjocod')";
           if($this->result = mysqli_query($this->SQL, $this->query) or
die(mysqli_error($this->SQL))){


      header('Location: ../../views/motorista/index.php?alert=1');
            }else{

      header('Location: ../../views/motorista/index.php?alert=0');
            }
      }//InsertItens

  public function editMotorista($value) {
    $this->query = "SELECT *FROM `motorista` WHERE `motcod` = '$value'";
    $this->result   =   mysqli_query($this->SQL,   $this->query)   or   die
( mysqli_error($this->SQL));

    if($row = mysqli_fetch_array($this->result)){
      $motcod = $row['motcod'];
      $motNome = $row['motnome'];
      $motRua = $row['motrua'];
      $motBairro = $row['motbairro'];
      $motCidade = $row['motcidade'];
      $motEstado = $row['motestado'];
      $motEmail = $row['motemail'];
      $motSenha = $row['motsenha'];
      $motFone = $row['motfone'];
      $motCpf = $row['motcpf'];
      $motRg = $row['motrg'];
      $motCnh = $row['motcnh'];
      $motDtAdmissao = $row['motdtadmissao'];
      $motDtAfast = $row['motdtafast'];
      $motSituacao = $row['motsituacao'];
      $regrasjornada_rjocod = $row['regrasjornada_rjocod'];

      return $resp = array('Itens' => ['motcod' => $motcod,
      'motnome'   => $motNome,
      'motrua' => $motRua,
      'motbairro' => $motBairro,
      'motcidade'   => $motCidade,
      'motestado' => $motEstado,
      'motemail' => $motEmail,
      'motsenha'   => $motSenha,
      'motfone' => $motFone,
      'motcpf' => $motCpf,
      'motrg'   => $motRg,
      'motcnh' => $motCnh,
      'motdtadmissao' => $motDtAdmissao,
      'motdtafast'   => $motDtAfast,
      'motsituacao' => $motSituacao,
      'regrasjornada_rjocod' => $regrasjornada_rjocod] , );
     }
  }
```

```
  public function updateMotorista($motCod, $motNome, $motRua, $motBairro,
$motCidade, $motEstado, $motEmail, $motSenha, $motFone, $motCpf, $motRg,
$motCnh, $motDtAdmissao, $motDtAfast, $motSituacao, $regrasjornada_rjocod)
{
    $this->query = "UPDATE `motorista` SET
                    `motnome`= '$motNome',
                    `motrua`='$motRua',
                    `motbairro`='$motBairro',
                    `motcidade`= '$motCidade',
                    `motestado`='$motEstado',
                    `motemail`='$motEmail',
                    `motsenha`= '$motSenha',
                    `motfone`='$motFone',
                    `motcpf`='$motCpf',
                    `motrg`= '$motRg',
                    `motcnh`='$motCnh',
                    `motdtadmissao`='$motDtAdmissao',
                    `motdtafast`= '$motDtAfast',
                    `motsituacao`='$motSituacao',
                    `regrasjornada_rjocod`='$regrasjornada_rjocod',
                    WHERE `motcod`= '$motCod'";

    if   ($this->result   =   mysqli_query($this->SQL,   $this->query)   or
die(mysqli_error($this->SQL))){
      header('Location: ../../views/motorista/index.php?alert=1');
    } else {
      header('Location: ../../views/motorista/index.php?alert=0');
    }
  }

  public function listMotorista($value = NULL){
    $this->query = "SELECT * FROM `motorista`";
    $this->result   =   mysqli_query($this->SQL,   $this->query)   or   die
( mysqli_error($this->SQL));
    if($this->result){
      while ($row = mysqli_fetch_array($this->result)) {
            if($value == $row['motcod']){
               $selected = "selected";
           }else{
               $selected = "";
           }
        echo              '<option              value="'.$row['motcod'].'"
'.$selected.' >'.$row['motnome'].'</option>';
      }
    }
  }
 }

 $motoristas = new Motorista;
```