UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE INFORMÁTICA

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

LUCAS ISTAK

RENATO WILLIAN MACHADO

SISTEMA DE MAPEAMENTO DE PROCEDIMENTOS DE SAÚDE PARA PACIENTES DE UMA UNIDADE DE PRONTO ATENDIMENTO DE PONTA GROSSA

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA

2019

LUCAS ISTAK

RENATO WILLIAN MACHADO

SISTEMA DE MAPEAMENTO DE PROCEDIMENTOS DE SAÚDE PARA PACIENTES DE UMA UNIDADE DE PRONTO ATENDIMENTO DE PONTA GROSSA

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, do Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Diego Roberto Antunes.

PONTA GROSSA

2019

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Ponta Grossa

Diretoria de Graduação e Educação Profissional
Departamento Acadêmico de Informática
Tecnologia em Análise e Desenvolvimento de Sistemas

TERMO DE APROVAÇÃO

SISTEMA DE MAPEAMENTO DE PROCEDIMENTOS DE SAÚDE PARA PACIENTES DE UMA UNIDADE DE PRONTO ATENDIMENTO DE PONTA GROSSA

por

LUCAS ISTAK
RENATO WILLIAN MACHADO

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 13 de novembro de 2019 como requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas. Os candidatos foram arguidos pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Prof. Dr. Diego Roberto Antunes
Orientador

_____
Prof. Msc. Victor Schnepper Lacerda
Membro titular

_____
Prof. Msc. Giancarlo Rodrigues
Membro titular

_____
Prof. MSc. Geraldo Ranthum
Responsável pelo Trabalho de Conclusão de Curso

_____
Prof. Dr. André Pinz Borges
Coordenador do curso

AGRADECIMENTOS

Primeiramente gostaríamos de agradecer a Deus, por ter nos cedido forças para vencer os obstáculos que surgiram nessa caminhada.

Agradecemos aos nossos familiares por sempre estarem ao nosso lado, nos apoiando e incentivando com sabedoria.

Agradecemos ao nosso orientador Prof. Dr. Diego Roberto Antunes pela orientação e por não medir esforços para esclarecer nossas dúvidas no decorrer desse trabalho.

Agradecemos a UPA Santa Paula, por terem nos disponibilizado a estrutura e demais recursos para que o desenvolvimento desse trabalho fosse possível.

E por fim, a todos os professores da UTFPR que fizeram parte dessa caminhada, pelos ensinamentos de cada um que contribuíram para nossa formação.

RESUMO

ISTAK, Lucas; MACHADO, Renato Willian. **Sistema de Mapeamento de Procedimentos de Saúde para Pacientes de uma Unidade de Pronto Atendimento de Ponta Grossa**. 2019. 74 f. Trabalho de Conclusão de Curso (Tecnologia em Análise Desenvolvimento de Sistemas) - Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2019.

Pacientes de Unidades de Pronto Atendimento, muitas vezes, não possuem fácil acesso a informações referentes ao atendimento durante sua permanência na unidade, reduzindo assim sua participação no seu cuidado. Buscando a solução para tal problema, este trabalho propôs o desenvolvimento de um sistema *web* capaz de disponibilizar informações do atendimento aos pacientes de uma UPA 24h, a qual está sediada na cidade de Ponta Grossa/PR e recebe um fluxo diário de 300 a 400 pacientes em média. Para tal feito, o servidor foi desenvolvido com o *framework* PHP Laravel e o banco de dados relacional MariaDB, o qual armazena informações coletadas do sistema de gestão hospitalar Philips Tasy EMR. Esses dados são apresentados em tempo real para o cliente utilizando tecnologias como *websockets* e *task scheduling*, para mantê-los atualizados, e o *framework* JavaScript Vue, para apresentação desses dados de forma reativa, disponibilizando uma melhor interação com o cliente.

Palavras-chave: Sistema *web*. *Framework* Laravel. MariaDB. Tasy EMR. *Websockets*. *Task Scheduling*. *Framework* Vue.js.

ABSTRACT

ISTAK, Lucas; MACHADO, Renato Willian. **Health Procedures Mapping System for Patients at a Ponta Grossa Emergency Care Unity**. 2019. 74 p. Work of Conclusion Course (Graduation in Technology in Analysis and Development of Systems) - Federal Technology University - Paraná. Ponta Grossa, 2019.

Patients in Emergency Care Units often do not have easy access to information regarding care during their stay in the unit, thus reducing their participation in their care. Seeking the solution to this problem, this work proposes the development of a web system capable of providing patient care information of a UPA 24h, which is located in the city of Ponta Grossa/PR and receives a daily flow of 300 to 400 patients on average. To do this, the server was developed with the PHP Laravel framework and the MariaDB relational database, which stores information collected from the Philips Tasy EMR hospital management system. This data is presented in real time to the client using technologies such as websockets and task scheduling to keep it up to date, and the JavaScript Vue framework for reactively presenting this data, providing better interaction with the client.

Keywords: WEB System. Laravel Framework. MariaDB. Tasy EMR. Websockets. Task Scheduling. Vue.js Framework.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

API	Application Programming Interface
CFM	Conselho Federal de Medicina
EMR	Electronical Medical Record
ERP	Enterprise Resource Planning
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IBSP	Instituto Brasileiro para Segurança do Paciente
IoC	Inversion of Control Container
IOM	Institute Of Medicine
MVC	Model-View-Controller
NEP	Núcleo de Educação Permanente
OMS	Organização Mundial de Saúde
ORM	Object Relational Mapping
PHP	Personal Home Page
PL/SQL	Procedural Language/Structured Query Language
PSM	Patient Safety Movement
PWA	Progressive Web App
RAU	Rede de Atenção às Urgências
SGBD	Sistema Gerenciador de Banco de Dados
SMS	Secretaria Municipal de Saúde
SPA	Single Page Application
SQL	Structured Query Language
SSH	Secure Shell
XML	Extensible Markup Language

LISTA DE ACRÔNIMOS

AJAX	Asynchronous JavaScript And XML
COREN-SP	Conselho Regional de Enfermagem de São Paulo
DOM	Document Object Model
EUA	Estados Unidos da América
JSON	JavaScript Object Notation
ONA	Organização Nacional de Acreditação
PEP	Prontuário Eletrônico do Paciente
REBRAENSP	Rede Brasileira de Enfermagem e Segurança do Paciente
SAMU	Serviço de Atendimento Móvel de Urgência
SBIS	Sociedade Brasileira de Informática em Saúde
TICS	Tecnologia da Informação e Comunicação em Saúde
UPA	Unidade de Pronto Atendimento
VSCode	Visual Studio Code

SUMÁRIO

1 INTRODUÇÃO

Ambientes em que há baixa interação do profissional de saúde com o paciente, devido à falta de informações cedidas a este, são considerados inseguros. Contudo, segundo Silva et al. (2016), há uma inversão de cenário na proporção em que essa interação aumenta.

O paciente tem o direito de requerer uma cópia de seu prontuário, e o médico não pode negar esse acesso, conforme mencionado pela Agência CNJ de Notícias (2015), contudo, a burocracia que envolve este direito torna, muitas vezes, o acesso demorado e complexo. Visto que a obtenção dos dados do prontuário já é um direito do paciente, é interessante disponibilizá-los durante o atendimento a fim de informar ao paciente a respeito de seu próprio atendimento em uma instituição de cuidado à saúde.

Segundo Laranjo et al. (2013), o compartilhamento de informações clínicas deve ser estimulado, visto que este processo auxilia na capacitação do profissional, fornece mais autonomia ao paciente e, assim, resulta em uma maior satisfação das pessoas e, principalmente, em possíveis avanços tecnológicos e disseminação dos Sistemas Personalizados de Informação de Saúde.

De acordo com o Instituto Brasileiro para Segurança do Paciente (IBSP) (PSM, 2018), estima-se que 80% dos erros clínicos graves envolvam a falta de comunicação entre os profissionais da saúde durante alguma transição do paciente. Assim, é de extrema importância que o paciente contribua para a qualidade do seu atendimento devido ao conhecimento sobre o seu histórico de saúde e experiências com os tratamentos aos quais já foi submetido.

Um ambiente que torna os pacientes agentes ativos em prol de sua segurança, promove interesse, motivação e satisfação com o cuidado prestado, possibilitado um bom resultado nas condições de saúde. Uma das maneiras de envolver o paciente na assistência prestada, é disponibilizar a ele informações pertinentes do seu atendimento durante sua permanência na unidade de saúde (AVELAR et al., 2010).

No mercado, existem alguns sistemas de informação que auxiliam nesta questão, porém, ainda possuem poucos recursos destinados ao paciente como, por

exemplo, os sistemas da Frischmann Aisengart[1] e o MedCloud[2] nos quais somente é possível visualizar informações pertinentes ao resultado de exames, o da SmartClinic[3] e da MV[4], cujo foco é voltado para a gestão clínica, disponibilizando ao paciente apenas algumas funcionalidades como: informações sobre a clínica, equipe médica e consultas. Ou seja, ainda é possível propor melhorias e desenvolver novas soluções.

Considerando o problema relacionado à falta de informações no atendimento de um paciente, este trabalho teve por finalidade desenvolver um sistema de mapeamento de procedimentos a fim de fornecer informações ao paciente em tempo real. Tendo acesso às informações de seu atendimento, o paciente pode dialogar com os profissionais de saúde em prol de sua própria segurança e, além disso, melhorar a confiança entre médico e paciente para gerar respostas positivas em seu atendimento, como mostra um estudo realizado por Batalden et al. (2016).

Tal sistema foi desenvolvido para ambiente *web*, utilizando o *framework* Laravel da linguagem PHP e outras tecnologias que possibilitem a comunicação com a base de dados localizada em uma Unidade de Pronto Atendimento (UPA 24h) e, que dessa forma, os resultados possam ser visualizados pelos usuários de maneira clara e interativa por meio de qualquer dispositivo com acesso à internet (computador, *smartphone*, *tablet*, etc.).

[1] https://labfa.com.br/
[2] https://www.medcloud.com.br/
[3] https://mysmartclinic.com.br/
[4] http://www.mv.com.br/pt/

1.1 OBJETIVO GERAL

O objetivo geral desse trabalho é desenvolver um sistema web capaz de disponibilizar informações sobre o atendimento ao paciente durante sua permanência na unidade de saúde.

1.2 OBJETIVOS ESPECÍFICOS

- Mapear os processos relacionados ao atendimento, com o intuito de criar uma linha do tempo para o atendimento;
- Pesquisar tecnologias que possibilitem o agendamento de tarefas e sincronização de dados em servidores *web*;
- Desenvolver o sistema aplicando as tecnologias definidas.

2 REFERENCIAL TEÓRICO

Neste capítulo é apresentado o referencial teórico necessário para o desenvolvimento desse trabalho, como, informações referentes ao estabelecimento onde este foi desenvolvido incluindo o sistema de gestão hospitalar que utilizam, além de informações referentes as tecnologias e métodos utilizados para o desenvolvimento do sistema proposto.

2.1 UPA 24H – UNIDADE DE PRONTO ATENDIMENTO

De acordo com o Ministério da Saúde (2017), UPA 24h é um estabelecimento de saúde articulado com a Atenção Básica, Serviço de Atendimento Móvel de Urgência (SAMU 192), Atenção Domiciliar e a Atenção Hospitalar. Seu objetivo é prover melhor funcionamento da Rede de Atenção às Urgências (RAU), concentrando os atendimentos de saúde de complexidade intermediária.

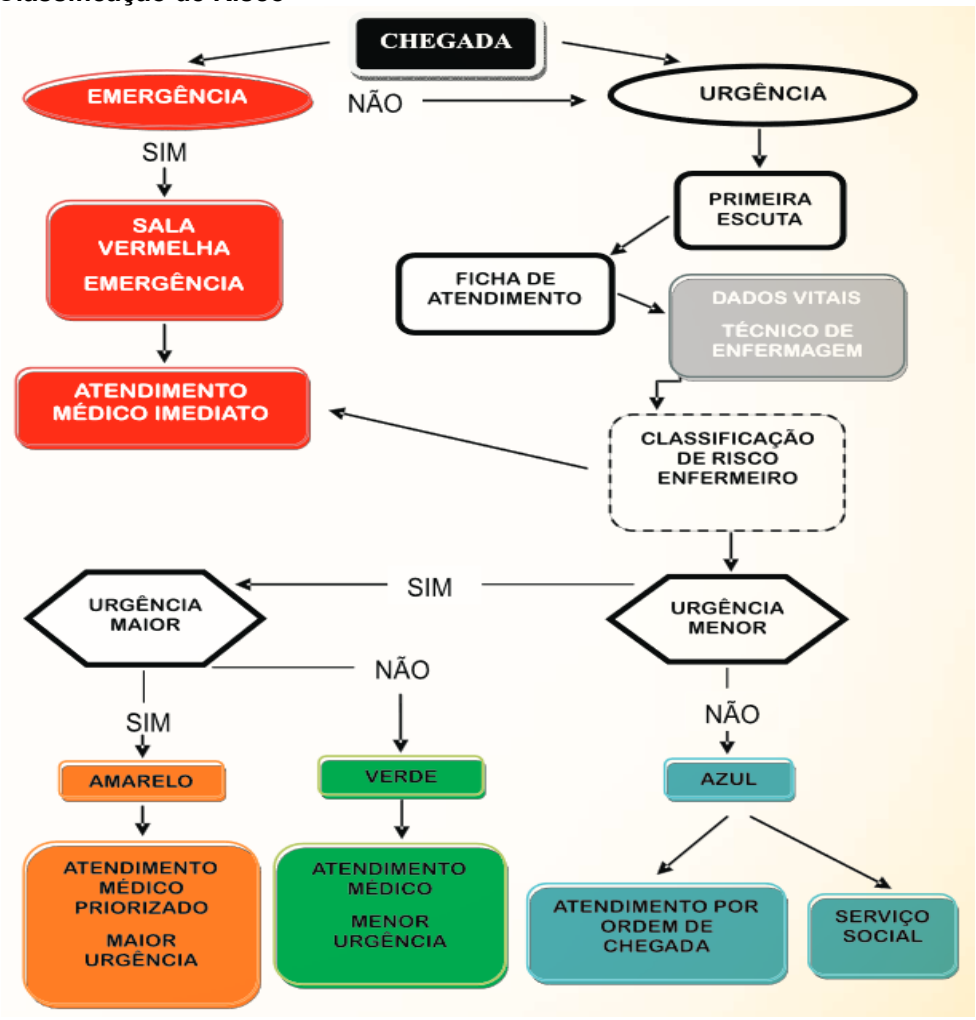
A escolha do estabelecimento de saúde para implantação deste projeto foi a UPA Santa Paula a qual possui vários protocolos clínicos implantados e é uma unidade acreditada plena, pela Organização Nacional de Acreditação (ONA), a qual avalia e certifica a qualidade de serviços de saúde. Além disso, ela faz uso de um sistema de gestão hospitalar denominado Tasy EMR[5], um sistema ERP[6] (*Enterprise Resource Planning*) que possui dentre seus módulos o Prontuário Eletrônico do Paciente (PEP).

Segundo as diretrizes do modelo assistencial e financiamento de UPA 24h de Pronto Atendimento do Ministério da Saúde (BARROS, 2017), a UPA 24h deve conter ferramenta de classificação de risco, acolhimento do paciente, equipe assistencial multiprofissional com quantidade de profissionais compatível com a necessidade de atendimento e atendimento ininterrupto 24 horas em todos os dias da semana, incluindo feriados e pontos facultativo. Seu fluxo de atendimento, como representado pela Figura 1, se divide nos atendimentos de urgência, cujos pacientes

[5] https://www.philips.com.br/healthcare/resources/landing/solucao-tasy
[6] Sistema responsável pela gestão e integração de diversos setores de uma empresa (PEDROSO, 2011).

serão atendidos de acordo com a classificação de risco, e emergência, onde os pacientes são imediatamente atendidos na sala de emergência.

Figura 1 - Fluxograma de Atendimento do Protocolo de Acolhimento com Classificação de Risco



Fonte: Protocolo de Acolhimento com Classificação de Risco (SERVIN, 2002)

Sendo a classificação de risco um objeto de padronização do Ministério da Saúde (2009), como parte do sistema de humanização da assistência, esta é definida como uma ferramenta que organiza a fila de espera e propõe outra ordem de atendimento que não a ordem de chegada. Além disso, a classificação de risco tem como objetivo garantir o atendimento imediato do paciente com grau de risco elevado, informar paciente e familiares que não corre risco imediato, assim como o tempo provável de espera, promover o trabalho em equipe com a avaliação contínua do processo, prover melhores condições de trabalho para os profissionais pela discussão da ambiência e implantação do cuidado horizontalizado, aumentar a

satisfação dos pacientes e, principalmente, possibilitar e instigar a pactuação e a construção de redes internas e externas de atendimento.

No Quadro 1 são descritos a prioridade, característica, definição, cor e o tempo para ser atendido, de cada classificação de risco da UPA Santa Paula.

Quadro 1 - Definições das Classificações de Risco da UPA Santa Paula

Prioridade	Característica	Definição	Cor	Tempo de Atendimento
0	Emergência	Risco imediato de perder a vida	Vermelho	IMEDIATO
1	Urgente	Condição que pode se agravar sem atendimento	Amarelo	Até 1 hora
2	Pouco Urgente	Baixo risco imediato à saúde	Verde	Até 2 horas
3	Não Urgente	Sem risco imediato de agravo à saúde	Azul	Até 4 horas

Fonte: Adaptado de UPA Santa Paula (2017)

2.2 SEGURANÇA DO PACIENTE

A segurança do paciente abrange uma vasta lista de cuidados a serem tomados, com o objetivo de evitar incidentes relacionados ao atendimento do paciente. O PSM (*Patient Safety Movement*) disponibiliza um documento com práticas que buscam melhorar os processos relacionados a segurança do paciente, como por exemplo, maneiras de evitar erros de medicação, infecções hospitalares, entre outros problemas comuns encontrados nos ambientes de cuidado à saúde (ZAMBON, 2018b).

Segundo Silva et al. (2016), os incidentes são considerados consequências do cuidado hospitalar que não possuam relação direta ao motivo pelo qual o paciente procurou atendimento. Estes são dispostos em três classificações:

- **Incidentes sem danos:** nesse caso o dano não é causado ao paciente, porém houve o risco;

- **Eventos adversos:** quando algum tipo de dano é causado ao paciente, como permanência no ambiente hospitalar, piora na saúde, realização de procedimentos extras e até mesmo óbito;
- **Quase erro:** quando ocorre uma situação em que poderia acontecer um incidente, porém foi previamente evitada, não atingindo o paciente.

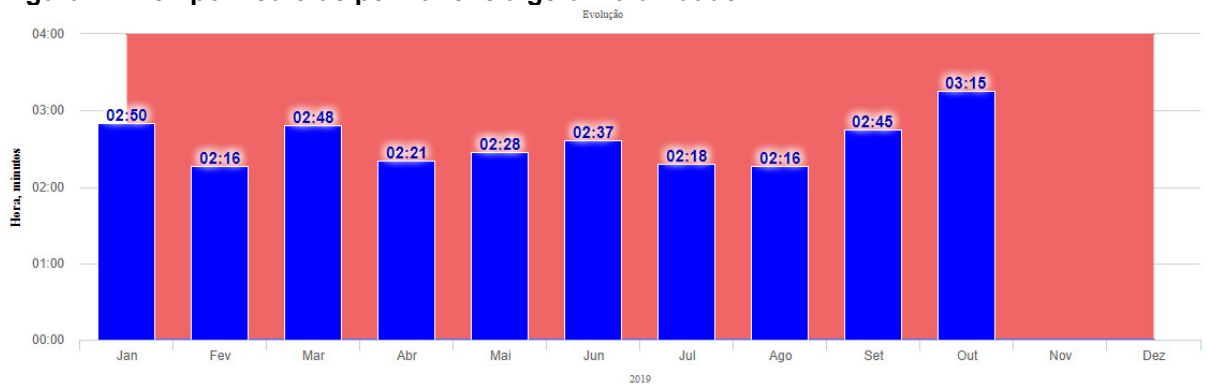
Cerca de 30% dos incidentes são causados antes mesmo dos pacientes adentrarem ao hospital, sendo em consultórios, ambulatórios, etc. Na maioria dos casos esses incidentes são causados por erros de prescrição, tratamento e diagnósticos (ZAMBON, 2018a).

Outro estudo realizado em uma clínica cirúrgica mostra que incidentes como a omissão de cuidado, falha nas checagens e manutenção de equipamentos, erros de medicação, compõem 82% dos incidentes sem danos. A principal preocupação relacionada a segurança do paciente, entretanto, consiste em evitar a ocorrência de eventos adversos, os quais atingem 10% dos pacientes a nível mundial, já no Brasil esse número varia entre 6% e 18,7%. Esse estudo também mostra que aproximadamente 66,7% dessas ocorrências podem ser evitadas (SILVA et al., 2016).

Dentre as boas práticas de enfermagem definidas pela REBRAENSP em conjunto com o COREN-SP, destaca-se o item que apresenta o paciente envolvido em sua segurança. As principais causas desses incidentes estão relacionadas a falha de comunicação entre o paciente e os profissionais de saúde.

Assim como a identificação do paciente é de extrema importância, a comunicação no ambiente hospitalar de forma correta pode fazer muita diferença na causa de um evento adverso, evitando danos graves e até mesmo o óbito (ZAMBON, 2017). Segundo Silva et al. (2016), 46% dos pacientes que tem algum problema de comunicação com o profissional a qual requer atendimento estão sujeitos a um evento adverso, contra 20% dos pacientes que não possuem esse tipo de problema. No caso da UPA Santa Paula, os pacientes ficam aproximadamente 2,5h em média no processo de atendimento como mostra a Figura 2, portanto, durante todo este tempo a troca de informações entre setores e a informação para o paciente é essencial para evitar um possível evento adverso.

Figura 2 – Tempo médio de permanência geral na unidade



Fonte: UPA Santa Paula (2019)

Um dos objetivos fundamentais implementados no sistema de saúde dos Estados Unidos da América (EUA) pelo *Institute of Medicine* (IOM) está centrado no cuidado com o paciente, o qual evidencia que pacientes ativos podem trazer bons resultados para a sua saúde (BATALDEN et al., 2016). Relacionado a isso, no Brasil a Organização Mundial de Saúde (OMS) criou o programa "Pacientes para a Segurança do Paciente", que consiste em reunir profissionais da saúde, pacientes e colaboradores, com o objetivo de tornar o paciente um membro ativo e determinante no seu cuidado.

Outro programa semelhante criado pela *Joint Commission* em 2005 conhecido como *"Speak Up"*, o qual tem por objetivo informar e estimular o paciente a se envolver nos processos relacionados ao seu atendimento por meio de perguntas e até mesmo auxiliando na tomada de decisões (SILVA et al., 2016).

Outra falha cada vez mais encontrada nos ambientes hospitalares, ocorre pelos erros de comunicação entre os profissionais durante uma transição de cuidado, por conta da evolução na especialização das áreas de cuidado. Devido a isso, o Centro de Saúde da Universidade da Califórnia (EUA) adotou um sistema que busca padronizar esse tipo de transição realizando registros de forma eletrônica. Segundo um estudo publicado no *The New England Journal of Medicine*, esse sistema propõe uma redução de 23% nos erros causados, e 30% nos eventos adversos (ZAMBON, 2018b).

2.3 PRONTUÁRIO DO PACIENTE

O prontuário médico é um documento único, padronizado e altamente sigiloso, o qual contém informações, imagens e sinais pertinentes ao paciente. Possui como principal objetivo guardar essas informações para facilitar a assistência ao paciente e a comunicação entre membros da equipe envolvida no atendimento.

Mesmo que o prontuário sirva como ferramenta de trabalho para o médico e demais profissionais da saúde, esse documento é de propriedade do paciente, cabendo à instituição responsável manter sua segurança e seu adequado armazenamento. Com o avanço tecnológico a tendência é que o paciente possa ter acesso ao seu prontuário a qualquer momento, estando disponível em diversos atendimentos, nos quais novas informações também podem ser acrescentadas (FARINA, 1999).

O prontuário médico deve ser elaborado somente pelo profissional de saúde responsável pelo atendimento do paciente. Deve conter alguns itens obrigatórios, como: identificação do paciente, sinais vitais, diagnósticos, anamnese, evolução diária com data e hora, resultado de exames e prescrição do paciente de forma correta, além da assinatura, carimbo e inscrição no conselho de classe do profissional responsável pelo preenchimento (ALMEIDA, 2010). Destes, alguns serão descritos com maiores detalhes a seguir.

2.3.1 Sinais Vitais

São sinais que auxiliam no diagnóstico inicial de um paciente, os quais podem acusar sintomas relacionados a diversas enfermidades, principalmente doenças infecciosas. São eles: temperatura, pressão arterial, pulso e respiração. Machado também destaca que é indispensável analisar os sinais vitais durante a evolução do quadro clínico do paciente (MACHADO, 1975).

2.3.2 Anamnese

É realizado um questionário ao paciente no momento em que o mesmo adentra em um atendimento clínico ou hospitalar. Esse questionário segue um

padrão definido em cada instituição de saúde, o qual tem como principal objetivo adequar o melhor tratamento ao paciente, por exemplo, buscando conhecer se ele possui alguma alergia a determinados medicamentos, se possui alguma doença, se está utilizando algum medicamento, entre outros (ALBUQUERQUE, 2015).

2.3.3 Evolução do Paciente

Documento anexado ao prontuário que descreve a evolução diária do paciente em ordem cronológica. Essa descrição pode ser realizada com auxílio do próprio paciente, dos familiares do mesmo, da equipe médica que acompanha o seu tratamento, e com a realização da análise dos sinais vitais.

Esse documento deve conter as identificações dos profissionais envolvidos, assim como deve descrever de forma clara os procedimentos ao qual o paciente foi ou será submetido, a fim de adequar o tratamento para sua evolução (CFM, 2002).

2.3.4 Prescrição Médica

A prescrição médica é composta por orientações específicas para um determinado paciente, as quais devem ser seguidas de forma rígida, pois fazem parte do tratamento e buscam a evolução no quadro clínico do mesmo.

As prescrições podem ser, entre outras, de caráter fisioterapêutico, educacional e principalmente medicamentosas. Segundo Guimarães (2014), esse documento deve ser identificado pelo médico responsável para ter validade.

2.3.5 Resultado de Exames

Os resultados de exames são necessários para apoiar decisões médicas, diagnosticar ou acompanhar a evolução clínica de um paciente. Esses podem ser de análises clínicas, imagem, entre outros. Os resultados de exames são emitidos e entregues após a alta do paciente, durante o atendimento em casos de internação, ou no prazo estipulado pelo laboratório contratado (GUIMARÃES, 2014).

Na UPA Santa Paula os resultados de exames, independentemente do tipo, não são entregues diretamente ao paciente. Contudo, como o prontuário é um direito

do paciente e os resultados de exames o compõe, é concedido ao paciente o direito de protocolar uma solicitação na Prefeitura Municipal de Ponta Grossa ou retirar o resultado de exames laboratoriais em uma Unidade Básica de Saúde (UPA SANTA PAULA, 2016).

2.3.6 Prontuário Eletrônico do Paciente

O prontuário, quando realizado de grafia manuscrita e armazenado em papel, pode trazer diversas dificuldades nos procedimentos de saúde, sendo problemas de ilegibilidade, segurança, mobilidade, integração e incompletude de informações, os quais podem ser solucionados com a utilização da tecnologia disponível atualmente: o Prontuário Eletrônico do Paciente (PEP).

O PEP é uma das principais ferramentas em Tecnologia da Informação e Comunicação em Saúde (TICS), que consiste em um registro eletrônico disponibilizado aos profissionais de saúde e pacientes. Esse pode ser acessado em diferentes locais de maneira atualizada, legível e exata, o que contribui na continuidade de tratamentos devido sua facilidade de compartilhamento, e assim evita a redundância de informações (GONÇALVES et al., 2013).

O PEP pode interagir com outros sistemas, auxiliando no apoio a decisão, emitindo alertas, buscando diminuir erros e aumentando a segurança do paciente, além de possibilitar a redução de custos ao ambiente hospitalar em que se encontra implantado (CFM, 2012).

2.4 SISTEMA DE GESTÃO HOSPITALAR

Ferramentas que auxiliam na excelência da gestão hospitalar são fundamentais para assegurar o direito à saúde de cada indivíduo. A implementação adequada de um sistema ERP voltado para a área da saúde possibilita: maior confidencialidade dos dados do paciente, redução de erros na conduta médica, maior segurança à saúde do paciente, gerenciamento de processos assistenciais e administrativos, entre outros (PEREIRA et al., 2012).

2.4.1 Tasy EMR

O Philips Tasy EMR é uma solução em saúde que integra todas as áreas de uma instituição, visando o cuidado ao paciente e a otimização de processos. Atende diversas realidades de negócio, como hospitais, clínicas, bancos de sangue e *home care*, possibilitando eficiência nas atividades administrativas, financeiras, assistenciais e operacionais (PHILIPS, 2019). Esse foi o primeiro *software* de Prontuário Eletrônico do Paciente do Brasil a possuir a certificação emitida pela SBIS (Sociedade Brasileira de Informática em Saúde) em sua nova versão de 2016, assegurando a privacidade, confidencialidade e a integridade das informações de saúde, o que garante a segurança do paciente (CARRASCO, 2017).

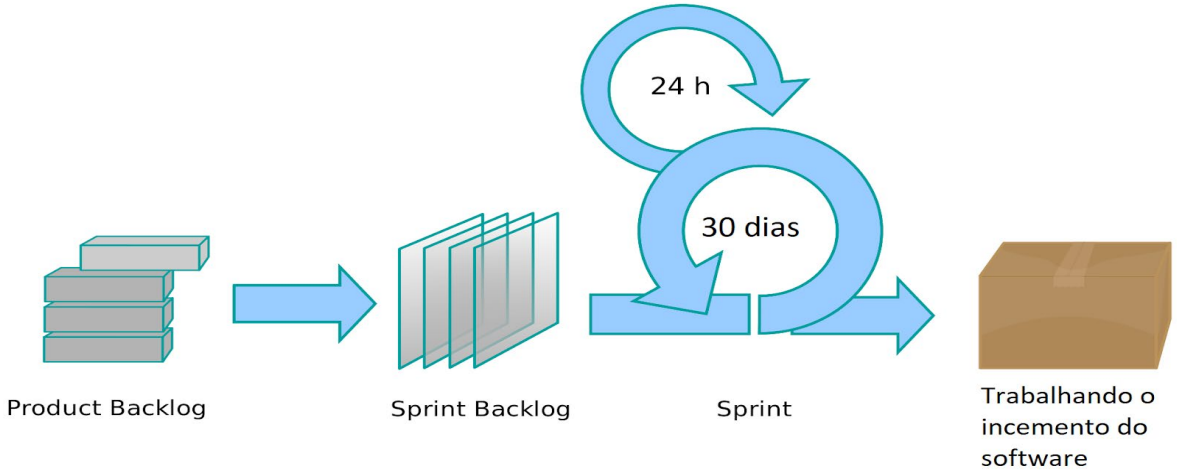
2.5 SCRUM

O SCRUM tem como objetivo estabelecer regras e gerenciamento para que um projeto seja desenvolvido com sucesso. Esse gerenciamento permite a todos os envolvidos saberem exatamente em qual patamar o projeto se encontra (ANDRADE et al., 2012).

O projeto deve ser planejado de forma clara e objetiva no seu início, buscando os detalhes e direcionamentos suficientes para que se possa iniciar o seu desenvolvimento. No SCRUM trabalha-se com o mínimo de documentação possível, priorizando o desenvolvimento de maneira ágil (OTÁVIO, 2015).

A metodologia ágil SCRUM representada na Figura 3, possibilita que o projeto possa ser desenvolvido e dividido em diversos ciclos de atividades (*Sprints*), com a realização de reuniões em cada ciclo com o intuito de alinhar o que está sendo desenvolvido e aperfeiçoar o processo continuamente, possibilitando melhorias na organização das tarefas e no controle de riscos por meio de uma abordagem iterativa e incremental (PAULA, 2016).

Figura 3 - Ciclo do SCRUM



Fonte: Adaptado de Paula (2016)

2.5.1 Product Backlog

É uma lista ordenada de todas atividades (módulos) a serem realizadas durante o desenvolvimento do projeto, sendo esse um fator dinâmico, pois os itens que inicialmente compõem o *Product Backlog* são criados de acordo com o primeiro levantamento de requisitos, sendo assim, a cada alteração nos mesmos o *Product Backlog* também é alterado, isso acontece também a medida que o projeto evolui. Essa lista é disposta de acordo com a importância de cada item, devendo os itens mais importantes possuírem um maior detalhamento (SCHWABER, 2013).

2.5.2 Sprint Planning Meeting

São reuniões que antecedem uma *Sprint*, realizadas juntamente com o cliente buscando selecionar alguns itens do *Product Backlog* e planejar como os mesmos serão desenvolvidos (ANDRADE et al., 2012). Em alguns casos nessa etapa também é discutido o conceito de *time-box*, ou seja, é estipulado um tempo para a realização de determinadas atividades, o qual não deve ultrapassar 8 horas (OTÁVIO, 2015).

2.5.3 Sprint Backlog

O *Sprint Backlog* consiste em um conjunto de itens selecionados do *Product Backlog* resultantes do *Sprint Planning Meeting* a serem desenvolvidos naquele *Sprint*. O resultado dessa etapa deve estar apto a ser apresentado ao cliente (FERREIRA et al., 2012).

2.5.4 Daily Scrum

São reuniões diárias, buscando discutir e selecionar os trabalhos a serem realizados naquele dia. Nessa reunião também é realizado um *feedback* do dia anterior, analisando e identificando impedimentos para uma melhora no decorrer do desenvolvimento (PAULA, 2016). Essas reuniões, segundo Andrade (2012), devem ser realizadas de forma rápida e objetiva, por isso elas são realizadas com os participantes em pé, e devem ter uma duração de no máximo 15 minutos.

2.5.5 Sprint Review Meeting

É uma reunião informal que busca analisar os resultados daquele *Sprint*, verifica se os objetivos propostos pelo *Sprint Backlog* foram atingidos e, por fim, são realizados incrementos e modificações no *Product Backlog* caso necessário (OTÁVIO, 2015).

2.6 GIT-FLOW

O git-flow[7] é um método utilizado para realizar o controle de versão do projeto e para uma melhor organização no desenvolvimento do sistema, principalmente quando está se desenvolvendo em equipe. Esta tecnologia se trata de um conjunto de extensões, provendo operações de alto nível para repositórios e para a organização de *branches* utilizando o modelo *Git branching* proposto por

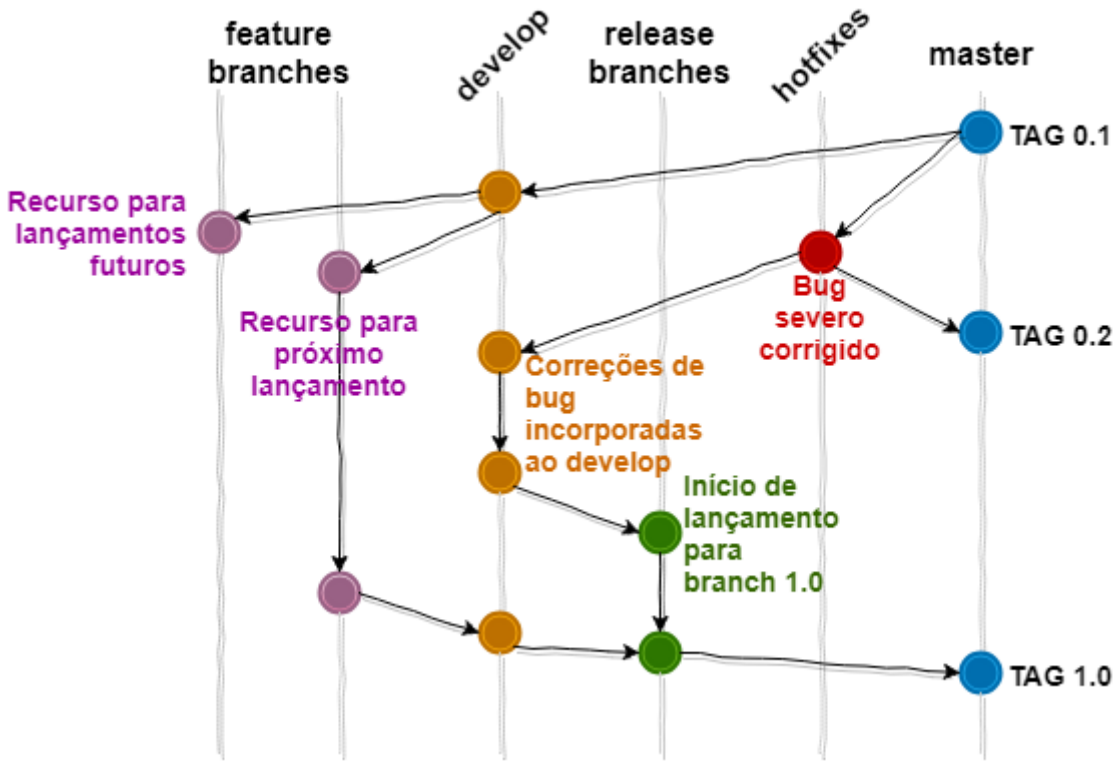
[7] https://danielkummer.github.io/git-flow-cheatsheet/index.pt_BR.html

Driessen (2010). As *branches* podem ser entendidas como versões distintas e identificadas do mesmo código-fonte, que podem servir para segurança, *backup* ou controle do desenvolvimento de novas funcionalidades.

Segundo Panzolini (2018), o modelo não impõe uma regra a ser seguida, mas sim diretrizes, as quais podem ser adaptadas de acordo com as necessidades de uso.

Na Figura 4 destacam-se as etapas do desenvolvimento ou ramificações (*branches*) utilizadas no modelo git-flow.

Figura 4 - Exemplo de desenvolvimento utilizando o git-flow



Fonte: Adaptado de Driessen (2010)

2.6.1 Branches Principais

- ***Branch Master*:** Consiste na ramificação principal cujo código fonte sempre reflete um estado pronto para produção, todo código finalizado em algum momento do desenvolvimento é juntado a esse *branch*.
- ***Branch Develop*:** Após a conclusão de uma *feature*, ela é juntada a *branch develop* e, posteriormente, realizados testes. Quando essa *branch*

fica estável, ou seja, quando não existem mais *features* para serem anexados, ela passa por mais um período de testes e, se necessário, a realização de algumas implementações, e por fim se junta a *branch master*, criando uma nova *tag* (GOMES, 2016).

2.6.2 Branches de Apoio

- ***Feature branches*:** inicia o desenvolvimento de uma nova funcionalidade criando uma cópia do *branch develop*. Quando finalizada, a funcionalidade é mesclada ao *branch develop* e, em seguida, removida.
- ***Release branches*:** suporta a preparação de uma nova versão de produção criando uma cópia do *branch develop*. O momento mais adequado para a criação de um *release branch* é quando o *develop* reflete o estado desejado para o lançamento de uma nova versão. Quando este *release* estiver pronto para ser lançado como uma versão definitiva, é feito sua mesclagem com o *master* e, posteriormente, em *develop*, sendo o primeiro para disponibilizar a nova versão definitiva e o segundo para disponibilizar as alterações feitas para possíveis versões futuras a serem desenvolvidas.
- ***Hotfix branches*:** Surgem da necessidade de ação imediata e urgente relacionada a um problema ou erro na atual versão de produção (*master*). Quando finalizada, a correção precisa ser mesclada novamente ao *master* e, posteriormente, em *develop*, pelos mesmos motivos que no estágio de conclusão de um *release branch*.

2.7 VUE.JS

Vue[8] é um *framework* JavaScript progressivo, capaz de construir interfaces de usuário com reatividade no DOM (*Document Object Model*)[9]. Seus modelos de

[8] https://br.vuejs.org/v2/guide/
[9] Interface que representa como os documentos HTML e XML são renderizados pelo *browser* (MALDONADO, 2018).

dados são objetos JavaScript puros que, quando modificados, atualizam a interface visual, o que torna seu sistema de reatividade não obstrusivo[10], ou seja, não é necessário a adição de atributos ou informações extras na marcação. O termo dado a reatividade dos elementos na interface é o *two-way data binding*, fazendo com que toda alteração que aconteça no DOM reflita no JavaScript e vice-versa (VUE.JS, Guia 2.x, 2019).

O *framework* Vue também proporciona que uma aplicação *web* completa seja gerenciada em uma única página, ou seja, uma *Single Page Application* (SPA) possibilitando que informações sejam trocadas e atualizadas na interface sem a necessidade de recarregamento (*reload*) da página. Isso é possível fazendo uso de alguns *plugins* como o Vue-Router[11], Vue Resource[12] e o Vuex[13] (ARAÚJO, 2016).

As características desse framework permitem ao sistema desenvolvido nesse trabalho, apresentar dados ao cliente em tempo real, sem a necessidade de atualização da página, evitando sobrecarga.

2.8 LARAVEL

Laravel é um *framework web* PHP *full stack*[14], livre e de código aberto, criado por Taylor Otwell e destinado ao desenvolvimento de aplicações *web* seguindo o padrão arquitetural *model-view-controller* (MVC)[15]. Este *framework* fornece autenticação, roteamento, gerenciador de sessão, *cache*, gerenciamento de dependências de classes através da ferramenta *container* IoC, a qual permite que dependências sejam “injetadas” em tempo de execução quando necessário, e diversos outros componentes. Também provê ferramentas de migração de banco de dados e suporte a testes, possibilitando aos desenvolvedores a capacidade de construir aplicações complexas de forma ágil, segura e organizada (LARAVEL, *Documentation v*. 5.8, 2019). As subseções abaixo discutem alguns aspectos desse *framework*.

---

[10] https://blog.caelum.com.br/boas-praticas-com-javascript-e-jquery-codigo-nao-obstrusivo/
[11] https://router.vuejs.org/
[12] https://github.com/pagekit/vue-resource
[13] https://vuex.vuejs.org/
[14] *Framework* que possuí diversas ferramentas capaz de desenvolver servidor e cliente (MARQUES, 2017).
[15] https://www.devmedia.com.br/introducao-ao-padrao-mvc/29308

2.8.1 Laravel Websockets

Quando é necessário obter dados de um servidor *web*, o cliente (navegador) envia pedidos via protocolo HTTP e espera por alguma resposta, o problema surge no momento em que esses dados devem ser apresentados em tempo real. Algumas aplicações utilizam o AJAX (*Asynchronous JavaScript and XML*) para executar tal tarefa, porém o mesmo trabalha enviando diversas requisições ao servidor dentro de um determinado tempo, o que seria uma técnica inadequada devido ao baixo desempenho e o consumo de dados (trafego) mais alto.

Websockets é a tecnologia que trabalha com comunicação bidirecional, ou seja, após o cliente estabelecer uma conexão com o servidor *web*, o mesmo pode enviar requisições e aguardar por um evento do servidor que retorne a resposta desejada através do mesmo canal. Isso elimina a necessidade de enviar diversas requisições ao servidor, o que evita uma sobrecarga do sistema e torna a comunicação mais interativa (POCIOT, 2018).

O Laravel WebSockets é um pacote desenvolvido pela beyondcode[16] para as versões do Laravel 5.7 ou superiores, o qual permite realizar a criação de controladores *websockets* personalizados para as mais diversas necessidades, contando com uma ampla documentação (BEYONDCODE, 2019).

2.8.2 Artisan

É a interface de linha de comandos própria do Laravel, a qual vem incluída no *framework* e disponibiliza uma diversidade de comandos que auxiliam o desenvolvedor durante a criação de um projeto. Seus comandos possibilitam a configuração do ambiente da aplicação, criação de modelos (classes) para *migrations* onde está o esquema para o banco de dados da aplicação desenvolvida, facilitando sua alteração e compartilhamento, *controllers* onde são armazenadas as lógicas de manipulação de solicitação, *models* que referenciam um objeto do banco de dados na classe, autenticação, entre outras.

[16] https://beyondco.de/

Além da vasta gama de comandos oferecidos pelo Artisan, podem ser criados comandos personalizados de acordo com as necessidades de cada projeto (LARAVEL, *Documentation v*. 5.8, 2019).

2.8.3 Template Blade

Em projetos *web*, comumente o conteúdo HTML deve ser repetido várias vezes, como o cabeçalho e rodapé das páginas. Isso traz alguns problemas como a repetição de código e a dificuldade na manutenção, pois quando um elemento fixo, como o cabeçalho, é alterado, é necessário modificar todas as outras páginas que contiverem o mesmo cabeçalho, a fim de manter a padronização.

Na tentativa de solucionar esse problema, a maioria dos *frameworks* possui sistemas para gerenciamento de interfaces que permitem a criação de *templates*, ou seja, as partes fixas das interfaces que ficam contidas na camada de visão, onde seu conteúdo pode ser reutilizado em diversas partes do projeto evitando então a repetição de código (SILVA, 2016).

No *framework* Laravel, o mecanismo de *template* responsável por gerenciar as interfaces do sistema é o Blade. Esse possui alguns diferenciais, como a utilização de códigos PHP simples em seus arquivos e a capacidade de compilar estes, armazenando-os em *cache*[17] (lugar temporário onde as páginas são armazenadas até que haja uma alteração) até que alguma alteração seja realizada, não gerando sobrecarga sobre a aplicação (LARAVEL, *Documentation v*. 5.8, 2019).

Na Figura 5 é apresentada uma implementação do *layout* do Blade, onde a *tag* “@yeld” representa o lugar onde a seção (@section) presente na página que herda esse layout será inserida.

[17] https://laravel.com/docs/5.7/blade

Figura 5 – Implementação do *layout* Blade

```
<!-- Stored in resources/views/layouts/app.blade.php -->

<html>
    <head>
        <title>App Name - @yield('title')</title>
    </head>
    <body>
        <div class="container">
            @yield('content')
        </div>
    </body>
</html>
```

Fonte: Adaptado de Laravel *Documentation* v. 5.8 (2019)

A Figura 6 apresenta a implementação de uma página “filha” que herda o layout acima. Essa possuí um conteúdo dentro de uma seção delimitada pelas tags “@section(‘content’)” e “@endsection”, a qual será inserida dentro da *tag* “@yeld”, presente no layout pai apresentado na Figura 5.

Figura 6 – Página filha herdando um *layout*

```
<!-- Stored in resources/views/child.blade.php -->

@extends('layouts.app')

@section('title', 'Page Title')

@section('content')
    <p>This is my body content.</p>
@endsection
```

Fonte: Adaptado de Laravel *Documentation* v. 5.8 (2019)

2.8.4 Eloquent ORM

ORM (*Object Relational Mapping*) é uma técnica utilizada pelo padrão de projeto *Active Record*, o qual consiste em mapear uma tabela do banco de dados a uma classe objeto (POLO, 2009). Cada tabela do banco de dados tem uma classe de modelo (*Models*) correspondente, utilizada para interagir com a respectiva tabela. Além dessa técnica, o Eloquent ORM, incluso no Laravel possibilita a migração das tabelas do banco para os *Models*, utilizando comandos do Artisan (LARAVEL, *Documentation v*. 5.8, 2019).

2.8.5 Task Scheduling

Comumente, cada tarefa (ex: envio de email, consulta ao banco de dados) a ser agendada é configurada separadamente para ser executada no servidor por um "agendador de tarefas" nativo do sistema operacional. Dessa forma, o cronograma de tarefas não fica centralizado na aplicação e é necessário SSH[18] no servidor para acessos adicionais de tarefas agendadas.

O agendador de tarefas do Laravel permite que diversas tarefas sejam gerenciadas pelo próprio *framework*, definindo de forma fluente e expressiva o cronograma de tarefas a serem executadas (LARAVEL, *Documentation v*. 5.8, 2019). Para isso, é necessário a configuração de uma única entrada *Cron*[19] (ferramenta do sistema operacional que permite agendar a execução de uma tarefa) no servidor, o qual irá chamar o agendador do Laravel a cada minuto para executar as tarefas vencidas e estimar a execução das pendentes.

2.8.6 Jobs

São classes onde são armazenadas uma ou mais tarefas a serem executadas, como o envio de um *email* ou uma consulta no banco de dados.

---

[18] Protocolo que permite ao usuário realizar modificações em um servidor de forma remota (SILVA, 2010)

[19] https://www.infowester.com/linuxcron.php

Geralmente essas classes são processadas pelas *Queues* (filas) do Laravel, que permitem enfileirar ou, até mesmo, adiar a execução de uma *job.*

Os *jobs* permanecem em um controlador de trabalhos em estado de espera, até que chegue o momento certo para que sejam executados. Esses métodos podem ser criados através de um comando Artisan e, geralmente, são utilizados para manipulação e proteção de dados (LARAVEL, *Documentation v*. 5.8, 2019). Contudo, neste projeto, os *jobs* são executados por meio do Laravel Task Scheduling sem a necessidade de enfileiramento.

2.8.7 Events

Um evento, basicamente, é uma ação que acontece durante a execução de uma aplicação e a partir disso outras tarefas podem ser executadas, como por exemplo, identificar quando um usuário acessa uma determinada rota do sistema. Sendo assim, quando for identificado o evento “acessar a rota x” uma tarefa “y” será executada (FERREIRA, 2018).

Os *Events* do Laravel, podem criar (registrar) os próprios eventos (localizados em “app/Events”) para executar tarefas pré-definidas. Também são capazes de dissociar vários aspectos da aplicação, sendo que um único evento pode conter vários *listeners* (ouvintes) independentes. Os *listeners* são métodos manipuladores de eventos que “escutam” um determinado evento para que possam realizar uma ação de forma automática (localizados em “app/Listeners”). Além disso, o *Events* fornece uma implementação do “*observer*”[20], uma classe que permite agrupar diversos *listeners*.

Para isso, é necessário definir os assinantes (*subscribe*) de eventos, os quais identificam quando ocorre um determinado evento, como por exemplo, “realizar *login*”. Esses devem possuir um método, e quando ocorrer o evento, um ou mais *listeners* serão acionados. Na Figura 7 é apresentado um exemplo de uma classe de assinantes de eventos (LARAVEL, *Documentation v*. 5.8, 2019).

[20] https://laravel.com/docs/5.8/eloquent#observers

Figura 7 - Classe de assinantes

```php
<?php

namespace App\Listeners;

class UserEventSubscriber
{
    public function handleUserLogin($event) {}

    public function handleUserLogout($event) {}

    /**
     * @param  \Illuminate\Events\Dispatcher  $events
     */
    public function subscribe($events)
    {
        $events->listen(
            'Illuminate\Auth\Events\Login',
            'App\Listeners\UserEventSubscriber@handleUserLogin'
        );

        $events->listen(
            'Illuminate\Auth\Events\Logout',
            'App\Listeners\UserEventSubscriber@handleUserLogout'
        );
    }
}
```

Fonte: Adaptado de Laravel *Documentation* v. 5.8 (2019)

2.8.8 Laravel Homestead

O Laravel Homestead utiliza um pacote pré-configurado do Vagrant[21], que provê uma maneira simples de gerenciar e preparar Máquinas Virtuais. Dessa forma, não é necessário instalar o PHP, servidor *web* e nenhum outro *software* na máquina

[21] https://www.vagrantup.com/intro/index.html

local, sendo que o Laravel Homestead cria e configura todo o ambiente necessário para implementação da aplicação com o sistema operacional Linux Ubuntu 18.04 (LARAVEL, *Documentation v*. 5.8, 2019).

2.9 BANCO DE DADOS

Segundo Date (2004), na computação, banco de dados é um sistema capaz de armazenar e realizar a manutenção de registros eletrônicos. De forma geral, são armazenadas informações pelos usuários para que posteriormente possam ser pesquisadas, atualizadas, filtradas, entre outras operações necessárias para auxiliar o processo em que o indivíduo ou organização necessite consumir esses dados.

Nesse projeto são utilizados dois SGBD's (Sistema Gerenciador de Banco de Dados) que interagem de maneiras distintas com o sistema, o MariaDB[22] e o Oracle Database[23].

2.9.1 MariaDB

O MariaDB é um dos SGBD's relacionais mais populares do mundo, desenvolvido em C, C++, Perl e Bash, sendo um sistema de código aberto e tendo como linguagem de manipulação de dados padrão o SQL (MARIADB, 2019). Também possui diversos *plugins* e ferramentas auxiliares que o torna um sistema adaptável a diversas situações.

Nesse projeto, foi utilizado o MariaDB na versão 10.3.13, o qual é responsável pelo armazenamento de dados de toda a aplicação, interagindo diretamente com a API. O cliente nunca se comunica diretamente com a base de dados real do sistema Tasy EMR, sendo o MariaDB a base de dados intermediária entre estes, que armazena somente os dados pertinentes à aplicação.

[22] https://mariadb.org/about/
[23] https://docs.oracle.com/cd/E17781_01/index.htm

2.9.2 Oracle Database

O Oracle Database é um dos SGBD's pioneiros a utilizar o modelo relacional, sendo desenvolvido em Assembly, C, C++ e Java, por Larry Ellison, Bob Miner e Ed Oates. Sendo um dos mais populares do mundo, este SGBD tem alta capacidade de desempenho e escalabilidade, e tornou-se o primeiro SGBD a ser comercializado na história pela empresa Oracle Corporation na década de 70 (LEGATTI, 2007). Atualmente encontra-se na versão 19c, dando suporte às linguagens de manipulação de dados SQL, JSON, XML, e linguagens procedurais como PL/SQL, Java, C e C ++ (ORACLE, 2019).

3 ARQUITETURA DO SOFTWARE

O sistema desenvolvido propõe uma interface para disponibilizar dados de atendimento aos pacientes, os quais podem ser visualizados em dispositivos móveis e estações de trabalho. Esses dados são obtidos por meio de uma consulta no banco de dados do sistema Tasy EMR, armazenados no banco de dados da aplicação desenvolvida e, posteriormente, apresentados ao usuário. Para tornar a aplicação mais interativa, foram configurados mecanismos de agendamento de tarefas, eventos e *websockets*, os quais funcionam de forma sincronizada para entregar dados em tempo real e de forma dinâmica aos usuários.

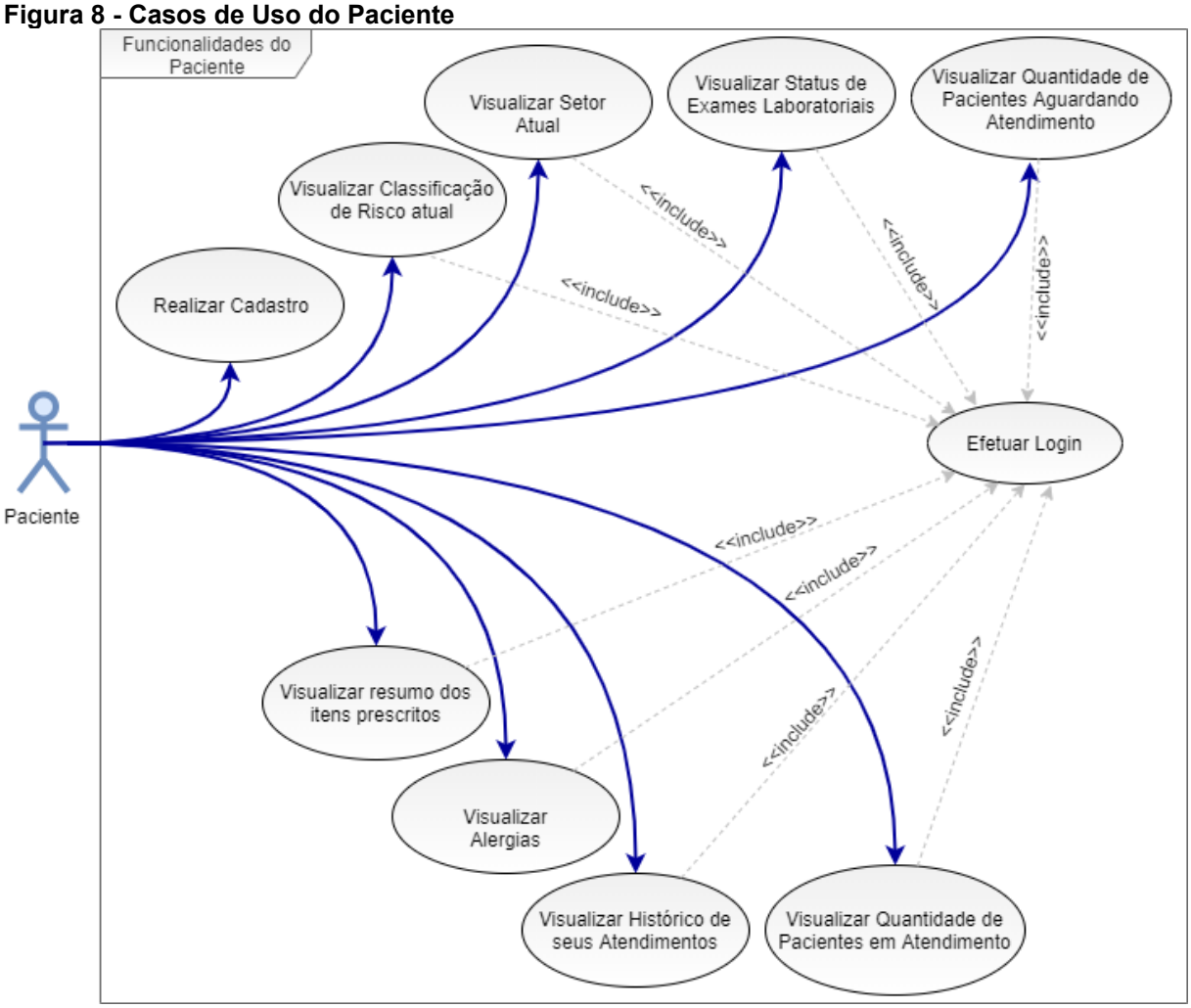
Todos os recursos e tecnologias utilizadas para o melhor funcionamento do sistema estão detalhados nos tópicos a seguir.

3.1 APLICAÇÃO CLIENTE

O projeto foi realizado contemplando tecnologias atuais, as quais integram com o Laravel e provêm uma experiência dinâmica com o usuário final. Dentre as tecnologias utilizadas no cenário do cliente, destacam-se o Vue.js, como ferramenta de interação visual, e o Laravel Websockets, como ferramenta de interação direta com o servidor.

O sistema pode ser acessado por pacientes, os quais, por questões de segurança, somente podem acessar o sistema através de um mecanismo de autenticação (*login*) para visualização de seus dados de atendimento. Além disso o sistema provê informações públicas de quantidade de pacientes em espera e em atendimento médico. Na Figura 8 temos a representação das funcionalidades do cliente utilizando um diagrama de Casos de Uso (diagrama que apresenta as principais funcionalidades do sistema).

Figura 8 - Casos de Uso do Paciente



Fonte: Autoria Própria (2019)

3.1.1 Módulo do Paciente

Por questões de segurança, o paciente somente pode se cadastrar na aplicação se o seu CPF e data de nascimento existirem no seu cadastro no sistema Tasy EMR, devendo também ser informado o número do Cartão Nacional de Saúde (CNS), nome completo, e-mail e senha. Na tela de *login*, o paciente deverá informar seu CPF e a senha pessoal cadastrada.

O paciente poderá interagir com seu fluxo de atendimento em tempo real, obtendo informações como sua linha do tempo de atendimento, classificação de risco, tempo médio de espera, *status* de exames laboratoriais, quantidade de itens prescritos e o histórico de atendimentos. Não estando em atendimento, o paciente

terá acesso somente ao seu histórico atualizado de atendimentos no referido estabelecimento de saúde.

3.2 APLICAÇÃO SERVIDOR

Quando um cliente (usuário) efetuar *login* no sistema, será estabelecida uma conexão via *websocket* entre o cliente e o servidor, de modo a possibilitar o envio de dados para o cliente sem que haja a necessidade de uma requisição por meio de tecnologias presentes no *framework* Laravel, mantendo-o sempre atualizado.

3.2.1 Módulo do Paciente

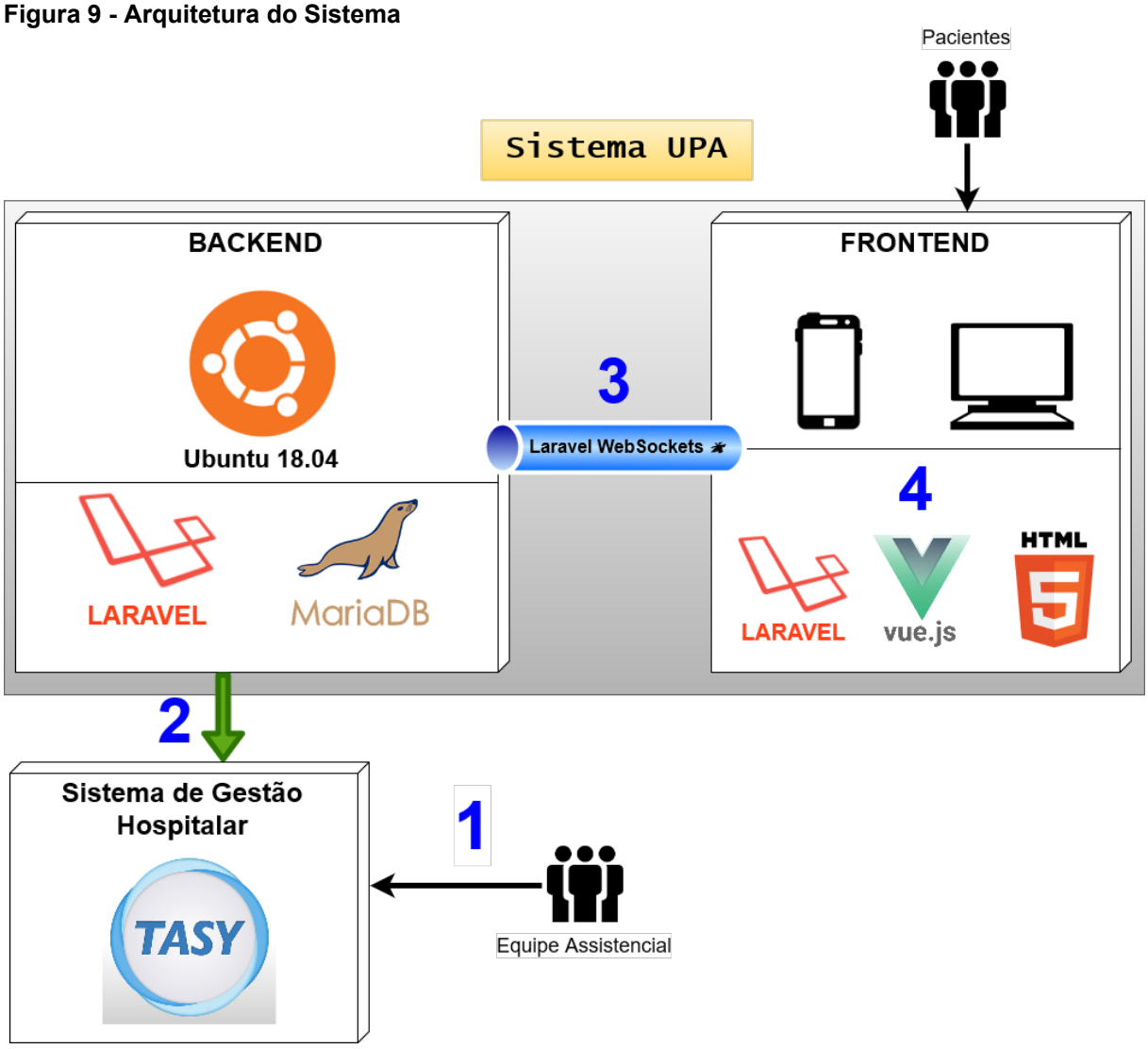
O sistema sempre enviará para o paciente os dados mais atualizados possíveis, incluindo seu histórico de atendimento no referido estabelecimento de saúde. Quando o paciente estiver em atendimento, no mesmo momento em que está conectado ao sistema, será realizada uma busca por atualizações no banco de dados do Tasy EMR (Oracle Database), referentes ao atendimento, por meio da *task scheduling* do Laravel que acionará um *job*. Se novas atualizações forem encontradas, o *job* irá disparar um evento (*event*) que atualiza a base de dados do sistema (MariaDB) e apresenta ao paciente os dados atuais do atendimento via *websocket*.

3.3 ESCOPO GERAL

O sistema operacional utilizado para a aplicação foi o Ubuntu na versão 18.04 LTS Server, por ser o sistema padrão do Laravel Homestead. No *backend*, utilizou-se o *framework* PHP Laravel na versão 5.8 e o banco de dados MariaDB. Já no *frontend*, além do Laravel, foi utilizado o framework Vue.js em conjunto com o HTML 5 e CSS para estilização e arquitetura do cliente.

Na Figura 9 está representado o funcionamento do sistema, demonstrando de uma forma geral a interação entre os sistemas Tasy e o sistema UPA, cuja sequência de funcionamento está representada na legenda numérica da figura.

Figura 9 - Arquitetura do Sistema



Fonte: Autoria Própria (2019)

1. A equipe assistencial, como médicos, enfermeiros e farmacêuticos, utilizam o sistema Tasy como um meio eletrônico para realizar o registro dos procedimentos de saúde dos pacientes. A base de dados responsável por armazenar todas as informações deste sistema é o Oracle Database, na versão 11g.
2. O sistema UPA utiliza o banco de dados do sistema Tasy para coletar os dados de atendimento do paciente, armazená-los no banco de dados MariaDB e enviá-los aos clientes. O acesso ao banco de dados do Tasy é

realizado no momento do *login* do paciente para atualizar todo seu histórico no sistema UPA e também a cada minuto, se o paciente estiver em atendimento, para mantê-lo atualizado em “tempo real”.

3. O sistema UPA utiliza uma tecnologia de *websockets* para enviar os dados ao cliente sem a necessidade de receber uma requisição do mesmo, deixando a aplicação mais interativa e possibilitando tornar esta aplicação em tempo real. Os detalhes desta comunicação em tempo real serão apresentados nos tópicos a seguir.
4. O sistema UPA possui uma interface para disponibilizar dados de atendimento aos pacientes, os quais poderão ser visualizados em dispositivos móveis e estações de trabalho de forma dinâmica e reativa. Isso foi possível utilizando o framework Vue.js, que é capaz de alterar a disposição do DOM sem a necessidade de atualizar a página web. Também foram utilizadas tecnologias como o HTML 5 e o CSS para a estilização e arquitetura do cliente.

3.4 TESTES DO SISTEMA

Para a realização de testes que auxiliam na implementação do código, o *framework* Laravel (2019) disponibiliza ferramentas e métodos auxiliares, os quais realizam testes unitários e de recursos.

- **Testes Unitários:** são testes concentrados em porções menores do código, como um único método;
- **Testes de Recursos:** possibilitam testar uma parte maior do código, como uma interação de objetos e até mesmo requisições HTTP.

Após o término de cada módulo desenvolvido, foram realizados testes unitários para a validação dos mesmos e, posteriormente, a disponibilização em ambiente de produção. Após finalizadas as etapas de produção, foram aplicados testes experimentais para analisar o comportamento do sistema.

4 METODOLOGIA

O desenvolvimento desse projeto foi realizado seguindo a metodologia ágil SCRUM, com o auxílio da ferramenta Git-Flow para a organização do mesmo. Primeiramente foi realizado um levantamento de requisitos funcionais do *software* desenvolvido na UPA Santa Paula, onde o mesmo foi implementado com informações coletadas do sistema Tasy EMR. Com isso, foi realizada uma análise de viabilidade buscando refinar ou complementar os requisitos com o auxílio de diagramas.

Após a primeira etapa definida, foi realizado um levantamento de requisitos físicos, selecionando os materiais necessários para o início no desenvolvimento do sistema. Com todos os requisitos pré-estabelecidos deu-se início no desenvolvimento, onde foram realizadas reuniões semanais presenciais para discutir o andamento do projeto, realizar a documentação e estabelecer metas a serem desenvolvidas na semana seguinte.

4.1 AMBIENTE DE DESENVOLVIMENTO E IMPLANTAÇÃO

A aplicação foi desenvolvida na UPA Santa Paula, após aprovação da solicitação para pesquisa concedida pelo Núcleo de Educação Permanente (NEP) da Secretaria Municipal de Saúde de Ponta Grossa (SMS), e do Diretor Executivo da UPA Santa Paula, ver anexos A e B. O desenvolvimento do sistema dentro da unidade foi necessário para realizar a comunicação deste sistema com o banco de dados do sistema Tasy EMR para coleta de dados dos pacientes.

O desenvolvimento foi distribuído em dois ambientes, sendo o ambiente local e o ambiente online. No segundo caso foram utilizadas a ferramenta Trello, responsável por gerenciar as tarefas a serem desenvolvidas seguindo a metodologia ágil SCRUM, e a ferramenta GitHub, que armazena o principal repositório do projeto, onde ficam as versões de produção do código fonte.

No ambiente local, foram utilizadas as ferramentas Git-Flow, responsável por gerenciar as versões do sistema desenvolvido, o GitHub Desktop, responsável por exportar (realizar *commit*) as versões de produção do Git-Flow para o repositório

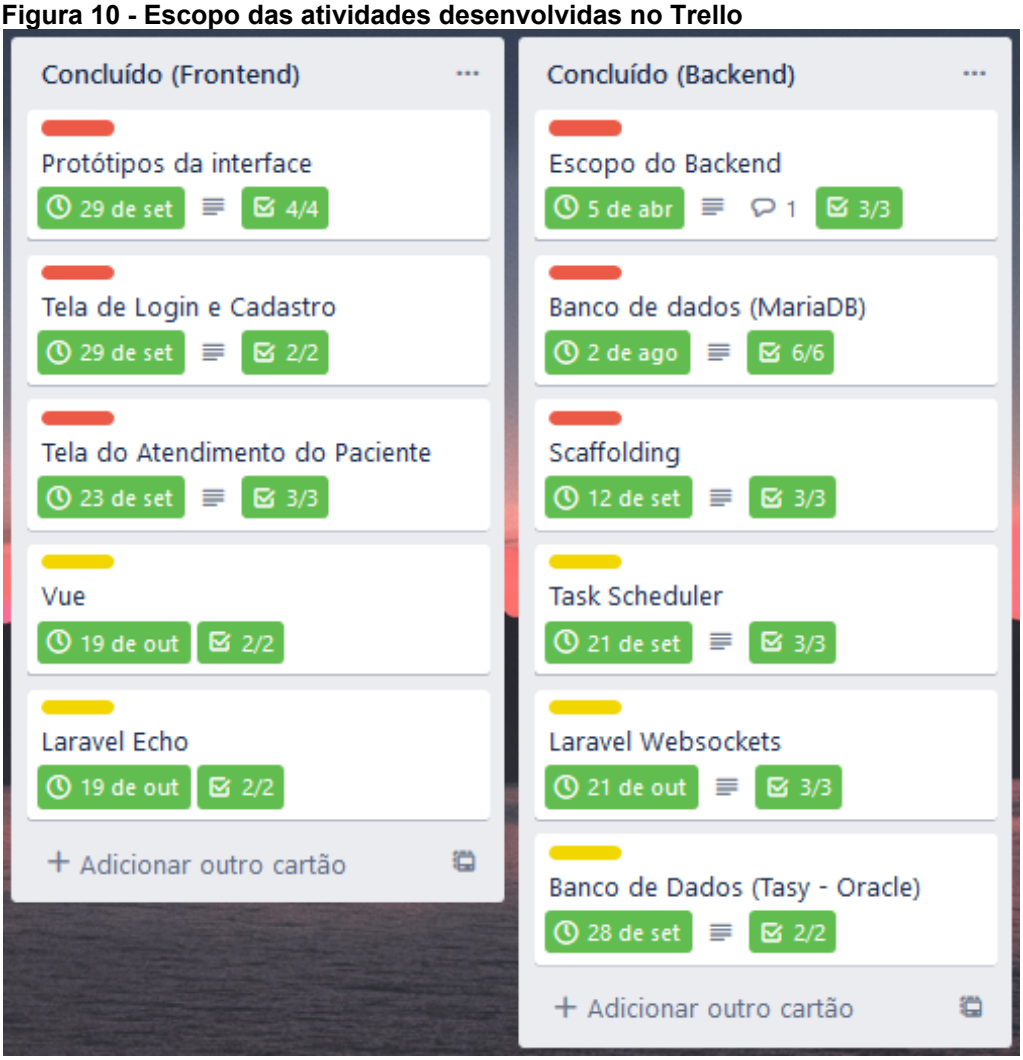
online do GitHub, e o *software* de máquina virtual VirtualBox, onde foi configurado o servidor Laravel Homestead incluindo o servidor *web* Nginx[24], por se tratar de um *software open source*, onde foram realizados os testes de implementação do sistema. Nesse ambiente, também foi utilizado a ferramenta Visual Studio Code (VSCode)[25] para auxiliar na escrita de códigos, a qual possui *plugins* visuais e de *auto complete*[26] (sugestões para completar a trechos de código fornecendo agilidade na prograrmação), que contribuem nesse processo.

4.2 DESENVOLVIMENTO

O desenvolvimento foi iniciado criando uma lista ordenada de tarefas a serem realizadas com o auxílio da ferramenta Trello[27], como mostra a Figura 10, a qual foi seguida durante todo o período de desenvolvimento desse sistema com base na metodologia SCRUM.

[24] https://www.nginx.com/
[25] https://code.visualstudio.com/
[26] https://docs.microsoft.com/en-us/archive/msdn-magazine/2008/launch/sql-code-completion-subversion-tools-agile-development
[27] https://trello.com/

Figura 10 - Escopo das atividades desenvolvidas no Trello



Fonte: Autoria Própria (2019)

Para versionamento do projeto foi utilizado o Git-Flow, o qual é manipulado com o auxílio da ferramenta Git Bash, onde foram criados os *branches* para o desenvolvimento das funcionalidades e posteriormente realizado o *deploy* para o repositório principal no Github.

Mesmo o Laravel Homestead disponibilizando diversos recursos necessários para o desenvolvimento de um sistema *web*, foi necessária a instalação do Oracle *Client* para configuração do driver OCI8 possibilitando a conexão com a base de dados do sistema Tasy. Também foi necessária a configuração de um comando CRON no Linux (Ubuntu) para executar o módulo de *Task Scheduler* do Laravel a cada minuto.

Sendo assim, a estrutura do sistema foi dividida em três partes principais, sendo elas a comunicação com a base de dados do sistema Tasy, a implementação de tecnologias que possibilitam a transmissão de dados em tempo real, e as

tecnologias usadas na interface do cliente. A seguir é discutida tal estruturação de forma detalhada.

4.2.1 Comunicação com o Sistema Tasy

Tasy é o sistema de gestão hospitalar utilizado pela UPA Santa Paula, local onde o *software* proposto nesse trabalho foi desenvolvido. Esse sistema é responsável por registrar e manter todas as informações de procedimentos de cuidado a saúde dos pacientes desde sua entrada até o momento de sua saída da unidade. Para isso, utiliza o SGBD Oracle na versão 11g, o qual é responsável por fornecer as informações necessárias para a API desenvolvida nesse projeto.

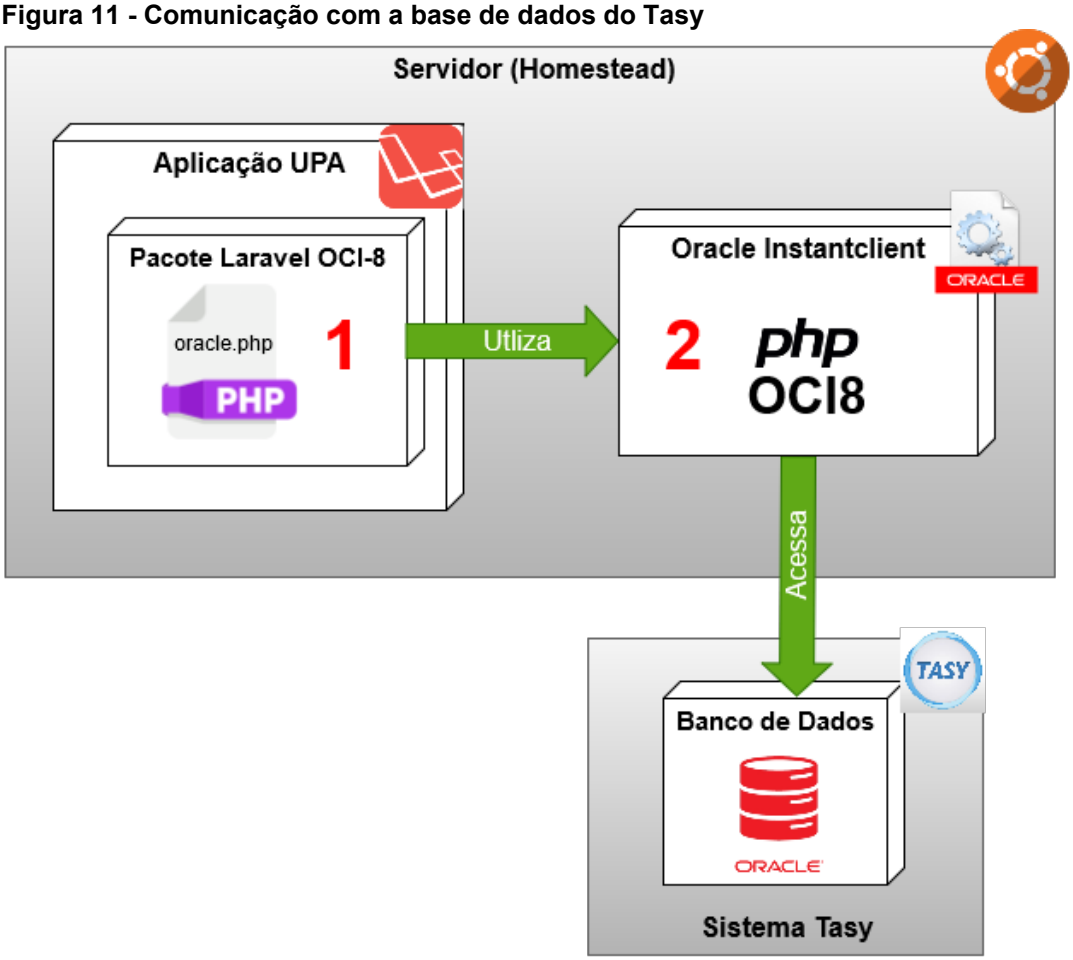
Para realizar a conexão com o banco de dados Oracle, foi utilizado o pacote “Laravel-OCI8”[28] na versão 5.8 e a instalação e configuração do “Oracle Instant Client”[29] na versão 12.2.0.1.0. Basicamente, o Laravel-OCI8 é uma extensão do Illuminate Database, utilizado pelo Laravel, que facilita a configuração do Oracle e faz uso da extensão OCI8 do PHP para comunicação com este banco de dados.Contudo, para que a extensão OCI8 do PHP consiga acessar um banco de dados da Oracle, é necessário a configuração do Oracle Instant Client.

A Figura 11 apresenta como este processo acontece, representado em forma sequencial pela legenda numérica.

[28] https://github.com/yajra/laravel-oci8

[29] https://www.oracle.com/technetwork/pt/articles/dsl/instalacao-do-php-instant-client-2106518-ptb.html

Figura 11 - Comunicação com a base de dados do Tasy



Fonte: Autoria Própria (2019)

1. Após instalação do pacote Laravel-OCI8, toda configuração de acesso ao banco de dados Oracle é realizada no arquivo “oracle.php” conforme mostra a Figura 12.
2. Com a instalação e configuração do Oracle Client no servidor, foi possível utilizar a extensão OCI8 do PHP e acessar o banco de dados da Oracle.

Figura 12 - Arquivo de configuração "oracle.php"

```php
return [
    'oracle' => [
        'driver'         => 'oracle',
        'host'           => env('DB_ORACLE_HOST', ''),
        'port'           => env('DB_ORACLE_PORT', '1521'),
        'database'       => env('DB_ORACLE_DATABASE', ''),
        'username'       => env('DB_ORACLE_USERNAME', ''),
        'password'       => env('DB_ORACLE_PASSWORD', ''),
        'charset'        => env('DB_CHARSET', 'AL32UTF8'),
        'prefix'         => env('DB_PREFIX', ''),
        'prefix_schema'  => env('DB_SCHEMA_PREFIX', ''),
        'edition'        => env('DB_EDITION', 'ora$base'),
        'server_version' => env('DB_SERVER_VERSION', '11g'),
    ],
];
```

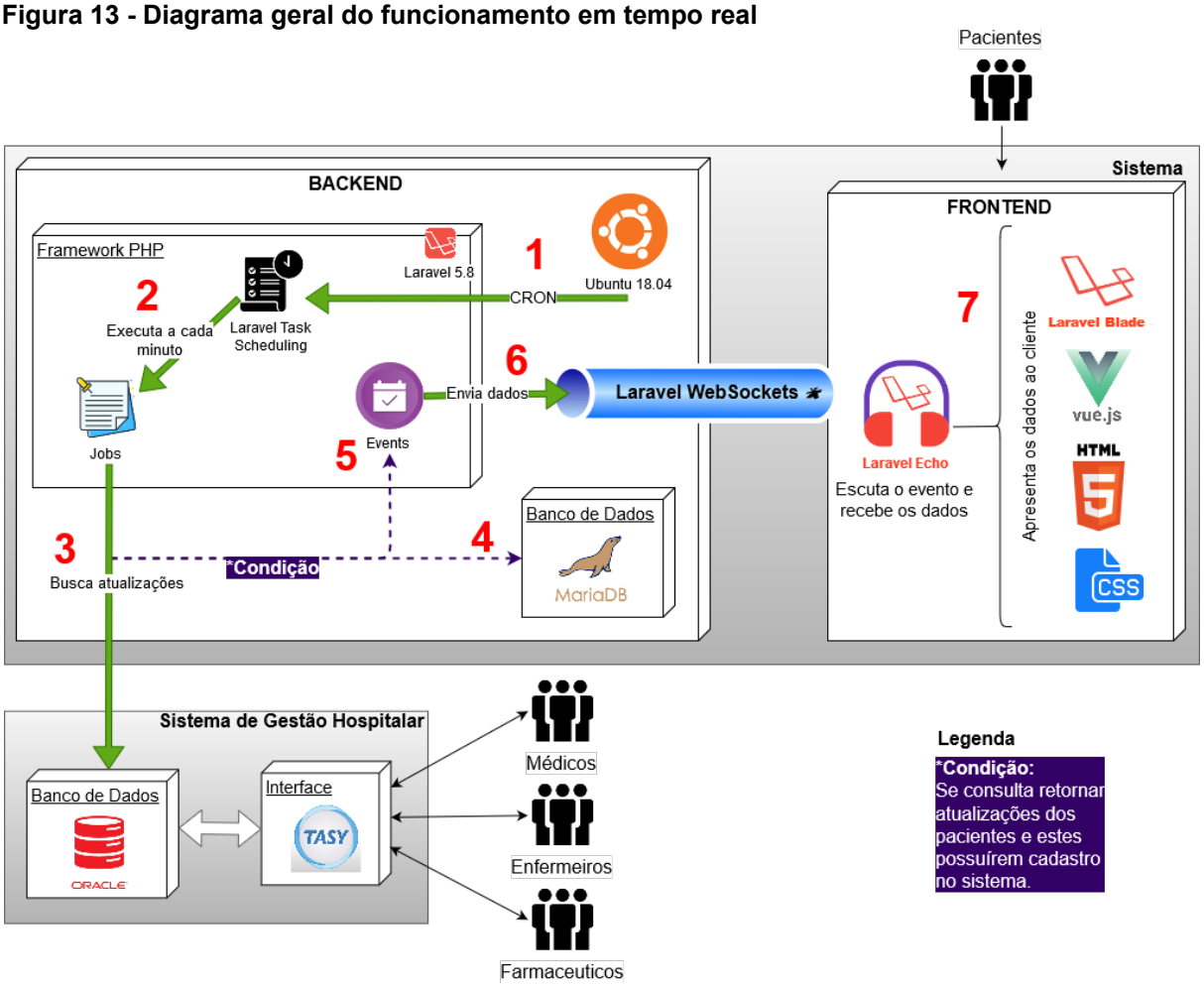
Fonte: Autoria Própria (2019)

Após a realização da configuração de conexão com o Oracle DB, o acesso aos dados do sistema Tasy estão habilitados.

4.2.2 Atualização Em Tempo Real

A Figura 13 apresenta na íntegra a sequência deste processo, e a Figura 14 apresenta de forma mais específica sua execução, ambas representadas em forma sequencial pela legenda numérica. Em seguida estão descritos os métodos como as tecnologias descritas foram utilizadas.

Figura 13 - Diagrama geral do funcionamento em tempo real

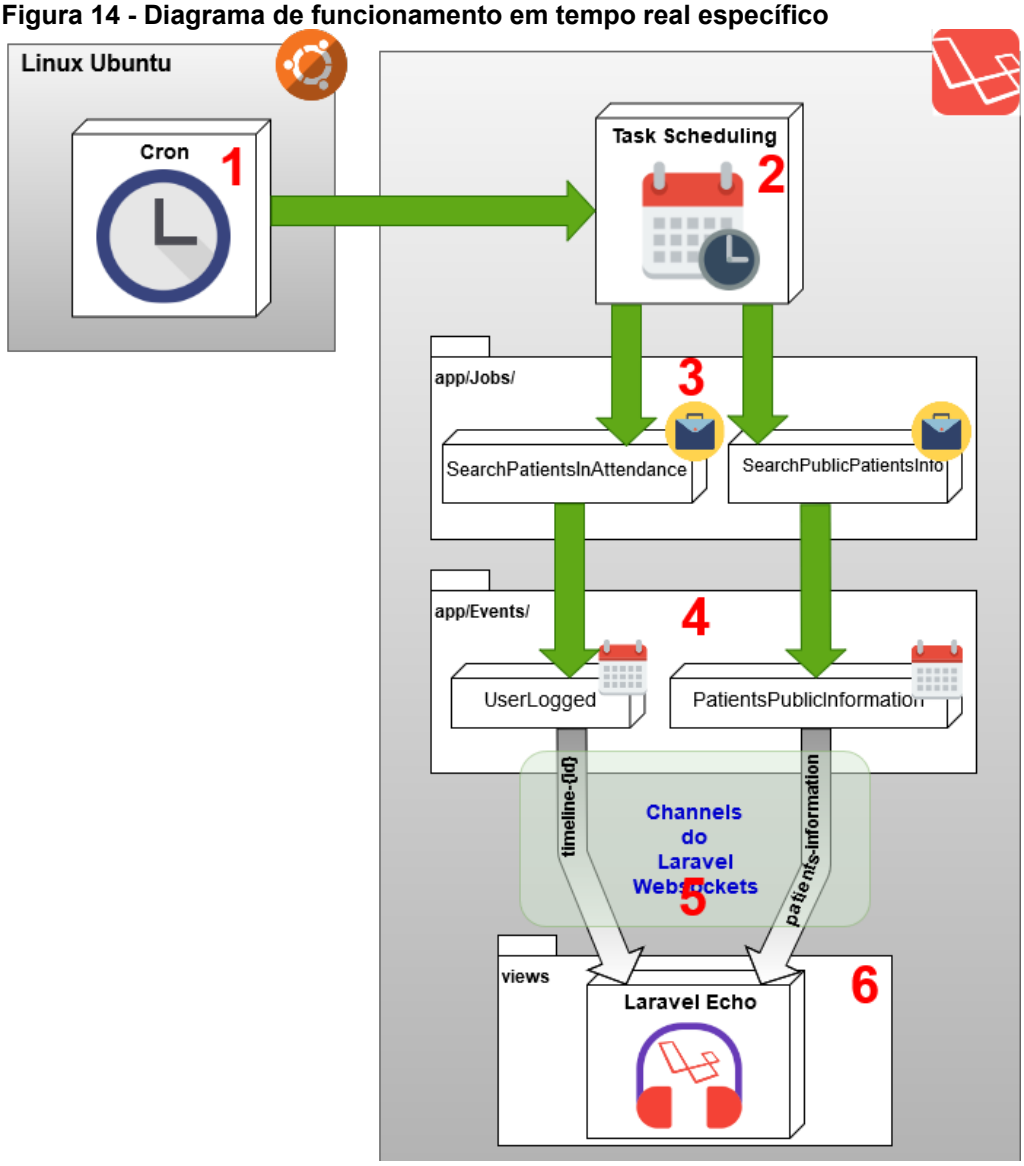


Fonte: Autoria Própria (2019)

1. O Cron, agendador de tarefas do servidor Ubuntu, aciona a Task Scheduling do Laravel a cada minuto, conforme configurado no arquivo “crontab”.
2. A Task Scheduling do Laravel verifica se há trabalhos (*jobs*) a serem executados, os quais também foram programados a cada minuto, fazendo com que sempre sejam executados a cada chamada à Task Scheduling.
3. Os *jobs* buscam por atualizações na base de dados do sistema Tasy e, se encontradas, as próximas etapas serão executadas.
4. Como novos dados foram encontrados, estes são inseridos ou atualizados no banco de dados da aplicação UPA, MariaDB, com o intuito de armazenamento do histórico e facilidade de acesso.
5. Estes mesmos dados são também enviados para seus respectivos eventos, os quais são responsáveis por organizá-los e enviá-los ao cliente.

6. O meio utilizado para este envio são os canais (*channels*) do Laravel Websockets, o qual assegura que somente os "ouvintes" destes canais receberão os dados.
7. No lado cliente, o recurso Laravel Echo[30] é o responsável por "inscrever" o cliente em um determinado canal, "escutar" os eventos que são acionados e receber os dados enviados por eles. Estes dados são apresentados ao cliente por meio de tecnologias como *framework* Vue.js, Laravel Blade, HTML 5 e CSS.

Figura 14 - Diagrama de funcionamento em tempo real específico



Fonte: Autoria Própria (2019)

[30] https://laravel.com/docs/5.8/broadcasting

1. Como explicado no diagrama anterior, o agendador de tarefas do servidor é executado a cada minuto, acionando o Task Scheduling do Laravel.
2. O Task Scheduling do Laravel aciona, também a cada minuto, dois *jobs* a serem executados: SearchPatientsInAttendance e SearchPublicPatientsInfo.
3. Os *jobs* SearchPatientsInAttendance e SearchPublicPatientsInfo realizam suas respectivas tarefas para coleta de dados e acionam, respectivamente, os eventos UserLogged e o PatientsPublicInformation, enviando os dados que foram coletados.
4. Ambos os eventos recebem os dados que foram enviados e os envia ao cliente por meio de canais de *websockets*. Contudo, por se tratar de dados confidenciais, o evento UserLogged envia os dados por meio de um canal privado de websocket. O evento PatientsPublicInformation gerencia dados públicos, podendo estes serem enviados por meio de um canal público: “patients-information”.
5. O nome do canal de *websocket* privado é único para cada usuário, substituindo a variável “{id}” pelo identificador único do usuário a que se destinam os dados. Já o canal público não tem variação no nome, fazendo com que os dados sejam enviados sempre para o mesmo canal.
6. Para que o cliente possa receber os dados enviados pelos canais de *websockets*, foi configurado o recurso Laravel Echo, o qual realiza a inscrição nos canais e escuta a execução dos eventos para receber os dados que são enviados.

4.2.2.1Task Scheduling

A tarefa que executa o *Task Scheduling* do Laravel[31] a cada minuto foi inserida no arquivo de configuração do Cron, com a seguinte linha de comando:

```
* * * * * cd /code/UPA24H && php artisan schedule:run >> /dev/null 2>&1
```

O arquivo de configuração da *Task Scheduling* do Laravel, invocado pelo Cron do Ubuntu, está localizado no diretório “App/Console/Kernel.php”, onde foram

[31] https://laravel.com/docs/5.8/scheduling

definidos os intervalos de tempo e os trabalhos a serem realizados (*jobs*) como mostra a Figura 15. Quando chamada, a *Schedule* do Laravel irá verificar se existe alguma tarefa a ser realizada.

Figura 15 - Arquivo de configuração da *Task Scheduling*

```
/**
 * Define the application's command schedule.
 *
 * @param  \Illuminate\Console\Scheduling\Schedule  $schedule
 * @return void
 */
protected function schedule(Schedule $schedule)
{
    $schedule->job(new SearchPatientsInAttendance)->everyMinute();
    $schedule->job(new SearchPuplicPatientsInfo)->everyMinute();
}
```

Fonte: Autoria Própria (2019)

4.2.2.2Jobs

Os *Jobs* estão localizados no diretório “app/Jobs”, e foram criados através dos seguintes comandos Artisan:

```
php artisan make:job SearchPatientsInAttendance
php artisan make:job SearchPublicPatientsInfo
```

O *job* "*SearchPatientsInAttendance*", apresentado na Figura 16, é responsável por realizar uma consulta na base de dados do Tasy, buscando os pacientes que se encontram em atendimento no momento de sua execução. Após a obtenção desses dados, é realizada uma análise comparativa para verificar se os pacientes em atendimento possuem cadastro na base de dados do sistema desenvolvido. Em caso afirmativo, os dados do atendimento atual do paciente são atualizados no banco de dados MariaDB e, posteriormente, é disparado o evento "*UserLogged*", o qual envia por parâmetro os dados do usuário e de seu respectivo atendimento.

Figura 16 - *Job "SearchPatientsInAttendance"*

```php
/**
 * Execute the job.
 *
 * @return void
 */
public function handle()
{
    $atendimentos = DB::connection('oracle')
        ->table('upa_atendimentos_tcc_v')
        ->where('dt_alta', '=', null)
        ->where('dt_entrada', '>=', DB::raw('sysdate-10'))
        ->select('*')
        ->get();
    foreach ($atendimentos as $atendimento) {
        $user = User::where('cd_pessoa_fisica', $atendimento->cd_pessoa_fisica)->first();
        if ($user) {
            AttendanceController::update($atendimento, $user->id);

            event(new UserLogged($user, $atendimento));
        }
    }
}
```

Fonte: Autoria Própria (2019)

O *job* "*SearchPublicPatientsInfo*", apresentado na Figura 17, é responsável por buscar dados públicos de atendimento no sistema Tasy, como a quantidade de pacientes em atendimento e a quantidade de pacientes aguardando no momento de sua execução. Após a obtenção desses dados, o evento "*PatientsPublicInformation*" é disparado, passando como parâmetro os dados obtidos na consulta.

Figura 17 - *Job "SearchPublicPatientsInfo"*

```php
/**
 * Execute the job.
 *
 * @return void
 */
public function handle()
{
    $info_clinicas = DB::connection('oracle')
        ->table('upa_gestao_fluxo_atendimento_v')
        ->select('*')
        ->get();

    event(new PatientsPublicInformation($info_clinicas));
}
```

Fonte: Autoria Própria (2019)

4.2.2.3Events

Foram implementados dois eventos no sistema, localizados no diretório “app/Events”, os quais foram previamente registrados dentro do arquivo “app/Providers/EventServiceProvider.php” e geradas as classes através do seguinte comando Artisan:

```
php artisan event:generate
```

O evento “*UserLogged*” recebe parâmetros no construtor da classe que identificam um determinado paciente por meio do “código de pessoa física”, sendo este um atributo único de cada paciente que consta tanto na base de dados do Tasy quanto na base de dados da aplicação desenvolvida. Em seguida foi criado um "canal privado" denominado “*timeline*” acompanhado do “código de pessoa física”, por meio do qual são enviadas, via *websockets*, as informações privadas de atendimento referentes ao paciente, conforme apresentado na Figura 18.

Figura 18 - *Event "UserLogged"*

```php
public function __construct(User $user, $atendimento_paciente)
{
    $this->user = $user;
    $this->atendimento_paciente = $atendimento_paciente;
}

/**
 * Get the channels the event should broadcast on.
 *
 * @return \Illuminate\Broadcasting\Channel|array
 */
public function broadcastOn()
{
    return new PrivateChannel('timeline.' . $this->user->cd_pessoa_fisica);
}
```

Fonte: Autoria Própria (2019)

O evento "*PatientsPublicInformation*" recebe informações gerais de atendimento como parâmetro no construtor da classe. Em seguida é criado um canal público denominado "*patients-information*", por meio do qual essas informações são enviadas via *websockets* para todos que estiverem conectados a este canal, como está apresentado na Figura 19.

Figura 19 - *Event "PatientsPublicInformation"*

```php
public function __construct($info_clinicas)
{
    $this->info_clinicas = $info_clinicas;
}

/**
 * Get the channels the event should broadcast on.
 *
 * @return \Illuminate\Broadcasting\Channel|array
 */
public function broadcastOn()
{
    return new Channel('patients-information');
}
```

Fonte: Autoria Própria (2019)

4.2.2.4Laravel Echo

Na Figura 20, é apresentado um trecho de código JavaScript utilizando o Laravel Echo, o qual está implementado em um componente do Vue.js. Onde o cliente é inscrito no canal privado identificado como "*timeline*" seguido do código de pessoa física referente a ele. Então o método fica aguardando (*"listening"*) uma resposta do evento "*UserLogged*" contendo seus dados de atendimento, recebidos via *websocket*.

Figura 20 - Cliente escutando em um canal privado

```
Echo.private('timeline.' + this.user)
  .listen('UserLogged', (response) => {
    this.atendimento = response.atendimento
    this.classificacao = response.classificacao
    this.linha_tempo = response.linha_tempo
    this.prescricoes = response.prescricoes
    this.sinais_vitais = response.sinais_vitais
    this.ultimo_update = moment(response.ultimo_update).format('LTS')
  })
```

Fonte: Autoria Própria (2019)

No trecho de código apresentado na Figura 21, o cliente é inscrito no canal público identificado como "*patients-information*", então o método fica aguardando (*"listening"*) uma resposta do evento "*PatientsPublicInformation*" contendo dados de informações gerais, os quais serão recebidos por meio do *websocket*.

Figura 21 - Cliente escutando em um canal público

```
Echo.channel('patients-information')
  .listen('PatientsPublicInformation', (response) => {
    this.infoClinicas = response.info_clinicas
  })
```

Fonte: Autoria Própria (2019)

4.2.3 Requisição de Dados da API

No trecho de código abaixo está representada a implementação de uma rota de API, a qual está localizada no arquivo "routes/api.php". A mesma é protegida pelo

middleware “auth:api” que utiliza o pacote Laravel Passport[32], o qual impede que usuários não autenticados na aplicação consigam acessá-la. Essa rota é do tipo GET, e para consumi-la é necessário enviar o número de atendimento como parâmetro. Sua tarefa é invocar o método “*getAttendanceInfo*” do “*AttendanceController*” que retorna os dados de histórico referente ao paciente.

Figura 22 - Implementação de uma rota tipo "GET"

```
Route::namespace('Api')->group(function () {
    Route::middleware('auth:api')->get('/attendance/{nr_atendimento}',
    function (Request $request, $nr_atendimento) {
        return AttendanceController::getAttendanceInfo($request, $nr_atendimento);
    });
});
```

Fonte: Autoria Própria (2019)

[32] https://laravel.com/docs/5.8/passport

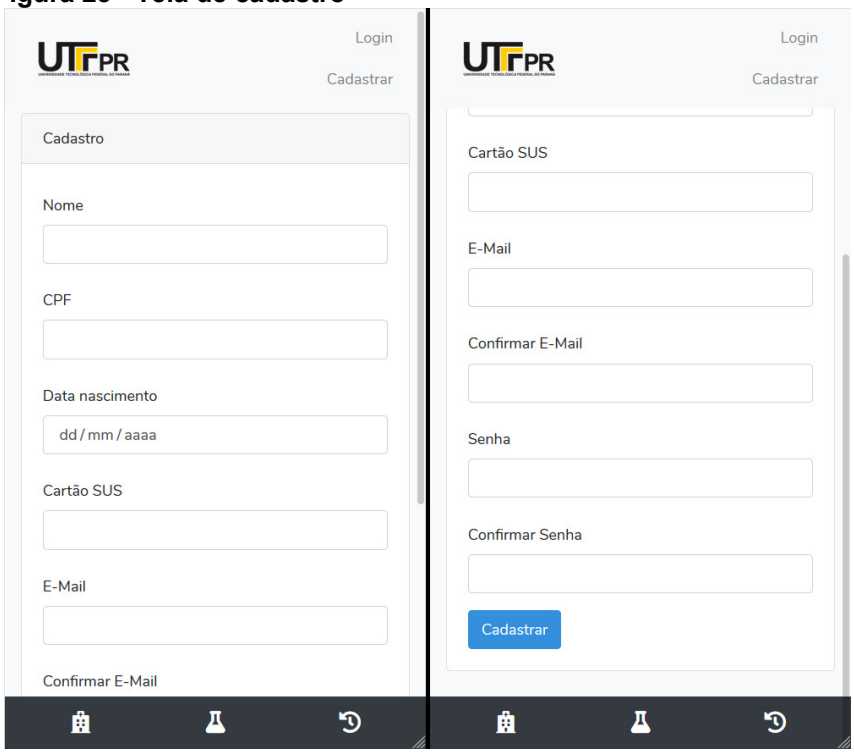
5 RESULTADOS

Após a implementação das tecnologias descritas no capítulo anterior, obteve-se como resultado um sistema totalmente funcional, o qual apresenta suas características principais nos itens a seguir.

5.1 CADASTRO

Para que o usuário possa acessar o sistema, primeiramente deve ser efetuado um cadastro, informando os dados solicitados na tela como mostra a Figura 23. Após preencher os campos e enviar o formulário, é realizada uma confirmação no servidor, o qual verifica se os dados digitados no campo “CPF” e no campo “Data nascimento” correspondem aos respectivos dados presentes no sistema Tasy. Após o cadastro, o paciente é logado automaticamente no sistema e redirecionado para a tela principal (linha do tempo).

Figura 23 - Tela de cadastro

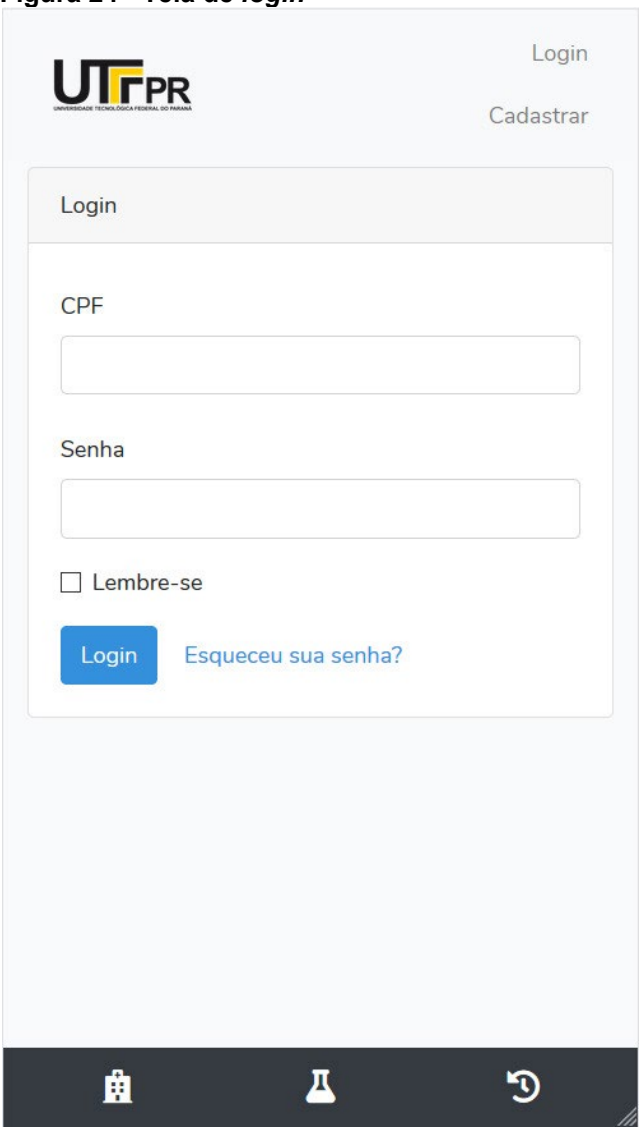


Fonte: Autoria Própria (2019)

5.2 LOGIN

Para que o paciente possa acessar a aplicação, obrigatoriamente deve ser efetuado o *login*, informando seu CPF e sua senha previamente cadastrados como mostra a Figura 24. Após a submissão desse formulário o servidor valida as informações e o paciente é redirecionado para a tela principal do sistema (linha do tempo).

Figura 24 - Tela de *login*

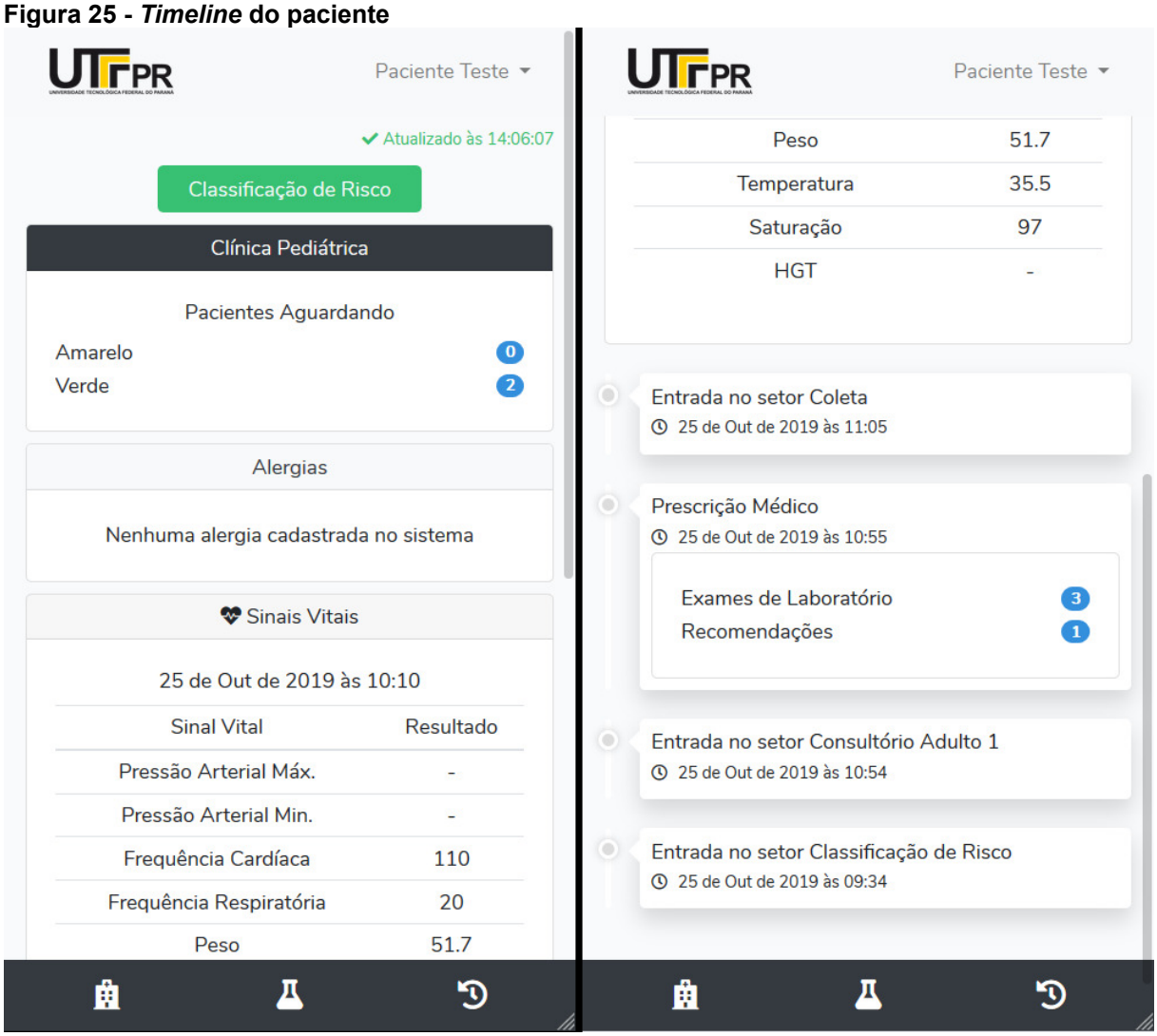


Fonte: Autoria Própria (2019)

5.3 LINHA DO TEMPO

A tela principal do sistema é apresentada na Figura 25. A mesma só é exibida desta maneira quando o paciente estiver em atendimento, caso contrário será exibida uma mensagem informando que os dados de atendimento estão indisponíveis conforme apresentado na Figura 26.

Figura 25 - *Timeline* do paciente

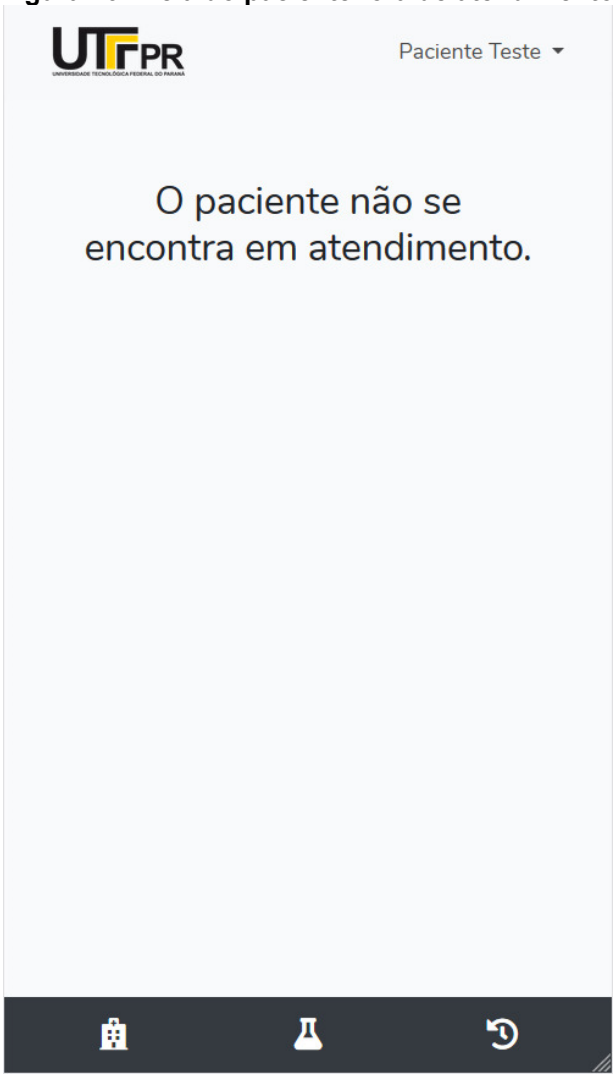


Fonte: Autoria Própria (2019)

Logo abaixo do cabeçalho é apresentado o horário da última atualização seguido da classificação de risco, a qual é definida pela cor apresentada. O *card* (área que delimita um determinado conteúdo de maneira flexível na interface) abaixo apresenta a qual clínica o paciente pertence, no caso da imagem “Clínica Pediátrica”, onde são apresentadas a quantidade de pacientes “aguardando” e a

quantidade de pacientes “em atendimento”, agrupados de acordo com a classificação de risco. Abaixo é apresentado o *card* “Alergias” seguido do *card* “Sinais Vitais”, onde tem-se informações coletadas no processo de triagem do paciente. Por último é exibida a linha do tempo, onde são adicionados os eventos referente ao atendimento atual do paciente, podendo em alguns casos ser expandido para exibição de detalhes, como no caso de “Prescrição Médico”.

Figura 26 - Tela do paciente fora de atendimento



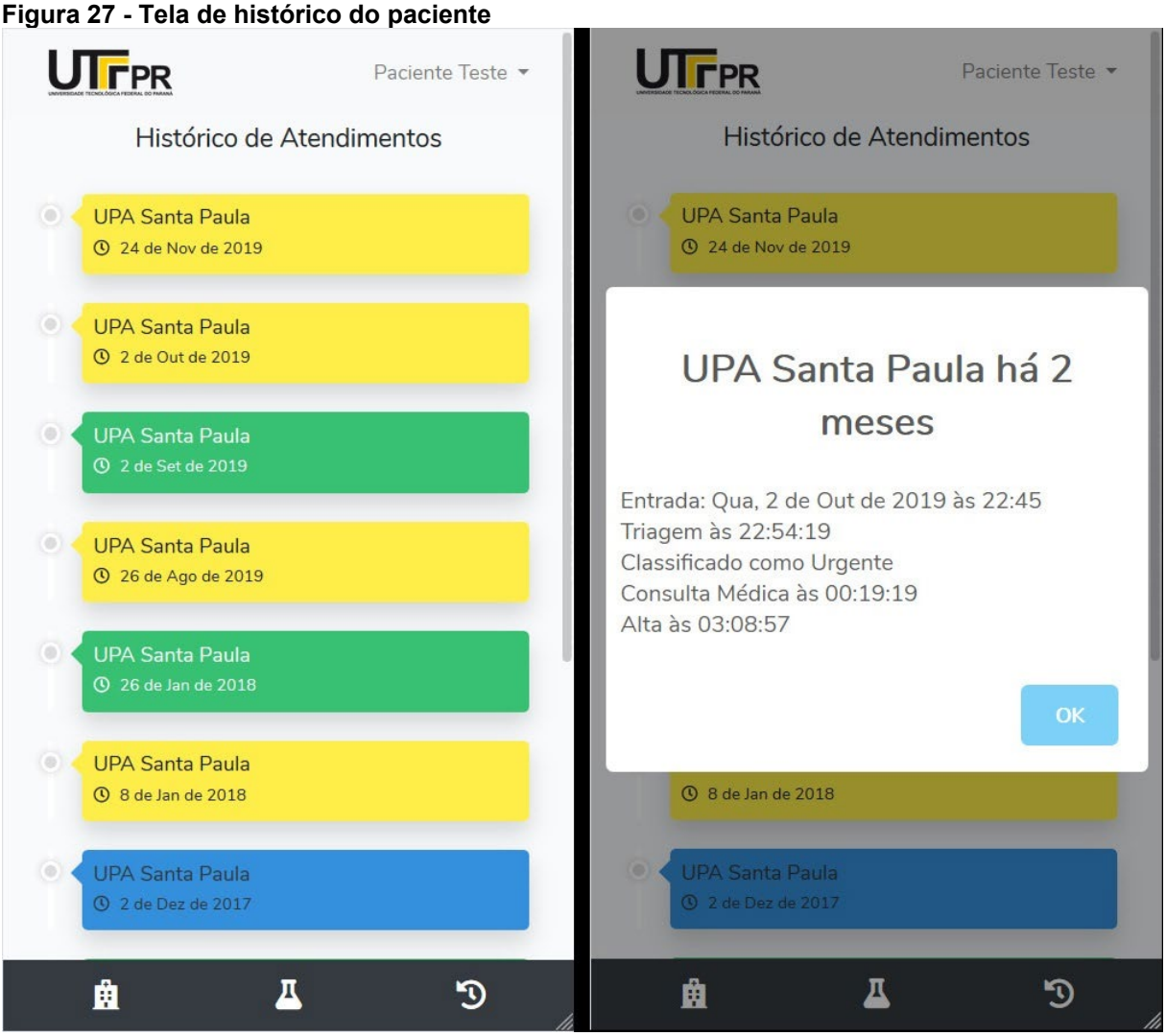
Fonte: Autoria Própria (2019)

A tela acima é apresentada quando o paciente não se encontra em atendimento, não estando disponível a visualização da linha do tempo. Porém é possível o acesso ao histórico de atendimentos e ao *status* de exames laboratoriais.

5.4 HISTÓRICO DO PACIENTE

Na tela de histórico do paciente, cada *card* representa um atendimento finalizado na UPA Santa Paula, o qual é apresentado com a cor referente a classificação de risco. Quando clicado é exibido um *modal*, contendo os dados principais referentes ao atendimento selecionado como mostra a Figura 27.

Figura 27 - Tela de histórico do paciente

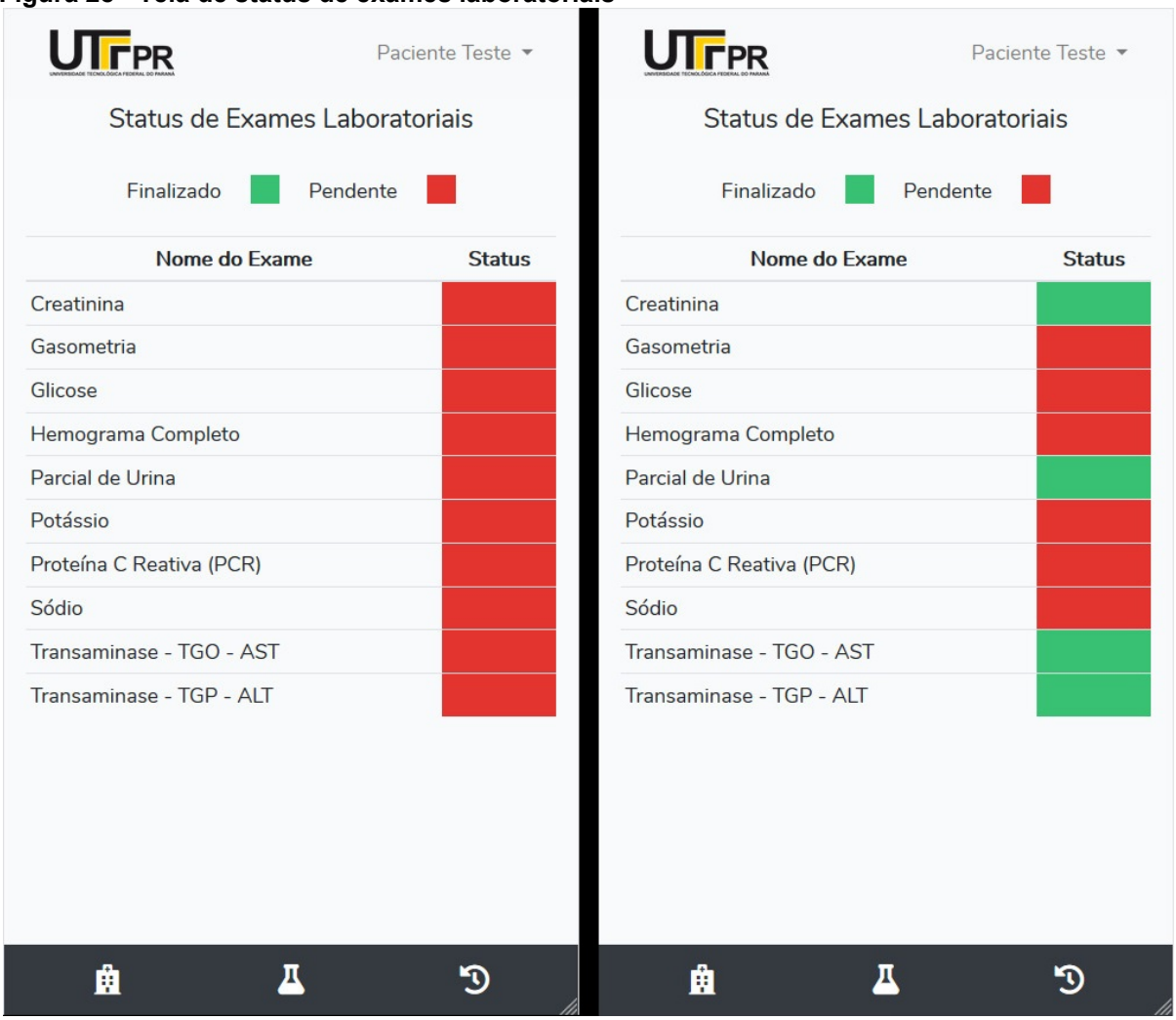


Fonte: Autoria Própria (2019)

5.5 STATUS DE EXAMES

Por meio da tela apresentada na Figura 28, é possível que o paciente visualize o status de seus exames laboratoriais (finalizado ou pendente).

Figura 28 - Tela de status de exames laboratoriais



Fonte: Autoria Própria (2019)

6 CONCLUSÃO

Este trabalho propôs o desenvolvimento de um sistema *web* e *mobile* com o objetivo de disponibilizar informações de cuidado a saúde para pacientes da UPA Santa Paula durante seu atendimento na unidade. O objetivo do sistema foi tornar o paciente mais participativo no seu cuidado, garantindo o sigilo e privacidade das informações disponibilizadas.

Para que o sistema pudesse ser desenvolvido, foi necessário realizar estudos referentes aos procedimentos de cuidado a saúde para compreendê-los da melhor forma possível. Com isso foi possível coletar, armazenar e apresentar informações pertinentes ao cuidado do paciente durante o atendimento, de forma correta e compreensível.

A partir disso, iniciou-se o desenvolvimento do sistema com a utilização de *frameworks* em conjunto a tecnologias, tais como, WebScokets e *frameworks* JavaScript para a reatividade da *interface*, o que contribuiu de forma positiva para sua elaboração. Utilizou-se o *framework* Laravel para o desenvolvimento do servidor e o Vue.js para desenvolvimento do cliente, sendo estas tecnologias que se integram de maneira simples e fluida. Outras tecnologias necessárias para o funcionamento do sistema são o Laravel Websockets com o *Task Scheduling*, para o envio seguro de informações em tempo real.

A utilização da aplicação requer um cadastro prévio para sua utilização, informações as quais são confrontadas com seu respectivo cadastro no Tasy, garantindo a segurança das informações disponibilizadas. O usuário pode visualizar informações durante seu atendimento, tais como sua classificação de risco, quantidade de pacientes aguardando e em atendimento, alergias medicamentosas, sinais vitais aferidos, setores assistenciais ao qual foi atendido, resumo de itens prescritos e os status de exames laboratoriais, além do histórico de atendimentos na unidade.

Apesar deste sistema não ter sido implantado, todo seu desenvolvimento foi realizado em uma máquina virtual com Laravel Homestead, a qual simula um ambiente de produção, bastando apenas configurar o acesso ao banco de dados do Tasy e a tarefa do *task scheduling* no *cron* do servidor, para que seja realizada a implantação em ambiente real de produção. Sua implantação tem a possibilidade de

reduzir eventos adversos ao paciente durante seu atendimento, além de garantir acesso às informações que são de sua propriedade por direito.

6.1 DIFICULDADES E LIMITAÇÕES

Durante a realização desse trabalho, algumas dificuldades foram encontradas, como o processo burocrático para conseguir autorização do NEP da Secretaria Municipal de Saúde, a qual permitiu o uso dos dados de pacientes da UPA Santa Paula para a realização desse projeto. Também foi necessária a obtenção de um documento de autorização da Diretoria Executiva da UPA Santa Paula, o qual permitiu o acesso dos autores na unidade para que pudessem realizar o desenvolvimento deste projeto com dados do sistema Tasy. Além disso, também houveram contratempos ao realizar o projeto baseando-se em tecnologias pouco estudadas até o momento, dentre as quais, algumas ainda possuem uma documentação vaga, como por exemplo, o Laravel Websocket, Laravel Homestead e a biblioteca yajra, necessitando a busca por materiais de apoio.

Suprida as dificuldades para o desenvolvimento, houveram limitações na disponibilização de informações, sendo elas, dados descritivos de prescrição médica e detalhamento de informações referentes ao histórico de atendimentos, as quais não foi concedida a permissão para que pudessem ser apresentadas no sistema.

6.2 TRABALHOS FUTUROS

Com a realização desse trabalho foi possível suprir o objetivo principal proposto de apresentar informações em tempo real aos pacientes em atendimento na UPA Santa Paula, garantindo mais acesso a suas informações e possibilitando um maior entendimento dos processos pelo qual passa dentro da unidade. Porém, pensando em expandir este sistema, pode ser criado um módulo de parametrização que permita a utilização do sistema desenvolvido em outros estabelecimentos de saúde que utilizem um sistema de gestão hospitalar, possibilitando ao paciente

acesso às informações de todos os seus respectivos atendimentos nas unidades onde foi atendido.

Utilizando as mesmas tecnologias e módulos já desenvolvidos, pode ser criada uma interface de painel (*dashboard*) para apresentar dados em tempo real de pacientes em atendimento e resultado de exames, focado nos setores médico e de atendimento. Isso tende a minimizar o tempo de espera para atendimento, uma vez que cada setor poderá receber notificações sobre cada mudança na linha do tempo de um paciente.

Outro fator importante é a implementação do conceito PWA (*Progressive Web Apps*) neste sistema, o qual permite que aplicações *web* se comportem como os aplicativos nativos de dispositivos *mobile*. Este conceito emprega características como a progressividade, segurança, responsividade e a conectividade independente, sendo detectado como aplicativo pelos mecanismos de busca (GOOGLE CODELABS, 2019).

Visto que o sistema desenvolvido apresenta dados sigilosos de pacientes, é necessário garantir a proteção dessas informações, aplicando tecnologias que reforçam a segurança no acesso ao sistema, como a autenticação de dois fatores. Essa tecnologia exige que além do usuário digitar sua senha para acessar o sistema, ele insira um código temporário (*token*), o qual pode ser recebido, em alguns casos, por mensagem no celular ou por *email* (PRASS, 2019).

REFERÊNCIAS

Agência CNJ de Notícias. **CNJ Serviço: Todo paciente tem direito à cópia do prontuário médico**, 2015. Disponível em: <https://www.cnj.jus.br/cnj-servico-todo-paciente-tem-direito-a-copia-do-prontuario-medico/>. Acesso em: 17 set. 2018.

ALBUQUERQUE, Catia. Gestão da Saúde: A importância do Prontuário do Paciente. **Blog da Qualidade**, 2015. Disponível em: <https://blogdaqualidade.com.br/gestao-da-saude-prontuario-do-paciente/>. Acesso em 16 nov. 2018.

ALMEIDA, Gilson. **Ministério da Saúde**. Secretaria de Atenção à Saúde. Portaria nº 279, de 8 de outubro de 2010. Disponível em: <http://bvsms.saude.gov.br/bvs/saudelegis/inc/2010/prt0279_08_10_2010.html>. Acesso em 15 nov. 2018.

ANDRADE, Antonio José F.; et al. Gestão de Projeto com Scrum: Um Estudo de Caso. **ENUCOMP – Encontro Unificado de Computação em Parnaíba**, 2012. Disponível em: <https://www.enucomp.com.br/2012/conteudos/artigos/scrum.pdf>. Acesso em 12 jan. 2019.

ARAÚJO, Wesley Serafim. Vue.js — Construindo um SPA (Single Page Application) com vue-cli, vue-router, vue-resouce e Vuex (Parte 1). **Codeburst**, 2016. Disponível em: <https://codeburst.io/vue-js-construindo-um-spa-single-page-application-com-vue-cli-vue-router-vue-resouce-e-vue-ex-116bf877e158>. Acesso em 22 jul. 2019.

AVELAR, Ariane Ferreira Machado; et al. 10 Passos para a segurança do paciente. **COREN-SP**, 2010. Disponível em: <http://portal.coren-sp.gov.br/sites/default/files/10_passos_seguranca_paciente_0.pdf>. Acesso em 28 ago. 2018.

BARROS, Ricardo. **Ministério da Saúde**. Gabinete do Ministro. Portaria nº 10 de 3 de janeiro de 2017. Disponível em: <http://bvsms.saude.gov.br/bvs/saudelegis/gm/2017/prt0010_03_01_2017.html>. Acesso em 28 ago. 2018.

BATALDEN, Maren; et al. Coproduction of healthcare service. **BMJ Quality Safety**, 2016. Disponível em: <https://qualitysafety.bmj.com/content/25/7/509>. Acesso em: 29 ago. 2018.

BEYONDCODE. **Laravel WebSockets**, 2019. Disponível em: <https://docs.beyondco.de/laravel-websockets/>. Acesso em 22 jul. 2019.

CARRASCO, Ileana. Tasy conquista a certificação SBIS-CFM. **Philips**, ago. 2017. Disponível em: <https://www.philips.com.br/a-w/about/news/archive/standard/news/press/2017/20170826-tasy-conquista-a-certificacao-sbis-cfm.html>. Acesso em 12 jan. 2019.

CONSELHO FEDERAL DE MEDICINA (CFM); SOCIEDADE BRASILEIRA DE INFORMÁTICA EM SAÚDE (SBIS). **Prontuário Eletrônico:** A Certificação de Sistemas de Registro Eletrônico de Saúde. Segurança e Confidencialidade para a Informação do Paciente, fev. 2012. Disponível em: <http://portal.cfm.org.br/crmdigital/Cartilha_SBIS_CFM_Prontuario_Eletronico_fev_2012.pdf>. Acesso em 22 nov. 2018.

CONSELHO FEDERAL DE MEDICINA (CFM). Resolução CFM nº 1.638/2002. **Diário Oficial da União**, Brasília, 10 jul. 2002. Seção I, p. 184-185. Disponível em: <http://www.portalmedico.org.br/resolucoes/cfm/2002/1638_2002.htm>. Acesso em 16 nov. 2018.

DATE, C. J. **Introdução a sistemas de bancos de dados**. Rio de Janeiro: Elsevier Brasil, 865 p., 2004.

DRIESSEN, Vicent. **A successful Git branching model**, 2010. Disponível em: <https://nvie.com/posts/a-successful-git-branching-model/>. Acesso em 10 set. 2018.

FARINA, Aguiar. Prontuário Médico. **Conselho Federal de Medicina**, 1999. Disponível em: <http://portal.cfm.org.br/index.php?option=com_content&id=20462:prontuario-medico>. Acesso em 15 nov. 2018.

FERREIRA, Carlos. **Aprenda como Trabalhar com Eventos no Laravel**. EspecializaTI, 2018. Disponível em: <https://blog.especializati.com.br/aprenda-como-trabalhar-com-eventos-no-laravel/>. Acesso em 22 nov. 2019.

FERREIRA, D.; et al. Scrum: um modelo ágil para gestão de projetos de software. **Faculdade de Engenharia Universidade do Porto**, 2012.

GOMES, Luis Fernando. **Fluxo de versionamento de software com git flow**, 2016. Disponível em: <https://blog.ateliedocodigo.com.br/fluxo-de-versionamento-de-software-com-git-flow-b9f5195c679e>. Acesso em 10 set. 2018.

GONÇALVES, João Paulo Pereira; et al. Prontuário Eletrônico: uma ferramenta que pode contribuir para a integração das Redes de Atenção à Saúde. **Saúde em Debate**, Rio de Janeiro (RJ), v. 37, n. 96, p. 43-50, jan./mar. 2013.

GOOGLE CODELABS. **Your First Progressive Web App**, 2019. Disponível em: <https://codelabs.developers.google.com/codelabs/your-first-pwapp/#0>. Acesso em 20 out. 2019.

GUIMARÃES, José Ricardo; KLÜCK, Mariza Machado. Prontuário de pacientes finalidades preenchimento e questões éticas e legais. **Medicinanet**, 2014. Disponível em: <http://www.medicinanet.com.br/conteudos/revisoes/5795/prontuario_de_pacientes_finalidades_preenchimento_e_questoes_eticas_e_legais.htm>. Acesso em 16 nov. 2018.

LARANJO, Liliana; et al. Acesso dos Pacientes aos seus Processos Clínicos. **Acta Médica Portuguesa**, v. 26, n. 3, p. 265-270, mai-jun, 2013.

LARAVEL. **Documentation v. 5.8**, 2019. Disponível em: <https://laravel.com/docs/5.8>. Acesso em 10 jan. 2019.

LEGATTI, Eduardo. A evolução dos bancos de dados Oracle. **Database Blog**, abr. 2007. Disponível em: <https://eduardolegatti.blogspot.com/2007/04/evoluo-dos-bancos-de-dados-oracle.html>. Acesso em 15 fev. 2019.

MACHADO, Maria Helena; DUARTE, Else; RUFFINO, Márcia Caron. Análise da Importância Clínica da Rotina da Verificação da Freqüência Respiratória em Pacientes Hospitalizados. **Rev. Bras. Enferm.**, Brasília, v. 28, n. 2, p. 23-27, jun. 1975.

MALDONADO, Leonardo. Entendendo o DOM (Document Object Model). **Tableless**, 2018. Disponível em: <https://tableless.com.br/entendendo-o-dom-document-object-model/>. Acesso em 22 jul. 2019.

MARIADB Foundation. **About MariaDB**, 2019. Disponível em: <https://mariadb.org/about/>. Acesso em 15 fev. 2019.

MARQUES, Keise de Leone. **Back-end vs Front-end vs Full-Stack:** qual é a melhor escolha?. Becode, 2017. Disponível em: <https://becode.com.br/back-end-front-end-full-stack>. Acesso em 20 fev. 2019.

MINISTÉRIO DA SAÚDE. Acolhimento e Classificação de Risco nos Serviços de Urgência. **Secretaria de Atenção à Saúde**, 2009. Disponível em: <http://bvsms.saude.gov.br/bvs/publicacoes/acolhimento_classificaao_risco_servico_urgencia.pdf>. Acesso em 26 nov. 2018.

MINISTÉRIO DA SAÚDE. Unidade de Pronto Atendimento (UPA 24h): o que é, quando usar, diretrizes e competências. **Secretaria de Atenção à Saúde**, 2017. Disponível em: <http://portalms.saude.gov.br/saude-de-a-z/unidade-de-pronto-atendimento-upa-24h>. Acesso em 28 ago. 2018.

ORACLE. **Oracle Database Documentation**, 2019. Disponível em: <https://docs.oracle.com/en/database/oracle/oracle-database/index.html>. Acesso em: 15 fev. 2019.

OTÁVIO, João. Introdução ao Scrum. Guia Completo de Scrum. **Devmedia**, 2015. Disponível em: <https://www.devmedia.com.br/introducao-ao-scrum/33724>. Acesso em 12 jan. 2019.

PANZOLINI, Breno. **Gerenciando seus branches com o Git Flow:** Explicação dos conceitos básicos e fundamentais do Git Flow. Tableless, 2018. Disponível em: <https://tableless.com.br/git-flow-introducao/>. Acesso em 10 set. 2018.

PATIENT SAFETY MOVEMENT (PSM); Transição do cuidado: ferramentas para evitar erros na comunicação. **IBSP - Instituto Brasileiro para Segurança do Paciente**, 2018. Disponível em: <https://www.segurancadopaciente.com.br/qualidade-assist/transicao-do-cuidado-ferramentas-comunicacao/>. Acesso em: 26 set. 2018.

PAULA, Gilles B. de. Tudo sobre Metodologia Scrum: o que é e como essa ferramenta pode te ajudar a poupar tempo e gerir melhor seus projetos. **Treasy Planejamento e Controladoria**, fev. 2016. Disponível em: <https://www.treasy.com.br/blog/scrum/>. Acesso em 15 set. 2018.

PEDROSO, Carmenci. ERP Sistema integrado de gestao empresarial. **Centro Universitário Leonardo da Vinci – UNIASSELVI**, 2011. Disponível em: <http://www.administradores.com.br/artigos/tecnologia/erp-sistema-integrado-de-gestao-empresarial/56841/>. Acesso em 28 ago. 2018.

PEREIRA, Samáris Ramiro; et al. Sistemas de Informação para Gestão Hospitalar. **Journal of Health Informatics**, v. 4, n. 4, p. 170-175, out./dez. 2012.

PHILIPS. **Tasy**, 2019. Disponível em: <https://www.philips.com.br/healthcare/resources/landing/solucao-tasy>. Acesso em 12 jan. 2019.

POCIOT, Marcel; HERTEN, Freek Van der. Introducing laravel-websockets, an easy to use WebSocket server implemented in PHP. **Freek.dev**, 2018. Disponível em: <https://freek.dev/1228-introducing-laravel-websockets-an-easy-to-use-websocket-server-implemented-in-php>. Acesso em 22 jul. 2019.

POLO, Rafael Gomes de Oliveira; ARÊAS, SABRINA. **Desenvolvimento ágil de sistemas para a web 2.0 com Rails**. 2009. 52 f. Monografia (Trabalho de Conclusão de Curso) – Centro Universitário Metodista Bennett, Rio de Janeiro, nov. 2009.

PRASS, Ronaldo. Saiba o que é autenticação em dois fatores e por que ela é importante. **G1 Economia – Blog do Ronaldo Prass**. 14 ago. 2019. Disponível em: <https://g1.globo.com/economia/tecnologia/blog/ronaldo-prass/post/2019/08/14/saiba-o-que-e-autenticacao-em-dois-fatores-e-por-que-ela-e-importante.ghtml>. Acesso em 27 nov. 2019.

SCHWABER, Ken; SUTHERLAND, Jeff. Um guia definitivo para o Scrum: As regras do jogo. **Guia do Scrum**, 2013. Disponível em: <https://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-Portuguese-BR.pdf>. Acesso em 15 set. 2018.

SERVIN, S. C. N.; et al. Protocolo de Acolhimento com Classificação de Risco. **SUS – Sistema Único de Saúde**, 2002. Disponível em: <http://bvsms.saude.gov.br/bvs/publicacoes/protocolo_acolhimento_classificacao_risco.pdf>. Acesso em 18 nov. 2018.

SILVA, Luciano A. de F. Blade Engine: Utilizando templates no Laravel. **Guia de Referência Laravel**, 2016. Disponível em: <https://www.devmedia.com.br/blade-engine-utilizando-templates-no-laravel/36749>. Acesso em 10 jan. 2019.

SILVA, Renan Rosso da; CASAGRANDE, Rogério Antônio. **Redes de Computadores - Gerenciamento de servidores Linux, a partir de dispositivos celulares com suporte a J2ME, utilizando protocolo de comunicação SSH**. Anais V SULCOMP, v. 5, p. 3-4, 2010.

SILVA, Thaynara de Oliveira; et al. O envolvimento do paciente na segurança do cuidado: revisão integrativa. **Revista Eletrônica de Enfermagem**, v. 18, jun. 2016.

UPA SANTA PAULA. IT: Retirada Externa de Prontuários. Ponta Grossa, 2016. 2 p.

UPA SANTA PAULA. Protocolo de Classificação de Risco Adulto. Ponta Grossa, 2017. 14 p.

VUE.JS. **Guia 2.x**, 2019. Disponível em: <https://br.vuejs.org/v2/guide/index.html>. Acesso em 22 jul. 2019.

ZAMBON, Lucas Santos. Qual a relevância da comunicação efetiva no ambiente hospitalar?. **IBSP – Instituto Brasileiro para Segurança do Paciente**, 2017. Disponível em: <https://www.segurancadopaciente.com.br/seguranca-e-gestao/qual-relevancia-da-comunicacao-efetiva-no-ambiente-hospitalar/>. Acesso em 28 nov. 2018.

ZAMBON, Lucas Santos. Read back: uma estratégia de comunicação para ser usada também com pacientes. **IBSP – Instituto Brasileiro para Segurança do Paciente**, 2018a. Disponível em: <https://www.segurancadopaciente.com.br/qualidade-assist/read-back-uma-estrategia-de-comunicacao-para-ser-usada-tambem-com-pacientes/>. Acesso em 28 nov. 2018.

ZAMBON, Lucas Santos. Transição do cuidado: ferramentas para evitar erros na comunicação. **IBSP – Instituto Brasileiro para Segurança do Paciente**, 2018b. Disponível em: <https://www.segurancadopaciente.com.br/qualidade-assist/transicao-do-cuidado-ferramentas-comunicacao/>. Acesso em 28 nov. 2018.

ANEXO A - Autorização da Prefeitura Municipal de Ponta Grossa

Núcleo de Educação Permanente
SMS-PMPG

PREFEITURA MUNICIPAL DE PONTA GROSSA
SECRETARIA MUNICIPAL DE SAÚDE DE PONTA GROSSA
NÚCLEO DE EDUCAÇÃO PERMANENTE – NEP

CARTA DE AUTORIZAÇÃO

CARTA DE AUTORIZAÇÃO

Eu, CARLOS EDUARDO CORADASSI, coordenador do NEP-SMS, autorizo a realização do projeto: Sistema de mapeamento de procedimentos de saúde para pacientes de uma unidade de pronto atendimento de Ponta Grossa, realizada UTF/PR. que será desenvolvida pela pesquisadora Lucas Istak, Renato Willian Machado, sob orientação do Prof. Dr.Diego Roberto Antunes

Ressalto que qualquer publicação oriunda desta pesquisa deverá conter logo da instituição e respectiva citação.

Ponta Grossa, **29/08/2019**

Carlos Eduardo Coradassi

Coordenador NEP-SMS-PMPG

Ponta Grossa, 29 agosto de 2019.

---

sei! assinatura eletrônica

Documento assinado eletronicamente por **CARLOS EDUARDO CORADASSI**, **Coordenador**, em 29/08/2019, às 17:45, horário oficial de brasília, conforme o Decreto Municipal nº 14.369 de 03/05/2018.

A autenticidade do documento pode ser conferida no site http://sei.pontagrossa.pr.gov.br/validar informando o código verificador **0224778** e o código CRC **D98E9804**.

Link de acesso externo: SEI00396/2019

RUA ALFREDO GUIMARÃES VILELA 383- JARDIM CARVALHO – PONTA GROSSA- PR CEP 84035-680
TELEFONE (42) 3226-8566

ANEXO B - Autorização da UPA Santa Paula

Ministério da Educação

Universidade Tecnológica Federal do Paraná

Câmpus Ponta Grossa

Diretoria de Graduação e Educação Profissional

Departamento Acadêmico de Informática

Tecnologia em Análise e Desenvolvimento de Sistemas

SOLICITAÇÃO

AUTORIZAÇÃO PARA COLETA DE DADOS DA PESQUISA INTITULADA "SISTEMA DE MAPEAMENTO DE PROCEDIMENTOS DE SAÚDE PARA PACIENTES DE UMA UNIDADE DE PRONTO ATENDIMENTO DE PONTA GROSSA"

por

LUCAS ISTAK

RENATO WILLIAN MACHADO

Prezado Diretor Juliano Steurer,

Conforme autorização concebida pela FMS/NEP, processo nº 1620499/2019, para realização da pesquisa para o Trabalho de Conclusão de Curso intitulado "SISTEMA DE MAPEAMENTO DE PROCEDIMENTOS DE SAÚDE PARA PACIENTES DE UMA UNIDADE DE PRONTO ATENDIMENTO DE PONTA GROSSA", solicitamos a autorização para realização da etapa "coleta de dados" na UPA Santa Paula, no período de 24/09/2019 à 15/12/2019, previsto no horário entre 18h e 22h nos dias úteis e finais de semana no período diurno.

Este documento discrimina a elaboração de trabalho científico e não caracteriza vínculo empregatício durante sua execução. A lista de presença com data e horários deverão ser anexados a este documento após seu término.

Juliano Steurer
Diretor Executivo

Rafaela Chiczta
Núcleo de Educação Permanente

Lucas Istak
Membro titular

Renato Willian Machado
Membro titular